**Moderné vzdelávanie pre vedomostnú spoločnosť/
Projekt je spolufinancovaný zo zdrojov EÚ**

POČÍTAČOVÉ SYSTÉMY V RIADENÍ

Fakulta elektrotechniky a informatiky

Ján Jadlovský, Peter Papcun

Európska únia
Európsky sociálny fond

VÝSKUMNÁ AGENTÚRA

Operačný program VZDELÁVANIE

Táto publikácia vznikla za finančnej podpory z **Európskeho sociálneho fondu** v rámci Operačného programu **VZDELÁVANIE.**

Prioritná os 1 Reforma vzdelávania a odbornej prípravy

Opatrenie 1.2 Vysoké školy a výskum a vývoj ako motory rozvoja vedomostnej spoločnosti.

Názov projektu: **Balík doplnkov pre ďalšiu reformu vzdelávania na TUKE**

ITMS 26110230093

NÁZOV: Počítačové systémy v riadení
AUTORI: doc. Ing. Ján Jadlovský, CSc., Ing. Peter Papcun
VYDAVATEĽ: Technická univerzita v Košiciach
ROK: 2015
ROZSAH: 415 strán
NÁKLAD: 70 ks
VYDANIE: prvé
ISBN: 978-80-553-2102-8

Anotácia

Učebnica je prioritne určená študentom bakalárskeho štúdia študijného odboru Kybernetika a informačno-riadiace systémy. Jej cieľom je pomôcť čitateľovi zorientovať sa v oblasti základných vlastností počítačov tak, aby na úrovni PC vedel pripojiť periférie, resp. iné technické zariadenia k počítačom PC a to po hardvérovej, programovej a sieťovej stránke, teda za pomoci počítača PC „vedel vytvoriť“ merací, riadiaci prípadne komunikačný systém. Študentom inžinierskeho štúdia môže poskytnúť určitý prehľad a metodiku pre realizáciu úloh na počítačoch PC vyžadujúcich pripojenie rôznych periférii a technických zariadení, ich ovládanie, komunikáciu s obsluhou a logické a fyzické pripojenie do zložitejších Informačných systémov.

Predkladaná učebnica vychádza zo všeobecných znalosti o počítačoch, vytvára ich stručný prehľad, vrátane základných historických faktov v súvislosti s kompatibilitou jednotlivých na seba nadväzujúcich generácii počítačov. Zameriava sa na osobné počítače typu IBM PC a ich aplikačné využitie v oblasti merania, regulácie, riadenia, komunikácie medzi počítačmi na rôznych úrovniach a vytváranie komunikačného rozhrania medzi človekom a počítačom v textovom a grafickom režime. Predpokladá sa, že takéto aplikácie budú vytvárane na najnižšej úrovni s využitím štandardných a rozširujúcich hardwarových a programových modulov. Využívané hardwarové moduly sú vyhotovené buď vo forme kariet, ktoré sa zasúvajú priamo do konektorov zberníc počítačov PC, alebo vo forme nezávislých modulov, ktoré sú k počítačom PC pripájané cez štandardné rozhrania (USB, RS-232, CENTRONIX, atď.), sieťové rozhrania (napr. ETHERNET TCP/IP), alebo špeciálne rozhrania, ktoré sú vytvárané pomocou komunikačných kariet, zasúvajúcich sa do konektorov zbernice. Predpokladá sa, že programové moduly budú vytvárané pod operačným systémom v objektovo-orientovanom jazyku s využitím príslušných knižníc. Vo väzbe na hardvér sa predpokladá využitie prerušovacieho systému, systému priameho prístupu do pamäte, obrazového podsystému PC a podobne.

Vydanie tejto vysokoškolskej učebnice bolo finančne podporené projektom:
Balík doplnkov pre ďalšiu reformu vzdelávania na TUKE, ITMS 26110230093

ISBN 978-80-553-2102-8

Obsah

Úvod

Predkladaný učebný text vychádza zo všeobecných znalosti o počítačoch, vytvára ich stručný prehľad vrátane základných historických faktov v súvislosti s kompatibilitou jednotlivých na seba nadväzujúcich generácii počítačov. Zameriava sa na osobné počítače typu IBM PC a ich aplikačné využitie v oblasti merania, regulácie, riadenia, komunikácie medzi počítačmi na rôznych úrovniach a vytváranie komunikačného rozhrania medzi človekom a počítačom v textovom a grafickom režime. Predpokladá sa, že takéto aplikácie budú vytvárané na najnižšej úrovni s využitím štandardných a rozširujúcich hardwarových a programových modulov. Využívané hardwarové moduly sú vyhotovené buď vo forme kariet, ktoré sa zasúvajú priamo do konektorov zberníc počítačov PC, alebo vo forme nezávislých modulov, ktoré sú k počítačom PC pripájané cez štandardné rozhrania (USB, RS-232, CENTRONIX, atď.), sieťové rozhrania (napr. ETHERNET TCP/IP), alebo špeciálne rozhrania, ktoré sú vytvárané pomocou komunikačných kariet, zasúvajúcich sa do konektorov zbernice. Predpokladá sa, že programové moduly budú vytvárané pod operačným systémom v objektovo-orientovanom jazyku s využitím príslušných knižníc. Vo väzbe na hardvér sa predpokladá využitie prerušovacieho systému, systému priameho prístupu do pamäte, obrazového podsystému PC a pod.

Po vecnej stránke je rozdelená učebnica do 8 kapitol. Prvá kapitola je zameraná na architektúru počítačov. V úvode kapitoly sú popísané základné architektúry počítačov, historický vývoj počítačov a popis základných častí počítača PC.

V druhej kapitole sú popísané základné podsystémy počítača PC s dôrazom na tie častí (hardvérové, softvérové a logické), ktoré sa nejakým spôsobom podieľajú na komunikácii počítača s okolím.

Tretia kapitola je zameraná na zbernice počítačov PC. Sú uvedené princípy zberníc, popis jednotlivých signálov dostupných na konektoroch príslušnej zbernice a ich aplikačné použitie.

Štvrtá, piata a šiesta kapitola sa zameriava na počítačové rozhranie realizované pomocou počítačových sietí. Uvedené sú princípy počítačových sietí, základné topológie, medzinárodné normy a odporúčania, základné typy počítačových sietí a spôsoby riadenia jednotlivých typov sietí. Bol zvolený aplikačný prístup, to znamená, že pri každom type siete je popísaný princíp, metódy riadenia siete, komunikačné protokoly a príklady technickej realizácie konkrétneho typu siete ako prostriedku komunikácie medzi procesmi. V závere piatej a šiestej kapitoly je popísaná technológia Internetu a jej aplikácia v procese riadenia.

Siedma kapitola je zameraná na popis a aplikáciu najbežnejších rozhraní PC. Jedná sa o štandardné sériové, paralelné a technologické rozhrania. Je tu uvedený princíp a príklady aplikácie.

Ôsma kapitola sa zameriava na popis metodiky tvorby programových aplikačných rozhraní pod OS WINDOWS. Sú popísané jednotlivé typy operačných systémov, spôsoby vytvárania aplikačných programových rozhraní s využitím objektov jadra OS.

Učebnica je prioritne určená študentom bakalárskeho štúdia študijného odboru Kybernetika a informačno-riadiace systémy. Jej cieľom je pomôcť čitateľovi zorientovať sa v oblasti základných vlastností počítačov tak, aby na úrovni PC vedel pripojiť periférie, resp. iné technické zariadenia k počítačom PC a to po hardvérovej, programovej a sieťovej stránke, teda za pomoci počítača PC „vedel vytvoriť“ merací, riadiaci prípadne komunikačný systém.

Študentom inžinierskeho štúdia poskytuje určitý prehľad a metodiku pre realizáciu úloh na počítačoch PC vyžadujúcich pripojenie rôznych periférii a technických zariadení, ich ovládanie, komunikáciu s obsluhou a logické a fyzické pripojenie do zložitejších Informačných systémov.

1 Architektúra počítačov

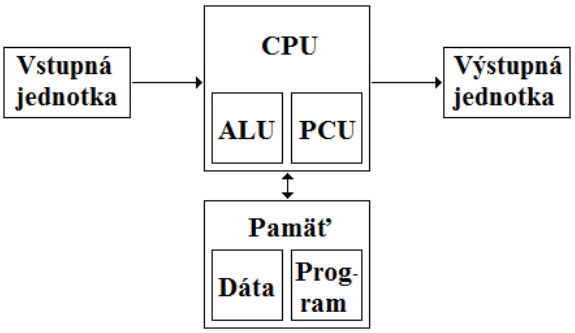
Táto kapitola priblíži architektúru počítačov od teoretickej architektúry až po architektúru dnešných počítačov. Teoretickou architektúrou počítačov sa budú zaoberať podkapitoly Von Neumannová architektúra a Harwardská architektúra. Ďalej sa kapitola bude povrchne zaoberať históriou počítačov, ich jednotlivými generáciami až po súčasnosť. Pri architektúre netreba zabúdať na základné hardvérové súčasti počítača, ktoré sú tiež rozobraté v tejto kapitole. Posledná časť kapitoly popisuje po odrážkach architektúru počítačov s procesorom Intel.

1.1 Von Neumannová architektúra

V 40-tych rokoch 20. storočia John von Neumann predložil základný princíp *počítača riadeného tokom inštrukcií*, ktorý sa stal základom jednej triedy súčasných počítačov. Táto trieda počítačov býva tiež označovaná ako von Neumannovské počítače.

Vyznačuje sa tým, že jednotlivé inštrukcie programu sa vykonávajú postupne za sebou, tak ako sú uložené v pamäti. V súčasnosti existujú aj výpočtové systémy, v ktorých sa inštrukcie nevykonávajú v poradí, v akom sú uložené v pamäti (napr. počítače riadené tokom údajov, neurónové počítače a iné).

Základné črty von Neumannovského počítača (Obr. 1.1):



Obr. 1.1 Základná von Neumannová architektúra počítača

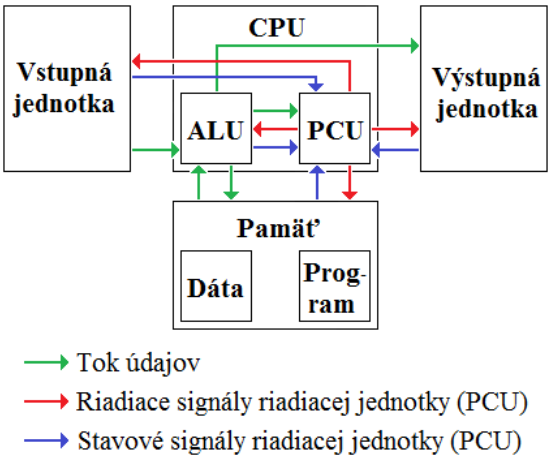
- **Pamäť** je používaná na uloženie inštrukcií (programu) aj údajov (dát).
- **Riadiaca jednotka** (PCU – program control unit) je používaná na výber inštrukcií z pamäte.
- **Aritmeticko-logická jednotka** (ALU – arithmetic-logic unit) je používaná na vykonávanie špecifikovaných operácií nad údajmi.
- **Procesor** (CPU – central processing unit) zastrešuje ALU a PCU.
- **Vstupná jednotka** je používaná na vstup údajov.
- **Výstupná jednotka** je používaná na výstup údajov.

1.1.1 Výpočtový proces

Vykonávanie výpočtu sa nazýva výpočtový proces, alebo procedúra. Aby sa mohol vykonávať výpočtový proces, pre každú operáciu tohto procesu musia byť známe tieto informácie:

- **operandy** s ktorými má byť operácia vykonaná (operandom nemusí byť len číslo, môže to byť ľubovoľný objekt)
- **sémantika** vykonávanej operácie – jej zmysel
- **kde zaznamenať výsledok** (napr. do ktorého stĺpca vo formulári, v akej premennej bude výsledok uložený)
- **ktorú operáciu vykonať ako nasledujúcu** (ak šlo o operáciu rozhodovania, musíme vedieť, ako postupovať pri rôznych výsledkoch).

Súhrn týchto informácií nazývame inštrukcia. Výpočtový proces realizovaný počítačom nazývame program. Bloková schéma von Neumannového počítača je na Obr. 1.2:



Obr. 1.2 Bloková schéma von Neumannového počítača

Činnosť celého počítača riadi riadiaca jednotka (PCU), ktorá odovzdáva povely operačnej pamäti, ALU a vstupno-výstupným zariadeniam a späť od nich dostáva stavové hlásenia. PCU číta z operačnej pamäte inštrukcie, tie dekóduje a prevádza na postupnosť signálov. Dáta číta procesor z operačnej pamäte alebo z vstupného zariadenia. Procesor dáta tiež ukladá do pamäte alebo zapisuje na výstupné zariadenia.

Von Neumannov počítač IAS je významný preto, že až na malé výnimky je jeho schéma platná dodnes.

Riadiaca jednotka (PCU) a **aritmeticko-logická jednotka** (ALU) sú zvyčajne realizované ako jeden funkčný blok, ktorý sa nazýva centrálna procesná jednotka (CPU) alebo skrátene **procesor**.

Vstupno-výstupné jednotky, každá z týchto častí vykonáva určitú funkciu a ich vzájomne riadená spolupráca pomocou procesora a operačnej pamäte tvorí celkový chod číslicového počítača.

V **operačnej pamäti** sa nachádza program a údaje ktoré sa spracúvajú pomocou procesora, označuje sa aj ako pracovná pamäť.

1.1.2 Operačná pamäť

Pamäť je tá časť číslicového počítača, ktorá je schopná zapamätať si informáciu a v prípade potreby ju vydať (pamätí v číslicovom počítači je viacej druhov). Operačná pamäť (hlavná, v dnešných počítačoch označovaná ako RAM - Random Access Memory) je množina rovnakých buniek, z ktorých každá je identifikovateľná poradovým číslom – adresou.

Inštrukcie a údaje uložené v pamäti sú zakódované binárnym (dvojkovým) kódom (bistabilné konštrukčné prvky – bit). Informácia uložená v bunke je slovo (postupnosť znakov, ktorými je zobrazený *operand*, *výsledok operácie*, *inštrukcia*). Počet bitov, ktoré môžeme uložiť do pamäte určujú kapacitu pamäte.

1.1.3 Procesor

ALU je centrálna časť procesora (CPU). Je to tá časť číslicového počítača v ktorej sa vykonávajú základné aritmetické a logické operácie s číslami: sčítanie, odčítanie, násobenie, delenie, logický posun, negácia, komplement, a pod.

Inštrukcia je príkaz pre PCU, ktorý určuje, aká operácia sa má vykonať a s ktorými údajmi. Inštrukcia je príkaz na vykonanie operácie (vykonávajú sa postupne za sebou tak, ako sú uložené v pamäti).
Zloženie inštrukcie:

- kód operácie – druh operácie (binárne zakódované inštrukcie sú označované ako strojové inštrukcie),
- adresná časť – určuje operandy (ich umiestenie v operačnej pamäti, nad ktorými sa má operácia vykonať).

Všetky typy inštrukcií, ktoré môže ČP vykonávať nazývame inštrukčný súbor.

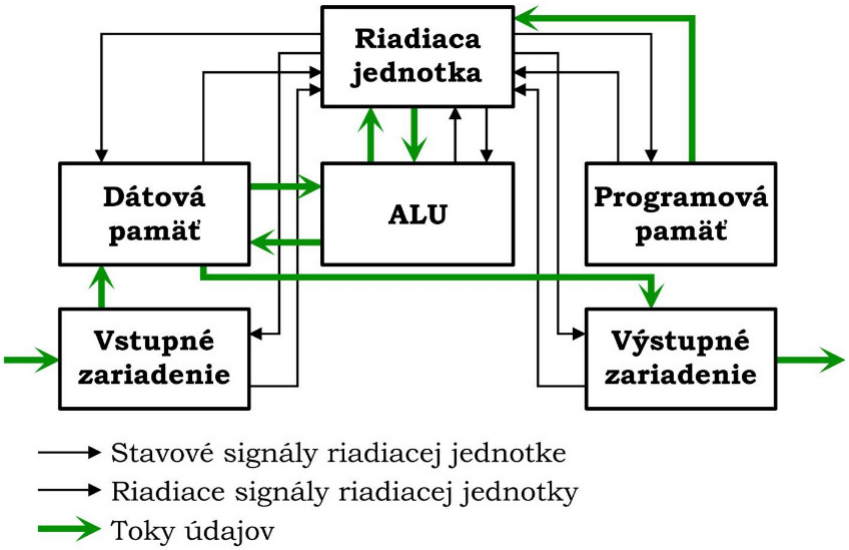
PCU je časť procesora, ktorá riadi vykonávanie operácie a chod celého procesora podľa inštrukcií programu.
PCU pozostáva z týchto časti:

- **register inštrukcií**, ktorý uchováva operačný znak inštrukcie počas jej vykonania,
- **dekodér inštrukcií** podľa operačného kódu v inštrukcii určí o akú operáciu ide,
- **čítač inštrukcie** v každom kroku výpočtu udáva adresu pamäťovej bunky, kde je uložená inštrukcia, ktorá sa má vykonať ako nasledujúca.

Vykonanie ďalšej inštrukcie je podmienené signálom, ktorý hlási ukončenie predchádzajúcej inštrukcie. Inštrukcie vstupov a výstupov, kde sa spolupracuje s vstupno-výstupnými jednotkami, tieto jednotky majú cyklus práce iný. PCU tu uskutočňuje štartovanie, pričom ďalšie riadenie odovzdáva príslušnému vstupnému, alebo výstupnému zariadeniu.

1.2 Harwardská architektúra

Počítače s Harwardskou architektúrou majú oddelený adresný priestor pre program a pre údaje. Táto architektúra sa v súčasnosti používa pri niektorých jednočípových mikropočítačoch (jednočípový mikropočítač – všetky štruktúrne prvky ako procesor, pamäť, vstupné a výstupné obvody sú integrované na jedinom polovodičovom čipe). Procesor používa na adresáciu obidvoch pamätí a na prenos údajov a inštrukcií spoločné adresované a údajové vodiče. Rozlíšenie medzi prístupom k pamäti programu a k pamäti údajov je aktiváciou odlišných riadiacich signálov. Bloková schéma počítača s Harwardskou architektúrou (Obr. 1.3):



Obr. 1.3 Bloková schéma počítača s Harwardskou architektúrou

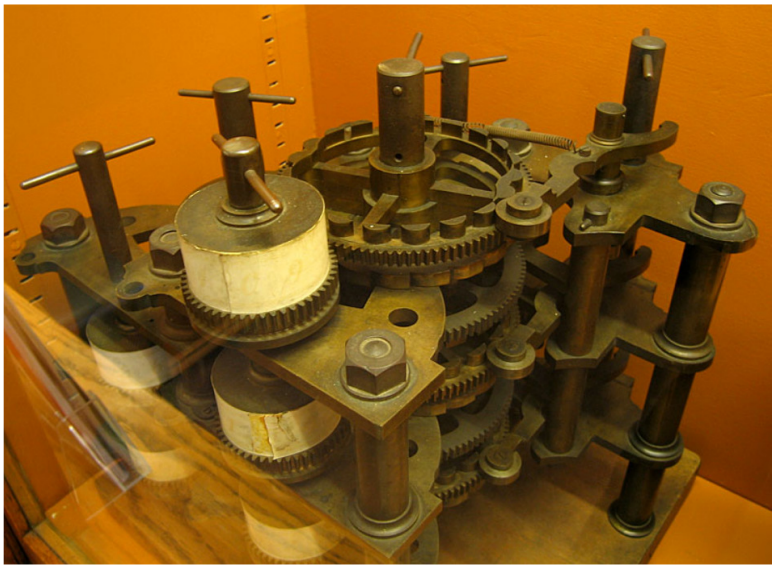
1.3 História počítačov

Generácie počítačov sa dajú rozdeliť do piatich generácií, od nultej až po piatu. Pričom nultá generácia predstavuje všetku výpočtovú technológiu do doby kým neprišli prvé elektrotechnické počítače, piatu generáciu počítačov ešte len nástup čaká.

1.3.1 Nultá generácia počítačov

Nultá generácia počítačov je všetka výpočtová technika až po nástup prvých počítačov, ktoré fungovali na základe elektronickej súčiastky relé.

Prvým výpočtovým zariadením by sme mohli nazvať *abacus* (ručné počítadlo), ktorého vznik sa datuje na rok približne 4000 pred naším letopočtom. V starovekej rímskej ríši sa používal *prístroj na meranie prejdenej vzdialenosti*, ktorý fungoval na základe ozubených drevených kolies a guľôčok. Do vstupného zásobníka sa uložili guľôčky a vo výstupnom zásobníku bol taký počet guľôčok, koľko míľ sa prešlo. Ďalším veľkým vynálezom bola *knihtlač*, ktorú vynašiel Johannes Gutenberg v rokoch 1444 až 1448. Mechanickú výpočtovú techniku najviac vyvinul Leonardo da Vinci (1452 - 1519) od mechanických bojových strojov až po *mechanickú kalkulačku*, ktorá fungovala na základe ozubených kolies. Vynález mechanickej kalkulačky zdokonalil Blais Pascal, ktorému sa pripísala zásluha za jej vynález (pascalina – rok 1642) a taktiež Gottfried Wilhelm von Leibnitz pridal funkcie tejto kalkulačke ako násobenie, delenie, odmocnina (rok 1674). V rokoch 1805 až 1808 vynašiel Joseph Marie Jacquard *automatický tkáčsky stroj*, kde vzor bol zakódovaný na prvých diernych štítkoch (kartičkách). O pár rokov neskôr v roku 1833 Charles Babbage vynašiel *univerzálny počítací stroj* (Obr. 1.4), tento stroj bol stále iba mechanický.

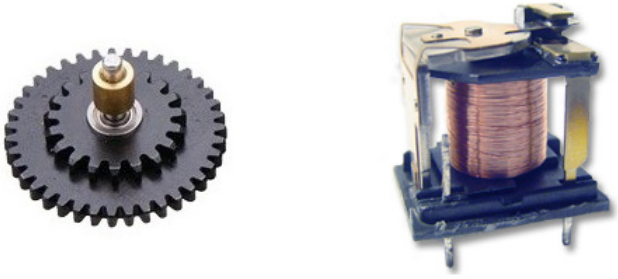


Obr. 1.4 Univerzálny počítací stroj

Herman Hollerith v roku 1890 vytvorili *tabulátor*, ktorý sa použil v spojení s triediacou skrinkou pri spracúvaní výsledkov sčítania ľudu v USA. Ručne dierované štítky sa vkladali do matrice a ich dierky určovali prechod elektrického prúdu, čo potom prenášalo na panel tabelačného stroja, bol to prvý elektrotechnický výpočtový stroj. Konrád Zuse v rokoch 1935 až 1945 vytvoril výpočtové stroje *Z1, Z2*, *Z3*, *Z4* ktoré fungovali na báze relé. Počas druhej svetovej vojny vznikalo viacero výpočtových strojov ktorých prvková základňa bolo relé napríklad: dešifrovací stroj *Enigma*, *Mark 1*, *Collossus, Mark 2*.

Základné charakteristiky počítačov 0. generácie:

- prvková základňa: mechanická (Obr. 1.5), elektromechanická, relé (Obr. 1.5),
- pamäť: mechanická (prepojky v lepšom prípade dierna páska a dierny štítok, nerozlišuje sa vnútorná a vonkajšia pamäť),
- jediná centrálna procesorová jednotka dekomponovaná na riadiacu a aritmetickú jednotku, mnoho funkcií v pevnom tvare,
- aritmetika v pevnej rádovej čiarke, sekvenčný program,
- programovanie: mechanické (prepojky, spínače),
- prenosové pamäťové médium: dierne štítky,
- individuálny prístup používateľa k počítaču,
- predstavitelia: MARK 1,2 (USA 1943), RMV (ZSSR), SAPO (ČSSR 1958).



Obr. 1.5 Prvková základňa procesora počítačov 0. generácie (mechanická vľavo, relé vpravo)

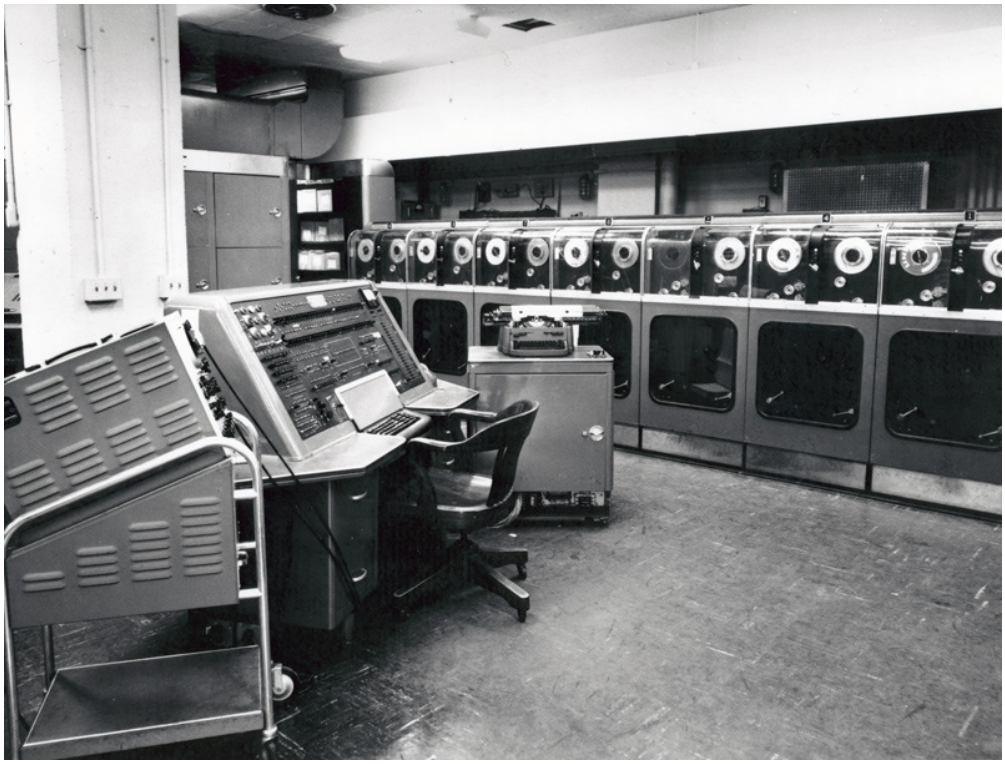
Parametre počítača Mark 2:

- 13 000 relé,
- 1 – 8 inštrukcií za sekundu (1 – 8 IPS),
- operačná pamäť 100 čísel s 10 platnými miestami (cca 0,5 kB).

1.3.2 Prvá generácia počítačov

Spočiatku počítače boli vyvíjané školami alebo vynálezcami za podpory vlády a súkromného sektoru. Vynálezci sami obsluhovali počítač. Iba tak ho mohli používať ďalší vedci, inžinieri alebo vláda. Vstup počítačov do sveta komercie je jednou z charakteristík prvej generácie počítačov.

Prvý počítač, ktorý našiel uplatnenie v obchode a priemysle bol univerzálny, automatický počítač UNIVAC (Obr. 1.6). Vyvinuli ho J. Presper Eckert a J. Mauchly, tvorcovia ENIAC-u, rýchlo našli komerčné využitie počítača. Títo dvaja vedci formovali a sprivatizovali spoločnosť a navrhli výrobnú dielňu pre UNIVAC. Avšak, pre nedostatok finančných prostriedkov ju predali spoločnosti Remington-Rand Comporation. Ako prvá začala využívať výhody UNIVAC-u vláda, no veľmi skoro sa našlo jeho využitie v obchode a priemysle.



Obr. 1.6 Počítač UNIVAC

Počítač nebol limitovaný na jeden účel. Môže počítať inventár, kalkulovať mzdovú listinu, monitorovať príjmové účty a kontrolovať hlavnú účtovnú knihu. Napriek tomu, že niekoľko tuctov ľudí obsluhovalo počítač, UNIVAC a iné počítače prvej generácie robili prácu za mnoho účtovníkov a účtovných revízorov. Teda spoločnosť môže oceniť počiatočnú

investíciu a zamerať sa na nákup počítačov a zakúpenie tuctu špecializovaných programov, pre zvýšenie presnosti a rýchlosti práce a efektívnejšie využitie personálnych zdrojov. Účtovníci a účtoví revízori nemuseli celé dni a hodiny kontrolovať účty. Ich novou úlohou bolo tlmočiť dáta vytvorené počítačom. Teda použitie počítačov prvej generácie v biznise nemalo dôsledok zníženia veľkého počtu zamestnancov, ale zmenu úlohy v zamestnaní.

Počítače prvej generácie používali *elektrónky*, zavedené Atanasoffom a Berrym. Elektrónky sú elektrické spínače, ktoré pracujú oveľa rýchlejšie ako v počítači Mark 1 mechanické spínacie zariadenia. Stroje s elektrónkami môžu vykonávať tisíc individuálnych operácii za sekundu, pomalšie ako dnešný štandard, ale na svoju dobu rýchlejšie. Bohužiaľ elektrónky sa prehrievali, čo zapríčinilo, že sa skoro vypálili. To spôsobovalo časté poruchy a krátke elektrické výkyvy. Počítače prvej generácie mali *klimatizované vnútro*. Vnútorné priestory mali veľmi veľké, pretože sa v nich nachádzalo viacero elektrónok rôznych veľkostí. Typický počítač prvej generácie mal veľkosť obývacej izby. So skorým príchodom počítačov prvej generácie, prišli i *dierne štítky* podobné tým, ktoré sa používali pri automatickom tkáčskom stroji. Ich počiatočné významy boli zamerané na vstup a výstup dát. Čítačky diernych štítkov, ktoré mohli prečítať nepatrné dierky vyrazené do štítku, mohli spracovať až 130 znakov za sekundu. Počítače prvej generácie nemali vonkajšiu pamäť iba vnútornú (operačnú pamäť). Veľa skorých počítačov používalo magnetickú bubnovú pamäť na uskladnenie a spracovanie údajov.

Softvér počítačov prvej generácie mal niekoľko nedostatkov závisiacich od rôznych typov úloh, ktoré mohli tieto počítače vykonať. Najviac počítačov prvej generácie mohlo vykonávať jednoduchý program limitovaný nastavením údajov. Na začiatku boli všetky programy v binárnom kóde. Písanie programov bolo extrémne náročné, veľa času sa strávilo drobnou prácou a programy často obsahovali chyby. Program sa spracovával priamo na dierny štítok. Neskôr sa vyvinul aj assembler.

Základné charakteristiky počítačov 1. generácie:

- prvková základňa: elektrónky (Obr. 1.7),
- pamäť: magnetická bubnová (nerozlišuje sa vnútorná a vonkajšia pamäť),
- jediná centrálna procesorová jednotka - CPU (skladajúca sa z ALU a PCU),
- aritmetika v pevnej rádovej čiarke, využívajúca programové počítadlo, inštrukcie vetvenia a akumulátor,
- účasť CPU na všetkých pamäťových a vstupno-výstupných operáciách,
- programovanie v strojovom jazyku alebo v assembleri,
- prenosové pamäťové médium: dierne štítky a pásky,
- individuálny prístup používateľa k počítaču,
- predstavitelia: ENIAC (USA 1946), IBM650 (USA 1953), URAL (ZSSR 1957), EPOS 1 (ČSSR 1961).

Obr. 1.7 Prvková základňa procesora počítačov 1. generácie (elektrónka)

Parametre počítača ENIAC:

- 17 468 elektróniek, 1 500 relé,
- 35 až 357 operácií za sekundu (35 – 357 IPS),
- operačná pamäť 100 čísel s 10 platnými miestami (cca 0,5 kB).

Parametre počítača EPOS 1:

- 7 200 elektróniek, 15 100 tranzistorov
- 5 000 až 20 000 operácií za sekundu (5 – 20 kIPS),
- operačná pamäť 1024 slov po 65 bitov (cca 8kB).

1.3.3 Druhá generácia počítačov

Druhá generácia, ktorá začala okolo roku 1959, až do polovice 1960, bola charakterizovaná *používaním tranzistorov* namiesto elektrónok. Tranzistory robili tú istú prácu ako elektrónky, ale boli menšie a rýchlejšie, potrebovali menšiu elektrickú energiu, boli viac hodnovernejšie a poskytovali oveľa väčšiu pamäť pre skladovanie inštrukcií a počítanie. Počítače druhej generácie mohli vykonávať viac ako 230 000 operácii za sekundu. Oproti tomu počítače prvej generácie iba 3500 až 1700 operácii za sekundu. Pretože tranzistory potrebujú menej energie ako elekrónky (asi 1/100 energie), druhá generácia počítačov bola menej nákladná na obsluhu ako jej predchodcovia.

Tak ako počítače prvej generácie, počítače druhej generácie boli obmedzované v typoch a množstve úloh, ktoré mohli vykonávať. V tejto generácii počítačov boli v obchode najpoužívanejšie hlavne účtovnícke programy. Vo väčšom obchode a v priemysle boli to práce v dávkach - veľké skupiny dát sa spracovávali počas jednej doby. Napríklad: spoločnosť zhromažďovala faktúry za dobu týždňa a uložila všetky tieto dáta pre spracovanie na jeden deň. Tento typ spracovania dát sa nazýva *dávkové spracovanie* (batch processing). Používalo sa na spracovanie mzdového listu, inventáru, splatnej faktúry atď. Dôležité bolo používanie externej pamäti na ukladanie dát. Pamäťové bunky boli oveľa rýchlejšie a spoľahlivejšie ako tie, ktoré sa používali v prvej generácii. Jeden z prvých typov elektronického ukladania dát bol založený na malom magnete okrúhleho tvaru, nazývaného ferit (core). *Feritová pamäť* bola rýchlejšia a viac spoľahlivejšia ako bubnová pamäť používaná v počítačoch prvej generácie. Počítače druhej generácie sa vyznačovali využívaním feritovej pamäte.

Ďalší dôležitý rozdiel bol v zavedení nezávislých *off-line zariadení*. Nemali trvalú komunikáciu s počítačom, ale boli k dispozícii, keď ich počítač potreboval. Napríklad keď počítač potreboval dáta z čítača diernych štítkov, čítač bol aktivovaný, dáta prečítal do počítača, a potom ostal v nečinnosti, až kým počítač nepotreboval opäť dáta. Dáta mohli byť poslané do nezávislej tlačiarne, a počítač mohol opäť začať spracúvať ďalšiu skupinu dát.

Jedným z off-line médií bola magnetická páska. Eckert a Mauchly vyvinuli tento typ média pre počítače prvej generácie, ale počítače druhej generácie boli prvé, ktoré to využívali vo veľkom rozsahu. Počítače mohli posielať informácie na pásky, na ktorých sa informácie ukladali a neskôr sa mohli vložiť z pásky späť do počítača. Vloženie bolo oveľa rýchlejšie s magnetickými páskami ako so štítkami. Informácie vkladané cez dierne štítky mohli byť vkladané 130 znakov za sekundu (16,25 B/s), počítač používajúci magnetickú pásku mohol čítať viac ako 6500 znakov za sekundu (0,79 kB/s).

Ďalší pokrok začal počas druhej generácie a ešte dnes sa používa *vývoj magnetického disku*. Spracovanie magnetickej pásky bolo pomalšie, pretože na obnovenie informácií magnetickej pásky počítač musel čítať pásku postupne (sekvenčne). Počítač číta

pásky zo začiatku pásky až po miesto, kde boli informácie uložené. S diskami počítač mohol pristupovať k žiaducim informáciám priamo, tak disk mohol oveľa rýchlejšie pracovať.

Počas druhej generácie počítačov sa začali vyvíjať programovacie jazyky. Najznámejšie programovacie jazyky boli *COBOL*, obchodne orientovaného jazyka, a *FORTRAN*, vyvíjaného firmou IBM pre vedcov a inžinierov. Vývojom vyšších programovacích jazykov úzko súvisí vývoj inštrukcií navrhovaných na kontrolu počítačových zdrojov. S vývojom off-line zariadení inštrukcie museli byť vyvíjané tak, aby mohli posielať alebo prijímať informácie do týchto zariadení, keď boli on-line.

Prvé operačné systémy boli primitívne. Modernejšie operačné systémy museli počkať na vývoj tretej generácie počítačov.

Základné charakteristiky počítačov 2. generácie:

- prvková základňa: tranzistory (Obr. 1.8),
- operačná pamäť: feritová,
- dvojúrovňová pamäť: vnútorná (operačná) pamäť a vonkajšia pamäť,
- aritmetika v pohyblivej rádovej čiarke, multiplexovanie pamäte, indexové registre,
- vstupno-výstupné (V/V) operácie vykonáva V/V jednotka – radič (V/V procesor),
- programovanie v jazykoch vyššej úrovne (Fortran, Algol, Cobol),
- prenosové pamäťové médium: magnetické pásky,
- vznik jednoduchších operačných systémov, začiatok uplatňovania filozofie prideľovania času a jednoduchých prerušovacích systémov, dávkový spôsob prístupu používateľa k počítaču,
- predstavitelia: IBM1401 (USA 1959), MINSK (ZSSR), Tesla200 (ČSSR 1966), ZPA600 (ČSSR 1968).

Obr. 1.8 Prvková základňa procesora počítačov 2. generácie (tranzistor)

Parametre počítača ZPA600 (komerčne predávaný EPOS 2):

- 20 000 tranzistorov,
- 40 000 operácií za sekundu (40 kIPS),
- operačná pamäť: 40 000 slov po 12 dekadických čísel (cca 300 kB) – feritová pamäť,
- pevná pamäť (off-line pamäť): 4 magnetické páskové pamäti,
- fotoelektrický snímač diernej pásky 1500 zn./s. (187,5 B/s),
- diernoštítková jednotka (mechanika) - snímač diernych štítkov,
- elektrický písací stroj, snímač a dierkovač diernej pásky 20 zn./s. (2,5 B/s), tlačiareň,

1.3.4 Tretia generácia počítačov

Vývoj a používanie integrovaných obvodov je znakom tretej generácie výpočtovej techniky . Táto generácia trvala od roku 1964 až do roku 1970. Integrovaný obvod pozostáva z tisícok obvodov vytlačených na malú silikónovú kartu nazývanú čip.

Ďalším dôležitým krokom vo vývoji tretej generácie počítačov bolo predstavenie rodín počítačov (family computers). Najdôležitejší detail je, že rodiny používajú rovnaké čipy a podieľajú sa na rovnakom operačnom systéme alebo metóde kontrolovania počítača. Počas roku 1960 IBM vyvinulo jednu z prvých počítačových rodín, série centrálnych počítačov nazývaných System/360. IBM System/360 alebo S/360 pozostával zo šiestich vzostupne kompatibilných počítačov. Vzostupne kompatibilný znamená, že programy, ktoré fungovali na starších (menších) modeloch boli funkčné aj na novších.

Pre túto kompatibilitu obchod mohol začať s malým počítačom a postúpil k väčšiemu počítaču bez nutnosti zmeniť software a preškoliť počítačových operátorov. Táto vlastnosť bola zvlášť atraktívna pre mnohých menších obchodníkov s menším kapitálom ako veľké firmy. IBM predalo viac ako 30 000 týchto sérií počítačov typu System/360.

Neskôr IBM vyvinulo novšiu sériu rodín počítačov 370. Tieto série 20 počítačov s doplnkovým hardwarom a softwarom boli tiež vzájomne kompatibilné. Teraz opäť firmy mohli začať s malými počítačmi a potom pokračovať s väčšími a výkonnejšími počítačmi.

So svojimi rodinami počítačov si IBM zabezpečili svoju pozíciu na čele počítačového priemyslu. Hoci veľa firiem kupovalo počítače, ďalšie stále neprestávali cítiť potrebu investovať do vlastných systémov. Ďalší vývoj tretej generácie diaľkových počítačových terminálov umožňoval týmto firmám spájať sa do jednoduchých veľkých centrálnych počítačov. Títo diaľkoví užívatelia, ako napríklad malé firmy, mali platiť vlastníkovi veľkého počítača poplatok za dobu, počas ktorej používali centrálny počítač. Malé firmy mohli napríklad používať diaľkový terminál na robenie svojich faktúr, alebo malý školský obvod mohol používať diaľkový terminál na zoznam miestností alebo študentov.

Najdôležitejší detail je, že zamestnanci v malých i veľkých firmách nestrácali zamestnanie s počítačmi počas tejto generácie. Začala sa objavovať nepatrná zmena vo firemnom manažmente . Počítače umožňovali firmám zachovať presné kópie záznamov. Toto viedlo k zmene úloh zamestnancov v spoločnosti. Napríklad účtovníci s prístupom k počítaču mohli stráviť viac času interpretáciou účtovníckych informácii a vytvárať odporúčania pre vlastníkov firmy. S príchodom počítačov stačilo nového zamestnanca zaučiť správne vkladať údaje do počítača a obsluhovať účtovnícky, alebo iný program.

Ďalšia dôležitá vlastnosť počítačov tretej generácie bolo zvýšené používanie magnetických diskových zariadení na ukladanie dát. Magnetický disk je vynikajúci pretože umožňuje priamy prístup k dátam a nie, ako to bolo predtým sekvenčne. Priamy prístup k dátam podstatne zvýšil rýchlosť výpočtu.

Počas tretej generácie počítačov boli vyvinuté nové programovacie jazyky ako BASIC, ktorý sa dal ľahko naučiť a bol na všeobecné použitie. Ďalší bol PASCAL. Pretože tieto programovacie jazyky sa dali ľahko naučiť a používať, veľa počítačových užívateľov si mohli vytvoriť taký program, ktorý potrebovali.

Dôležitá pre túto generáciu bola práca na skvalitnení operačných systémov, ktoré sa objavili počas druhej generácie. Operačné systémy spracúvajú dáta novým spôsobom. Rýchlosť spracovania dát tiež ovplyvňovala rýchlosť vstupných a výstupných zariadení. Počítač nie je v nečinnosti, pokiaľ sa prečítajú dáta z pomalého vstupného zariadenia, ale pokračuje v ďalšej úlohe. Operačný systém tiež umožňuje pristupovať viacerým užívateľom k tým istým dátam súčasne, napr. v knižniciach, na letiskách.

Základné charakteristiky počítačov 3. generácie:

- prvková základňa: tranzistorové mikromoduly (Obr. 1.9), integrované obvody SSI (small scale integration – malá integrácia: do 100 tranzistorov) a MSI (middle scale integration – stredná integrácia: do 1 000 tranzistorov),
- operačná pamäť:
  - ferit, polovodič
  - kapacita operačnej pamäte: rádovo 0,1 až 10 MB,
- prenosové pamäťové médium: magnetické pásky,
- existencia rodín počítačov kompatibilných smerom od jednoduchších k zložitejším modelom,
- mikroprogramové riadenia CPU,
- prúdové spracovanie (pipelining),
- ďalší rozvoj vyšších programovacích jazykov, rozšírenie o jazyky simulačné,
- multiprogramovanie podporujúce viacpoužívateľský prístup prostredníctvom prekrývania činnosti CPU a V/V jednotiek,
- operačný systém na podporu virtuálneho pamäťového priestoru so zdieľaním zdrojov,
- aplikácie v oblasti informačných a riadiacich systémov pracujúcich v reálnom čase,
- predstavitelia: IBM370 (USA 1964), EC1045 (ZSSR), EC1027 (ČSSR 1989).

Obr. 1.9 Prvková základňa procesora počítačov 3. generácie (tranzistorový mikromodul, najmenší tranzistor 3. generácie mal iba 10 μm)

Parametre počítača EC1027 (Obr. 1.10):

- CPU IBM 370 kompatibilný (procesory 3. generácie od IBM dosahovali výkon 2 až 1 800 kIPS),
- 8 MB RAM,
- 2x 100MB vymeniteľné disky Aritma a 6x 317 MB pevné disky MEMOREX,
- 4x 1600 BPI (bits per inch) magnetické pásky,
- 3 tlačiarne, 12 terminálov, 2 disketové čítačky,
- poľský teleprocesor (16 liniek synchro-asynchro 9600 b/s).

Obr. 1.10 Počítač EC1027

1.3.5 Štvrtá generácia počítačov

Miniaturizácia integrovaných obvodov je charakteristická pre štvrtú generáciu. (od 1970 po dnes). Mikročíp alebo mikroprocesor vykonáva milióny až miliardy operácii za sekundu. Firma Intel Corporation vyvinula prvý mikroprocesor, ktorý nazvala 4004, neskôr to bol 8008. Éru mikropočítačov zahájil mikroprocesor Intel 8080. Vážil iba niekoľko gramov a zaberal niekoľko štvorcových centimetrov. V porovnaní so skoršími počítačmi to bol obrovský prevrat. Elektronický čip je v dnešnej dobe výkonnejší, ekonomicky dostupnejší a je menší ako minca.

Ďalšia dôležitá vlastnosť štvrtej generácie počítačov je ich neobyčajne rozsiahle využitie. Počítače môžeme nájsť skutočne v každej malej firme, v každej škole a v miliónoch domácnostiach, pretože sú pomerne lacné. Štvrtá generácia počítačov dáva na výber aplikácie podľa účelov, nie len obmedzené aplikácie. Mikropočítače sa používajú na úradoch, vo veľkoobchodoch, v rôznych servisoch a vo všetkých druhoch podnikania.

Rozvoj mikroprocesorov bol sprevádzaný rozvojom ďalšieho hardvéru. Čo sa týka podstaty pamäte, moderné mikropočítače používajú pre vnútornú pamäť (operačná pamäť – RAM) polovodiče: metal-oxide semiconductor MOS. MOS je špeciálny čip, ktorý zásobuje veľké množstvo informácií na veľmi malé miesto. Obvody polovodičovej pamäte sú veľmi podobné mikroprocesoru pripojenému k silikónovým čipom. Polovodiče sú veľmi rýchle, avšak sú nestále, teda, pokiaľ je polovodič vypnutý, stráca všetko, čo je v ňom uskladnené.

Všetky rozvojové stupne technológií sprevádza pokrok v používaní externej pamäte a uskladnenie dát na disk. Mikropočítače používajú okrem disku malý "floppy disk" ako formu prídavnej pamäť pre uskladnenie dát. Neskôr vývoj priniesol optické disky a „flash pamäte“ (USB kľúče, SD). S mikropočítačmi, počítačovými programami muselo existovať aj pravidelné ukladanie do pamäte (v určitých intervaloch), pretože pamäť môže

dosiaľ neuložené dáta stratiť a to vtedy, ak sa predčasne sám vypne. Dáta teda môžu byť uskladnené na magnetickom disku pre neskoršie použitie.

Dôležitým softvérovým rozvojom štvrtej generácie počítačov sú databázové systémy. Databázové programy dovoľujú užívateľom počítačov zásobené dáta uložiť do iných formátov. Fakulty a univerzity, napríklad, používajú databázové programy na zásobovanie informácií o študentoch a usmerniť dáta podľa rôznych ciest (napr. podľa mena, bezpečnostného čísla atď.)

Vyššie programovacie jazyky BASIC, Pascal, rozvinuté v priebehu tretej generácie, a jazyk C, ktorý sa rozvinul počiatkom štvrtej generácie, nahrádzajú objektovo orientované jazyky ako napríklad Java a C#.

Parametre niektorých počítačov 4. generácie sú uvedené v kapitole 1.5

Základné charakteristiky počítačov 4. generácie:

- prvková základňa: integrované obvody LSI, VLSI, ULSI mikroprocesory,
- hlavná (operačná) pamäť: polovodičová (dynamická), uplatňovanie nových fyzikálnych princípov (holografia, laserová technika a pod.), od 1 MiB do 32 GiB (k roku 2014),
- kapacita vedľajšej pamäte 5 MB až 6 TB (k roku 2014),
- architektúry výkonných paralelných počítačových systémov (multiprocesorových a multipočítačových) so zdieľanou a distribuovanou pamäťou: rozvoj superpočítačov, výkonných personálnych počítačov, pracovných staníc a počítačových sietí, technická podpora riešenia vektorových operácii (vektorové procesory), operačné systémy, jazyky a kompilátory na podporu paralelného spracovania procesov (multiprocessing),
- vysoko špecializované a konverzačné jazyky, zjednocujúce styk používateľa s počítačom,
- podpora riešenia systémových programov technickými prostriedkami,
- viacprocesorové koncepcie počítačových systémov umožňujúcich paralelný prístup veľkého počtu používateľov (distribuované počítačové systémy),
- rozvoj lokálnych a regionálnych počítačových sietí.
- predstavitelia: mnoho.

Obr. 1.11 Prvková základňa procesora počítačov 4. generácie (procesor Intel Core i7 tranzistor dosahuje veľkosť 14 nm k roku 2014)

1.3.6 Piata generácia počítačov

Väčšina literatúr udáva začiatok 5. generácie počítačov v dobe, keď v domácnostiach budú algoritmy umelej inteligencie prevyšovať ostatné algoritmy. Hlavnými vstupnými zariadeniami nebudú klávesnica, myš a dotyková obrazovka, ale mikrofón, kamera a 3D senzory (napríklad zariadenia podobné Kinect-u), takže počítače sa budú riadiť povelmi zadávanými hlasom poprípade gestom.

Toto je len prognóza ako by 5. generácia mohla vyzerať, skutočnosť môže byť aj taká, že počítače prejdú na novú technológiu a to bude začiatok novej generácie. Napríklad tranzistor na báze uhlíka (gefran) by mohol byť procesor novej generácie.

Základné charakteristiky počítačov 5. generácie (prognóza):

- prvková základňa: integrované obvody s hustotou rádovo 10^{10} diskrétnych prvkov na čipe, vďaka využívaniu nových technológií,
- architektúry nových počítačových systémov v triede paralelných počítačov s extrémnym počtom procesorov resp. procesorových elementov (masívne paralelné počítače), počítačov na logické programovanie, databázových a znalostných počítačov, personálnych počítačov a počítačov sieťových prostredí,
- uplatňovanie princípov skalability pri návrhu architektúr počítačových systémov s dôrazom na dosiahnutie vysokých hodnôt spoľahlivostných parametrov a parametrov výkonnosť/cena,
- heterogénne počítačové systémy na riešenie rozsiahlych problémov (heterogeneous processing),
- inteligentný medzi styk (porozumenie reči, obrazu a prirodzeného jazyka) a dôraz na riešenie úloh umelej inteligencie,
- počítačové systémy integrovaných komunikácií so zdôraznením telekomunikačnej techniky a multimediálnej techniky.

1.4 Základné hardvérové súčasti počítača

Počítač sa skladá z viacerých súčastí zvaných hardvér. Tento hardvér môžeme rozdeliť do troch skupín a to základný hardvér počítača, rozširujúci hardvér počítača a periférie.

Základným hardvérom chápeme súčasti ktoré sú nutné pre chod počítača, týmito hardvérmi sú procesor (CPU), matičná doska, operačná pamäť (RAM), pevný disk (HDD, SSD), zdroj, šasi (skrinka).

Rozširujúcim hardvérom sú súčasti ktoré sa nachádzajú v šasi počítača a sú nutné k pripojeniu periférnych zariadení, ktoré nemajú radič na matičnej doske (alebo je potrebné nahradiť radič na matičnej doske lepším) sú to napríklad grafická karta, zvuková karta, mechanika pre optické disky, atď.

Periférnych zariadení počítača je mnoho rozdeľujeme ich na pamäťové, vstupné, výstupné a vstupno-výstupné. Pamäťové periférie sú optické disky (CD, DVD, BD, PCD), flash pamäte (SD, USB kľúče), ale aj samotné pevné disky. Vstupné sú klávesnica, myš, touchpad, joyistick, skener, kamera atď. Výstupné sú monitor, tlačiareň, projektor, atď. Vstupno-výstupné sú dotykové panely, dotykové obrazovky, atď.

V ďalšej časti tejto podkapitoly sa rozoberie spomínaný hardvér, jeho funkcionalita a parametre, ktoré môže nadobúdať.

1.4.1 Procesor

Procesor je najdôležitejšia súčiastka v počítači. Fyzicky je tvorený jedným čipom, ktorý má v sebe ukrytých niekoľko miliónov až miliárd elektronických súčiastok – tranzistorov.

Je jednou zo základných častí počítača. Jeho funkcia je interpretovať inštrukcie programu, ktorý je uložený v hlavnej (operačnej) pamäti. Pri tejto činnosti sa inštrukcie vyberajú z hlavnej pamäte, vykonávajú sa požadované operácie s operandami špecifikovanými v inštrukciách a uskutočňujú sa príslušné prenosy informácií medzi jednotlivými časťami počítača.

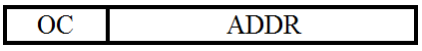
Procesor sa skladá z 2 hlavných častí:

- operačná časť
- riadiaca časť

Operačná časť vykonáva operácie s operandami na základe povelov z riadiacej časti. O výsledkoch operácií informuje riadiacu časť prostredníctvom príznakov. Operačná časť je zložená z ALU, registrov a komunikačných obvodov. *ALU* je určená na bezprostredné vykonanie aritmetických, logických a iných operácií s operandami. *Registre* (zápisniková pamäť) slúžia na prechodné uloženie operandov, vstupujúcich do operácií, ako aj na uloženie výsledku. *Komunikačné obvody* umožňujú vykonávanie medziregistrových prenosov použitím zbernice.

ALU slúži na vykonanie elementárnych aritmetických a logických operácií, na základe ktorých je možné realizovať ľubovoľné algoritmicky definované spracovanie číselných a nečíselných údajov. ALU realizuje aj operácie (posuvy, predikáty). Základ ALU na úrovni logických obvodov na realizáciu základných aritmetických operácií predstavuje *paralelná dvojková sčítačka*.

Riadiaca časť vyberá inštrukcie z pamäte, dekóduje ich a zabezpečuje ich vykonanie. Riadi spoluprácu procesora s okolím (komunikácia s pamäťou a V/V zariadeniami, obsluha prerušenia). Riadiaca časť uskutočňuje výber a dekódovanie inštrukcií a zabezpečuje ich vykonanie. Riadi spoluprácu procesora s okolím. *Inštrukcia* je príkaz pre procesor, ktorý mu určuje akú činnosť má vykonať. Vo všeobecnosti sa inštrukcia skladá z viacerých polí (Obr. 1.12):



Obr. 1.12 Všeobecný formát inštrukcie

Pole OC (operačný kód) špecifikuje operáciu, ktorá sa má vykonať. *Pole ADDR* obsahuje adresu operandu/operandov pre príslušnú operáciu. Vo všeobecnosti sa používajú jedno-, dvoj- alebo trojoperandové inštrukcie. Nie všetky inštrukcie potrebujú pole *ADDR*, napr. inštrukcia pre povolenie prerušenia.

Inštrukcie (inštrukčný súbor) delíme podľa vykonávanej činnosti na (typy inštrukcií):

- presunové inštrukcie,
- výpočtové inštrukcie,
- skokové inštrukcie,
- riadiace inštrukcie.

Presunové inštrukcie slúžia na presun údajov medzi registrami procesora, medzi registrom a OP (operačnou pamäťou), registrom a V/V zariadením alebo pamäťovými miestami navzájom.

Výpočtové (operačné) inštrukcie predpisujú vykonávanie aritmetických, logických alebo iných operácií nad operandami.

Skokové inštrukcie slúžia na zmenu lineárneho vykonávania programu. Vykoná sa inštrukcia, adresa ktorej je špecifikovaná v práve vykonávanej inštrukcii.

Riadiace inštrukcie sú určené na vykonanie špeciálnych operácií, ktoré priamo súvisia s riadiacou alebo operačnou časťou, napr. povolenie alebo zakázanie externého prerušenia, softvérové prerušene a pod.

Procesor s inštrukčnou sadou CISC (Complex Instruction Set Computer) sa vyznačuje zložitým inštrukčným súborom, ktorý je navrhnutý tak, aby priamo podporoval preklad z vyšších programovacích jazykov do strojového kódu procesora.

Procesor s inštrukčnou sadou RISC (Reduced Instruction Set Computer) sa vyznačuje redukovaným inštrukčným súborom. Inštrukcie sú jednoduché, ich vykonanie trvá krátko (v jednom strojovom cykle).

Inštrukčný cyklus sa vo všeobecnosti skladá z týchto fáz (prúdové spracovanie inštrukcií):

1. výber inštrukcie z pamäte,
2. dekódovanie inštrukcie,
3. výber operandov (ak sú v pamäti alebo vo vstupnom zariadení),
4. vykonanie požadovanej operácie nad operandami,
5. zápis výsledku do pamäte alebo výstupného zariadenia (ak má výsledok ostať v registri, táto fáza odpadá).

Procesor je k OP pripojený pomocou dátovej vonkajšej zbernice. Podľa jej šírky (16, 32, 64) vieme povedať koľko informácií sa presunie z/do OP počas pracovného cyklu procesora.

Ďalšia zbernica, ktorá sa nachádza v procesore je adresná zbernica. Jej šírka (16, 20, 24, 32, 36, 40) určuje akú veľkú časť pamäte RAM dokáže procesor zaadresovať.

Rýchlosť práce procesora je určená synchronizačnými hodinami (pamäťové moduly majú pomalšiu rýchlosť ako procesor). Aby bola možná bezproblémová komunikácia je zriadená vyrovnávacia pamäť CACHE. Existujú tri typy cache pamätí, ktoré procesor bezprostredne využíva:

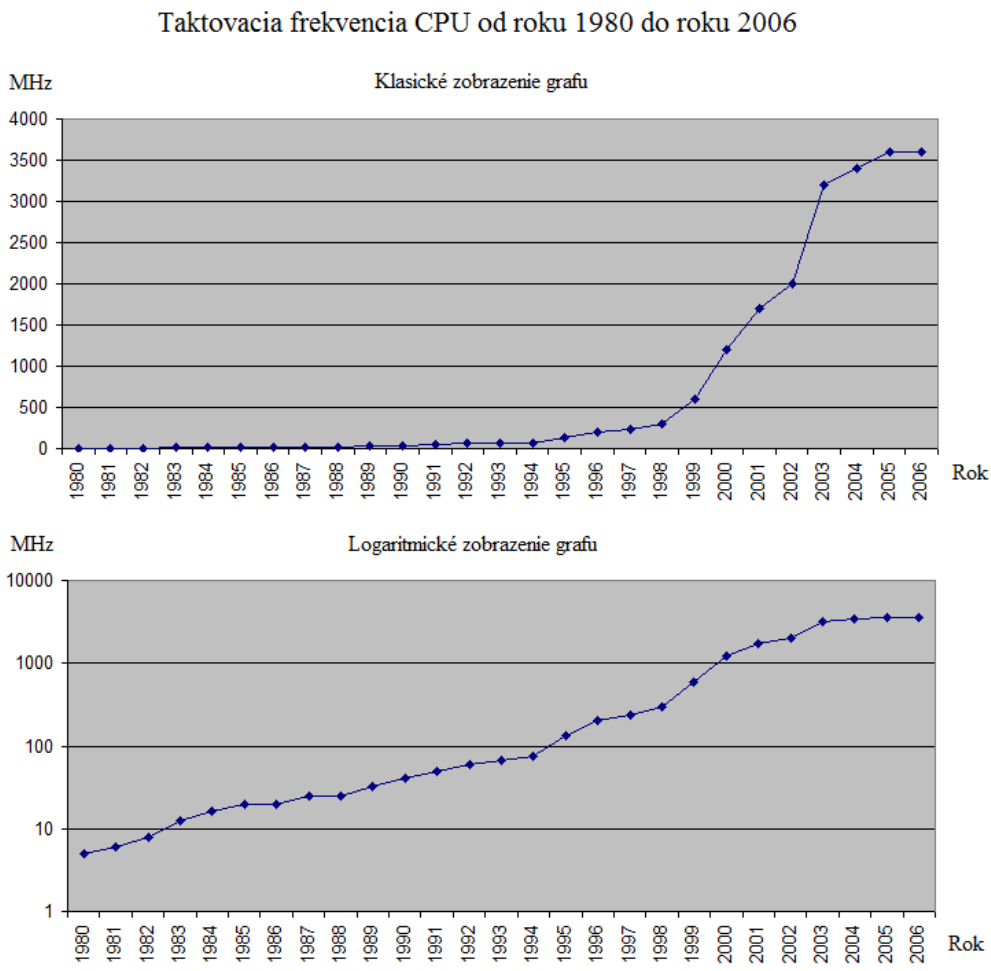
- L1 cache pamäť môže mať 8 až 128 kB (k roku 2014)
- L2 cache pamäť môže mať 256 kB až 4 MB (k roku 2014)
- L3 cache pamäť môže mať 4 MB až 20 MB (k roku 2014)

Od roku 2007 sa na trh dostali aj procesory s viacerými jadrami, čiže jeden procesor sa skladá z viacerých „podprocesorov“ nazývaných jadrá. Tým sa práca procesora sparalelizovala. Tieto jadrá majú vlastné CACHE pamäte L1 a L2 a jednu spoločnú CACHE pamäť L3.

Parametre:

- taktovacia frekvencia (kHz - GHz),
- frekvencia zbernice - FSB (kHz - GHz),
- architektúra:
  - šírky jednotlivých zberníc (16, 20, 24, 32, 64, ...),
  - počet jadier (1, 2, 4, 6, 8, ...),
  - inštrukčná sada (CISC, RISC, MMX, SIMD, ...),
- veľkosť pamätí CACHE - L1 (kB), L2 (kB, MB), L3 (MB),
- socket, pätica (AM1, AM3, FM2, 478, LGA775, LGA2011, ...),
- spotreba elektrickej energie (W).

Na nasledujúcom obrázku (Obr. 1.13) je znázornený vývoj taktovacej frekvencie CPU od roku 1980 do roku 2006. Rok 2006 je zvolený preto, lebo to bol posledný rok, keď sa na trhu ešte nedal kúpiť dvoj jadrový procesor. Takže po rok 2006 je ešte taktovacia frekvencia celkom dobrým výpovedným meradlom na porovnanie CPU. V Tab. 1.1 je časový vývoj veľkostí tranzistorov v nanometroch (10^{-9}m).



Obr. 1.13 Taktovacia frekvencia CPU od roku 1980 do roku 2006

Tab. 1.1 Časový vývoj veľkostí tranzistorov

Rok	Veľkosť tranzistora [nm]	Rok	Veľkosť tranzistora [nm]	Rok	Veľkosť tranzistora [nm]	Rok	Veľkosť tranzistora [nm]
1980	3000	1992	800	1999	180	2008	45
1981	2000	1994	600	2001	130	2010	32
1982	1500	1995	350	2004	90	2012	22
1985	1000	1998	250	2006	65	2014	14

1.4.2 Matičná doska

Matičná doska funguje ako hlavná prepojovacia jednotka medzi všetkými súčasťami počítača.

Podľa toho aký typ CPU základná doska podporuje sa matičné dosky líšia použitým socketom (konektorom) pre osadenie jednotlivých typov mikroprocesorov (Socket 478, LGA775, LGA1155, LGA2011, atď.)

Okrem šírky spracovaného slova je dôležitým faktorom pri hodnotení základnej dosky aj rýchlosť spracovania údajov v CPU od PC/XT s CPU 8086 alebo 8088 s frekvenciou približne 5 MHz cez Pentium II s 0,5 GHz po Intel Core i7 s 3,5 GHz a štyrmi jadrami.

Dôležitým pojmom pri matičných doskách je *systémová zbernica*, ktorá predstavuje prepojenie medzi jednotlivými obvodmi základnej procesorovej dosky (matičnej dosky).

Matičná doska obsahuje *sloty* sú to miesta (konektory), do ktorých sa zasúvajú prídavné dosky (karty) na jednotlivé zbernice, taktiež sa sloty nazývajú aj konektory pre operačné pamäte (RAM).

Staršie matičné dosky obsahovali konektor (socket) pre matematický koprocesor bol to taktiež integrovaný obvod ako CPU určený na rýchle výpočty v pohyblivej rádovej čiarke, bol nápomocný hlavnému CPU. V súčasnosti je integrovaný do CPU (z hľadiska histórie sa s CPU 80486SX predaj samostatnej súčiastky ukončil).

Matičná doska obsahuje riadiace obvody procesorovej (matičnej) dosky zvaných *chipset*. Tieto obvody sprostredkovávajú komunikáciu medzi mikroprocesorom, operačnou pamäťou a ostatnými časťami matičnej dosky vrátane zberníc, riadenia taktovacích frekvencií, riadenia napájania, či niektorých portov.

Čipová sada sa delí na časti, ktoré sa špecializujú na istú formu komunikácie:

a) northbridge,
b) southbridge,
c) super I/O.

Northbridge zabezpečuje komunikáciu medzi procesorovou zbernicou, grafickou zbernicou a zbernicou komunikujúcou s ďalšími časťami systému.

Southbridge zabezpečuje prepojenie medzi PCI a ISA zbernicami, ktoré pracujú na rôznych frekvenciách a CMOS pamäťou

Super I/O zabezpečuje pripojenie periférií k systému (radič sériového portu, paralelného portu, klávesnice a myši, dávnejšie obsahoval a aj radič disketovej mechaniky).

Pri novších doskách sa namiesto čipov southbridge a northbridge používajú rozbočovače (hub). Northbridge používa MCH (memory controler hub) a southbridge používa ICH (input/output controler HUB). MCH obsahuje prepojenie medzi RAM a CPU a radiče pre rýchly PCI-express (pre grafiku), dávnejšie obsahoval radič pre AGP. ICH obsahuje radiče pre IDE, USB, Ethernet, zvuk, obsahuje taktiež prepojenie s integrovanou grafikou a super I/O.

V ROM BIOS-e sa nachádza základný softvér, ktorý umožňuje základnú komunikáciu s hardvérom základnej dosky a zároveň uchováva informáciu o konfigurácii systému. V tomto obvode sa nachádza základná diagnostika komponentov a vykonáva sa vždy po štarte systému.

Takže základné časti, ktoré môžeme nájsť na matičnej doske sú: pätica pre procesor, čipovú sadu, vstupno/výstupné obvody, ROM BIOS (na pamäti flash), pätice pre pamäťové moduly, pätice pre rozširujúce karty a batériu na udržanie systémového času, keď nie je matičná doska napájaná.

Systémová zbernica FSB (Front Side Bus) je základným komunikačným kanálom moderných základných dosiek. Prostredníctvom nej komunikuje mikroprocesor s čipovou

súpravou (čipsetom - jedným alebo dvoma integrovanými obvodmi, ktorých úlohou je riadenie zberníc a dohliadanie na „plynulosť“ prepravy údajov vnútri počítača) a ďalšími komponentmi (pamäťou cache, operačnou pamäťou a niektorými ďalšími zariadeniami). Zbernica FSB je 64 bitová a môže pracovať s frekvenciami 66 MHz až 300 MHz resp. v moderných základných doskách býva takt zbernice násobený štyrmi, takže sa dá dosiahnuť aj 1,2 MHz, hoci „fyzicky“ je zbernica taktovaná nižšie (300 MHz, 19,2 GB/s).

Obr. 1.14 Matičná doska MSI Z77A-GD65

Parametre:

- frekvencia zbernice – FSB (kHz - GHz),
- socket, pätica na CPU (AM1, AM3, FM2, 478, LGA775, LGA2011, ...),
- typ slotu na pamäť RAM (SIPP, SIMM 30, SIMM 72, SDRAM, DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM, DDR4 SDRAM, ...),
- verzia zbernice IDE (PATA, SATA) a ich počet,
- chipset – čipová sada,
- integrované radiče (radič grafiky, radič zvuku, radič USB, atď.),
- počet slotov PCI, PCI-e pre rozširujúci hardvér (pred tým ISA, EISA, AGP),
- veľkosť (najčastejšie: ATX, miniATX a microATX)

1.4.3 Operačná pamäť

Hlavná (operačná) pamäť obsahuje práve vykonávané programy a spracúvané údaje. Kapacitou pamäte sa rozumie množstvo informácií, ktoré je schopná pamäť uchovať. Hlavná pamäť je realizovaná ako samostatný funkčný blok, ktorý sa skladá z jedného alebo niekoľkých integrovaných obvodov. Hlavná pamäť je realizovaná dynamickými pamäťami, ktoré majú pri nízkej spotrebe energie veľkú kapacitu. Doba prístupu hlavnej pamäte je rádovo v jednotkách až desiatkách ns.

Operačné pamäte v počítačoch sú pamäte s náhodným prístupom (RAM – Random Access Memory) pre tieto pamäte platí, že doba prístupu pre jednotlivé bunky je rovnaká, nezáleží od ich umiestnenia v pamäti (hlavná pamäť von Neumannovského typu). Operačná pamäť sa skladá z dvoch častí, z ktorých jedna je typu ROM a druhá RWM.

Pamäte pre čítanie a zápis (RWM – Read/write memory) v týchto pamätiach sa informácia dá v priebehu činnosti kedykoľvek zapísať a kedykoľvek čítať. Obsah pamäte sa po vypnutí napájacieho napätia vymaže.

Pamäte iba pre čítanie (ROM – Read only memory) z týchto pamätí sa informácia dá iba čítať.

Z hľadiska fyzickej realizácie je hlavná pamäť vytvorená z polovodičových pamätí . Polovodičové pamäti typu RWM sú realizované na polovodičovom čipe a vyrábajú sa ako statické alebo dynamické. V dnešných počítačoch sa používajú dynamické pamäti.

Pri statických polovodičových pamätiach RWM (SRAM) je základný pamäťový element realizovaný ako preklápací obvod, ktorý po zápise informácie zostáva v stabilnom stave, ktorý sa zmení zápisom novej hodnoty.

Pri dynamických pamätiach RWM (DRAM) je základný pamäťový element realizovaný pomocou parazitnej kapacity, ktorá sa pri zápise nabije. Vplyvom zvodového prúdu sa elektrický náboj vybíja. Pri dynamických pamätiach je nutné informáciu periodicky obnovovať (refresh).

Hlavná pamäť je realizovaná z polovodičových pamätí, ktoré sú fyzicky realizované ako integrované obvody. Organizácia pamäťových buniek v rámci jedného integrovaného obvodu je:

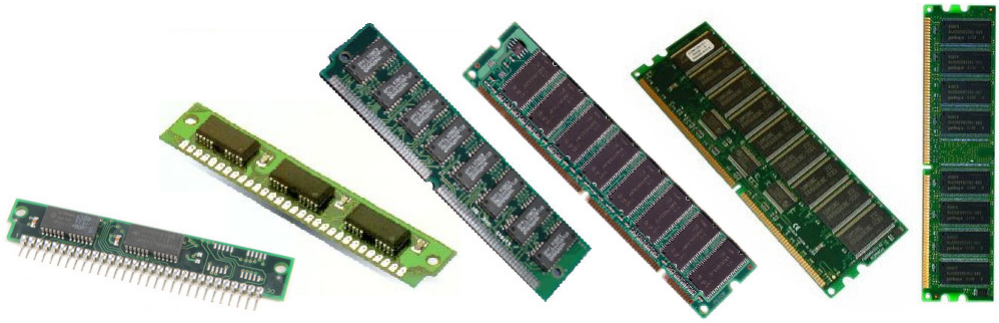
a) pri statických pamätiach: pamäte RWM a pamäte ROM
b) pri dynamických pamätiach: pamäte RWM

Statické pamäte RWM a pamäte ROM majú obyčajne slabikovú organizáciu, teda do jednej bunky pamäte je možné uložiť slabiku (B). Šírka údajovej zbernice pamäte je 8 bitov. Kapacita pamäte pri slabikovej organizácií sa udáva v bajtoch (B) a vypočíta sa 2^n, kde *n* označuje šírku adresnej zbernice pamäte.

Dynamické pamäte RWM majú bitovú organizáciu, teda do jednej bunky pamäte je možné uložiť 1 bit. Šírka údajovej zbernice je 1 bit. Kapacita pamäte pri bitovej organizácií sa udáva v bitoch (b) a vypočíta sa ako mocnina 2^n, *n* je šírka adresnej zbernice pamäte.

Reálne má hlavná pamäť slabikovú organizáciu, preto ak sa na jej výstavbu použijú pamäti s bitovou organizáciou, pri slabikovej organizácii hlavnej pamäte je potrebné použiť osmice integrovaných obvodov, ktoré majú spoločný výberový signál a spoločné riadiace a adresné signály.

Prvá vymeniteľná operačná pamäť v personálnych počítačoch (IBM PC, viď kapitolu 1.5) bola nazvaná SIPP (single in-line pin package, Obr. 1.15), ale výmena aj tak vyžadovala kvalifikovanejšiu montáž (cínovanie). Ďalším vývojovým krokom bola pamäť SIMM (single in-line memory module, Obr. 1.15) mala 30 pinov, tak ako SIPP a piny mali zhodné poradie jediný rozdiel bol v tom, že matičné dosky už mali potrebný slot (podobný dnešnému len kratší) pre pamäťové moduly a inštalácia bola rovnaká ako v dnešných počítačoch. Po 30 pinovej pamäti SIMM nasledovala 72 pinová pamäť SIMM (Obr. 1.15). Po pamätiach SIMM, už prišli dnešné SDRAM (Synchronous Dynamic Random Access Memory), ale tie už tiež prešli istým vývojom. Prvá bola klasická SDRAM (Obr. 1.15), nasledovala DDR SDRAM (Obr. 1.15), potom DDR2 SDRAM, ďalej DDR3 SDRAM (Obr. 1.15) a najnovšia je DDR4 SDRAM (k roku 2014). Tieto typy pamätí RAM nie sú kompatibilne, nakoľko majú rovnakú dĺžku, bola vytvorená na moduloch a slotoch zarážka, ktorá sa pri každej verzií pamäte posúva, aby ju nebolo možné nainštalovať (vložiť) do nesprávnej matičnej dosky.



Obr. 1.15 Operačné pamäte podľa vývoja (z ľavej strany do pravej: SIPP, SIMM 30, SIMM 72, SDRAM, DDR SDRAM, DDR3 SDRAM)

Parametre:

- frekvencia zbernice – FSB (kHz - GHz),
- typ pamäte (SIPP, SIMM30, SIMM72, SDRAM, DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM, DDR4 SDRAM),
- šírka zberníc,
- kapacita (MiB - GiB),
- spotreba (W).

Pre väčší prehľad je ponúknutá Tab. 1.2 s časovým vývojom typov pamätí RAM, kapacitou ich modulov ako aj kapacitou najpredávanejšieho modulu danej pamäte a tabuľka obsahuje aj rok kedy sa daný modul už bežne vyskytoval na trhu (v maloobchodných reťazcoch).

Tab. 1.2 Časový vývoj pamätí RAM, ich kapacity a výskyt na trhu

Rok vývinu	Typ pamäte RAM	Kapacita modulu [MB]	Najpredávanejší modul [MB]	Rok bežného výskytu na trhu
1978	SIPP	do 1	0,25	1982
1983	SIMM 30	0,25 - 16	4	1990
1987	SIMM 72	1 - 128	16	1995
1993	SDRAM	8 - 1024	512	1999
2000	DDR SDRAM	128 - 2048	1024	2003
2003	DDR2 SDRAM	128 - 4096	2048	2006
2007	DDR3 SDRAM	512 - 8192	4096	2010
2014	DDR4 SDRAM	4096 - 8192	8192	asi 2016

1.4.4 Pevný disk

Pevný disk (HDD – hard disk drive) slúži na uchovanie informácií, ktoré sa momentálne nepoužívajú a na archiváciu informácií. Na rozdiel od hlavnej pamäte, k vonkajším pamätiam (HDD, DVD, CD, SD, ...) CPU pristupuje ako ku V/V zariadeniam. Pevné disky majú dobu prístupu v jednotkách ms a kapacitu niekoľko GB až TB.

Pevné disky sú pamäte so sekvenčným prístupom, teda adresované miesto sa sprístupní až po prehľadaní predošlých buniek.

V dnešnej dobe sa ako základné vedľajšie pamäťové médium v počítačoch používa aj SSD (Solid state drive) okrem magnetického HDD. SSD je mechanika s nepohyblivým médiom a informácie sa ukladajú na flash pamäte, ktoré sú v ňom inštalované.

Zatiaľ najpredávanejšie a najpoužívanejšie základné vedľajšie pamäte sú magnetické disky. Prvé kapacity týchto diskov sa rátali v MB, dnes sú to už TB. Magnetické disky menia mikroskopicky malé oblasti povrchu disku tak, že niektoré predstavujú nuly a niektoré jedničky.

Organizácia dát na disku záleží od súborového systému operačného systému (OS). Existuje viacero typov DOS a prvé verzie OS Windows používali FAT, FAT16, FAT32, dnes OS Windows používa NTFS. OS Unix používa mnoho súborových systémov, príkladmi sú UFS, alebo ext2. Tieto súborové systémy delia disk na dve časti a to na súborovú tabuľku (adresy jednotlivých súborov) a samotné dáta.

Magnetický disk pri zápise dát budí pomocou hlavy (elektromagnetu) nad kovovou platňou magnetické pole, ktoré usporiada elektróny podľa potreby (zakóduje rád jednotiek a núl – vytvorí informáciu). Pri čítaní hlava nebudí magnetické pole, ale usporiadané elektróny budia magnetické pole na elektromagnetickej hlave, vďaka čomu elektromagnet toto pole prerobí na elektrický prúd, ktorý sa prečíta a dekóduje na informáciu.

Keďže každý prvok v počítači má inú rýchlosť zápisu a čítania, tak aj magnetické HDD a SSD obsahujú vyrovnávaciu pamäť CACHE.

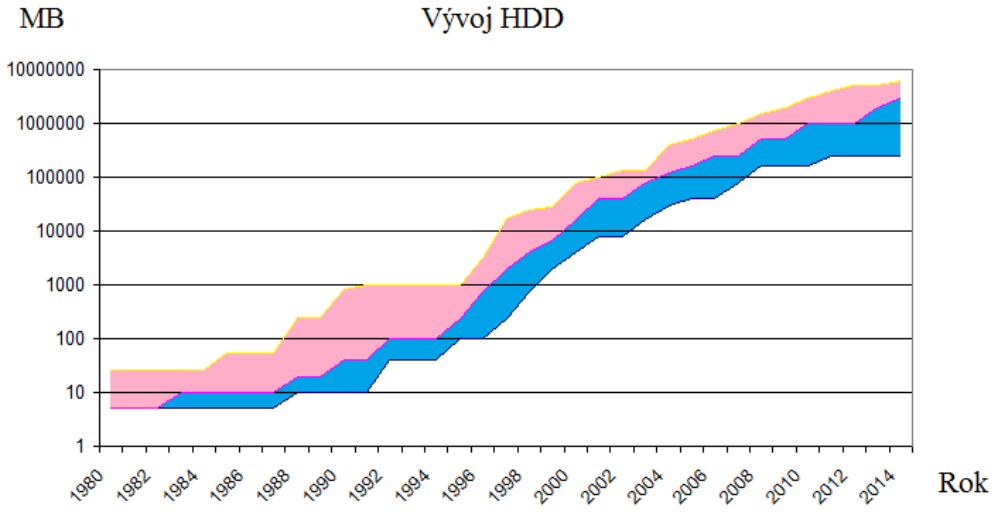
Obr. 1.16 Pevný magnetický disk (HDD) od firmy Intel

Parametre:

- kapacita (MB až TB),
- rýchlosť zápisu a čítania (kb/s až Gb/s),
- kapacita pamäte CACHE (kB - MB),
- počet otáčok pri HDD (3 600 - 15 000 RPM) / počet zápisov na jednu pamäťovú bunku pri SSD,
- dátový konektor (PATA, SATA) a štandard (ATA-1 až ATA-8, SATA 1,5 Gb/s, SATA 3Gb/s, SATA 6 Gb/s),
- spotreba (W).

Tak ako u všetkých počítačových komponentov aj pri HDD sa kapacita zvyšovala rádovo. Nasledujúci obrázok (Obr. 1.17) vystihuje situáciu najlepších a najdrahších

magnetických HDD (najvyššia hodnota na grafe), HDD za prijateľnú sumu (stredná hodnota na grafe, tá suma je okolo 100 USD) a HDD za najnižšiu sumu (najnižšia hodnota na grafe). Pri najnižšej sume sa neberie ohľad na výpredaj HDD pod nákupnú cenu výrobcu.



Obr. 1.17 Vývoj kapacít magnetických HDD

1.4.5 Zdroj a šasi

Skôr ako šasi počítača sa používa skrinka počítača, slúži na uloženie všetkého základného a rozširujúceho hardvéru a taktiež na uloženie zdroja. Skrinky majú 3 základné veľkosti (midi, mini a micro), ich veľkosť je prispôsobená veľkosti matičnej dosky (ATX, miniATX, microATX), sú aj iné veľkosti skriniek, ale tieto sú najpredávanejšie. Základné prevedenia sú taktiež tri a to tower, desktop a normované do serverových rack-ov. Vzhľady sú rôznorodé, jediné praktické parametre skriniek sú asi počet otvorov pre optickú mechaniku, počet USB portov a počet čítačiek pamäťových kariet, ale všetky tieto parametre sú obmedzené veľkosťou skrinky.

Parametre skrinky počítača:

- veľkosť (midi pre matičnú dosku veľkosti ATX, mini pre matičnú dosku veľkosti miniATX, micro pre matičnú dosku veľkosti microATX, ...),
- prevedenie (tower, desktop, 1U, 2U, 3U, ...),
- počet otvorov pre optickú mechaniku (0, 1, 2, 3, ...),
- počet USB portov (0, 1, 2, 3, 4, ...),
- počet čítačiek pamäťových kariet (najčastejšie SD).

Najdôležitejším parametrom zdroja (Obr. 1.18) je výkon, ak máme hardvér z vysokou spotrebou (príkonom) potrebujeme tomu prispôsobiť aj zdroj. Priamo úmerne s výkonom sa zvyšuje aj príkon (300 W až 1000 W). Veľmi dôležitý je aj typ napájania, ale keďže najčastejšie sa takýto tovar nakupuje na domácom trhu, tak všetky zdroje by mali byť na elektrické napájanie 230 V. Ďalšími dôležitými parametrami sú typy a počet napájacích konektorov pre pevné a optické disky (PATA, SATA) a napájací konektor pre matičnú dosku (tento konektor je už normovaný). Nové počítače už konektor PATA

nevyužívajú, takže typ konektora už nie je až takým sledovaným parametrom a ich počet. Posledným merateľným parametrom je hlučnosť zdroja.

Obr. 1.18 Zdroj

Parametre zdroja:

- napájanie (230 V),
- výkon (W),
- príkon (W),
- typy napájacích konektorov (SATA, PATA),
- počet napájacích konektorov (1, 2, 3, ...),
- hlučnosť (dB).

1.4.6 Rozširujúci hardvér

Najčastejším rozširujúcim hardvérom sú grafické, zvukové a sieťové karty. Ktoré sú už po väčšine aj integrované na matičnej doske, ale stále sú výkonnejšie externé karty ako integrované radiče na matičnej doske. Okrem spomínaného rozširujúceho hardvéru ešte existuje mnoho ďalších ako televízna karta na spracovanie televízneho analógového signálu, rozširujúca karta pre pamäťové karty, bluetooth karta, karta s USB radičom, karty rôznych technologických rozhraní, a mnohé ďalšie.

Grafické karty (Obr. 1.19) pozostávajú z vlastného procesora a operačnej pamäte, ich úlohou je odbremeniť CPU a RAM počítača od výpočtov spojených z grafikou vykresľovanou na monitore počítača.

Úlohou zvukových kariet je interpretovať zvukový signál na stereo zvuk, poprípade zvuk domáceho kina (dolby surround 5.1, 7.1).

Sieťové karty slúžia na pripojenie počítača do siete, toto pripojenie nemusí byť pomocou sieťového kábla ale aj bezkáblové (bezdrôtové) pripojenie (WiFi).

V dnešných počítačoch sa tieto rozširujúce karty vkladajú do slotov zbernice PCI-express, staršími zbernicami pre vstupno-výstupné zariadenia boli PCI, AGP (vyhradená pre grafickú kartu), EISA, ISA.

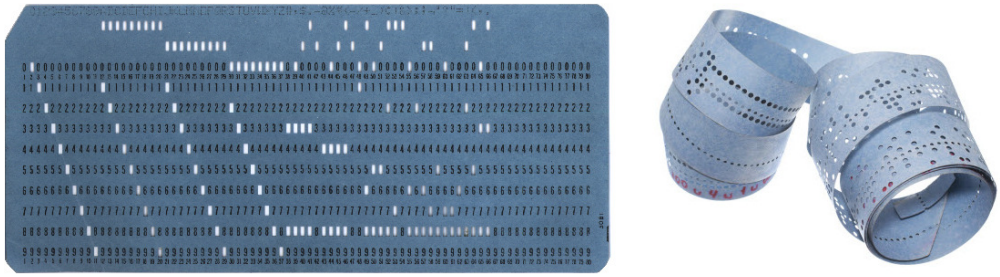
Obr. 1.19 Grafická karta od firmy Asus

1.4.7 Pamäťové periférie

Pamäťové periférie slúžia na prenos a archiváciu informácií. Môžeme ich rozdeliť podľa princípu funkčnosti na:

- mechanické (dierne štítky, dierne pásky),
- magnetické (magnetické pásky, disky a diskety),
- optické (CD, DVD, BD),
- tranzistorové (flash pamäte: SD, USB kľúče, SSD a iné).

Dierne štítky a pásky (Obr. 1.20) fungovali na jednoduchom princípe, dierka na páske zodpovedala jednotke a neprederavené miest zodpovedalo nule. Klasický dierny štítok mal 80, alebo 90 stĺpcov a 10 riadkov (100 B, 112 B). Dierne pásky mali šírku 5 až 8 značiek, pričom každý záznam zodpovedal jednej informačnej jednotke, takže najčastejšie využívanou páskou, bola páska s ôsmimi značkami. Kapacita záležala na dĺžke pásky. K týmto médiám existovali aj čítacie a zapisovacie (automatické dierkovače) mechaniky.



Obr. 1.20 Dierny štítok a páska

Nástupcom diernych štítkov a pások bola magnetická páska, ktorá slúžila okrem prenosového média aj ako pevná pamäť (viď Obr. 1.6, alebo menší variant magnetofónová páska do magnetofónu). Princíp fungovania je rovnaký, ako bol vysvetlený pri magnetických HDD (kapitola 1.4.4.) len použitý materiál na ktorý sú dáta nanášané je iný. Ďalším vývojovým stupňom periférnych médií bol magnetický HDD, ktorý sa používa dodnes (k roku 2014). Neskôr prišli magnetické diskety FDD (floppy disk drive, Obr. 1.21). Princíp fungovania bol podobný ako pri predošlých magnetických médiách. FDD sa vyrábali v niekoľkých rôznych veľkostiach najznámejšie z nich boli 8"(203 mm), 5,25"(133 mm) a 3,5"(89 mm). Diskety o veľkosti 8" mali kapacitu od 79,75 kB do 1200kB a používali sa najmä v 70. rokoch. Nástupcom boli 5,25" diskety s kapacitou od 87,5 kB do 1,2 MB a práve 1,2 MB boli najpoužívanejšie, používali sa v 80. rokoch. Diskety s veľkosťou 3,5" mali kapacitu od 280 kB po 750 MB (ZIP disketa) najpopulárnejšie a najrozšírenejšie boli diskety s kapacitou 1,44 MB a používali sa v 90. rokoch.

Obr. 1.21 Diskety (z ľavej strany do pravej: 8", 5,25" a 3,5" disketa)

Po magnetických disketách nastúpila éra optických médií (Obr. 1.22). Výskum optických médií začal v dobe keď sa magnetické diskety dostali na trh (70. roky). Prvý komerčne najznámejší opticky disk bol CD (Compact Disc). Jeho predchodcom bol LD (Laser Disc), ale ten bol známi najmä v Japonsku, prehrávače sa v Severnej Amerike pre cenu zle predávali a do Európy sa skoro ani nedostali. LD slúžil najmä na záznam videí (filmov), jeho kapacita bola približne 300 MB. Najbežnejšia kapacita CD je 750 MB, v 80. rokoch slúžil ako audio médium (na prehrávanie hudby). Mechanika na zaznamenávanie dát („napaľovačka") s CD-R sa na trh dostala začiatkom 90. rokov. Nástupcom CD bolo DVD (Digital Versatile Disc) prioritne určené ako nosič filmov na trh prišiel v roku 1997 a prvá zapisovacia mechanika v roku 2001. Dnes sa DVD používa aj na audio záznamy (hudbu) a na dáta (softvér, archivácia), najbežnejšia kapacita je 4,7 GB. Po DVD prišli HD DVD a Blu-Ray Disc (BD) približne s rovnakou technológiou, ale na trhu sa udržali BD s kapacitou 25 až 128 GB. Prvé BD prehrávače sa dali kúpiť v roku 2006 a v tom istom roku prišli na trh aj zapisovacie mechaniky. Čítanie dát z optických diskov funguje na princípe odrazivosti svetla, pričom čítacia hlava obsahuje vysielač a prijímač lúčov. Na mieste čítania dát vysielač vyšle lúč, potom prijímač tento lúč zachytí a na základe

množstva odrazených lúčov sa vyhodnotí či ide o logickú nulu, alebo jednotku. Rozdiel medzi jednotlivými optickými diskami je vlnová dĺžka vyslaných lúčov, čím je vlnová dĺžka nižšia tým menšie body sa dajú čítať. CD využíva vlnovú dĺžku 780 nm (hranica infračerveného žiarenia a viditeľného červeného svetla/lúča), DVD využíva 650 nm (hranica medzi červeným a oranžovým lúčom), BD využíva 405 nm (fialový lúč, blízko hranice ultrafialového žiarenia). Zapisovacie („napaľovacie“) mechaniky pomocou lasera nahrejú dátové miesto tak, aby sa pri čítaní odrážalo svetlo podľa potreby. Ďalší spôsob zápisu na optické disky je lisovaním (predpripravená hliníková fólia s dátami), zápis dát je skoro okamžitý, ale finančne sa to oplatí iba pri veľmi vysokom množstve objednávok diskov s rovnakým obsahom dát. Výskum optických diskov pokračuje, už teraz vo výskume existujú optické disky, ktorých teoretická kapacita dosahuje desiatky TB. Normovaný priemer pre optické disky je 12 cm a 8 cm (mini), LD mal priemer 20 cm a menšia verzia 12 cm.

Obr. 1.22 Optické disky (z ľavej strany do pravej: CD, DVD, BD)

Ďalšou pamäťovou perifériou je pamäť flash. Pomocou tejto pamäte je prenos dát najjednoduchší. Najprv slúžila ako pamäťové médium pre digitálne fotoaparáty (90. roky), neskôr sa začali používať v smartfónoch, kamerách, tabletoch, atď. Tieto flash pamäte nazývame pamäťové karty. Najpopulárnejším prenosovým médiom flash pamätí je USB kľúč. Prvý USB kľúč sa dal kúpiť pred Vianocami v roku 2000 a mal kapacitu 8 MB, dnes (k roku 2014) sa na trhu vyskytujú USB kľúče s kapacitou od 4 GB do 1 TB (Obr. 1.23). Najrozšírenejšou pamäťovou kartou je SD (Secure Digital) jej kapacita na trhu je od 2 GB do 256 GB (Obr. 1.23). Ďalším menej rozšírenými pamäťovými kartami sú MMC (Multimedia Card), CF (Compact Flash) a MS (Memory Stick). Medzi flash pamäte patrí aj SSD. Základom flash pamäte je zrýchlená EEPROM (Electrically Erasable Programmable Read-Only Memory), ktorá využíva tranzistor s dvoma hradlami riadiacim a plávajúcim. Riadiace hradlo uväzní plávajúce ktoré nesie informáciu. Riadiace hradlo sa otvára pri čítaní alebo zapisovaní ak flash pamäť nie je aktívna riadiace hradlo je zatvorené. Plávajúce hradlo funguje podobne ako pri pamäti RAM (buď je nabité alebo nie). Samozrejme ak sa riadiace hradlo otvorí je napájané aj plávajúce hradlo, aby sa informácia nestratila.

Obr. 1.23 Flash pamäte (z ľavej strany do pravej: USB kľúč, pamäťová karta SD)

1.4.8 Vstupné periférie

Vstupné periférie sú zariadenia, ktoré slúžia na vysielanie vstupných informácií do počítača (CPU). Príkladmi vstupných zariadení sú klávesnica, myš, trackpoint, touchpad, joyistick, atď.

Klávesnica (Obr. 1.24) a myš je zatiaľ najrozšírenejšie vstupné zariadenie počítača. Jednotlivé typy klávesníc sa odlišujú svojou veľkosťou, tvarom a počtom klávesov pri notbookoch, usporiadaním a popisom klávesov pri rôznych jazykoch. Ďalšou odlišnosťou je rozhranie obvykle je klávesnica pripojená k základnej doske počítača prostredníctvom konektoru PS/2, dávnejšie (matičné dosky pred PC 486) sa používal päťkolíkový konektor DIN, taktiež sú bežné klávesnice pripojené pomocou USB. Ďalšie možnosti pripojenia klávesnice sú bezdrôtové a to pomocou infračerveného snímača, bluetooth, alebo WiFi. Pri každom stlačení klávesa je vyvolané prerušenie a následne prevedená obslužná rutina tohto prerušenia. Prerušeniam sa bude venovať kapitola 2.3. Klávesnica je obvykle konštruovaná ako sústava spínačov zapojená v matici. Zopnutie jednotlivých spínačov (t.j. stlačenie klávesa) vyhodnocuje radič v klávesnici tak, že postupne (v rýchlom slede) pripája určitú napäťovú úroveň na jednotlivé riadky matice, pričom sleduje stĺpce matice, či sa niekde daná napäťová úroveň objaví. Zapojenie obyčajnej matice spínačov pri niektorých kombináciách zopnutia viacerých spínačov nedovoľuje rozlíšiť, presne ktoré spínače boli zopnuté, preto sa spínače klávesov, ktoré sa typicky používajú v kombináciách s inými klávesmi (Shift, Ctrl, Alt) pripájajú osobitne, mimo hlavnej matice.

Obr. 1.24 Rolovacia klávesnica

Myš (Obr. 1.25) je polohovacie zariadenie určené k ovládaniu pozície kurzoru na obrazovke. Predchodcom dnešnej optickej myši bola guľôčková myš. Činnosť guľôčkovej myši bola založená na optomechanickom princípe snímania pohybu. Guľôčka umiestnená vo vnútri myši sa trením o podložku uvádzala do pohybu, pričom jej otáčanie snímala sústava dvoch valčekov. Pohyb oboch valčekov bol snímaný optoelektronickými snímačmi, ktorých signály boli následne vyhodnocované a prevádzané na informácie o rýchlosti a smere pohybu myši. Dnešné optické myši fungujú na inom princípe. Na mieste kde bola pôvodne guľôčka je teraz CCD (charge-coupled device) alebo CMOS (Complementary Metal Oxide Semiconductor) čip, teda čiernobiela kamera. Táto kamera má veľmi nízke rozlíšenie omnoho menej ako 1 kPix, ale vysokú obnovovaciu frekvenciu cez 1 kHz (klasické digitálne kamery majú obnovovaciu frekvenciu 25 Hz, výkonnejšie 200 Hz). Pri kamere je umiestnená červená LED, aby osvetľovala podložku. Nasnímané obrázky putujú do radiča (CPU) v myši, ktorý vyhodnotí pomocou dvoch po sebe idúcich obrázkoch rýchlosť a smer myši. Tieto informácie potom putujú do počítača. K počítaču sa obyčajne pripája prostredníctvom sériového rozhrania PS/2 alebo USB. V časoch keď sa klávesnice pripájali pomocou päťkolikového konektora DIN, tak sa myši pripájali do sériového rozhrania RS-232. Myš sa môže samozrejme pripojiť k počítaču aj bezdrôtovo pomocou infračerveného snímača, bluetooth, alebo WiFi. Pri pohybe myši (alebo stlačení tlačidla na myši) sa taktiež vyvolá prerušenie.

Obr. 1.25 Ergonomická myš pre hranie hier, alebo počítačovú grafiku od firmy Logitech

Trackpoint je „kolík“ na klávesnici, ktorý sa ručne natáča a spôsobuje tak pohyb kurzoru. Trackpoint Touchpad je dotyková doštička, ktorá umožňuje ovládať kurzor ľahkým pohybom prstu po doštičke. Existujú aj vstupné periférie vyhradené najmä na hranie hier ako sú joysticky (pákové ovládače) gamepady, volanty, atď. Ďalšími nespomenutými vstupnými zariadeniami sú napríklad skenery a kamery.

1.4.9 Výstupné periférie

Výstupné periférie sú zariadenia, ktoré slúžia na prijímanie výstupných informácií z počítača (CPU). Výstupnými perifériami sú napríklad monitor, tlačiareň, reproduktor, projektor, atď.

Monitor (Obr. 1.26) je najrozšírenejšie výstupné zariadenie. Slúži na vykresľovanie grafického prostredia OS a rôznych aplikácií. Dnes sa používajú tri typy monitorov a to LCD (liquid crystal display - displej s kvapalnými kryštálmi), LED (light-emitting diode - luminiscenčná dióda) a OLED (organic LED – organická LED). Starším typom monitorov je CRT. Monitory sa bližšie opíšu v kapitole 2.5.

Obr. 1.26 OLED monitor od firmy LG

Tlačiarne sú po monitoroch ďalším najviac rozšíreným výstupným zariadením. Čiernobiele tlačiarne na tlač využívajú iba čiernu farbu. Farebné tlačiarne používajú subtraktívny mechanizmus skladania (miešaním) farieb. Používa sa model CMY (Cyan, Magenta, Yellow – azúrová, purpurová, žltá). Pri tlačiarňach platí, že ak zmiešame všetky farby tak výsledkom bude čierna farba (opačne ako pri monitoroch, kde zmiešaním všetkých farieb dostávame bielu). Ostatné farby sa dosahujú pomerovým miešaním azúrovej, purpurovej a žltej. Tlačiarne delíme podľa princípu tlače na:

- ihličkové,
- atramentové,
- laserové,
- tepelné.

Ihličkové tlačiarne sa používajú aj dnes pri pokladniach, pri tlači faktúr, alebo pri tlači výplatných pások. Znaky na papieri sú vytvárané sériou úderov ihličiek tlačiacej hlavy pohybujúcej sa cez farbiacu pásku. Jednotlivé ihličky sú ovládané elektromagnetmi. Obvykle sú tlačiarne 9 a 24 ihličkové. Sú pomerne veľmi hlučné. Kvalita tlače závisí na opotrebovaní farbiacej pásky. Hustota tlače je obyčajne 100 až 300 DPI. Ihličkové tlačiarne majú nízke prevádzkové náklady. Táto tlačiareň je ako jediná zo spomenutých typov úderová to je hlavný dôvod prečo sa ešte používa. V pokladniach sú výhodou nízke prevádzkovateľné náklady. Pri faktúrach je výhodou, že sa pomocou kopiráka môžu tlačiť naraz viaceré kópie faktúr. Pri výplatných páskach sa nepoužíva farbiaca páska len sa tlač pretlačí do vnútra výplatnej pásky, takže ani zamestnanec ktorý tlačí výplatné pásky nevidí výplaty svojich kolegov.

V atramentových tlačiarňach vzniká znak podobne ako v ihličkových tlačiarňach, len namiesto ihličiek tlačiacej hlavy dopadajú na papier kvapky rýchloschnúceho atramentu. Rýchlosť tlače je porovnateľná s ihličkovými tlačiarňami. Kvalita tlače je porovnateľná s laserovými tlačiarňami. Prevádzka je skoro bezhlučná, hlučnejšie je preberanie papiera a jeho posúvanie než samotná tlač (posúvanie tlačiacej hlavy). Kvalita tlače závisí na kvalite použitého papieru (najlepšia kvalita farebnej tlače obrázkov je na fotografickom papieri). Po dlhšej dobe nepoužívania môže atramentová náplň zaschnúť. Prevádzkové náklady sú vyššie ako u ihličkovej tlačiarni. Cena tlačiarní je nízka. Najznámejšie sú 2 typy technológie atramentovej tlače *bubble jet* a *piezo*.

Pri technológii *bubble jet* sa tryskové komôrky tlačiacej hlavy plnia automaticky kapilárnymi silami atramentom o objeme približne 10 pl. Keď sa má tlačiť, zapne sa na krátku dobu (asi 2μs) topné teliesko, ktoré zahreje atrament na teplotu zhruba 300 °C a vznikajúca parná bublinka vytláča atrament z tela trysky. Atrament nakoniec opustí trysku vo forme malej kvapky rýchlosťou približne 100 km/h a dopadá na list papieru. Tlačiarne s touto technológiou sú lacnejšie než s technológiou piezzo. Prvý krát sa tieto tlačiarne objavili v roku 1985 (Hewlett Packard – Thinkjet).

Pri technológii *piezzo* sa k vystreleniu kvapky atramentu používa piezoelektrický menič („doštička, ktorá sa po priložení elektrického napätia deformuje – prehne“). Deformáciou piezoelektrického meniča vznikajú v kanáliku s atramentom tlakové vlny, ktoré vystreľujú kvapky atramentu. Výhodou oproti bubble jet je to, že elektrické napätie je priamo prevádzané na mechanický pohyb (vyššia rýchlosť). Prvá takáto tlačiareň sa vyrobila v roku 1977 (Siemens – PT 80i).

Laserová tlačiareň využíva fotoelektrické vlastností polovodičov (selénu), ktorý je nanesený na kovovom tlačiarenskom (fotocitlivom) valci. Neosvetlený selén sa chová ako izolátor, a preto je možné povrch fotocitlivého valcu nabiť elektrostatickým nábojom. Fotocitlivý valec je najprv nabitý. Potom sa na nabitý povrch fotocitlivého valca laserovým lúčom nakreslí obraz, ktorý má byť vytlačený. Miesta, ktoré boli laserom osvetlené, sa vybijú. Na povrchu valca tak vznikne skrytý (latentný) obraz, ktorý je nutné v ďalších krokoch zviditeľniť a preniesť na tlačiarenské médium. Zviditeľnenie obrazu sa robí nanesením farbiaceho prášku (toner), ktorý sa prichytí na valci len na vybitých miestach. Toner má obyčajne rovnaký náboj ako povrch valce, preto tam, kde valec zostal nabitý sa toner neuchytí. Z valca sa toner prenesie na papier, na ktorom je tepelne fixovaný (zažehlený) prechodom papiera cez vyhrievané valce. Laserové tlačiarne majú vysokú kvalitu tlače a vysokú rýchlosť tlače (6 až 40 strán za minútu, podľa typu tlačiarne). Tlačiarne majú takmer bezhlučnú prevádzku, ale vyššiu nákupnú a prevádzkovú cenu. Mal by sa používať xerografický papier z dôvodu menšieho opotrebenia tlačiarenského valca a vyššej kvality tlače. Pri prevádzke laserovej tlačiarne vzniká zdraviu škodlivý ozón, ktorý spôsobuje dráždenie sliznice, kašeľ a bolesti hlavy, preto bývajú moderné laserové tlačiarne vybavené ozónovým filtrom.

Tepelné tlačiarne vyvolávajú teplom chemickú reakciu, ktorá spôsobí zmenu farby papieru. Pri týchto tlačiarňach je nutné používať špeciálny teplocitlivý papier (ako napr. pri faxe). Tlačiarne majú nízku kvalitu, rýchlosť aj trvanlivosť tlače. Majú takmer bezhlučnú prevádzku. Používajú sa pri niektorých typoch pokladní, alebo sú tlačiarňami niektorých laboratórnych prístrojov.

Reproduktory môžu byť buď pasívne alebo aktívne. *Pasívne* reproduktory obsahujú iba samostatné reproduktory (bez zosilovača) a nevyžadujú externé napájanie. *Aktívne* reproduktory obsahujú zosilovač spoločne s ďalšími obvodmi (napr. pre reguláciu hĺbok, výšok atď.), vyžadujú externé napájanie, sú kvalitnejšie.

1.4.10 Vstupno-výstupné periférie

Vstupno-výstupné periférie sú zariadenia, ktoré slúžia aj na vysielanie informácií do procesora aj na príjem informácií z procesora. Takýmito zariadeniami sú napríklad dotykové panely a dotykové obrazovky. Zjednodušene povedané sú to monitory na ktoré sa dá klikať prstom. Technológia vykresľovania obrazu bude vysvetlená v podkapitole 2.5. Čo sa týka rozpoznania polohy prsta tak sa používajú tieto prístupy:

- optický,
- rezistívny,
- kapacitný,
- ultrazvukový s povrchovou vlnou.

Optická dotyková obrazovka má v ráme displeja zabudovaný rad LED a oproti nim rad fotodetektorov. Niekedy je vytvorená matica pridaním podobnej sústavy aj v druhom smere. Výhodou optickej dotykovej obrazovky je, že na „dotyk“ sa môže použiť akýkoľvek nepriehľadný predmet. Nevýhodou je malé rozlíšenie. Relatívnou nevýhodou je aj to, že sa neregistruje dotyk ale prerušenie lúča, čo nastane skôr než sa prst dotkne povrchu displeja (k dotyku ani nemusí dôjsť) a to vyvoláva pocit neistého ovládania. Optická dotyková obrazovka je dnes už zriedkavo používaná.

Rezistívnu dotykovú obrazovku tvoria dve transparentné fólie s priehľadnými rezistívnymi elektródami, umiestnené navzájom rovnobežne s malou medzerou. Dotyk spôsobí skrat medzi elektródami, vyhodnotením pomeru odporov medzi jednotlivými rohmi elektród sa určí miesto dotyku. Rezistívna dotyková obrazovka je relatívne lacná a široko používaná, ale má menšiu odolnosť a životnosť ako ostatné technológie.

Kapacitná dotyková obrazovka je tvorená priehľadnou rezistívnou elektródou prikrytou odolnou vrstvou (sklo, plastová fólia). V niekoľkých bodoch sa elektróda napája malým striedavým napätím. Pri dotyku prstom sa kapacitne zvedie časť signálu, vyhodnotením pomeru prúdov z jednotlivých bodov je možné určiť bod dotyku. Kapacitné dotykové obrazovky sa vyznačujú vysokou odolnosťou voči opotrebovaniu a zničeniu, nie sú však vhodné do prostredí so silným elektromagnetickým rušením, problematická je aj prítomnosť kovových predmetov v ich blízkosti, a nie je možné ich používať pre dotyk nevodivými predmetmi (problém predstavujú niekedy aj rukavice).

Dotyková obrazovka s povrchovou vlnou je označovaná aj SAW (Surface Acoustic Wave). Ultrazvukový transducer umiestnený na povrchu sklenenej platne vyvolá povrchovú vlnu, ktorá sa vďaka sústave odrazových plôch šíri cez celý povrch snímača až k prijímaciemu transduceru. Dotyk prstom utlmí vlnu, čo sa prejaví ako výpadok v prijímanom signály, z času v ktorom nastal výpadok a z rýchlosti šírenia povrchovej vlny je možné určiť bod dotyku. Tieto dotykové obrazovky sú citlivé na poškriabanie a znečistenie, sú však pomerne presné, niektoré dokážu dokonca snímať silu tlaku prsta a vytvoriť tak „tretiu súradnicu“.

1.5 Architektúra počítačov s procesorom Intel

V tejto podkapitole si rozoberieme počítače s procesorom Intel, pretože procesory od tejto firmy boli stále vo vývoji v popredí pred ostatnými firmami a taktiež s firmou IBM štandardizovali IBM PC, tieto počítače boli prvé personálne počítače dnešnej doby (rovnaká veľkosť, rovnaký spôsob skladania a rovnaký základný hardvér).

Procesory 4. generácie počítačov od firmy Intel:

4-bitové procesory:
- Intel 4004 (1971, taktovacia frekvencia 740 kHz),
- Intel 4040 (1974, taktovacia frekvencia 740 kHz),

8-bitové procesory:
- Intel 8008 (1972, taktovacia frekvencia 500 kHz),
- Intel 8080 (1974, taktovacia frekvencia 2 MHz),
- Intel 8085 (1976, taktovacia frekvencia 3 MHz),

Jednočipové mikropočítače:
- Intel 8048 (1976, taktovacia frekvencia 11 MHz),
- Intel 8051 (1980, taktovacia frekvencia 12 MHz),
- Intel 80151,
- Intel 80251 (1996, taktovacia frekvencia 12 MHz),
- Intel 8096,
- Intel 80196,
- Intel 80296,

16-bitové procesory (architektúra x86):
- 8086 (1978, taktovacia frekvencia 10 MHz),
- **8088** (1979, taktovacia frekvencia 8 MHz),
- 80186 (1982, taktovacia frekvencia 6 MHz),
- 80188,
- **80286,**

32-bitové procesory (mimo architektúry x86):
- iAPX 432,
- i960 aka 80960,
- i860 aka 80860,
- XScale,

32-bitové procesory (architektúra x86):
- Procesory 386: **80386DX**, **80386SX**, 80386SL, 80386EX,
- **Procesory 486**: 80486DX, 80486SX, 80486DX2, 80486SL, 80486DX4,
- **Pentium**, Pentium s technológiou MMX,
- Pentium Pro, **Pentium II**, Celeron (Pentium II-based),
- **Pentium III**, Pentium II and III Xeon, Celeron (Pentium III Coppermine-based), Celeron (Pentium III Tualatin-based), Pentium M, Celeron M, Intel Core,
- **Pentium 4**, Xeon, Mobile Pentium 4-M, Pentium 4 EE, Pentium 4E, Pentium 4F,

64-bitové jednojadrové procesory:
- Itanium,
- Itanium 2,
- **Pentium 4F,**

64-bitové viac jadrové procesory:
- Pentium D,

- Pentium Extreme Edition,
- Xeon,
- **Dual-Core,**
- **Core2Duo,**
- **Core2Quad,**
- **Core i3,**
- **Core i5,**
- **Core i7.**

V ďalšej časti sa podrobne rozoberie architektúra x86 a x64. Počítače architektúry x86 sa najviac podobajú dnešným, pretože v šasi (skrinke) počítača obsahovali tie isté základné hardvérové súčasti ako dnešné počítače a servery (CPU, MB, RAM, HDD, atď.) a taktiež skrinky (šasi) a matičné dosky mali podobnú veľkosť.

1.5.1 PC – XT

Rok: 1978

Zbernice:
- adresná zbernica: 20 bitov,
- dátová zbernica: 16 bitov, ale dátový prenos prebieha iba na 8 bitoch,
- vstupno-výstupný kanál je realizovaný pomocou 8 bitovej zbernice PC-Bus so 62 pinmi,

Procesor:
- procesor 8088 + koprocesor 8087,
- taktovacia frekvencia 5 MHz a 8 MHz,
- 29 000 tranzistorov s veľkosťou 3 μm,

Radiče:
- prerušovací podsystém je realizovaný pomocou adaptéra 8259A: 8 prerušení,
- priamy prístup do pamäte (direct memory access - DMA) je realizovaný pomocou adaptéra 8237A: 4 DMA kanály,
- čítač/časovač realizovaný pomocou 8253A: 3 kanály čítača/časovača,
- vstup z klávesnice je realizovaný radičom Intel 8049,
- adaptér reproduktorov,

Ďalšie špecifikácie:
- pamäť RAM: 640 kiB,
- pamäť ROM: 48kiB,
- adresovateľný priestor: 2^{20} => 1 MiB,
- HDD: 5 MB až 26 MB (najbežnejší 5 MB)
- vyvinutý OS DOS.

1.5.2 PC – AT

Rok: 1982

Zbernice:

- adresná zbernica: 24 bitov, dátová zbernica: 16 bitov,
- vstupno-výstupný kanál realizovaný pomocou 16 bitovej zbernice ISA s 98 pinmi,

Procesor:

- procesor 80286 s možnosťou pripojiť matematický koprocesor,
- taktovacia frekvencia od 6 MHz do 25 MHz,
- 134 000 tranzistorov s veľkosťou 1,5 μm,

Radiče:

- prerušovací podsystém je realizovaný pomocou dvoch adaptérov 8259A: 15 prerušení,
- priamy prístup do pamäte (DMA) je realizovaný pomocou dvoch adaptérov 8237A: 7 kanálov DMA,
- čítač/časovač realizovaný pomocou 8254,
- vstup z klávesnice je realizovaný pomocou radiča Intel 8042,
- adaptér repro, hodinový obvod MC 14818,

Ďalšie špecifikácie:

- pamäť RAM: od 1 MiB do 16 MiB, najbežnejšie bola pamäť typu SIPP,
- pamäť ROM: 128 kiB,
- adresovateľný priestor: 2^{24} => 16 MiB,
- HDD: 5 MB až 55 MB (najbežnejší 10 MB),
- OS DOS bol vyvinutý pre 640kiB (adresovanie 1 MiB), takže bola nutná zmena na 16MB.

1.5.3 PC – 386/SX

Rok: 1988

Zbernice:

- adresná zbernica: 32 bitov,
- dátová zbernica: 16 bitov, ale logický funguje ako 32 bitová,
- vstupno-výstupný kanál realizovaný pomocou 16 bitovej zbernice ISA,

Procesor:

- procesor s možnosťou použitia matematického koprocesora,
- taktovacia frekvencia od 16 MHz do 33 MHz,
- 275 000 tranzistorov s veľkosťou 1 μm,

Radiče a špecifikácie:

- 15 prerušení,

- 7 kanálov DMA,
- vstupno-výstupný kanál je realizovaný pomocou zbernice ISA,
- adresovateľný priestor: 2^{32} => 4 GiB,
- pamäť RAM: najbežnejšie MB mali sloty pre 30 pinové pamäte SIMM a obsahovali 2 až 8 slotov,
- HDD: 5 MB až 250 MB (najbežnejší 20 MB).

1.5.4 PC – 386/DX

Rok: 1989 (1985)

Zbernice:

- adresná zbernica: 32 bitov,
- dátová zbernica: 32 bitov,
- vstupno-výstupný kanál realizovaný pomocou 32 bitovej zbernice EISA,

Procesor:

- procesor s možnosťou použitia matematického koprocesora,
- taktovacia frekvencia od 16 MHz do 33 MHz,
- 275 000 tranzistorov s veľkosťou 1 μm,

Radiče a špecifikácia:

- 15 prerušení IRQ
- 7 kanálov DMA
- vstupno-výstupný kanál je realizovaný pomocou zbernice EISA, táto zbernica má viac pinov ako ISA, ale je kompatibilná s ISA
- adresovateľný priestor 4 GB
- pamäť RAM: najbežnejšie MB mali sloty pre 30 pinové pamäte SIMM a obsahovali 2 až 8 slotov,
- HDD: 5 MB až 250 MB (najbežnejší 20 MB).

1.5.5 PC – 486

Rok: 1989

Zbernice:

- adresná a dátová zbernica: 32 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice EISA, neskôr PCI,

Procesor:

- procesor obsahuje integrovaný matematický koprocesor,
- taktovacia frekvencia od 25 MHz do 100 MHz,
- 1 600 000 tranzistorov s veľkosťou 0,6 μm,
- procesor obsahuje už aj pamäť CACHE (8kB),

Radiče a špecifikácia:

- 15 prerušení IRQ a 7 DMA kanálov,
- adresovateľný priestor 4 GB,
- počítač na porovnanie vykonal 50x rýchlejšie tú istú inštrukciu ako PC-XT,
- pamäť RAM: matičné dosky mali sloty pre 72 pinové pamäte SIMM a obsahovali 2 až 8 slotov,
- HDD: 10 MB až 1 GB (najbežnejší 40 MB).

1.5.6 Počítač s procesorom Pentium

Rok: 1993

Zbernice:

- adresná a dátová zbernica: 32 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a EISA,
- 66 MHz frekvencia zberníc,

Procesor:

- RISC – reduced instruction set computer (architektúra superscalar), neskôr MMX technológia,
- taktovacia frekvencia od 60 MHz do 233 MHz,
- 4 500 000 tranzistorov s veľkosťou 350 nm,
- rozšírená pamäť CACHE na dátovú a inštrukčnú po 8 kB neskôr 16 kB,

Radiče a špecifikácia:

- 24 prerušení IRQ a 7 DMA kanálov,
- pamäť RAM: matičné dosky obsahovali 72 pinovú SIMM, po 2 až 8 slotov,
- HDD: 40 MB až 3200 MB (najbežnejší 100 MB).

1.5.7 Počítač s procesorom Pentium II

Rok: 1997 (1995)

Zbernice:

- adresná a dátová zbernica: 32 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI,
- 66 MHz frekvencia zberníc,
- zbernica grafiky AGP,

Procesor:

- zavedenie externej CACHE pamäte L2 (512kB), pôvodná dostala názov L1 (32kB),
- 7 500 000 tranzistorov s veľkosťou 250 nm,
- taktovacia frekvencia od 233 MHz do 500 MHz,

Radiče a špecifikácie:

- 24 prerušení IRQ
- pamäť RAM: SDRAM PC66 (priepustnosť 533 MB/s) po dvoch až štyroch slotoch, niektoré typy starších matičných dosiek ešte používali 72 pinovú SIMM,
- HDD: 240 MB až 25 GB (najbežnejší 4 GB).

1.5.8 Počítač s procesorom Pentium III

Rok: 1999

Zbernice:

- adresná a dátová zbernica: 32 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI,
- frekvencia zberníc: 100 MHz, 133 MHz a 150 MHz (podľa varianty),
- zbernica grafiky AGP,

Procesor:

- pamäť CASH L2 sa integrovala do procesora,
- nová inštrukčná sada SIMD,
- 28 100 000 tranzistorov s veľkosťou 130 nm,
- taktovacia frekvencia od 0,5 GHz do 1,4 GHz,

Radiče a špecifikácie:

- 24 prerušení IRQ,
- pamäť RAM: SDRAM PC100, PC133 a PC150 (priepustnosť 800,1066, 1200MB/s), 2 až 4 sloty,
- HDD: 2 GB až 25 GB (najbežnejší 6,4 GB), tieto MB sa používali aj neskôr preto je bežné, že tieto počítače majú kapacitu 40 GB a môžu mať maximálne 500 GB.

1.5.9 Počítač s procesorom Pentium 4

Rok: 2000

Zbernice:

- adresná a dátová zbernica: 32 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI,
- maximálna frekvencia zberníc: 800 MHz,
- zbernica grafiky AGP,

Procesor:

- taktovacia frekvencia od 1,4 GHz do 3,4 GHz,
- 55 000 000 tranzistorov s veľkosťou 130 nm,

Radiče a špecifikácie:

- prerušenia IRQ fungujú na báze *message signals* pomocou signálu SERIRQ,
- pamäť RAM: DDR SDRAM (priepustnosť od 1,6 GB/s do 4,8 GB/s podľa typu),
- HDD: 16 GB až 500 GB.

1.5.10 Počítač s procesorom Pentium 4F

Rok: 2005

Zbernice:

- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI,

- maximálna frekvencia zberníc: 800 MHz,
- zbernica grafiky AGP,

Procesor:
- taktovacia frekvencia od 2,8 GHz do 3,8 GHz,
- 125 000 000 tranzistorov s veľkosťou 65 nm,
- rozšírená pamäť CASHE L1 na 64 kB a L2 na 2 MB,
- zavedená technológia EIST (Enhanced Intel SpeedStep Technology),

Radiče a špecifikácie:
- prerušenia IRQ fungujú na báze *message signals* pomocou signálu SERIRQ,
- pamäť RAM: DDR2 SDRAM (priepustnosť od 3,2 GB/s do 9,6 GB/s podľa typu),
- zavedie SATA,
- HDD: 160 GB až 6 TB (k roku 2014).

1.5.11 Počítač s procesorom Intel Dual-Core

Rok: 2007

Zbernice:
- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a PCI-express (PCI-e),
- frekvencia zberníc: 800 MHz a 1066 MHz,
- zbernica grafiky PCI-e, alebo AGP,

Procesor:
- 2 jadrá,
- taktovacia frekvencia oboch procesorov samostatne od 1,6 GHz do 3,3 GHz,
- spolu 167 000 000 tranzistorov s veľkosťou 65 nm, alebo 45 nm,

Radiče a špecifikácie:
- prerušenia IRQ fungujú na báze *message signals* pomocou signálu SERIRQ,
- pamäť RAM: DDR3 SDRAM (priepustnosť od 6,4 GB/s do 8,5 GB/s podľa typu),
- adresovateľný priestor: 2^{36} => 64GiB,
- HDD: 250 GB až 6 TB (k roku 2014).

1.5.12 Počítač s procesorom Intel Core2Duo

Rok: 2007 (prvý CPU bol predstavený už v roku 2006)

Zbernice:
- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a PCI-express (PCI-e),
- frekvencia zberníc od 800 MHz do 1600 MHz,
- zbernica grafiky PCI-e, alebo AGP,

Procesor:
- 2 jadrá,
- taktovacia frekvencia oboch procesorov samostatne od 1,8 GHz do 3,3 GHz,
- spolu 167 mil. – 410 mil. tranzistorov s veľkosťou 65 nm, alebo 45 nm,
- rozšírená pamäť CASHE L2 na 6MB (2 x 3 MB),

Radiče a špecifikácie:
- prerušenia IRQ fungujú na báze *message signals* pomocou signálu SERIRQ
- pamäť RAM: DDR3 SDRAM (priepustnosť od 6,4 GB/s do 12,8 GB/s podľa typu)
- HDD: 250 GB až 6 TB (k roku 2014).

1.5.13 Počítač s procesorom Intel Core2Quad

Rok: 2007

Zbernice:
- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a PCI-express (PCI-e),
- frekvencia zberníc od 800 MHz do 1600 MHz,
- zbernica grafiky PCI-e, alebo AGP,

Procesor:
- 4 jadrá,
- taktovacia frekvencia oboch procesorov samostatne od 2,3 GHz do 3,2 GHz,
- spolu 586 mil. – 820 mil. tranzistorov s veľkosťou 65 nm, alebo 45 nm,
- rozšírená pamäť CASHE L2 na 12MB (4 x 3 MB),

Radiče a špecifikácie:
- prerušenia IRQ fungujú na báze *message signals* pomocou signálu SERIRQ,
- pamäť RAM: DDR3 SDRAM (priepustnosť od 6,4 GB/s do 12,8 GB/s podľa typu),
- HDD: 250 GB až 6 TB (k roku 2014).

1.5.14 Počítač s procesorom Intel Core i3

Rok: 2010

Zbernice:
- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a PCI-express (PCI-e),
- frekvencia zberníc od 800 MHz do 1600 MHz,
- zbernica grafiky PCI-e,

Procesor:
- 2 jadrá a 4 vlákna,
- taktovacia frekvencia oboch procesorov samostatne od 2,4 GHz do 3,7 GHz,
- spolu 624 mil. tranzistorov s veľkosťou 32 nm, alebo 22 nm,
- zavedenie a integrovanie CASHE pamäte L3 (L1 64 kB, L2 512 kB, L3 4 MB),
- integrovanie HD GPU,

Špecifikácie:

- pamäť RAM: DDR3 SDRAM (priepustnosť od 6,4 GB/s do 12,8 GB/s podľa typu),
- HDD: 250 GB až 6 TB (k roku 2014).

1.5.15 Počítač s procesorom Intel Core i5

Rok: 2010 (prvý CPU bol predstavený už v roku 2009)

Zbernice:

- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a PCI-express (PCI-e),
- frekvencia zberníc od 800 MHz do 1600 MHz,
- zbernica grafiky PCI-e,

Procesor:

- 2 alebo 4 jadrá a 4 vlákna,
- taktovacia frekvencia oboch procesorov samostatne od 2 GHz do 3,9 GHz,
- spolu 995 mil. tranzistorov s veľkosťou 32 nm, alebo 22 nm,
- CACHE L3 zvýšená na 6 MB,

Špecifikácie:

- pamäť RAM: DDR3 SDRAM (priepustnosť od 6,4 GB/s do 12,8 GB/s podľa typu),
- HDD: 250 GB až 6 TB (k roku 2014).

1.5.16 Počítač s procesorom Intel Core i7

Rok: 2010 (prvý CPU bol predstavený už v roku 2008)

Zbernice:

- adresná, dátová zbernica: 64 bitov, adresná však využíva iba 36 bitov,
- vstupno-výstupný kanál realizovaný pomocou zbernice PCI a PCI-express (PCI-e),
- frekvencia zberníc od 800 MHz do 1600 MHz,
- zbernica grafiky PCI-e,

Procesor:

- 4 jadrá a 8 vlákien (novšie verzie 6 jadier / 12 vlákien a 8 jadier / 16 vlákien),
- taktovacia frekvencia oboch procesorov samostatne od 2 GHz do 4,4 GHz
- spolu 1,4 mld. tranzistorov s veľkosťou 32 nm, alebo 22 nm
- CACHE L3 zvýšená pri najnovšej verzií na 20 MB (k roku 2014),

Špecifikácie:

- pamäť RAM: DDR3 SDRAM (priepustnosť od 6,4 GB/s do 12,8 GB/s podľa typu),
    - 6 a 8 jadrové CPU používajú DDR4 SDRAM (17 GB/s),
- HDD: 250 GB až 6 TB (k roku 2014).

Najnovšia technológia Intelu vychádza z tranzistorov o veľkosti 14 nm, najväčším konkurentom spoločnosti Intel je firma AMD. Procesory AMD:

- **Am286:** 1987, 8 - 20 MHz, 1,5 μm,
- **Am386:** 1991, 20 - 40 MHz, 0,8 μm,
- **Am486:** 1993, 25 - 40 MHz, L1 8kB, 0,6 μm,
- **Am5x86:** 1995, 133 - 150 MHz (zbernica: 25 - 33 MHz), L1 16 kB, 350 nm,
- **AMD K5:** 1996, 75 - 133 MHz (zbernica: 50 - 66 MHz), L1 8 + 16 kB, 350nm,
- **AMD K6:** 1997, 166 - 550 MHz (zbernica: 66 - 100 MHz), L1 32 + 32 kB, 250nm,
- **AMD K7:** 1999, 0,5 - 2,3 GHz (zbernica: 0,2 - 0,4 GHz), L1 64 + 64 kB, L2 512 kB, 250nm a 130 nm (Athlon, Duron, Sempron),
- **AMD K8:** 2003, 1,6 - 3,2 GHz (zbernica: 0,8 - 1 GHz), L1 64 + 64 kB, L2 1 MB, 130 nm a 90 nm (Opteron, Sempron, Athlon 64, Duron 64, Turion 64),
- **AMD K10:** 2007, 1,7 - 3,7 GHz (zbernica 1 - 2 GHz), L1 po 64 + 64 kB na jadro, L2 po 1 MB na jadro, L3 6 MB pre všetky jadrá, 65 a 45 nm (Phenom x4, …),
- 2011 – vkladanie GPU do jedného čipu s CPU.

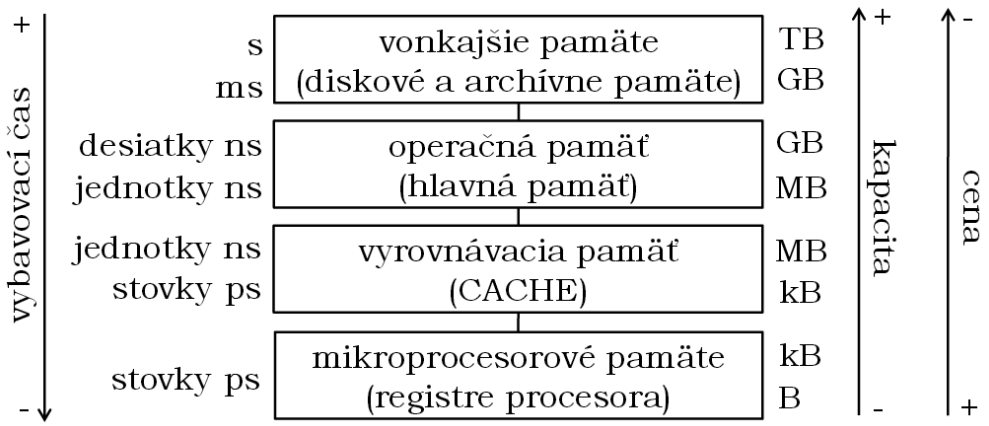
2 Základné podsystémy počítačov

Táto kapitola bližšie priblíži základné podsystémy počítačov, ktorými sú pamäťový podsystém, vstupno-výstupný podsystém, prerušovací podsystém, podsystém priameho prístupu do pamäte a obrazový podsystém.

2.1 Pamäťový podsystém počítačov

Pamäťový podsystém slúži na uloženie programu a údajov, ktoré sa práve používajú, ako aj na ich archiváciu.

Pamäťové prostriedky počítača je možné rozdeliť do niekoľkých úrovní, ktoré sa líšia spôsobom použitia v procese spracovania informácií. Jednotlivé úrovne majú rozdielnu kapacitu a operačnú rýchlosť. Pamäťový podsystém počítača má hierarchickú organizáciu, ktorú môžete vidieť na obrázku Obr. 2.1.



Obr. 2.1 Hierarchická organizácia pamäťového podsystému

Registre procesora sa nachádzajú na čipe procesora. Slúžia na prechodné uchovanie informácií počas ich spracovania v procesore. Registre sú najrýchlejšie zo všetkých častí pamäťového podsystému – majú najkratšiu dobu prístupu. Doba prístupu je čas, ktorý uplynie od nastavenia požiadavky na pamäť (register) po poskytnutie požadovaného údaja (stovky pikosekúnd).

Vyrovnávacia pamäť (CACHE) slúži na preklenutie rádového rozdielu medzi prístupovou dobou registrov procesora a hlavnej pamäte. Vyrovnávacia pamäť je rýchla pamäť, rádovo menšej kapacity ako hlavná pamäť, umiestnená medzi procesor a hlavnú pamäť. Do vyrovnávacej pamäte sa presunie časť obsahu hlavnej pamäte a procesor sprístupňuje informácie z vyrovnávacej pamäte vyššou rýchlosťou. Vyrovnávacia pamäť riadi vlastný riadiaci obvod, ktorý zabezpečuje:

- presun požadovaných informácií z hlavnej pamäte do vyrovnávacej pamäte,
- rieši situáciu, keď CPU modifikoval údaj vo vyrovnávacej pamäti a túto zmenu je nutné vykonať aj v hlavnej pamäti.

Operačná pamäť obsahuje práve vykonávaný program a spracúvané údaje (nedosahuje rýchlosť registrov). Kapacitou pamäte sa rozumie množstvo informácií, ktoré je schopná pamäť uchovať. Hlavná pamäť je realizovaná ako samostatný funkčný blok, ktorý

sa skladá z jedného alebo niekoľkých integrovaných obvodov. Je vzdialená od CPU takže doba prístupu je dlhšia. Hlavná pamäť je realizovaná dynamickými pamäťami, ktoré majú pri nízkej spotrebe energie veľkú kapacitu. Doba prístupu hlavnej pamäte sú rádovo desiatky ns až jednotky ns.

Vonkajšie pamäte slúžia na uchovanie informácií, ktoré sa momentálne nepoužívajú a na archiváciu informácií. Na rozdiel od hlavnej pamäte, k vonkajším pamätiam CPU pristupuje ako ku V/V zariadeniam (HDD, flash pamäte, CD-ROM, diskety,...). Doba prístupu je rôzna. Záleží od typu vonkajších pamätí (pevné disky majú dobu prístupu jednotky ms a kapacitu od stoviek GB po TB).

2.1.1 Rozdelenie pamätí

Podľa spôsobu prístupu k informáciám:

a) pamäte s náhodným prístupom (RAM – Random Access Memory) – doba prístupu pre jednotlivé bunky je rovnaká, nezáleží od ich umiestnenia v pamäti (hlavná pamäť von Neumannovského typu).
b) pamäte so sekvenčným prístupom – adresované miesto sa sprístupní až po prehľadaní predošlých buniek. Doba prístupu záleží od umiestnenia adresovanej bunky v pamäti (napr. disková pamäť).
c) pamäte adresované obsahom (CAM – Content Access Memory) – sprístupnenie pamäťového miesta sa pri asociatívnej pamäti uskutočňuje nie na základe adresy, ale porovnaním všetkých buniek s tzv. výberovým kľúčom, (data – flow počítače).

Podľa možnosti čítania a zápisu:

a) pamäte pre čítanie a zápis (RWM – read/write memory) do týchto pamätí sa informácia dá v priebehu činnosti kedykoľvek zapísať a kedykoľvek čítať, napr. hlavná pamäť počítača, pri hlavnej pamäti počítača sa obsah pamäte po vypnutí napájacieho napätia vymaže.
b) pamäte iba pre čítanie (ROM – read only memory) z pamäte ROM sa informácia dá iba čítať, prvotný zápis informácie sa vykoná pri výrobe pamäte, alebo si ju naprogramuje užívateľ. Pamäte ROM uchovávajú svoj obsah aj po vypnutí napätia.

2.1.2 Pamäte

Hlavná pamäť

Hlavná pamäť (operačná pamäť) obsahuje práve vykonávaný program a spracúvané údaje. Je to pamäť s náhodným prístupom (RAM) a skladá sa z dvoch častí, z ktorých jedna je typu ROM a druhá RWM. Z hľadiska fyzickej realizácie je hlavná pamäť vytvorená z polovodičových pamätí. Polovodičové pamäti typu RWM sú realizované na polovodičovom čipe a vyrábajú sa ako statické alebo dynamické. Viac informácií o hlavnej pamäti nájdete v podkapitole 1.4.3.

Polovodičové pamäte ROM

Svoj obsah si uchovávajú aj po vypnutí napájacieho napätia. Okrem polovodičových pamätí ROM, v ktorých je zápis informácie vykonaný už pri výrobe (technologickou maskou), existujú používateľom programovateľné pamäti ROM.

Pamäť typu PROM (Programmable ROM) je jedenkrát naprogramovateľná polovodičová pamäť ROM. Naprogramovanie je trvalé.

Pamäte typu EPROM (Erasable PROM) sú programovateľné ROM s možnosťou vymazania a opätovného naprogramovania. Pamäť sa maže pôsobením ultrafialového žiarenia na pamäťové elementy.

Pamäte typu EEPROM (Electrically Erasable PROM) sú elektricky mazateľné pamäte PROM.

Komunikácia procesora s pamäťou
Čítanie z pamäte:

Procesor musí nastaviť na adresnej zbernici platnú adresu pamäťovej bunky, z ktorej chce načítať informáciu a nastaviť aktívnu úroveň signálu pre čítanie z pamäte MEMR#. Signál musí byť v aktívnej úrovni dostatočne dlhý čas. Po istom čase (doba prístupu pamäte) pamäť vyšle na údajovú zbernicu platnú informáciu.
Zápis do pamäte:

Procesor musí nastaviť na adresnej zbernici adresu bunky pamäte, do ktorej chce zapísať a na údajovú zbernicu vyslať platné údaje. Potom môže nastaviť do aktívnej úrovne signál pre zápis do pamäte MEMW#.

Virtuálna pamäť

Princíp virtuálnej pamäte spočíva v tom, že nie celý program alebo všetky údaje sa nachádzajú naraz v hlavnej pamäti PC. Nachádza sa tam iba tá časť, ktorá sa práve používa, zvyšná časť programu alebo údajov je uložená vo vonkajšej pamäti, napr. na pevnom disku. Najpoužívanejší spôsob realizácie virtuálnej pamäte je segmentovanie a stránkovanie.

Pamäť Cache

Je to rýchla pamäť rádovo menšej kapacity ako má hlavná pamäť (typu RWM). Slúži na preklenutie v podstate rádového rozdielu medzi prístupovou dobou registrov procesora a hlavnej pamäte. Do vyrovnávacej pamäte sa presunie časť obsahu hlavnej pamäte a CPU sprístupňuje informácie z vyrovnávacej pamäte vyššou rýchlosťou ako z hlavnej pamäte. Pamäť typu CACHE je možné vytvoriť technickými, ale aj programovými prostriedkami.

Keď časť spracovávaných údajov z pamäťového média s dlhším vybavovacím časom udržuje v pamäti s kratším vybavovacím časom, hovoríme, že pomalšie pamäte „kešujeme“ v rýchlejšej pamäti. Technicky sa najčastejšie „kešuje“ operačná pamäť. Robí sa to špeciálnymi obvodmi - „cache radičmi“, ktoré majú pre tento účel pridané rýchle pamäte RWM. Programovými prostriedkami sa „kešuje“ disk. (Ovládače OS).

Pamäť CACHE sa delí na 3 typy (k roku 2014) podľa hierarchickej organizácie a to na L1, L2 a L3, pričom pamäť L1 je najrýchlejšia (vybavovací čas sa ráta v stovkách ps) s najmenšou kapacitou (8 kiB až 64 kiB) a pamäť L3 je najpomalšia (vybavovací čas sa ráta v jednotkách ns) s najväčšou kapacitou (4 MiB až 20 MiB). Pamäť CACHE sa zaviedla v počítačoch PC – 486 kde už frekvencia zbernice pripojenej k pamäti RAM a taktiež taktovacia frekvencie RAM bola nižšia ako taktovacia frekvencia CPU. V tých časoch sa pamäť L1 (8kiB) integrovala na čip CPU a pri drahých modeloch bola externe pripojená CACHE L2 k najbližšiemu bridge-u k CPU. Pamäť L2 (0,5 MiB) bola externe zavádzaná na každej matičnej doske až pri socketoch pre Pentium II a už Pentium III mal pamäť L2 integrovanú na čipe. Drahé modely matičných dosiek od socketov pre Pentium 4 mali externú CACHE L3 (cena matičnej dosky prevyšovala 1000 USD). Pri zavedení viacjadrových CPU malo každé jadro vlastné CACHE pamäte L1 a L2 a neskôr sa ku jadrám na čip CPU pridala spoločná CACHE pamäť L3.

Pamäť CMOS

Ide o pamäť s kapacitou minimálne 64B (v PC – AT). Je to pamäť RWM vyrobená špeciálnou technológiou CMOS, ktorá sa vyznačuje malou spotrebou energie. V tejto pamäti sú uchovávané základné informácie o konfigurácii počítača. Jej obsah je zálohovaný

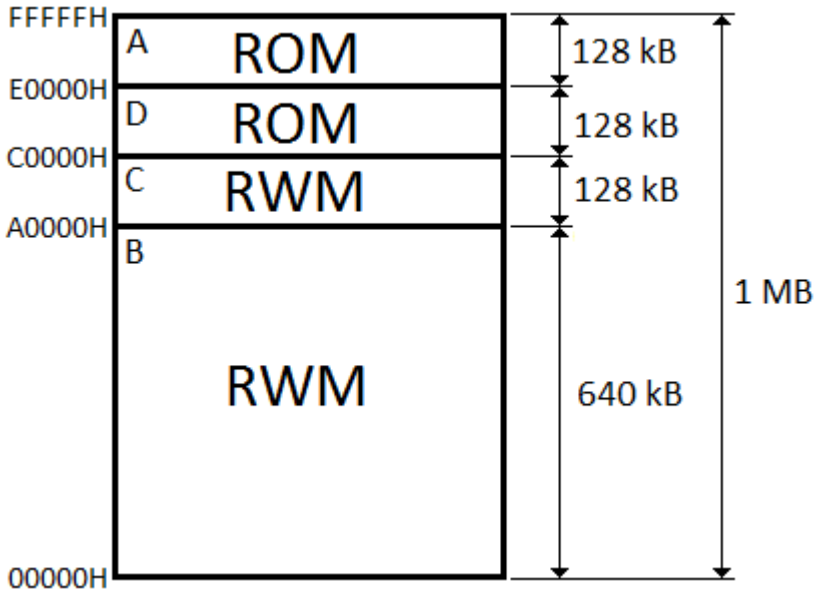
pomocou špeciálnej batérie a ostáva zálohovaný aj po vypnutí napájacieho zdroja napätia PC. Zmena obsahu tejto pamäte sa vykonáva napr. pri výmene batérie, alebo zmene konfigurácie počítača (špeciálnym programom SETUP). SETUP býva súčasťou BIOS-u a je uložený v ROM pamäti počítača.

Flash pamäť

Je špeciálna pamäť EEPROM, ktorá sa nachádza aj priamo na matičnej doske, kde je uložený BIOS.

2.1.3 Rozloženie adresovateľného priestoru pamäte

Ako sa dalo vyššie dočítať (kapitola 1.5.1) PC – XT vedel adresovať 1 MB pamäte pomocou 20 bitovej adresnej zbernice (2^{20} = 1 MB). Tento adresný priestor sa delil na 4 základné časti A, B, C, D (Obr. 2.2).



Obr. 2.2 Rozloženie adresovateľného priestoru pamäte v OS DOS

Pamäť A:

Vzhľadom na to, že po spustení alebo resetovaní začne CPU vykonávať inštrukciu uloženú na adrese FFFF0H, je táto časť pamäte typu ROM. Táto pamäť má 128 kiB, zahŕňa základný BIOS s tým, že pri prvých počítačoch v ňom bol aj programovací jazyk BASIC.

Pamäť B:

Programy vektora prerušení, ktoré majú byť modifikovateľné, procesor číta z adries 00000H až 003FFH preto je táto pamäť typu RWM. Operačný systém túto pamäťovú časť zahŕňa do súvislého bloku 640 kiB pamäti na adresách 0H až 09FFFH. Táto pamäť je určená pre OS a ukladanie dát bežiacich aplikácií.

Pamäť C:

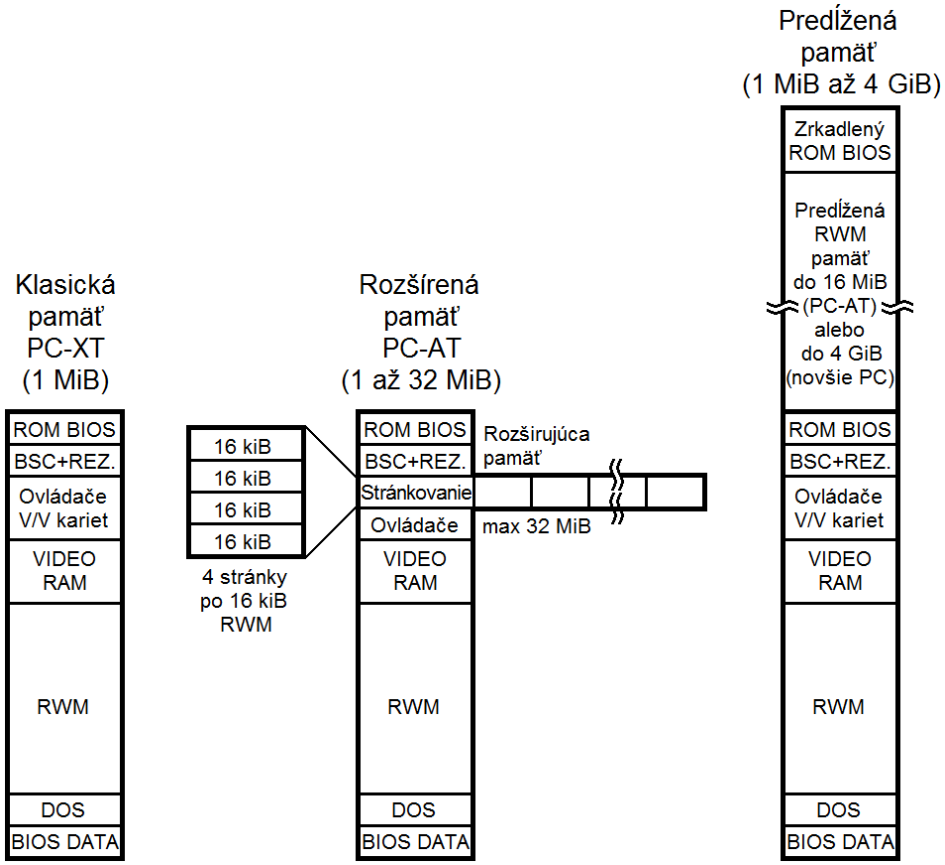
V tejto pamäti o veľkosti 128 kiB sú uložené informácie vykresľovaného obrazu na obrazovke (video-RAM), keďže vykresľovaný obraz sa neustále mení je táto pamäť typu RWM. Kapacita určená na 128 kiB predpokladá grafický režim obrazovky (kapitola 2.5).

Pamäť D:

Táto pamäť (128 kiB) obsahuje ovládače periférií pre BIOS

S postupným vývojom počítačov sa pamäťový priestor rozširoval, už PC – AT mal adresnú zbernicu so šírkou 24 bitov to znamenalo, že PC – AT dokázal zaadresovať 16 MiB pamäte (2^{24} = 16 MiB). Rozšírenie adresnej zbernice sa prejavilo v samotnom adresnom priestore. Pre zachovanie kompatibility medzi PC a OS bolo nutné vylepšiť DOS. Pre zväčšenie adresovateľného priestoru sa vyvinuli 2 spôsoby:

- rozšírená pamäť (EMS – Expanded Memory Specification),
- predĺžená pamäť (XMS – eXtended Memory Specification).



Obr. 2.3 Vývoj rozloženia adresovateľného priestoru pamäte v OS DOS

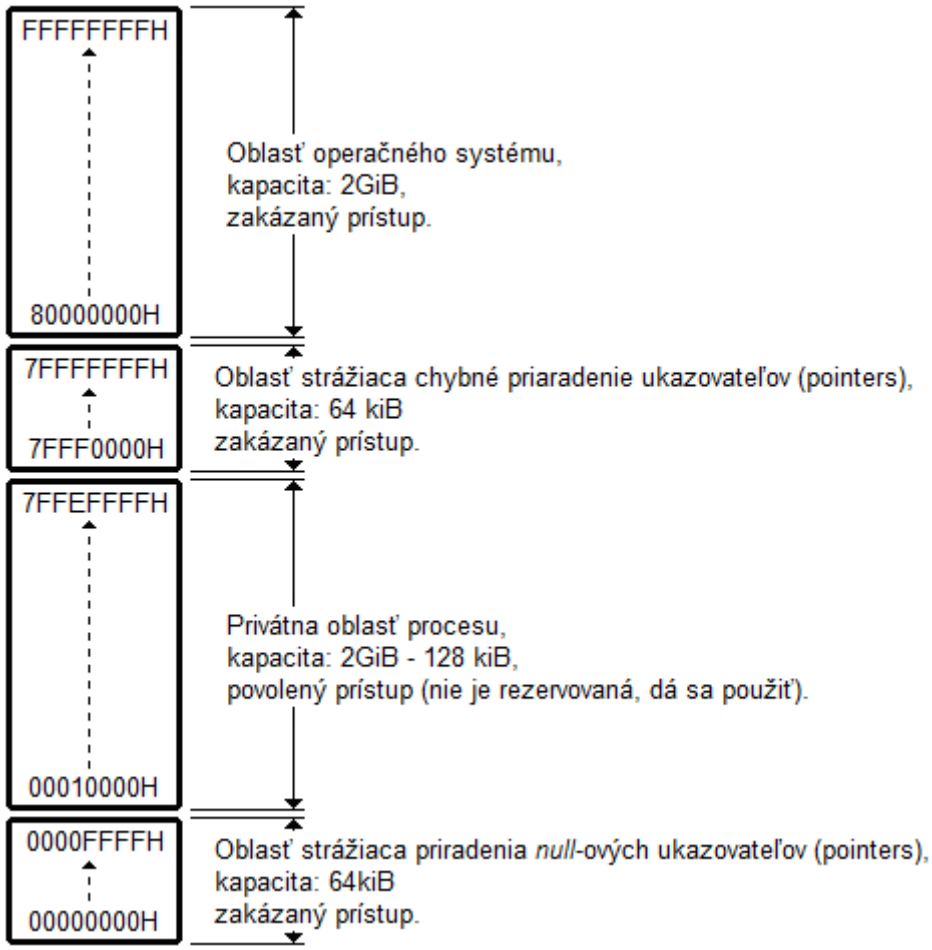
Rozšírená pamäť sa dosiahla stránkovaním, takým spôsobom, že v pamäti sa vytvorí stránkový register, ktorý odkazuje na blok dát v rozšírenej pamäti, tým pádom sa nezvýši počet adresných bitov, ale takýto prístup je obmedzený. Špecifikácia EMS 3.0

dokázala využiť 4 MiB operačnej pamäte, špecifikácia EMS 3.2 využívala 8 MiB a EMS 4.0 dokázala obsluhovať 32 MiB pamäte.

Predĺžená pamäť si už vyžadovala zväčšenie počtu adresných bitov, ktoré musel OS obsluhovať. Taktiež tu boli špecifikácie kde XMS 1.0 bola využitá pre PC – AT na obsluhu 16 MiB, XMS 2.0 obsluhovala 64 MiB a XMS 3.0 obsluhovala 4GiB, čo pre OS DOS postačovalo až do jeho nahradenia inými operačnými systémami.

Na obrázku Obr. 2.3 sú znázornené rozloženia adresovateľnej pamäte od PC – XT po predĺženú pamäť, ktorá mohla dosahovať 4 GiB. Predĺžená štruktúra pamäte bola prispôsobená okrem OS DOS aj na OS Windows 3.0, 3.1, 3.11, 95, 98 a Millennium. Windows 1.0 a Windows 2.0 sa komerčne nerozšíril a používal iba rozšírenú pamäť. Pri OS Windows NT bola zavedená nová filozofia mapovania pamäte.

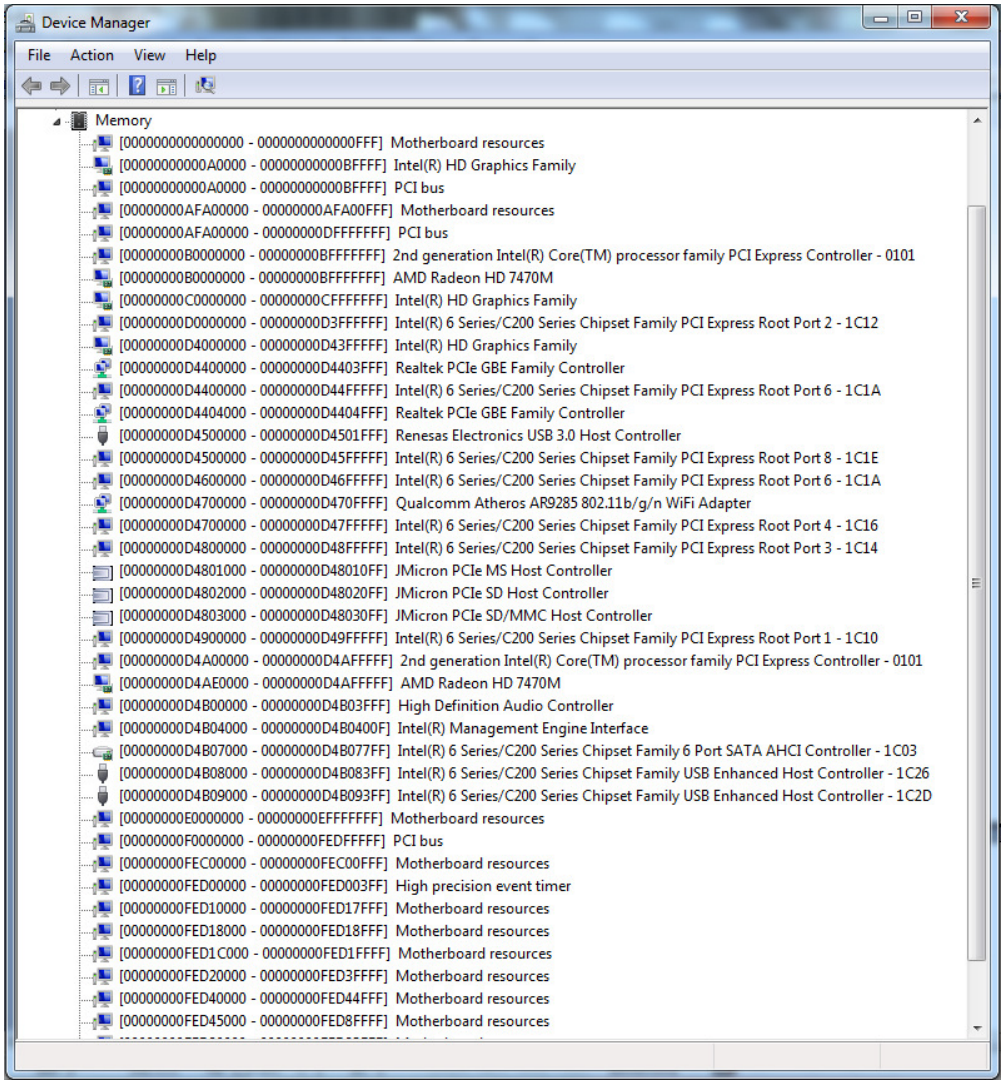
Základné adresovanie virtuálneho adresného priestoru pamäte procesov (viď podkapitolu 8.4) Windows NT 4.0 je na nasledujúcom obrázku (Obr. 2.4):



Obr. 2.4 Adresovanie pamäte v OS Windows NT 4.0

V dnešných OS od firmy Microsoft sa adresovanie pamäte dá prezrieť pomocou nástroja Správca zariadení (Device manager). Ktorý je umiestnený medzi nástrojmi Ovládacieho panela (Control Panel). V nástroji Správca zariadení sa v hornom menu

v položke „Zobraziť“ („View“) vyberie „Prostriedky podľa typu“ („Resources by type“) alebo „Prostriedky podľa pripojenia“ („Resources by connection“) a následne sa rozbalí zložka „Pamäť“ („Memory“). Na obrázku Obr. 2.5 je príklad takéhoto rozloženia adresovania v OS Windows 7 pre konkrétny model PC.



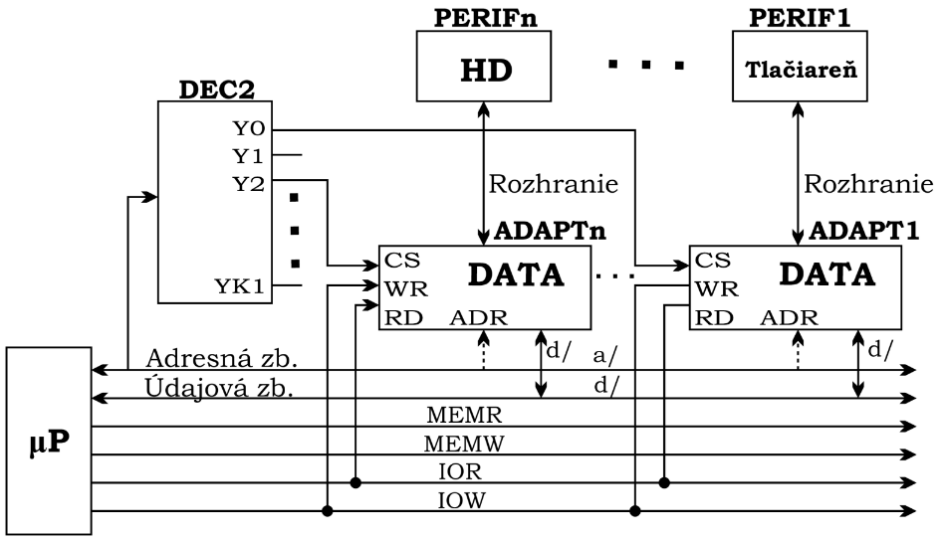
Obr. 2.5 Adresovanie pamäte zobrazené pomocou nástroja Správca zariadení (Device Manager)

2.2 Vstupno-výstupný podsystém počítačov

Vstupno-výstupný podsystém počítačov slúži na vstup respektíve výstup údajov (informácií). Umožňuje komunikáciu procesora s rôznymi vstupnými a výstupnými (periférnymi) zariadeniami. Príkladom vstupného zariadenia je klávesnica, výstupného zariadenia je monitor, vstupno-výstupného zariadenia je pevný disk.

2.2.1 Pripojenie periférnych zariadení k zbernici PC

Periférnych zariadení (ako aj pamäťových obvodov) môže byť k zbernici PC pripojených niekoľko. Mikroprocesor musí byť schopný rozlíšiť, s ktorým periférnym zariadením bude pracovať. Na obrázku Obr. 2.6 je znázornené pripojenie periférnych zariadení k zbernici PC.



Obr. 2.6 Pripojenie periférnych zariadení k zbernici PC

Dekodér DEC2 na základe aktuálnej adresy na adresnej zbernici PC určuje, s ktorým adaptérom sa práve pracuje. Vyberá práve jeden aktívny adaptér (vyznačené čiarkovane na obrázku Obr. 2.6. Vylúčenie konfliktov medzi adaptérmi a pamäťovými zariadeniami je zabezpečené tým, že adaptéry a pamäťové zariadenia majú vlastné riadiace signály, ktoré nikdy nie sú aktívne súčasne. Pamäťové obvody používajú na zápis MEMW#, na čítanie MEMR#. Adaptéry používajú signály IOW# a IOR#.

2.2.2 Komunikácia procesora s adaptérom periférneho zariadenia

Adaptér vytvára nevyhnutné rozhranie medzi zbernicou PC a periférnym zariadením, nakoľko priame pripojenie periférneho zariadenia na zbernicu PC nie je možné (odlišné napäťové úrovne, spôsob prenosu údajov). Adaptéry sú obyčajne programovateľné, t.j. CPU pred vlastným prenosom údajov vyšle do adaptéra riadiace slová.

Rozhranie je štandardizované, čo umožňuje k PC pripojiť periférne zariadenia od ľubovoľného výrobcu, ktoré toto rozhranie rešpektuje. Neštandardné rozhranie sa vytvára pri použití PC na špeciálne aplikácie, napríklad na riadenie technologických procesov.

Čítanie z adaptéra funguje tak, že mikroprocesor musí vyslať na adresnú zbernicu adresu adaptéra, z ktorého chce načítať informáciu. CPU musí nastaviť aktívnu úroveň signálu pre čítanie IOR# (musí trvať dostatočne dlhý čas). Po istom čase (doba prístupu) adaptér vyšle na údajovú zbernicu platnú informáciu.

Zápis do adaptéra zabezpečuje CPU vyslaním adresy adaptéra na adresnú zbernicu do ktorého chce zapísať dáta a na dátovú zbernicu vyšle platné údaje (dáta). CPU nastaví do aktívnej úrovne signál pre zápis IOW#

2.2.3 Štandardné rozhrania na pripojenie periférnych zariadení

Pre štandardné rozhranie sú definované použité napäťové úrovne, konektory, spôsob prenosu, protokoly atď. Medzi štandardné rozhrania patrí:

a) paralelne rozhranie – CENTRONICS,
b) sériové rozhranie – RS-232,
c) USB,
d) PS/2 (myš, klávesnica),
e) AUX (audio – zvuk: audio vstup, audio výstup, mikrofón),
f) Ethernetové rozhranie (RJ-45 slot),
g) Video rozhranie (VGA, DVI, HDMI,...)
h) atď...

2.2.4 Spojenie PC s technologickým prostredím

V prípade, že sa PC používa na riadiacu aplikáciu je priamo spojený s technologickým prostredím. Výstupné periférne zariadenia sú akčné členy a vstupné periférne zariadenia sú senzory. Prostredníctvom akčných členov počítač vstupuje do prostredia a prostredníctvom senzorov načítava stavové informácie z prostredia.

Ak PC obsahuje akčné členy spolu so senzormi, je možné vykonávať riadenie so spätnou väzbou, ktoré sa vyznačuje tým, že PC môže na základe stavovej informácie sledovať výsledok svojho riadiaceho zásahu a korigovať ho na základe rôznych algoritmov riadenia.

2.2.5 Metódy vstupno-výstupných prenosov

Pod vstupno-výstupným prenosom rozumieme prenos údajov medzi periférnym zariadením a CPU alebo medzi periférnym zariadením a pamäťou.

Podľa toho, kto riadi zbernicu PC počas prenosu údajov z a do periférneho zariadenia rozdeľujeme V/V prenosy na:

a) prenosy s účasťou procesora:
   - Pri V/V prenosoch s účasťou procesora generuje riadiace signály zbernice procesor. Týmto spôsobom sa vykonáva prenos jednotlivých údajov a údaje sa prenášajú medzi procesorom a V/V zariadením. Vykonať prenos údajov z a do periférneho zariadenia nemusí byť možné v každom okamihu. Podľa toho, akým spôsobom sa rozhodne o okamihu odštartovania prenosu údajov rozlišujeme tieto V/V prenosy:
     - nepodmienený V/V prenos:
       - CPU implicitne považuje V/V zariadenie v ľubovoľnom okamihu za pripravené na prenos, t.j. kedykoľvek môže z vstupného zariadenia údaj načítať a do výstupného zariadenia kedykoľvek údaj zapísať. Prenos je rýchly, pretože sa vykoná rýchlosťou procesora (načítanie stavových slov).
     - podmienený V/V prenos:
       - Procesor pred vlastným vykonaním prenosu údajov najskôr testuje pripravenosť zariadenia prijať resp. vyslať údaje. CPU

zisťuje pripravenosť takým spôsobom, že z adaptéra načíta stavové slovo, v ktorom bity nesú informáciu o pripravenosti zariadenia. Len keď je zariadenie k prenosu pripravené, procesor vykoná vlastný prenos údajov. Prenos rešpektuje pracovnú rýchlosť zariadenia (procesor veľa času strávi čakaním).

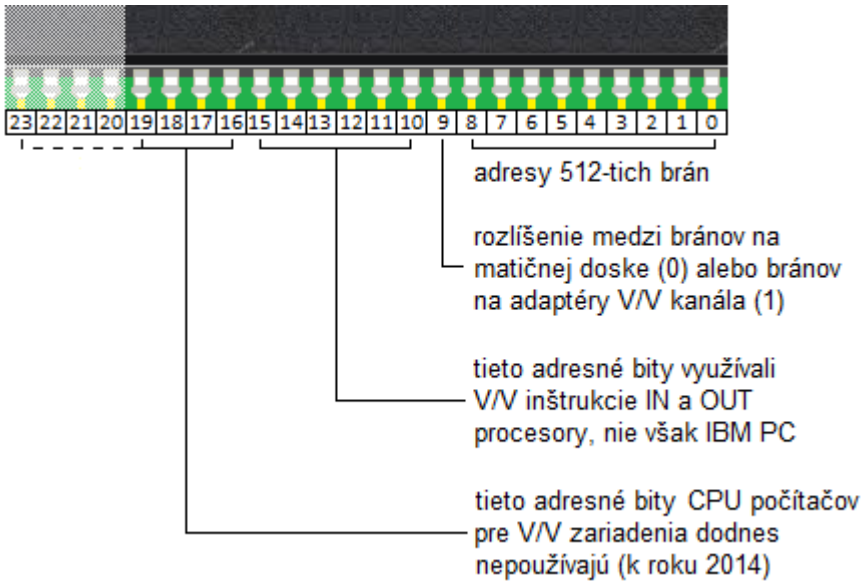
- Prenos s prerušením (podmienený V/V prenos):
  - Nevýhody podmieneného prenosu odstraňuje V/V prenos s prerušením. Pri tomto spôsobe prenosu procesor netestuje pred prenosom údajov pripravenosť V/V zariadenia, ale V/V zariadenie v prípade svojej pripravenosti k prenosu vygeneruje žiadosť o prerušenie. CPU preruší práve prebiehajúci program a v rámci obslužného programu prerušenia (bez otestovania pripravenosti zariadenia) vykoná vlastný prenos údajov. Po jeho skončení pokračuje v prerušenom programe. Ak je periférne zariadenie schopné prijať alebo vyslať ďalší údaj, opäť vygeneruje novú žiadosť o prerušenie. Je úplne vylúčené neefektívne čakanie CPU.

b) prenosy bez účasti procesora:
   - Vyznačujú sa tým, že počas prenosu údajov riadi zbernicu počítača riadiaci obvod DMA a procesor je od zbernice odpojený (má svoje výstupy v stave vysokej impedancie). Údaje sa prenášajú medzi pamäťou a V/V zariadením. Tento spôsob prenosu sa nazýva priamy prístup do pamäte (DMA – Direct Memory Access) a typicky sa používa blokový prenos údajov, napr. pri práci s pevným diskom. Ako riadiaci obvod DMA je použitý buď špecializovaný programovateľný obvod alebo špeciálny V/V procesor. Tento spôsob prenosu sa vykonáva vtedy, ak riadiaci obvod DMA vykonáva prenos údajov rýchlejšie ako procesor. Procesor odovzdá riadiacemu obvodu DMA požiadavky na prenos a ďalej už len čaká na jeho vykonanie. V/V prenos, aj keď je bez účasti procesora, je inicializovaný procesorom. Pred vlastným uskutočnením prenosu údajov musí procesor oznámiť riadiacemu obvodu DMA požiadavky na prenos, teda odkiaľ a kam sa majú údaje prenášať a koľko ich má byť. Až potom riadiaci obvod DMA požiada procesor o pridelenie zbernice a keď mu ju procesor pridelí, vykoná vlastný prenos údajov. Po ukončení prenosu vráti riadenie zbernice späť procesoru, ktorý môže pokračovať v činnosti.

2.2.6 Adresovanie vstupno-výstupných periférií

Vstupno-výstupný podsystém slúži v každom počítači k obojsmernému prenosu dát a k riadeniu komunikácie medzi CPU a pamäťou na prvej strane a vonkajším svetom na druhej strane. Nielen bežné periférne zariadenia, ale prakticky celý vonkajší svet sa k IBM PC pripája prostredníctvom adaptérov zasúvaných do vstupno-výstupných kanálov (zbernice PC). Technicky sa pripojenie realizuje vstupnými a výstupnými 8, 16, 32, alebo 64-bitovými bránami (portami). Šírka brány ja závislá od šírky dátovej zbernice počítača, ale zariadenie nemusí používať plnú šírku dátovej zbernice. Tieto brány, bez ohľadu nato, že či sú jednoduché a samostatné, alebo sú súčasťou veľmi zložitých prepojovacích obvodov, sú k PC pripojené ako izolované vstupno-výstupné zariadenia a každý z nich má svoju vlastnú vstupnú alebo výstupnú adresu. Príkladom môže biť aj jednoduchá sériová linka, ktorá používa brány 3F8 až 3FF pre jednu linku (viď podkapitolu 7.1). Mikroprocesor 8088

mohol teoreticky pracovať so 65536 (16 adresovateľných bitov) bránami so šírkou 8 bitov a mikroprocesor 80286 buď so 65536 bránami so šírkou 8 bitov, alebo s 32768 bránami so šírkou 16 bitov. Návrhári IBM PC sa však rozhodli obmedziť vstupno-výstupný priestor na 1024 brán u oboch modeloch. Navyše ešte spodná polovica (512 brán) bola vyhradená pre systémovú (matičnú) dosku. Vzhľadom na to, že PC sú Von Neumann-ovskej architektúry je adresný a dátový priestor oddelený a znázornenie vstupno-výstupného priestoru je na obrázku Obr. 2.7.



Obr. 2.7 Adresovanie vstupno-výstupného priestoru počítačov PC-XT a PC-AT

Z obrázka Obr. 2.7 je viditeľné, že pre adresovanie sa využívalo 10 najvyšších bitov, 0. až 8. bit rozlišoval medzi 512-timi bránami (portami) a 9. bit slúžil ako prepínač pre adresovanie V/V adaptérov na matičnej doske (0) a pre adresovanie adaptérov V/V kanálov (externých zariadení). Takže zariadenia na matičnej doske mali adresy od 000H po 1FFH a zariadenia mimo matičnej dosky mali adresy 200H až 3FFH. V prípade, že k počítaču boli pripojené adaptéry so 16 bitovou adresnou a 8 bitovou dátovou zbernicou a počítač mal 16 bitovú dátovú zbernicu bolo potrebné vytvoriť dodatočnú dekódovaciu logiku, ktorá umožňovala aplikovať túto logiku s väčšími dátovými zbernicami.

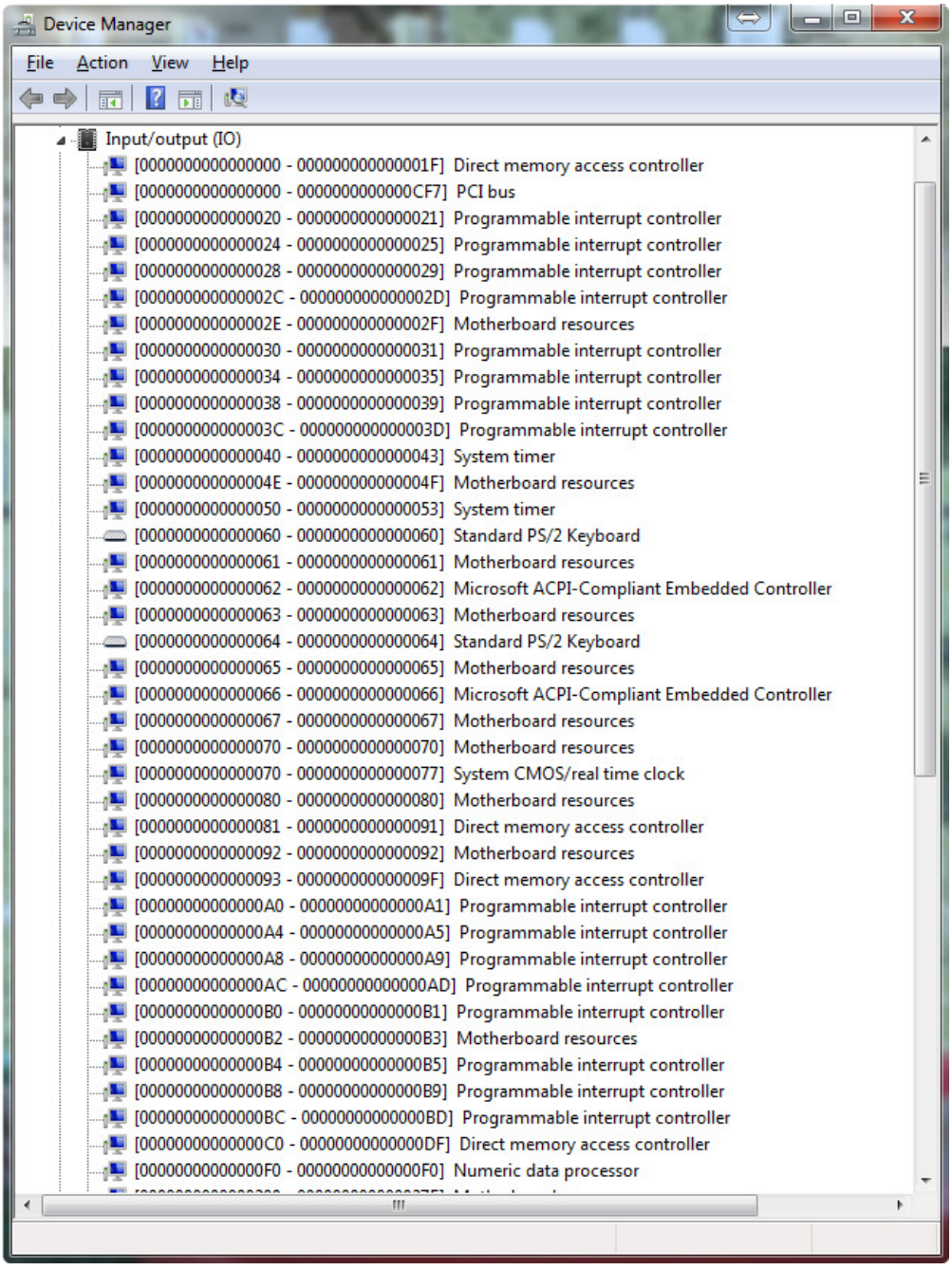
V tých časoch sa plno adries bežne využívaných V/V adaptérov určili ako nemenné a dodnes platia, ak sa v počítači vyskytujú, tieto adresy sú v tabuľke Tab. 2.1. V spomínanej tabuľke je poslednou adresou konfigurácia PCI, kde je viditeľné, že pri nástupe PCI kariet sa adresný priestor V/V adaptérov rozšíril. Pri nástupe PCI kariet sa adresný priestor rozšíril na 16 bitov s ktorými komunikovali karty s okolím a vyššie bity (nakoľko už boli 32 bitové počítače) využívali na internú komunikáciu s vlastnými radičmi. Nástupom PCIe zostala adresovanie zariadení stále na úrovni 16 bitov (65536 adresovateľných zariadení je dostatočný počet pre jeden dnešný PC).

Adresný priestor vstupno-výstupných zariadení po adresu 03FF sa rokmi zaplnil a stal sa akýmsi štandardom, ktorý dnešné počítače ak dané zariadenie (radič) obsahujú dodržiavajú. Väčšina týchto zariadení je už súčasťou southbridge-u takže sú priamo na matičnej doske. Zoznam adries vstupno-výstupných adaptérov sú v tabuľke Tab. 2.1.

V dnešných OS od firmy Microsoft sa adresovanie vstupno-výstupných zariadení dá prezrieť pomocou nástroja Správca zariadení (Device manager). Ktorý je umiestnený medzi nástrojmi Ovládacieho panela (Control Panel). V nástroji Správca zariadení sa v hornom menu v položke „Zobraziť“ („View“) vyberie „Prostriedky podľa typu“ („Resources by type“) alebo „Prostriedky podľa pripojenia“ („Resources by connection“) a následne sa rozbalí zložka „Vstup alebo výstup“ („Input/Output (IO)“). Na obrázku Obr. 2.5 je príklad takéhoto rozloženia adresovania v OS Windows 7 pre konkrétny model PC.

Tab. 2.1 Adresy V/V adaptérov

Adresa	**Adaptér**	**Adresa**	**Adaptér**
00 – 1F	Prvý DMA radič	220 – 233	Radič zvuku (Sound blaster)
20 – 3F	Prvý programovateľný radič prerušení, Master	278 – 27F	Paralelný port 2
40 – 5F	Programovateľný časovač (systémový časovač)	2B0 – 2DF	Radič EGA (grafický radič)
60 – 6F	Klávesnica	2E8 – 2EF	Sériový port 4
70 – 7F	Reálne hodiny, maska NMI	2E2 – 2E3	Akvizícia dát
80 – 9F	Stránkový register DMA	2F8 – 2FF	Sériový port 2
87	DMA kanál 0	300 – 31F	Prototypová karta
83	DMA kanál 1	320 – 323	Kompatibilný radič pevných diskov
81	DMA kanál 2	330 – 331	Radič zvuku (MPU-401 MIDI)
82	DMA kanál 3	340 – 35F	Primárny radič SCSI
8B	DMA kanál 5	370 – 377	Sekundárny radič diskiet
89	DMA kanál 6	378 – 37F	Paralelný port 1
8A	DMA kanál 7	380 – 38C	Sekundárny adaptér synchrónnej komunikácie SDLC
8F	Refresh, reštart, obnovenie	388 – 389	Radič zvuku (Music Synthesizer Card)
A0 – BF	Druhý programovateľný radič prerušení, Slave	3A0 – 3A9	Primárny adaptér synchrónnej komunikácie SDLC
C0 – DF	Druhý DMA radič	3B0 – 3BB	Radič MDA (grafický radič)
F0 – FF	Matematický koprocesor	3BC – 3BF	Paralelný port na karte MDA
100 – 10F	Programovateľné nastavenia (PS/2)	3C0 – 3CF	EGA port
110 – 1EF	Systémový V/V kanál	3D0 – 3DF	Radič CGA (grafický radič)
140 – 15F	Sekundárny radič SCSI	3E8 – 3EF	Sériový port 3
170 – 177	Sekundárny radič PATA diskov	3F0 – 3F7	Primárny radič diskiet
1F0 – 1F7	Primárny radič PATA diskov	3F8 – 3FF	Sériový port 1
200 – 20F	Port pre joyistick	CF8 – CFC	Konfigurácia PCI



Obr. 2.8 Adresovanie vstupno-výstupných zariadení zobrazené pomocou nástroja Správca zariadení (Device Manager)

2.2.7 Programovanie periférií pripojených na vstupno-výstupný podsystém

Zápis a čítanie zo vstupno-výstupného podsystému pomocou programovacieho jazyka C a C++ sa deje najčastejšie pomocou knižnice *dos.h* a to konkrétne týmito funkciami s príslušnou syntaxou:

```
int hodnota = inport(unsigned int adresa);
int hodnota = inportb(unsigned int adresa);
outport(unsigned int adresa, unsigned int hodnota);
outportb(unsigned int adresa, int hodnota);
```

Funkcie `inport()` a `inportb()` slúžia na vyčítanie hodnoty z príslušného registra periférneho zariadenia. Rozdiel medzi týmito funkciami je ten, že `inportb()` vyčíta iba 1B (hodnotu od 0 po 255) a `inport()` vyčíta celé slovo (2B – hodnota od 0 po 65 535). Funkcie `outport()` a `outportb()` do daného registra periférneho zariadenie hodnotu zapisujú. Rozdiel medzi nimi je rovnaký ako pri funkciách `inport()` a `inportb()`, funkcia `outportb()` dokáže do registra zapísať iba 1B a funkcia `outport()` dokáže zapísať celé slovo. Vysvetlenie premenných:

- `adresa` – adresa daného registra (portu) vstupno-výstupného zariadenia (Tab. 2.1),
- `hodnota` – hodnota ktorú chceme zapísať, alebo vyčítať z daného registra (portu) vstupno-výstupného zariadenia.

Kompilátory v novších operačných systémoch od firmy Microsoft (nové Windows-y) zakazujú priamy zápis na porty a tak sa využíva dynamická knižnica *inpout32.dll*, alebo *inpout64.dll*, ktoré vedia na porty vstupno-výstupných zariadení zapisovať.

V programovacom jazyku C# sa importujú konkrétne funkcie do projektu pomocou príkazu `DllImport` v nasledujúcom prípade sa to bude konkrétne týkať funkcií `Out32` a `Inp32` z knižnice `inpout32.dll` a nahradia sa internými funkciami projektu `Out()` a `In()`:

```
[DllImport("inpout32.dll", EntryPoint = "Out32")]
public static extern void Out(int address, int value);

[DllImport("inpout32.dll", EntryPoint = "Inp32")]
    public static extern int In(int address);
```

ako je viditeľné z parametrov nové funkcie `Out()` a `In()` nahradia funkcie `inportb()/inport()` a `outportb()/outport()`.

K zavolaniu funkcie `DllImport` je potrebné nalinkovať nový „name space“ pomocou príkazu `using` v hlavičke programu:

```
using System.Runtime.InteropServices;
```

Po týchto krokoch sa funkcie používajú takto:

```
Out(unsigned int adresa, int hodnota);
int hodnota = In(unsigned int adresa);
```

2.3 Prerušovací podsystém počítačov

Prerušovací podsystém počítača slúži na prerušenie činnosti procesora a nahradením prerušeného programu programom z vektora prerušení. Mnoho hardvéru pripojených k PC žiada o prerušenie. Tým asi najzákladnejším je počítačová myš. Aj keď je obslužný program myši jednoduchý, nemusí byť spustený po celý čas v OS ako služba (bežiace programy na pozadí OS), ale jej program je uložený vo vektore prerušení a vykoná sa iba v okamihu, keď sa myšou pohne, alebo sa stlačí tlačidlo.

V prvých PC, ktoré mali jedno-procesný (jedno-vláknový) CPU a OS boli taktiež jedno-užívateľské a jedno-procesné mali prerušenia veľmi dôležitú úlohu, nakoľko počas behu programu sa mohol vykonať iný dôležitejší program (stlačenie klávesa na klávesnici, pohnutie myšou, uloženie/vytlačenie stavu obrazovky – Print Screen, atď...). Prerušenia majú výhodu aj v dnešných počítačoch, nakoľko v OS nie sú nutné spustené služby pre zariadenia, ktoré môžu využívať výhody prerušovacieho podsystému. Tým pádom procesory nie sú spomaľované neustále bežiacimi procesmi a službami.

Charakteristickým znakom súčasných procesorov je vyspelý prerušovací systém, ktorý umožňuje efektívnu implementáciu viacpoužívateľských a viacprogramových OS a rýchlu odozvu na externé udalosti.

Mikroprocesor vykonáva program a nastane požiadavka okamžitej obsluhy novej udalosti (požiadavka o prerušenie). V tom momente musí CPU prerušiť vykonávanie práve bežiaceho programu a začať vykonávať nový program – obslužný program prerušenia. Po skončení obslužného programu bude CPU pokračovať v pôvodnom prerušenom stave. Takže prerušovací podsystém zabezpečuje presmerovanie kódu programu na obslužnú rutinu prerušenia a po vykonaní obslužnej rutiny prerušenia vráti chod programu na miesto, kde bol prerušený.

Prerušenia vieme rozdeliť na:

- softvérové (synchrónne) prerušenia,
- hardvérové (asynchrónne) prerušenia.

Synchrónne prerušenia majú pevne určené štartovacie adresy obslužných programov. *Asynchrónnym prerušeniam* sa štartovacia adresa určuje prostredníctvom vektora prerušení. *Vektor prerušení* (prerušovací vektor) je ukazovateľ tabuľky štartovacích adries obslužných programov prerušení. Tento vektor je načítaný CPU z dátovej zbernice po prijatí prerušenia v špeciálnom cykle potvrdenia prerušenia (Interrupt Acknowledge Cycle - IAC).

2.3.1 Softvérové prerušenia

Synchrónne prerušenie priamo súvisí s vykonávanými inštrukciami a nie je ho možné zakázať. Softvérové prerušenie je generované po vykonaní špeciálnej riadiacej inštrukcie. Parametrom tejto inštrukcie je číslo prerušenia (INT 0 až INT 255), ktoré sa má obslúžiť. Toto prerušenie sa používa pri volaní funkcií OS.

Zvláštnym typom synchrónneho prerušenia je výnimka (exception), ktorá sa generuje automaticky, ak nastane chyba pri vykonaní inštrukcie (napríklad delenie nulou, pretečenie premennej).

Softvérové žiadosti o prerušenie (PC-XT a PC-AT)

Prerušenie je pri PC-XT a PC-AT možné vyvolať programovo inštrukciami INT *n*. Podľa konkrétnej hodnoty *n* sa vypočíta adresa ukazovateľa do tabuľky vektorov prerušení (napr.: *n* = 1AH prislúcha adresa 104). Pomocou programových prerušení využíval

programátor služieb BIOS-u. V tabuľke Tab. 2.2 je zoznam adries jednotlivých softvérových a hardvérových prerušení a ich význam pri počítačoch PC-XT a PC-AT.

Tab. 2.2 Význam jednotlivých typov prerušenia (softvérové a hardvérové prerušenia) u PC-XT a PC-AT

Preru-šenie	Význam	Preru-šenie	Význam
0H	Delenie nulou	18H	Rezidentný BASIC
1H	Krokovanie programu	19H	Zavádzací program
2H	NMI	1AH	Systémový čas
3H	Zastavenie programu (break point)	1BH	Klávesa BREAK
4H	Aritmetické pretečenie	1CH	Časový skok
5H	Tlač obrazovky (BIOS)	1DH	Inicializácia zobrazenia
6H	Rezerva	1EH	Parametre diskety
7H	Rezerva	1FH	Parametre grafického zobrazenia
8H	**hardvérový časovač (prvý radič 8259A)**	20H až 3FH	Služby operačného systému (MS – DOS)
9H	**Klávesnica (prvý radič 8259A)**	40H až 5FH	Rezerva
AH	**Rezerva (prvý radič 8259A)**	60H až 67H	Prerušenie z užívateľského programu
BH	**Asynchrónna komunikácia 2 (prvý radič 8259A)**	68H až 6FH	Nepoužité
CH	**Asynchrónna komunikácia 1 (prvý radič 8259A)**	70H	**Reálne hodiny (druhý radič 8259A)**
DH	**Tlačiareň 2 / LPT 2 (prvý radič 8259A)**	71H	**IRQ 9 (druhý radič 8259A)**
EH	**Disketa (prvý radič 8259A)**	72H	**IRQ 10 (druhý radič 8259A)**
FH	**Tlačiareň 1 / LPT 1 (prvý radič 8259A)**	73H	**IRQ 11 (druhý radič 8259A)**
10H	Zobrazenie znaku	74H	**IRQ 12 (druhý radič 8259A)**
11H	Chyba zariadenia	75H	**IRQ 13 presmerované na NMI (druhý radič 8259A)**
12H	Veľkosť pamäte	76H	**IRQ 14 (druhý radič 8259A)**
13H	Disketa / disk	77H	**IRQ 15 (druhý radič 8259A)**
14H	Obsluha RS-232 (BIOS)	78H až 7FH	Nepoužité
15H	Kazetový magnetofón	80H až 85H	Rezervované pre rezidentný BASIC
16H	Vstup z klávesnice (BIOS)	86H až F0H	Používa rezidentný BASIC
17H	Výstup na tlačiareň (BIOS)	F1H až FFH	Nepoužité

2.3.2 Hardvérové prerušenie

Asynchrónne prerušenie priamo nesúvisí s vykonávanými inštrukciami a môže nastať kedykoľvek. Je to tzv. externé (hardvérové) prerušenie a typicky je požadované niektorým vstupno-výstupným zariadením, keď je toto pripravené na prenos.

Procesor má dva prerušovacie vstupy pre externé prerušenie:

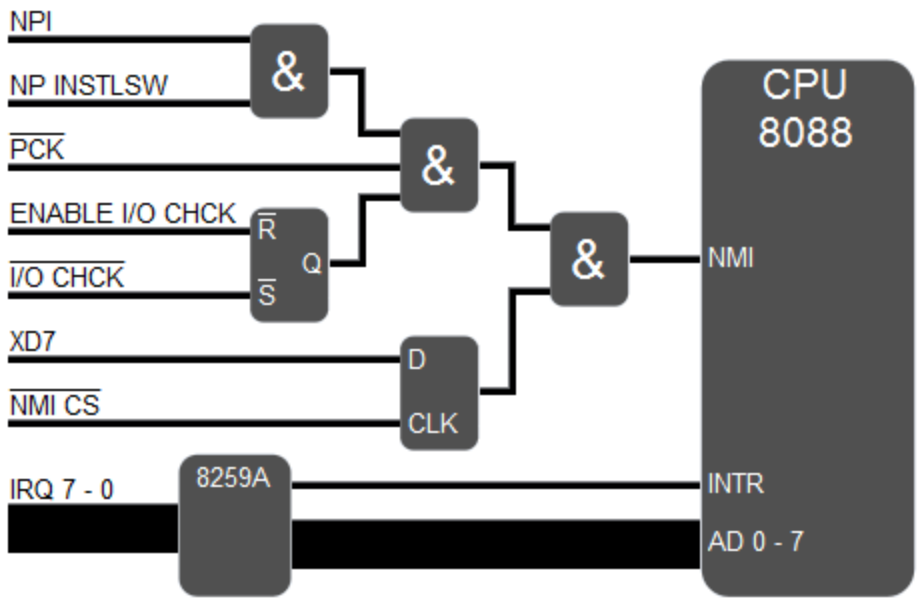
- vstup maskovateľného (podmieneného) prerušenia – inštrukčný súbor procesora obsahuje v tomto prípade špeciálne inštrukcie, ktoré umožňujú povoliť/zakázať prijatie požiadavky z tohto vstupu, je to podmienené prerušenie, ktoré je vyvolávané externým hardvérom (napríklad myš, klávesnica, atď.), označenie IRQ 0 – IRQ 15 (interrupt request),
- vstup nemaskovateľného (nepodmieneného) prerušenia – toto prerušenie nie je možné zakázať (napríklad výpadok napájanie, prehriatie procesora, chyba pamäte /check sum error/, porucha vstupno-výstupného kanála atď.), označenie NMI (non mask interrupt).

2.3.3 Prerušovací podsystém PC-XT až PC-486

Prerušovací podsystém PC-XT

Procesor 8088 mal dva vstupy žiadosti o prerušenie NMI (nepodmienené prerušenie) a INTR (podmienené prerušenie). Obrázok prerušovacieho podsystému PC-XT je na obrázku Obr. 2.9.

Žiadateľmi o prerušenie pripojené k NMI mohli byť numerický koprocesor (NPI), chyba parity pamäte (PCK), chyba vstupno-výstupného zariadenia (I/O CHCK). Nemaskovateľný bol tento vstup iba v samotnom mikroprocesore. Jeho vonkajšie obvody sa však zamaskovať dali, numerický koprocesor sa maskoval signálom NP INSTLSW (maskoval sa v prípade, že koprocesor nebol vôbec pripojený k PC), chyba vstupno-výstupného zariadenia sa maskovala signálom ENABLE I/O CHCK. Pomocou bitu XD7 (7. bit na adrese NMI CS, konkrétne na adrese 0A0H) sa dali zamaskovať všetky žiadosti o prerušenie NMI.

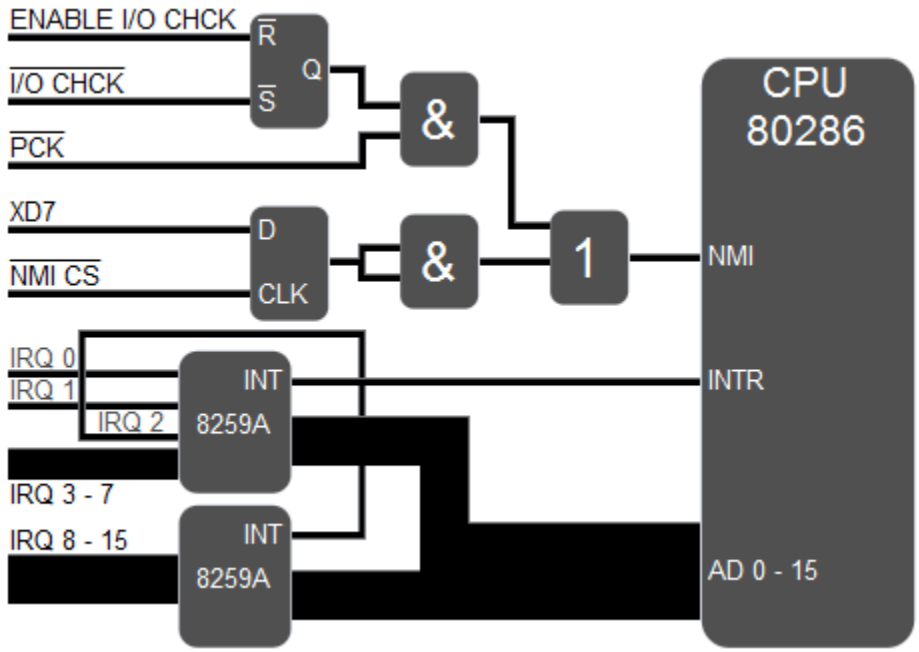


Obr. 2.9 Prerušovací podsystém PC-XT

Druhý vstup žiadosti o prerušenie označovaný INTR, ktorý generoval radič prerušení 8259A bol naprogramovaný na registráciu aktívnej úrovne a šiestich jeho vstupov (IRQ 2 až IRQ 7) boli prístupné na konektoroch vstupno-výstupného kanálu (na zbernici PC-BUS). Prerušenia IRQ 0 a IRQ 1 boli použité ako žiadosti z matičnej dosky, konkrétne IRQ 0 bola žiadosť o prerušenie od časovača a IRQ 1 bola žiadosť o prerušenie od klávesnice, pričom označenia IRQ 0 a IRQ 1 sa dodnes používajú na označenie týchto žiadostí.

Prerušovací podsystém PC-AT až PC-486

Štruktúra prerušovacieho podsystému PC-AT je na obrázku Obr. 2.10. Tento podsystém mal rovnaký základ ako pri PC-XT založený na obvode 8259A, aj tu sa pomocou signálu NMI CS dali zamaskovať všetky žiadosti o prerušenie. Na tomto vstupe sa však tento krát uplatňovala iba žiadosť spôsobená chybou parity pamäte PCK (ktorý sa nedá zamaskovať) alebo chyba vstupno-výstupného kanálu I/O CHCK (ktorý sa maskuje pomocou ENABLE I/O CHCK). Prerušenie od koprocesora sa tento krát predávalo už signálom IRQ 13, takže toto prerušenie už nepatrilo do skupiny NMI. Škála možných žiadostí, ktoré žiadali prerušenie pomocou signálu INTR sa rozšíril z 8 na 15. Radič prerušenia tvoril v tomto prípade 2 obvody 8259A zapojené kaskádovito a pribudlo tak ďalších 7 vstupov. Žiadosti o prerušenie IRQ 9 až IRQ 12, IRQ 14 a IRQ 15 boli prístupné na zbernici ISA. IRQ 9 nahradilo žiadosť IRQ 2, pretože IRQ 2 sa použila pre kaskádovité zapojenie oboch obvodov 8259A. Podľa tohto zapojenia sú logické aj prednastavené priority prerušenia a to IRQ 0, IRQ 1, IRQ 8, IRQ 9, IRQ 10, IRQ 11, IRQ 12, IRQ 13, IRQ 14, IRQ 15, IRQ 3, IRQ 4, IRQ 5, IRQ 6, IRQ 7. K ďalšiemu rozširovaniu prerušovacieho podsystému pomocou 8259A už nedošlo, pretože na matičných doskách chýbali signály pre zapojenie viacerých radičov a neskôr sa táto architektúra prerušovacieho podsystému už zmenila (príchodom PCI).



Obr. 2.10 Prerušovací podsystém PC-AT

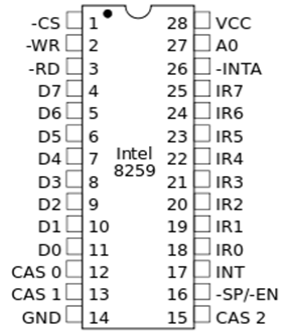
V tabuľke Tab. 2.3 je prehľad hardvérových prerušovacích zdrojov a adries ich vektorov pre PC-XT a PC-AT.

Tab. 2.3 Zoznam hardvérových prerušení

Prerušenie	Adresa vektoru prerušenia	Typ	Význam pre XT	Význam pre AT
NMI	008H	2	chyba parity, V/V kanálu, koprocesora	chyba parity, alebo V/V kanálu
IRQ 0	020H	8	systémový časovač	Systémový časovač
IRQ 1	024H	9	Klávesnica	Klávesnica
IRQ 2	028H	10	Rezerva	kaskádovité zapojenie druhého radiča
IRQ 3	02CH	11	voliteľný asynchrónny adaptér	Voliteľný asynchrónny adaptér
IRQ 4	030H	12	základný asynchrónny adaptér	Základný asynchrónny adaptér
IRQ 5	034H	13	Winchester disk	2. paralelný adaptér
IRQ 6	038H	14	Disketa	Disketa
IRQ 7	03CH	15	paralelný adaptér	1. paralelný adaptér
IRQ 8	1C0H	112		kalendárový obvod
IRQ 9	1C4H	113		rezerva (pre ISA)
IRQ 10	1C8H	114		rezerva (pre ISA)
IRQ 11	1CCH	115		rezerva (pre ISA)
IRQ 12	1D0H	116		rezerva (pre ISA)
IRQ 13	1D4H	117		Koprocesor
IRQ 14	1D8H	118		Winchester disk
IRQ 15	1DCH	119		rezerva (pre ISA)

Radič prerušenia

Základným elementom prerušovacieho podsystému bol radič prerušení 8259A. Na obrázku Obr. 2.11 je znázornený radič 8259A.



Obr. 2.11 Označenie vývodov na radiči 8259A

V tabuľke Tab. 2.4 je vysvetlený význam kontaktov na radiči prerušenia 8259A.

Tab. 2.4 Význam vývodov (pinov) radiča prerušení 8259A

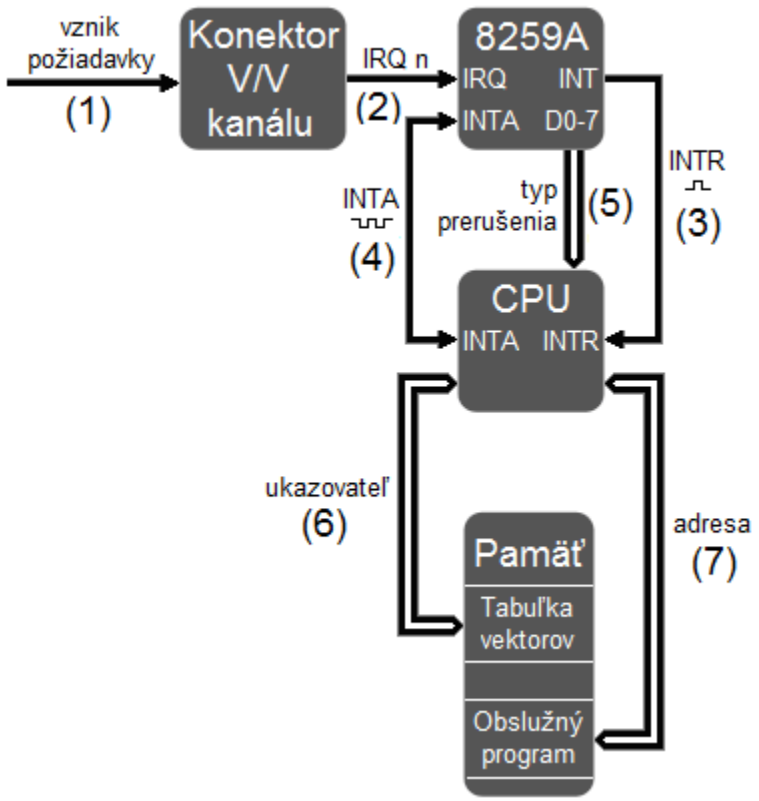
Kontakt (pin)	Signál	Význam
1	CS	Chip Select – vstupný výberový signál puzdra (aktívna je nízka úroveň),
2	WR	Write – vstupný riadiaci signál pre zápis (aktívna je nízka úroveň),
3	RD	Read – vstupný riadiaci signál pre čítanie (aktívna je nízka úroveň),
4 – 11	D0 – D7	Data – obojsmerná dátová zbernica,
12, 13, 15	CAS0 – CAS2	Cascade Lines – signály umožňujúce kaskádovité zapojenie niekoľko obvodov 8259A, pracujú ako výstupy pre riadiaci obvod a ako vstupy pre podriadené obvody,
14	GND	Ground – uzemnenie,
16	SP / EN	Slave Program / Enable Buffer – v režime práce s vonkajším budičom dátovej zbernice slúži signál EN pre riadenie budiča dátovej zbernice, v režime bez vonkajšieho budiča slúži signál k rozlíšeniu riadiaceho obvodu, alebo podriadeného obvodu, aktívna je teda nízka úroveň,
17	INT	Interrupt – výstupný signál výslednej žiadosti o prerušenie (pripája sa obvykle na CPU konkrétne na vstup INT),
18 – 25	IR0 – IR7	Interrupt Request – vstupné signály žiadosti o prerušenie, žiadosť o prerušenie je vyvolaná buď nábežnou hranou, alebo vysokou úrovňou signálu IR v závislosti na režime práce obvodu,
26	INTA	Interrupt Acknowledge – vstupný signál oznamujúci prijatie žiadosti o prerušenie činnosti CPU a umožňujúci zaslanie typu prerušenia na dátovú zbernicu (aktívna je nízka úroveň),
27	A0	Address – vstupný signál pre výber príkazových (stavových) slov, ktoré sú do obvodu zapisované (čítané), obvykle je signál pripojený k jednej z adresných liniek (napr. A0),
28	VCC	VCC – elektrické napájanie.

Týmto obvodom sa poradie dôležitosti (priorita) mohlo dynamicky (naprogramovaním) meniť a jednotlivé prerušenia zamaskovať. Žiadosť o prerušenie zachytával radič prerušenia do *vstupného registra IRR*, ten sa mohol naprogramovať tak, aby reagoval na nábežnú hranu, alebo na vysokú úroveň, PC boli naprogramované väčšinou na aktívnu (vysokú) úroveň signálu. Maskovací register bol označený ako *IMR*. Zápisom nuly do niektorého z bitov maskovacieho registra sa zapríčinilo, že systém na dané prerušenie nereagoval. Výsledok vyhodnotenia priority sa zapisoval do takzvaného registra

žiadosti o obsluhu prerušenia *ISR*. Takže v tomto registri boli prerušenia zoradené podľa priority a tak pristupovali k CPU. K programovaniu radiča prerušení sa používali riadiace registre (ICW/OCW viď nižšie) a programovanie registrov prebiehalo v inicializačnej časti programu, no niektoré parametre sa dali nastaviť (alebo zmeniť) aj behom činnosti programu. Bolo taktiež možné programovo zisťovať okamžitý stav prerušovacieho podsystému (pomocou ICW/OCW).

Postupnosť činností pri obsluhe prerušenia pomocou radičov 8259A

Pomocou obrázka Obr. 2.12 sa vysvetlí postupnosť akcií, ktoré začínajú vznikom prerušovacej udalosti a končia zápisom vektora adresy obsluhy prerušenia do adresného registra pamäte, ktorá poskytuje prvú inštrukciu obslužného programu. Jednotlivé kroky budú číslované v zátvorkách presne podľa obrázka (Obr. 2.12). Aby táto postupnosť mohla prebehnúť, je potrebné najskôr prerušovací systém naprogramovať. Toto naprogramovanie zaisťuje inicializačná časť BIOS-u. Niektoré parametre môže užívateľ (programátor) v rámci svojho programu zmeniť.



Obr. 2.12 Schéma postupu obsluhy prerušenia podľa jednotlivých zúčastnených prvkov počítača

Prvou udalosťou súvisiacou s prerušením je vznik požiadavky **(1)**, ktorý sa generuje v niektorom z adaptérov. Prerušenie sa k ďalšiemu spracovaniu predáva pomocou linky IRQ*n* **(2)** na zbernici niektorého adaptéra, ktorý žiada o prerušenie. Najdôležitejšiu požiadavku (ak v registri žiadosti o obsluhu prerušenia sú ešte nespracované žiadosti) vyhodnotí radič prerušenia **(3)** a aktivuje signál INTR. Ďalší postup záleží na tom, či je dané

prerušenie na procesore programovo povolené. V kladnom prípade sa dokončí rozpracovaná inštrukcia, uloží sa ukazovateľ (miesto nasledujúcej inštrukcie) prebiehajúceho programu do stavového registra a procesor vyšle signál INTA **(4)**. V zápornom prípade pokračuje procesor nerušene vo svojej činnosti, ako keby signál INTR neprišiel. Situáciu zmení iba inštrukcia STI (povolenie prerušenia).

Potvrdzovací signál INTA vysiela procesor k radiču prerušenia vždy dva krát. Na základe prijatia prvého signálu rozhodne radič o najdôležitejšej požiadavke a po prijatí druhého signálu vyšle opačným smerom po dátovej zbernici 8 bitovú hodnotu *typu prerušenia* **(5)**. Teoreticky sú síce možné všetky kombinácie z množiny <0; 255>, ale pri klasických počítačoch (IBM PC) sú najskôr BIOS-om naprogramované typy prerušenia na 8H až FH (8 až 15) a 70H až 77H (112 až 119), tieto prerušenia sú v tabuľke Tab. 2.2 vyznačené hrubým písmom. Typ prerušenia prepočíta procesor na ukazovateľ do tabuľky prerušenia (4 x typ prerušenia) a po adresnej zbernici vyšle adresu položky **(6)**, na ktorej sú 4 bajty ukazovetaľa (vektor) na obslužný program prerušenia. Vektor si procesor uloží a týmto okamihom začína spracovávať požiadavku o prerušenie **(7)**.

V obslužnom programe by sa mala vynulovať pôvodná žiadosť o prerušenie, pretože bola uplatnená a periférie sú obvykle konštruované tak, že toto programové nulovanie vyžadujú. Obslužný program musí záverom vynulovať aj bit odpovedajúci práve obsluhovanému prerušeniu v registri ISR (register žiadosti o obsluhu prerušenia) radiča 8259A. Spraví tak zaslaním príkazu EOI (end of interrupt) pomocou príkazu OCW2 pre 8259A. Prechodom na obslužný program prerušenia sa automatický zakáže každé ďalšie prerušenie. Zmeniť sa to dá novou inštrukciou STI behom obslužného programu, alebo automaticky pri prevedení návratovej inštrukcie nahratím starého stavu, ktorý je uschovaný v zásobníku stavového registra procesora.

Ak procesor po vykonaní obslužného programu už nemá na vstupe aktívny signál INTR, vyčíta zo stavového registra ukazovateľ na program v ktorom skončil pred prijatím signálu INTR a pokračuje v danom programe.

Tabuľka vektorov prerušenia neobsahuje len položky aktivované technickými prostriedkami prerušovacieho podsystému. Sú v nej aj štartovacie adresy programov spúšťaných inštrukciou INT *n* (softvérové prerušenia). Zoznam týchto prerušení bol zobrazený v tabuľke Tab. 2.2.

Zhrnutie postupu pri prerušení:

1. vznik požiadavky o prerušenie,
2. vyslanie signálu IRQ *n* perifémym zariadením a prijatie signálu radičom prerušenia,
3. Vyhodnotenie priority prerušenia a vyslanie signálu INTR radičom prerušenia a prijatie signálu procesorom
4. procesor vyhodnotí povolenie prerušenia, ak je prerušenie povolené dokončí inštrukciu, stav uloží do stavového registra a vyšle signál INTA radiču prerušenia (2x po sebe); ak prerušenie povolené nie je pokračuje procesor ďalej v práci,
5. radič prerušenia po druhom signáli INTA pošle po dátovej zbernici *typ prerušenia* procesoru,
6. procesor prepočíta ukazovateľ do tabuľky prerušenia a po adresnej zbernici vyšle adresu ukazovateľa, pričom sa procesoru vráti vektor prerušenia uložený v danej tabuľke,
7. procesor pomocou prijatého vektora spracuje požiadavku o prerušenie, po vykonaní programu obsluhy prerušenia sa procesor vráti k pôvodnému programu, ktorého polohu v pamäti má uloženú v stavovom registri.

Inicializácia prerušovacieho podsystému

Počítače PC – XT až PC – 486 využívali 1 kiB z počiatku adresovaného priestoru pamäti ako tabuľku vektorov obslužných programov prerušení. Vojde sa do nej maximálne 256 štvor-bajtových položiek. Ak chce programátor zmeniť program obsluhy prerušenia musí na adrese daného vektora prepísať program.

Základný radič prerušenia sa adresuje ako vstupno-výstupné zariadenie (funkcie `inportb()`, `outportb()` v programovacom jazyku C) na adresách 020H, 021H (prvý radič) a 0A0H, 0A1H (druhý radič, takže tieto adresy pri modely PC-XT neplatia). Podrobnosti sú v nasledujúcej tabuľke (Tab. 2.5):

Tab. 2.5 Adresy registrov radiča prerušení

Adresa radiča 1. obvod	**Adresa radiča 2. obvod**	**Register**
20H	A0H	ICW1
21H	A1H	ICW2
21H	A1H	ICW3
21H	A1H	ICW4
21H	A1H	OCW1
20H	A0H	OCW2
20H	A0H	OCW3

Informácie zapisované do obvodu 8259A delíme na *inicializačné príkazové slová* ICW*n* (ICW – Initialization command word) a na operačné príkazové slová OCW*n* (operation command word), ktorými sa menia alebo nastavujú rôzne činnosti, ako napríklad maskovanie žiadosti o prerušenie, koniec prerušenia, zmena priority prerušenie, atď.

Inicializácia začína vždy zaslaním ICW1, následne ICW2 a ICW4. Výsledkom inicializácie pomocou BIOS-u je takéto nastavenie prerušovacieho systému:

- vstupné registre IRR reagujú na nábežnú hranu žiadosti o prerušenie,
- vynuluje sa register IMR,
- vstupu IR7 sa priradí najnižšia priorita 7,
- adresa podriadeného obvodu sa nastaví na 7,
- čítanie stavu sa nastaví na IRR a nuluje sa špeciálny režim masky.

Popis jednotlivých inicializačných príkazových slov:

- ICW1 (adresa 20H/A0H): umožňuje zadať akým spôsobom budú ovládané registre radiča prerušení t.j. či budú citlivé k nábežnej hrane alebo k vysokej úrovni. Implicitné nastavenie z BIOSu je k nábežnej hrane, ak sa použije nastavenie k vysokej úrovni, tak treba pred EOI deaktivovať vstup požiadavky,
- ICW2 (adresa 21H/A1H): deklaruje posuv požiadavky IRQ*n* na prerušenie INT(*n* + 8H),
- ICW3 (adresa 21H/A1H): definuje ku ktorému bitu Master 8259A je pripojený ďalší Slave 8259A,
- ICW4 (adresa 21H/A1H): definuje typ konca prerušenia, napríklad automatický EOI, normálny celkový EOI, špecifikovaný EOI a pod.

BIOS pri nábehu systému nastaví tieto inicializačné príkazové slová následne:

- ICW1 nastaví na 13H,
- ICW2 nastaví na 08H,
- ICW3 nenastavuje,
- ICW4 nastaví na 09H.

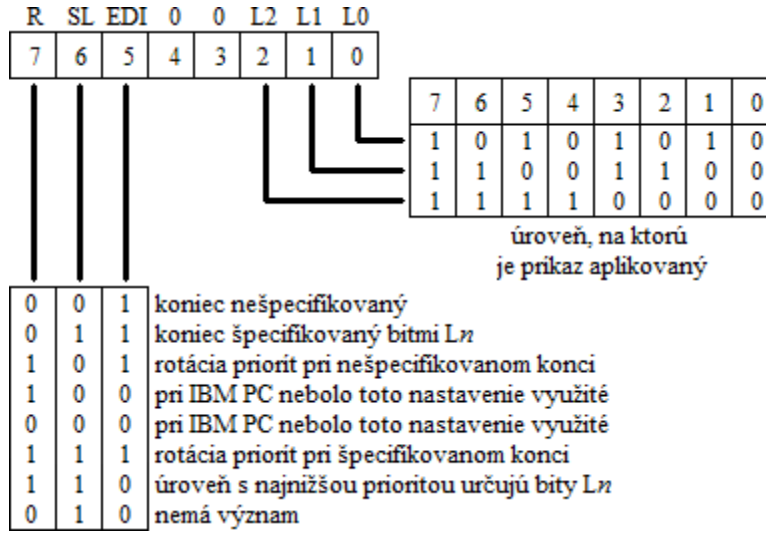
Radič prerušenia obsahuje technické prostriedky, ktorými sa dajú kedykoľvek po naprogramovaní zmeniť niektoré parametre prerušovacieho podsystému. Každú zo žiadostí o prerušenie je možné individuálne zamaskovať a tak selektívne zabrániť jej obsluhe. Deje sa tak zápisom masky na vstupno-výstupnú adresu 020H. Štruktúra masky je na obrázku Obr. 2.13

7	6	5	4	3	2	1	0
M7	M6	M5	M4	M3	M2	M1	M0

Obr. 2.13 Operačné príkazové slovo OCW1 (021H, 0A1H) – prerušovacia maska

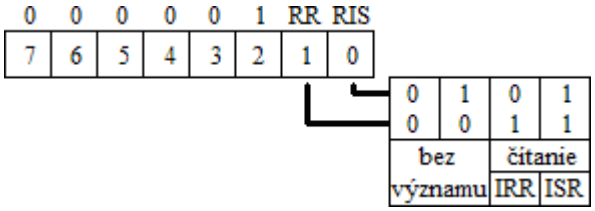
Ak sa nastaví príslušný bit masky na hodnotu 0, znamená to, že príslušná žiadosť je povolená. Ale naopak keď sa nastaví bit na hodnotu 1 je žiadosť maskovaná, teda nepovolená. Takže, ak je prerušenie maskované môže k nemu dôjsť a môže mať najvyššiu prioritu, ale k vykonaniu prerušenia bežiaceho programu nedôjde.

Pri inicializácií prerušovacieho podsystému nastaví BIOS obvody 8259A tak, že koniec spracovania prerušenia a teda pripravenosť prijatia novej žiadosti z rovnakého vstupu musí užívateľ (programátor) prerušovaciemu radiču explicitne oznámiť. Možnosť mu k tomu poskytuje operačné príkazové slovo OCW2, ktorým sa povinne signalizuje radiču prerušenia ukončenie spracovania daného prerušenia vynulovaním príslušného bitu v jeho registri ISR. Týmto slovom sa vykonávajú aj prípadné zmeny v priorite prerušovacích žiadostí. Možnosti tohto príkazového slova sú veľmi široké. Ukončiť sa dá nie len to prerušenie, ktoré sa práve spracováva, ale ktorékoľvek, ktoré sa špecifikuje číslom úrovne na najnižších troch bitoch zápisom konštanty. Formát slova OCW2 uvádza obrázok Obr. 2.14.



Obr. 2.14 Formát operačného príkazového slova OCW2 (020H, 0A0H)

Operačné príkazové slovo OCW3 sa dá využiť len pre čítanie stavu prerušovacieho podsystému, konkrétne čítanie registra *žiadosti o prerušenie* (IRR) a registra *žiadosti v obsluhe* (ISR). Zápisom tohto slova (`outportb`) sa však nastaví len režim čítania určitého slova registrov a následne sa požadovaná informácia vyčíta (`inportb`). Formát príkazového slova, z ktorého je zjavný spôsob vyberania konkrétneho stavového slova (registra), je na obrázku Obr. 2.15.



Obr. 2.15 Formát operačného príkazového slova OCW3 (020H, 0A0H)

Odporúča sa užívateľovi (programátorovi), aby pred ukončením práce obslužného programu prerušenia zaslal na adresu 20H, alebo A0H (podľa prerušenia) hodnotu 20H (`outportb(0x20,0x20)`, alebo `outportb(0xA0, 0x20)`). Týmto príkazom sa ukončí práve to prerušenie, ktoré sa práve obsluhuje. K zmenám v prioritách pritom nedôjde.

2.3.4 Prerušovací podsystém pri zbernici PCI

Zbernica PCI sa plne začala implementovať do Pentií. PCI obsahuje 4 IRQ signály, značené INTA, INTB, INTC a INTD. Tieto sú na platforme IBM PC mapované dodatočným obvodom (maticový prepínač) na pôvodné IRQ zbernice ISA (IRQ 9 až IRQ 12 a IRQ 15). Schéma mapovania sa konfiguruje mechanizmom PnP (Plug aNd Play), počiatočne nastavenie obstará BIOS, prípadne pridá ruku k dielu aj operačný systém.

V mnohých operačných systémoch je bežné, že niekoľko PCI zariadení zdieľa jedno prerušenie. Pri zachovaní pôvodnej logiky prerušovacieho podsystému to ani inač nešlo. Dokonca aj pôvodné ISA zariadenia pri strojoch PC–386 a PC–486 už museli kanály zdieľať. Pri inštalácií driver-a pre PCI zariadenie operačný systém priradí voľné prostriedky danému zariadeniu, takže už v tomto prípade nešlo natvrdo nastaviť zariadeniu presnú adresu prerušenia (IRQ), ktoré by používalo zariadenie pri každom PC.

Pri počítačoch, ktoré už obsahovali PCI boli hlavné obvody súčasťou Northbridge a Southbridge (viď kapitolu 3.7). Zaujímavosťou je že obvod PIC (Programmable Interrupt Controller – sústava obvodov 8259A) je súčasťou Southbridge, ktorý je mostom medzi ISA a PCI, takže signál samotného prerušenia musí z PCI putovať na nižšiu úroveň a až tak na tu najvyššiu, ktorým je CPU. Najskôr to tak bolo pre zachovanie spätnej kompatibility.

Signál INTA prestal pri tejto novej architektúre s PCI existovať a spolu zo signálom INTR bol zasielaný aj typ prerušenia.

Podľa špecifikácie PCI 2.2 môžu samotné koncové zariadenia (rozširujúce karty) žiadať o prerušenie priamo CPU pomocou *Message-signaled interrupt*, ale tento prístup sa začal plne využívať až pri zbernici PCI Express. Dôležitou zmenou, ktorá musela nastať, bolo zavedenie APIC-u. Týmto prístupom sa zvýšila efektivita obsluhy prerušení

APIC

Skratka APIC znamená Advanced Programmable Interrupt Controller ide o novú vývojovú generáciu radiča prerušení na platforme PC. Jeho najviditeľnejším prínosom je to,

že ruší bývalé obmedzenie šestnástich IRQ signálov. Najbežnejší APIC od firmy Intel, či už samostatné alebo integrované v čipsetoch, rozširujú počet IRQ signálov na 24. Špecifikácia APIC však žiadne takéto obmedzenie neobsahuje, takže na to nespolieha ani implementácia softvérovej obsluhy APIC-ov v operačných systémoch. Je možné skonštruovať systém so stovkami IRQ liniek

Praktickým dôsledkom nasadenia APIC-ov je, že nie je potrebné zdieľať IRQ signály, čo zrýchľuje odozvu systému na jednotlivé IRQ, znižuje záťaž CPU a odstraňuje možný zdroj nestability systému. K zníženiu záťaže rovnako prispieva vylepšená signalizácia, pri APIC-och už nie je potrebné potvrdzovať prijatie prerušenia a ukončenie jeho obsluhy.

V skutočnosti sa jedná o viacero obvodov ktoré vzájomne spolupracujú a dajú sa rozdeliť do dvoch skupín:

- I/O APIC – práve tento obvod je náhradou PIC, tento obvod združuje jednotlivé IRQ linky a informáciu o vyvolanom prerušení predáva jednotným spôsobom na procesor (procesory),
- Local APIC (LAPIC) – je integrovanou súčasťou procesora, firma Intel zavádzala LAPIC do svojich CPU už od prvých Pentií (presne od P24T, čo bola nadstavba pätice 486).

Po nasadení Pentií mal jeden počítač po jednom I/O APIC-u a LAPIC-u. Jeden procesor (jadro) obsahuje len jeden LAPIC, ale na matičnej doske môže byť viacero I/O APIC-ov. Architektúra APIC teda umožňuje aj distribuované konfigurácie s niekoľkými I/O APIC-mi a niekoľkými LAPIC-mi. Oba komplementárne časti spolu komunikujú po špeciálnej sériovej zbernici nazvanej *APIC bus*. I/O APIC zozbiera všetky žiadosti o prerušenie z jednotlivých elektrických signálov a zoradí ich podľa priorít, následne LAPIC zariadi obsluhu prerušenia na procesore. Udalosť nazvaná prerušenie teda už nie je prenášaná jednotlivými elektrickými signálmi, ale ako správa na zbernici APIC bus.

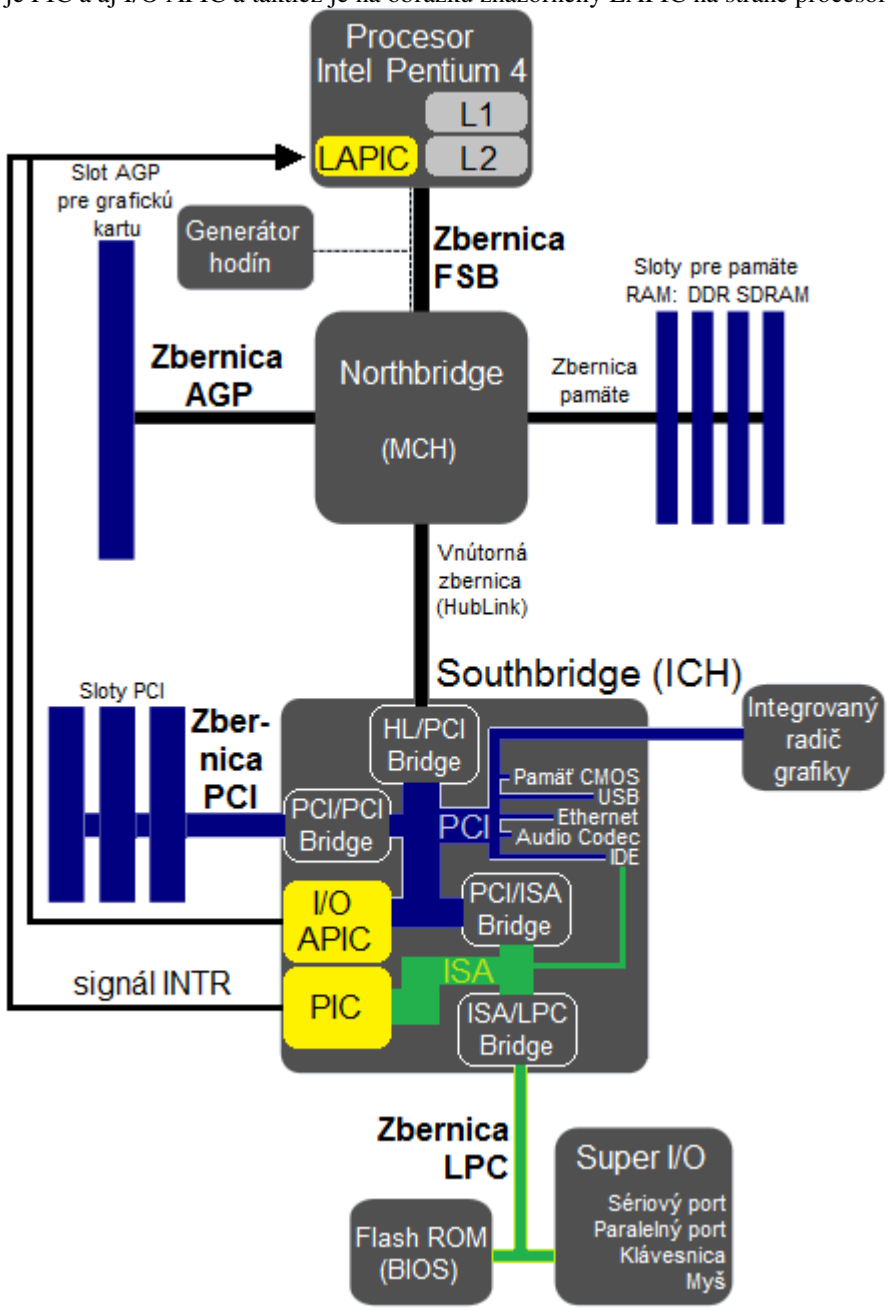
APIC je nutnou súčasťou viac-jadrových počítačov. Logické je to, aby jedno prerušenie obstarával len jeden procesor a jedine pomocou APICu sa dá smerovať prerušenie z rôznych IRQ liniek rôznym procesorom, čo je jediný možný prístup s ohľadom na rozkladanie záťaže.

Väčší počet I/O APIC-ov v systéme umožňuje obsluhovať väčší počet prerušení periférnych zariadení bez potreby zdieľať IRQ signály. Pokiaľ nestačí jeden čip I/O APIC s 24 alebo 32 vstupnými linkami, použije sa ich viac. Vedľajším dôsledkom je, že môžu byť bližšie k svojím perifériám, takže sa šetrí dĺžkou IRQ signálov a plochou dosky plošných spojov. Väčší počet I/O APIC-ov v systéme nepredstavuje problém, pretože sú prepojené spomínanou sériovou zbernicou APIC bus, na ktorej ma každý svoj unikátny identifikátor.

Kvôli spätnej kompatibilite dodnes stroje vybavené APIC-om obsahujú klasický PIC a štartujú s jeho pomocou, t.j. s prerušeniami obsluhovanými konvenčným spôsobom a mapovanými s obmedzením na pôvodnú sadu 16 IRQ signálov (viac-jadrové počítače z tohto dôvodu štartujú v uniprocesorovom režime). Použitie APIC-u je potrebné povoliť v BIOS-e a jeho vlastnú inicializáciu zariaďuje operačný systém. Takže ani v najmladšom DOS-e sa APIC neuplatní.

V APIC režime sa zníži výskyt zdieľaných prerušení, čo zvyšuje rýchlosť obsluhy prerušenia a zamedzuje plytvaním procesorovým časom. Zároveň v prípade APIC-u odpadajú dva potvrdzovacie cykly po zbernici PCI (jeden z nich dokonca až po ISA), ktoré boli dôležité pri pôvodnom PIC (sústava 8259A). Obsluha prerušenia pomocou APIC-u je po všetkých stránkach čistejšia, rýchlejšia a efektívnejšia ako to bolo tradičným spôsobom cez PIC.

Na obrázku Obr. 2.16 je podrobne znázornený Southbridge, ktorého súčasťou je PIC a aj I/O APIC a taktiež je na obrázku znázornený LAPIC na strane procesora.



Obr. 2.16 Prerušovací podsystém na matičnej doske PC so zbernicou PCI

APIC vs. ACPI

Popis zapojenia IRQ signálov od jednotlivých PCI a iných zariadení na rôzne vstupné piny jednotlivých I/O APIC-ov je na konkrétnej základnej doske uložený v jej ACPI

BIOS-e. ACPI hovorí aj do PnP, konkrétne do smerovania prerušenia. Od ACPI BIOS-u sa operačný systém pred inicializáciou APIC-u dozvie, ako sú prerušenia na základnej doske zapojené. Preto so môže tvrdiť, že vlastnosti APIC a ACPI síce nie sú totožné, ale prakticky spolu úzko súvisia.

LPC

Zaujímavá je zbernica LPC (viď kapitolu 3.8), cez ktorú sú pripojené pomalé periférie. Zbernica LPC obsahuje štvorbitovú dátovú cestu plus niekoľko riadiacich signálov, používa viacslovné „transakcie“ a na tomto základe emuluje zbernicu ISA. IRQ signály sa neprenášajú transakciami, ale jediným signálom SERIRQ, ktorý je privedený priamo na odpovedajúci kompatibilný vstup radiča prerušenia.

2.3.5 Prerušovací podsystém pri zbernici PCI Express

Konektor zbernice PCI-E vôbec nemá vodiče pre tradičné diskrétne signály prerušenia. Zbernica PCI-E totiž definitívne a dôsledne uplatňuje nový spôsob doručovania prerušenia na platforme PC: doručovanie pomocou správ (tzv. message-signaled interrupts).

Message-signaled interrupt sa objavili ako voliteľná vlastnosť už na klasickej paralelnej zbernici PCI od verzie 2.2, ale pretože paralelná PCI na platforme PC je VŽDY vybavená IRQ signálmi (INTA až INTD, v novších čipsetoch v kombinácií s I/O APIC), táto vlastnosť sa pri PCI prakticky nevyužívala.

Respektíve vlastne využívala. Súčasťou tejto pointy je odpoveď na otázku, ako sa v moderných Intelských čipsetoch doručuje prerušenie od I/O APIC-u (súčasť Southbridge) cez HubLink (emuluje PCI) a Northbridge na procesor, a tiež na otázku, prečo majú tieto čipsety v porovnaní s konkurenciou tak dlhú a nedeterministickú latenciu obsluhy (pod záťažou PCI latencia IRQ rastie). I/O APIC, ktorý je súčasťou Southbridge, posiela prerušenie smerom k procesoru v podobe správ na zbernici HubLink (PCI). Než odošle túto špeciálnu správu, čaká na samovoľné vyprázdnenie odosielacích buffer-ov rozhrania HubLink. Preto je latencia obsluhy IRQ väčšia, pokiaľ z PCI tečie do HubLink-u nejaká komunikácia.

Klasický I/O APIC doručuje IRQ udalostí smerom k procesoru prostredníctvom jednoúčelovej sériovej zbernice zvanej APIC Bus. Novou variantu IO/APIC-u, ktorá už pracuje s doručovaním pomocou správ po zbernici PCI, značí firma Intel veľmi často skratkou IO(x)APIC. Pochopiteľne v oboch prípadoch musí daný spôsob doručovania IRQ podporovať použitý procesor, respektíve jeho on-chip podsystém Local APIC.

Message-signaled interrupt-y sú v každom prípade softvérovo spätne kompatibilné, t.j. transparentné voči staršiemu softvéru, ktorý po ich existencií nepátra. Transparentný prevod message-signaled interrupt-ov na tradičný mechanizmus prerušenia IBM PC prebieha za asistencie host bridge (súčasť Northbridge) a predovšetkým modulu Local APIC, ktorý je súčasťou procesora od Pentia výše.

PCI Express žiadne IRQ signály neobsahuje (viď kapitolu 3.9). Preto pokiaľ jej prostredníctvom chceme sprostredkovať komunikáciu k procesoru z koncového zariadenia, je potrebné, aby IRQ signály tohto zariadenia obslúžil nejaký radič prerušení. I/O APIC-y prvej generácie so sériovou zbernicou v samostatnom puzdre sa však už prakticky nepoužívajú, a IO(x)APIC integrovaný v Southbridge-i je alokovaný IRQ signálmi od svojho vlastného PCI portu. Elegantným riešením je teda, integrovať potrebný IO/APIC do použitého PCI/PCI-E bridge-u. Konkrétne doska 6700PXH obsahuje dva IO(x)APIC-y, na každý port PCI-X jeden, ktoré majú spolu 16 diskrétnych IRQ vstupov.

Message-signaled interupt je vlastne dosť široký pojem, do ktorého spadá niekoľko rôznych stupňov, spôsobov, či formátov doručovania IRQ udalostí:

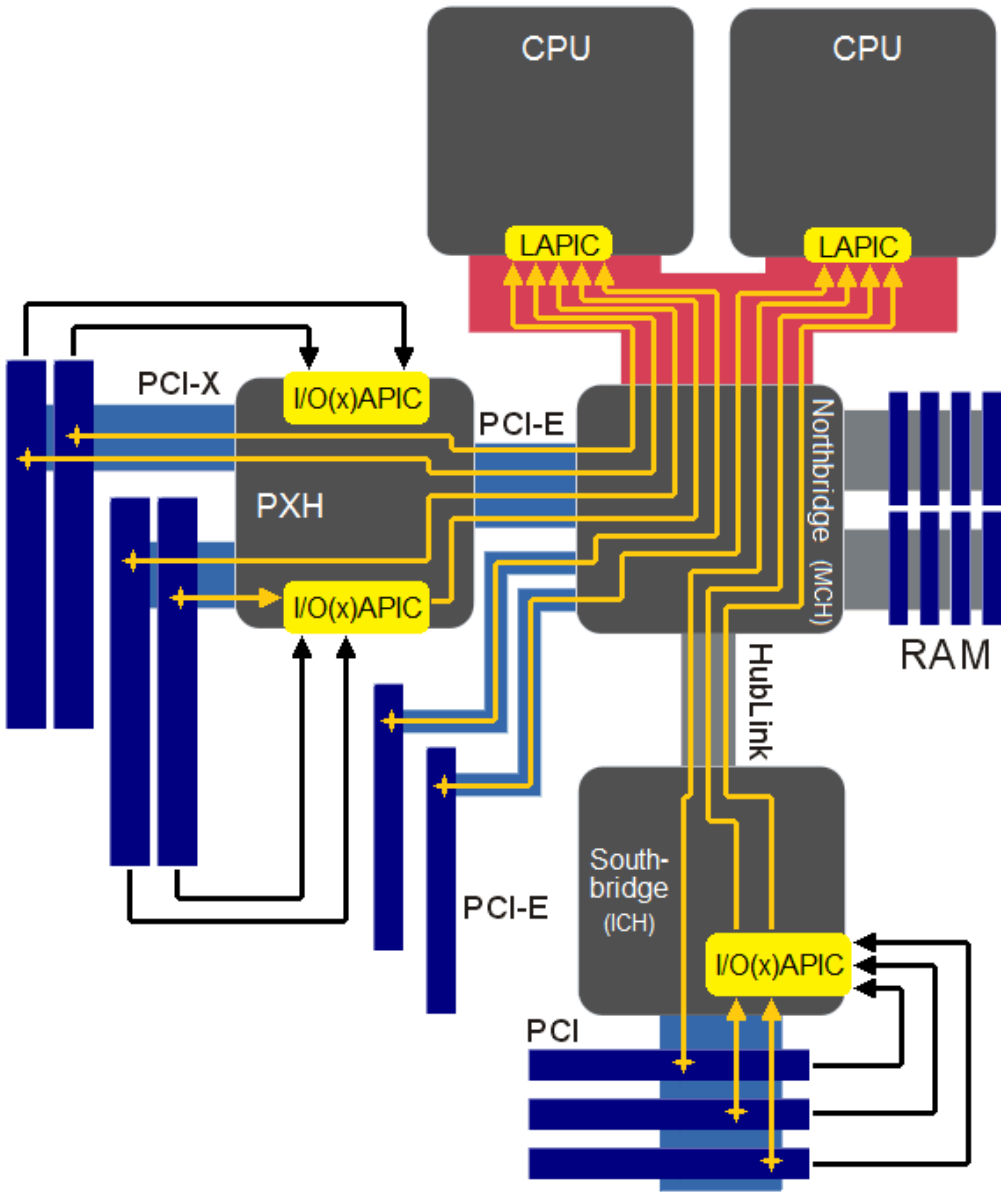
- Pôvodný message-signaled interrupt na paralelnej PCI je vlastne transakcia typu *bus-master memory write*. Periférnemu PCI zariadeniu sa predá adresa (v adresnom priestore zbernice), na ktorú má IRQ správy posielať, t.j. zapisovať. Táto adresa zhodou okolností patrí I/O APIC-u – tomu, ktorý je súčasťou Southbridge alebo novšie PCI/PCI-E bridge. Takže I/O APIC tieto koncové IRQ správy zachytí a sprostredkuje ich doručenie ďalej smerom k procesoru. Ak sa jedná o IO(x)APIC (pri PCIe), bude ich ďalej doručovať opäť pomocou *message-signaled interrupt* IRQ transakcií.
- IO(x)APIC používa trochu iný druh prerušovacích správ, ale princíp je podobný: po obdŕžaní prerušenia (tak isto ako vo vrchnom bode) IO(x)APIC spraví v zásade transakciu zápisu do pamäti, kde cieľovou adresou je adresa vyhradená pre tieto účely. Túto adresu sleduje procesor, resp. jeho Local APIC. Nie je to tak, že by procesor pravidelne kontroloval konkrétne pamäťové miesto v externej DRAM. Na doručenie IRQ správ samotnému procesoru sa totiž podieľa aj Northbridge, ktorý vie, že správy tohto typu má doručovať priamo procesoru. Takže spomínaná pamäťovo mapovaná oblasť je umiestnená priamo v procesore, a jeho Local APIC dostane udalosť priamo.
- Presnejšie povedané, k procesoru vlastne už nedorazí ani PCI transakcia pre zápis do pamäti. Northbridge ju totiž letmo preloží na takzvanú *IRQ transakciu*, definovanú v rámci protokolu FSB. Multiprocesorový Northbridge dokonca inteligentne distribuuje IRQ udalosti niekoľkým procesorom podľa priority. Tabuľku mapovania IRQ na jednotlivé procesory si Northbridge pravidelne aktualizuje.
- IO(x)APIC teda zbiera IRQ jednak z diskrétnych vodičov, jednak z *message-signaled interrupt* správ od koncových zariadení, a spracováva ich do vlastného formátu správ, ktoré zasiela ďalej. Samozrejme aj tu je alternatívna možnosť: IO(x)APIC, resp. PCI bridge ktorého je I/O APIC súčasťou, môže periférne správy *message-signaled interrupt* priamo preposielať procesoru (bez medzispracovania).
- V rámci PCIe sú *IRQ transakcie* súčasťou širšej triedy transakcií typu *message* (táto trieda je novinkou oproti tradičnej zbernici PCI). Natívne periférne zariadenie s rozhraním PCIe môžu tento druh IRQ zasielať priamo, avšak PCI/PCIe bridge (resp. jeho IO(x)APIC) tento druh správ *message-signaled interrupt* tiež využíva.
- Navyše k tomu všetkému používa prinajmenšom zbernica PCIe taktiež IRQ správy typu *Legacy INTx* - tieto správy prídu k slovu v prípade, že systém je nakonfigurovaný aby nepoužíval APIC. Používajú sa pri štarte systému, kedy APIC ešte nie je inicializovaný, a umožňuje taktiež kompatibilitu so starším softvérom, ktorý APIC vôbec nepozná.

Do celého systému doručovania IRQ sa môže vložiť softvér, ktorý si je vedomí správ *message-signaled interrupt* a to takýmto spôsobom, že aktivácia IRQ sa v softvéry prejaví asynchrónnym volaním obslužnej rutiny. Operační systém by mal ponúkať jednotné interné API pre registráciu obslužných rutín, a pokiaľ má tu možnosť, mal by sa snažiť o minimalizáciu zdieľaných IRQ signálov, pretože zdieľanie IRQ znižuje výkon.

O konfiguráciu dvoch úrovní APIC-ov a niekoľko transformácii pôvodných udalostí prerušenia cez *„čísla IRQ“* až po vektory obslužných rutín sa stará ACPI BIOS a na neho tesne naviaže operačný systém. Operačný systém môže maximálne upraviť konfiguráciu

mapovania v APIC-och. V praxi sa však skôr musí spoliehať, že ACPI tabuľky v BIOS-e popisujúce spôsob prepojenia jednotlivých diskrétnych IRQ signálov na APIC-y sú v poriadku.

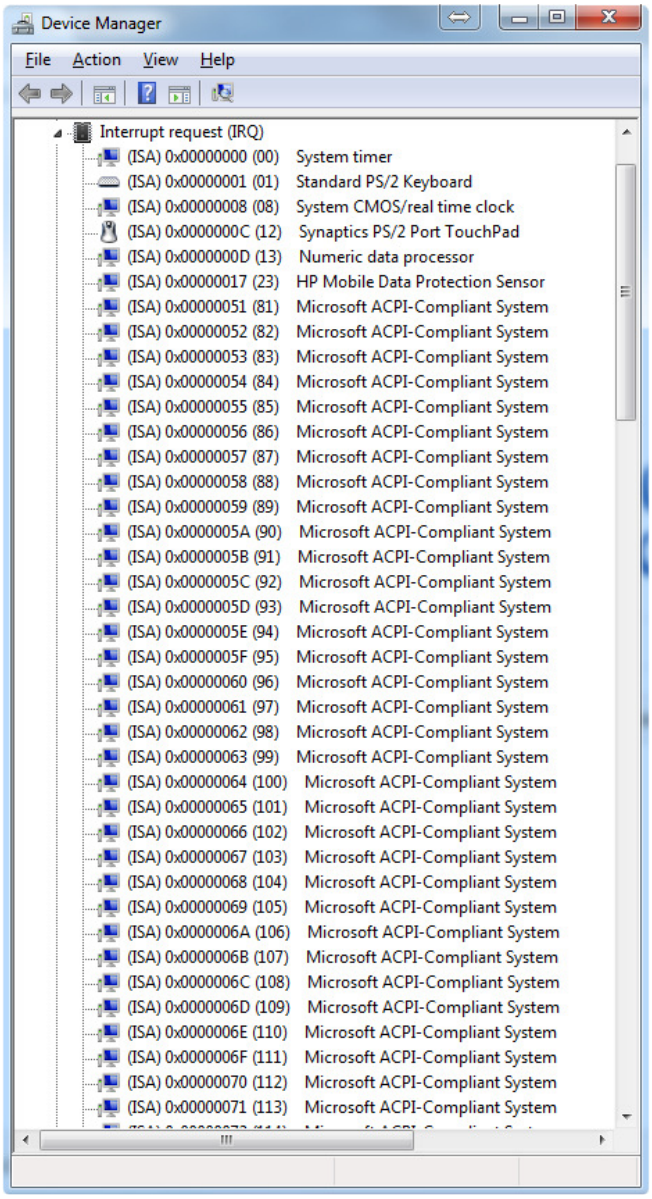
Na obrázku Obr. 2.17 je znázornený prerušovací podsystém počítača, ktorý obsahuje zbernicu PCI Express. Pričom na obrázku sú znázornené správy *message-signaled interupt* (oranžová farba), kde je viditeľné, že nie všetky správy musia byť sprostredkované pomocou I/O(x)APIC. Zariadenia pripojené k PCI Express dokonca ani nemajú pripojenie k I/O(x)APIC a každú správu (*message-signaled interupt*) smerujú priamo k LAPIC. Čiernou farbou sú znázornené linky (vodiče) pre žiadosti o prerušenie (IRQ).



Obr. 2.17 Prerušovací podsystém na matičnej doske PC so zbernicami PCI a PCI Express

2.3.6 Adresovanie prerušovacieho podsystému

Tak ako adresovanie pamäte a vstupno-výstupného podsystému aj prerušovací podsystém sa dá v dnešných OS od firmy Microsoft prezrieť pomocou nástroja Správca zariadení (Device manager). Ktorý je umiestnený medzi nástrojmi Ovládacieho panela (Control Panel). V nástroji Správca zariadení sa v hornom menu v položke „Zobraziť“ („View“) vyberie „Prostriedky podľa typu“ („Resources by type“) alebo „Prostriedky podľa pripojenia“ („Resources by connection“) a následne sa rozbalí zložka „Požiadavky o prerušenie“ („Interrupt request (IRQ)“). Na obrázku Obr. 2.18 je príklad takéhoto prerušovacieho podsystému v OS Windows 7 pre konkrétny model PC.



Obr. 2.18 Časť prerušovacieho podsystému znázorneného pomocou Správcu zariadení (Device Manager)

2.3.7 Programovanie prerušovacieho podsystému

Táto podkapitola sa bude venovať programovaniu prerušovacieho podsystému. Konkrétne metóde prepisu obslužného programu (rutiny) prerušenia za vlastný program. Pri programovaní prerušení je potrebné poznať jeden typ premennej, ktorým sa bude dať deklarovať samotný program prerušenia, tento typ je zadefinovaný v knižnici *dos.h*:

```
interrupt
```

A taktiež treba poznať dva príkazy (funkcie) z knižnice *dos.h*:

```
interrupt prerusenie = getvect(int adresa);
 setvect(int adresa, interrupt prerusenie)
```

Funkcia `getvect()` stiahne pomocou adresy (`adresa`) z tabuľky vektorov obslužný program prerušenia do danej premennej (`prerusenie`). Funkcia `setvect()` prepíše obslužný program prerušenia pomocou adresy (`adresa`) tabuľky vektorov a prepíše ho za program vložený v premennej `prerusenie`. Typ deklarácie `interrupt` deklaruje premennú typu prerušenia, čiže daná premenná bude nositeľkou samotného programu prerušenia.

Pri prepisovaní obslužných programov prerušení je potrebné brať ohľad na operačný systém, takže pri programovaní prerušení, ktoré využíva operačný systém je potrebné pred skončením vytváraného programu prepísať obslužný program prerušenia späť na pôvodný obslužný program. Samozrejme ak sa jedná o nový hardvér, ktorého prerušenie sa bude vedome využívať aj po skončení vytváraného programu, vtedy sa spomínaný krok nespraví. Pri novom hardvéri je potrebné vedieť ktoré prerušenie vyvoláva a keď je aj hardvér navrhovaný a realizovaný (nové vstupno-výstupné zariadenie pre zbernicu ISA) je potrebné vedieť, že sa preto vyhradené IRQ kanály a to konkrétne IRQ 9 až IRQ 12 a IRQ 15 (viď tabuľku Tab. 2.3).

V ďalšej časti podkapitoly budú riešené vzorové zadania v programovacom jazyku C, tieto zadania budú určené pre prepisovanie obslužných programov prerušenia a taktiež budú tieto riešenia vysvetlené.

Zadanie 1:

Prepíšte obslužný program prerušenia delenia nulou tak, aby program pri delení nulou nespadol (klasické ošetrenie delenia nulou pomocou prerušenia) ale napísal hlásenie „Vysledok je nekonecno“. Tento nový obslužný program prerušenia aj otestujte tak, aby Váš základný program uprostred numerických operácií delil nulou. Nakoniec prepíšte obslužný program prerušenia za starý obslužný program.

Riešenie:

```
1    #include<stdio.h>
2    #include<conio.h>
3    #include<dos.h>
4
5    void interrupt (*old_v)();
6    void interrupt obsluha();
7
8    int i;
```

```
9    int j;
10
11   void main()
12   {
13        old_v = getvect(0x0);
14        setvect(0x0, obsluha);
15        clrscr();
16        for(i = -10; i < 10; i++)
17        {
18           j = 10/i;
19           printf("i=%d j=%d\n",i,j);
20        }
21        setvect(0x0,old_v);
22        getch();
23   }
24
25   void interrupt obsluha()
26   {
27        puts("Vysledok je nekonecno !!!!");
28        i=1;
29   }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých knižníc k programu.

5, 6: Deklarácia premenných typu interrupt, ktoré budú neskôr obsahovať starý a nový obslužný program prerušenia.

8,9: Deklarácia premenných pre numerické operácie ktoré sú žiadané v zadaní tejto úlohy.

11: Deklarácia a definovanie hlavného programu.

13: Do premennej `old_v` sa vloží program, na ktorý odkazuje tabuľka vektorov na adrese 0H (viď tabuľku Tab. 2.2).

14: Prepísanie starého obslužného programu programom uloženým v premennej `obsluha`. K prepísaniu dôjde na mieste, kde odkazuje tabuľka vektorov na adrese 0H (viď tabuľku Tab. 2.2).

15: Vyčistenie obrazovky.

16: Začiatok cyklu, ktorý bude testom novej obsluhy prerušenia podľa zadania.

18: Numerické operácie, ktoré boli žiadané v zadaní. Sú programovo upravené tak, aby uprostred (po jedenástich iteráciách) nastalo delenie nulou.

19: Výpis výsledku numerickej operácie na obrazovku.

21: Prepísanie nového obslužného programu starým programom uloženým v premennej `old_v`.

22: Čakanie na stlačenie klávesa (aby sa program nevypol hneď a bol viditeľný výstup).

25: Definovanie novej obsluhy prerušenia.

27: Výpis žiadaného textu (podľa zadania).

28: Prepísanie premennej, ktorá obsahuje hodnotu 0 na hodnotu 1. Tento prepis sa deje preto, aby inštrukcia delenia nezacyklila program, nakoľko pri iterácií o hodnotu 0 sa inštrukcia delenia nepohne z miesta. Toto je aj dôvod prečo obsluha prerušenia delenia 0 existuje.

Zadanie 2:

Prepíšte obslužný program prerušenia časovača tak, aby po každom vyvolanom prerušení od časovača zapísal systémový čas do súboru. Hlavný program sa po piatich sekundách vypne (91 prerušení od časovača).

Poznámka: Frekvencií a nastavovaniu časovača sa bude venovať iná kapitola.

Riešenie:

```
1    #include<stdio.h>
2    #include<conio.h>
3    #include<dos.h>
4
5    void interrupt (*old_v)();
6    void interrupt obsluha();
7
8    FILE *subor;
9    int poc = 0;
10   struct time cas;
11
12   void main()
13   {
14        old_v = getvect(0x8);
15        setvect(0x8, obsluha);
16        clrscr();
17        printf("Zapisujem cas do suboru z.dat...");
18        subor=fopen("z.dat","a");
19        while(poc<91);
20        setvect(0x8,old_v);
21        fclose(subor);
22   }
23
24   void interrupt obsluha()
25   {
26        poc++;
27        (*old_v)();
28        time(&cas);
29        fprintf(subor,"%s",ctime(&cas));
30   }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých knižníc k programu.

5, 6: Deklarácia premenných typu interrupt, ktoré budú neskôr obsahovať starý a nový obslužný program prerušenia.

8-10: Deklarácia súboru (`subor`), počítadla prerušení od časovača (`poc`) a systémového času (`cas`).

12: Deklarácia a definovanie hlavného programu.

14: Do premennej `old_v` sa vloží program, na ktorý odkazuje tabuľka vektorov na adrese 8H (viď tabuľku Tab. 2.2).

15: Prepísanie starého obslužného programu programom uloženým v premennej `obsluha`. K prepísaniu dôjde na mieste, kde odkazuje tabuľka vektorov na adrese 8H (viď tabuľku Tab. 2.2).

16: Vyčistenie obrazovky.
17: Vypísanie textu *„Zapisujem cas do suboru z.dat...“* na obrazovku počítača.
18: Otvorenie súboru *z.dat*.
19: Čakanie kým je premenná `poc` (počítadlo prerušení od časovača) menšia ako 91 (prázdny cyklus *while*).
20: Prepísanie nového obslužného programu starým programom uloženým v premennej `old_v`.
21: Zatvorenie súboru *z.dat*.
24: Definovanie novej obsluhy prerušenia.
26: Inkrementácia premennej `poc` – počítadla prerušení od časovača.
27: Spustenie celého starého programu obsluhy prerušenia od časovača.
28: Načítanie systémového času do premennej `cas`.
29: Zapísanie systémového času do súboru *z.dat*.

Zadanie 3:

Bola vytvorená jednoduchá vstupno-výstupná karta s tlačidlom, ktorého stlačenie vyvoláva prerušenie IRQ 3. Vytvorte program ktorý pri stlačení tohto tlačidla vypíše na obrazovku „stlacene tlacitko!!“ a do súboru *cas.dat* zapíše systémový čas stlačenia tohto tlačidla.

Riešenie:

```
1    #include<dos.h>
2    #include<stdio.h>
3    #include<time.h>
4
5    void interrupt(* stare_pr)(void);
6    void interrupt(nove_pr)(void);
7
8    void interrupt nove_pr(void)
9    {
10        FILE  *fp;
11        time_t t;
12        outportb(0x21, (inportb(0x21)|0x08));
13        printf("stlacene tlacitko!! \n");
14        time(&t);
15        if((fp=fopen("cas.dat","a")) !=NULL)
16             fprintf(fp,"%s",ctime(&t));
17        fclose(fp);
18        outportb(0x21,(inportb(0x21)&0xf7));
19        outportb(0x20,0x20);
20   }
21
22   void main(void)
23   {
24        int i;
25        stare_pr = getvect(0x0b);
26        setvect(0x0b,nove_pr);
27        outportb(0x21,(inportb(0x21)&0xf7));
28        printf("\n\n");
```

```
29        for (i=0;i<100;i++) delay(200);
30        setvect(0x0b,stare_pr);
31    }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých knižníc k programu.
5,6: Deklarácia premenných typu interrupt, ktoré budú neskôr obsahovať starý a nový obslužný program prerušenia.
8: Definovanie novej obsluhy prerušenia.
10,11: Deklarácia súboru (`fp`) a premennej systémového času (`t`).
12: Zákaz prerušenia od IRQ3 (pomocou masky operačného riadiaceho slova OCW1).
13: Výpis textu „*stlacene tlacitko!!*“.
14: Načítanie systémového času do premennej *t*.
15,16: Otvorenie súboru *cas.dat*, ak sa otvorenie podarilo, tak sa do súboru zapíše systémový čas.
17: Zatvorenie súboru *cas.dat*.
18: Povolenie prerušenia od IRQ3 (pomocou masky operačného riadiaceho slova OCW1).
19: Nešpecifikované ukončenie prerušenia (OCW3).
22: Deklarácia a definovanie hlavného programu.
24: Deklarácia premennej `i` pre pomocný cyklus.
25: Do premennej `stare_pr` sa vloží program, na ktorý odkazuje tabuľka vektorov na adrese BH (viď tabuľku Tab. 2.2).
26: Prepísanie starého obslužného programu programom uloženým v premennej `nove_pr`. K prepísaniu dôjde na mieste, kde odkazuje tabuľka vektorov na adrese BH (viď tabuľku Tab. 2.2).
27: Povolenie prerušenia od IRQ3 (pomocou masky operačného riadiaceho slova OCW1).
28: Vloženie nových riadkov na obrazovku počítača.
29: Pomocný cyklus ktorý sa vykoná 100 krát a za každým bude čakať 200 ms. V podstate tento riadok má za úlohu podržať beh programu na 20 sekúnd.
30: Prepísanie nového obslužného programu starým programom uloženým v premennej `stare_pr`.

Programátor okrem prepisovania programov obslúh prerušení, môže tieto programy aj využiť a to volaním obsluhy prerušenia s parametrom pomocou funkcie (príkazu):

```
int86()
```

ktorá je súčasťou knižnice *dos.h* a presná syntax vyzerá následne:

```
union REGS r;
r.h.al = int hodnota;
r.h.ah = int parameter;
r.x.dx = int id;
int86(int adresa,&r,&r);
return(r.h.ah);
```

Vysvetlenie jednotlivých riadkov syntaxe:

- deklarácia parametrov pre funkciu `int86()`,
- parameter `r.h.al` vyjadruje hodnotu, ktorú chceme zapísať ako parameter obsluhy prerušenia,
- parameter `r.h.ah` vyjadruje poradie parametra obsluhy prerušenia,
- parameter `r.x.dx` vyjadruje identifikátor zariadenia (napr.: LPT1, LPT2, alebo COM1, COM2),
- volanie funkcie knižnice *dos.h* `int86()` jej prvým parametrom je adresa obslužného programu v tabuľke vektorov, druhým sú v podstate prednastavené parametre `r.h.al`, `r.h.ah` a `r.x.dx`, a tretím je výstup funkcie,
- ak má obsluha výstup tak je najčastejšie zapísaná v `r.h.ah`.

Zadanie 4:

Naprogramujte program, ktorý pomocou rozhrania Centronics (LPT) a ihličkovej tlačiarne vytlačí text:
„Dobry den,

Srdecne Vas zdravim.“.

K naprogramovaniu použite obslužný program prerušenia od tlačiarne (paralelného portu) INT 17.

Analýza zadania:

Keďže v tomto programe je potrebné využívať funkciu `int86()`, tak je nutné prilinkovať k programu okrem knižnice *stdio.h* aj knižnicu *dos.h*. Pred programovaním funkcií a hlavného programu je potrebné zadeklarovať globálnu premennú typu *union REGS* pre prácu s funkciou `int86()`.

Program pre lepšiu prehľadnosť využije niekoľko funkcií:

- vyslanie znaku pre tlač,
- zistenie stavu tlačiarne,
- inicializácia tlačiarne,
- tlač reťazca.

Funkcia na vyslanie znaku pre tlač v podstate len nastaví parametre služby 00H (tlačenie znaku) v obsluhe prerušenia paralelného portu (INT 17H). Konkrétne nastaví znak určený pre tlač (*al*), číslo tlačiarne (*dx*) a poradie služby (*ah*), potom sa tieto parametre pošlu na spracovanie pomocou funkcie `int86()`.

Zistenie stavu tlačiarne bude tiež len jednoduchá funkcia, ktorá nastaví parametre služby 02H v obsluhe prerušenia INT 17H, ale táto funkcia bude mať aj návratovú hodnotu. V tomto prípade sa nastavia iba dva parametre a to číslo tlačiarne (*dx*) a poradie služby (*ah*). Parametre sa vložia do funkcie `int86()` a výstup bude v parametry *ah*, kódovanie (poradie bitov) je znázornené na obrázku Obr. 7.30. Tento výstup sa nastaví ako návratová hodnota funkcie.

Inicializáciu tlačiarne bude zabezpečovať funkcia, ktorá nastáví dva parametre číslo tlačiarne (*dx*) a poradie služby (*ah*). Následne už iba pomocou funkcie `int86()` sa táto služba spustí a zinicializuje tlačiareň.

Funkcia určená pre tlač reťazca už bude o čosi zložitejšia. Vstupným argumentom funkcie bude reťazec (pole znakov). Pomocou cyklu s podmienkou na začiatku sa prejde celým reťazcom. V cykle sa najprv zistí stav tlačiarne pomocou funkcie, ktorá bude nato určená. Ak došiel papier vypíše sa o tom hlásenie a program bude čakať na vloženie papiera. Keď sa počas čakania na papier stlačí ľubovoľný kláves, tak sa aplikácia vypne. V prípade, že sa vloží papier alebo papier vôbec nedošiel, tak sa budú vysielať jednotlivé znaky pre tlač každých 50ms v spomínanom cykle pomocou funkcie na vyslamie znaku pre tlač.

Hlavný program najskôr vyčistí obrazovku, zinicializuje tlačiareň, zistí stav tlačiarne. Ak je tlačiareň v chybe, tak sa daná chyba vypíše a to pomocou usporiadania bitov (kódu) v premennej stavu tlačiarne (Obr. 7.30). V prípade, že tlačiareň v chybe nie je spustí sa funkcia určená pre tlač reťazca.

Riešenie:

```
1    #include <dos.h>
2    #include <stdio.h>
3    union REGS r;
4
5    void vysli(int c)
6    {
7         r.h.al=c;
8         r.h.ah=0;
9         r.x.dx=0;
10        int86(0x17,&r,&r);
11   }
12
13   int stav_tlac(void)
14   {
15        r.h.ah=2;
16        r.x.dx=0;
17        int86(0x17,&r,&r);
18        return(r.h.ah);
19   }
20
21   void init(void)
22   {
23        r.h.ah=1;
24        r.x.dx=0;
25        int86(0x17,&r,&r);
26   }
27
28   void tlac_ret(char *str)
29   {
30        int stav;
31        stav = stav_tlac();
32        while(*str != 0)
33        if((stav = stav_tlac())&0x20)
```

```
34        {
35             printf("Dosiel papier, cakam nan !\n");
36             while(stav_tlac()&0x20) if(kbhit())
37             {
38                  getch();
39                  exit(1);
40             }
41        }
42        else
43        {
44             delay(50);
45             vysli(*str++);
46        }
47   }
48
49   void main(void)
50   {
51        int stav;
52        clrscr();
53        init();
54        stav = stav_tlac();
55        if(stav&0x40)
56        {
57             printf("\nTlaciaren nepripojena !\n");
58             exit(1);
59        }
60        if(stav&0x01)
61        {
62             printf("\nTime out");
63             exit(1);
64        }
65        if(stav&0x04)
66        {
67             printf("\nChyba vstupno-vystupneho zariadenia");
68             exit(1);
69        }
70        if(stav&0x20)
71        {
72             printf("\nTlaciaren nepripravena");
73             exit(1);
74        }
75        tlac_ret("Dobry den,\n\n\nSrdecne Vas zdravim.\n");
76   }
```

Vysvetlenie:

1-2: Prilinkovanie dôležitých knižníc k programu.

3: Deklarácia parametrov pre funkciu `int86()`.

5: Deklarácia a definovanie funkcie `vysli()`.

7: Vloženie znaku (vstupného parametra funkcie `vysli()`) do parametra obsluhy prerušenia.
8: Určenie nultého parametra (znak určený k tlačeniu), ako vstupného parametra obsluhy prerušenia.
9: Vybratie nultej tlačiarne (LPT1).
10: Zaslanie parametrov obsluhe prerušenia tlačiarne a spustenie tejto obsluhy s danými parametrami.
13: Deklarácia a definovanie funkcie `stav_tlac()`.
15: Určenie druhého parametra (žiadosť o návrat stavu tlačiarne), ako vstupného parametra obsluhy prerušenia.
16: Vybratie nultej tlačiarne (LPT1).
17: Zaslanie parametrov obsluhe prerušenia tlačiarne a spustenie tejto obsluhy s danými parametrami.
18: Vrátenie výstupnej premennej obsluhy prerušenia (vrátenie stavu tlačiarne).
21: Definovanie funkcie `init()`.
23: Určenie prvého parametra (žiadosť o inicializáciu tlačiarne), ako vstupného parametra obsluhy prerušenia.
24: Vybratie nultej tlačiarne (LPT1).
25: Zaslanie parametrov obsluhe prerušenia tlačiarne a spustenie tejto obsluhy s danými parametrami (inicializácia tlačiarne).
28: Deklarácia a definovanie funkcie `tlac_ret()`.
30: Deklarácia premennej `stav` (stav tlačiarne).
31: Do premennej `stav` sa vloží návratová hodnota funkcie `stav_tlac()`.
32-46: Cyklus ktorý prejde celým vstupným argumentom po jednom znaku.
  33: Podmienka ktorá kontroluje stav papiera v tlačiarni (či v tlačiarni nedošiel papier).
  35: Ak v tlačiarni došiel papier, tak sa na obrazovke zobrazí hlásenie o tej skutočnosti.
  36-40: Počas vkladania papiera čaká program na stlačenie ktoréhokoľvek klávesa, ak sa kláves stlačí, tak sa program vypne, inač po vložení papiera bude hlavný cyklus pokračovať v svojej činnosti.
  42-46: Ak je v tlačiarni papier, tak každých 50 ms sa spustí funkcia `vysli()` s argumentom jedného znaku, ktorý je práve predmetom hlavného cyklu.
49: Deklarácia a definovanie hlavného programu.
51: Deklarácia premennej `stav` (stav tlačiarne).
52: Vyčistenie obrazovky.
53: Spustenie funkcie `init()`.
54: Do premennej `stav` sa vloží návratová hodnota funkcie `stav_tlac()`.
55-59: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Tlačiareň nepripojená!*“ a následné vypnutie programu.
60-64: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Time out*“ a následné vypnutie programu.
65-69: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Chyba vstupno-výstupného zariadenia*“ a následné vypnutie programu.
70-74: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Tlačiareň nepripravená*“ a následné vypnutie programu.
75: Spustenie funkcie `tlac_ret()` s argumentom „Dobry den,\n\n\nSrdecne Vas zdravim.\n“.

2.4 Podsystém priameho prístupu do pamäte

Úvodom sa vysvetlí pojem *vstupno-výstupný prenos*. Pod vstupno-výstupným prenosom sa rozumie prenos údajov medzi periférnym zariadením a CPU alebo medzi periférnym zariadením a pamäťou.

Riadenie zbernice PC počas V/V prenosu:

Podľa toho, čo riadi zbernicu PC počas prenosu údajov z/do periférneho zariadenia rozdeľujeme V/V prenosy na:

a) prenosy s účasťou procesora,
b) prenosy bez účasti procesora.

Prenosy s účasťou procesora:

Pri V/V prenosoch s účasťou procesora generuje riadiace signály zbernice procesor. Týmto spôsobom sa vykonáva prenos jednotlivých údajov a údaje sa prenášajú *medzi procesorom a V/V zariadením*. Vykonať prenos údajov z/do periférneho zariadenia nemusí byť možné v každom okamihu.

Podľa toho, akým spôsobom sa rozhodne o okamihu odštartovania prenosu údajov rozlišujeme tieto V/V prenosy:

I. nepodmienený V/V prenos,
II. podmienený V/V prenos.

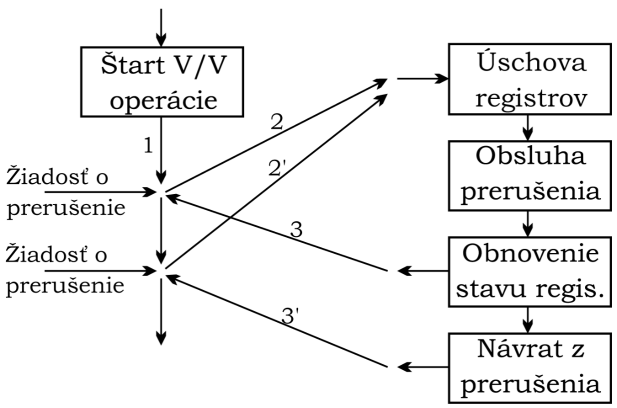
I. Nepodmienený V/V prenos:

Procesor implicitne považuje V/V zariadenie v ľubovoľnom okamihu za pripravené na prenos, t.j. kedykoľvek môže z vstupného zariadenia údaj načítať a do výstupného zariadenia kedykoľvek údaj zapísať. Prenos je rýchly, pretože sa vykoná rýchlosťou procesora (načítanie stavových slov).

II. Podmienený V/V prenos:

Procesor pred vlastným vykonaním prenosu údajov najskôr testuje pripravenosť zariadenia prijať resp. vyslať údaje. Procesor zisťuje pripravenosť takým spôsobom, že z adaptéra načíta stavové slovo, v ktorom bity nesú informáciu o pripravenosti zariadenia. Len keď je zariadenie k prenosu pripravené, procesor vykoná vlastný prenos údajov. Prenos rešpektuje pracovnú rýchlosť zariadenia (procesor veľa času strávi čakaním).

Nevýhody podmieneného prenosu odstraňuje V/V prenos s prerušením. Princíp riadenia V/V prenosu s využitím prerušení je na obrázku Obr. 2.19.



Obr. 2.19 Princíp riadenia V/V prenosu s využitím prerušení

Pri tomto spôsobe prenosu procesor netestuje pred prenosom údajov pripravenosť V/V zariadenia, ale V/V zariadenie v prípade svojej pripravenosti k prenosu vygeneruje žiadosť o prerušenie.

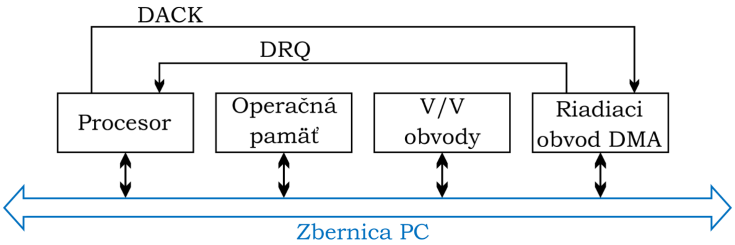
Procesor preruší práve prebiehajúci program a v rámci obslužného programu prerušenia (bez otestovania pripravenosti zariadenia) vykoná vlastný prenos údajov. Po jeho skončení pokračuje v prerušenom programe. Ak je periférne zariadenie schopné prijať/vyslať ďalší údaj, opäť vygeneruje novú žiadosť o prerušenie. Je úplne vylúčené neefektívne čakanie CPU.

Prenosy bez účasti procesora:

Vyznačujú sa tým, že počas prenosu údajov riadi zbernicu počítača riadiaci obvod DMA a procesor je od zbernice odpojený (má svoje výstupy v stave vysokej impedancie). Údaje sa prenášajú medzi pamäťou a V/V zariadením. Tento spôsob prenosu sa nazýva *priamy prístup do pamäte* (DMA – Direct Memory Access) a typicky sa používa blokový prenos údajov, napr. pri práci s pevným diskom. Ako riadiaci obvod DMA je použitý buď špecializovaný programovateľný obvod alebo špeciálny V/V procesor. Tento spôsob prenosu sa vykonáva vtedy, ak riadiaci obvod DMA vykonáva prenos údajov rýchlejšie ako procesor.

Procesor odovzdá riadiacemu obvodu DMA požiadavky na prenos a ďalej už len čaká na jeho vykonanie. Vstupno-výstupný prenos, aj keď je bez účasti procesora, je inicializovaný procesorom. Pred vlastným uskutočnením prenosu údajov musí procesor oznámiť riadiacemu obvodu DMA požiadavky na prenos, teda odkiaľ a kam sa majú údaje prenášať a koľko ich má byť. Až potom riadiaci obvod DMA požiada procesor o pridelenie zbernice a keď mu ju procesor pridelí, vykoná vlastný prenos údajov. Po ukončení prenosu vráti riadenie zbernice späť procesoru, ktorý môže pokračovať v činnosti.

Na obrázku Obr. 2.20 je typická konfigurácia PC so zbernicovou architektúrou. Je vyznačené pripojenie riadiaceho obvodu DMA, pripojenie signálu žiadosti o zbernicu DRQ (DMA Request) z riadiaceho obvodu DMA do CPU, ako aj pripojenie signálu o pridelení zbernice DACK (DMA Acknowledge) z procesora do riadiaceho obvodu DMA.



Obr. 2.20 Počítač s riadiacim obvodom DMA

Podporná vlastnosť počítačov pomocou ktorej údaje z periférnych zariadení sú priamo prenášané do určitej pamäte počítača smerom nadol alebo nahor sa nazýva priamy prístup do pamäte (DMA).

Priamy prístup do pamäte nezabezpečuje prenos dát len medzi periférnym zariadením a pamäťou, ale aj vzájomne medzi pamäťami, takže DMA zabezpečuje prenos dát medzi:

- pamäťou a periférnym zariadením,
- pamäťou a pamäťou.

Teoreticky (nie v PC) pomocou priameho prístupu do pamäte, možno realizovať vysoko rýchlostný prenos dát. Nevýhodou takéhoto prenosu je nemožnosť predspracovania prenášaných dát. Nutnosť obnovovania obsahu dynamických pamätí v PC neumožňuje plne využívať súvislý prenos (dávkový a blokový) ale len pomalší prenos DMA v jednom cykle s následným pozastavením PC po 1 cykle.

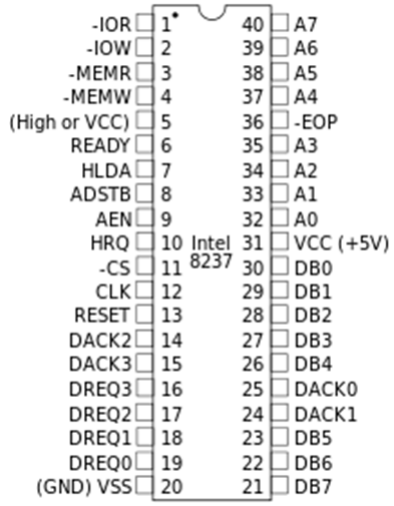
DMA prenos je špeciálny typ prenosu, bez účasti mikroprocesora (výstupy mikroprocesora ako aj ďalšie buffer-e na zbernici sú signálom BUSEN, generovaným mikroprocesorom, počas prenosu uvedené do tzv. tretieho stavu a sú nevodivé), kde špeciálny programovateľný obvod - kontrolér DMA (DMAC - často typ 8237A) preberá kontrolu nad riadením zberníc. Vlastnosti kontroléra DMAC (typu 8237A) možno zhrnúť do nasledujúcich bodov:

- DMAC obsahuje 4 prenosové kanály a každý z nich môže adresovať až 64 KB pamäti.
- Na základe požiadavky DREQ od I/O zariadenia preberá DMAC funkciu riadenia zbernice. V prípade, že žiadajúcich zariadení je viacej, uplatní sa vstavaný prioritný systém (voľba pevnej alebo rotujúcej priority).
- Hoci obvod DMAC možno naprogramovať aj do režimu súvislého prenosu dát, v PC sa činnosť DMAC pozastavuje len na vykonanie prenosu jedného slova, aby sa nenarušili podmienky obnovovania obsahu dynamických pamätí.

Priamy prístup do pamäte zabezpečuje radič 8237A v kaskádovitom zapojení, ale taktiež ako to bolo pri prerušeniach aj tu pri PC-XT nebolo kaskádovito zapojenie, DMA zabezpečoval iba jeden radič.

Prenos sa realizuje po definovaných blokoch o dĺžke 64 kiB. Vzhľadom na to, aby sa zabezpečilo obnovenie (refresh) dynamickej pamäte nie je možné, aby celý blok dát sa preniesol naraz, ale robí sa to takzvanou metódou *kradnutia cyklov*. Procesor sa uvedie do tretieho stavu v rámci prvého cyklu kedy umožní DMA prenos 1 bajtu a vráti sa k svojej činnosti. Takto pokračuje pokiaľ sa neprenesie celý blok. Takýmto spôsobom funguje prenos z pamäte do periférneho zariadenia a prenos medzi pamäťami.

Na obrázku Obr. 2.21 je znázornený radič 8237A aj s jeho vývodmi (pinmi).



Obr. 2.21 Radič 8237A s vývodmi (pinmi)

Význam jednotlivých pinov (vývodov) radiča priameho prístupu do pamäte 8237A je v tabuľke Tab. 2.6.

Tab. 2.6 Význam vývodov (pinov) radiča prerušení 8237A

Kontakt (pin)	Signál	Význam
1, 2	IOR, IOW	obojsmerné – vývody pre riadenie spolupráce so vstupno-výstupným zariadením,
3, 4	MEMR, MEMW	obojsmerné – vývody pre riadenie spolupráce s pamäťou,
5, 31	VCC	VCC – elektrické napájanie,
6	READY	vstup – synchronizačný signál pseudosynchrónnej zbernice,
7	HLDA	vstup – povolenie DMA prenosu,
8	ADSTB	výstup – zápisný synchronizačný signál pre horné adresné bity do vnútornej záchytnej pamäti,
9	AEN	výstup – potvrdenie platnosti DMA adresy na vodičoch adresnej zbernice,
10	HRQ	výstup – žiadosť radiča o povolenie DMA prenosu,
11	CS	vstup – výber obvodu,
12	CLK	vstup – hodinový synchronizačný signál,
13	RESET	vstup – nulovanie radiča
25, 24, 14, 15	DACK0 – DACK3	výstup – potvrdenie DMA prenosu na danom kanály,
19, 18, 17, 16	DREQ0 – DREQ3	vstup – žiadosť V/V zariadenia o DMA prenos,
20	GND	uzemnenie,
30, 29, 28, 27, 26, 23, 22, 21	DB0 – DB7	obojsmerné – vývody dátovej zbernice,
32, 33, 34, 35, 37, 38, 39, 40	A0 – A7	obojsmerné – vývody adresnej zbernice,
36	EOP	obojsmerné – koniec vstupu a výstupu

Samotný radič 8237A zahŕňa 4 nezávislé kanály DMA. Ako bolo spomenuté na úrovni PC-XT bol jeden radič DMA a na úrovni PC-AT a vyšších rád sú využité 2 radiče zapojené v kaskáde, čo ponúka 7 nezávislých kanálov DMA.

2.4.1 Inicializácia prenosu DMA

Pred vykonaním konkrétneho prenosu, cez niektorý z kanálov DMA, musí byť nastavený:

- Typ a smer prenosu (z pamäti alebo do pamäti, pre 8 alebo 16 bitové dáta).
- Register začiatočnej adresy pamäte. Pretože v PC je adresa najmenej 20 bitov, používa sa na doplnenie adresy z 8 bitov na potrebných 20 bitov,

resp. 32 bitov ešte ďalší tzv. register stránky. Adresa sa teda ukladá po častiach pomocou špeciálneho synchronizačného signálu ADSTB (Address strobe).

- Register dĺžky prenosu (počet bajtov). V PC sa používa DMA prenos po 1 slove.

Inicializácia kontroléra DMAC sa uskutočňuje zápisom niekoľkých riadiacich slov do vnútorných registrov DMAC, ktoré sú adresované pomocou adries na zbernicových vodičoch A0 - A3. Pretože DMAC 8237A má len 8 bitovú dátovú zbernicu, tak niektoré adresy predstavujúce 16 bitové bunky sa nahrávajú na dva krát po 8 bitových slovách.

Inicializáciou radiča DMA sa rozumie zápis niekoľko riadiacich slov, ktoré určujú, či bude obvod riadiť čítanie z pamäti, alebo naopak zápis. Ďalej sa týmito slovami nastavuje dĺžka jednej informácie (8 bitov, 16 bitov), dĺžka prenášaného bloku, priorita kanálov, počiatočná adresa prenášaného bloku v pamäti a povolenie DMA na danom kanály.

Obvod 8237A má celkom 16 adries, na ktorých sú registre slúžiace ako k inicializácií, tak k zisťovaniu aktuálneho stavu prenosu. V skutočnosti je týchto registrov podstatne viac (obvod obsahuje 344 bitov vnútornej RWM pamäte), pretože niektoré adresy predstavujú 16 bitové bunky. No a práve do nich sa 8 bitová informácia nahráva na dvakrát (2x po sebe sa pošle na danú adresu informácia). Význam DMA registrov pre zápis v počítačoch PC-XT a PC-AT ilustruje tabuľka Tab. 2.7 a registre pre čítanie sú v tabuľke Tab. 2.8.

Tab. 2.7 Vstupno-výstupné adresy registrov radičov DMA obsahujúce stavovú informáciu

Vstupno-výstupná adresa (čítanie)	**Význam**
00H	Aktuálna adresa pamäte (16b) pre 0. kanál DMA
01H	Aktuálna dĺžka prenášaného bloku (16b) pre 0. kanál DMA
02H	Aktuálna adresa pamäte (16b) pre 1. kanál DMA
03H	Aktuálna dĺžka prenášaného bloku (16b) pre 1. kanál DMA
04H	Aktuálna adresa pamäte (16b) pre 2. kanál DMA
05H	Aktuálna dĺžka prenášaného bloku (16b) pre 2. kanál DMA
06H	Aktuálna adresa pamäte (16b) pre 3. kanál DMA
07H	Aktuálna dĺžka prenášaného bloku (16b) pre 3. kanál DMA
08H	Stavový register prvého radiča DMA
09H – 0FH	Nepoužitá adresa
C4H	Aktuálna adresa pamäte (16b) pre 5. kanál DMA
C6H	Aktuálna dĺžka prenášaného bloku (16b) pre 5. kanál DMA
C8H	Aktuálna adresa pamäte (16b) pre 6. kanál DMA
CAH	Aktuálna dĺžka prenášaného bloku (16b) pre 6. kanál DMA
CCH	Aktuálna adresa pamäte (16b) pre 7. kanál DMA
CEH	Aktuálna dĺžka prenášaného bloku (16b) pre 7. kanál DMA
D0H	Stavový register druhého radiča DMA
D2H – DEH	Nepoužitá adresa

Registre pre počiatočnú adresu bloku pamäte sa nahrávajú kedykoľvek pred prevádzaným prenosom a to zaslaním dvoch osembitových hodnôt vždy na tú istú adresu. Obvod 8237 totiž nemá pre 16-bitovú konštantu dostatočne širokú dátovú zbernicu. Ako obvykle sa najprv zapisuje spodný bajt a potom horný bajt.

Tab. 2.8 Adresovanie stránkových registrov DMA

Vstupno-výstupná adresa	Stránkový register kanálu
81H	DMA 2 (aj PC-XT)
82H	DMA 3 (aj PC-XT)
83H	DMA 1 (aj PC-XT)
87H	DMA 0
88H	DMA 5
89H	DMA 6
8AH	DMA 7
8FH	obnovenie dynamickej pamäte

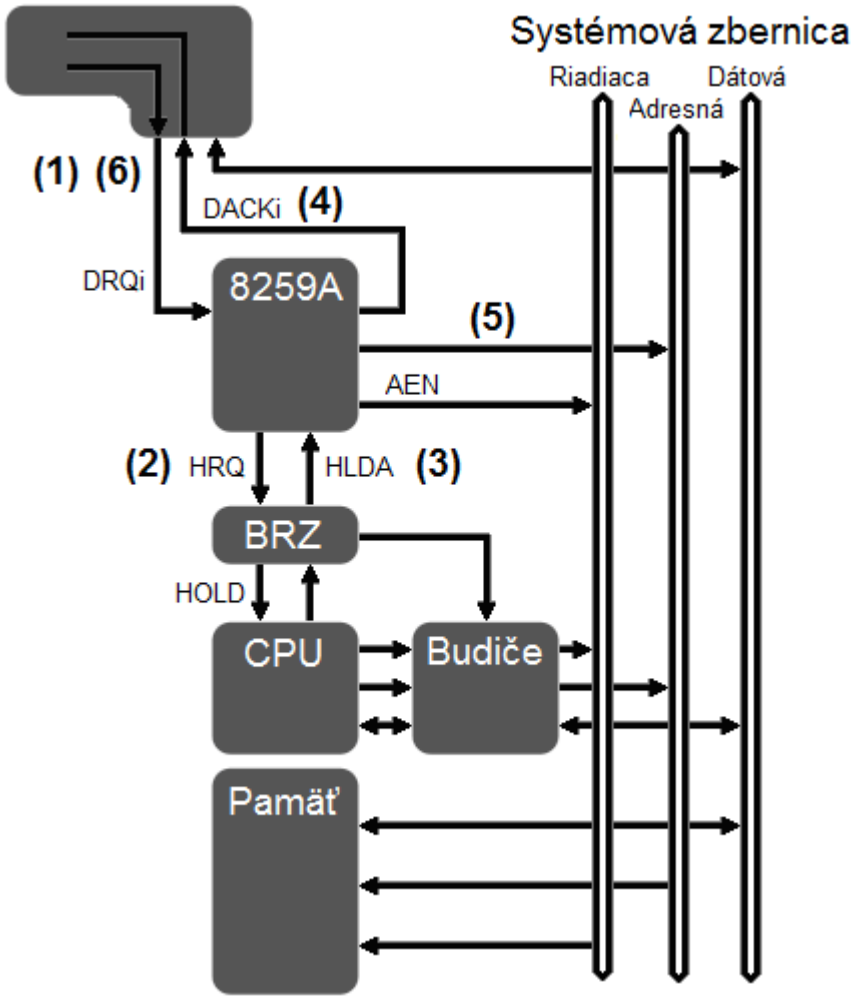
2.4.2 Priebeh prenosu DMA

Prenos DMA sa vykonáva na žiadosť DREQ niektorého z vonkajších adaptérov.

- DMAC rozhodne o priorite žiadosti konkrétnej DREQ s ohľadom na prioritu ostatných DREQ vyšle na zbernicu signál HRQ. Tento sa zosynchronizuje s hodinami mikroprocesora na zbernicový signál HOLD.
- Mikroprocesor v reakcii na signál HOLD zakončí prebiehajúci strojový cyklus, uvedie svoje výstupy na zbernici do nevodivého stavu a vyšle naspäť k DMAC signál odpovede HOLDA (Hold Acknowledge), čím signalizuje že zbernica bude počnúc nasledujúcim cyklom voľná pre začatie DMA prenosu. Podobný signál (BUSEN - Bus enable) vyšle mikroprocesor aj smerom k budičom zberníc AB, DB, a CB, ktoré odpoja mikroprocesor od systémovej zbernice.
- DMAC na príchod signálu HLDA zareaguje vyslaním signálu DACK k príslušnému adaptéru. (Pre žiadajúci adaptér sa tento signál stane výberovým, umožňujúcim mu prístup na zbernicu).
- DMAC sa teraz môže ujať riadenia prenosu. Najprv vyšle hornú časť adresy v sprievode riadiacich signálov AEN a ADSTB, ktorá reprezentuje časť adresy zdroja alebo cieľa prenosu. Po uložení hornej časti adresy pomocou signálu ADSTB sa vodiče D0 - D7 stanú znova súčasťou zbernice DB a vodiče A0 - A7 znovu dolnou časťou zbernice AB. Signál AEN súčasne inicializuje výstup signálu BUSEN.
- DMAC odteraz už ovláda riadenie prenosu dát cez zbernicu a počas niekoľkých hodinových cyklov uskutoční prenos jednej položky údajov. Na rozdiel od mikroprocesora pomocou signálov MEMR a IOW ovláda súčasne prenos dát z pamäti na I/O zariadenie, resp. súčasne prenos opačným smerom pomocou signálov MEMW a IOR. (Na takýto typ prenosu stačí jeden kanál DMA. Na prenos typu pamäť - pamäť sú však potrené 2 kanály DMA, pretože signály MEMR a MEMW sa nemôžu generovať súčasne. Napríklad kanál DMA0 môže špecifikovať zdroj dát a kanál DMA1 zase cieľ prenosu dát. V tomto prípade sa nevyžaduje dialóg DREQ - DACK.)
- V prípade dávkového alebo blokového prenosu by DMA prenos dát riadený DMAC typu 8237A pokračoval až do zakončenia prenosu celého bloku dát (terminal count - keď počítadlo bajtov dopočíta do 0) alebo

do okamihu pozastavenia DMA prenosu vonkajším signálom EOP (End of process). Z dôvodov potreby obnovy obsahu dynamických pamätí sa však v PC používa len jednoslovný DMA prenos. Preto po skončení 1 prenosu DMA sa preruší žiadosť HRQ a mikroprocesor prestane vysielať aktívny signál HLDA. Budiče systémovej zbernice sa pripoja k mikroprocesoru a začne prebiehať normálny procesorový zbernicový cyklus.

Zbernicové cykly mikroprocesora sa teda prekladajú podľa požiadavky (realizovanej pomocou prerušenia INT) zbernicovými cyklami DMA. Najvyšší dosiahnuteľný výkon DMA podsystému by nastal teda v prípade pravidelného striedania zbernicových cyklov DMA a zbernicových cyklov mikroprocesora. V skutočnosti z dôvodov zabezpečenia synchronizácie mikroprocesor prekladá viacej svojich cyklov s cyklom DMA a preto je výsledná rýchlosť DMA prenosu nižšia.



Obr. 2.22 Postupnosť činností pri obsluhe žiadosti o DMA prenos

Lepšie bude vysvetlenie priebehu prenosu DMA pomocou schémy na obrázku Obr. 2.22, čísla ktoré sú na obrázku v zátvorkách budú zodpovedať číslam v zátvorkách v tomto odseku, tieto čísla určujú aj poradie krokov. Najprv niektorý z adaptérov zašle prostredníctvom signálu DRQi žiadosť o prenos dát (1). Radič DMA rozhodne o priorite žiadosti vzhľadom k ostatným a smerom k bloku riadenia zbernice (BRZ) vyšle signál HRQ (2). Tento obvod zosynchronizuje žiadosť HRQ s hodinami mikroprocesora a ako signál HOLD ju pošle ďalej. Na čo mikroprocesor reaguje tak, že dokončí práve prebiehajúci strojový cyklus a bloku riadenia zbernice odpovie. Smerom k radiču DMA začne BRZ vysielať signál HLDA (3) oznamujúci, že zbernica bude počnúc ďalším hodinovým taktom voľná a DMA prenos môže začať. Podobné signály vyšle i smerom k budičom adresnej, dátovej a riadiacej zbernice, a tie odpoja mikroprocesor od systémovej zbernice. Všetky tieto signály ale na zbernici nie je vidieť, pretože sú vnútornými signálmi systémovej dosky. Radič DMA na príchod signálu HLDA zareaguje vyslaním zodpovedajúceho DACKi na zbernicu (4). Pre žiadajúci adaptér je tento signál signálom výberovým, umožňuje mu prístup na zbernicu. Riadeniu prenosu sa teraz ujme radič DMA (5) a behom niekoľko hodinových taktov uskutoční prenos jednej položky údajov. Hodinový generátor, z ktorého odvodzuje radič DMA trvanie jednotlivých taktov, nie je obyčajne totožný s hodinovým generátorom procesora. Po obdŕžaní signálu DACKi ukončí adaptér vysielanie žiadosti DRQi (6). Radič DMA zase po skončení zbernicového cyklu zhodí žiadosť HRQ a procesor prestane vysielať aktívny signál HLDA. Budiče všetkých častí systémovej zbernice opäť pripojí CPU a začne prebiehať normálny procesorový zbernicový cyklus.

2.4.3 Pridelenie DMA kanálov

Pre riadenie priameho prístupu do pamäte sa využívajú dva integrované radiče DMA. Obvody sú radené kaskádovito prostredníctvom štvrtého kanála druhého obvodu DMA. Z celkového počtu ôsmich kanálov je ich teda k dispozícií sedem.

Štandardizované sú však iba dva kanály a to kanál 1 a kanál 2 (viď Tab. 2.9). Ostatné nie sú obsadené a sú prístupné na zbernici. Oproti modelu PC-XT nie je obsadený ani kanál 0 pre obnovenie dynamickej pamäte (v nových modeloch je obnovenie zaistené špeciálnym technickým vybavením), ani kanál pre disk typu Winchester.

Pridelenie DMA kanálov v modely PC-XT a PC-AT sú v tabuľke Tab. 2.9

Tab. 2.9 Pridelenie DMA kanálov

Kanál	Pridelené zariadenie (PC-XT)	Pridelené zariadenie (PC-AT)
0	refresh dynamickej pamäte (8 bitový prenos),	rezerva (8 bitový prenos),
1	rezerva (8 bitový prenos),	komunikačný adaptér (8 bitový prenos),
2	adaptér diskety (8 bitový prenos),	adaptér diskety (8 bitový prenos),
3	adaptér disku Winchester (8 bitový prenos).	rezerva (8 bitový prenos),
4		prepojenie radičov DMA,
5		rezerva (16 bitový prenos),
6		rezerva (16 bitový prenos),
7		rezerva (16 bitový prenos),

Prvý radič 8237A obsahuje kanály 0 až 3 a je určený k riadeniu 8-bitových dátových prenosov medzi 8-bitovými vstupno-výstupnými adaptérmi a 16-bitovou pamäťou.

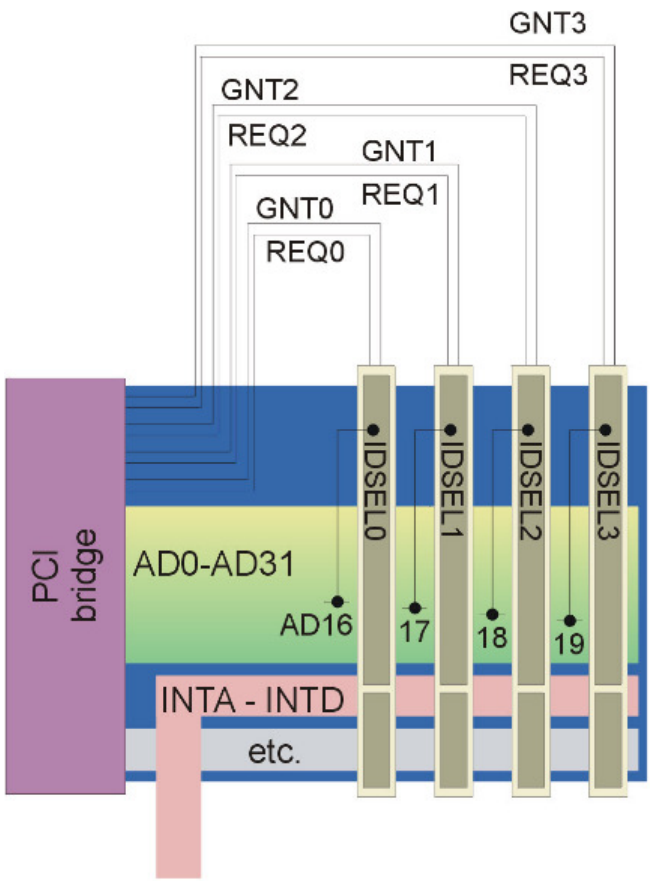
Každý kanál môže adresovať pamäťový priestor maximálne 16 MiB, ale však po blokoch dlhých maximálne 64 kiB. Druhý radič (kanály 5 až 7) zaisťuje prenos dát medzi pamäťami a 16-bitovými vstupno-výstupnými adaptérmi. Adresovať môže opäť 16 MiB pamäte, ale dĺžka prenášaného bloku môže byť až 128 kiB pamäte. Musí však začínať na párnej adrese. Adresy stránkových registrov konkrétnych DMA kanálov sú v tabuľke Tab. 2.8.

Dĺžka bloku pri kanáloch 5, 6 a 7 je 128 kiB, pričom začiatok bloku v pamäti vždy začína na párnej adrese.

2.4.4 PCI DMA

Mechanizmus priameho prístupu do pamäte na platforme PCI („PCI DMA“) je realizovaný pomocou BUS mastering-u. PCI DMA vyzerá z hľadiska operačného systému podobne ako klasický priamy prístup do pamäte (viď vyššie), je však vykonávaný pomocou BUS – master transakcií zbernice PCI.

Zdieľanie DMA kanálov (ako to bolo klasické pri prerušeniach) je možné iba teoreticky, prakticky sa však nepoužíva. Zariadenia, ktoré používali ISA DMA, sa v PC ťažko zišli v dostatočnom počte, aby bola o DMA kanály núdza. A pokiaľ by sa aj zišli, zdieľanie DMA kanálu by najskôr v takejto situácií nepredstavovalo zhodnú variantu.



Obr. 2.23 Signály zabezpečujúce bus mastering medzi slotmi PCI a PCI bridge-om

Na zbernici PCI existuje niekoľko režijných signálov, ktoré sú individuálne pre každý slot, respektíve PCI zariadenie – najdôležitejšie z týchto signálov sú tieto:

- IDSEL – výstup PCI bridge-u – používa sa pri konfigurácií zariadenia na zbernici, k jednoznačnej identifikácií slotu pre pridelením prostriedkov,
- REQ – vstup PCI bridge-u – žiadosť o pridelenie zbernice (bus request), použije ho zariadenie, ktoré chce previesť bus-master transakciu,
- GNT – výstup PCI bridge-u – potvrdenie o pridelení zbernice (bus grant), zariadenie, ktoré žiadalo o pridelenie zbernice, môže teraz previesť bus-master transakciu,

Bus-master transakcia slúži podobnému účelu ako klasický DMA, t.j. k prenosom väčších objemov dát po zbernici bez priamej účasti procesora. PCI DMA sa dá používať aj medzi zariadeniami navzájom, nie len z periférneho zariadenia do operačnej pamäte hostiteľského systému (alebo medzi pamäťami). Väčšina periférnych PCI kariet na bus-masteringu vyslovene závisia – ako príklady sa dajú uviesť grafické karty, sieťové karty, zvukové karty alebo diskové radiče každého druhu.

Na obrázku Obr. 2.23 sú zakreslené signály GNT, signály IDSEL sú vedené analogicky, a signály REQ analogicky opačným smerom. Signál IDSEL je nevyhnutne nutný k sprevádzkovaniu PCI slotu (zariadenia). Tento signál teda nechýba pri žiadnom slote či integrovanom zariadení. Signály REQ a GNT by teoreticky nemuseli byť pri každom zariadení, ale takýto slot (zariadenie) by nebol schopný bus-masteringu.

Signály REQ a GNT nepatria medzi zdroje spravované pomocou PnP či vôbec nejako softvérovo viditeľné, a preto nie je možné z operačného systému zistiť ich priradenie ani ich akokoľvek konfigurovať.

Bus mastering je funkcia, ktorej hlavnou náplňou je riadenie zbernice to znamená, že procesor počítačového systému prechodne prevezme kontrolu nad zbernicou adaptéra karty (master zbernice). Podpora viacerých architektúr zberníc umožňuje, aby zariadenie pripojené k zbernici zahájilo operáciu v určitom čase a mieste. Zbernica master funguje ako akýsi most, alebo samostatný procesor. To je označované ako úplne riadenie zbernice („First-party DMA“, „bus mastering DMA“) a značí to, že vstupno-výstupné zariadenie je schopné robiť zložitejšie sekvencie operácií bez zásahu CPU, systém DMA radiča vytvára prenos. To obyčajne znamená, že vstupno-výstupné zariadenie obsahuje vlastný radič. V jednoduchej architektúre môže len jeden procesor riadiť zbernicu. To znamená, že celá komunikácia medzi („slave“) vstupno-výstupným zariadením musí zahŕňať CPU.

Táto architektúra umožňuje ďalšie prostriedky, ktoré sa striedajú v ovládaní zbernice. To umožňuje, napríklad radič sieťovej karty pre prístup k radiču disku, pričom procesor prevedie ďalšie úlohy, ktoré nevyžadujú zbernice, napríklad načítanie kódu zo svojej vyrovnávacej pamäti. Riadenie zbernice PCI a AGP umožňuje viacero zariadení na zbernici Master, a tím výrazne zlepšuje výkon pre operačné systémy k bežnému použitiu. Typickými príkladmi sú sieťové karty, radiče diskov, zvukové karty a grafické karty, ktoré môžu mať schopnosť riadenia zbernice.

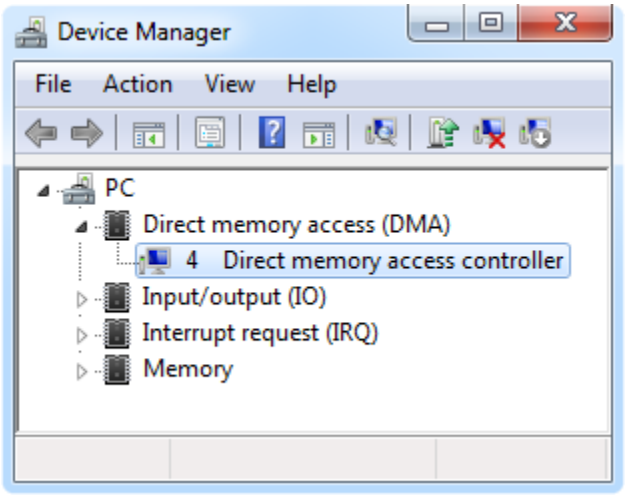
Bus mastering teoreticky umožňuje, aby jedno periférne zariadenie priamo komunikovalo s iným, v praxi takmer všetky periférie zvládnu pomocou zbernice vykonať DMA do hlavnej pamäte.

Každý prístroj môže riadiť dáta na dátovej zbernici aj keď ich procesor číta z rovnakého zariadenia, ale iba riadiaca zbernica riadi adresnú zbernicu a riadiace signály. Bližšie informácie o fungovaní zberníc sú popísané v 3. kapitole.

PCI Express využíva taktiež pri DMA bus mastering, len funguje iným spôsobom. V prvom rade musí zariadenie nastaviť „Bus master enable“, ibaže toto sa už nedeje pomocou samotných signálov (REQ, GNT), ale pomocou packet-u TLP (Transaction Layer Packet – packet transakčnej vrstvy). Ďalším dôležitým krokom je vyčítať partnerovu fyzickú adresu (fyzickú adresu zariadenia/pamäte z ktorým sa má vytvoriť DMA). Po týchto dvoch krokoch už nasleduje prenos pomocou DMA. Aj tu je menší rozdiel a to taký, že komunikácia beží pomocou packet-ov (nie pomocou priameho kódovania informácie).

2.4.5 Popis DMA v operačnom systéme

Tak ako adresovanie pamäte, adresovanie vstupno-výstupného podsystému a adresovanie prerušovacieho podsystému aj DMA sa dá v dnešných OS od firmy Microsoft prezrieť pomocou nástroja Správca zariadení (Device manager). Ktorý je umiestnený medzi nástrojmi Ovládacieho panela (Control Panel). V nástroji Správca zariadení sa v hornom menu v položke „Zobraziť“ („View“) vyberie „Prostriedky podľa typu“ („Resources by type“) alebo „Prostriedky podľa pripojenia“ („Resources by connection“) a následne sa rozbalí zložka „Priamy prístup do pamäte“ („Direct memory access (DMA)“). Na obrázku Obr. 2.24 je príklad takéhoto podsystému priameho prístupu do pamäte v OS Windows 7 pre konkrétny model PC.



Obr. 2.24 DMA zobrazené pomocou nástroja Správca zariadení (Device Manager)

2.5 Obrazový podsystém počítačov

Táto podkapitola v úvode vysvetlí fungovanie technológií od prvých CRT monitorov, cez LCD monitory po dnešné LED až OLED monitory. Existuje aj plazmová obrazovka, ale táto sa pre monitory nepoužívala nakoľko musela mať veľké rozmery a mala vysokú spotrebu elektrickej energie, preto sa touto technológiu v tejto podkapitole zaoberať nebudeme (a taktiež je táto technológia na ústupe). Ďalšia časť podkapitoly sa venuje alfanumerickému a grafickému režimu. Nasledujúca časť kapitoly sa zaoberá vývojom obrazového podsystému. Predposledná časť sa venuje rozhraniam umožňujúcim prepojenie grafických adaptérov a monitorov počítačov. Posledná časť podkapitoly rozoberá možné spôsoby programovania obrazového podsystému.

2.5.1 Fungovanie CRT monitorov

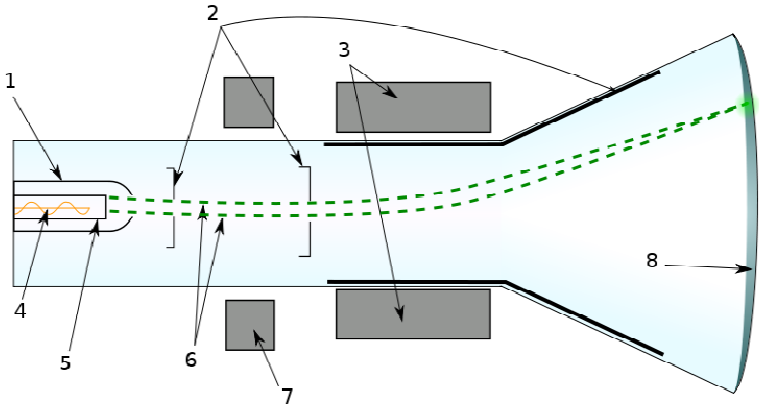
Obrazovka alebo (zobrazovacia) katódová trubica (angl. CRT - Cathode Ray Tube) je vákuová elektrónka, ktorá premieňa elektrický signál na viditeľný obraz, pričom tento obraz sa vytvára tak, že pohybujúci sa elektrónový zväzok s premenlivou intenzitou dopadá na fluorescenčné tienidlo. Používa sa napríklad v televízoroch, *počítačových monitoroch* a osciloskopoch.

Prúd elektrónov vychádzajúci z nepriamo žeravenej katódy sa tvaruje do tenkého nerozbiehavého zväzku sústavou elektród pripojených na vhodné napätia. Celá sústava sa nazýva elektrónové delo a je umiestnená v hrdle obrazovky (úzky valec v osi obrazovky). Elektrónový lúč je elektromagneticky vychyľovaný sústavou cievok umiestnených zvonka na prechode z hrdla do banky obrazovky tak, aby sa stopa lúča v mieste dopadu na tienidlo pohybovala v riadkoch umiestnených tesne jeden pod druhým a tým postupne pokryla celé tienidlo. Tienidlo je pokryté vrstvou luminoforu schopného katódoluminescencie (t.j. po dopade elektrónov vyžaruje (emituje) svetlo jednej vlnovej dĺžky (jednej farby)) s pomerne krátkym dosvitom (aby pri pohyblivom obraze pohybujúce sa svetlé objekty „neťahali za sebou chvost“). Prúd katódy (ovplyvňuje množstvo elektrónov v lúči) je modulovaný tak, aby odrážaný lúč vytvoril žiadaný obraz. Vnútro banky obrazovky je pokryté vodivou (uhlíkovou) vrstvou a tvorí vlastne anódu obrazovky, pričom je na ňu (zvláštnou elektródou vo vrchnej časti banky) privedené urýchľujúce napätie 15 až 25 kV.

Farebná obrazovka má v hrdle umiestnenú trojicu samostatne modulovaných katód (po jednej zo základných farieb - červená, zelená, modrá - RGB) so spoločnou elektrónovou optikou. Vychyľovacia sústava je tiež spoločná, avšak zahŕňa ešte sústavu korekčných permanentných magnetov a cievok. Tienidlo je pokryté pravidelnou sústavou bodov s luminoformi emitujúcimi svetlo troch farieb a tesne pred tienidlom je umiestnená kovová mriežka, ktorá zabraňuje dopadu rozptýlených elektrónov na luminofor vedľajšej farby, než pre ktorý je lúč určený.

Schematický rez obrazovkou s elektromagnetickým vychyľovaním a zaostrovaním je na obrázku Obr. 2.25, jednotlivé označené čísla na obrázku majú takýto význam:

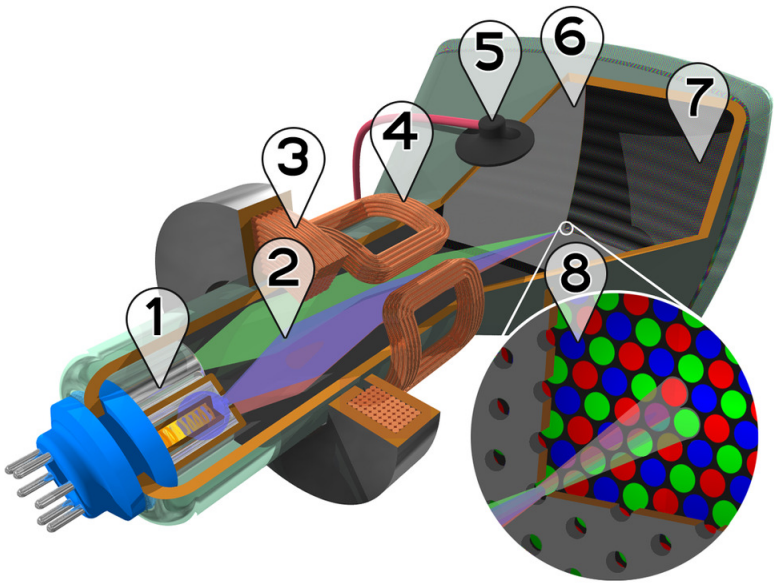
1. riadiaca mriežka,
2. anóda,
3. vychyľovacie cievky,
4. žeraviace vlákno,
5. katóda,
6. elektrónový lúč,
7. zaostrovacie cievky,
8. luminoforová vrstva.



Obr. 2.25 Schematický rez obrazovkou s elektromagnetickým vychyľovaním a zaostrovaním

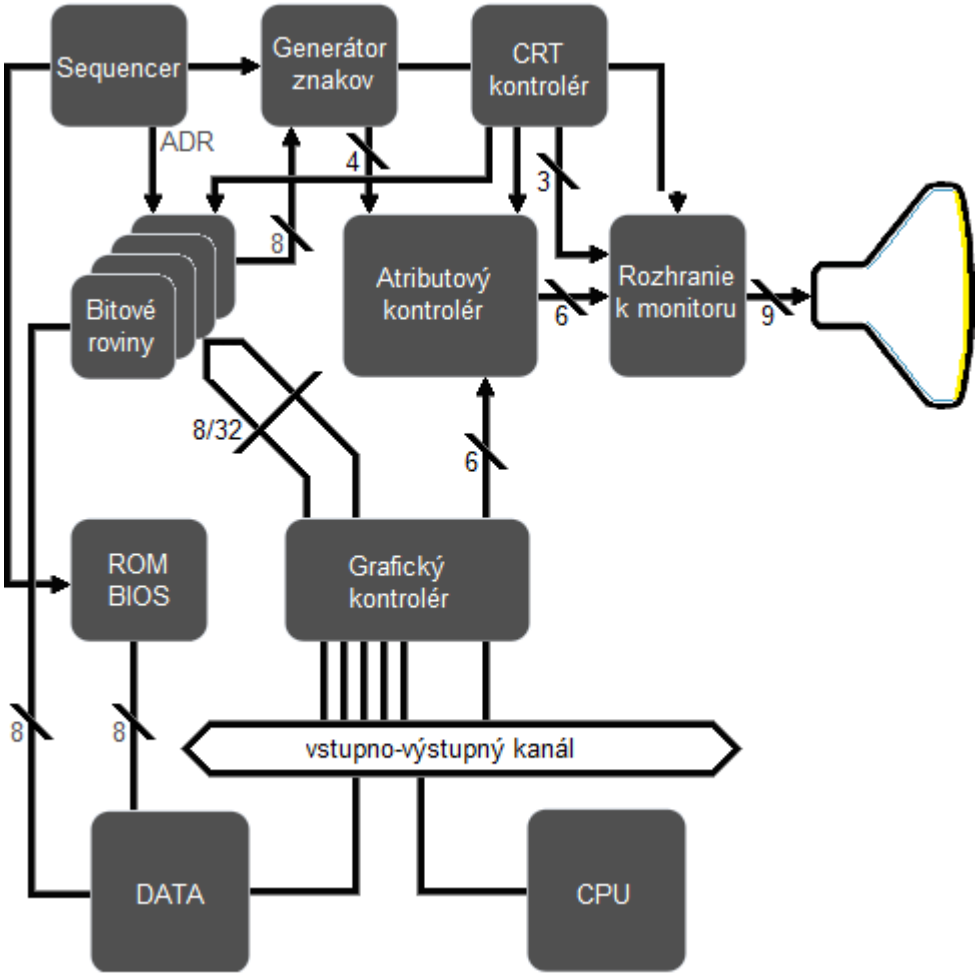
Rez farebnou televíznou obrazovkou (Obr. 2.26):

1. elektrónové delo,
2. tri samostatné elektrónové lúče pre červenú, zelenú a modrú zložku obrazu,
3. zaostrovacie cievky,
4. vychyľovacie cievky,
5. prívod anódového napätia,
6. dierovaná maska zabezpečujúca dopad lúčov jednotlivých farebných zložiek na luminofory zodpovedajúcej farby,
7. luminoforová vrstva s červenými, zelenými a modrými luminoformi,
8. detail luminoforovej vrstvy a funkcia dierovanej masky,



Obr. 2.26 Rez farebnou televíznou obrazovkou

Na obrázku Obr. 2.27 sú jednotlivé časti analógovej grafickej karty pre adaptéry EGA a VGA: atribútový kontrolér, CRT kontrolér, grafický kontrolér, sequencer:



Obr. 2.27 Schéma analógového adaptéra EGA a VGA (analógová grafická karta)

Vysvetlenie jednotlivých blokov obrázka:

- *Sequencer* – zabezpečuje adresovanie obrazovej pamäte, pamäte ktorá je tvorená štyrmi bitovými rovinami,
- *CRT kontrolér* – zabezpečuje synchronizáciu monitora a podporu kurzora,
- *Atribútový kontrolér* – vytvára jednotlivé farby popredia a pozadia a v prípade, že výstup z atribútového kontroléra priamo ovláda D/A prevodník,
- *Grafický kontrolér* – riadi dátový prenos medzi CPU, video pamäťou a atribútovým kontrolérom.

Bitové roviny:

1. ASCII,
2. atribúty,
3. definícia znaku,
4. v alfanumerickom režime nevyužitá.

2.5.2 Fungovanie LCD a LED monitorov

Začiatkom 90. rokoch sa začali objavovať prvé farebné LCD monitory na notebookoch, ale tie boli obmedzené len na niekoľko farieb (16, 256), no po roku 1996 sa začali objavovať prvé televízory a monitory s „true color“. Dôvodom prečo v rokoch 1996 až 2004 rozširovali viac plazmové televízory ako LCD televízory bolo rozlíšenie, pretože už koncom 90. rokoch plazma dosahovala kvalitu HDTV. Plazmové obrazovky všetky veľké firmy už z výroby vyradzujú a tak budú nasledovať technológiu CRT. Pri počítačových technológiách bolo spomínané obdobie (1996-2004) vynechané, pretože plazmový monitor by pri počítači bol obrovský s vysokou spotrebou, a tak sa prešlo od CRT monitorov rovno k LCD monitorom. Kvôli vysokej cene LCD monitorov sa tak začalo diať až začiatkom nového tisícročia a najmä po roku 2006, keď už aj monitor počítača mohol mať rozlíšenie Full HD. CRT monitory začali z európskeho trhu miznúť v roku 2003, keď cena nového CRT monitora bola rovnaká ako cena LCD monitora s rovnakými parametrami.

Monitor LCD alebo LCD panel je monitor, ktorého zobrazovacím prvkom je displej z tekutých kryštálov. Zobrazovacia časť LCD monitora je zložená z kvapalných kryštálov a ich ovládacích elektród na nosiči, polarizačných filtrov a zdroja svetla. Kvapalné kryštály umožňujú dynamické riadenie jasu jednotlivých bodov monitora, čo umožňuje zobraziť aj rýchlo sa pohybujúci farebný obraz.

Displej z tekutých kryštálov (Liquid crystal display, LCD) je tenké a ploché zobrazovacie zariadenie, ktorého obraz sa skladá z farebných alebo monochromatických bodov zoradených pred zdrojom svetla. Pri prevádzke vyžaduje relatívne malé množstvo energie a preto ho je možné použiť i pri napájaní z batérií. Zdrojom svetla LCD displeja je pasívna reflexná vrstva umiestnená za displejom, ktorá odráža svetlo dopadajúce na displej, alebo aktívna biela rozptylná plocha zozadu osvetlená výbojkou (obvykle dve trubice uložené po stranách displeja), alebo radom bielych LED diód (LED monitor).

Každý bod (pixel) LCD displeja sa skladá z molekúl kvapalných (tekutých) kryštálov, ktoré sú umiestnené medzi dvoma priehľadnými elektródami. Nad a pod elektródami sa nachádzajú polarizačné filtre. Zadná stena je rovnomerne osvetlená pasívnym zdrojom svetla - neónovými trubicami, LED a pod. Filtre a natočenie molekúl kvapalných kryštálov (bez napätia sú molekuly v tzv. chaotickom stave) spôsobujú, že svetlo zo zdroja neprejde cez LCD vrstvu. Privedením napätia na elektródy sa tekuté kryštály natočia do špirálovej štruktúry tak, že rotujúce svetlo prejde cez polarizačný filter a LCD bod sa javí ako priehľadný. V okamihu pustenia elektrického prúdu do elektród sú molekuly kvapalného kryštálu ťahané rovnomerne s elektrickým poľom, čo znižuje rotáciu vstupujúceho svetla. Ak nie sú kryštály natočené vôbec, prechádzajúce svetlo bude polarizované kolmo k druhému filtru, a svetlo bude teda blokované a bod sa javí ako tmavý. Pomocou natočenia kryštálov je teda možné riadiť množstvo svetla prechádzajúce bodom, a teda jas bodu (pixelu).

Pre finančné úspory sú lacnejšie LCD multiplexované, tzn., displej je riadený riadkom a stĺpcom elektród. Jedinečné prekríženie riadku a stĺpca je vlastne bod. V danom okamihu teda nesvietia všetky body, len jeden riadok. Prepínanie je však také rýchle, že obraz sa javí ako kompaktný. V TFT displejoch je každý bod – pixel, resp. subpixel riadený vlastným tranzistorom.

Vo farebných LCD displejoch je každý pixel rozdelený do troch subpixelov a to červeného, zeleného a modrého (teda klasické RGB ako pri CRT monitoroch), ktoré sú tvorené farebnými filtrami. Svietivosť každého subpixelu je možné kontrolovať samostatne, a tak je možné dosiahnuť milióny farebných kombinácií.

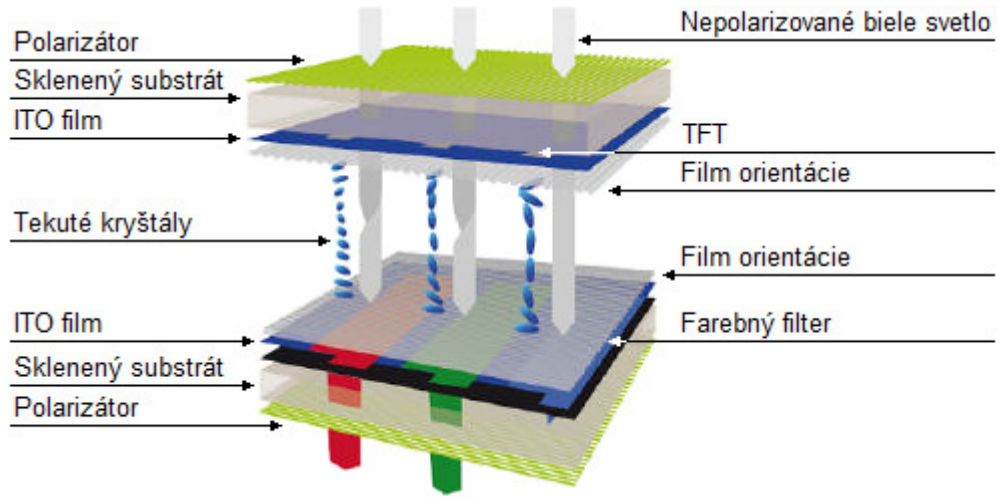
Celá spomínaná technológia je zhrnutá na obrázku Obr. 2.28, jedinou nespomínanou vecou je ITO film, ITO je skratka od slovného spojenia Indium Tin Oxide

čo predstavuje pevnú zlúčeninu oxidu inditého (In_2O_3 – 90%) a oxidu ciničitého (SnO_2 – 10%), ktorá slúži ako priehľadná elektrický vodivá povrchová vrstva.

LCD monitory nahradili technológiu monitorov CRT. Majú oproti CRT monitorom viaceré výhody. Ich najväčšími výhodami je úspora miesta, stabilný obraz (obraz sa neobnovuje), nízka spotreba elektrickej energie, žiadne negatívne žiarenie, ostrý obraz a nízka váha. LCD panely majú dlhú životnosť, väčšina ich parametrov sa v čase prakticky nemení.

Nevýhodou je, že obrazovka funguje ostro len v tzv. natívnom rozlíšení, pretože má pevný počet bodov. Pri zmene rozlíšenia na iné sa obraz javí ako neostrý. (napr. zobrazenie v režime 800x600 na monitore s natívnym rozlíšením 1024x768). Ďalšou nevýhodou je možnosť poruchy bodu (chybný subpixel). Pixel potom žiari jednou farbou, čo pôsobí rušivo. Táto chyba je neopraviteľná. Niektorí výrobcovia delia monitory do tried, pričom v niektorých triedach nezaručujú, že všetky body sú v poriadku. Ďalším problémom je ich odozva - časové oneskorenie, ktoré spôsobuje „duchovanie“ alebo opisovanie stopy v aktívnych častiach scény (napr. hry alebo filmy). Odozva sa udáva v prechode medzi čiernou a bielou farbou, moderné LCD monitory majú odozvu nie vyššiu ako 10 ms čo je dostatočné pre akékoľvek činnosti. Rýchlosť odozvy môže ovplyvniť napr. prostredie v ktorom sa monitor používa, v chladnom prostredí je odozva podstatne nižšia ako pri izbovej teplote. Je to spôsobené tým, že tekuté kryštály tuhnú a prechod medzi čiernou a bielou farbou tým trvá oveľa dlhšie. Tento jav je najčastejšie pozorovateľný na LCD displejoch mobilných telefónov. LCD kryštály fungujú len v určitom teplotnom rozmedzí. Priveľký chlad (mráz), alebo naopak prehriatie (priame slnko) ich môže nenávratne poškodiť.

LCD technológia umožnila aj rýchly rozvoj širokouhlých LCD monitorov, ktorých zobrazovací pomer je iný, ako štandardných 4:3 - podobne, ako je tomu pri TV. Môžme sa stretnúť s pomerom 16:9, ale aj inými. Výhodou širokouhlej obrazovky (anglicky wide) je to, že je možné na displeji zobraziť viac informácii naraz, preto sú širokouhlé monitory obľúbené na sledovanie filmov, alebo pri práci s viacerými oknami. Niektoré monitory sú vybavené funkciou Pivot tá umožňuje otočiť monitor o 90 stupňov a užívateľ získa viac miesta pre prezeranie textu. S výhodou je táto funkcia využívaná pri DTP, pretože umožňuje vidieť naraz celú stránku textu. Monitory s touto funkciu majú v zadnej časti obrazovky kĺb, ktorý umožňuje monitor natočiť.



Obr. 2.28 Prierez LCD panelom

LED monitor je nepresné označenie, pretože je to v podstate LCD monitor s LED podsvietením. Termín LED TV (monitor) je spochybňovaný, pretože sa nejedná o obraz tvorený LED diódami, ale iba o podsvietenie obrazových bodov (pixelov), obrazové body sa aj pri týchto paneloch skladajú z tekutých kryštálov (LCD). Takže sa stále jedná o LCD televízory s novým marketingovým názvom, v ktorom sa úplne a technicky nesprávne vypúšťa pojem LCD.

Spoločnosť ASA (Advertising Standards Autority) oznámila, že termín "LED TV" sa vo Veľkej Británii môže používať iba ak je ďalej uvedené že televízor (monitor) používa technológiu LED podsvietenia.

Klasické LED diódy sú v súčasnosti ešte príliš veľké, aby z nich bolo možné vyrobiť jednotlivé pixely do bežného monitora. Použitie skutočného LED displeja je teda možné len pre omnoho väčšie obrazovky napríklad športovisko alebo veľké reklamné plochy. Je pravdepodobné, že sa výrobcovia rozhodli označiť svoje vyššie rady LCD televízorov a monitorov ako LED TV, aby niečo získali pred nastupujúcou technológiou OLED televízorov a monitorov.

Existuje niekoľko spôsobov podsvietenia LCD panelov pomocou LED. Všetky majú priaznivý vplyv na spotrebu a životnosť displeja s výnimkou technológie Edge LED, a taktiež všetky tieto podsvietenia (vrátane Edge LED) majú prínos v kvalite obrazu:

- *RGB LED* - Používajú sa skupiny štyroch LED (červená, modrá, dve zelené), ktoré sú rozmiestnené maticovo po celej ploche panela. Pri tejto technológií sa dá použiť tzv. „local dimming“, čo je stlmenie jednotlivých LED v mieste kde je potrebné dosiahnuť sýtejšej čiernej farby. Taktiež je možné dosiahnuť vyšších hodnôt farebného spektra ako pri iných spôsoboch podsvietenia.
- *Direct LED* - Opäť sa jedná o maticové rozloženie LED za panelom, ale používajú sa iba biele LED. Tiež sa dá použiť funkcia „local dimming“ a dosiahnuť vyššieho kontrastu.
- *Edge LED* - Biele LED sú umiestnené iba po okrajoch panela a pomocou siete špeciálnych svetlovodov s odraznými plôškami sa svetlo z LED rovnomerne rozptýli za LCD panelom. Výhodou tejto technológie je použitie menšieho počtu LED a tým aj zníženie nákladov na výrobu a teda aj ceny, panel vďaka zredukovaniu LED môže byť veľmi tenký. Nevýhodou je, že funkciu „local dimming“ nie je možné použiť. Obraz súčasných LED TV s týmto systémom podsvietenia aj tak patrí k tomu najdokonalejšiemu čo súčasný trh v rámci LED displejov ponúka. Svetlovody za obrazovým panelom sú generačne už inde ako u prvých LED TV, rozdiel medzi podsvietením siete LED, ktorý sa nazýva RGB LED a EDGE LED je nerozpoznateľný, tomuto napomáhajú aj rôzne špeciálne fólie z prednej strany panela čím výrobcovia úplne dorovnali možno aj predbehli všetky výhody systému RGB LED.

Podľa obrázka Obr. 2.28 sa dá predstaviť aj fungovanie tejto technológie, len treba brať na vedomie, že zdrojom nepolarizovaného svetla sú LED diódy. Tieto monitory začali prichádzať na trh v roku 2006, v podstate v dobe prvých Full HD LCD monitorov.

2.5.3 Fungovanie OLED monitorov

OLED (organická svetlo-emitujúca dióda) je elektroluminiscenčná dióda v ktorej na vyžarovanie svetla z elektroluminiscenčnej vrstvy sú použité organické látky. Táto vrstva organického polovodiča sa nachádza medzi dvoma elektródami, pričom jedna z elektród býva priehľadná. OLED sa používa v celej škále výrobkov, od miniatúrnych displejov v MP3 prehrávačoch, mobilných telefónoch až po veľkorozmerné ploché zobrazovacie displeje a televízie. Intenzívny výskum prebieha v oblasti bielych OLED pre osvetľovaciu techniku a zariadenia.

Rozlišujú sa dva základné typy OLED: jeden používa malé molekuly a druhý polyméry. Pridaním pohyblivých iónov do OLED vrstvy sa vytvorí svetlo emitujúca elektrochemická bunka (LEC – Light Emitting Cell) s mierne odlišným prevádzkovým režimom. OLED displeje sa podľa spôsobu zapojenia delia aj na pasívne (PMOLED) alebo aktívne (AMOLED). Pre OLED s aktívnou maticou (Active-matrix OLEDs - AMOLED) je nutný tenký film tranzistorov na spodnej vrstve elektród pre zapínanie a vypínanie každého pixelu, toto riešenie ale poskytuje výhodu vyššieho rozlíšenia alebo tvorbu veľkých rozmerov displejov.

Displej z OLED nepotrebuje pre svoju prevádzku podsvietenie. Na základe tohto konštrukčného rozdielu, keďže klasický LCD ho pre svoju funkciu potrebuje, je obraz z OLED displeja podstatne kontrastnejší a živší, dokáže "zobraziť" hlbokú čiernu, a najmä je podstatne tenší a ľahší.

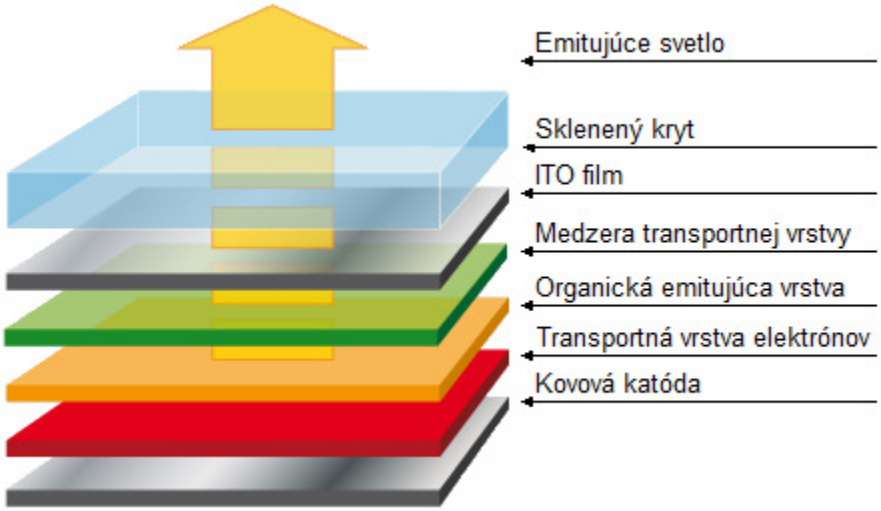
Veľkou výhodou pri konštrukcii OLED je možnosť aplikácie aktívnej elektroluminiscenčnej vrstvy na pružné podkladové médiá, čo umožňuje vytváranie pružných a ohybných displejov a zobrazovacích plôch (Obr. 2.29). Klasické LCD pre funkciu potrebujú sklenený substrát, čím sa takmer vylučuje ohybnosť a pružnosť takéhoto displeja.

Obr. 2.29 Príklad využitia ohybnosti OLED displejov

Typická OLED je tvorená vrstvou organického materiálu medzi dvoma elektródami, anódou a katódou, zapuzdrené v substráte. Organické molekuly sú elektricky (podmienene) vodivé a výsledkom je presun elektrónov spôsobený chemickou konjugáciou cez časť alebo celú molekulu. Tieto materiály majú mieru vodivosti od izolantu po vodič, preto sú považované za organické polovodiče. Najvyššie obsadené a najnižšie neobsadené molekulové orbitály (HOMO/LUMO|HOMO a LUMO) organických polovodičov sú analógiou k valenčným a vodivým väzbám anorganických polovodičov (Obr. 2.30).

Pôvodne sa základný OLED polymér skladal iba z jednej organickej vrstvy. Môžu sa však vytvárať viacvrstvové OLED s dvoma či viacerými vrstvami za účelom zlepšenia

účinnosti. Veľa moderných OLED má jednoduchú dvojvrstvovú štruktúru, ktorá sa skladá z vodivej a emisnej vrstvy (Obr. 2.30). Posledný vývoj v architektúre OLED zvyšuje kvantovú účinnosť (až o 19%) pomocou odstupňovanej heterogenity.



Obr. 2.30 Prierez OLED panelom

Prvé OLED panely (monitory a televízie) sa na trh dostali v roku 2007, vyrábala ich firma SONY, ale cena bola a je veľmi vysoká. Po roku 2010 sa výroby chytili aj ďalšie firmy a v roku 2013 prišli na trh oblé panely, ktorých cena sa stále ráta v tisíckach eur (k roku 2015).

OLED môže byť vytlačené na akýkoľvek vhodný podkladový substrát technológiou atramentovej tryskovej tlače alebo aj obrazovej tlače, a teoreticky dosiahnuť nižšie náklady na výrobu ako u LCD či plazmových displejov. Zatiaľ je však výroba a príprava vhodného substrátu drahšia, než u TFT LCD. Cena však výrazne klesne pri zahájení masovej produkcie. Metóda výroby dlhometrážnych rolí naparovaním (chemickou plynnou depozíciou) pre organické zariadenia dovoľuje vyrábať tisíce displejov za minútu pri minimálnych nákladoch, aj keď táto technológia pri viacvrstvových zariadeniach znamená veľkú výzvu kvôli nutnosti dodržať extrémnu presnosť pri prekrývaní jednotlivých vrstiev.

Displeje OLED môžu byť vyrábané na tenkých podkladoch, čo umožňuje výrobu vysoko ohybných a rolovateľných displejov, zobrazovacích panelov, vložených do textílií alebo šiat. Ako substrát je použiteľný napríklad aj Polyetylénтereftalát (PET). OLED umožňuje väčší odstup v kontrastnom pomere (v dynamickom aj statickom rozsahu) a podstatne väčší uhol viditeľnosti obrazu oproti LCD, pretože jednotlivé pixely vyžarujú svetlo priamo. Taktiež farba jednotlivých pixelov nemení v závislosti od uhla pohľadu tak výrazne svoj odtieň.

Polarizačné filtre používané pri LCD prepúšťajú len časť svetla produkovaného podsvietením. Zároveň nedokážu plne zablokovať priechod svetla, preto táto technológia nedokáže zobraziť úplnú čiernu. OLED technológia umožňuje zobraziť čiernu práve tým, že daný pixel je jednoducho vypnutý. Odstránením nutnosti podsvietenia je OLED podstatne tenší, odľahčený o nepotrebné vrstvy. Zároveň to činí celú konštrukciu jednoduchšou a na výrobu lacnejšou. Displeje OLED taktiež umožňujú podstatne rýchlejšie reakcie než štandardné LCD panely. Kým sú LCD schopné odozvy medzi 1-16 ms a snímaciu frekvenciu

medzi 60-480 Hz, OLED displej teoreticky dokáže ísť až k hranici 0,01 ms, čo znamená zmenu obrazu takmer na úrovni 100 000 Hz. Na nasledujúcom obrázku Obr. 2.31 je OLED televízor od firmy LG.

Obr. 2.31 OLED TV od firmy LG

2.5.4 Alfanumerický a grafický režim

Táto podkapitola vysvetlí princíp realizácie obrazu v alfanumerickom a grafickom režime.

Alfanumerický režim:

- na obrazovke sú zobrazované iba znaky (písmena, čísla, iné znaky), využívané kódovanie bolo ASCII, takže tých znakov bolo 256 (okrem špeciálnych znakov ako napr. Enter, NULL, Backspace, atď…) a po väčšine to bol biely text na čiernom podklade (pozadí), ale nebola to podmienka,
- pre zobrazenie boli potrebné: ASCII kódy jednotlivých znakov, farba pozadia a farba znakov,
- pre zobrazenie jedného znaku boli potrebné dva bajty:
    - 1. bajt je ASCII kód znaku,
    - 2. bajt je znakový režim/mód (hrubé písmo, kurzíva, podčiarknuté, prečiarknuté, blikajúce)

Grafický režim:

- v grafickom režime je základný prvok pixel (obrazový bod),
- pixel je zobrazovaný na obrazovke počítača ako jeho najmenší zobrazovaný bod, ktorý môže nadobúdať len jednu farbu,
- pixely sú v pamäti usporiadané od vrchného ľavého rohu až po dolný pravý roh,
- dnes je najčastejšie pixel zobrazovaný pomocou troch farebných rovín RGB (červená, zelená, modrá), ktoré jednotlivo zaberajú v pamäti jeden bajt ktorý uvádza jas jednotlivých základných farieb (pre bližšie informácie je potrebné si naštudovať Počítačové videnie), takže jeden pixel je zakódovaný troma bajtmi.

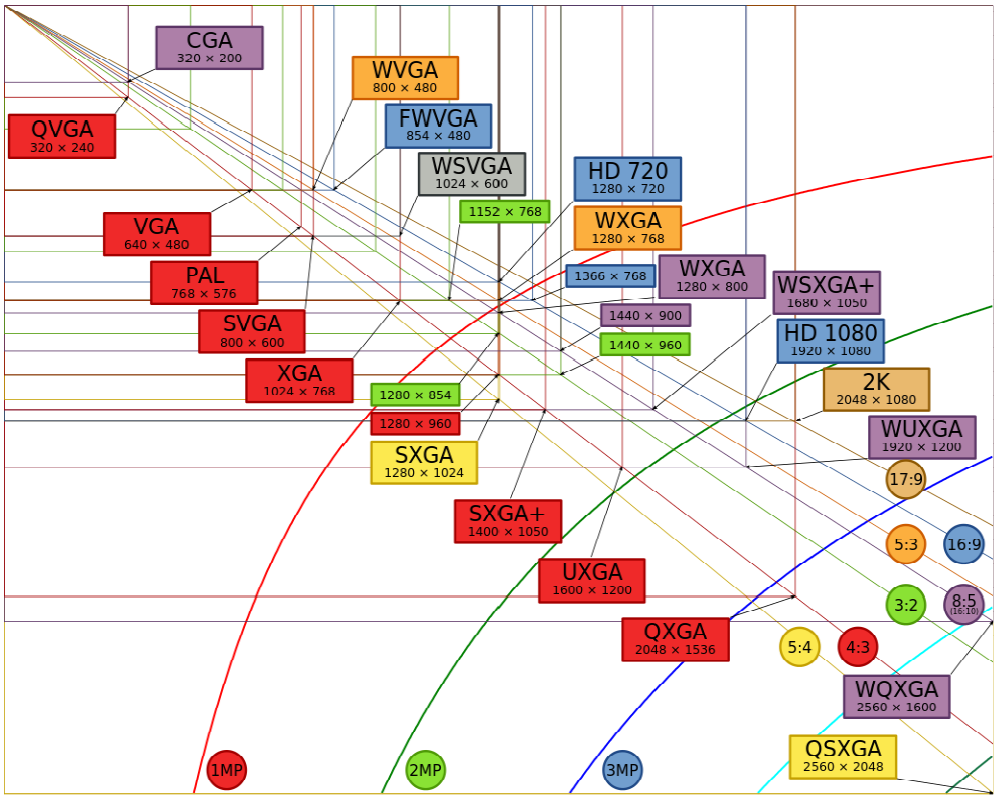
2.5.5 Vývoj počítačového obrazového podsystému

V tejto časti je jednoducho po bodoch popísaný vývoj obrazového podsystému od prvého grafického adaptéra, ktorý klasický zobrazoval alfanumerický režim (80 stĺpcov a 25 riadkov), až po Full HD:

- MDA: Monochrome display adapter (1981):
  - fungoval len v alfanumerickom režime,
  - rozlíšenie textu: 80 stĺpcov (znakov) a 25 riadkov,
  - jeden znak mal rozlíšenie 9 x 14 pixelov,
  - takže celkové rozlíšenie obrazovky bolo 720x350,
  - jedna obrazovka zaberá vo video RAM: 80 x 25 x 2 B = 4000 B = **4 kB** (3,91 kiB).
- CGA: Color graphic adapter (1981):
  - adaptér určený pre grafický režim,
  - dokázal fungovať aj v alfanumerickom režime, ale rozlíšenie znaku bolo iba 8 x 8 pixelov,
  - rozlíšenie: 640x200,
  - jedna obrazovka zaberá vo video RAM: 640 x 200 = 128 000 b = 16 000 B = **16 kB** (15,63 kiB).
- HGC: Hercules graphic card (1984):
  - bol to adaptér MGA (monochrome graphic adapter),
  - fungoval v grafickom aj alfanumerickom režime,
  - využíval iba jednu farbu, najčastejšie zelenú,
  - predchodca tohto adaptéra bol MDA (z neho bol vyvinutý),
  - rozlíšenie: 720x350,
  - v tej dobe lacnejší ako CGA,
  - jedna obrazovka zaberá vo video RAM: 720 x 350 = 252 000 b = 31 500 B = 31,5 kB (30,77 kiB) ≈ **32 kB**.
- HGC+: Hercules graphic card plus (1986):
  - užívateľ vie definovať tvar znaku pomocou generátora znakov (character generator).
- InColor: Hercules InColor Card (1987):
  - 16 farieb (2^4 = 16 farieb, takže 4 bity = 0,5 B),
  - jedna obrazovka zaberá vo video RAM: 720 x 350 x 0,5 B = 126 000 B = 126 kB (123,05 kiB) ≈ **128 kB**.

- EGA: Enhanced graphic adapter (1985):
  - zobrazovanie 16-tich farieb z palety 64 farieb,
  - rozlíšenie 640 x 350 pixelov,
  - jedna obrazovka zaberá vo video RAM: 640 x 350 x 0,5 B = 112 000 B = 112 kB (109,38 kiB) ≈ **128 kB** (pri CRT zobrazovaní, každého druhého riadku to bolo **64kB** a neskôr mal adaptér aj **256 kB**).
- MCGA: Multi-color graphic adapter:
  - zobrazovanie 256-tich farieb z palety 262 144 farieb (1farba=1B),
  - rozlíšenie 640 x 480 pixelov,
  - jedna obrazovka zaberá vo video RAM: 640 x 480 x 1 B = 307 200 B = 307,2 kB (**300 kiB**).
- VGA: Video graphic array (1987):
  - zobrazovanie 16-tich farieb z palety 262 144 farieb,
  - rozlíšenie 640 x 480 pixelov,
  - jedna obrazovka zaberá vo video RAM: 640 x 480 x 0,5 B = 153 600 B = 153,6 kB (**150 kiB**).
- SVGA: Super VGA (1989):
  - zobrazovanie 256-tich farieb z palety 262 144 farieb,
  - rozlíšenie 800x600,
  - jedna obrazovka zaberá vo video RAM: 800 x 600 x 1 B = 480 000 B = 480 kB (468,75 kiB) ≈ **512 kB**.
- XGA: Extended Graphics Array (1990):
  - 65536 (2^{16}) farieb pri rozlíšení 800 x 600 pixelov,
  - jedna obrazovka zaberá vo video RAM: 800 x 600 x 2 B = 960 000 B = 960 kB (937,5 kiB) ≈ **1 MB**,
  - 256 (2^{8}) farieb pri rozlíšení 1024x768 pixelov,
  - jedna obrazovka zaberá vo video RAM: 1024 x 768 x 1 B = 786 432 B = 786,44 kB (768 kiB) = **0,75 MiB**.
- SXGA: Super Extended Graphics Array (1990):
  - zmena pomeru šírky a výšky obrazovky zo 4:3 na 5:4,
  - maximálna hĺbka kódovania farieb: 24 bitov (2^{24} farieb = 16 777 216 farieb),
  - rozlíšenie 1280 x 1024 pixelov,
  - jedna obrazovka zaberá vo video RAM: 1280 x 1024 x 3 B = 3 932 160 B = 3 933 kB (3 840 kiB) ≈ **4 MB**.
- UXGA: Ultra Extended Graphics Array:
  - rozlíšenie 1600 x 1200 pixelov (4:3),
  - 24 bitové kódovanie,
  - jedna obrazovka zaberá vo video RAM: 1600 x 1200 x 3 B = 5 760 000 B = 5 760 kB (5 625 kiB) ≈ **5,5 MiB**,
  - 1 sekunda záznamu (24 Hz) zaberá 135 MiB (139 MB) pamäte (samozrejme bez kompresií),
- Full HD (1080p):
  - rozlíšenie 1920 x 1080 pixelov (16:9),
  - 24 bitové kódovanie,
  - jedna obrazovka zaberá vo video RAM: 1920 x 1080 x 3 B = 6 220 800 B = 6 221 kB (6 075 kiB) ≈ **6 MiB**.
  - 1 sekunda záznamu (60 Hz) zaberá 356 MiB (374 MB) pamäte,

Na obrázku Obr. 2.32 sú znázornené jednotlivé štandardy obrazového podsystému počítačov:



Obr. 2.32 Rozlíšenia jednotlivých grafických štandardov

Ak by sa nepoužíval žiaden druh kompresie, tak jeden 90 minútový film by zaberal na disku (šírka x výška x farebné kódovanie x obnovovacia frekvencia x 60 sekúnd x 90 minút = 1920 x 1080 x 3 x 60 x 60 x 90) 1,84 TiB (2,02 TB). Pri kódovaní ktoré sa používa pri BRD (Blue-Ray Disk) 90 minútový film zaberá 50 GB pamäte (pri 60 Hz, pri klasických 30 Hz zaberá 25 GB pamäte). Pri kódovaní MKV takýto film zaberá 15 GB (7,5 GB pri 30 Hz). Full HD vysielanie má maximálnu prenosovú rýchlosť 8 Mbit/s čo predstavuje najvyššiu kompresiu a jeden 90 minútový film takto zaberá maximálne 5,4 GB.

Okrem spomínaných 4-bitových (16 farieb), 8-bitových (256 farieb), 16-bitových (65 536 farieb) a 24-bitových kódovaní farieb existujú aj 30-bitové (10 bitov na jednu farebnú rovinu), 36-bitové (12 bitov na jednu farebnú rovinu) a 48-bitové (16 bitov na jednu farebnú rovinu) kódovania, ale tie sa používajú len zriedka.

Ak sa sleduje film s Full HD rozlíšením a divák je od televízora v dostatočnej vzdialenosti (vzdialenosť sa určuje podľa uhlopriečky televízora je to od 1 m do 4 m), tak je to pre ľudské oko maximálne možné rozlíšenie, takže ľudské oko nevníma rozdiel medzi kvalitou obrazu reálneho sveta a kvalitou obrazu na obrazovke (je to maximálne možné rozlíšenie pre približne 6 000 000 čapíkov v jednom oku človeka). Samozrejme v kinách sú obrazovky väčšie a je nutné vyššie rozlíšenie. Napriek tejto skutočnosti sa dnes (rok 2014) už vyrábajú televízory s rozlíšením 4K (3840 x 2160) a vo veľmi blízkej budúcnosti (rok 2015-2016) majú prísť na trh televízory s rozlíšením 8K (7680 x 4320).

2.5.6 Rozhrania obrazového podsystému

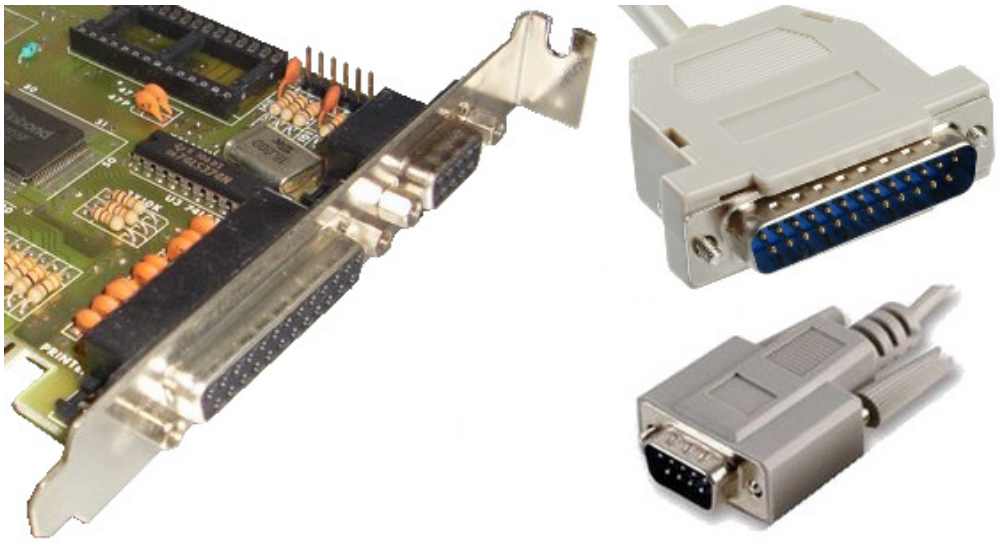
Táto časť podkapitoly obrazového podsystému pripomenie všetky rozhrania ktoré sa u PC používali pre prepojenie monitora (obrazovky) s počítačom.

Prvé PC používali ako monitor domáce televízory (CRT TV) a preto sa pre pripojenie používal najskôr koaxiálny kábel (Obr. 2.33) a neskôr RCA káble (Obr. 2.34). Tieto rozhrania boli plne analógové a na grafickej karte bol príslušný D/A prevodník s radičom, ktorý obraz v pamäti video RAM previedol na analógový signál.

Obr. 2.33 Konektor koaxiálneho kábla a koncovka grafickej karty

Obr. 2.34 RCA káble a koncovky grafickej karty

S prvými grafickými štandardami sa štandardizovali aj konektory a ich koncovky (Obr. 2.35) a tak v podstate vznikli prvé počítačové monitory.

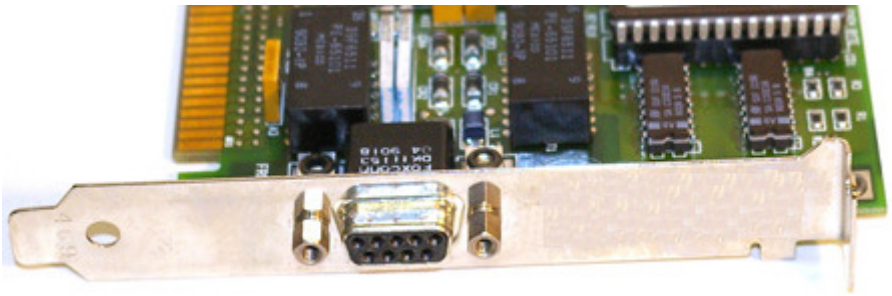


Obr. 2.35 Grafická karta so štandardizovanými konektormi DB-25 a DB-9

Môžeme ich nazvať počítačovými monitormi, pretože to boli obrazovky, ktoré mali len elektrické napájanie a príslušný konektor pre pripojenie monitora k počítaču (tým pádom sa obrazovka nedala použiť ako TV prijímač). Prvé konektory obrazového podsystému boli DB-25 (LPT konektor) a DB-9 (rovnaký konektor ako samica sériového portu). Na obrázku Obr. 2.35 je grafická karta, ktorá má oba výstupy DB-25 aj DB-9, ale boli karty, ktoré obsahovali buď iba DB-25, alebo DB-9. Tieto konektory používali grafické karty adaptérov MDA, CGA, HGC, HGC+ a InColor. Pri týchto konektoroch sa ešte občas objavili na grafických kartách aj RCA konektory (Obr. 2.34), ale grafické karty s RCA konektormi boli drahšie nakoľko bol potrebný prevod na analógový signál.

Konektor DB-25 je programovaný (driver) pomocou paralelného rozhrania (viď kapitolu 7.2), takže je to digitálny prenos údajov. Konektor DB-9 bol taktiež digitálny (TTL logika), lenže pri každom grafickom adaptéry fungoval inač, pretože každý podporoval iný počet farieb. V grafickom režime jeden prenos údajov podával väčšinou informáciu o farbe jedného pixelu. Monitory s konektorom DB-9 boli väčšinou spätne kompatibilné, monitor pre EGA fungoval aj s adaptérom MDA.

Adaptér EGA už vôbec nepodporoval konektor DB-25 už sa využíval iba EGA konektor (DB-9, Obr. 2.36), ktorý vedel zakódovať 64 farieb pomocou šiestich pinov, ďalšie dva boli určené pre časovú synchronizáciu (hodiny) a jeden pin bol uzemňovací. Takže EGA konektor bol taktiež digitálny.



Obr. 2.36 Konektor grafickej karty EGA (EGA konektor)

Najdlhšie používaným grafickým rozhraním (konektorom) je VGA začal sa používať už začiatkom 90. rokov a dodnes sa vyrábajú monitory (už aj televízory) pre toto rozhranie. Toto rozhranie spravilo akýsi krok späť, keďže grafické rozhrania od MDA až po EGA boli digitálne a rozhranie VGA je analógové. Tento krok bol spravený preto, lebo už prvý VGA adaptér rozlišoval 262 144 farieb a adaptér SXGA rozlišoval 16 777 216 farieb a využíval ten istý konektor. Takže sa to vyriešilo tak, že je jeden analógový pin pre červenú farbu, jeden pre zelenú a jeden pre modrú. Takže hodnota jasu jednotlivých farieb je zakódovaná analógovo (postačujú 3 piny pre taký vysoký počet farieb a nie 24 pinov).

Nástupcom VGA rozhrania bolo rozhranie DVI (Digital Visual Interface) v roku 1999, no napriek tomu sa dodnes používa aj rozhranie VGA. Štandard bol vytvorený za účelom bezproblémovej komunikácie medzi zobrazovacími zariadeniami ako napr. LCD monitorom alebo dátovým projektorom a grafickou kartou počítača. Vyvinula ho skupina firiem zoskupených pod názvom Digital Display Working Group (DDWG). Primárne je určený na prenos nekomprimovaných digitálnych video dát. Je čiastočne kompatibilný s rozhraním HDMI. Toto rozhranie je digitálne aj analógové, môže byť čisto analógové (DVI-A), alebo poprípade čisto digitálne (DVI-D), alebo kombinované (DVI-I). DVI-A a DVI-I boli vyvinuté najmä pre spätnú kompatibilitu so starými monitormi (CRT) a projektormi, pretože v časoch vývoja ešte boli bežne používané.

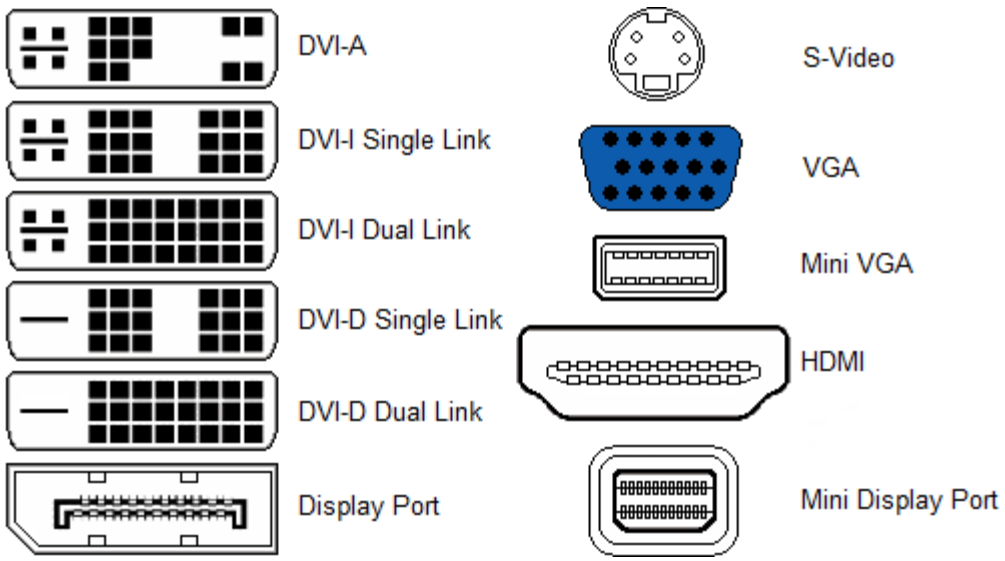
Display Port je nástupcom DVI začal sa produkovať v roku 2008. Display Port je digitálny konektor slúžiaci na prenos nekomprimovaného digitálneho obsahu s podporou 8 kanálového zvuku a ochrany DPCP (DisplayPort Content Protection) využívajúci 128 bitové šifrovanie AES. Podporuje rýchlosť prenosu až 10,8 Gb/s, na vzdialenosť do troch metrov dokáže prenášať obraz s rozlíšením 2560x1600 pixelov a na vzdialenosť do 15 metrov obraz s rozlíšením 1920x1080 pixelov. Navrhla ho organizácia VESA (Video Electronics Standards Association) a je široko podporovaný firmami ako je Intel, AMD, Dell, Nvidia atď. Je navrhnutý tak, aby nahradil digitálne (DVI) a analógové (VGA) konektory v monitoroch počítačov rovnako ako v grafických kartách. Má všetky funkcie HDMI, no nepredpokladá sa, že by mal nahradiť HDMI v oblasti domácej spotrebnej elektroniky, keďže je určený predovšetkým na kancelárske a IT využitie.

Ďalšie rozhrania (štandardy) zvyknú byť súčasťou grafických kariet preto sa spomenú a v skratke aj popíšu. Sú nimi S-Video a HDMI.

S-Video (Separate video), tiež známy ako Y/C je štandard prenosu analógového video signálu používajúci na prenos obrazových dát dve oddelené cesty (jas a farbu) na rozdiel od kompozitného videa prenášajúceho celý signál jednou cestou. S-Video je najčastejšie používané na prenos obrazu v štandardnom televíznom rozlíšení. Zvuk sa neprenáša spoločne s obrazom v jednom kábli. S-Video má 4 piny: jeden je signál pre jas, druhý jeho uzemnenie, tretí je signál farby a štvrtý jej uzemnenie.

High-Definition Multimedia Interface (HDMI) je rozhranie pre audiovizuálne vybavenie ako televízory a monitory s vysokým rozlíšením alebo systémy domáceho kina. Má 19 signálových vodičov v jednom kábli. Rozhranie HDMI 2 (štandard z roku 2013) je schopné prenášať nekomprimovaný obraz s rozlíšením maximálne 4096 x 2160 60Hz (18 Gb/s), prvá verzia HDMI (štandard z roku 2002) dokázala prenášať maximálne Full HD obraz (5 Gb/s). HDMI je digitálne rozhranie kompatibilné s DVI, jediným problémom v kompatibilite je, že DVI neprenáša zvuk.

Na obrázku Obr. 2.37 sú schematický naznačené konektory moderného obrazového podsystému.



Obr. 2.37 Konektory moderného obrazového podsystému

2.5.7 Programovanie obrazového podsystému

V tejto časti podkapitoly budú uvedené 2 vzorové príklady ako z pamäte video RAM vyčítať práve uložený obraz. Prvý príklad bude vyčítanie video RAM z alfanumerického režimu pomocou programovacieho jazyka C. Druhý príklad bude vyčítanie video RAM z grafického režimu pomocou programovacieho jazyka C#.

V prvom príklade sa využije funkcia `peek()` ostatné neznáme príkazy sú spojené s programovaním paralelného rozhrania, ktorému sa venuje kapitola 7. Spomínaná funkcia je v knižnici *dos.h* a jej syntax vyzerá takto:

```
int peek(int segment, unsigned int offset);
```

Táto funkcia vyčíta hodnotu z konkrétnej adresy (z konkrétneho *segmentu*) pamäťového podsystému so zadaným *offset-om*. V podstate je to príkaz, ktorý sa mohol spomenúť už pri pamäťovom podsystéme, ale na tomto mieste má aspoň praktické využitie, pretože od adresy B800H je uložená obrazovka alfanumerického režimu (video RAM) a tak sa tento príkaz využije na odprezentovanie oboch podsystémov.

Druhý príklad uloží do premennej stav obrazovky v grafickom režime a tento stav vykreslí na obrazovku (do okna) aplikácie. Stav obrazovky vyčítava metóda `CopyFromScreen()`, túto metódu obsahuje trieda `Graphics`. Potrebné argumenty tejto metódy pre kopírovanie obrazovky sú v riešení druhého zadania. Tento jednoduchý program ešte využíva triedy `Bitmap`, `Screen` a `CopyPixelOperation`.

Zadanie 1:

Naprogramujte program pomocou programovacieho jazyka C pod operačným systémom MS-DOS, ktorý bude mať výstup na obrazovke počítača a tento výstup aj vytlačí pomocou tlačiarne. Výstup obrazovky sa nebude ukladať do žiadnej premennej, ale vyčíta sa z pamäte video RAM.

Riešenie:

```
1    #define RIAD_REG 0x37A
2    #define DAT_REG 0x378
3    #define STAV_REG 0x379
4
5    #include <conio.h>
6    #include <stdio.h>
7    #include <dos.h>
8
9    void vystup (int znak)
10   {
11        outportb(RIAD_REG,0x0c);
12        while(!(inportb(STAV_REG) & 0x80));
13        outportb(DAT_REG,znak);
14        outportb(RIAD_REG,0x0d);
15        delay(1);
16        outportb(RIAD_REG,0x0c);
17   }
18
19   void main(void)
20   {
21        int stav,x,y;
```

```
22        outportb(RIAD_REG,0x0c);
23        stav = inportb(STAV_REG);
24        if(stav & 0x80) printf("\n Nie je zaneprazdnena");
25        if(stav & 0x40) printf("\n Nie je ackling");
26        if(stav & 0x20) printf("\n Koniec papiera");
27        if(stav & 0x10) printf("\n Je vybrata");
28        if(stav & 0x08) printf("\n Nie je error");
29        getch();
30
31        for(y=0;y<50;y+=2)
32        {
33             for(x=0;x<160;x=x+2)
34             {
35                  vystup (peek(0xb800,y*80+x));
36             }
37             vystup(10);
38        }
39   }
```

Vysvetlenie:

1-3: Definovanie využívaných adries vstupno-výstupného podsystému, konkrétne sú to adresy registrov paralelného portu.

5-7: Prilinkovanie dôležitých knižníc k programu.

9-17: Funkcia `void vystup (int znak)` slúžiaca na vytlačenie jedného znaku pomocou tlačiarne pripojene k počítaču pomocou paralelného portu (bližšie informácie k jednotlivým príkazom a funkciám sú v kapitole 7).

19: Deklarácia a definovanie hlavného programu.

21: Deklarácia premenných `stav` (stav tlačiarne), `x` (stĺpec v pamäti video RAM) a `y` (riadok v pamäti video RAM), tieto premenné sú typu `int`.

22: Inicializácia tlačiarne pomocou zápisu do riadiaceho registra paralelného portu.

23: Vyčítanie stavu tlačiarne pomocou vyčítania stavového registra paralelného portu.

24-28: Vypísanie stavu tlačiarne na monitor počítača.

29: Čakanie na stlačenie ľubovoľného klávesa na klávesnici.

31-38: Vytlačenie textu z pamäte video RAM pomocou tlačiarne:
Vonkajší cyklus sa posúva po riadkoch na obrazovke a vnútorný po jednotlivých znakoch (stĺpcoch). Dôvod prečo sa premenné inkrementujú o 2 a nie o 1 je ten, že každá druhá informácia je o stave daného znaku (viď podkapitolu 2.5.4) nie jeho ASCII kód. Vo vnútornom cykle sa pomocou funkcie `peek()` vyčíta znak z pamäte video RAM a vytlačia jednotlivé znaky v riadku. Vonkajší cyklus „vytlačí“ nový riadok (posunie valec tlačiarne na nový riadok), posunie sa na ďalší riadok a vráti sa do vnútorného cyklu. Toto sa deje až kým cykly neprejdú celou pamäťou video RAM.

Zadanie 2:

Naprogramujte program pomocou programovacieho jazyka C# pod operačným systémom Windows XP (alebo novším), ktorý po stlačení tlačidla vykreslí stav obrazovky ako obrázok v okne aplikácie.

Riešenie:

Pomocou dizajnéra sa navrhne okno aplikácie, ktoré bude obsahovať jeden `pictureBox` s názvom `pictureBox1` a jeden `button` s názvom `button1`.

```
1    using System;
2    using System.Drawing;
3    using System.Windows.Forms;
4
5    namespace KopirovanieObrazovky
6    {
7        public partial class Form1 : Form
8        {
9            public Form1()
10           {
11               InitializeComponent();
12           }
13           private void button1_Click(object sender,
             EventArgs e)
14           {
15               Bitmap Obrazovka = new Bitmap
                 (pictureBox1.Size.Width,
                 pictureBox1.Size.Height);
16               Graphics g = Graphics.FromImage(Obrazovka);
17               g.CopyFromScreen
                 (Screen.PrimaryScreen.Bounds.X,
                 Screen.PrimaryScreen.Bounds.Y, 0, 0,
                 Screen.PrimaryScreen.Bounds.Size,
                 CopyPixelOperation.SourceCopy);
18               pictureBox1.Image = Obrazovka;
19               g.Dispose();
20           }
21       }
22   }
```

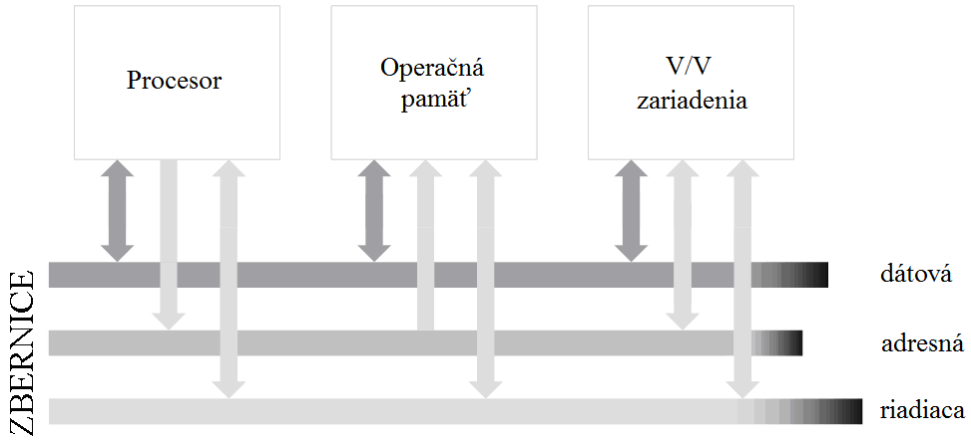
Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.
5: Deklarácia a definovanie namespace-u tohto programu.
7: Deklarácia a definovanie hlavnej triedy programu.
9: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.
11: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).
13: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla.
15: Vytvorenie inštancie triedy `Bitmap`, ktorá pomocou konštruktora vytvorí objekt `Obrazovka`.
16: Vytvorenie inštancie triedy `Graphics`, ktorá pomocou statickej triedy FromImage vytvorí objekt `g` (priradenie objektu `g` grafiku objektu `Obrazovka`).
17: Skopírovanie obrazovky monitora z video RAM do objektu `g` (argumenty: šírka obrazovky, výška obrazovky, destinácia X, destinácia Y, veľkosť obrazovky, akcia).
18: Vloženie obsahu objektu Obrazovka do objektu framework-u .NET `pictureBox1`.
19: Odstrániť grafiku v objekte g.

3 Zbernice

Zbernica je množina liniek (vodičov), ktorá navzájom prepája všetky prvky na danej štruktúrnej úrovni. Umožňuje spojenie každého s každým, ale v danom okamihu môže údaje na zbernicu vysielať iba jediné zariadenie. Zbernica počítača navzájom prepája procesor, pamäť a V/V zariadenia (HDD, optická mechanika, klávesnica, atď.).

Každá zbernica disponuje troma základnými zložkami (dátová, adresná, riadiaca), ktoré môžete vidieť na blokovej schéme (Obr. 3.1):



Obr. 3.1 Bloková schéma zapojenia počítačovej zbernice

Dátová zbernica slúži na prenos samotných dát, adresná zbernica adresuje príslušné zariadenie, respektíve pamäťovú bunku a riadiaca zbernica určuje, ktoré zariadenie v určitom okamihu môže komunikovať, prípadne typ komunikácie.

Po dátovej zbernici sa prenášajú údaje a inštrukcie z OP do CPU a údaje medzi CPU a OP, CPU a V/V alebo medzi OP a V/V (8, 16, 32, 64-bit PC).

Po adresnej zbernici sa prenášajú adresy, ktoré sú generované nadriadeným prvkom zbernice, t.j. procesorom alebo riadiacim obvodom DMA. Adresa identifikuje bunku v OP, respektíve V/V zariadenie, s ktorým sa bude pracovať. (16, 20, 24, 32, 36 bitov).

Signály riadiacej zbernice sa skladajú z povelov, generovaných nadriadeným radičom zbernice (signál čítania/zápisu) a zo žiadostí, ktorými sa podriadení obracajú na nadriadeného (žiadosť o prerušenie).

V jednom cykle zbernice sa vykoná prenos jediného údaja po zbernici (ak je šírka údajovej zbernice 8b, tak v jednom cykle sa prenesie 1 slabika).

Vzhľadom na to, že na matičnej doske môže pracovať niekoľko riadiacich prvkov, ktoré zabezpečujú transformáciu dát medzi vstupno-výstupnými zariadeniami a pamäťou, alebo transformáciu dát medzi rôznymi vstupno-výstupnými zariadeniami, tak z tohto dôvodu sa na zbernice nemožno pozerať len ako na súbor signálov, ale aj na systém, ktorý zabezpečuje logiku riadenia počítača. Za prenos dát po zbernici zodpovedá radič zbernice, ktorý je vždy nadradený (master) nad všetkými ostatnými (slaves) blokmi na zbernici. V prípade, že v rámci jedného počítačového systému je použitých viac rodičov vstupno-výstupných zariadení, musí mať zbernica určitý prideľovací systém. Tento systém nazývame *arbiter*. Arbiter ma za úlohu určiť aktívny radič v prípade, že riadenia zbernice sa dožaduje viac zberníc. Preto nesmieme stotožňovať úlohu radiča zbernice a procesoru. Arbiter zbernice môže byť buď centralizovaný, alebo distribuovaný.

Centralizovaný arbiter sa skladá z jedného modulu. *Distribuovaný arbiter* má logiku riadenia rozdelenú do viacerých modulov. Pričom distribuovaný arbiter delíme na hierarchický a alternatívne demokratický. Hierarchický má jeden hlavný radič a ostatné sú podriadené. Pri alternatívne demokratickom arbitri majú všetky radiče z hľadiska priority rovnaký prístup k zbernici a hlavný radič ma za úlohu len zbernicu prepožičiavať jednotlivým radičom (radič, ktorý pravé vysiela sa správa ako master).

Arbiter zbernice vybavuje žiadosti o pridelenie zbernice podľa priority. Priorita sa zavádza, aby dôležitejšia žiadosť sa vybavila prednostne. Žiadosti sa spracúvajú *sériovým*, *paralelným*, alebo *sériovo-paralelným* prioritným zapojením. Patria tu 3 skupiny vodičov:

- Žiadosť – o pridelenie zbernice,
- Súhlas – vyhodnotenie žiadosti,
- Potvrdenie – prevzatia zbernice, ktorým žiadateľ potvrdzuje prijatie súhlasu a hlási ostatným blokom obsadenie zbernice.

Sériové prioritné zapojenie používa jeden vodič pre žiadosť, jeden pre súhlas a jeden pre potvrdenie. Paralelne prioritné zapojenie používa pre každého žiadateľa jeden vodič žiadosti a jeden vodič súhlasu, takže namiesto dvoch vodičov sa použije dva krát toľko vodičov koľko je žiadateľov. Paralelne zapojenie používa taktiež len jeden potvrdzovací vodič. Pre sériové zbernice je charakteristické, že dáta sú posielané sériovo jedným vodičom a u paralelných je počet vodičov závislý od dĺžky prenášaného slova (8, 16, 32, 64 bitov).

Ďalšou funkciou zbernice je predanie žiadosti o prerušenie. Podobne, ako bolo uvedené vyššie (sériovo, paralelne, alebo sériovo-paralelne), je predaná žiadosť o prerušenie. Aj v tomto prípade sa súčasný výskyt žiadosti vyhodnocuje prioritne a k ich spracovaniu slúži radič prerušenia (viď kapitolu 2.3). Podľa usporiadania delíme zbernice na sériové, sériovo-paralelne a paralelne, podľa smeru prenosu na *jednosmerné* (polo-duplexný prenos, half-duplex) a *obojsmerné* (plno-duplexný prenos, full-duplex). Modul pripojený k zbernici môže byť *vysielačom*, alebo *prijímačom*. Z funkčného hľadiska sa obvykle zbernice skladajú z týchto častí:

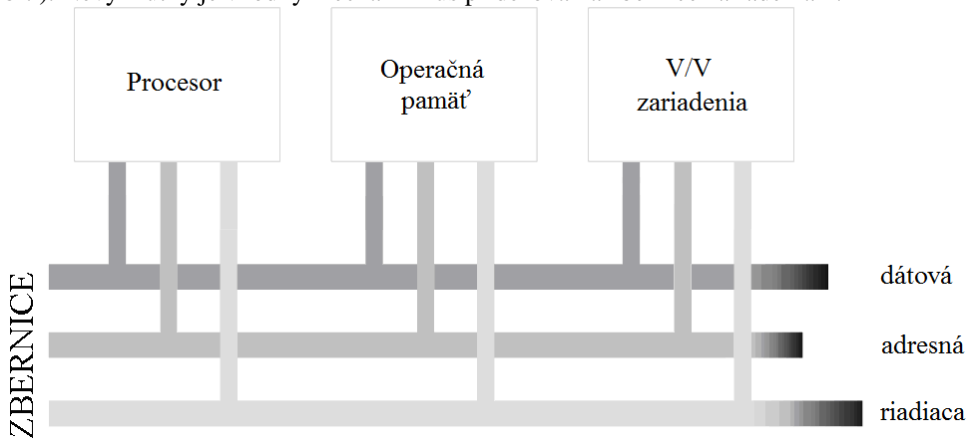
- *adresnej*,
- *dátovej*,
- *riadiacej,*
- *napájacej*,
- *signály prerušenia* (napr.: IRQ),
- *signály súvisiace s priamym prístupom do pamäte* (napr.: DRQ).

Niektoré zbernice používajú pre adresnú a dátovú časť rovnaké vodiče a sú *zdieľané*. Aktuálny význam signálu je definovaný *protokolom*, podľa ktorého sa signály v čase *multiplexujú* (napr. PCI). Z hľadiska synchronizácie prenosu, t.j. určenie okamžiku, kedy sú signály na zbernici (adresné a dátové) platné, rozlišujeme zbernice *synchrónne* a *asynchrónne*.

Okamžik platnosti údajov sa u synchrónnych zberniciach jednoznačne určuje centrálnym hodinovým kmitočtom. Protokol niektorých zberníc dovoľuje meškanie prenosu o určitý počet períód centrálnych hodín (signálom *nepripravený* podriadeného obvodu) a preto ich odlišujeme označením *pseudosynchrónne* zbernice. Pri asynchrónnych zberniciach určuje okamžik platnosti postupnosť riadiacich a stavových signálov. Pri jednostrannom riadení, ktoré je jednoduchšie, však nie je prenos dostatočne pružný, pretože podriadený modul musí splniť časový limit operácie. Častejšie sa používa korešpondenčný (kvitovaný) režim, čiže systém výzvy a potvrdenia. Výhodou synchrónneho režimu je jednoduchosť a rýchlosť. Asynchrónny prenos s obojstranným kvitovaním sa automaticky prispôsobuje rýchlosti komunikujúcich modulov. Často používaným typom súčasných počítačových zberníc je paralelná asynchrónna zbernica.

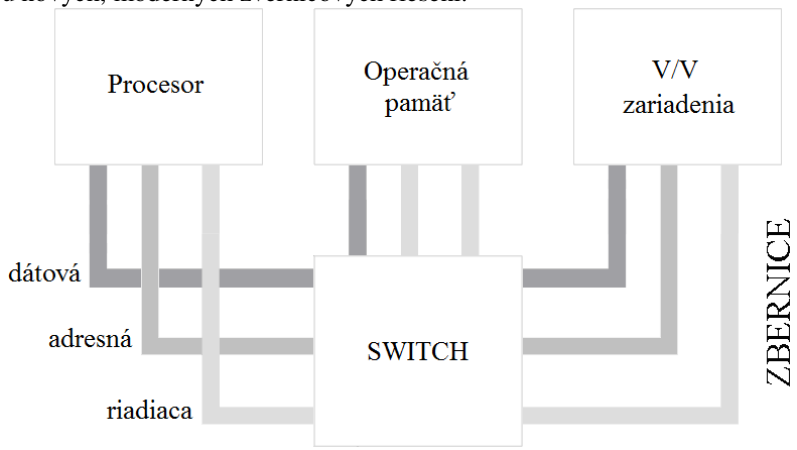
Medzi technické parametre zberníc patrí ich maximálna dĺžka, maximálny počet blokov sústavy, rozteč konektorov, rozmery kariet, charakteristická impedancia a spôsob budenia zbernice, maximálne zaťaženie jednotlivých vodičov, atď. K budeniu zberníc sa používajú *TTL* (transistor-transistor logic – tranzistorovo-tranzistorová logika) *obvody*, obvody s *trojstavovým výstupom*, alebo obvody s *otvoreným kolektorom*. Trojstavové obvody sú rýchlejšie, ale na rovnakej linke nemôže byť súčasne viac než jeden vysielač aktívny. Obvody s otvoreným kolektorom sú pomalšie, ale dovoľujú realizáciu montážnych členov. Z hľadiska odolnosti voči poruchám (šumu) je výhodnejšie, keď prijímače reagujú na úrovne, nie na hrany signálu. Dôležitou charakteristikou zbernice je jej *kapacita*, t.j. maximálny počet bitov, alebo bajtov dát prenesených za sekundu. Častejšie sa používa pojem *prenosová rýchlosť* ako kapacita. Prenos môže prebiehať po blokoch, alebo po bajtoch, či slovách.

Zbernica je spravidla riešená ako zdieľaná štruktúra (využíva ju viacero zariadení, obr.). Nevyhnutný je vhodný mechanizmus prideľovania zbernice zariadeniam.



Obr. 3.2 Zbernica so zdieľanou štruktúrou

Staré systémy (pred 4. generáciou) používali nezdieľané štruktúry. Najmodernejšie zbernice PCI-Express sa vracajú ku koncepcii point-to-point nezdieľanej topológie. Táto koncepcia zbernice bola v minulosti opustená, v súčasnosti sa v rôznych modifikáciách objavuje u nových, moderných zvernicových riešení.



Obr. 3.3 Zbernica s nezdieľanou štruktúrou (point-to-point)

Zbernice zjednodušujú pripájanie ďalších zariadení ku výpočtovému systému. Sloty pre rozširujúce karty sú súčasťou každej základnej dosky počítača, a aj keď sa nie vždy alebo všetky využijú, stále je tento princíp jednoduchší a lacnejší, ako pre každé zariadenie osobitne vyvíjať špeciálny systém komunikácie s počítačom.

Potencionálne zbernice predstavujú možné *úzke miesto* (spomaľovanie toku dát zapríčinené pomalšími perifériami než je výpočtový systém) v dátových tokoch výpočtovým systémom. Pripájané zariadenia často pracujú na rôznych princípoch a majú často veľmi odlišné požiadavky na charakter komunikácie – medzi zbernicu a zariadenie je potrebné pripojiť vhodné komunikačné rozhranie (napr. medzi monitor a systémovú zbernicu sa musí zapojiť grafický adaptér, medzi tlačiareň a systémovú zbernicu sa musí pripojiť rozhranie Centronics alebo rozhranie USB atď.). Zvyčajne je definovaný presný počet zariadení, ktoré je možné ku zbernici pripojiť. Z hľadiska fyzikálnych zákonitostí šírenia elektrických signálov vyplývajú mnohé obmedzenia v oblasti maximálnych dĺžok vodičov zbernice, maximálne použiteľnej frekvencie a podobne.

Pri transformácii prenosu dát z paralelného typu prenosu na sériový je taktiež nevyhnutné multiplexovanie (prepínanie jednotlivých bitových vodičov paralelnej zbernice do jediného dátového toku po sériovej linke). Ak má zostať zachovaná prenosová rýchlosť sústavy, musí byť takt sériovej zbernice toľko krát vyšší, koľko bitov paralelnej zbernice multiplexuje. Na vyrovnávanie krátkodobých rozdielov v priepustnosti sústav (technické príčiny poklesu prenosovej rýchlosti) sa využívajú vyrovnávacie pamäte – buffer-y (cache).

Z pohľadu účelu zbernice môžeme u počítačov vyčleniť dve hlavné skupiny zberníc *systémovú* a *vonkajšiu*. Systémová zbernica prenáša údaje medzi procesorom, operačnou pamäťou a V/V obvodmi. Vonkajšia zbernica (zbernica rozhrania) slúži na prenos dát medzi samotným počítačom a periférnymi zariadeniami.

Systémová zbernica sa nachádza priamo na základnej doske počítača. Je tvorená skupinou adresných, dátových a riadiacich vodičov, taktovacím obvodom a obvodmi pre riadenie komunikácie. Do nedávnej doby platilo, že je vždy paralelná, posledný vývoj však ukázal výhody sériového riešenia zbernice. Pôvodné riešenia zberníc osobných počítačov používali iba jedinú zbernicu pre komunikáciu v rámci základnej dosky. Synchronizáciu zaisťovali hodiny, ktoré taktovali spoločne na rovnakej frekvencii procesor, zbernicu, pamäť aj prídavné karty v slotoch (PC Bus). Postupne sa osamostatnili ako samostatné funkčné časti systémová zbernica Front Side Bus – FSB, ktorá zabezpečuje prenos dát medzi procesorom a operačnou pamäťou, a rozširujúca zbernica, ktorá pracuje na nižšej taktovacej frekvencii a ktorá zaisťuje pripojenie rozširujúcich kariet a rozhraní (napr. zbernica PCI). Rozširujúca zbernica je pripojená cez rozhranie vonkajšej zbernice (novšie cez čipovú sadu - chipset). Rozširujúca zbernica obsahuje sloty na pripojenie rozširujúcich I/O obvodov – kariet. Niektoré štandardy umožňovali pripojiť priamo ku systémovej zbernici i niektoré rozširujúce karty – potom sa zbernica zvykla nazývať lokálna (touto koncepciou sa vyznačoval napr. VL BUS).

3.1 Sériový a paralelný prenos

Prenos po sériovej zbernici si vyžaduje, aby dáta boli zoradené do jediného bit-streamu a sú prenášané po jedinom dátovom vodiči (resp. jedinom prenosovom kanále). Organizácia práce sériovej zbernice v porovnaní s paralelným prenosom je jednoduchšia, a však pri určitej taktovacej rýchlosti je prenosová rýchlosť paralelnej zbernice oproti sériovej toľko krát rýchlejšia, koľkými kanálmi disponuje. V počiatkoch výpočtovej techniky boli s ohľadom na zložitosť technického riešenia uprednostňované sériové zbernice, až potom neskôr, s vývojom techniky boli stále častejšie používané paralelné systémy. V ére osobných počítačov ovládli paralelné zbernice prakticky celú architektúru PC – systémové

zbernice, zbernice na pripojenie periférií aj vonkajších pamätí boli organizované ako paralelné. Sériové zbernice sa stali okrajovou záležitosťou. S postupným nárastom taktovacích frekvencií zberníc, ktoré sa na prelome tisícročia dostali na gigahertzové hodnoty, sa prejavili viaceré problémy paralelných zberníc:

- *parazitné javy*: susedné vodiče zbernice, oddelené tenkou vrstvou izolantu, predstavujú z elektrotechnického hľadiska kondenzátory. Frekvenčná charakteristika kondenzátorov s malou kapacitou so zvyšovaním frekvencie predstavuje hornú priepusť – izolácia medzi jednotlivými vodičmi prestáva plniť svoju funkciu s ohľadom na kapacitanciu a imaginárnu zložku impedancie. Vodiče zbernice samotné zase majú vlastné indukčnosti, ktoré sú tiež frekvenčne veľmi závislé. Pri vysokých frekvenciách teda predstavuje paralelná zbernica komplikovanú RLC sústavu, ktorá je frekvenčne veľmi závislá,
- *indukovanie napätia v susedných vodičoch* (crosstalk): striedavý prúd s vysokou frekvenciou generuje elektromagnetické pole, ktoré v susedných vodičoch vyvoláva indukované napätie. Toto je však z pohľadu signálu v susednom vodiči vnímané ako rušenie, ktoré môže spôsobiť superpozíciu signálov obidvoch vodičov, navyše fázovo posunutú. Tomuto javu sa hovorí crosstalk (presluch) a významne degraduje prenášaný signál,
- pri neharmonických signáloch, ktorými sú všetky digitálne signály, je nutné prenášať aj *vyššie harmonické*. Pôsobením hore opísaných parazitných javov však dochádza ku odfiltrovaniu týchto vyšších harmonických a ku degradácii prenášaného signálu,
- aby bola zaistená synchronizácia všetkých bitov v rámci paralelnej zbernice, *musia budiace tranzistory spínať v presne stanovených okamihoch s presnosťou v rádu desatín až stotín nanosekúnd*. Táto požiadavka však naráža na problémy - tranzistory majú konečné rýchlosti spínania, významná je aj výrobná tolerancia parametrov jednotlivých tranzistorov. Výsledkom je posunutie („skew“) signálov na jednotlivých linkách a problémy so synchronizáciou signálu jednotlivých liniek paralelnej zbernice.

V dôsledku uvedených problémov môžme v období po roku 2000 pozorovať odklon od paralelných zbernicových riešení a všeobecný príklon ku sériovým zberniciam. Dá sa predpokladať, že v budúcnosti, keď budú úspešne vyriešené problémy s parazitnými javmi a technologické problémy s budiacimi tranzistormi, príde ku opätovnému príklonu ku paralelným zberniciam.

3.2 Hlavné parametre zberníc

Najdôležitejším parametrom zbernice je *prenosová rýchlosť* (priepustnosť), ktorá sa udáva v bitoch za sekundu alebo bajtoch za sekundu a ich násobkoch (kb/s, MB/s a pod.). Prenosová rýchlosť je daná jednak spôsobom spracovania signálu (použitým kódovaním), ďalej *taktovacou frekvenciou zbernice* v Hz a u paralelných zberníc *bitovou šírkou dátovej zbernice* (počtom dátových vodičov zvyčajne 8, 16, 32 alebo 64). *Bitová šírka adresnej zbernice* (16, 20, 24, 32, 36) potom udáva adresný rozsah fyzickej pamäte, aký je zbernica schopná adresovať.

Spôsob prenosu môže byť bez modulácie (základné pásmo), alebo s moduláciou (preložené pásmo). *Modulačná rýchlosť* udáva počet prenosových stavov („znakov“) za jednotku času, jednotka je Baud.

U každej zbernice je definovaná jej *fyzická vrstva* a *logická vrstva*. Pod fyzickou vrstvou rozumieme parametre ako typ zbernice, počty vodičov, úlohy a charakteristiky signálových vodičov, napájanie, budenie, metóda zabezpečenia synchronizácie, úrovne napätí signálov, metóda kódovania, interpretácia signálov do logickej hodnoty, prevedenie konektorov, zapojenie pinov, atď. Pod logickou vrstvou rozumieme interpretáciu dát v systéme, spoluprácu zbernice s operačným systémom, ovládače zbernice, resp. rozhraní a pod.

U konkrétnych vyhotovení zberníc – *architektúr zbernicových systémov* – sú všetky parametre fyzickej aj logickej vrstvy presne definované, zvyčajne v systéme noriem IEEE, ANSI, ISO a pod. Definovanie všetkých dôležitých parametrov je nevyhnutné pre zabezpečenie kompatibility zbernice s prídavnými zariadeniami, ale aj so samotným výpočtovým systémom. Ak sú príslušné normy zverejnené a nelicencované, hovoríme o otvorenej architektúre (napr. ISA), ak sú príslušné parametre výhradným majetkom firmy, resp. viazané licenciami, ide o uzatvorenú architektúru (napr. zbernica MCA).

Zhrnutie hlavný parametrov:

- prenosová rýchlosť (MB/s, Gb/s),
- taktovacia frekvencia zbernice (MHz),
- bitová šírka dátovej zbernice (4, 8, 16, 32, 64),
- bitová šírka adresnej zbernice (16, 20, 24, 32, 36),
- spôsob prenosu (bez modulácie, s moduláciou),
- modulačná rýchlosť (baud),
- parametre fyzickej vrstvy (počet pinov, napájanie, budenie, kódovanie, atď.),
- parametre logickej vrstvy (interpretácia dát v systéme, ovládače, atď.),
- architektúra zbernicových systémov (otvorená, uzatvorená).

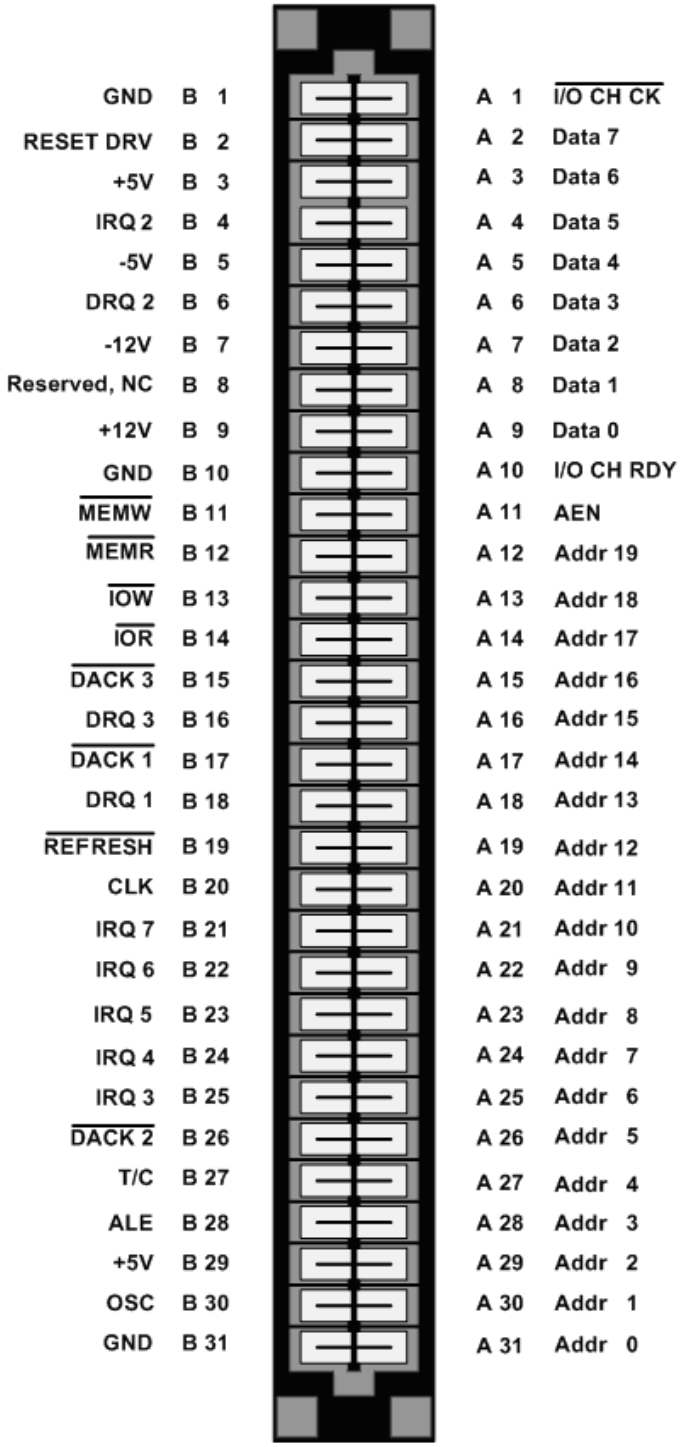
3.3 PC-BUS

Zbernica PC-BUS je určená pre procesor Intel 8088 (počítač PC XT), ktorý bol vnútorne 16 bitový, komunikoval s okolím prostredníctvom 8 bitovej dátovej zbernice (PC-BUS).

Zbernica bola centrálne riadená a obojsmerná. Konektor bol 62 pinový, pričom jeden bol rezervovaný (nepoužitý). Disponovala 20 bitovou adresnou zbernicou (Addr 0 až Addr 19), samozrejme 8 bitovou dátovou zbernicou (Data 0 až Data 7), šiestimi kanálmi prerušenia IRQ (číslo 2 až 7, IRQ 2 až IRQ 7), troma DMA kanálmi (č. 1 až 3, DRQ 1 až DRQ 3, DACK 1 až DACK 3 a REFRESH). Významy signálov prerušenia a priameho prístupu do pamäte sú popísané v kapitole 2. Nakoľko adresná zbernica PC-BUS mala 20 bitov adresovať sa dal iba 1 MB pamäte, čo bol aj v podstate maximum pre PC-XT.

Zbernica obsahovala napájacie kontakty + 12V, +5V, -5V a zem. Systémová časť zbernice obsahuje systémové hodiny (CLK) s periódou 210 ns (frekvencia 4,7 MHz) a nesynchronizované hodiny (signál základného oscilátora) so systémom (OSC) s frekvenciou 14,32 MHz (perióda 70 ns). Taktiež zbernica obsahuje signály typu prenosového cyklu (IOW, IOR, MEMR, MEMW) a signál pseudosynchrónnej zbernice I/O CH RDY, ktorý môžu využívať pomalšie moduly a vyžadovať spomalenie až o 10 períód hodín (2100 ns). Ďalej ešte obsahuje signál centrálneho nulovania (RESET DRV), diagnostický signál chyby prenosu (I/O CH CK), signalizáciu pretečenia čítača prenosu ľubovoľného DMA kanálu (signál T/C ukončí DMA prenos).

Na obrázku Obr. 3.4 je znázornený slot zbernice PC-BUS s rozložením jednotlivých pinov (signálov) a ich význam. Následne po obrázku v tabuľke Tab. 3.1 je vysvetlený význam a popis týchto signálov.



Obr. 3.4 Slot PC-BUS s rozložením pinov (signálov)

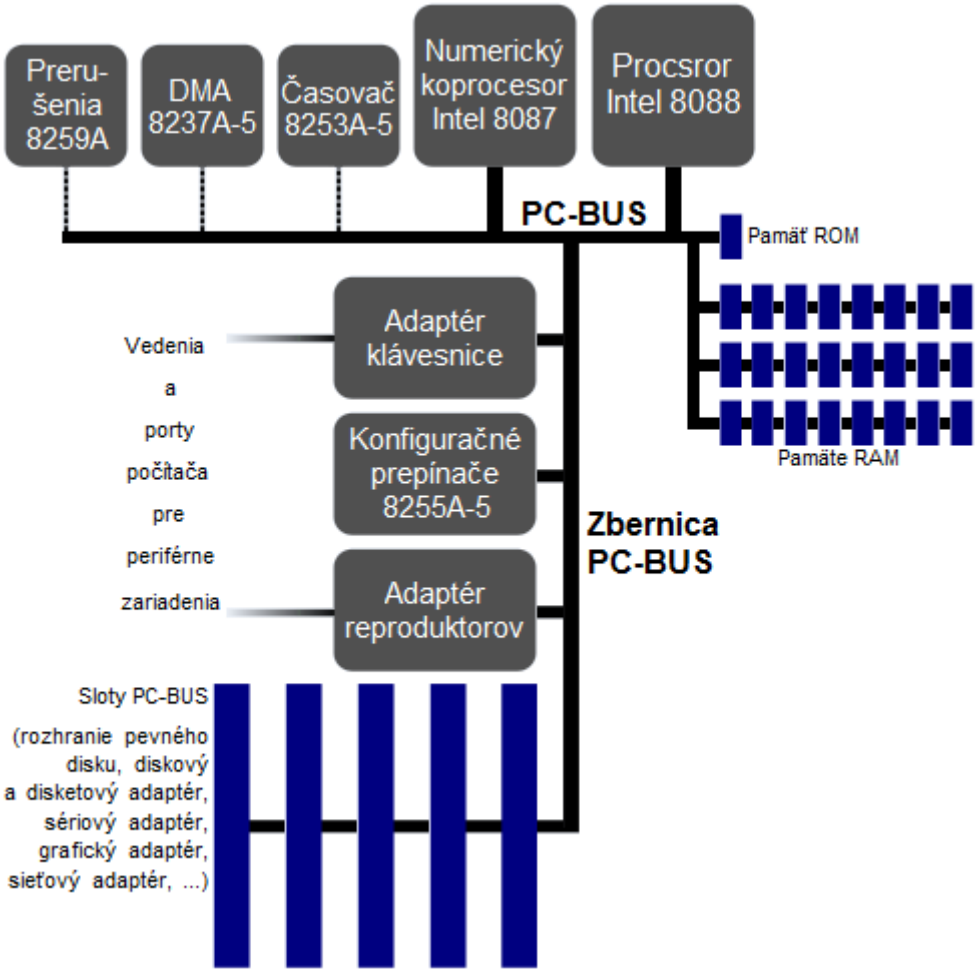
Tab. 3.1 Význam pinov na zbernici PC-BUS

Kontakt (pin)	Signál	Význam
A1	**I/O CH CK**	Diagnostický signál chyby prenosu (neg.).
A2 – A9	Data	Dátové signály (8 pinov).
A10	I/O CH RDY	Pripravenosť adaptéru k prenosu po zbernici.
A11	AEN	Potvrdenie adresy generovanej radičom DMA.
A12 – A31	Addr	Adresné signály (20 pinov).
B1, B10, B31	GND	Uzemnenie.
B2	RESET DRV	Inicializácia adaptéru.
B3, B5, B7, B9, B29	± 5 V, ± 12 V	Napájanie + 5 V, - 5 V, + 12 V, - 12 V.
B4, B21 – B25	IRQ 2 až IRQ 7	Signály prerušenia pre kanály 2 až 7.
B6, B15 – B18, B26	DACK, DRQ	Signály priameho prístupu do pamäte (1 až 3).
B8	Reserved, NC	Nevyužitý signál, alebo neskôr (PC-AT a vyššie) signál indikoval rýchli 8 (ISA8) alebo 16-bitový adaptér (ISA).
B11	**MEMW**	Indikuje zápis dát do pamäte (neg.).
B12	**MEMR**	Indikuje čítanie dát z pamäti (neg.).
B13	**IOW**	Indikuje zápis dát do adaptéru (neg.).
B14	**IOR**	Indikuje čítanie dát z adaptéra (neg.).
B19	**Refresh**	Informácia o prebiehajúcom cykle obnovy dynamickej pamäte (neg.).
B20	CLK	Zbernicový hodinový synchronizačný signál.
B27	T/C	Ukončenie prenosu bloku dát niektorým kanálom DMA.
B28	ALE	Platnosť adresy na pinoch A12 až A31.
B30	OSC	Hodinová frekvencia 14,32 MHz (nesynchron.).

Signál ALE synchronizuje uloženie adresy do vyrovnávacej pamäti, takže deklaruje jej platnosť na adresných linkách. Signálom AEN získava radič DMA riadenie adresnej, dátovej a riadiacej zbernice a odpája od nich CPU, signál Refresh indikuje prebiehajúci cyklus obnovy informácie v dynamických pamätiach.

Pre zobrazenie (komunikáciu) sa používajú TTL úrovne. Žiaden modul nesmie zaťažiť ľubovoľnú linku zbernice viac než dvoma ekvivalentnými vstupmi. Tolerancia napájacích napätí pre zdroje + 5 V a + 12 V je 5 % a pre zdroj – 12 V je 10 %.

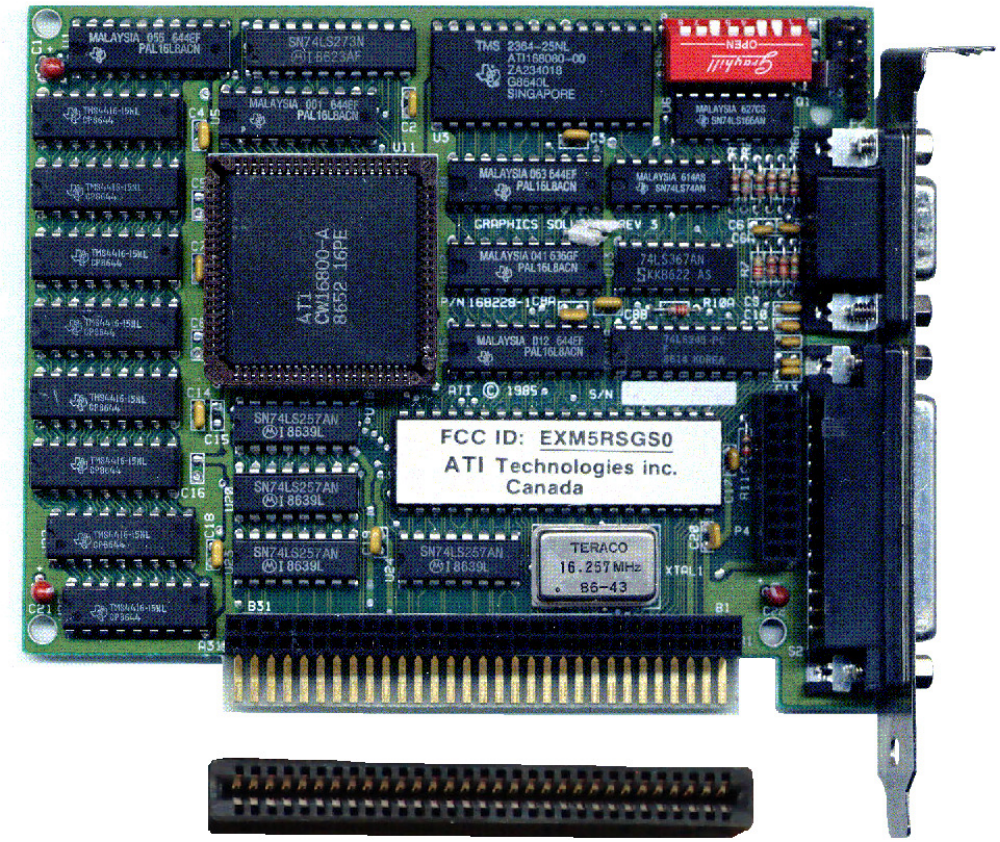
Táto zbernica v počítači PC-XT bola zároveň aj systémovou zbernicou, takže všetka komunikácia (aj priamo na matičnej doske) prebiehala na tejto zbernici. Časovanie zbernice vychádza z časovania mikroprocesora 8088. Cyklus zbernice odpovedá jednému strojovému cyklu a má minimálne štyri takty (periódy systémových hodín). Tie stačia len pre komunikáciu s pamäťou, pri komunikácií s pomalšími obvodmi v adaptéroch (V/V adresy) sa musí procesor pozastavovať a zbernicový cyklus sa tak predĺži. Na obrázku Obr. 3.5 je znázornenie prepojenia všetkých prvkov (komponentov) počítača PC-XT pomocou zbernice PC-BUS, z tohto obrázka je zrejmé, že zbernica PC-BUS bola vstupno-výstupnou zbernicou a systémovou zbernicou súčasne.



Obr. 3.5 Zapojenie komponentov počítača PC-XT do zbernice PC-BUS

Niekedy sa táto zbernica označovala aj ako ISA 8, pričom číslo 8 naznačovalo šírku dátovej zbernice. Toto označenie sa začalo používať až po zavedení zbernice ISA.

Na obrázku Obr. 3.6 je príklad grafickej karty určenej pre zbernicu PC-BUS na tomto obrázku sa dá vidieť vzhľad konektoru tejto zbernice ako aj vzhľad slotu.



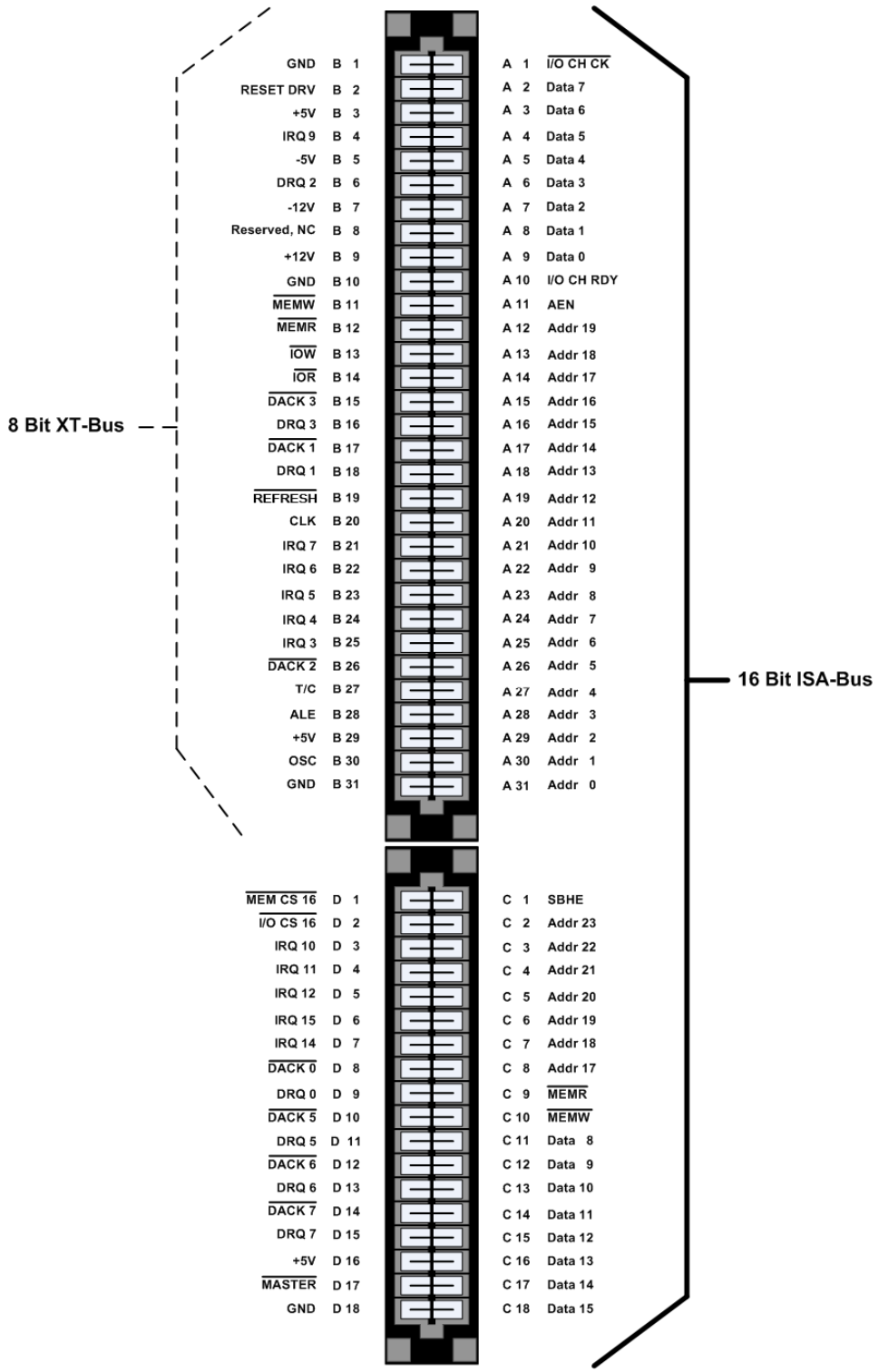
Obr. 3.6 Grafická karta pre zbernicu PC-BUS (hore) so slotom zbernice PC-BUS (dole)

3.4 ISA

Nástupom PC-AT (CPU 80286) sa štandardom stala 16 bitová zbernica ISA (Industry Standard Architecture), ktorá zachováva spätnú kompatibilitu s 8 bitovou PC-BUS. Zbernica ISA vychádzala zo zbernice PC-BUS, signály sú v podstate jej nadmnožinou. Sloty pre ISA karty sú riešené ako dvojdielne. Prvý diel je pôvodný 62 pinový a slúži aj na osadenie starších 8 bitových kariet, druhý diel rozširuje možnosti zbernice a umožňuje použiť karty pracujúce v 16 bitovom režime.

Okrem dátovej zbernice sa rozšírila aj adresná zbernica na 24 bitov čo rozšírilo adresovanie pamäte na 16 bitov. Šírka adresnej a dátovej zbernice ISA plne odpovedala možnostiam PC-AT (CPU 80286). K dispozícií je 7 DMA kanálov (0, 1, 2, 3, 5, 6, 7), čo je o 4 viac oproti PC-BUS. Rozšírili sa aj IRQ kanály okrem kanálov 2 až 7 pribudli kanály 10 až 12, 14 a 15. V ére procesorov 80286 dosiahla obrovského rozšírenia, bola používaná prakticky vo všetkých IBM PC kompatibilných zostavách.

Na obrázku Obr. 3.7 je znázornený slot zbernice ISA s rozložením jednotlivých pinov (signálov) a ich význam. Následne po obrázku v tabuľke Tab. 3.2 je vysvetlený význam a popis týchto signálov.



Obr. 3.7 Slot ISA s rozložením pinov (signálov)

Tab. 3.2 Význam pinov na zbernici ISA

Kontakt (pin)	Signál	Význam
A1 – A31, B1 – B31	PC-BUS	Rovnaký význam ako pri PC-BUS.
C1	SBHE	Indikuje použitie vrchných dátových bitov.
C2 – C8	Addr (vrchné)	Vrchné adresné signály (17. až 23. bit).
C9	**MEMR**	Čítanie dát z ktoréhokoľvek miesta v pamäti (pôvodný signál na B12 funguje len pre 1. MB).
C10	**MEMW**	Zápis dát na ktorékoľvek miesto v pamäti. (pôvodný signál na B11 funguje len pre 1. MB).
C11 – C18	Data (vrchné)	Vrchné dátové signály (8. až 15. bit).
D1	**MEM CS 16**	Indikuje 16 bitový prenos dát z pamäte a do pamäte.
D2	**I/O CS 16**	Indikuje 16 bitový prenos dát z V/V a do V/V zariadenia.
D3 – D7	IRQ	Signály prerušenia pre kanály 10 až 12, 15 a 14.
D8 – D15	DACK, DRQ	Signály priameho prístupu do pamäte (0, 5 – 7).
D16	+ 5 V	Napájanie + 5 V.
D17	**MASTER**	Potvrdenie prevzatia zbernice iným master-om, ten je súčasťou adaptéru, žiadosť sa predáva pomocou niektorého DRQ.
D18	GND	Uzemnenie.

Zbernica bola taktovaná na 6 až 25 MHz, bola synchronizovaná s taktom procesora. Veľmi dôležitým rozdielom je to, že je typu multimaster a jej arbiter je decentralizovaný. Adresná zbernica a väčšina signálov riadiacej zbernice je práve preto obojsmerná.

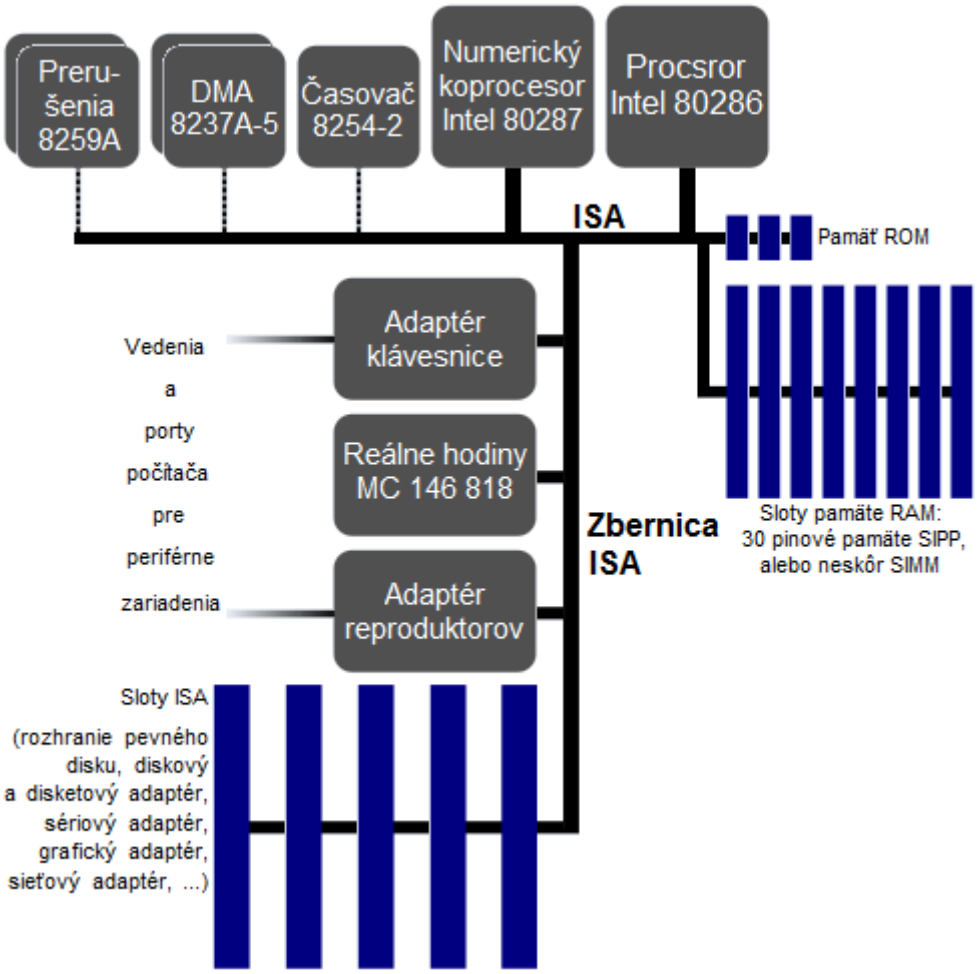
Na prvý pohľad je v adresnej časti nepochopiteľné zdvojenie troch adresných liniek: Addr 17, 18 a 19. Dôvod tohto plytvania pinov na konektore bola v kompatibilite. Kým mikroprocesory 8088 a 8086 generovali platnú adresu behom jedného taktu zbernicového cyklu, bola tá istá adresa pri CPU 80286 už k dispozícii (behom prvého taktu minulého zbernicového cyklu). Tak sa prejavovalo prúdové spracovanie, ktoré činnosť CPU pri spolupráci s pamäťou výrazne urýchľovalo. Pokiaľ bola celá pamäť na matičnej doske, bolo možné už pri návrhu túto skutočnosť zohľadniť, vybrať a osadiť také obvody, ktoré boli takejto spolupráce schopné. Predĺžená alebo rozšírená pamäť (viď kapitolu 2.1) sa však niekedy umiestňovala do pamäťového adaptéru a tak museli byť na zbernici k dispozícií adresné signály generované s týmto predstihom. Boli nimi signály na adresných vodičoch Addr 17 až Addr 23. Z dôvodov obmedzeného počtu pinov (nožičiek) konektoru nebolo možné zdvojiť aj ďalšie adresné vodiče, a tak nižšie adresné bity (Addr 0 až Addr 16) boli k dispozícií až o jeden takt neskôr. Medzi tým ale adresovaná doska odpovedala na vodiči MEM CS 16, že je pripravená na 16 bitový prenos dát.

Kompatibilitu medzi 16-bitovou matičnou doskou a 8-bitovými adaptérmi (kartami) zaisťovali zachytené adresné signály na vodičoch Addr 0 až Addr 19 (A12 až A31). Aby sa dodržalo správne časovanie, vkladala systémová doska takty navyše. Ak matičná doska dostala signál SBHE (požiadavka na 16-bitový prenos) a neprišla odpoveď MEM CS 16 alebo I/O CS 16, špeciálne obvody systémovej jednotky prevzali riadenie prenosu v dvoch nasledujúcich 8-bitových cykloch.

Signály pre riadenie čítania a zápisu pamäti boli taktiež zdvojené, aj tu bol dôvod zachovania spätnej kompatibility. Kým MEMR a MEMW na B11 a B12 obsluhoval 1 MB pamäti na matičnej doske, signály MEMR a MEMW na C9 a C10 riadili zápis a čítanie kdekoľvek v adresnom priestore. Kompatibilita bola zaistená aj prispôsobením rýchlosti zbernice rýchlym 8 a 16-bitovým adaptérom, pomocou signálu B8. Ak tento signál bol aktívny vynechá systémová doska niektoré alebo všetky dodatočné takty.

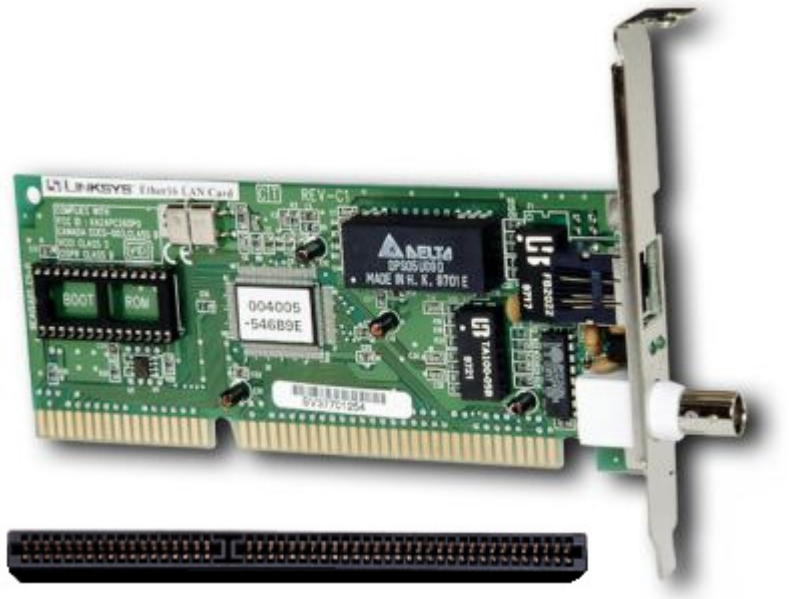
Zvyšné signály mali rovnaký význam ako signály na zbernici PC-BUS. Aj pri tejto zbernici sa pre zobrazenie (komunikáciu) používali TTL úrovne. Dvojdielny konektor mal 62 a 36 pinov, takže spolu 98 pinov (kontaktov).

Na obrázku Obr. 3.8 je znázornenie prepojenia všetkých prvkov počítača PC-AT pomocou zbernice ISA, z tohto obrázka je zrejmé, že zbernica ISA bola stále vstupno-výstupnou zbernicou a systémovou zbernicou ako pri zbernici PC-BUS.



Obr. 3.8 Zapojenie komponentov počítača PC-AT do zbernice ISA

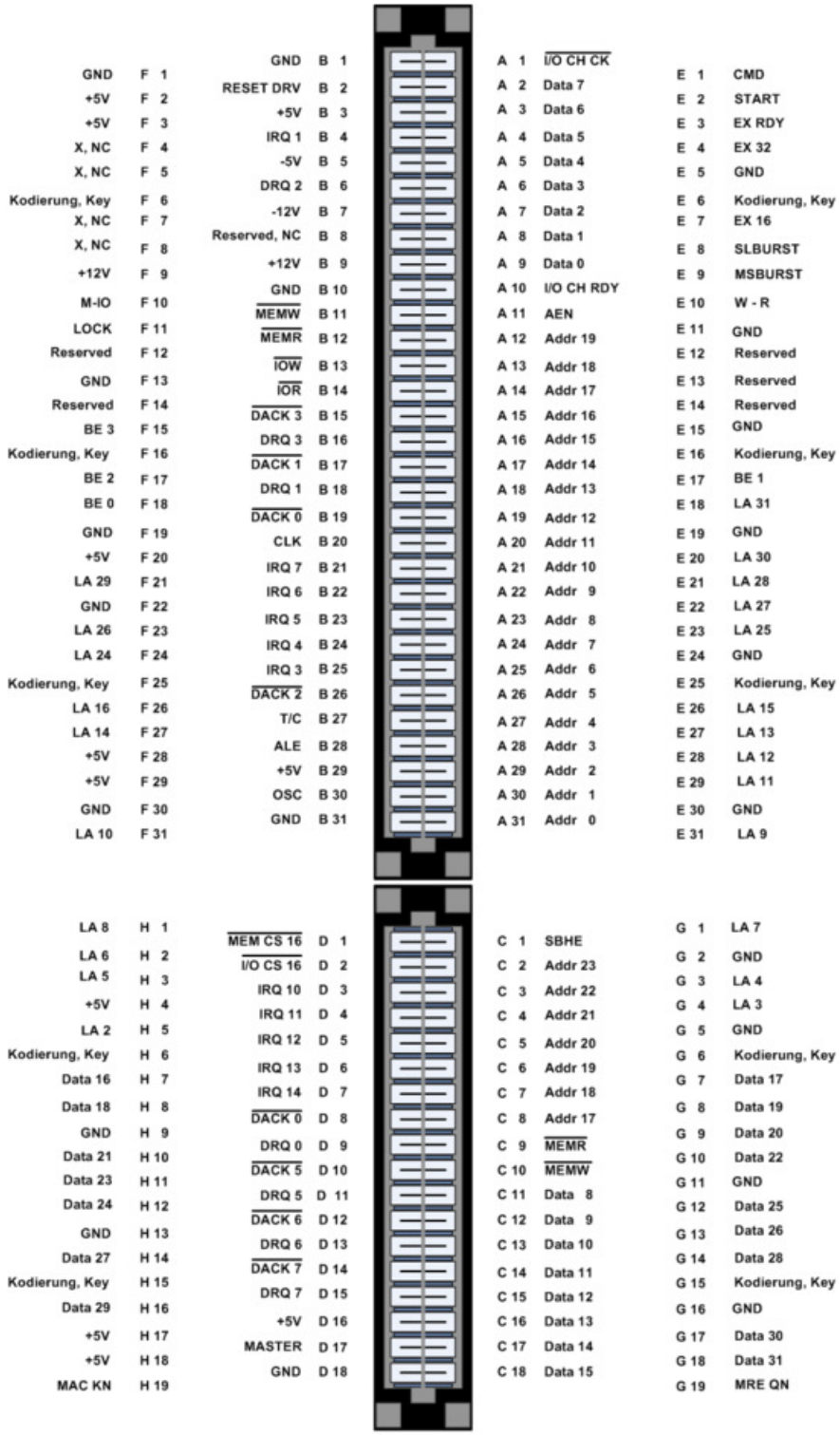
Na obrázku Obr. 3.9 je príklad sieťovej karty určenej pre zbernicu ISA, na tomto obrázku sa dá vidieť vzhľad konektoru tejto zbernice ako aj vzhľad slotu.



Obr. 3.9 Sieťová karta pre zbernicu ISA (hore) so slotom zbernice ISA (dole)

3.5 EISA

Vznikla v roku 1989 ako reakcia na MCA (konkurenčná 32 bitová zbernica). Firma Compaq spolu s ďalšími firmami vyvinuli konkurenčný projekt: Extended ISA – EISA. Je spätne plne kompatibilná s ISA má 32 dátových a 32 adresných vodičov (adresovanie 4 GiB), kanály IRQ č. 2 – 15, 64 I/O adries pre I/O zariadenia, zdieľané riadenie zberníc. Vzhľadom je slot pre EISA rovnaký, ako ISA slot, rozšírenie počtu vodičov je dosiahnuté umiestnením kontaktov v dvoch „poschodiach“ nad sebou. Slot EISA teda dokáže pracovať s 8 bitovými, 16 bitovými aj 32 bitovými adaptérmi – rozozná typ prídavnej dosky a automaticky nastaví parametre prenosu: 1 bajtový, blokový – viac bajtový. Riešenie sa používalo u drahých výkonných počítačov, ukázalo ako veľmi drahé a nepresadilo sa. S príchodom zbernice PCI potom stratilo celkom opodstatnenie. Na obrázku Obr. 3.10 je znázornený slot zbernice EISA s rozložením jednotlivých pinov (signálov) a ich význam. Následne po obrázku v tabuľke Tab. 3.3 je vysvetlený význam a popis týchto signálov.

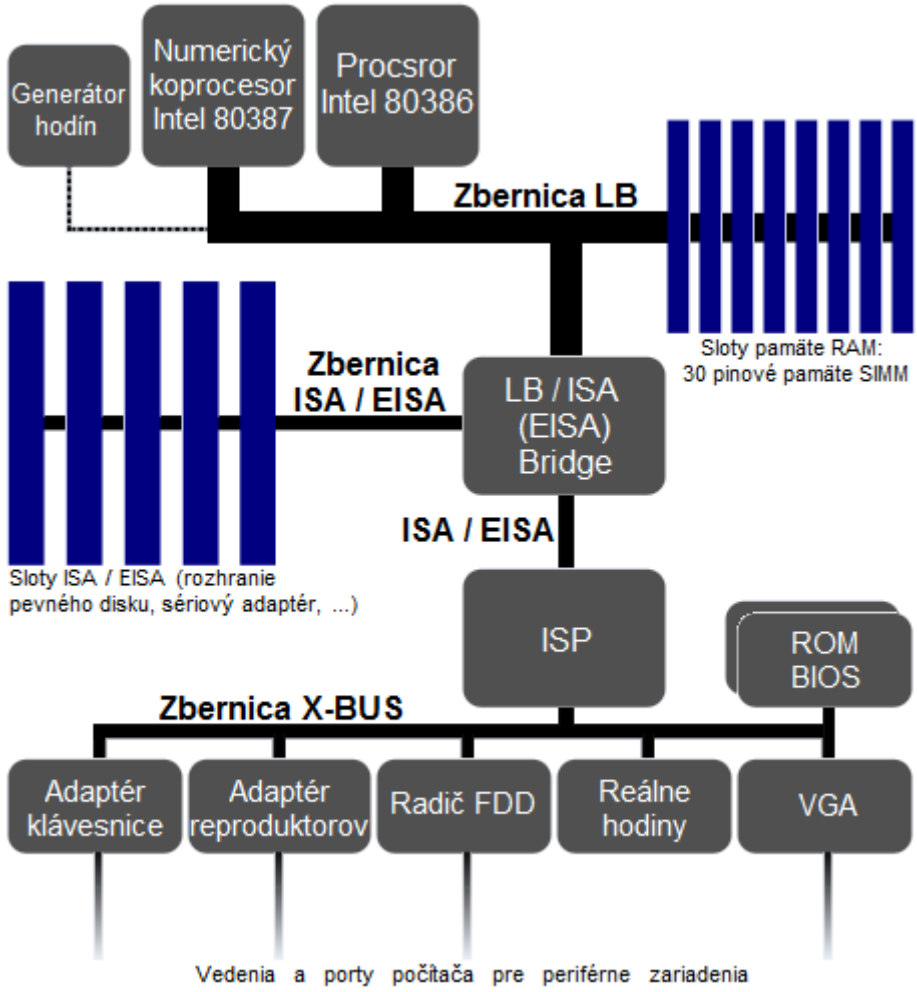


Obr. 3.10 Slot EISA s rozložením pinov (signálov)

Tab. 3.3 Význam pinov na zbernici EISA

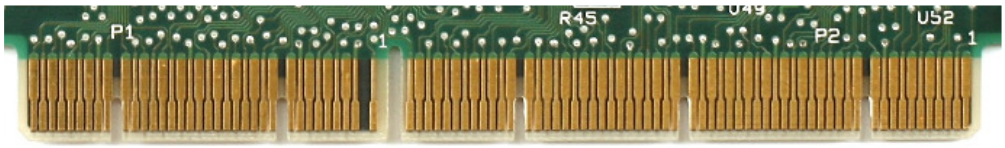
Kontakt (pin)	Signál	Význam
A, B, C, D	ISA	Rovnaký význam ako pri ISA.
E1	CMD	Časový signál behom EISA zbernicového cyklu
E2	START	Indikácia začiatku EISA zbernicového cyklu.
E3	EX RDY	Karta je pripravená ukončiť cyklus
E4	EX 32	Vložená 32 bitová EISA karta.
E5, E11, E15, E19, E24, E30, F1, F13, F19, F22, F30, G2, G5, G11, G16, H9, H13	GND	Uzemnenie.
E6, E16, E25, F6, F16, F25, G6, G15, H6, H15	Key	Fyzická zarážka pre kartu EISA.
E7	EX 16	Vložená 16 bitová EISA karta.
E8	SLBURST	Podriadená karta indikuje, že môže akceptovať blokové cykly.
E9	MSBURST	Nadriadená karta indikuje, že môže prevádzať blokové cykly.
E10	W/R	Čítanie a zápis dát na ktorékoľvek miesto v pamäti (podobne ako u ISA).
E12–E14, F12, F14	Reserved	Nevyužitý signál.
E17, F15, F17, F18	BE 0 až BE 3	Povolenie konkrétneho bajtu na zbernici.
E18, E20–E23, E31, E26–E29, F21, F23, F24, F26, F27, F31, G1, G3, G4, H1–H3, H5	LA	Adresné signály.
F2, F3, F9, F20, F28, F29, H4, H17, H18	+ 5 V, + 12 V	Napájanie + 5 V, + 12 V.
F4, F5, F7, F8	X, NC	Nevyužitý signál.
F10	M-IO	Rozlíšenie medzi pamäťovým a V/V cyklom.
F11	LOCK	Zamknutie.
G7–G10, G12–G14, G17, G18, H7, H8, H10–H12, H14, H16	Data	Dátové signály (16. až 31. bit).
G19	MRE QN	Žiadosť nadriadeného obvodu o zbernicu.
H19	MAC KN	Potvrdenie nadriadenému obvodu o pridelení zbernice.

V čase PC-386 sa stále využívala ISA ako vstupno-výstupná zbernica, ale museli nastať isté zmeny, nakoľko procesory už boli 32 bitové a taktovacia frekvencia procesora bola natoľko vysoká, že ISA nepostačovala na komunikáciu so samotnou operačnou pamäťou. Tieto zmeny sa vyriešili lokálnou zbernicou počítača (LB – Local Bus), ktorá slúžila na komunikáciu medzi procesormi (procesor a numerický koprocesor) a operačnou pamäťou. Nakoľko počítač už obsahoval dve zbernice s rôznou charakteristikou vznikol na počítači bridge. Na obrázku Obr. 3.11 je znázornené zapojenie komponentov počítača PC-386 do zberníc LB a ISA prepojených pomocou LB/ISA Bridge-u. Okrem zdvojenia zberníc sa na obrázku dá všimnúť, že obvody prerušenia a DMA zmizli, obvody DMA (8237A-5) sa stali súčasťou LB/ISA Bridge a obvody prerušenia (8259A) sa stali súčasťou ISP (Integrated system peripheral), ktorý okrem prerušení mal na starosti aj pripojenie integrovaných periférií s procesorom a pamäťou pomocou zbernice X-Bus. Neskôr aj počítače PC-386 obsahovali zbernice EISA, koncepcia sa zachovala, len LB/ISA Bridge nahradil LB/EISA Bridge (alebo presnejšie: EBC – EISA bus controller). Počítače PC-486, ktoré neobsahovali vstupno-výstupnú zbernicu VL BUS mali rovnakú koncepciu ako je na obrázku Obr. 3.11 len neobsahovali numerický koprocesor, pretože sa stal súčasťou procesora.



Obr. 3.11 Zapojenie komponentov počítača PC-386 do zberníc LB, ISA a X-Bus

Na obrázku Obr. 3.12 je konektor V/V karty zbernice EISA, dôvodom prečo nie je uvedený príklad celej karty je, aby sa dal všimnúť detail dvoch radov kontaktov a miesta nazvané Key v tabuľke Tab. 1.1Tab. 3.3. Slot EISA je veľkosťou aj vonkajším vzhľadom rovnaký ako ISA, líši sa len farbou (používala sa sivá až biela farba pre tento slot) a vnútornou konštrukciou.



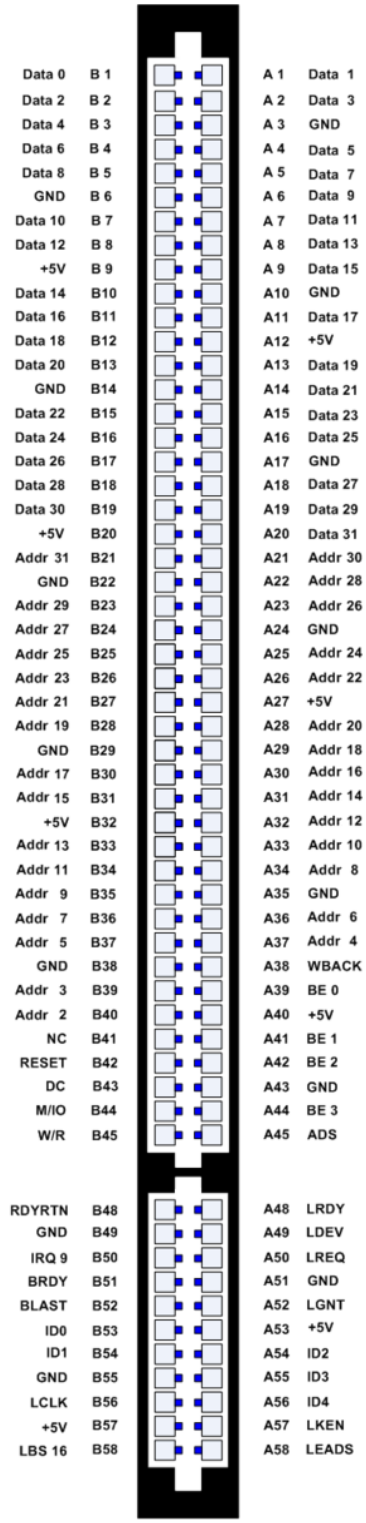
Obr. 3.12 Konektor zbernice EISA

3.6 VL BUS

Nárast požiadaviek na komunikáciu s grafickým rozhraním si vynútil posilnenie tohto komunikačného kanálu v podobe nového štandardu VL BUS (VESA Local Bus). Pripája grafický adaptér, prípadne rozhranie pre HDD a iné periférie s vysokými nárokmi na komunikáciu priamo na rýchlu lokálnu zbernicu, taktovanú na frekvencii procesora (až 50 MHz). Umožňuje použitie maximálne 3 rozširujúcich slotov na matičnej doske. Sloty sa používali spoločne s ISA slotmi ako rozširujúce pätice v jednom rade so slotmi ISA, mali 2x58 pinov. Ak boli všetky sloty obsadené, mohla pracovať max. na frekvencii 33 MHz. Bitová šírka adresnej aj dátovej zbernice je 32 bit, teoretická priepustnosť dát pri 33 MHz a 32 bitovej šírke dát je 132 MB/s. Vyšších prenosových rýchlostí dosahuje v „burst“ režime – adresa sa vyšle iba v prvom cykle, v troch nasledujúcich cykloch sa prenášajú iba dáta, ktoré sa zapisujú postupne do ďalších pamäťových buniek. Dá sa použiť iba pri zápise/čítaní do/z súvislej oblasti po sebe nadväzujúcich pamäťových buniek. Vonkajšia časť zbernice je k lokálnej pripojená cez radič ISA a pracuje na štandarde ISA. Zbernica sa používala pre zostavy s 80486 (a prvé procesory 5x86), jej veľkou nevýhodou bola priama závislosť na procesore s ktorým bola viazaná spoločne zdieľanou systémovou zbernicou.

Tento typ zbernice sa používal (vyrábal) v rokoch 1992 až 1994 pre PC – 486. Na obrázku Obr. 3.13 je grafická karta s konektorom a slotom zbernice VL BUS. Na obrázku Obr. 3.14 a tabuľke Tab. 3.4 je rozširujúci slot (bez časti ISA) s rozložením pinov a popisom.

Obr. 3.13 Grafická karta pre zbernicu VL BUS (hore) a slot zbernice VL BUS (dole)

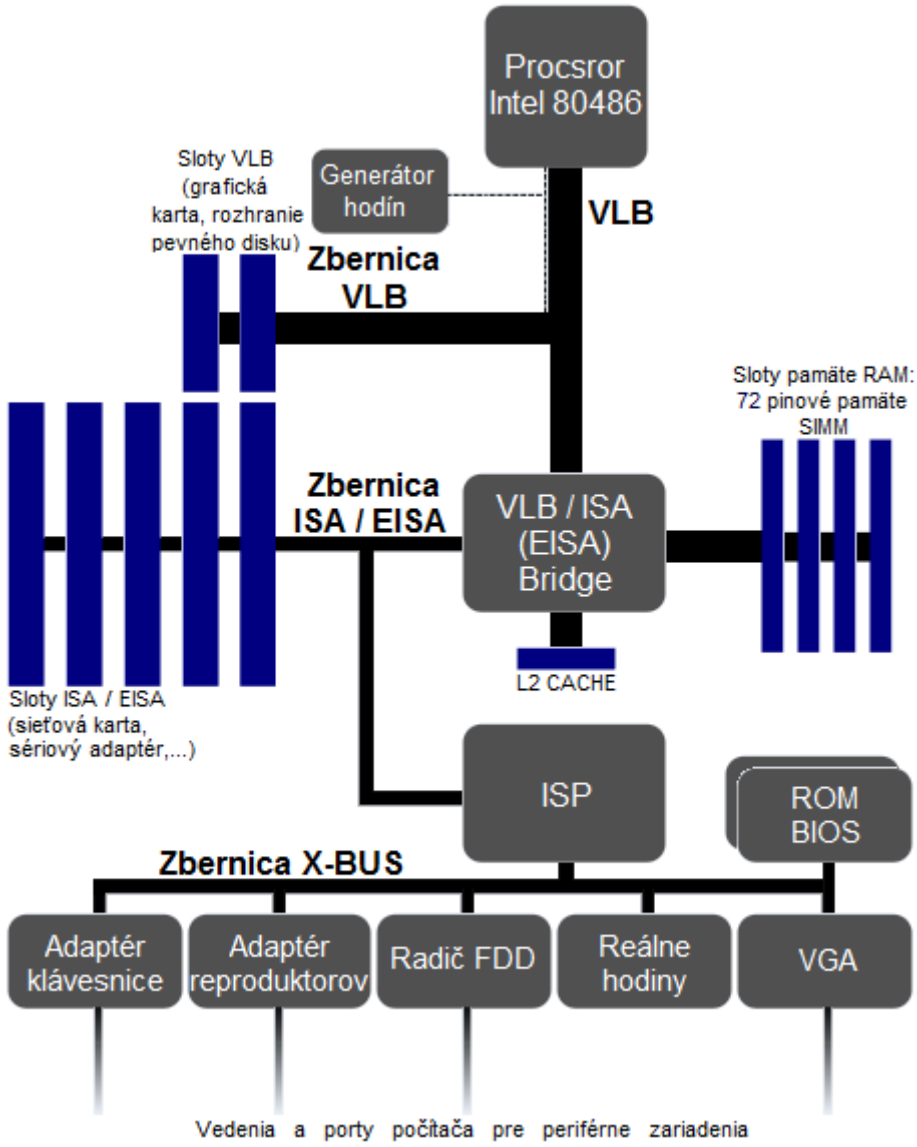


Obr. 3.14 Rozširujúci slot VL BUS s rozložením pinov (signálov)

Tab. 3.4 Význam pinov na zbernici VL BUS

Kontakt (pin)	Signál	Význam
Prvý slot	ISA	Rovnaký význam ako pri ISA.
A1, A2, A4 – A9, A11, A13 – A16, A18 – A20, B1 – B5, B7, B8, B10 – B13, B15 – B19	Data	Dátové signály.
A3, A10, A17, A24, A35, A43, A51, B6, B14, B22, B29, B38, B49, B55	GND	Uzemnenie.
A12, A27, A40, B9, A53, B20, B32, B57	+ 5 V	Napájanie + 5 V.
A21 – A23, A25, A26, A28 – A34, A36, A37, B21, B23 – B28, B30, B31, B33 – B37, B39, B40	Addr	Adresné signály.
A38	WBACK	Spätný zápis.
A39, A41, A42, A44	BE 0 až 3	Povolenie konkrétneho bajtu na zbernici.
A45	ADS	Adresný impulz.
A46, A47, B46, B47	Key	Fyzická zarážka karty VL BUS.
A48	LRDY	Lokálna rozširujúca karta je pripravená.
A49	LDEV	Lokálna rozširujúca karta je pripojená.
A50	LREQ	Požiadavka lokálnej rozširujúcej karty.
A52	LGNT	Udelenie práva vysielať pre lokálnu rozširujúcu kartu.
A54 – A56, B53, B54	ID	Identifikácia zariadenia
A57	LKEN	Lokálna rozširujúca karta je zapnutá.
A58	LEADS	Povolený adresný impulz lokálnej rozširujúcej karty.
B41	NC	Rezervovaný signál, nepoužitý.
B42	RESET	Vynulovanie (reset) pripojeného zariadenia.
B43	DC	Dátový príkaz.
B44	M-IO	Rozlíšenie medzi pamäťovým a V/V cyklom.
B45	W/R	Čítanie a zápis dát na ktorékoľvek miesto v pamäti (podobne ako u EISA).
B48	RDYRTN	Pripravený na vrátenie.
B50	IRQ 9	Prerušovací kanál IRQ 9.
B51	BRDY	Zrýchlený mód pripravený
B52	BLAST	Ukončenie zrýchleného módu.
B56	LCLK	Systémové hodiny lokálnej rozširujúcej karty.
B58	LBS 16	Lokálna rozširujúca karta je 16-bitová.

Na obrázku Obr. 3.15 je znázornenie zapojenia komponentov do zberníc v počítači PC-486. Local Bus sa kvôli svojmu rozšíreniu o vstupno-výstupné sloty premenovala na VLB. Úlohou VLB/ISA Bridge-u už bolo aj prepojenie procesora s operačnou pamäťou a tak tiež s externou CACHE pamäťou L2.



Obr. 3.15 Zapojenie komponentov počítača PC-486 do zberníc VLB, ISA (alebo EISA) a X-Bus

3.7 PCI

Keďže zbernica s VL BUS predstavovala iba rozšírenie či doplnok klasickej zbernice ISA, ukázalo sa, že s príchodom nových technológií je nevyhnutné vyvinúť celkom novú koncepciu zbernice s dostatočnou rezervou priepustnosti. Na trhu sa objavuje v ére

procesorov 80486, a u zostáv s týmito procesormi postupne nahrádzala už prekonanú VL BUS. Zostavy s Pentiami už boli prakticky výhradne postavené na nových základných doskách so zbernicou PCI (Peripheral Component Interconnect).

Koncepcia zbernicovej sústavy postavená na základe PCI už striktne rozdeľuje časť určenú na komunikáciu medzi procesorom a pamäťou a zbernicou pre V/V zariadenia. Zbernica pre komunikáciu medzi procesorom a pamäťou sa nazýva Front Side Bus (FSB) a zbernica pre vstupno-výstupné zariadenia sa nazýva PCI. Prepojenie medzi zbernicami je tvorené „bridges" – čipmi chipsetu základnej dosky. Táto koncepcia zabezpečuje PCI väčšiu univerzálnosť, presadila sa nielen vo svete PC, ale aj Apple a ďalších počítačov s procesormi RISC. Pamäťová zbernica, označená ako FSB je 64 bitová, taktovaná nezávisle od procesora na 33 MHz, až po 533 MHz. Bola osadzovaná pamäťami SDRAM až DDR3 SDRAM.

PCI ver. 1.0 (1992)

Taktovanie pôvodnej PCI zbernice je 33 MHz, používa. paralelný 32 bitový dátový prenos, pre signalizáciu používa 5V. Preberá spätnú kompatibilitu s ISA aj EISA, IRQ 2 – 15, DMA 1 – 7, zdieľané riadenie zberníc. Prenosová rýchlosť dosahuje 132 MB/s pre 32 bitovú verziu, pričom túto prenosovú rýchlosť zdieľajú všetky súbežne komunikujúce zariadenia na tejto zbernici. Zbernica používa časový multiplex na prenos adresy aj dát po tej istej fyzickej zbernici. Objavuje sa nový prvok v technológii rozširujúcich adaptérov – Plug & Play (zbernica sa „vie dohodnúť" s BIOS-om aj operačným systémom a nastaviť si sama vhodné hodnoty vektorov prerušenia pre jednotlivé zariadenia. Na základných doskách s PCI zbernicou sú z dôvodu kompatibility spravidla aj sloty pre ISA zbernicu – časť zbernice teda pracuje v režime ISA (tieto vymizli zo základných dosiek až s príchodom Pentia 4 koncom 90-tych rokov). Neskoršie verzie pracujú už so 64 bitovou dátovou šírkou, čo umožňuje prenos rýchlosťou až 264 MB/s. Objavuje sa signalizácia 3,3 V.

PCI ver. 2.1 (1995)

Taktovanie sa zvyšuje na 66 MHz, čo umožňuje zvýšenie prenosových rýchlostí na 264 MB/s pre 32 bitovú verziu a 528 MB/s pre 64 bitovú PCI Vzájomná kompatibilita: Kompatibilita funguje obojstranne; do 33 MHz zbernice je možné použiť 66 MHz kartu a naopak do 66 MHz zbernice sa dá použiť 33 MHz karta. V obidvoch prípadoch však takt celého zbernicového systému bude upravený na 33 MHz. Napájacie napätie PCI kariet sa tiež líši – existujú verzie s napájaním 5V aj verzie s napájaním 3,3V. Niektoré rozširujúce karty dokázali identifikovať napájacie napätie slotu a prispôsobiť sa, väčšina kariet však vyžadovala verziu PCI zbernice so správnym napájaním.

Závislosť od ISA

PCI je na rozdiel od väčšiny uvedených predchádzajúcich zberníc od ISA zbernice a jej služieb celkom nezávislá, môže sa používať na základnej doske celkom autonómne bez potreby spolupráce s ISA. Z dôvodov spätnej kompatibility však zbernica ISA figuruje na základných doskách aj v ére Pentií 4. V čase nástupu zbernice PCI bola jej priepustnosť celkom vyhovujúca aj pre grafické karty (priepustnosť VL BUS a PCI bola zhodná), a rozhranie PCI sa stalo štandardom aj v oblasti grafických adaptérov. S narastajúcimi požiadavkami na výkon grafického subsystému prestala byť priepustnosť PCI pre výkonné grafické karty vyhovujúca už od polovice 90-tych rokov. Prichádza špecializovaný grafický zbernicový systém AGP.

Zbernica PCI - X

PCI - X je novšia verzia konvenčného PCI štandardu s podporou vyšších prenosových rýchlostí. Používa rovnakú architektúru, kompatibilný konektor a je spätne

kompatibilná s klasickou 32/64 bitovou PCI, 66 MHz, 3,3 V. Používa paralelný aj 32 bitový prenos, existuje aj 64 bitová verzia. Používa nový komunikačný protokol na zlepšenie efektivity prenosu dát, (na prenos zavádza novú jednotku MTS – megatransfers per second) a najmä pracuje na vyšších taktovacích frekvenciách. Rozšírila sa najmä v serverových systémoch.

Verzia 1.0 špecifikuje:

- PCI-X – 66: je taktovaná na 66 MHz (264 MB/s),
- PCI-X – 133: je taktovaná na 133 MHz (533 MB/s) pre 32 bitovú verziu, alebo viac ako 1GB/s pre 64-bit zariadenia.

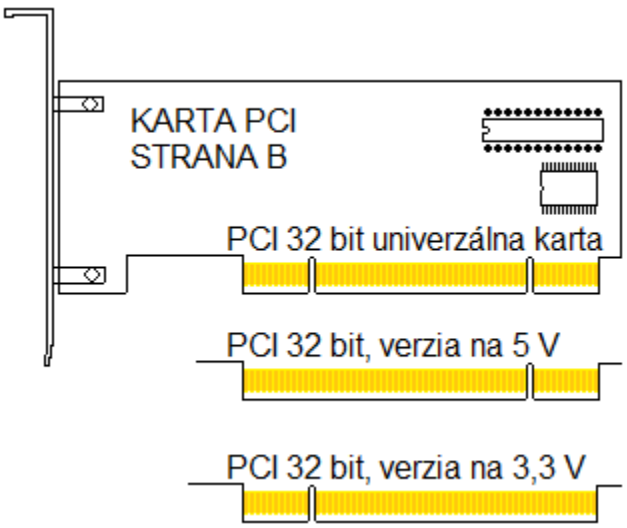
Verzia 2.0 špecifikuje:

- PCI-X 266 s transferom 266 MTS (2133 MB/s), fyzicky taktovaná na 133 MHz, umožňuje vykonať až dva prenosy (zápisy alebo čítania) v priebehu jediného taktu,
- PCI-X 533 s transferom 533 MTS (4224 MB/s), fyzicky taktovaná na 133 MHz, umožňuje vykonať až štyri prenosy (zápisy alebo čítania) v priebehu jediného taktu.

Prínosy a vlastnosti PCI - X 2.0:

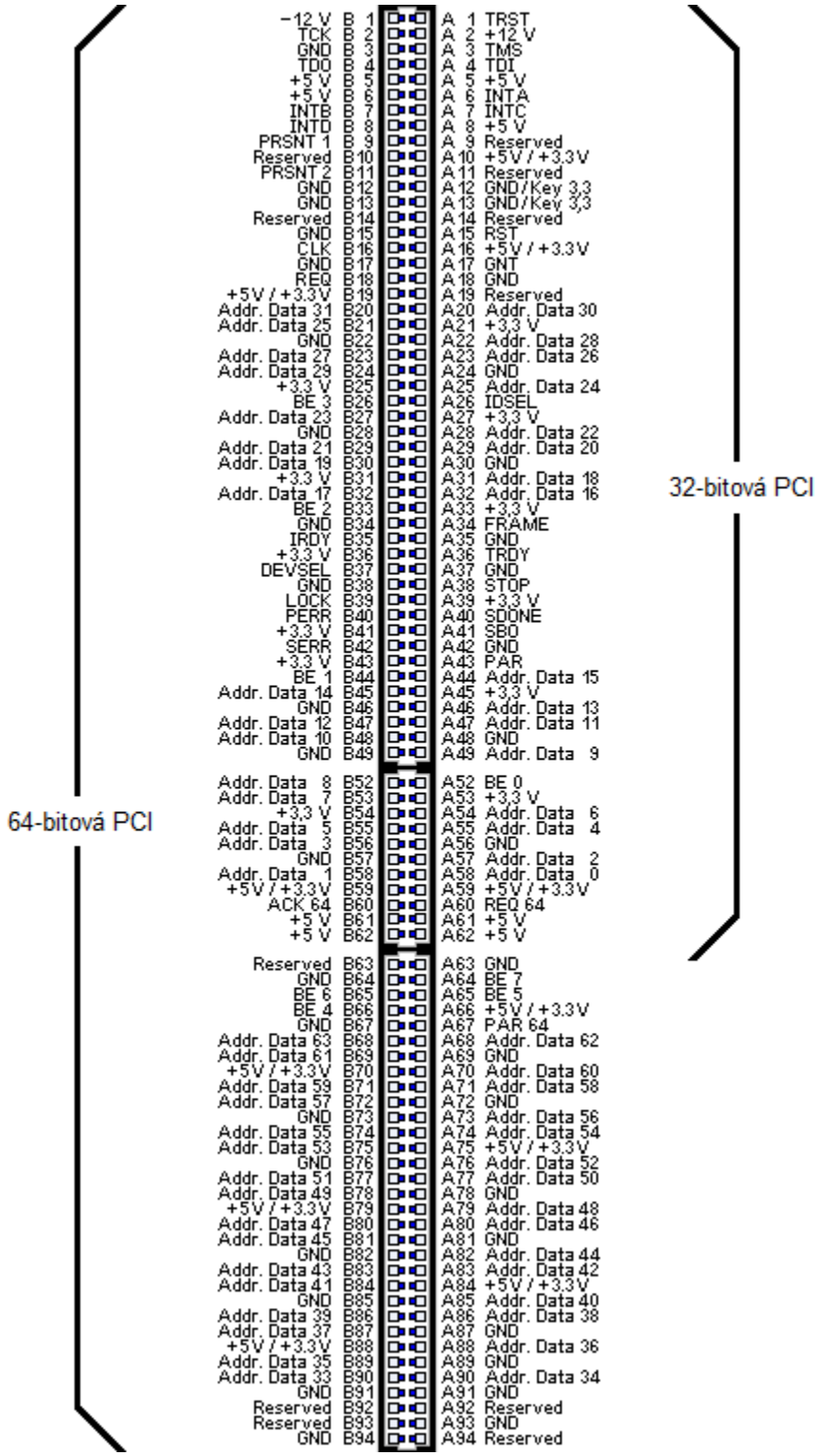
- dvakrát a štyrikrát väčšia prenosová rýchlosť ako PCI-X,
- používa rovnaké konektory, šírku zbernice a protokoly,
- umožňuje 10Gb technológie (Ethernet, Fibre Channel, ...),
- priepustnosť 32x väčšia ako prvá generácia PCI,
- plná hardwarová a softwarová kompatibilita s predošlými generáciami PCI,
- presnú špecifikáciu štandardu nájdete na stránkach PCI SIG.

Na nasledujúcom obrázku Obr. 3.16 sú 3 verzie 32-bitových konektorov zbernice PCI podľa 5V, alebo 3,3 V logiky.



Obr. 3.16 Verzie 32-bitových konektorov zbernice PCI

Na obrázku Obr. 3.17 je znázornený slot zbernice PCI s rozložením jednotlivých pinov (signálov) a ich význam. Následne po obrázku v tabuľke Tab. 3.5 je vysvetlený význam a popis týchto signálov.

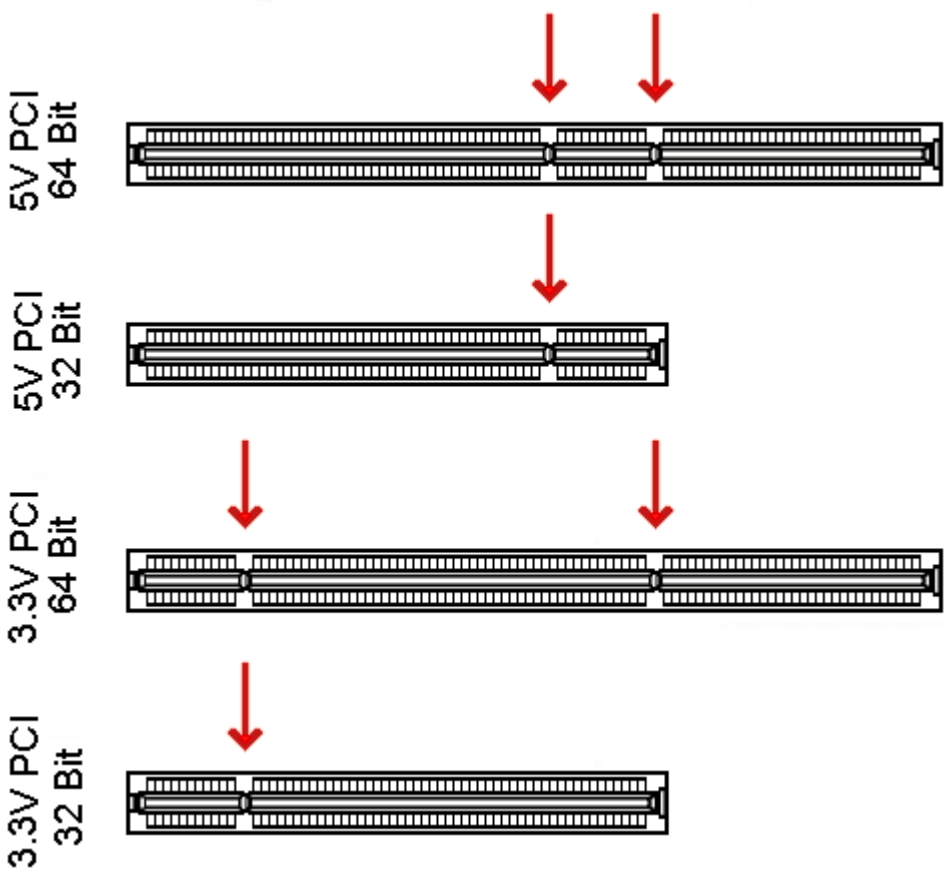


Obr. 3.17 Slot PCI (32-bitový a 64-bitový) s rozložením pinov (signálov)

Tab. 3.5 Význam pinov na zbernici PCI

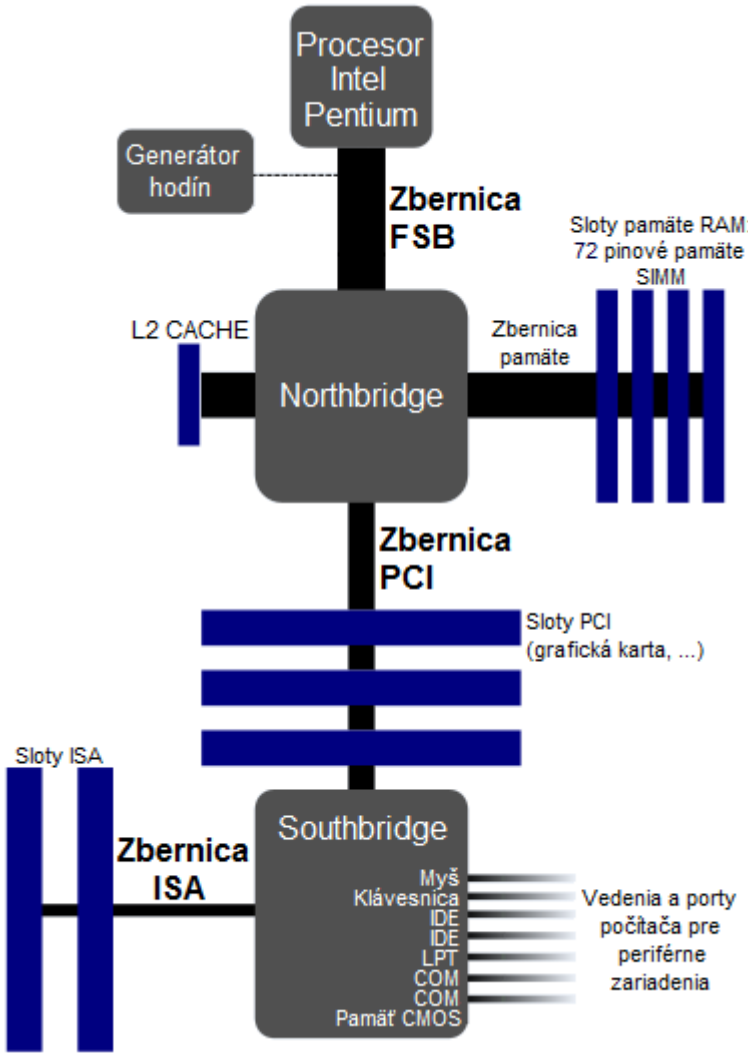
Kontakt (pin)	Signál	Význam
A1	TRST	Inicializuje test rozširujúcej karty.
A2, A5, ..., B79, B88	VCC	Napájanie + 12 V, + 5 V, + 3,3 V, - 12 V.
A3	TMS	Nastavenie režimu testovania rozširujúcej karty.
A4	TDI	Pomocou sériového signálu sa prijímajú testovacie dáta a inštrukcie behom testu.
A6, A7, B7, B8	INT A, B, C, D	Signály prerušení.
A9, A11, ..., B92, B93	Reserved	Rezervovaný signál, nepoužívaný.
A12, A13, A50, A51, B12, B13, B50, B51	Key / GND	Fyzická zarážka na rozlíšenie 5 V a 3,3 V verzie zbernice PCI, alebo uzemnenie.
A15	RST	Vynulovanie nastavení (reset).
A17	GNT	Potvrdzovací signál pre rozširujúcu kartu, ktorá sa stane nasledujúcim mastrom zbernice.
A18, A24,..., B91, B94	GND	Uzemnenie.
A20, A22,..., B89, B90	Addr, Data	Zdieľané adresné a dátové signály.
A26	IDSEL	Aktívna rozširujúca karta.
A34	FRAME	Aktívna karta posiela dáta.
A36	TRDY	Prijímač oznamuje, že je pripravený ukončiť aktuálny prenos dát.
A38	STOP	Prijímač oznamuje iniciátorovi, že má ukončiť prebiehajúcu transakciu.
A40	SDONE	Prehľadávanie pamäte cache skončilo.
A41	SBO	Zápis do pamäte cache neplatí.
A43	PAR	Kontrola parity (1 = párna).
A52, A64,..., B65, B66	BE 0 až 7	Povolenie konkrétneho bajtu na zbernici.
A60	REQ 64	Master indikuje, pripravenosť k 64-bit. prenosu.
A67	PAR 64	Kontrola parity pri 64-bitovom prenose.
B2	TCK	Signál umožňuje testovanie podľa IEEE 1149.1.
B4	TDO	Ako TDI, ale dáta sa vysielajú, nie prijímajú.
B9, B11	PRSNT	Signály indikujú požiadavky na napájanie a prítomnosť rozširujúcej karty.
B16	CLK	Systémové hodiny.
B18	REQ	Master indikuje, že je pripravený k prenosu dát.
B35	IRDY	Dáta sú platné a môžu sa čítať.
B37	DEVSEL	Aktivovanie zariadenia ako prijímača.
B39	LOCK	Zablokovanie rozširujúcej karty.
B40	PERR	Chyba parity.
B42	SERR	Chyba parity, ktorý najčastejšie vyvoláva NMI.
B60	ACK 64	Môže sa uskutočniť 64-bitový prenos.

Podľa obrázku Obr. 3.16 je zrejmé, že existuje 6 typov konektorov rozširujúcich kariet pre zbernicu PCI (3 typy 32-bitových a 3 typy 64-bitových kariet), ale sloty na matičnej doske sa rozlišujú podľa logiky (5 V, alebo 3,3 V) PCI rozširujúcej karty a podľa šírky zbernice, takže existujú len 4 typy slotov zbernice PCI (Obr. 3.18).



Obr. 3.18 Rôzne sloty PCI zbernice

Na obrázku Obr. 3.19 je znázornenie zapojenie komponentov v počítači s procesorom Intel Pentium. Keďže bridge-e mali stále viac a viac funkcií tak sa premenovali na Northbridge a Southbridge. Všetky integrované periférie sa dostali pod záštit Southbridge-u, ktorý bol aj „mostom“ medzi zbernicou PCI a ISA. Úloha Northbridge-u bola sprostredkovať informácie medzi zbernicami FSB a PCI a taktiež komunikácia s operačnou a CACHE pamäťou.



Obr. 3.19 Zapojenie komponentov počítača s procesorom Intel Pentium do jednotlivých zberníc

3.8 AGP

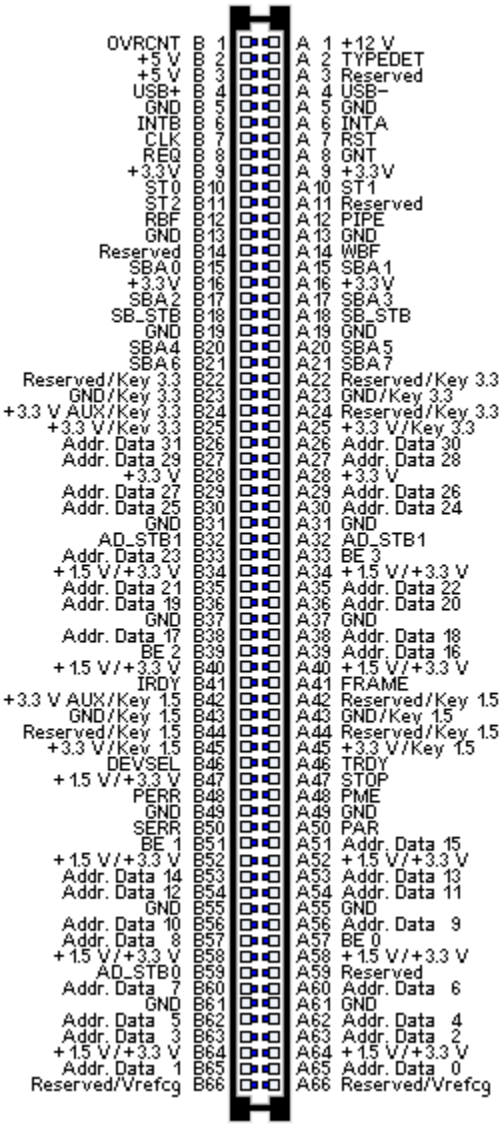
Podobne ako u ISA zbernice v posledných etapách jej používania sa ukázalo, že kriticky úzkym hrdlom je prenos dát smerom ku grafickému adaptéru. Aj pri PCI nastal podobný problém a tak bolo prijaté obdobné riešenie (ako VL BUS) a pre grafický adaptér bola pripravená špeciálna zbernica s názvom Accelerated Graphics Port – AGP, ktorá bola navrhnutá ako vysokorýchlostné rozšírenie PCI o novú vysokorýchlostnú zbernicu, určenú na prenos grafických dát smerom ku grafickému adaptéru.

Fyzický takt zbernice AGP bol 66 MHz, bitová šírka dát je 32 bit, teoretická maximálna prenosová rýchlosť je 264 MB/s. Výhodou je, že grafická karta sa na AGP nemusí deliť o prenos s inými komponentmi, AGP umožňuje grafickému procesoru rýchly prístup do RAM (v prípade ISA, PCI atď. nemohol grafický čip používať priamy prístup do RAM a musel na odkladanie vlastných údajov používať výhradne svoju vlastnú pamäť), takže dokáže za pomoci čipovej sady vyvolávať z RAM uložené textúry ako bitové mapy

tvoriace jediný celok a tým výrazne urýchliť proces renderovania (pokrývania povrchu objektov textúrami). AGP podporuje pipelining, umožňuje taký režim práce, pri ktorom ešte pred skončením údajového procesu – teda v čase hľadania požadovaných dát – je možné vystaviť žiadosť na ďalšie dáta.

Neskoršie štandardy AGP zbernice sú označované násobkom rýchlosti pôvodnej AGP. Napríklad AGP 2x využíva na prenos dát nielen nábežnú, ale aj zostupnú hranu hodinového signálu, takže pri 66 MHz dokáže pracovať na efektívnej frekvencii 133 MHz a preniesť 533 MB/s.

Na obrázku Obr. 3.20 je znázornený slot univerzálnej zbernice AGP s 1,5 V a 3,3 V logikou aj s rozložením jednotlivých pinov (signálov) a ich význam. Následne po obrázku v tabuľke Tab. 3.6 je vysvetlený význam a popis týchto signálov.



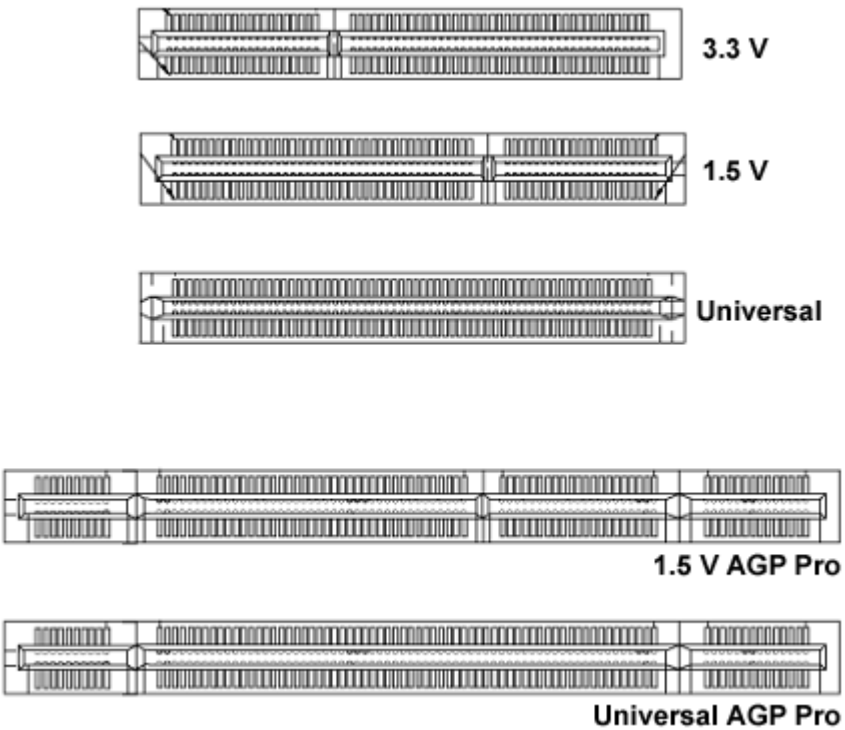
Obr. 3.20 Slot AGP (univerzálny 132 pinový) s rozložením pinov (signálov)

Tab. 3.6 Význam pinov na zbernici AGP

Kontakt (pin)	Signál	Význam
A1, A9, A16, A25, ..., B47, B52, B58, B64	+ 12 V, + 5 V, +3,3 V, +1,5 V	Napájanie + 12 V, + 5 V, + 3,3 V, + 1,5 V.
A2	TYPEDET	Detegovanie typu rozširujúcej karty (3,3 V alebo 1,5 V).
A3, A11, ..., B44, B66	Reserved	Rezervovaný signál, nepoužívaný.
A4, B4	USB-, USB+	Dátové USB signály.
A5, A13, ..., B55, B61	GND	Uzemnenie.
A6, B6	INTA, INTB	Signály prerušení.
A7	RST	Vynulovanie nastavení (reset).
A8	GNT	Potvrdzovací signál pre rozširujúcu kartu, ktorá sa stane nasledujúcim mastrom zbernice.
A10, B10, B11	ST0, ST1, ST2	Signály stavu zbernice.
A12	PIPE	Spustený pipelining.
A14	WBF	Buffer je naplnený, môže sa začať zapisovať.
A15, A17, A20, A21, B15, B17, B20, B21	SBA	Príkazy pre grafický adaptér.
A18, B18	SB_STB	Zbernica SBA je pripravená (STB =STROBE).
A22, A23, A24, A25, ..., B43, B44, B45	Key	Fyzická zarážka na rozlíšenie 3,3 V a 1,5 V verzie, univerzálny slot zarážky neobsahuje.
A26, A27,..., B63, B65	Addr, Data	Zdieľané adresné a dátové signály.
A32, B32, B59	AD_STB	Adresná alebo dátová zbernica je pripravená.
A33, A57, B39, B51	BE	Povolenie konkrétneho bajtu na zbernici.
A41	FRAME	Aktívna karta posiela dáta.
A46	TRDY	Prijímač oznamuje, že je pripravený ukončiť aktuálny prenos dát.
A47	STOP	Prijímač oznamuje iniciátorovi, že má ukončiť prebiehajúcu transakciu.
A48	PME	Signál rozširujúcej karty s nižšou spotrebou.
A50	PAR	Kontrola parity (1 = párna).
A66, B66	Vrefcg	Odlíšenie karty s logikou 0,8 V a 1,5 V.
B1	OVRCNT	Indikovanie USB zariadenia.
B7	CLK	Systémové hodiny.
B8	REQ	Master indikuje, že je pripravený k prenosu dát.
B12	RBF	Buffer je naplnený, môže sa začať čítať.
B41	IRDY	Dáta sú platné a môžu sa čítať.
B46	DEVSEL	Aktivovanie zariadenia ako prijímača.
B48	PERR	Chyba parity.
B50	SERR	Chyba parity, ktorý najčastejšie vyvoláva NMI.

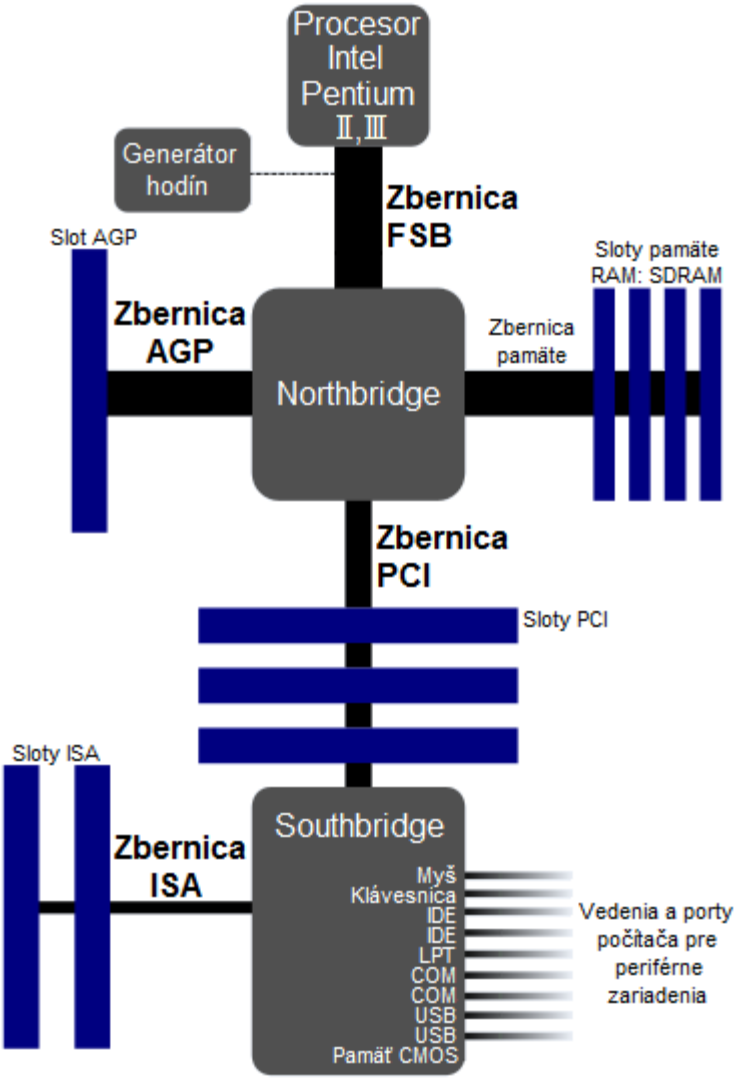
Ďalšie varianty sú označované ako AGP 4x a AGP 8x Zvýšenie efektívneho taktu až na štvornásobok až osemnásobok pôvodného taktu AGP prináša zvýšenie dátového toku až na 1 GB/s, resp. 2 GB/s. Podporované karty pre AGP disponujú samostatne taktovaným čipom GPU (Graphic Processing Unit) na frekvencii 200 až 750 MHz, takt pamäti je potom dosiahnutý násobičom a v súčasnosti sa pohybuje v blízkosti 1 GHz. Šírka dátovej zbernice medzi pamäťou karty a GPU potom nadobúda hodnôt 64, 128 a 256 bitov. Dátový tok potom dosahuje 30 GB/s.

Okrem klasickej AGP karty existovala aj karta AGP Pro, ktorá mala navyše 48 pinov. Prvých 20 pinov bolo napájacích na 3,3 V a uzemnenia, nasledovali kontakty klasického AGP slotu a nakoniec 28 napájacích a uzemňovacích pinov na 12 V.



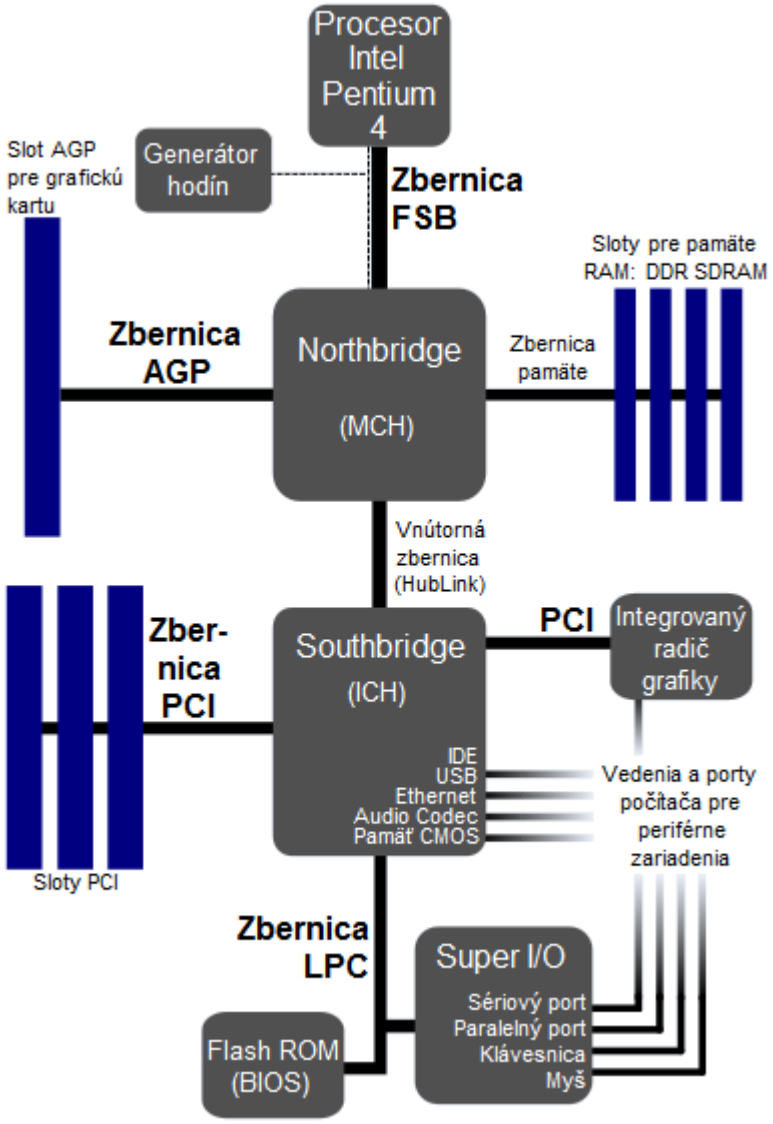
Obr. 3.21 Sloty AGP: AGP 3,3 V, AGP 1,5 V, univerzálny AGP slot, AGP Pro 1,5 V, univerzálny AGP Pro slot (z hora na dol)

S príchodom zbernice AGP pribudla pre Northbridge ďalšia úloha a to sprostredkovanie komunikácie aj so zbernicou AGP. Zapojenie zbernice AGP do počítača môžete vidieť na obrázku Obr. 3.22. Ďalšou zmenou pre Northbridge bolo integrovanie pamäte CACHE L2 do procesora.



Obr. 3.22 Zapojenie komponentov počítača s procesorom Intel Pentium II a III do jednotlivých zberníc

Na obrázku Obr. 3.22 je zjavné, že integrovaných radičov periférií na matičnej doske pribúda a tak pri počítačoch s procesorom Intel Pentium 4 sa niektoré integrované radiče od Southbridge-u oddelili. Boli to radiče pomalších periférií ako sériový port, paralelný port, klávesnica a myš. Tieto radiče sa stali súčasťou Super I/O, ktorá so Southbridge-om komunikovala pomocou zbernice LPC (Low pin count). Aj BIOS sa s pamäti CMOS dostal na pamäť Flash. Tieto skutočnosti sú viditeľné na obrázku Obr. 3.23.



Obr. 3.23 Zapojenie komponentov počítača s procesorom Intel Pentium 4 do jednotlivých zberníc

Na obrázku Obr. 3.21 sú vykreslené typy AGP slotov a na obrázku Obr. 3.24 je grafická karta VGA MSI nVidia FX5500-D256H určená pre zbernicu AGP s univerzálnym konektorom (vložíte ju do akéhokoľvek AGP slotu) a 3,3 V AGP slot.

Obr. 3.24 Grafická karta určená pre zbernicu AGP (hore) a 3,3 V AGP slot (dole)

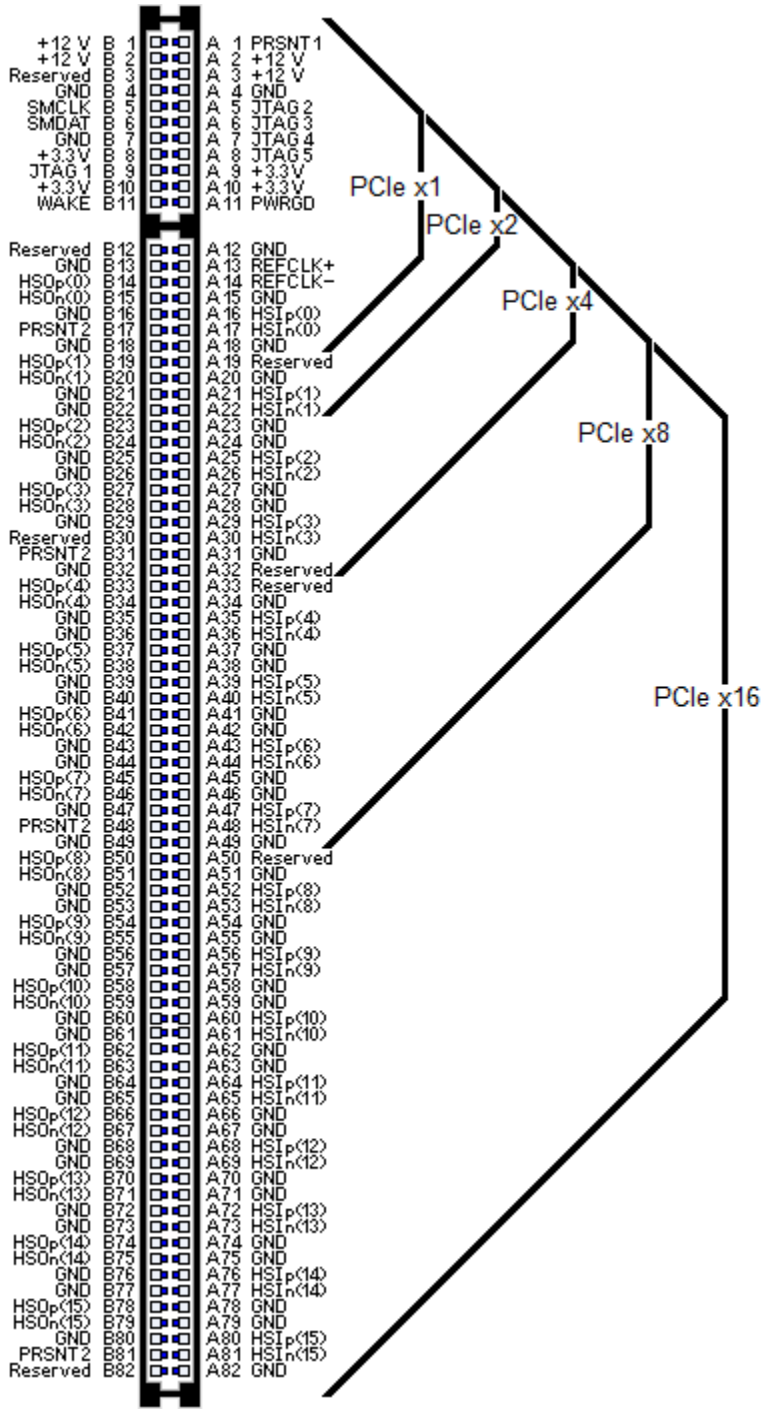
3.9 PCI Express

PCI Express (PCIe) predstavuje zásadný prelom v koncepcii zbernicových systémov PC. Prichádza s riešením sériového prenosu dát a z obvyklých 16, 32 či dokonca 64 dátových vodičov redukuje počet na 2 dátové kanály: jeden je určený pre upstream a jeden pre downstream (diferenčné páry). Táto koncepcia umožňuje využívať obojsmernú komunikáciu v režime full-duplex.

Taktovacia frekvencia PCI Express je 2,5 GHz . Každý kanál disponuje prenosovou rýchlosťou približne 2,5 Gb/s, čo umožňuje dosahovať v režime jednosmerného trvalého prenosu dát tok približne okolo 250 MB/s. Pri duplexnej prevádzke je u základného variantu PCI Express možné dosiahnuť trvalý dátový tok 2x250 MB/s (up/down). Dátový prenos je riešený prostredníctvom paketov, obdobne ako je tomu napríklad pri Ethernete.

Ďalší významný prelom prichádza v koncepcii zbernice ako zdieľaného média. Na rozdiel od paralelného PCI a predchádzajúcich typov zberníc, kde bola zbernica zdieľaná všetkými zariadeniami na ňu pripojenú, používa PCI Express technológiu dvojbodového spojenia medzi príslušným slotom a chipsetom, takže príslušnú prenosovú kapacitu má každé zariadenie výhradne pre seba. Rýchle prepínanie medzi chipsetom a príslušným zariadením zabezpečuje integrovaný switch. Odpadá potreba rozhodovacích algoritmov a rozhodovacích obvodov, ktoré prideľujú zbernicu v určitom momente ku konkrétnemu zariadeniu. Umožňuje implementovať algoritmus QoS na úrovni systémovej zbernice.

Na obrázku Obr. 3.25 je znázornený slot PCIe x16 s rozložením jednotlivých pinov (signálov), ale na obrázku je porozdeľovaný na všetky možné varianty zbernice PCIe a dá sa z obrázka vidieť aj vzhľad menších slotov ako PCIe x1, PCIe x2 (nevyrábaný slot k roku 2014), PCIe x4, PCIe x8. Následne po obrázku v tabuľke Tab. 3.7 je vysvetlený význam a popis týchto signálov.



Obr. 3.25 Sloty PCIe (PCIe x1 až PCIe x16) s rozložením pinov (signálov)

Tab. 3.7 Význam pinov na zbernici PCIe

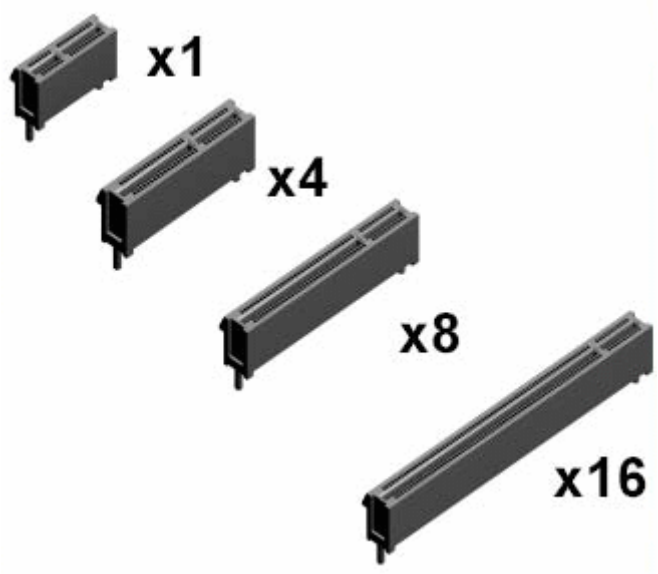
Kontakt (pin)	Signál	Význam
A1 B17, B31, B48, B81	PRSNT 1, PRSNT 2	Signál PRSNT 1 je na karte priamo prepojený s najvzdialenejším PRSNT 2, vďaka čomu matičná doska zistí prítomnosť a šírku karty.
A2, A3, B1, B2	+ 12 V	Napájanie + 12 V.
A4, B4, B7, A12, A15, A18, B13, B16, B18, A20, B21, B22, A23, A24, A27, A28, A31, B25, B26, B29, B32, A34, A37, A38, ..., B43, B44, B47, B49, A51, A54, A55, ..., B73, B76, B77, B80	GND	Uzemnenie.
B9	JTAG 1	TRST – Inicializuje test rozširujúcej karty.
A5	JTAG 2	TCK – Signál umožňuje testovanie časovania rozširujúcej karty podľa normy IEEE 1149.1.
A6	JTAG 3	TDI – Pomocou sériového signálu sa prijímajú testovacie dáta a inštrukcie behom testu.
A7	JTAG 4	TDO – Pomocou sériového signálu sa vysielajú testovacie dáta a inštrukcie behom testu.
A8	JTAG 5	TMS – Nastavenie režimu testovania rozširujúcej karty.
A9, A10, B8, B10	+ 3,3 V	Napájanie + 3,3 V.
A11	PWRGD	Signál správneho napájania.
B3, B12, A19, A32, B30, A33, A50, B82	Reserved	Rezervovaný signál, nepoužívaný.
B5	SMCLK	Zbernica SMBus – hodiny zbernice SMBus.
B6	SMDAT	Zbernica SMBus – dáta zbernice SMBus.
B11	WAKE	Signál určený na reaktiváciu rozširujúcej karty.
A13, A14	REFCLK +, –	Referenčné hodiny sériovej zbernice (dif. pár).
A16, A17, A21, A22, A25, A26, A29, A30, A35, A36, A39, A40, A43, A44, A47, A48, A52, A53, A56, ..., A76, A77, A80, A81	HSIp (číslo cesty), HSIn (číslo cesta)	Cesta (lane) sériového prijímača (diferenčný pár).
B14, B15, B19, B20, B23, B24, B27, B28, B33, B34, B37, B38, B41, B42, B45, B46, B50, B51, B54, ..., B74, B75, B78, B79	HSOn (číslo cesty), HSOn (číslo cesty)	Cesta (lane) sériového vysielača (diferenčný pár).

Každé zariadenie tak má vyhradenú vlastnú zbernicu, označovanú ako link. Každý link je tvorený jednou alebo viacerými cestami. Každá cesta (lane) umožňuje v jedinom okamihu sériovo prenášať dáta v režime full-duplex, pri prenose v základnom pásme dosahuje priepustnosť 2,5 Gb/s (PCIe 1.0). Tým dosiahla PCI Express schopnosť vysokej modularity, škálovateľnosti a budúceho vývoja. V praxi to znamená, že na komunikáciu jedného slotu je možné použiť viac ako jedinú cestu (lane) a agregovaním viacerých prenosových kanálov príslušne zvýšiť prenosovú rýchlosť. Jednoduchým použitím viacerých ciest pre link môžeme dosiahnuť 2x link, 4x link až 32x link. Na jedinej doske je možné kombinovať viaceré varianty PCI Express, podľa potrieb zariadení určených do daného slotu. Napríklad jednokanálové varianty PCI Express sú vhodné pre zvukové karty či pomalšie sieťové karty, kým viackanálové varianty sú určené pre gigabitové či 10 Gb/s sieťové karty, výkonné rozširujúce karty a pre grafické adaptéry. Konfigurácia prideľovaných ciest každému zariadeniu prebieha vždy pri štarte počítača. Obmedzenie počtu vodivých ciest prináša významné úspory pri stavbe základných dosiek, zjednodušuje ich návrh a umožňuje zmenšiť rozmery. Cenou za takto radikálnu zmenu technológie však je nekompatibilita so zariadeniami určenými do klasického PCI slotu.

Podľa predchádzajúceho obrázka (Obr. 3.25) a tabuľky (Tab. 3.7) môžeme usúdiť, že PCIe x1 (s 1 cestou) má 36 pinov (kontaktov), PCIe x2 (s 2 cestami) by mohla mať 44 pinov (zatiaľ sa nepoužíva a nemá signál PRSNT 2, ale na pozícií A19 je rezervovaná pozícia, ktorá by sa na to dala použiť), PCIe x4 (so 4 cestami) má 64 pinov (kontaktov), PCIe x8 (s 8 cestami) má 98 pinov a PCIe x16 má 164 pinov.

PCI Express sa reálne na trh dostala v priebehu roka 2005 a začala systematicky vytláčať klasické PCI riešenie.

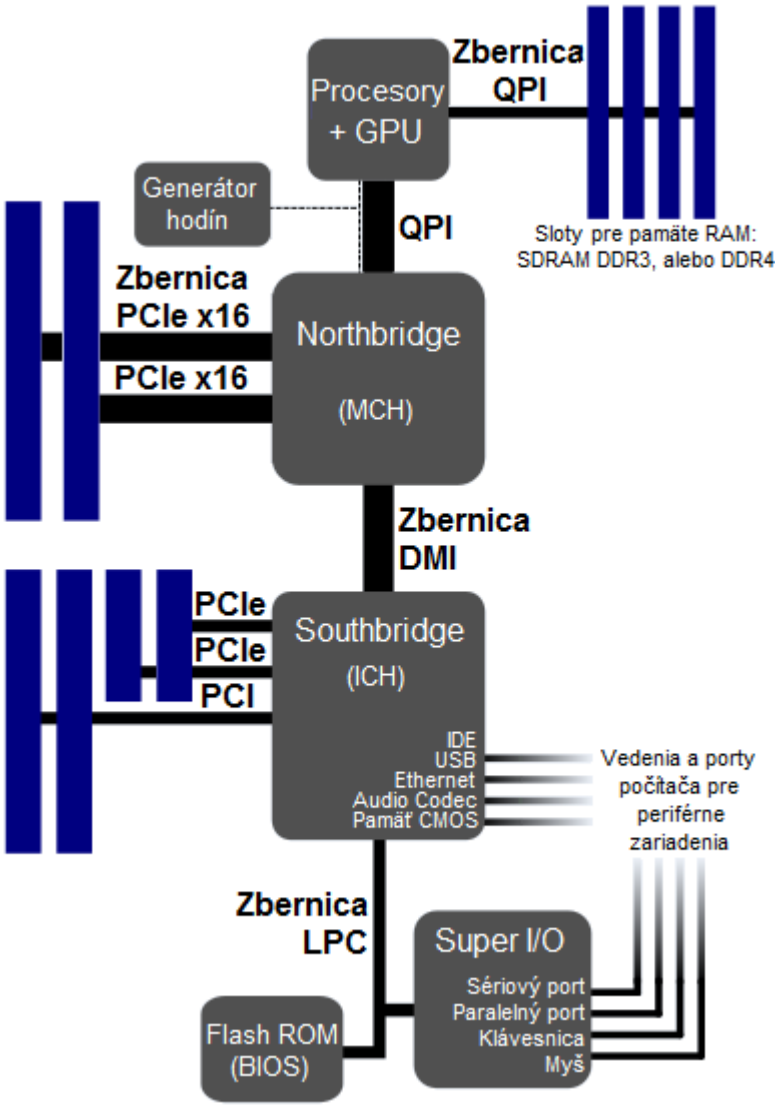
Na obrázku Obr. 3.26 sú všetky typy slotov zbernice PCIe a na obrázku Obr. 3.27 sú konektory PCIe x1 a PCIe x2 so slotom PCIe x16, tento obrázok má demonštrovať to, že aj do konektora PCIe x16 vieme vložiť kartu PCIe x1, ale do konektora PCIe x1 nevložíme kartu PCIe x16.



Obr. 3.26 Sloty PCIe (PCIe x1 až PCIe x16)

Obr. 3.27 Rozširujúce karty s konektormi PCIe x1, PCIe x16 a slot PCIe x16 (zhora nadol)

Obrázok Obr. 3.28 poukazuje na pripojenie zbernice PCIe do dnešných počítačov, tieto zbernice nahradili AGP a pomaly nahradzujú aj zbernicu PCI. Taktiež zbernica FSB bola už pomalá pre priamu komunikáciu s procesorom a tak bola nahradená zbernicou QPI (Quick path interconnect), pre zrýchlenie komunikácie medzi procesorom a operačnou pamäťou sú pamäte RAM priamo pripojené k procesoru. Komunikáciu medzi Northbridge a Southbridge sprostredkúva zbernica DMI (Direct media interface).

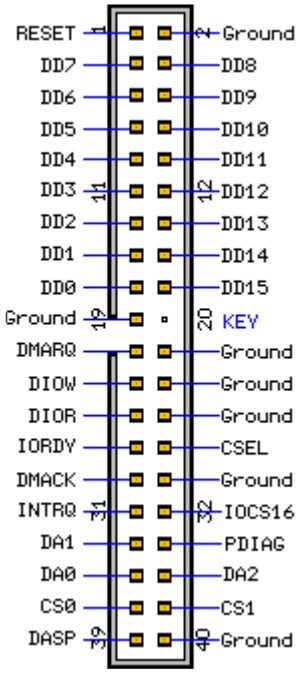


Obr. 3.28 Zapojenie komponentov počítača do zberníc v najmodernejších počítačoch (k roku 2014)

3.10 IDE

Skratka IDE znamená Integrated Drive Electronics (Elektronika integrovaných jednotiek). Táto zbernica slúži na pripojenie pevných diskov (HDD, SSD) k počítaču a tiež pre pripojenie optických mechaník. U starších počítačov sa pomocou tejto zbernice pripájali aj floppy mechaniky.

Štandardne rozhranie tejto zbernice má názov ATA (Advanced Technology Attachment) po vzniku štandardného rozhrania SATA (serial ATA) sa pôvodná ATA premenovala na PATA (parallel ATA).



Obr. 3.29 IDE konektor (PATA konektor)

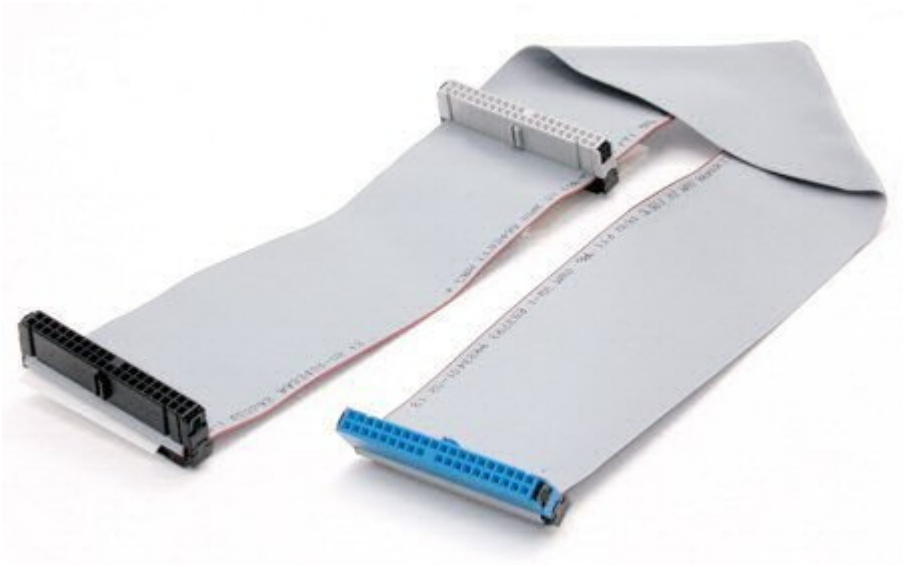
Tab. 3.8 Význam pinov na zbernici IDE

Kontakt (pin)	**Signál**	**Význam**
1	RESET	Vynulovanie nastavení jednotky (reset).
2, 19, 22, 24, 26, 30	Ground	Uzemnenie.
3 až 18	DD	Dátové signály.
20	KEY	Fyzická zarážka.
21	DMARQ	Požiadavka o priamy prístup do pamäte
23	DIOW	Impulz (strobe) pre zápis dát.
25	DIOR	Impulz (strobe) pre čítanie dát.
27	IORDY	Vstupno-výstupné zariadenie je pripravené.
28	CSEL	Komunikácia s master-om (1) alebo slave-om.
29	DMACK	Potvrdenie priameho prístupu do pamäte.
31	INTRQ	Požiadavka prerušenia.
32	IOCS16	Rozlíšenie medzi 8 a 16-bitovým dátovým prenosom.
33, 35, 36	DA	Adresné signály.
34	PDIAG	Diagnostika 80 vodičového ATA kábla.
37, 38	CS 0, CS 1	Výber riadiaceho bloku registrov (bázovej adresy).
39	DASP	Aktívne zariadenie.

IDE radič bol vyvinutý k odstráneniu výpisov pevnej štruktúry disku pri štarte počítača. Pred tým to fungovalo tak, že BIOS pri štarte počítača čítal informácie s CMOS, aby mohol priamo smerovať údaje medzi CPU a HDD. Po nastavení patričných hodnôt radič disku komunikoval s diskom, aby sa odstránil problém bol vyvinutý radič IDE.

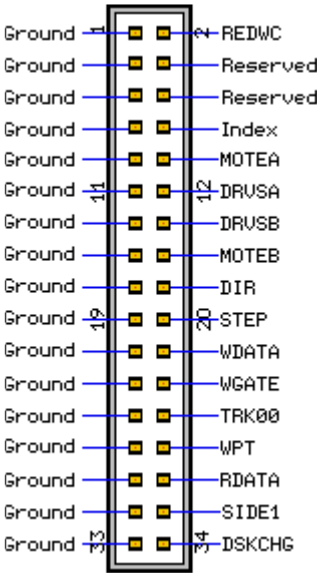
Po implementácii IDE to už funguje tak, že operačný systém vydá jednému zo zariadení všeobecný príkaz tak, ako keby CPU daný HDD fyzicky a geometricky poznal. Signáli od CPU sú preposlané ďalej ku konektorom IDE a riadiace signály informujú jednotku typu master a slave pre koho sú tie údaje určené. Zvolená jednotka preloží príkaz pre jednotku CMOS a to tak, že sa bude pracovať so skutočnou geometriou disku. Geometriou sa myslí to, že jednotka pozná polohu dát na disku a tak riadi polohovanie hlavy disku, čo sú funkcie ktoré zabezpečuje radič disku.

Najskôr radič IDE bol na samostatnej vstupno-výstupnej karte pre zbernice PC-BUS a ISA, neskôr sa radič implementoval priamo na matičnú dosku (na Southbridge). Pôvodný ATA (PATA) konektor mal 39 pinov a kľúč (fyzickú zarážku). Na obrázku Obr. 3.29 je IDE konektor so znázornenými signálmi, tento konektor sa používal pre pevné disky a optické mechaniky. V tabuľke Tab. 3.8 je popísaný význam signálov IDE zbernice. K jednému konektoru IDE bolo možné pripojiť 2 zariadenia pomocou IDE kábla (Obr. 3.30), modrý konektor sa vkladal do IDE slotu na matičnej doske (poprípade IDE slotu na vstupno-výstupnej karte IDE), sivý do konektora na slave zariadení, a čierny do master zariadenia.



Obr. 3.30 IDE kábel (PATA kábel)

Okrem 40 vodičovej verzie PATA existovala aj 80 vodičová, mala taktiež konektor so 40 pinmi. Každý pin mal pár vodičov, jeden vodič niesol samotný signál (rovnaký ako pri 40 vodičovom PATA) a druhý vodič bol uzemňovací. Disketové mechaniky mali vlastný IDE konektor s 34 pinmi, konektor so signálmi je na obrázku Obr. 3.31 a popis signálov je v tabuľke Tab. 3.9.



Obr. 3.31 IDE konektor pre disketové mechaniky

Tab. 3.9 Význam pinov na zbernici IDE pre disketové mechaniky

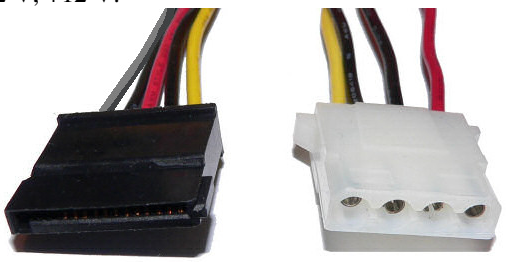
Kontakt (pin)	**Signál**	**Význam**
1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33	Ground	Uzemnenie.
2	REDWC	Hustota záznamu na diskete (0 = veľká).
4, 6	Reserved	Nepoužitý signál (rezervovaný).
8	Index	Index.
10	MOTEA	Zopnutie motora na prvom zariadení.
12	DRVSA	Vybratie prvého zariadenia.
14	DRVSB	Vybratie druhého zariadenia.
16	MOTEB	Zopnutie motora na druhom zariadení.
18	DIR	Smer.
20	STEP	Krok.
22	WDATA	Dáta pre zápis.
24	WGATE	Prebieha zápis na disketu.
26	TRK00	Stopa 00.
28	WPT	Zápis je na diskete zamknutý.
30	RDATA	Prečítané dáta.
32	SIDE1	Strana diskety s ktorej sa číta alebo zapisuje.
34	DSKCHG	Výmena disku.

ATA má viacero verzií:

- pre-ATA (IDE, ATA), rok 1986:
  - radič diskovej jednotky je umiestnený na disku,
  - má jednoduché programové rozhranie umožňujúce komunikovať diskovej jednotke s matičnou doskou v 512 B blokoch,
  - maximálna veľkosť disku 2,1 GB (2 GiB),
  - maximálna prenosová rýchlosť 3,3 MB/s (využitá technológia PIO 0).
- ATA-1 (IDE, ATA), rok 1994:
  - maximálna veľkosť disku 137 GB (128 GiB),
  - maximálna prenosová rýchlosť 8,3 MB/s (využitá technológia PIO 1 a 2).
- ATA-2 (EIDE), rok 1996:
  - skratka EIDE znamená Enhanced IDE (Rozšírené IDE),
  - maximálna veľkosť disku 137 GB (128 GiB),
  - maximálna prenosová rýchlosť 16,6 MB/s (využitá technológia PIO 3 a 4).
- ATA-3, rok 1997:
  - tie isté parametre ako ATA-2, ale umožňovali už aj funkcie SMART (Self-Monitoring, Analysis and Reporting Technology – Technológia sebakontroly, analýzy a hlásenia) tieto funkcie monitorovali pevný disk, aby detekovali a hlásili rôzne indikátory spoľahlivosti, aby mohli predvídať poruchy.
- ATA-4 (Ultra ATA/33), rok 1998:
  - zaviedla **ATAPI** (ATA Packet Interface) čo jej umožnilo pripájať aj ostatné pamäťové zariadenia (týkalo sa to najmä optických mechaník),
  - maximálna prenosová rýchlosť sa zvýšila na 33.3 MB/s (využitá technológia Ultra DMA 0 až 2).
- ATA-5 (Ultra ATA/66), rok 2000:
  - zaviedla 80 žilový kábel,
  - špecifikácia CompactFlash konektora (pred tým mal konektor ATA 44 pinov na napájanie pevného disku, navyše boli uzemnenie, 2x + 5 V a typ),
  - zvýšila sa prenosová rýchlosť na 66 MB/s (využitá technológia Ultra ATA 3 a 4).
- ATA-6 (Ultra ATA/100), rok 2002:
  - zvýšila možnú kapacitu disku na 144PB,
  - maximálnu prenosovú rýchlosť na 100 MB/s (využitá technológia Ultra ATA 5).
- SATA 1 (ATA-7 SATA), 2003:
  - zavedený sériový prenos dát (po 1 b) a obojsmerná komunikácia,
  - maximálna prenosová rýchlosť 150 MB/s (technológia Ultra ATA 6),
- SATA 2, rok 2005:
  - maximálna prenosová rýchlosť 300 MB/s.
- SATA 3, rok 2009:
  - maximálna prenosová rýchlosť 600 MB/s.

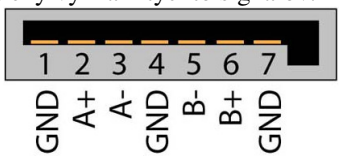
Od štandardu ATA-5 majú IDE zbernice oddelené vedenia pre napájanie a prenos dát, vzhľad napájacích konektorov je na obrázku Obr. 3.32 pričom žltý vodič je napájanie +12 V, čierne vodiče sú uzemnenia, červený je napájanie +5 V a sivý je napájanie +3,3 V, tento sivý vodič je iba u napájacieho konektora SATA. Na obrázku je ešte viditeľné, že napájací kábel pre SATA má viac ako 5 kontaktov má ich až 15 sú nimi: +3,3 V, + 3,3 V,

+3,3 V, uzemnenie, uzemnenie, uzemnenie,+5 V, +5 V, +5 V, uzemnenie, uzemnenie, uzemnenie, +12 V, +12 V, +12 V.



Obr. 3.32 Napájacie konektory SATA (v ľavo) a PATA (v pravo)

SATA konektor určený pre prenos dát a jeho signály sú na obrázku Obr. 3.33 a v tabuľke Tab. 3.10 je vysvetlený význam týchto signálov.

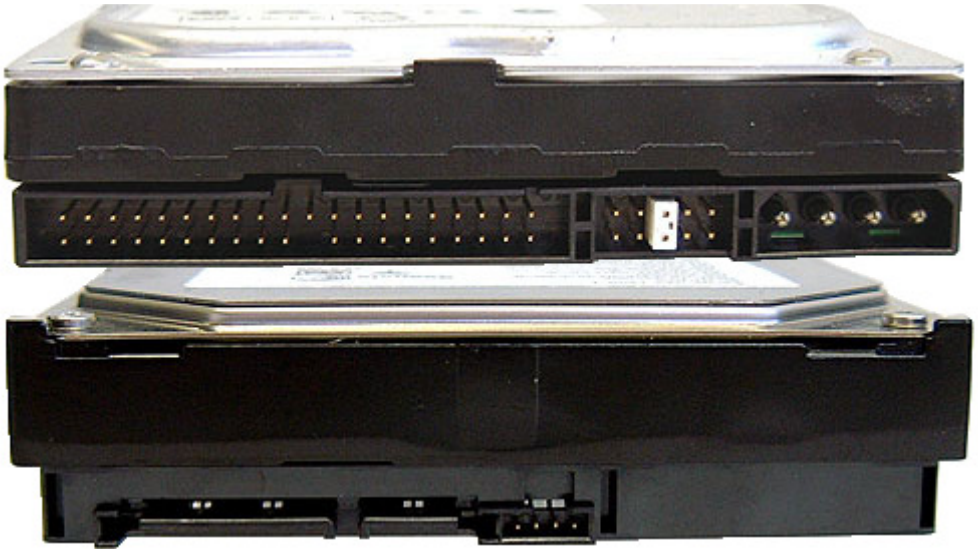


Obr. 3.33 IDE konektor (SATA konektor)

Tab. 3.10 Význam pinov na zbernici IDE (SATA konektor)

Kontakt (pin)	**Signál**	**Význam**
1, 4, 7	GND	Uzemnenie.
2, 3	A+, A–	Diferenčný pár vysielaných signálov.
5, 6	B–, B+	Diferenčný pár prijímaných signálov.

Na obrázku Obr. 3.34 sú porovnané pevné disky s technológiou PATA a s technológiou SATA.



Obr. 3.34 Pevné disky (hore s PATA konektormi, dole so SATA konektormi)

Dôležitým rozhraním pri SATA technológií je *Advanced Host Controller Interface* (AHCI) je to hardvérový mechanizmus, ktorý umožňuje softvéru komunikovať so zariadeniami SATA, ktoré sú navrhnuté tak, aby umožňovali vlastnosti, ktoré nie sú dostupné v PATA. Okrem vyššej rýchlosti je to možnosť odpájania zariadení za chodu a „native command queuing“ (prirodzené dopytovanie príkazov). Väčšina SATA radičov umožňuje zapnúť AHCI samostatne alebo v kombinácii s podporou RAID.

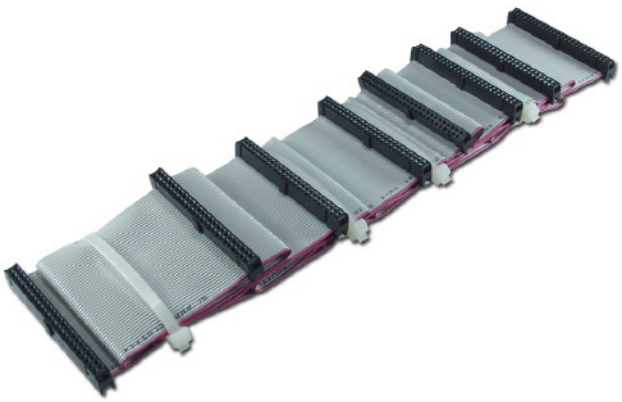
3.11 SCSI

Zbernica SCSI (Small computer system interface) je štandardné rozhranie a sada príkazov pre výmenu dát medzi externými alebo internými počítačovými zariadeniami a počítačovou zbernicou.

SCSI sa používa pre pripojenie pevných diskov. Pomocou SCSI sa dajú pripojiť aj iné zariadenia ako napr. skenery, jednotky CD-ROM alebo DVD. Osobné počítače alebo notebooky používajú SCSI iba výnimočne (dlhú dobu SCSI používala spoločnosť Apple), pre pevné disky sa používajú najmä SATA (pred tým PATA) a pre iné zariadenia je rozšírené USB.

Výhodou SCSI bola možnosť pripojenia väčšieho počtu pevných diskov (alebo iných periférií) ako pri rozhraní ATA/IDE (PATA), zbernica SCSI mala spravidla aj väčšiu prenosovú rýchlosť a reálny výkon aj vďaka protokolu prenosu. SCSI disky mali a majú spravidla väčšie otáčky, kratšiu prístupovú rýchlosť a vďaka zameraniu aj dlhšiu životnosť. Dodnes servery ktoré potrebujú vyššiu prenosovú rýchlosť, pripájať väčšie množstvo pevných diskov s vyššou spoľahlivosťou, alebo potrebujú silnejšiu hardvérovú redundanciu pevných diskov používajú disky ktoré sa pripájajú pomocou zbernice SCSI.

Tak ako pri IDE sa z paralelného rozhrania stalo časom sériové, tak sa to stalo aj pri SCSI. Paralelne rozhranie SCSI malo 25, 50, 68 a napokon pred príchodom sériového rozhrania až 80 kontaktov (pinov). Jeden radič SCSI umožňoval pripojiť 7 a neskôr 15 zariadení. Zariadenia sa pripájali do série, každé zariadenie v reťazci malo svoje identifikačné číslo od 0 do 7 (alebo 15) s tým, že číslo 7 (15) mal samotný radič SCSI, adresy 0 až 6 (14) boli pridelené perifériám. Týchto 7 zariadení sa pripájalo pomocou SCSI kábla, ktorý je na obrázku Obr. 3.35.



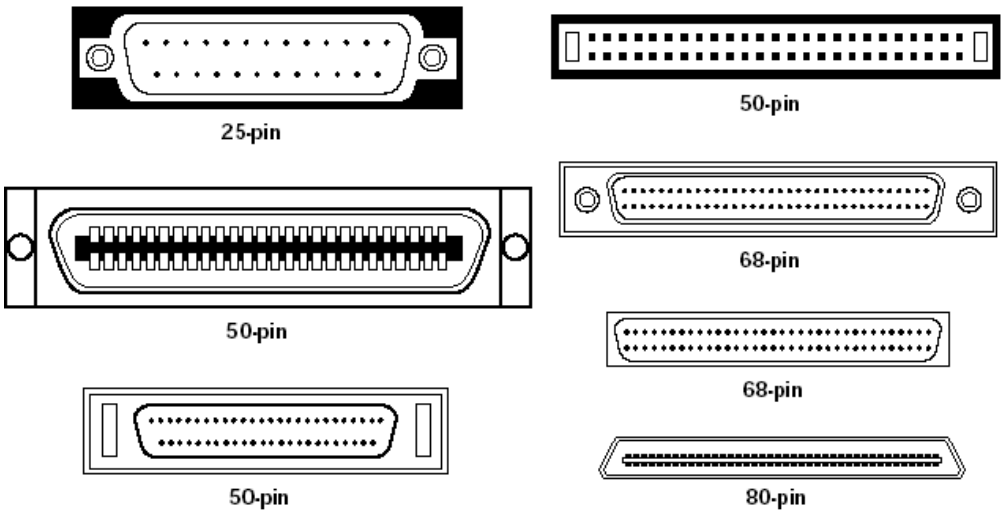
Obr. 3.35 SCSI kábel pre pripojenie 7 zariadení

Pripojenie periférií k radiču SCSI pomocou kábla z obrázku Obr. 3.35 je na obrázku Obr. 3.36.

Obr. 3.36 Pripojenie periférií k radiču SCSI

Pri rozhraní SCSI všetky zariadenia zdieľajú všetky dátové a všetky riadiace signály, pričom v jednom čase komunikujú 2 zariadenia. Keď konkrétna aplikácia prikáže komunikáciu s HDD radič sa stane master-om a pevný disk slave-om.

Pre veľké množstvo konektorov (Obr. 3.37) sa v tejto podkapitole nebudú rozoberať podrobne signály, ale iba pri jednotlivých verziách sa uvedie počet dátových a počet riadiacich signálov.



Obr. 3.37 Konektory paralelného rozhrania SCSI

Verzie paralelnej zbernice SCSI:

- SCSI-1, rok 1986:
  - 50 kontaktov (9 dátových, 11 riadiacich signálov, 26 tienení/uzemnení a 4 kontakty nepoužité),
  - existovala aj zúžená variant s 25 kontaktmi, kde signály boli tie isté, len tieniacich vodičov bolo menej a všetky kontakty boli použité,
  - šírka pásma: 8 bitová komunikácia,
  - počet zariadení na jeden radič: 7 + radič,
  - prenosová rýchlosť 5 MB/s,

- SCSI-2 (Fast SCSI (8 bit), Fast-Wide SCSI (16-bit)), rok 1994:
  - konektory boli rovnaké ako u SCSI-1, ale k jednému zariadeniu sa mohli pripojiť dve konektory (ak zariadenia mali takú možnosť),
  - šírka pásma: 8 bitová (s 1 konektorom), 16 bitová (s 2 konektormi),
  - počet zariadení na jeden radič: 7 + radič, alebo 15 + radič
  - prenosová rýchlosť 10 MB/s, alebo 20 MB/s (podľa šírky pásma),
- SCSI-3 SPI (Ultra SCSI, Ultra-Wide SCSI), rok 1996:
  - Ultra SCSI mal 50 pinov a Ultra-Wide SCSI mal 68 pinov (36 dátových pinov, 23 riadiacich pinov, 2 nepoužité piny, 7 uzemnení/tienení – všetky dátové a riadiace signály sú diferenčne kontrolované, takže reálne 2 signáli udávajú 1 údaj),
  - šírka pásma: 8 bitová (50-pinový konektor), 16 bitová (68-pinový konektor),
  - počet zariadení na jeden radič: 7, alebo 15,
  - prenosová rýchlosť: 20 MB/s, alebo 40 MB/s,
- SCSI-3 SPI-2 (Ultra2 SCSI (8 bit), Ultra2 Wide SCSI (16 bit)), rok 1997:
  - Ultra2 SCSI mal 50 pinov a Ultra2 Wide SCSI mal 68 a 80 pinov (36 dátových pinov, 29 riadiacich a oznamovacích pinov, 15 napájacích pinov s uzemnením),
  - šírka pásma: 8 a 16 bitov,
  - počet zariadení na jeden radič: 7, alebo 15,
  - prenosová rýchlosť: 40 MB/s, alebo 80 MB/s,
- SCSI-3 SPI-3 (Ultra3 SCSI), rok 1999:
  - počet pinov: 68, alebo 80,
  - šírka pásma: 16 bitov,
  - počet zariadení na jeden radič: 15,
  - prenosová rýchlosť: 160 MB/s,
- SCSI-3 SPI-4 (Ultra-320 SCSI), rok 2002:
  - počet pinov: 68 a 80, šírka pásma: 16 bitov,
  - počet zariadení na jeden radič: 15,
  - prenosová rýchlosť: 320 MB/s,
- SCSI-3 SPI-5 (Ultra-640 SCSI), rok: 2003:
  - počet pinov: 68 a 80, šírka pásma: 16 bitov,
  - počet zariadení na jeden radič: 15,
  - prenosová rýchlosť: 640 MB/s,

Niektoré sériové zbernice SCSI:

- *SSA* (Serial Storage Architecture): rok 1990, 40 MB/s, 95 zariadení,
- *FC-AL 1Gb/s* (Fiber Channel 1 Gbps): rok 1994, 100MB/s, 127 zariadení,
- FC-AL 2Gb/s: rok 1996, 200MB/s, 127 zariadení,
- FC-AL 4Gb/s: rok 1996, 400MB/s, 127 zariadení,
- SAS (Serial Attached SCSI) 1.1: rok 2006, 300 MB/s, 16255 zariadení,
- SAS 1.2: rok 2011, 600 MB/s, 16255 zariadení,
- SAS 3.0: rok 2013, 1200 MB/s, 16255 zariadení.

4 Architektúra sieti

Táto kapitola sa venuje architektúre sieti. Kapitola je rozdelená do štyroch hlavných podkapitol v ktorých sa venuje distribúcií sieti, prepojovacím štruktúram, ISO/OSI modelu a riadeniu prístupu k médiám.

4.1 Distribuované systémy riadenia

Distribuovaný výpočet, pri ktorom sa na cieľovom riešení zadanej úlohy podieľa viacero procesorov schopných prevádzať výpočet na rozdelených dátach súbežne, sa stáva veľmi častým riešením pre rad aplikácií. K rozloženiu výpočtu na viac procesorov môže viesť rad dôvodov.

Historicky najstarším dôvodom pre distribúciu výpočtu je prirodzená distribúcia dát a ich spracovanie. Hovoríme o geografickej distribúcií, ktorá je bežná v bankových systémoch, výpočtových systémoch veľkých firiem a štátnej správy, v rezervačných systémoch dopravných a hotelových spoločností, atď..

V rade systémov, ktoré sú navrhované pre reálne aplikácie (technologické riadiace systémy, bankové systémy), sa stretávame so situáciou, kedy je nutné jednotlivé procesory distribuovaného systému špeciálne vybaviť, aby boli schopné interakcie s reálnym prostredím. V takýchto systémoch jednotlivé procesory neplnia jednotnú výpočtovú funkciu, nie sú univerzálne vzájomne nahraditeľné. Hovoríme o distribúcii funkcií, špecifickej funkcii vyhradzujeme špecifický výpočet realizovaný na špecifickom procesore.

Ďalším dôvodom pre rozloženie výpočtu na viac procesorov môže byť jeho zložitosť a potreba zdieľať výpočtovú kapacitu viac procesorov. Hovoríme o *distribúcii výkonu* a opierame sa buď o systém navrhnutý špeciálne pre daný výpočet, alebo o univerzálny systém s dynamickým prideľovaním procesorov.

Konečne ako posledné si uvedieme požiadavky na spoľahlivosť, ktorú možno často splniť iba zálohovaním, schopnosťou systému *rekonfigurovať* svoju štruktúru a prideliť výpočtu, ktorému napríklad zlyhal procesor, procesor iný.

4.1.1 Základné pojmy

Ako distribuovaný systém budeme označovať taký výpočtový systém, ktorý zahŕňa *viac než jeden procesor* a má svoj *program rozdelený na časti*, ktoré si vzájomne predávajú dáta, a ktoré sú počítané na rôznych spolupracujúcich procesoroch systému.

Distribuovaná architektúra

Pod pojmom *distribuovaná architektúra* budeme rozumieť skupinu procesorov, ktorých technické vybavenie im dovolí vzájomnú spoluprácu pri výpočte distribuovaného programu. Kľúčovým prvkom distribuovanej architektúry je spôsob, akým si procesory distribuovaného systému vymieňajú dáta.

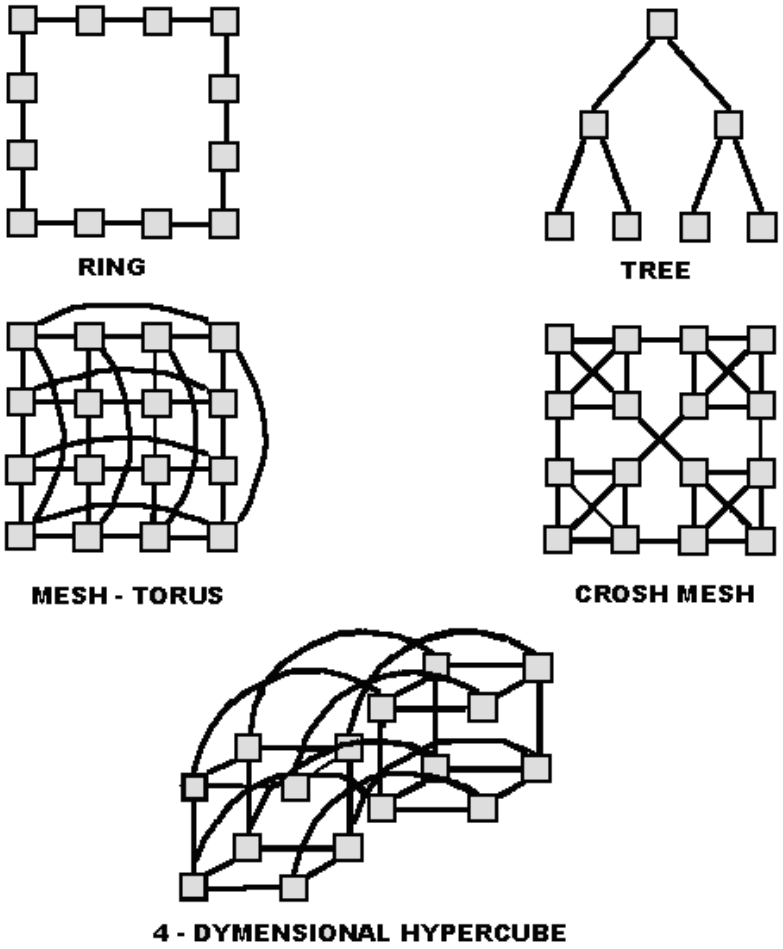
Na jednom konci spektra stoja architektúry opierajúce sa o zdieľanie pamäti. Takéto systémy označujeme ako *systémy s veľmi tesnou väzbou* alebo multipočítače (prípadne multiprocesory). Sú výhodné pri potrebe sústredenia vysokého výkonu na spracovanie veľmi úzko previazaných dát, príklady aplikácií sa nájdu najmä v oblasti výpočtovo náročných problémov (riešenie deferenciálnych rovníc pri modelovaní alebo analýze reálneho sveta). Systémy s týmto typom distribúcie výkonu obvykle označujeme ako paralelné počítače.

Úplne odlišný prístup ku spolupráci procesorov nájdeme u architektúr, ktoré sa opierajú o *výmenu správ*. Jednotlivé procesory systému sú vybavené perifériami - komunikačnými radičmi, a ich prostredníctvom sú pripojované na *dvojbodové* alebo *viacbodové komunikačné kanály*.

Dvojbodové kanály spájajú dvojice procesorov a sú väčšinou základom polygonálnych prepojení procesorov. Systémy opierajúce sa o prepojenia tohto typu označujeme ako systémy s *voľnou väzbou (loosely coupled).*

U polygonálne prepojených systémov sa stretávame s problémom sprostredkovania komunikácie procesorov, ktoré nie sú priamymi susedmi, dôvodom sú prevažne praktické obmedzenia (cena prepojenia, veľký počet rozhraní). Výsledkom môže byť buď vyčlenenie určitých procesov na všetkých procesoroch iba pre komunikačné funkcie, hovoríme o vytvorení komunikačnej - *transportnej služby*, alebo vyčlenení určitej skupiny procesorov, ktorá sa na predávanie správ špecializuje, hovoríme o *komunikačnom podsystéme*.

U geograficky malých systémoch, ktoré obvykle označujeme ako *paralelné systémy* a v ktorých nehrá podstatnú rolu celkový počet spojov medzi procesormi, často siahame po *regulárnych štruktúrach* prepojení. Prepojenia procesorov sú navrhované tak, aby sme s daným počtom rozhraní jedného procesora dosiahli čo najvýhodnejších parametrov siete - čo najmenšej *strednej* alebo *maximálnej vzdialenosti* medzi procesormi (meriame ich v počte spojov na ceste medzi procesormi), čo najvyššieho *stupňa súvislosti siete* a s ňou súvisiacou *priechodnosťou*. Príklady niektorých teoreticky zaujímavých regulárnych štruktúr a štruktúr využívaných pri konštrukcii reálnych systémov uvádza obrázok Obr. 4.1:

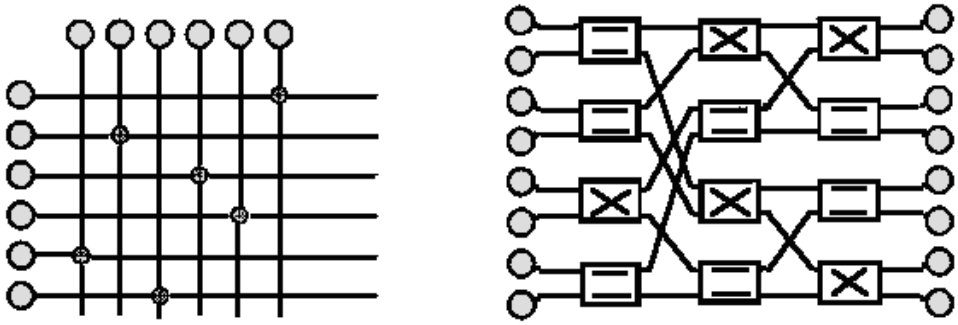


Obr. 4.1 Statické prepojovacie štruktúry

Kruh je topológiou výhodnou vzhľadom k nízkemu počtu prepojení a relatívnej efektivite algoritmov pracujúcich na kruhu. Praktické aplikácie, ktoré pre distribúciu problému využívajú hierarchický rozklad, preferujú *stromovú štruktúru*. Vysoký stupeň spojitosti majú pri pevnom počte rozhraní plošné a priestorové *mreže (meshes*), Obr. 4.1 ukazuje často využívaný toroid a hierarchicky štruktúrovanú mrežu s prekríženiami. Z teoretického hľadiska je veľmi zaujímavá *hyperkocka*, ktorá je považovaná za univerzálnu prepojovaciu štruktúru. Jej nevýhodou je s logaritmom počtu prepojených procesorov rastúci počet rozhraní.

Statické prepojenia procesorov sú vhodné v prípadoch, keď buď štruktúra prepojenia procesov odpovedá štruktúre prepojenia procesorov priamo (napr. výpočet na kruhu na kruhovej siete procesorov), alebo keď ju možno na štruktúru procesorov ľahko mapovať. Často sa dá hovoriť o potrebe prispôsobiť štruktúru prepojení procesorov štruktúre distribuovaného výpočtu.

Možností, ako prepojiť skupinu procesorov do konfigurácie odpovedajúcich konkrétnemu výpočtu je viac. Väčšinou sa opierajú o špeciálne prepojovacie prvky *krížového prepínača* alebo *prepojovacej siete*, ktorá dovolí prepojiť procesory systému podľa konkrétnych požiadaviek daného programu.



Obr. 4.2 Dinamické prepojovacie štruktúry

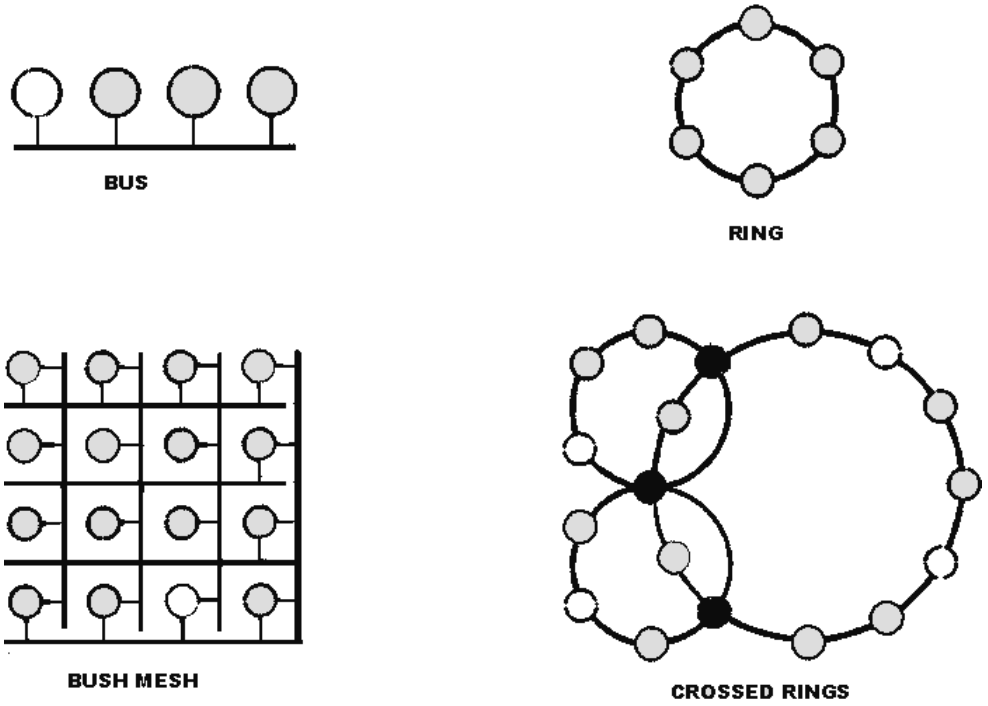
Ďalšou, univerzálne použiteľnou cestou, ako vyriešiť problém konkrétneho prepojenia procesorov prostredníctvom existujúcej statickej štruktúry je, vytvoriť komunikačný podsystém poskytujúci *virtuálne kanály* medzi procesormi. Táto cesta nemá teoretické obmedzenia bežné u mapovania a môže byť pri vhodnej technologickej podpore (špecializované komunikačné procesory) i dostatočne efektívna.

Viacbodové spoje dovoľujú vzájomné prepojenia celej skupiny procesorov, znižujú vzdialenosti medzi procesormi a dovoľujú využiť efektívnych metód *skupinovej komunikácie - multicast a broadcast*. Väčšinou sa opierajú o lokálne komunikačné systémy - zbernicové a kruhové siete s vysokou prenosovou rýchlosťou. Systémy tohto typu často označujeme ako systémy s tesnou väzbou. Ich obmedzením je, v porovnaní s prepojovacími architektúrami, obmedzená kapacita komunikačného kanálu a závislosť komunikačného podsystému na jedinom prvku.

Podobne, ako sme u dvojbodových spojov dosahovali väčšiu pružnosť sprostredkovaným prenosom - prepojovaním, možno i u viacbodových spojov prekonať rad obmedzení prepojením do zložitejších celkov. Príklady môžu byť *mreža zberníc (bus mesh)*, o ktorú sa opiera multipočítačový systém Linda, alebo prepojovanie nad kruhovými sieťami.

Distribuovaný program

Distribuovaný program možno pre výpočet na distribuovanej architektúre rozdeliť na relatívne veľké komponenty, ktoré realizujú logicky uzatvorené časti výpočtu, a ktoré si vzájomne vymieňajú minimum informácii. Hovoríme o *veľkozrnnej granularite výpočtu (coarse grain granularity).* Takýto spôsob distribúcie je typický pre systémy s málo výkonným komunikačným systémom, pre systémy s voľnou väzbou.



Obr. 4.3 Viacbodové spoje a štruktúry

Opačným extrémom vhodným pre systémy s veľmi tesnou väzbou je rozklad programu na čo najmenšie komponenty dovoľujúce optimálne využiť výpočtovú kapacitu procesoru, komunikačné náklady zanedbávame. Hovoríme o *jemnozrnnej granularite výpočtu (fine grain granularity).*

Spolupráca jednotlivých komponentov je daná spôsobom, akým si jednotlivé komponenty predávajú svoj výpočet. Pre predávanie dát, môžu využívať zdieľané premenné (ktoré sa dajú veľmi ľahko realizovať v architektúrach s veľmi tesnou väzbou), výmena správ medzi jedným odosielateľom a jedným príjemcom (ktorá sa dá veľmi ľahko realizovať v architektúrach s voľnou väzbou), alebo výmena správ medzi jedným odosielateľom a viac príjemcami - *broadcast* (ktorú sa dá veľmi ľahko realizovať v architektúrach s tesnou väzbou - lokálnych sieťach). Rozdelením programu na komponenty a definovaním ich vzájomných väzieb (a to vrátane metód) definujeme štruktúru distribuovaného programu.

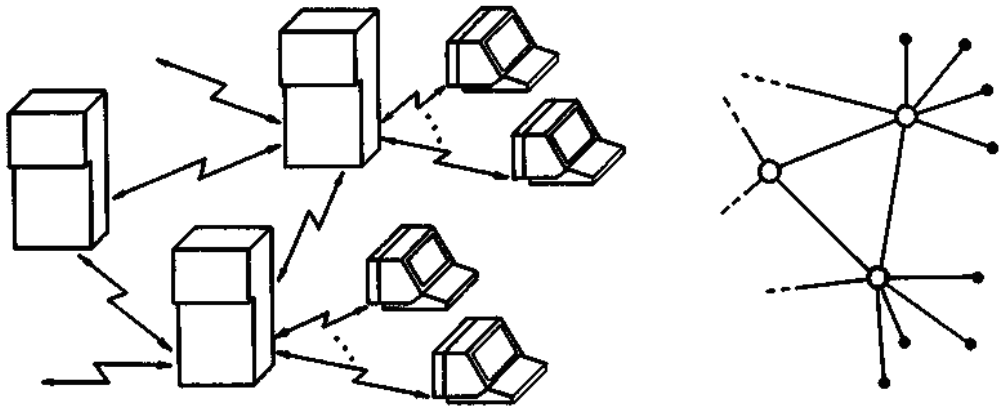
Dôležitým prvkom návrhu distribuovaného systému je rozdelenie jednotlivých komponentov distribuovaného programu na jednotlivé procesory distribuovanej architektúry. Hovoríme o *mapovaní* procesu na procesory, o mapovaní medziprocesorových väzieb na štruktúru prepojenia procesorov. Problém sa môže redukovať na stále veľmi komplikovaný problém optimálneho mapovania grafu medziprocesorových komunikácii (dvojbodových kanálov) na graf prepojenia procesorov (dvojbodovými spojmi). Často

sa však stretávame s potrebou realizovať aj zložitejšie mapovanie, akými sú mapovanie pamäte zdieľanej viacerými procesmi na architektúru bez zdieľania fyzickej pamäti (*hovoríme o distribuovanej zdieľanej pamäti*) alebo *mapovaní broadcast komunikácie* medzi procesy na polygonálni architektúru.

Mapovanie procesu na procesory a komunikačných prostriedkov programu na komunikačné prostriedky architektúry môže byť definované *staticky programom*, alebo *automaticky* pred alebo behom výpočtu.

4.2 Statické prepojovacie štruktúry

Klasické prepojovacie počítačové siete sa opierajú o špecializované uzlové počítače prepojené dvojbodovými dátovými spojmi. Úlohou uzlových počítačov je sprostredkovať prenos dát pre pripojené účastnícke počítače a terminály a zabezpečiť dáta proti chybám pri prenose na jednotlivých linkách. Nasledujúci obrázok uvádza korešpondenciu medzi štruktúrou prepojovacej siete a jej grafovým modelom.



Obr. 4.4 Prepojovacia sieť

Pre ochranu proti chybám na komunikačných kanáloch využívame detekčné kódy (obvykle cyklické) a o nich sa opierajú potvrdzovacie schémy - túto funkciu zaisťuje *linková* vrstva. Prenos dát medzi koncovými účastníkmi, sprostredkovaný radom medziľalhlých uzlových počítačov, je úlohou *sieťovej* vrstvy. Implementácia oboch vrstiev má vzhľadom k ich funkcii nutne distribuovaný charakter.

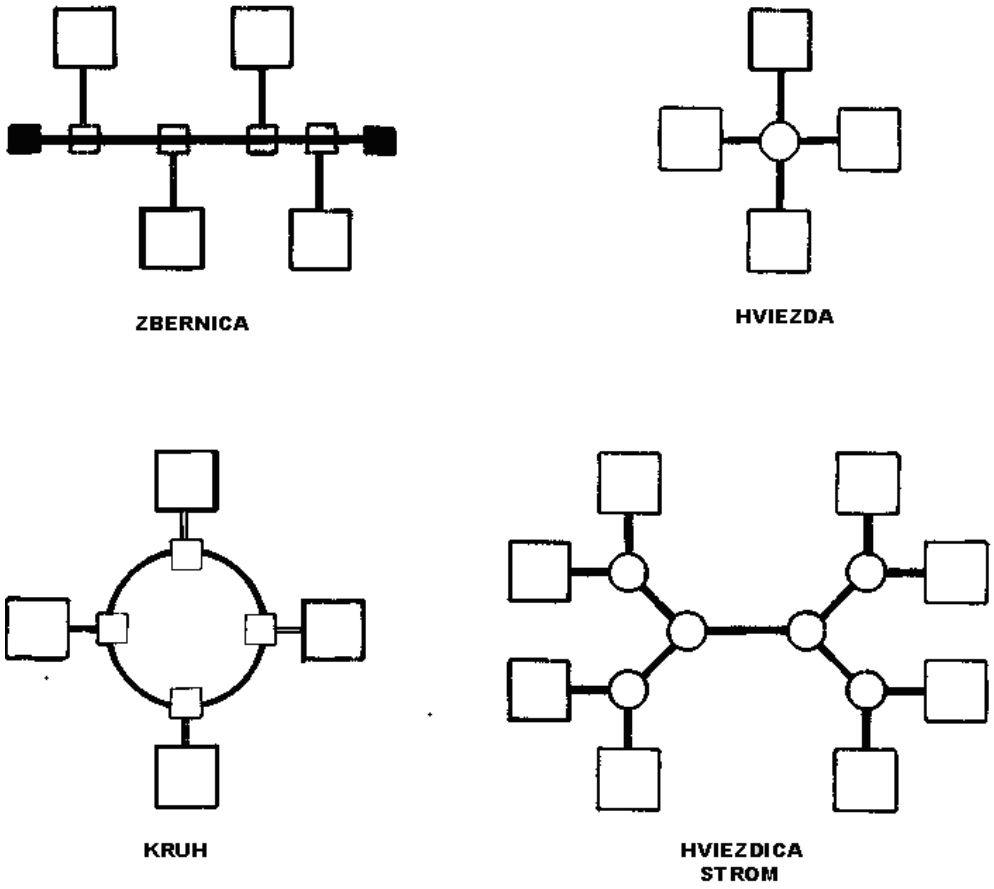
4.2.1 Topológia

Topológiou sa lokálne siete líšia od rozsiahlych počítačových sietí. Tie sa opierajú o prepojovanie paketov alebo správ - postupné predávanie správ medzi uzlami po dvojbodových spojoch(technika "store and forward"). Lokálne siete využívajú priameho prepojenia komunikačných staníc zdieľaným kanálom, signál vyslaný jednou zo staníc je prijímaný ostatným stanicami siete. Siete sú niekedy označované ako "broadcast " siete. Voľba topológie má vplyv na rad vlastností lokálnej siete:

- *rozšíriteľnosť* – možnosť a jednoduchosť doplňovania staníc do existujúcej siete,
- *rekonfigurovateľnosť* – možnosť modifikovať štruktúru siete pri chybe niektorých komponentov,

- *spoľahlivosť* – odolnosť siete voči výpadku jednotlivých komponentov, zložitosť obsluhy,
- *výkonnosť* – využitie prenosovej kapacity média.

V praxi sa stretávame s topológiou zbernicovou, hviezdicovou, stromovou a kruhovou, niektoré siete jednotlivé topológie kombinujú (napríklad ARCNet alebo dnešný Ethernet).



Obr. 4.5 Topológia lokálnych sieti

4.2.2 Zbernica

Základným prvkom zbernicovej siete je úsek prenosového média - zbernice, ku ktorej sú krátkymi odbočkami pripojené stanice siete. Prenosovým médiom je najčastejšie koaxiálny kábel alebo symetrické vedenie (točená dvojlinka), realizácia odbočiek u optického kábla je obtiažna. Vlastnosti zbernicovej siete sa dajú zhrnúť do týchto bodov:

- pasívne médium,
- jednoduché pripojovanie staníc k médiu, odolnosť proti výpadkom staníc.

Pre riadenie zbernicových sietí je využívaná rada deterministických i nedeterministických metód, ktoré využívajú fakt, že signál vysielaný jednou stanicou je prijímaný ostatnými len s malým oneskorením.

4.2.3 Hviezda

Stanice siete sú pripojené k centrálnemu uzlu samostatnými linkami. Centrálny uzol je označovaný ako HUB. Signál prichádzajúci z jednej linky rozdeľuje do ostatných liniek hviezdy. Rozlišujeme *pasívny hub*, v ktorom je signál iba delený (odporovým deličom), a **aktívny hub**, v ktorom je prijatý signál zosilňovaný tak, aby mal na všetkých linkách požadovanú úroveň. Vlastnosti topológie hviezda sa dajú zhrnúť takto:

- dvojbodové spoje medzi stanicami a centrálnym uzlom sa dajú ľahko realizovať, sieť je odolná voči výpadku jednotlivých staníc a liniek,
- sieť je však citlivá na poruchu centrálneho uzla.

Siete s topológiou hviezda ako sme ju práve popísali, sa tým, že signál jednej stanice môžu prijímať stanice ostatné, blížia zbernicovým a dajú sa u nich použiť aj obdobné metódy riadenia. Topológiu hviezda s pasívnym centrálnym uzlom často nachádzame u optických sietí.

4.2.4 Strom (hviezdica)

Stromová topológia je prirodzeným rozšírením topológie typu hviezda. Stretávame sa s ňou u širokopásmových sietí a u sietí využívajúcich pre prenos svetlovody. Vlastnosti stromovej topológie sú podobné ako u sietí typu hviezda:

- odolnosť siete voči výpadkom jednotlivých staníc a liniek,
- citlivosť na výpadky uzlov,
- ľahká rozšíriteľnosť,
- dvojbodové spoje.

Stromové (hviezdicové) siete používajú podobné metódy ako siete zbernicové.

4.2.5 Kruh

U kruhových sietí sú komunikačné stanice prepojené spojmi, ktoré sú využívané iba jednosmerne. Signál vyslaný jednou stanicou je postupne predávaný ostatným staniciam kruhu. Základným prvkom stanice je krátky posuvný register a po obehu sieťou sa vracia k stanici ktorá ho vyslala. Vlastnosti kruhových sietí sa dajú zhrnúť do týchto bodov:

- dvojbodové jednosmerné spoje sa dajú ľahko realizovať i na svetlovodoch,
- v sieti sa dajú kombinovať rôzne médiá (pre krátke spoje elektrické, pre dlhé svetlovody,
- sieť je však citlivá na výpadok ľubovoľného prvku (stanice alebo spoja).

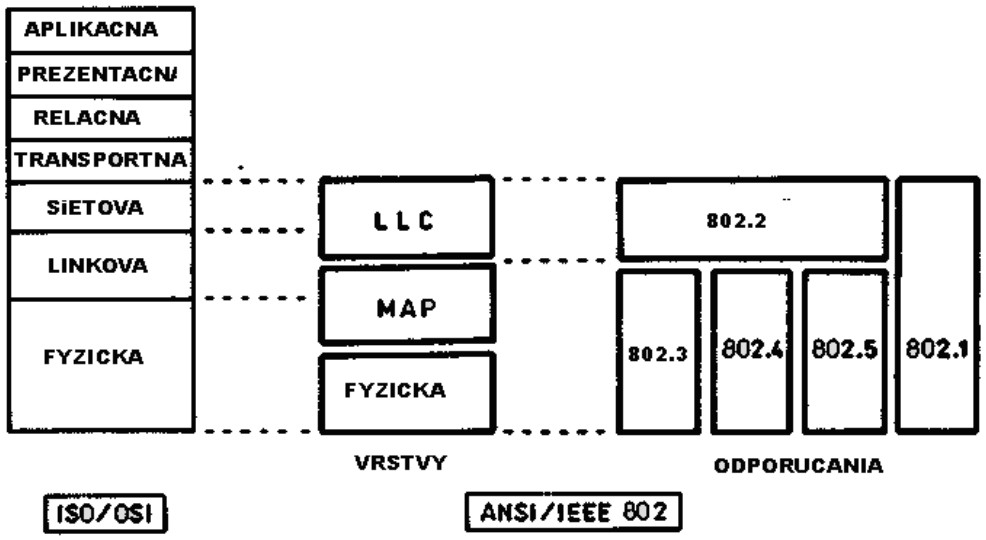
U kruhových sietí sú pravidelne používané deterministické metódy riadenia.

Uvedené delenie sietí na siete zbernicové, stromové a kruhové sa opiera o *elektrickú topológiu* teda o spôsob vzájomného prepojenia staníc. Z hľadiska vlastnosti siete má veľký vplyv aj *topológia fyzická* (spôsob vedenia káblov) a *topológia logická* (metóda spolupráce staníc u deterministických metód).

4.3 Model ISO/OSI

Zložitosť problémov, s ktorými sa stretávame pri prepojovaní počítačov do počítačových sietí, vyžaduje použitie vhodných modelov, ktoré umožnia vytvorenie štandardizovaných prvkov a uľahčia ich použitie v praktických implementáciách.

Na sieťové vybavenie, technické a programové sme zvyknutí sa pozerať ako na systém funkčných vrstiev, v ktorom každá vyššia vrstva rozširuje možnosti vrstvy nižšej. Pre prepojovacie počítačové siete, z ktorých sa na začiatku osemdesiatych rokov vyvinuli dnes prevádzkované verejné dátové siete bol vytvorený štandardný model sieťovej architektúry označovaný ako ISO/OSI (ISO-Open Systems Interconnection Standart). Model OSI popisuje komunikáciu zaisťovanú počítačmi ako hierarchiu siedmych technických a programových prostriedkov, kde každá z vrstiev zaisťuje funkcie potrebné pre vrstvu vyššiu a využíva služby vrstvy nižšej. Medzi jednotlivými vrstvami (formou štandardov a doporučení) definované rozhrania (medzivrsvové protokoly), medzi prvkami rovnakej vrstvy sú definované pravidlá komunikácie (vrstvové protokoly). Architektúru vrstiev ilustruje nasledujúci obrázok (Obr. 4.6):



Obr. 4.6 Sieťová architektúra

Fyzická vrstva definuje fyzické prepojenie medzi prvkami siete, jeho mechanické vlastnosti (konektory, typ média), elektrické vlastnosti (napäťové úrovne, spôsob kódovania a modulácie) a u lokálnych sietí aj spôsob prepojenia jednotlivých počítačov a metódu prístupu k prenosovému médiu.

Linková vrstva definuje pravidlá pre predávanie správ. Správy sú sieťou prenášané v pevne definovaných rámcoch, rámce dovoľujú chrániť predávané dáta proti chybám. Štruktúra rámca často limituje dĺžku prenášaných blokov dát a hovoríme o takzvaných paketoch.

Sieťová vrstva definuje spôsob akým sa pakety pohybujú po sieti ako si ich jednotlivé prvky siete predávajú na ich ceste od odosielateľa k adresátovi. Mechanizmy vrstvy sa konečne starajú aj o ochranu siete proti nadmernej záťaži.

Transportná vrstva definuje adresáciu počítačov a aplikačných programov v sieti, zaisťuje vytváranie dočasných komunikačných spojení medzi nimi a konečne aj rozklad správ do paketov a skladanie paketov do správ.

Relačná vrstva vyvára logické rozhranie pre aplikačné programy, ktoré používajú služby siete. Definuje spôsob komunikácie programov a užívateľský pohľad na komunikačný kanál.

Prezentačná vrstva transformuje prenášané dáta - zaisťuje prevody kódov a formátov dát pre nekompatibilné počítače, kompresiu prenášaných dát a konečne aj utajovanie prenášaných dát.

Aplikačná vrstva je konečne vrstvou štandardných aplikačných programov, ktoré sieť využívajú.

Model OSI sa stal základom i pre lokálne siete, ktoré používajú iné prenosové médiá a potvrdzovacie techniky a spôsob predávania správ ako staršie siete prepojovacie. Model IEEE 802 pokrýva tri najnižšie vrstvy architektúry OSI, vrstvu fyzickú, linkovú a sieťovú, a je členený na samostatné doporučenia týkajúce sa jednotlivých technológií.

4.4 Riadenie prístupu k médiam

Táto podkapitola podrobne rozoberie prístup k médiám podľa typu použitej siete. Podkapitola sa bude venovať lokálnym komunikačným systémom, statickému rozdeleniu kapacity kanálu, centralizovanému riadeniu, distribuovanému riadeniu a riadeniu sieti s náhodným prístupom.

4.4.1 Lokálne komunikačné systémy

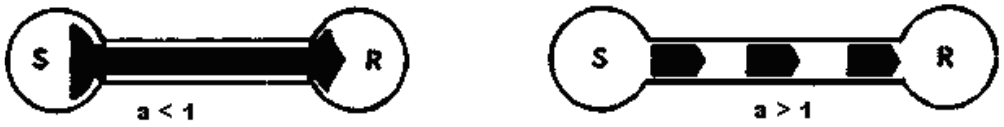
Lokálne počítačové siete sú vhodným technickým prostriedkom tam, kde je treba rozložiť výpočtovú kapacitu z dôvodov ľahkej dostupnosti, minimalizácie dátových prenosov alebo spoľahlivosti, a súčasne zaistiť zdieľanie dát. Aplikáciami sú meracie a sledovacie systémy vo vede a zdravotníctve, riadenie technologických procesov v priemysle a automatizácia administratívy. Oblasť lokálnych počítačových sieti je veľmi široká. Patrí sem problematika:

- prenosových médií, ich vlastnosti a využitia,
- topológia sieti, teda spôsobu, akým sú počítače prepojené prostredníctvom komunikačných kanálov,
- riadenie prenosu dát s ohľadom na využitie kapacít komunikačných kanálov,
- spolupráca aplikačných procesov, ktoré si vzájomne vymieňajú informácie prostredníctvom komunikačných kanálov - predávaním správ.

Rozľahlosť siete

Prívlastok "lokálna" používame k vyznačeniu skutočnosti, že siete pokrývajú malé územia. Rozmery siete však nie sú obmedzené našimi potrebami, ale technickými dôvodmi ktoré rozľahlosť siete obmedzujú.

Pokúsme sa rozľahlosť siete definovať ako pomer medzi oneskorením signálu v sieti a strednou dobou pre prenos jedného paketu to pri danej rýchlosti C. Pre siete ktoré označujeme ako rozľahlé platí a > 1. Siete ktoré budeme označovať ako lokálne majú a < 1. V lokálnych sieťach je prenosové médium využité v danom okamžiku pre prenos jediného paketu v rozľahlých sieťach môže byť médium využité na prenos viac paketov súčasne.



Obr. 4.7 Prenos v sústredenej a rozľahlej sieti

Medzi siete ktoré charakterizujeme ako rozľahlé patria aj lokálne siete s vysokou rýchlosťou prenosu a veľkými prekonávanými vzdialenosťami (optické mestské siete). Sústredené siete zahrňujú bežné lokálne siete a siete rádiové (malá prenosová rýchlosť). Pre rad metód riadenia musíme zaistiť veľmi malú hodnotu parametra a, typicky je a mensie ako 0,1.

Prenosové médium

Jedným z dôležitých prvkov ktorý charakterizuje konkrétnu lokálnu sieť je použité prenosové médium. Okrem malého počtu sietí ktoré používajú paralelný prenos po viacvodičových kábloch (napr. zbernicová sieť Cluster One alebo kruhová sieť Twentenet) ide u väčšiny sietí o prenos sériový. Najčastejšie sa stretávame s nesymetrickým (koaxiálny kábel) a symetrickým (krútená dvojlinka - twist) vedením, u niektorých sietí sa začínajú objavovať svetlovody.

Existujú aj siete ktoré pre prepojenie staníc využívajú vysokofrekvenčné rádiové kanály alebo kanály optické.

Symetrické vedenie

Symetrické vedenia vo forme krútenej dvojlinky (twisted pair), ako ho poznáme z telefónnych káblov je najlacnejším prenosovým médiom. Vo väčšine prípadov ide o tienený (STP - Shielded Twisted Pair) alebo netienený (UTP - Unshielded Twisted Pair), jednoduchá alebo dvojitá dvojlinka, ktorá dovoľuje prenášať signál na vzdialenosť niekoľko stoviek metrov prenosovou rýchlosťou až 16 Mb/s (kruhová sieť IBM Token Ring) najčastejšie sa však stretávame s prenosovými rýchlosťami do 10 Mb/s (sieť Ethernet 10BASE-T). Na krátke vzdialenosti sa však dá ísť aj vyššie (sieť FDDI, ktorá pracuje na 100 MB/s).

Symetrické vedenie je používané pre prenos kódovaných signálov v základnom pasme, veľmi často sa stretneme s použitím napäťových úrovni odpovedajúcich štandardným rozhraniam RS-422 EIA a RS-485 EIA.

Nesymetrické vedenie

Nesymetrické vedenia (koax.kabely) dovoľujú využitie 0-50 MHz v základnom pásme (kódovaný dátový signál) a pásma 50 - 500 MHz v prekladanom pásme (modulovaný signál).V základnom pásme sa dá dosiahnuť prenosová rýchlosť v rozmedzí 1 - 20 MB/s. V prekladanom pásme sa dá vytvoriť skupina prenosových kanálov s prenosovou rýchlosťou až 20 MB/s. Pri prenose v základnom pásme obmedzujú elektrické vlastnosti vedenia preklenutú vzdialenosť stoviek metrov, preto sú často využívané drahé špeciálne káble (ako je tomu napr. u siete Ethernet). Preložené pásmo sa dá využiť pre prenos na kilometrové vzdialenosti, podstatnou výhodou je možnosť použiť káble a ďalšie prvky určené

pre káblovú televíziu. Koaxiálny kábel je typickým médiom lokálnych sietí, má relatívne dobrú odolnosť proti rušeniu.

Svetlovodivé vlákna

Svetlovodivé vlákna využívajú infračervené a viditeľné oblasti svetelného spektra pre prenos dát rýchlosťami do 1Gb/s na kilometrové vzdialenosti. Výhodou svetelných vlákien je vysoká prenosová kapacita pri nízkej cene média a veľká odolnosť proti rušeniu, nevýhodu je vysoká cena konektorov. Stretávame sa s nimi v lokálnych sieťach s kruhovou alebo stromovou topológiou.

Kapacita prenosového kanála

Základným parametrom, ktorý obmedzuje prenosovú rýchlosť kanála je šírka použitého kmitočtového pásma. Spojitý signál, ktorý neobsahuje zložky s vyšším kmitočtom než W sa dá plne charakterizovať 2W vzorky za sekundu a z týchto vzorkou signál opäť rekonštruovať. Ináč povedané spojitým signálom s kmitočtovým spektrom obmedzeným kmitočtom W nemôžeme preniesť viac než 2W vzorkou za sekundu. Ak môže každá vzorka nadobúdať V diskrétnych hodnôt potom pre prenosovú rýchlosť C platí Nyquistova veta.

$$C = 2.W.\log_2(V) \qquad \text{[b/s, Hz]} \qquad (4.1).$$

Počet úrovní signálu V sa nedá s ohľadom na poškodenie spojitého signálu pri prenose (obvykle toto poškodenie charakterizujeme prídavným šumom) ľubovolne zvyšovať, teoretický limit prenosovej rýchlosti C kanálu o šírke W a odstupom signálu od šumu S/N udáva Shannonova veta.

$$C = W.\log_2(1 + S/N) \qquad \text{[b/s, Hz]} \qquad (4.2).$$

Kódovanie a modulácia

Neupravený dátový signál nie je vhodný pre priamy prenos dátovým kanálom. Obsahuje jednosmernú zložku, ktorej prenos je v niektorých prípadoch obtiažne zaistiť, či už pre elektrické vlastnosti kanálu alebo pre nutnosť galvanického oddelenia vlastného kanálu transformátorom. Ďalšou nepríjemnou vlastnosťou pôvodného signálu je nezaručený výskyt elektrických zmien, o ktoré sa dá oprieť pri vzorkovaní na strane prijímača.

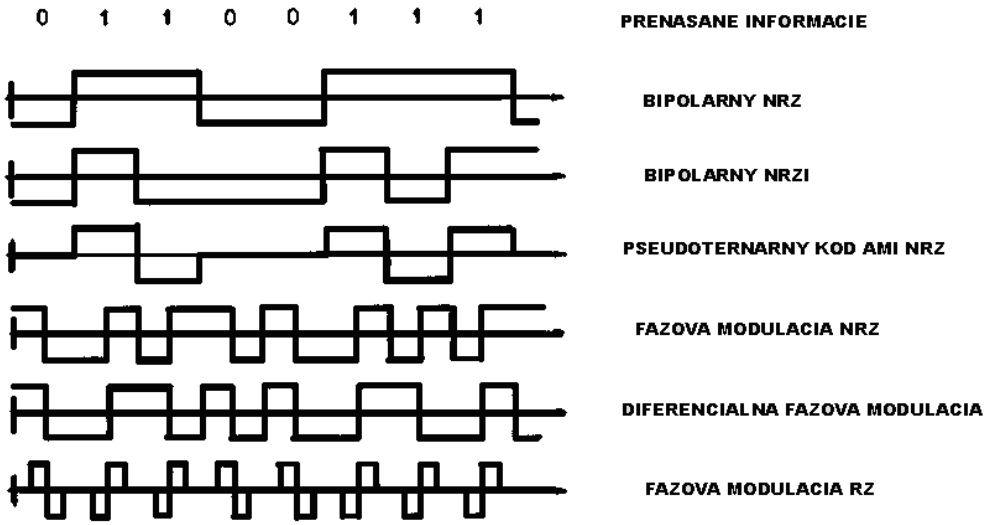
Dátový signál môžeme zbaviť jednosmernej zložky a doplniť o zmeny uľahčujúce jeho príjem vhodným kódovaním. Kód NRZI je používaný u sietí pracujúcich v *základnom pásme* a v spojení s moduláciou v sieťach širokopásmových. Fázovou moduláciou NRZ (označovanou ako PSK alebo kód Manchester) používa napríklad sieť ETHERNET). Diferenciálna fázová modulácia NRZ (označovaná aj ako DPSK alebo diferenciálny kód Manchester) je použitá v lokálnych sieťach podľa doporučenia IEEE 802.5.

Prenos kódovaného dátového signálu označujeme ako prenos v základnom pásme. Pokiaľ chceme pre prenos využiť frekvenčné pásmo, ktoré neobsahuje základné harmonické prenášaného dátového signálu musíme siahnuť k modulácii. Ak je nosným signálom harmonický signál.

$$u(t) = U.\sin(\omega.t + \phi) \qquad (4.3)$$

môžeme moduláciou ovplyvniť aj jeho amplitúdu, frekvenciu alebo fázu. V lokálnych sieťach využívajúcich elektrické kanály používame najčastejšie frekvenčnú alebo fázovú moduláciu, v lokálnych sieťach optických používame moduláciu amplitúdovú.

Frekvenčné spektrum modulovaného harmonického signálu leží v inej frekvenčnej oblasti, než spektrum signálu modulačného - hovoríme o prenose v *preloženom pásme.*



Obr. 4.8 Kódovanie dátového signálu v lokálnych sietiach

Zdieľanie kanálu

Pokiaľ prenosové médium poskytuje väčšiu šírku pásma (väčšiu prenosovú rýchlosť) než je potrebné pre realizáciu jediného prenosového kanálu môže médium zdieľať viacero kanálov.

Frekvenčný multiplex je založený na rozdelení pásma prenášaného médiom na oddelené frekvenčné intervaly, ktoré ďalej využívame pre vytvorenie samostatných prenosových kanálov. Pre prevod dátového signálu do daného frekvenčného pásma a naspäť používame modemy doplnené o selektívne filtre. Frekvenčný multiplex je základom širokopásmových lokálnych sietí.

Časový multiplex využíva fakt, že spojitý signál plne využívajúci pásma prenášaného médiom je schopný pracovať s najvyššou prenosovou rýchlosťou ako potrebujeme pre jeden dátový kanál. V takom prípade sa dá rýchly prenosový kanál zdieľať viac pomalými kanálmi. Časový multiplex je dnes ľahšie realizovateľný ako multiplex frekvenčný a jeho adaptívne formy (zdieľanie dátového kanálu takým spôsobom aby bola maximálne využitá jeho kapacita) sú princípom prevážnej väčšiny lokálnych sietí a sietí integrovaných služieb (ISDN).

Synchronizácia

Využitie kanálov schopných prenášať spojité signály (elektrické vedenia, svetlovody) pre prenos dát predpokladá, že prijímač bude prijímaný signál vzorkovať pravé v týchto miestach, ktoré použil pre uloženie prenášaných dát vysielač. Zaistiť vzájomnú synchronizáciu vysielača a prijímača majú za úlohu metódy bitovej synchronizácie.

Bitová synchronizácia sa dá zaistiť niekoľkými spôsobmi. Môžeme napríklad vedľa vlastného dátového signálu prenášať signál hodinový, ktorý nám označuje miesta, v ktorých mame vzorkovať.

Alebo môžeme prijímač vybaviť samostatným generátorom hodín a tento generátor fázovo synchronizovať na začiatku napríklad každého prenášaného znaku. Oscilátory vysielača a prijímača musia mať tak blízke frekvencie, aby sa nemohla prejaviť odchýlka

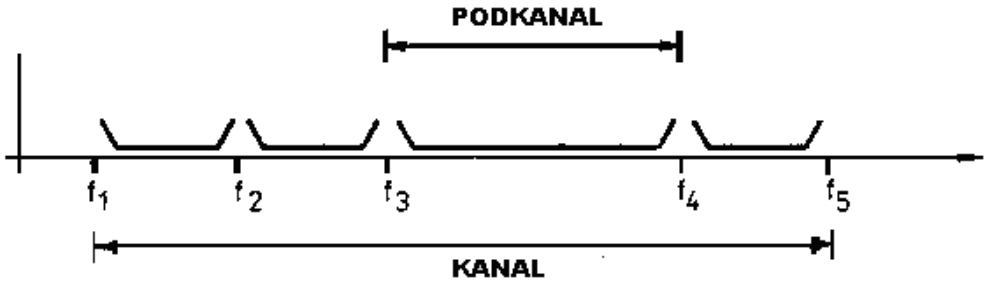
vo fáze ich signálu behom prijímania jedného znaku. Uvedenú metódu poznáme pod názvom aritmetický alebo asynchrónny prenos u terminálov a jej výhodou je súčasné zaistenie znakovej synchronizácie.
Ďalšou úlohou, ktorú musia obvody prijímača riešiť, je určenie začiatku jednotlivých znakov a začiatku správy v prenášanej bitovej postupnosti. Hovoríme o *znakovej* a *rámcovej* synchronizácii a obvykle ju zaisťujeme porovnávaním úseku prijímanej bitovej postupnosti so synchronizačným znakom alebo rámcovou značkou (krídlová značka, flag).

4.4.2 Statické rozdelenie kapacity kanálu

Pevné rozdelenie kapacity zdieľaného kanálu označujeme ako statické prideľovanie. Zahrňuje známe metódy rozdelenia kapacity - frekvenčný a časový multiplex.

Frekvenčný multiplex

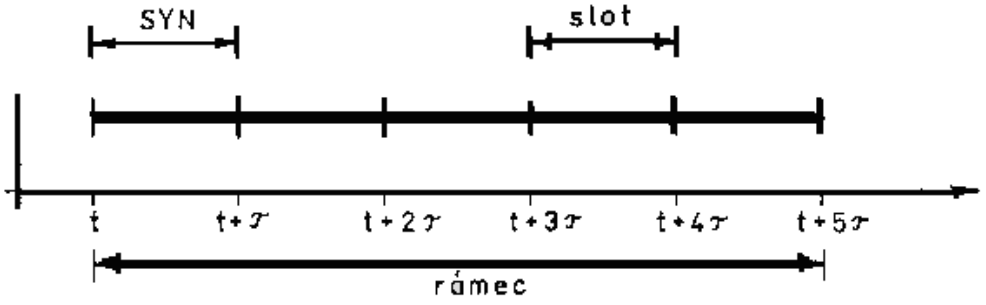
Frekvenčný multiplex (FDMA - Frequency Division Multiple Access) využíva skutočnosť, že pre prenos dát s danou prenosovou rýchlosťou vystačíme s určitou šírkou frekvenčného pásma. Ak je šírka pásma, ktorú nám poskytuje prenosový kanál väčšia, dá sa kanál rozdeliť na viac podkanálov a každý z nich použiť nezávisle. Frekvenčný multiplex je základom širokopásmových sietí.



Obr. 4.9 Princíp frekvenčného multiplexu

Časový multiplex

Pri časovom multiplexe (TDMA - Time Division Multiple Access) prideľujeme postupne prenosový kanál jednotlivým staniciam. Každej stanici je vyhradený časový úsek (slot), v ktorom môže vysielať paket určité dĺžky. Časové úseky jednotlivých staníc sa pravidelne striedajú s periódou, ktorú obvykle označujeme ako rámec (frame).



Obr. 4.10 Princíp časového multiplexu

Pre prenos dát sa zrejme nedá plne využiť kapacita kanála, v každom časovom slote je nutné venovať čas na sfázovanie prijímača a rámec je nutné doplniť synchronizačným slotom SYN. Metóda je použiteľná pre lokálne siete s malým rozsahom a je menšie ako 0,1.

Nevýhodou pevného rozdelenej kapacity zdieľaného kanálu je neschopnosť prispôsobiť jeho využitie nárazovému charakteru požiadaviek jednotlivých staníc. Optimálneho využitia kapacity by sme dosiahli v prípade, že by sme mali k dispozícii algoritmus, ktorý by evidoval požiadavky jednotlivých staníc a prideľoval podľa nich jednotlivým staniciam médium. Takému algoritmu sa môžeme vhodnými metódami riadenia iba do určitej miery priblížiť.

O metóde TDMA hovoríme ako o synchrónnom časovom multiplexe. Asynchrónny časový multiplex (ATDMA) ako ho poznáme napríklad u koncentrátorov terminálov odpovedá ideálnemu prideľovaniu M/M/1.

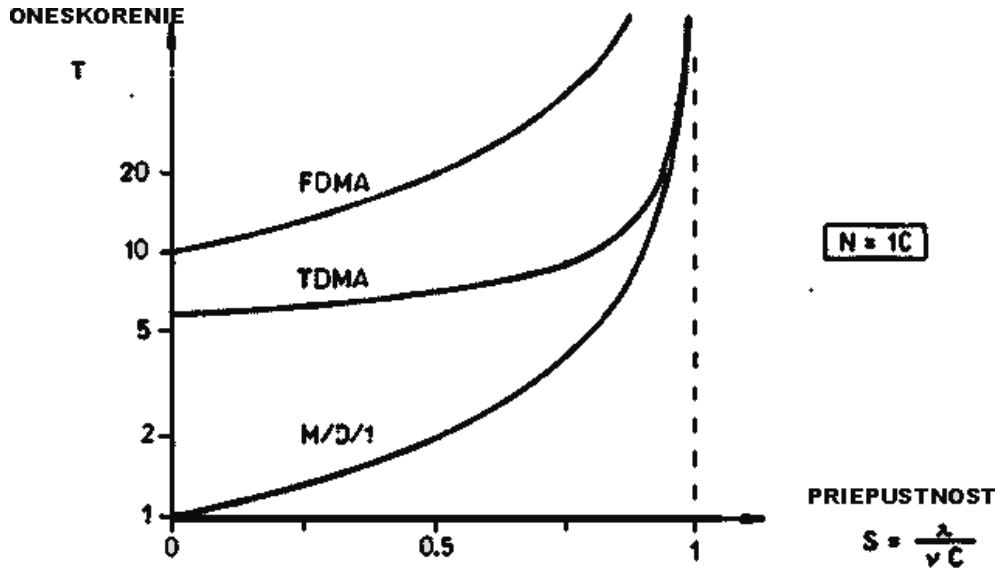
4.4.3 Centralizované riadenie

Najjednoduchšou cestou ako prispôsobiť riadenie prístupu jednotlivých staníc ku kanálu náhodnému charakteru ich požiadavkám pri zaistení čo najmenšieho oneskorenia, je vyhradiť jednu zo staníc ako stanicu riadiacu. Riadiaca stanica prideľuje kapacitu kanálu ostatným - podriadeným staniciam. Výhoda jednoduchého algoritmu je porušená potrebou obetovať časť kapacity kanálu (alebo špeciálny podkanál) pre žiadosti podriadených staníc.

Ďalšou nevýhodou je závislosť siete na spoľahlivosti riadiacej stanice.

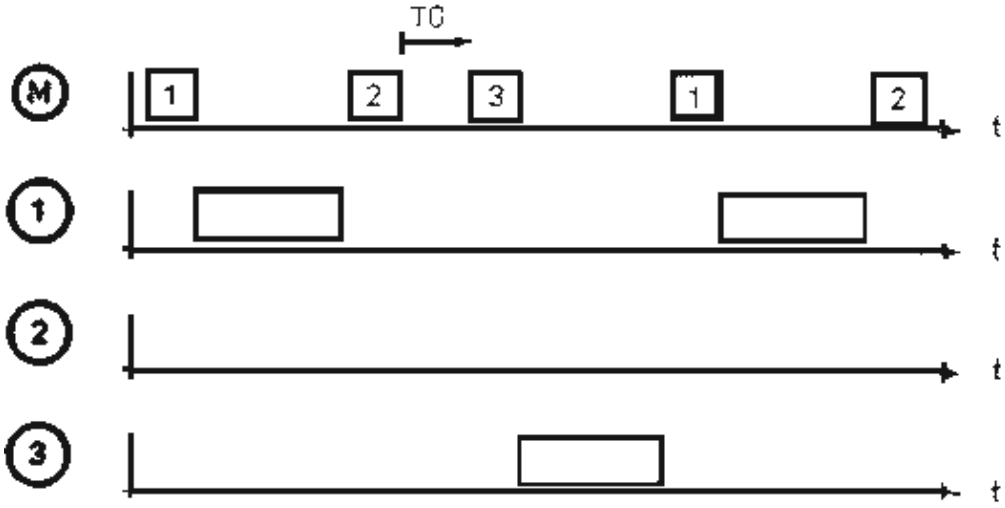
Prideľovanie na výzvu

Prideľovanie na výzvu je najstaršou metódou adaptívneho prideľovania kapacity prenosového kanálu (používajú ju napríklad linkové protokoly podľa ISO 1745). Najjednoduchšia modifikácia metódy je cyklická výzva.



Obr. 4.11 Závislosť oneskorenia paketu na záťaži

Centrálna stanica postupne vyzýva stanice podriadené. Pokiaľ má podriadená stanica pripravené dáta k odoslaniu, potom ich odošle, inak iba potvrdí výzvu alebo neodpovie. Cyklická výzva je výhodná pre malý počet staníc na spoji a malé oneskorenie signálu (a je mnesie 1).Príklad prenosu dát po kanále riadeného cyklickou výzvou a vplyv počtu staníc a záťaže na stredné oneskorenie paketu uvádza obrázok Obr. 4.12.



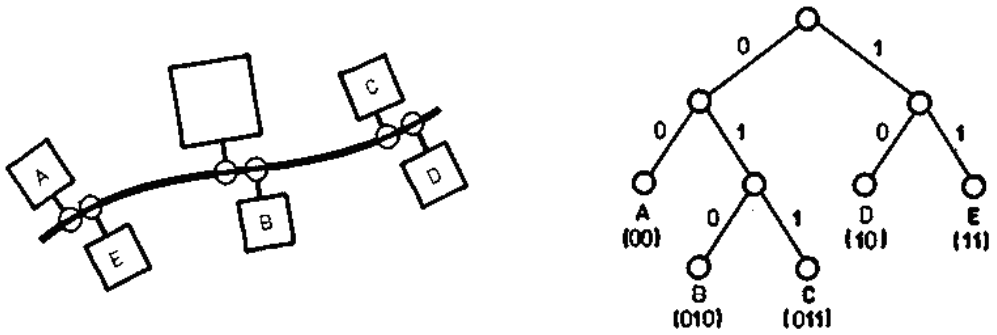
Obr. 4.12 Riadenie kanálu cyklickou výzvou

Metóda cyklickej výzvy ma rozumné správanie pri vysokom rovnomernom využití kapacity kanálu, pre malé zaťaženie kanálu a veľký počet staníc je oneskorenie paketu zbytočne dlhé.

Zlepšenie prináša modifikácia metódy použiteľná pri špeciálnom type kanálu, ktorý dovoľuje stanici rozoznať, či v danom okamihu vysiela jedna alebo viac staníc. Metóda je založená na fakte, že pri malom zaťažení a veľkom počte staníc sa dá aktívna stanica podstatne rýchlejšie nájsť v *binárnom vyhľadávaní.*

Pre binárne vyhľadávanie stanice rozdeľujeme do dvoch približne rovnako veľkých skupín, a každú skupinu ďalej rozdelíme do dvoch približne rovnako veľkých skupín, atď., až máme v každej skupine jedinú stanicu. Príklad rozdelenia staníc do skupín uvádza obrázok Obr. 4.13.

Riadiaca stanica pri vyhľadávaní aktívnej stanice postupne vyzýva skupiny staníc začínajúc od koreňa binárneho stromu, aktívne stanice odpovedajú signálom po zdieľanom kanále. Keď je vo vyzývanej skupine jediná aktívna stanica, tak môže zahájiť prenos paketu.Ak je aktívnych staníc viac, riadiaca stanica zostúpi v strome o jednu úroveň a výzvu opakuje.



Obr. 4.13 Metóda binárneho vyhľadávania

Algoritmus binárneho vyhľadávania je rýchlejší pre malé záťaže, algoritmus cyklickej výzvy pre záťaže veľké. Ak prispôsobíme úroveň, od ktorej prechádzame binárny strom, zmeranej záťaži, dajú sa dosiahnuť optimálne výsledky. Metódu označujeme ako metódu *adaptívnej výzvy*.

Bitbus

Sieť Bitbus bola navrhnutá firmou Intel ako lacná lokálna sieť pre distribuované systémy riadenia využívajúce jednočipové mikropočítače. Prenosovým médiom je dvojica točených vodičov, elektrické signály odpovedajú doporučeniu RS - 485 EIA (Bitbus), čo je modifikácia sériového rozhrania RS - 422 pre zbernicu. Na segment zbernice o dĺžke až 330 m sa dá pripojiť 28 staníc, jednotlivé segmenty je možné prepojovať opakovačmi, je však nutné dodržať dve obmedzenia - najviac 250 staníc v sieti a najviac tri opakovače medzi ľubovoľnými dvoma stanicami.

Dáta sú prenášané rýchlosťou 375 kb/s zakódované v kóde NRZI. Pri menších požiadavkách sa dá zvoliť pomalší variant siete s rýchlosťou 62.5 kb/s, ktorá na druhu stranu dovolí predĺžiť segment na dĺžku 1300m. Štruktúra rámca paketu je odvodená od bitovo orientovaných linkových procedúr.

Riadením siete je poverená jedna riadiaca stanica, ktorá vyzýva k vysielaniu jednotlivé stanice podriadené, algoritmus výzvy je podriadený potrebám konkrétnej aplikácie.

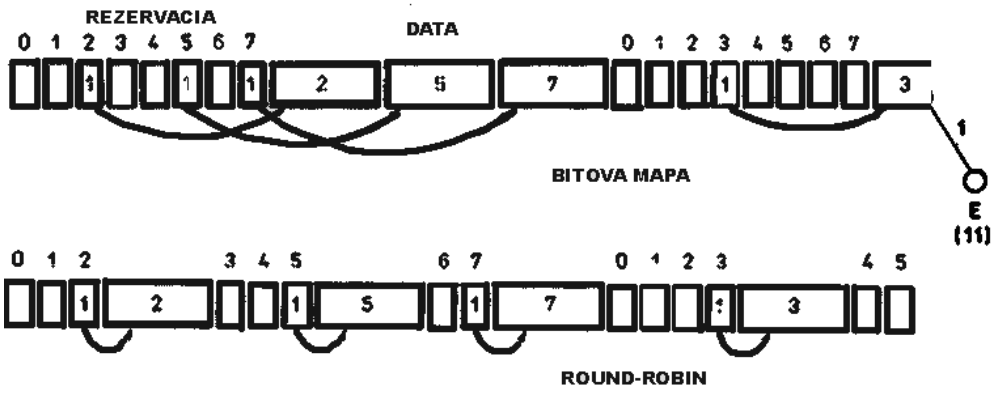
4.4.4 Distribuované riadenie

Nevýhodou centralizovaného prideľovania je závislosť na centrálnej stanici, výhodou (proti ďalej popisovaným metódam náhodného prístupu) je limitovaná doba predania paketu adresátovi. Tuto vlastnosť zachovávajú i deterministické metódy distribuovaného riadenia, ktoré odstraňujú závislosť na jedinej riadiacej stanici. Patria sem metódy, ktoré majú skôr teoreticky charakter, praktické použitie ma rezervačná metóda, metóda binárneho vyhľadávania (prioritného prístupu) a metóda logického kruhu (token passing bus).

Rezervácia kanála

Rezervačné metódy sú distribuovanou variantov prideľovania kanálu na žiadosť. Vyčleňujú z prenosového kanálu rezervačný rámec, v ktorom si jednotlivé stanice rezervujú pridelenie dátového kanálu. Rezervačný rámec ma charakter *bitovej mapy* - každej stanici je pridelený minislot o dĺžke odpovedajúcej dobe šírenia signálu médiom, v ktorom môže stanica požiadať o pridelenie dátového slotu. Dátové sloty sú prideľované v poradí minislotu

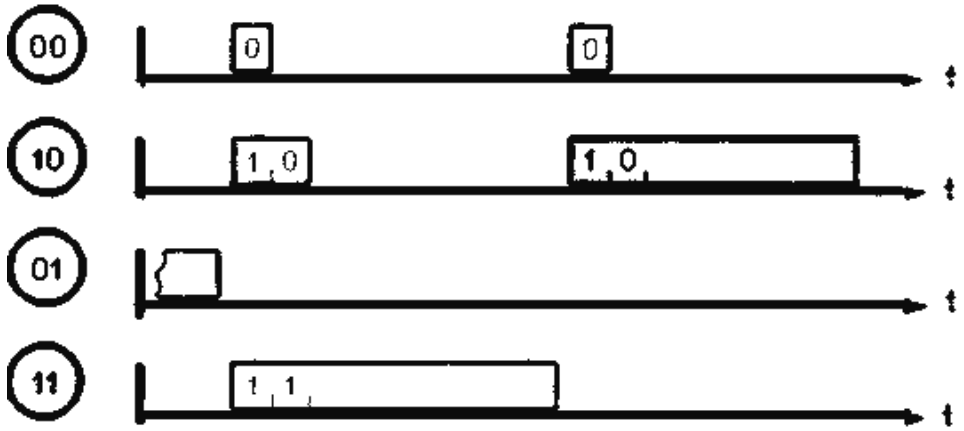
v rezervačnom rámci, algoritmus prideľovania beží synchrónne na všetkých staniciach. Obrázok Obr. 4.14 uvádza dve modifikácie rezervačnej metódy, popísanú metódu bitovej mapy a modifikáciu "round-robin".



Obr. 4.14 Rezervačné metódy

Binárne vyhľadávanie (prioritný prístup)

Ak pridelíme jednotlivým staniciam jednoznačné binárne adresy, môžeme ich využiť pre bezkolízne prideľovanie kanálu. Predpokladajme, že správu ma pripravenú k vyslaniu niekoľko staníc. Stanica zahajuje svoju činnosť vysielaním adresy začínajúc od najvyššieho bitu a vyhodnocuje situáciu na médiu. Ak stanica zisti na médiu bit rovnaký s vyslaným bitom adresy, môže vo vysielaní pokračovať, ak však tomu tak nie je, musí vysielanie zastaviť. Po odvysielaní adresy môže pravé jedna zo staníc pokračovať odosielaním pripravenej správy a celý postup sa cyklicky opakuje.



Obr. 4.15 Binárne vyhľadávanie

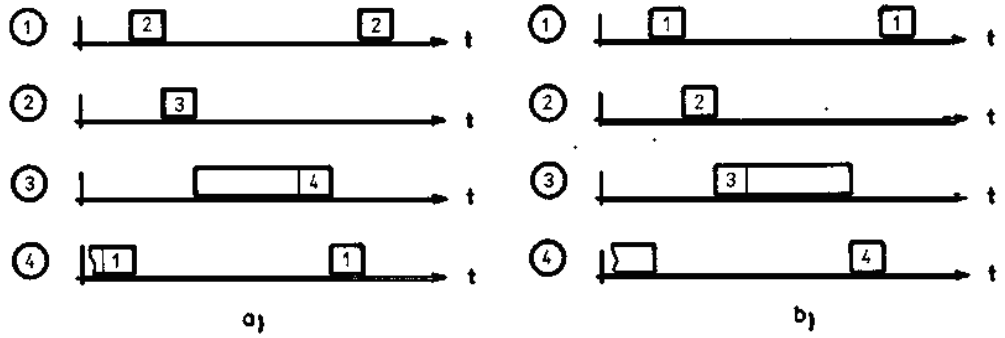
Adresácia staníc definuje ich prioritu a metóda je preto označovaná ako *prioritný prístup*. Ak umožníme zmeny adries staníc dosiahneme ich rovnoprávnosť.

Metoda binárneho vyhľadávania aktívnej stanice využíva špeciálne schopnosti niektorých prenosových kanálov - realizovať funkciu súčtu alebo súčinu signálu viacerých

staníc (takým kanálom je napríklad zbernica s otvorenými kolektormi). V praxi sa často ako signál slúžiaci k vyhľadávaniu používa šum.

Logický kruh (Token Passing Bus)

Prakticky univerzálne využívanou metódou distribuovaného prideľovania je metóda logického kruhu alebo niektorá jej modifikácia.

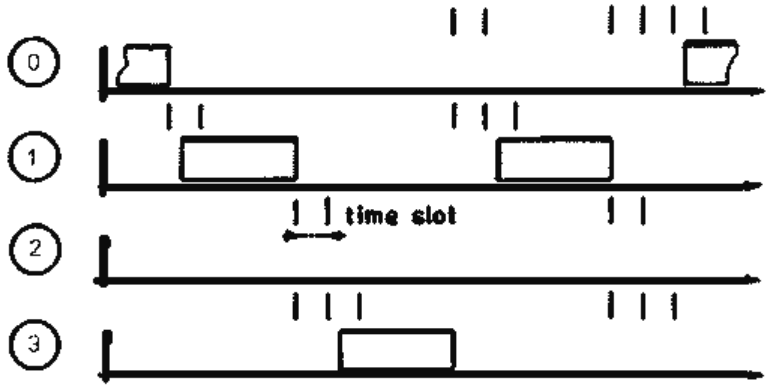


Obr. 4.16 Logický kruh

Stanice zdieľajúce prenosový kanál sú označené adresou a tieto adresy tvoria cyklickúpostupnosť. Každá zo staníc vie svoju vlastnú adresu a adresu stanice, ktorá smie vysielať po nej. Jedna zo staníc je vždy aktívna, v tomto stave smie odvysielať dátový paket, alebo predať riadenie nasledujúcej stanici špeciálnym paketom - *poverením* (označovaný ako *token*).Určitým problémom metódy je jej štartovanie a zmena postupnosti staníc pre stanice, ktoré počas fungovania siete z logického kruhu odstupujú alebo sa naopak do neho chcú zapojiť.Metódy pre modifikáciu postupnosti staníc v týchto prípadoch sú označované ako metódy *rekonfigurácie*, príklady rekonfigurácie ukazujeme pre sieť ARCNet a pre siete podľa doporučenia IEEE 802.4.

Virtuálny logický kruh

Výhodou logického kruhu je zaručenie časového limitu pre doručenie dátového paketu, nevýhodou je avšak podobne ako u predchádzajúcich metód distribuovaného riadenia zbytočne veľké oneskorenie pri malej záťaži. Znížením réžie spôsobené predávaním poverení radou neaktívnych staníc dosahuje metóda riadenia označovaná ako virtuálny logický kruh.



Obr. 4.17 Virtuálny logický kruh

Stanica (nech ma adresu m) sleduje premávku na médiu a ak dôjde po ukončení vysielania stanice s adresou n k uvoľneniu média na dobu ((m- n))modN).r, kde N je počet a r je doba šírenia signálu médiom, potom stanica, pokiaľ ma správu k vysielaniu, môže začať vysielať. Metóda má v oblasti malých záťaži lepšie správanie ako metóda logického kruhu.

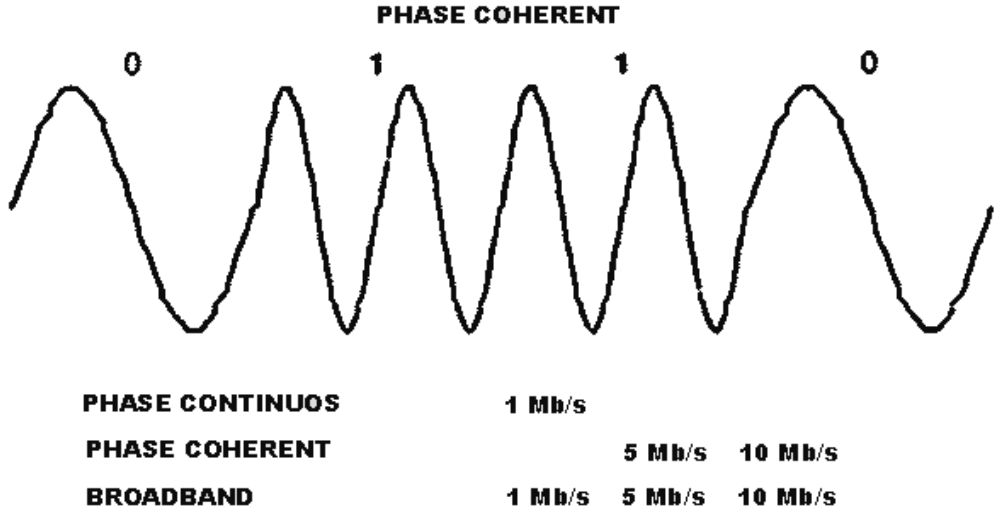
Metóda vyžaduje dobrú vzájomnú synchronizáciu staníc, ktorú je obtiažne spoľahlivo zaistiť. Ak však uvoľníme pravidlá pre prevzatie kanálu tak, že stanica smie zahájiť vysielanie do kanálu, ktorý bol po dobu N neobsadený, dostávame kanál s možnosťou kolízie - obdoba v ďalšej časti textu popisovanej metódy CSMA/CA.

MAP (IEEE 802.4)

Doporučenie ANSI/IEEE 802.4 definuje zbernicovú sieť s riadením typu logický kruh určenú pre aplikáciu v automatizovaných systémoch riadenia výroby (MAP-Manufacturing Automation Protocol firmy General Motors). Siete podľa doporučenia môžu vedľa prenosu v základnom pásme po optických vláknach vo hviezdicové topológii (prenosové rýchlosti 5,10 a 20 Mb/s) využívať i koaxiálny kábel o charakteristickej impedancii 75 ohmov (veľký výber typov) v pásme základnom i preloženom. Pre prenos v základnom pásme sa používa frekvenčná modulácia so spojitou zmenou fáze (Phase-Continuous FSK -1 Mb/s), a frekvenčná modulácia s koherentnou fázou (Phase-Coherent FSK -5 a 10 Mb/s). Pre prenos v preloženom pásmu je využívaná amplitúdovo-fázová modulácia (1, 5 a 10 Mb/s).

O doporučenie IEEE 802.4 sa opiera norma MAP (General Motors), ako vhodné sú uvádzane fázovo - koherentná frekvenčná modulácia 5 Mb/s a amplitúdovo-fázová moduláciou 10 Mb/s s rozdeleným pásmom.

Rovnako ako pre iné siete typu logický kruh je pre sieť IEEE 802.4 definovaný algoritmus predávania poverení a algoritmus rekonfigurácie (pridávanie a odoberanie staníc z logického kruhu).



Obr. 4.18 Modulácia v sieti MAP

Pre uvádzané algoritmy si každá stanica uchováva adresu svojho predchodcu a adresu svojho následníka. Stanica, ktorá príjme od svojho predchodcu poverenie sa stáva stanicou aktívnou a môže odoslať jeden paket. Po odoslaní paketu, alebo bezprostredne po prijatí poverenia (ak však nemá čo vysielať) predá aktívna stanica poverenie svojmu

následníkovi. Toto predanie musí prebehnúť do určeného časového limitu, inak je stanica považovaná za pokazenú a je naštartovaný algoritmus, ktorý ju z logického kruhu vyberie.

Začlenenie stanice do logického kruhu prebieha nasledovne. Každá aktívna stanica pred predaním poverenia periodicky vysiela výzvu, určenú staniciam s adresami ležiacimi v intervale medzi jej vlastnou adresou a adresou jej nasledovníka. Ak však na výzvu nedojde do časového limitu odpoveď, aktívna stanica predá riadenie následníkovi. Ak však na výzvu odpovie jediná stanica, je zaradená do logického kruhu a aktívna stanica jej predá riadenie ako svojmu následníkovi. Konečne, ak však na výzvu odpovie viac staníc, dochádza ku kolízii, aktívna stanica kolíziu rozpozná (chybná odpoveď) a spustí algoritmus vyhľadávania najbližšieho nasledovníka. Výber je obdobou binárneho vyhľadávania, v každom kroku sa berú do úvahy dva bity adresy, stanica využíva hodnotu týchto dvoch bitov k určeniu oneskorenia pre svoju odpoveď.

O vybratie z logického kruhu žiada aktívna stanica svojho predchodcu zvláštnou správou a predáva riadenie svojmu následníkovi.

Ak však stanica neodpovie na príjem poverenia vyslaním správy alebo poverenia do časového limitu (response window), je považovaná za pokazenú a jej predchodca vysiela otázku na jej nasledovníka. Ak sa však nasledovník ozve, je logicky kruh opäť naviazaný, v opačnom prípade stanica výživa ľubovoľnú stanicu a binárnym vyhľadávaním nájde najbližšieho nasledovníka.

Spustenie alebo znovuspustenie logického kruhu je realizovane vyslaním poverenia ľubovoľnou stanicou, ktorá príliš dlhú dobu detekuje kľud na médiu. Ak však stanica detekuje cudzie vysielanie na médiu, považuje ho za dôsledok viacnásobného výskytu poverenia v sieti a vzdáva sa riadenia (prechádza do neaktívneho stavu).

4.4.5 Náhodný prístup

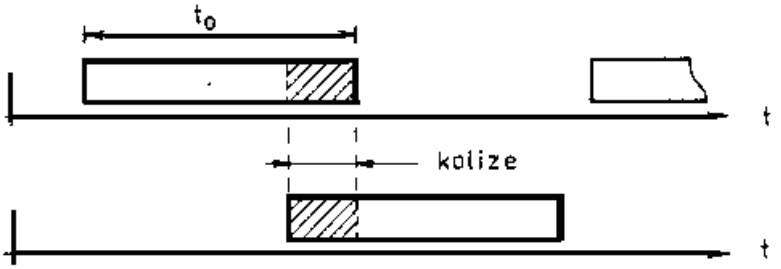
Náhodný prístup k zdieľaniu prenosového kanála môžeme považovať za protipól deterministických metód. Jednotlivé stanice podriaďujú prístup na kanál iba svojmu odhadu (prípadne pozorovaniu).

Metódy Aloha

Logickým predchodcom metód riadenia, ktoré používajú dnešné lokálne siete nasadzované v administratíve, sú metódy náhodného prístupu, ktoré boli vyvinuté pre komunikáciu na zdieľanom rádiovom kanále - metódy označované ako metódy Aloha.

Prosta Aloha

Najprimitívnejšou metódou náhodného prístupu je prosta Aloha, ktorá bola prvýkrát použitá v roku 1970 pre riadenie rádiovej siete na Havajskej univerzite. Stanica, ktorá ma paket pripravený k odoslaniu, začne vysielať bez ohľadu na prípadne obsadenie kanálu iným prenosom. Dôsledkom sú pochopiteľné kolízie, situáciu, v ktorej dochádza ku kolízii ilustruje obrázok Obr. 4.19:



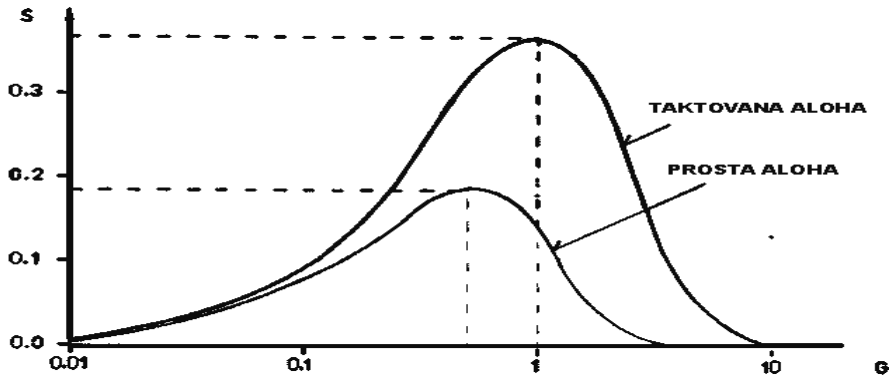
Obr. 4.19 Prostá Aloha – kolízia staníc

Pakety poškodené pri kolízii je nutné po uplynutí časového limitu, do ktorého mali byť potvrdené, opakovať, oneskorenie pred zahájením ďalšieho vysielania je volené náhodne, aby nedošlo k opakovaniu kolízie.

Ak budeme merať vstupný tok siete počtom paketov, ktoré majú byť prenesené, a tento tok označíme S, je zrejmé , že v ustálenom stave je tento tok rovný toku výstupnému (pakety prenesené sieťou). V dôsledku kolízie a z toho vyplývajúcej nutnosti opakovať poškodené pakety je celkový tok vnucovaný stanicami kanálu vyšší, označujeme ho G. Vzťah obidvoch tokov, prechádzajúceho S a celkového G sa dá (za predpokladu, že opakujúca stanica nesmie generovať nový paket) vyjadriť analyticky ako

$$S = G.e^{-2G} \qquad (4.4).$$

Priebeh tejto závislosti znázorňuje Obr. 4.20 a z grafu vyplýva, že aj u metódy prosta Aloha sa dá dosiahnuť využitie kapacity kanálu až 18.4%, pri dosiahnutí odpovedajúcej záťaže je každý paket v priemere trikrát vysielaný. Za povšimnutie stoji pokles priechodnosti pre rastúci celkový tok, tejto oblasti je nutné vyhýbať sa vhodným riadením.



Obr. 4.20 Charakteristiky metód Aloha

Taktovaná Aloha

Podstatného zvýšenia priechodnosti siete sa dá dosiahnuť jednoduchou modifikáciou metódy Aloha; staniciam dovolíme zahájiť vysielanie iba v okamihoch, ktoré definujú začiatky časových úsekov postačujúcich pre odoslanie jedného paketu.

Dôvodom zlepšenia, ktoré je zrejme z grafu na obr.3.18, je skrátenie tzv. kolízneho slotu, ktorého dĺžka odpovedala v prípade prostej Alohy dvojnásobku doby potrebnej pre odoslanie jedného paketu, na polovičku.

Výhodou metód Aloha je okamžité odvysielanie paketu. Ak prekročí však záťaž určitú hranicu, zvýši sa počet opakovaní paketov a silne poklesne pravdepodobnosť prenosu nepoškodeného paketu kolíziou. Sieť prechádza do tzv. zablokovaného stavu, z ktorého sa nedá bez modifikácie parametrov dostať. Metódy, ktoré nastavujú parameter opakovania v závislosti na záťaži, označujeme ako metódy riadené.

Riadená Aloha

Pakety, u ktorých došlo ku kolízii, sú u metód Aloha opakované po náhodne volenej dobe. Ak označíme strednú hodnotu tejto doby 1/a (o parametri a hovoríme ako o intenzite opakovania), dá sa voľbou parametru a dosiahnuť toho, že metóda Aloha pracuje vo výhodnejšej oblasti charakteristiky.

Pre voľbu parametra a existuje niekoľko heuristík, napríklad sledovanie priemerného počtu opakovania paketu alebo sledovanie podielu doby, počas ktorej je kanál obsadený. Najjednoduchšou a veľmi účinnou metódou je rada postupne klesajúcich hodnôt parametra a, ktoré stanica postupne používa pre výpočet odkladu ďalšieho opakovania.

Riadené opakovanie kolízii poškodených paketov ma podstatný význam nie len pre metódy Aloha, ale i pre metódy, ktoré uvádzame ďalej (metódy CSMA a ich modifikácie); bez riadenia sa nedá ani u týchto metód zaistiť trvalá efektívna činnosť.

Metódy CSMA

Metódy Aloha boli navrhnute pre rádiové siete a nevyužívali možnosti zistiť obsadenosť prenosového kanála pred zahájením vlastného vysielania. V lokálnych sieťach, ktoré sa vyznačujú malým oneskorením signálu a dokonalou počuteľnosťou staníc, je však takáto informácia užitočná a dovolí podstatne obmedziť pravdepodobnosť kolízie. Metódy, ktoré znalosť obsadenia kanálu využívajú, nazývame metódami náhodného prístupu s prízvukom nosné, skrátene metódami CSMA (Carier Sense Multiple Access).

Nenaliehajúca metóda CSMA

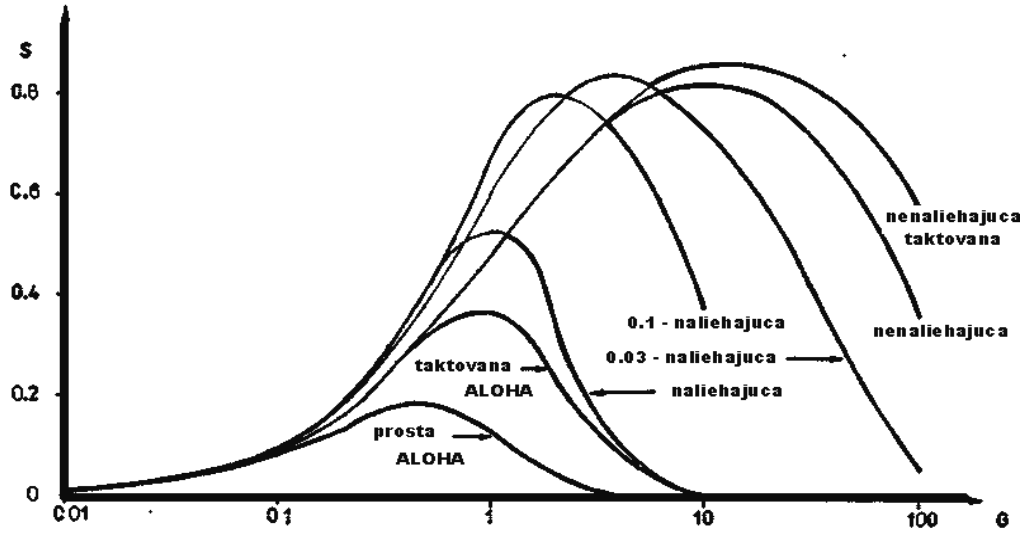
Stanica, ktorá používa *metódu nenaliehajúcej CSMA (non - persistent CSMA)*, pred odoslaním paketu testuje stav kanálu. Ak je kanál voľný, stanica začne vysielať. Ak však je kanál obsadený, stanica počká náhodne zvolený časový okamih a znovu testuje stav kanálu. Postup opakuje do odoslania paketu.

Voľbu náhodného časového okamihu obvykle prevádzame na voľbu náhodného násobku taktu, ktorý obvykle vyberáme tak, že odpovedá dobe prechodu signálu zbernicou. Závislosť prechodnosti kanálu na záťaži uvádza obrázok Obr. 4.21.

Naliehajúca metóda CSMA

Stanica, ktorá používa *metódu naliehajúcej CSMA (1-persistent CSMA)*, pred odoslaním paketu testuje stav kanálu. Ak je kanál obsadený, stanica odloží vysielanie na okamih jeho uvoľnenia.

Zjavnou nevýhodou tejto jednoduchej metódy je riziko kolízie staníc, ktoré čakajú na uvoľnenie kanálu. Toto pomerne vysoké riziko sa prejaví nižšou priechodnosťou kanálu (zhruba 53%, viď Obr. 4.21).

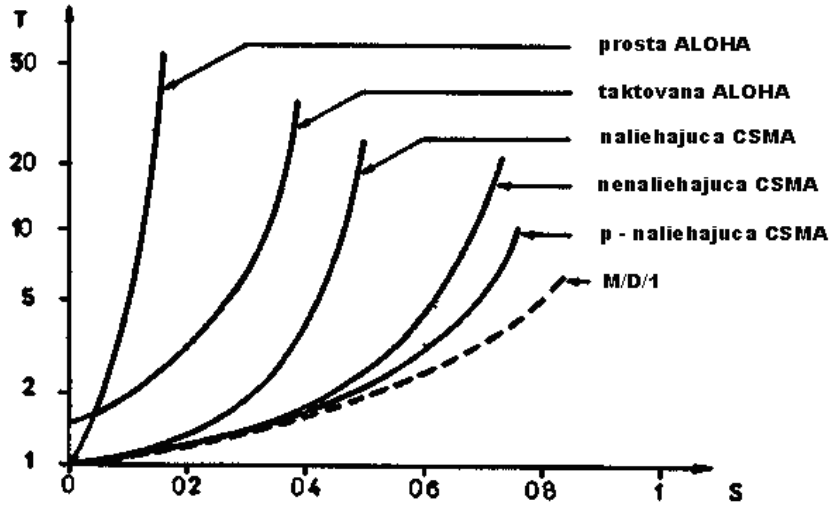


Obr. 4.21 Závislosť priepustnosti kanálu na záťaži

P- naliehajúca CSMA

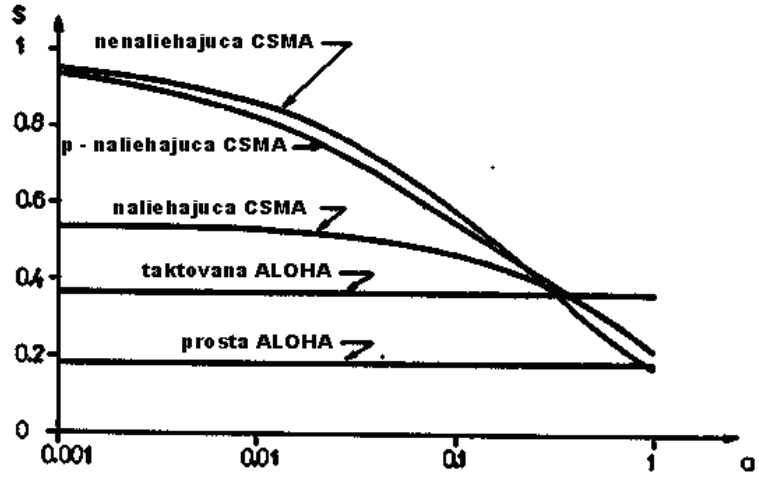
Stanica, ktorá používa *metódu p-naliehajúcej CSMA (p-persistent CSMA)*, pred odoslaním paketu testuje stav kanálu. Ak je kanál voľný, stanica začne vysielať. Ak však je kanál obsadený, stanica počká na uvoľnenie kanálu. Ak bol kanál voľný alebo sa pravé uvoľnil, začne stanica s pravdepodobnosťou p vysielať a s pravdepodobnosťou *q=1-p* odloží ďalšiu činnosť o krátky časový interval (môže odpovedať dĺžke šírenia signálu médiom). Po uplynutí tejto doby celu činnosť opakuje až do úspešného odoslania paketu.

Voľba parametra p dovolí optimálne nastaviť využitie kanálu a stredné oneskorenie paketu vzhľadom k záťaži. Pre p=1 metóda prechádza v naliehajúcu CSMA, pre p idúce k 0 sa síce priechodnosť kanálu blíži hodnote S=1, ale stredná doba prenosu paketu rastie nad všetky medze.



Obr. 4.22 Oneskorenie v metóde CSMA

Metódy CSMA samy o sebe nezaisťujú stabilitu riadenia. Pre udržanie kanálu v pracovnom stave je rovnako ako v prípade metód Aloha nutné použiť vhodnú metódu riadenia (napríklad meniť intenzitu opakovania alebo hodnotu p pri metóde p-naliehajúcej CSMA).



Obr. 4.23 Priepustnosť metód CSMA

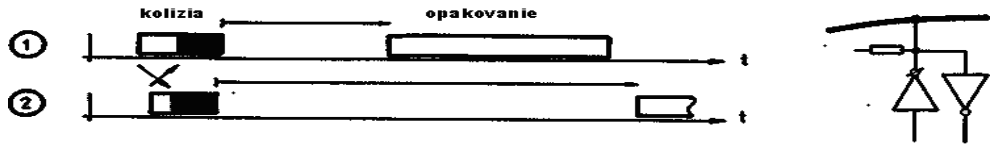
Metódy CSMA sú použiteľné iba v sieťach s malým rozsahom, v ktorých sa koeficient *a* pohybuje v medziach 0 je menšie ako *a* je menšie ako 0,1. Pre rozsiahlé lokálne siete efektivita metód klesá a pre hodnoty *a* idúce k 1 je dokonca horšia ako pri metódach Aloha.

V doteraz popisovaných metódach sme neuvažovali potrebu potvrdzovania prijatých paketov (presnejšie povedané, neuvažovali sme, že potvrdenia budú musieť súperiť o pridelenie kanálu). Na potvrdenie sa konečne môžeme pozerať ako na nutnú prídavnú záťaž, ktorá iba v určitom pomere zníži čistotu priechodnosti siete. Ak chceme tuto záťaž eliminovať, môžeme pre potvrdenie rezervovať časový interval bezprostredne naväzujúci na vysielaný paket a zaistiť, že žiadna zo staníc nesmie v tomto intervale zahájiť vysielanie nového dátového paketu. Takáto modifikácia býva označovaná ako CSMA/CA (Collission Avoidance) a je používaná v lacnejších, dnes už zriedkavejších sieťach (napr. Omninet).

Obdobou uvedenej metódy je modifikácia metódy CSMA, v ktorej povolíme stanici s adresou m zahájiť vysielanie paketu najskôr po dobe ((m-n) mod N).t po uvoľnení média stanicou s adresou n (N je celkový počet staníc siete a t je doba šírenia signálu médiom). Táto metóda je taktiež označovaná ako CSMA/CA, my sme ju poznali už skôr ako *virtuálny logický kruh (virtual token passing bus).*

Metódy CSMA/CD

Metódy CSMA nie sú schopné zabrániť kolízii, dokiaľ je časový interval medzi začatím vysielania dvoch staníc menší než určitá medza, daná konečnou rýchlosťou signálu v kanále, vzdialenosťou staníc a rýchlosťou reakcie detekčných obvodov. Dôsledkom nenulovej pravdepodobnosti kolízie a veľkej dĺžky paketu je zníženie priechodnosti kanálu.



Obr. 4.24 Riešenie kolízie v metóde CSMA/CD

Niektoré spôsoby realizácie zdieľaného kanálu dovoľujú rozpoznať, že došlo ku kolízii, ešte v dobe vysielania paketu, a vysielaný paket, ktorý je kolíziou poškodený, prerušiť. Najjednoduchším kanálom, ktorý detekciu kolízie umožňuje je zbernica typu otvorený kolektor, v praxi však obvykle kolíziu detekujeme inak (napr. sledovaním napätia na médiu, ktoré je budene prúdovými zdrojmi vysielačov). Aby sme zaistili, že kolíziu rozpoznajú všetky stanice ktoré spôsobili kolíziu, vyšle stanica po detekcii kolízie tzv. kolíznu postupnosť (jam). Detekcia kolízie zvyšuje využitie kanála vo všetkých metódach CSMA a môžeme hovoriť o naliehajúcej, p-naliehajúcej a nenaliehajúcej metóde CSMA/CD.

Poznamenajme, že v metóde CSMA/CD rovnako ako u všetkých metód CSMA musíme zaistiť stabilitu režimu prace.

Appletalk

Algoritmus metódy CSMA/CD nie je nutne viazaný na použitie detektoru kolízie v zapojení stanice. Príkladom je sieť Appletalk firmy Apple navrhnutá ako komunikačný prostriedok pre osobné počítače Apple a MacIntosh.

Funkciu špeciálneho detektoru kolízie v tejto sieti nahradzuje zvláštny spôsob rezervácie kanálu pre prenos správy. Stanica, ktorá chce získať neobsadený kanál (to zistí detektorom signálu média ako u metód CSMA), vyšle krátky paket adresátovi správy a vlastnú správu vysiela až po potvrdení tohto paketu. Ak však dôjde ku kolízii viac staníc

v tejto fáze, prejaví sa to poškodením žiadosti alebo odpovedi; stanica, ktorá neobdrží odpoveď do časového limitu, predpokladá kolíziu a odloží ďalší pokus o náhodne zvolený interval.

Zjednodušená metóda CSMA/Cd použitá v sieti Appletalk ma podobné vlastnosti ako základná CSMA/CD iba v prípade sietí s menšou prenosovou rýchlosťou a malým rozmerom. Sieť Appletalk používa prenosové rýchlosti 230.4kb/s na krútenej dvojlinke so signálmi podľa doporučenia RS- 422 EIA, dĺžka jedného segmentu zbernicovej siete je 300 m, do siete je možné pripojiť najviac 32 staníc.

Deterministické riešenie kolízie

Metóda CSMA/CD nie je posledným krokom v oblasti metód náhodného riadenia. Ďalšieho zlepšenia vlastnosti (zvýšenie priechodnosti a zníženie doby doručenia správy) dosahujú metódy, ktoré po zistení kolízie najskôr zaistia prenos správ pre stanice, ktoré sa kolízie zúčastnili, a až potom dovolí prístup staníc ostatných.

Najjednoduchšou metódou riešenia kolízie je metóda, ktorá sa opiera o vyhľadávanie aktívnych staníc metódou binárneho stromu. Ak dôjde ku kolízii v režime CSMA/CD, stanice, ktoré sa kolízie zúčastnili sa rozdelia do dvoch skupín (napríklad podľa najvýznamnejšieho bitu adresy). Stanica z prvej skupiny sa pokúsi o vyslanie správy, stanice druhej skupiny počkajú na ukončenie prenosu staníc v prvej skupine. Ak dôjde v prvej skupine opäť ku kolízii, celý postup delení skupiny sa opakuje (po konečnom počte krokov je v skupine jediná stanica)

Modifikáciou metódy, pri ktorej rozdelíme v každom kroku súperiace stanice na väčší počet skupín, môžeme dosiahnuť rýchlejšieho riešenia kolízie; krajným prípadom je rozdelenie staníc na skupiny o jednej stanici, ktorý pripomína deterministickú rezerváciu kanálu metódou round-robin. Takéto riadenie používa firma Intel pre komunikáciu medzi jednočipovymi mikropočítačmi i 80132 (prenosová rýchlosť je 2 Mb/s).

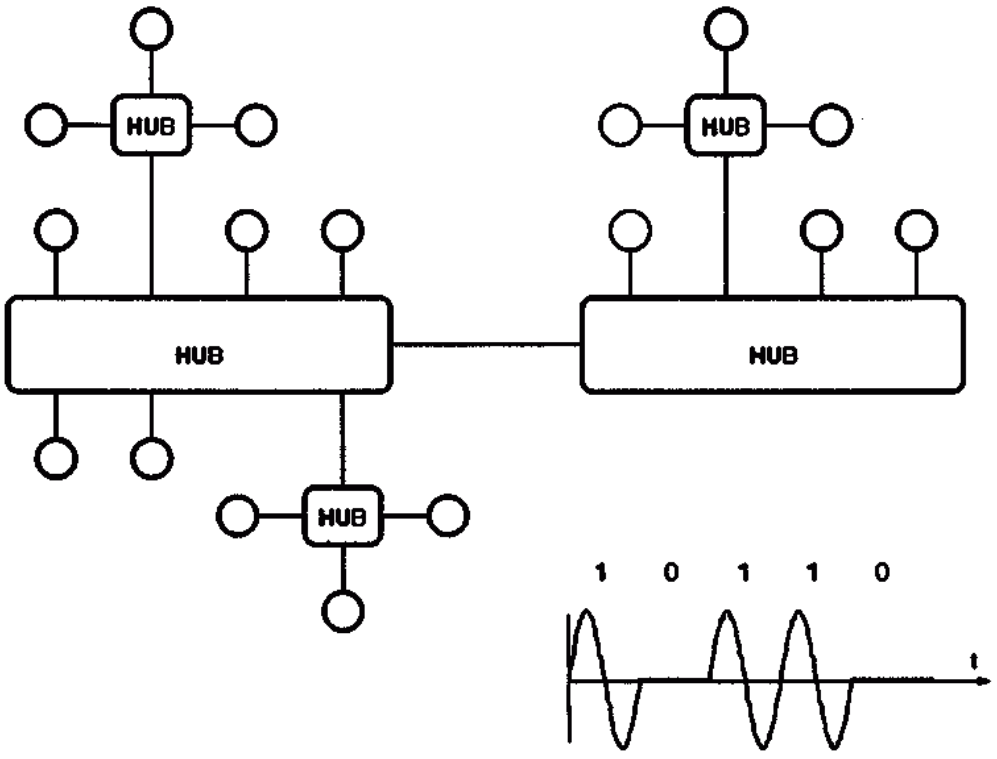
Prvý uvedený postup (binárne vyhľadávanie aktívnych staníc) je výhodnejší pre malé záťaže, druhý (postupne vyhľadávanie) pre záťaže veľké. Rada modifikácií metódy riešenia kolízie sa pokúša o nájdenie kompromisu medzi danými extrémami na základe informácií o okamžitom zaťažení siete.

5 Siete

Táto kapitola sa venuje vybratým sietiam ako ARCNET, Ethernet, široko pásmovým sietiam, kruhovým sietiam. Potom sa kapitola venuje štandardu sieťového rozhrania IEEE 802.2, architektúre internetu a prepájaniu lokálnych sieti.

5.1 ARCNET

Sieť ARCNet (Attached Resource Computer) bola vyvinutá firmou Datapoint v roku 1976 a rýchlo sa stala jednou z najrozšírenejších lokálnych sietí. Sieť ma stromovú topológiu (Obr. 5.1), stanice sú prepojené opakovačmi (active hub) úsekmi koaxiálneho kábla o charakteristickej impedancii 93 ohmov a maximálnej dĺžke 677 m, rovnaké obmedzenie káblov platí aj pre vzájomné prepojenie opakovačov. K jednému opakovaču sa dá pripojiť až 8 susedov (opakovačov alebo staníc), na ceste medzi dvoma stanicami smie byť najviac 9 opakovačov.



Obr. 5.1 Lokálna sieť ARCNET

Malý počet staníc, najviac štyri, sa dajú pripojiť pasívnym rozbočovačom (passive hub) do hviezdy, vzdialenosť medzi stanicami a rozbočovačom je najviac 30 m. Použitie rozbočovača v sieti s retranslátormi sa nedoporučuje. Niektorí výrobcovia dovoľujú aj zbernicovú štruktúru siete, použitie symetrických káblov (do 133 m) alebo optických vlákien (do 4 km).

Sieť ma prenosovú rýchlosť 2,5 Mb/s, k sieti sa dá pripojiť najviac 255 staníc, ktoré smú byť vzdialené najviac 6.5 km. Pre riadenie siete je použitá metóda logického kruhu a ďalej popísaná metóda rekonfigurácie.

V bežnej prevádzke stanica, ktorá príjme poverenie, odvysiela dátový paket (ak má nejaký pripravený) a predá riadenie následníkovi. Ten poverenie prevezme a do časového limitu, ktorý je daný dobou šírenia signálu v sieti sieť obsadí, buď prenosom dátového paketu, alebo predaním poverenia ďalšej stanici.

Ak vyprší uvedený časový limit, ktorý svojou hodnotou 31 mikro sekúnd odpovedá najzložitejšej konfigurácii siete, považuje stanica svojho nasledovníka za neaktívneho. V takom prípade stanica použije najbližšiu vyššiu adresu a pokúsi sa nájsť ďalšieho možného nasledovníka. To opakuje, až sa podarí logicky kruh opäť naviazať.

Výpadok aktívneho držiteľa poverenia vyvolá kľud na médiu po ešte dlhšiu dobu. Ľubovoľná stanica, ktorá indikuje kľud po dobu dlhšiu než 78 mikro sekúnd zahájí algoritmus výberu nového držiteľa poverenia. Ak však sa do doby *147.(255-adresa_stanice)* mikrosekund logický kruh neobnoví, stáva sa stanica aktívnym držiteľom poverenia a obnovuje sama funkciu logického kruhu.

Ak sa však dosiaľ neaktívna stanica chce zapojiť do logického kruhu, počká 840 ms na poverenie (ktoré pravdepodobne neobdrží) a potom postupnosť 756 jednotkových bitov naruší funkciu kruhu a vyžiada si tak znovuspustenie podľa predchádzajúceho odstavca.

V roku 1990 bola na trh uvedená modifikácia siete Advanced ARCNet, ktorá použitím šestnásťstavovej fázovo-amplitúdovej modulácie dosahuje rýchlosť prenosu 20 Mb/s.

5.2 Ethernet

Lokálna sieť Ethernet so zbernicovou architektúrou bola vyvinutá v polovici 70-tych rokov firmou Xerox a neskôr bola štandardizovaná firmami Xerox, Intel a DEC pre siete v administratíve (je označovaná ako norma DIX).

Prenosovým médiom pôvodnej siete Ethernet je koaxiálny kábel o charakteristickej impedancii 50 ohmov, výhodou kábla s nižšou charakteristickou impedanciou než je bežnejších 75 ohm je vyššia odolnosť proti parazitným kapacitám konektorov a proti vplyvu vonkajšieho rušenia. Dáta sú prenášané v základnom pásme v kóde Manchester, rýchlosť prenosu je 10 Mb/s.

Základom siete je *segment* - zbernica o dĺžke najviac 500m na ktorú sa dá pripojiť až 100 staníc. Rozsiahlejšie siete sa dajú vytvoriť prepojovaním segmentov pomocou retranslátorov (opakovačov) - limitom je 1024 staníc a vzdialenosť medzi najvzdialenejšími stanicami (merané po médiu) 2,5 km.

Stanica je k segmentu pripojená prostredníctvom tranceiveru (kombinácia vysielača a prijímača signálu média), ktorý je pripevnený priamo na kábel. Tranceiver je spojený so stanicou päťnásobnou točenou dvojlinkou na vzdialenosť až 50m. Rozhranie je označované ako AUI (Attachment Unit Interface).

Riadenie siete odpovedá metóde CSMA/CD. Stanica, ktorá behom vysielania zistí kolíziu na médiu preruší vysielanie paketu a odošle špeciálnu postupnosť (jam). Táto postupnosť je navrhnutá tak, aby vyvolala indikáciu kolízie aj u ostatných staníc. Výsledkom je uvoľnenie média všetkými stanicami najneskôr do doby odpovedajúcemu súčtu dvojnásobku doby šírenia signálu sieťou a doby vysielania kolíznej postupnosti. Tento súčet sa označuje ako *kolízny slot*.

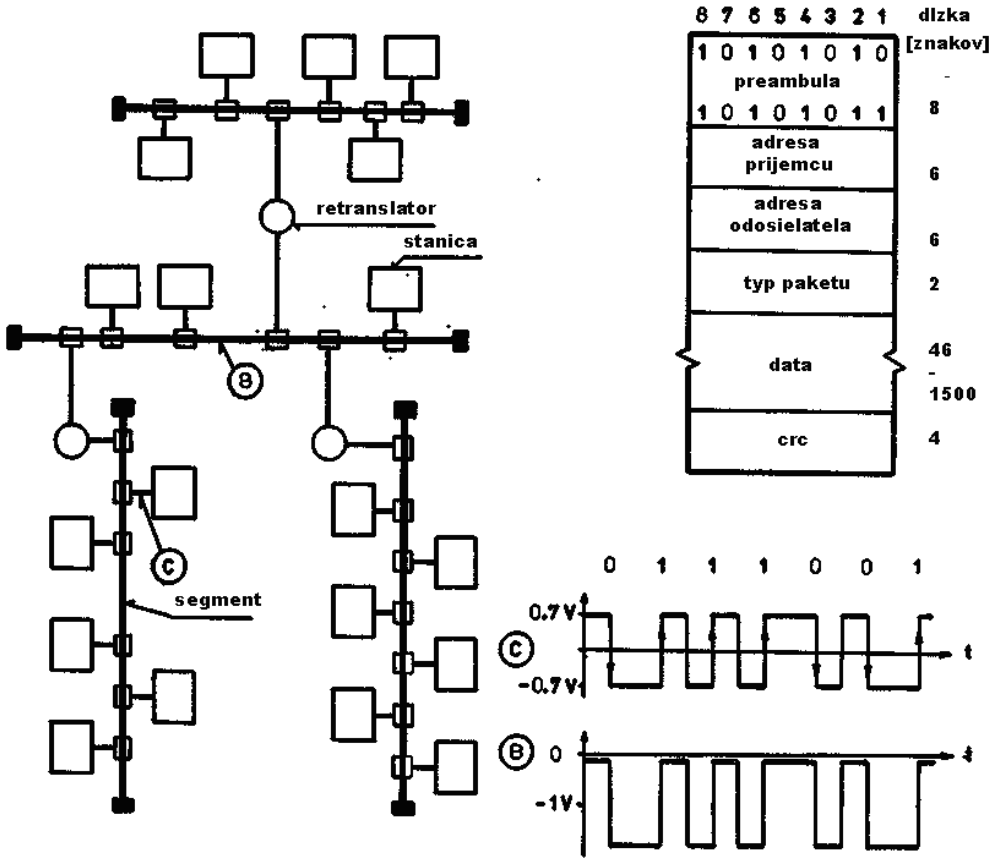
Zaujímavé je riadenie intenzity opakovania. Pri zistení kolízie je ďalší pokus plánovaný na r-tý kolízny slot, kde r je náhodne zvolené číslo z intervalu 0 je mensie r je

mensie pripadne = 2k ; exponent k je odvodený z počtu neúspešných pokusov o odoslanie paketu n, k=min (n,10). Táto metóda riadenia je označovaná ako "*exponential back- off*".

Štruktúra rámca odpovedá obrázok Obr. 5.2. Adresa stanice má dĺžku 48 bitov a je pre každý radič jedinečná. Dátová časť rámca ma dĺžku 46 až 1500 znakov, dĺžka najkratšieho rámca odpovedá dĺžke kolízneho slotu (512 bitov).

Za upozornenie stojí, že formát rámca podľa normy DIX sa líši od formátu rámca podľa IEEE 802.3. Zatiaľ čo IEEE Ethernet uvádza v hlavičke dĺžku LCC bloku, DIX Ethernet tu uvádza identifikáciu sieťového protokolu.

Lokálna sieť Ethernet dovoľuje využiť kapacitu média na 80 až 95% (podľa dĺžky správ) pri záťaži väčšej ako 50% však silno rastie doba prenosu. Podstatnou nevýhodou siete Ethernet je použitie drahého kvalitného koaxiálneho kábla, ktorý je nutný pre spoľahlivú funkciu detektora kolízie.



Obr. 5.2 Lokálna sieť Ethernet

5.2.1 IEEE 802.3

Zbenicová sieť odpovedajúca doporučeniu IEEE 802.3 je normalizovanou verziou siete Ethernet. Norma definuje viac verzií siete líšiacej sa použitým médiom, rozmerom siete a prenosovou rýchlosťou, tieto verzie sú označované ako Ethernet, Cheapernet a Starlan.

Ethernet(10BASE) - Thick Ethernet používa ako médium špeciálny koaxiálny kábel o impedancii 50 ohm a priemeru 10 mm, dĺžka segmentu môže byť až 500 m a dá sa naňho pripojiť najviac 100 staníc vzdialených od seba najmenej 2,5 m. Stanice sú rovnako ako u DIX Ethernetu pripojené pomocou tranceiveru. Medzi dvoma stanicami na rôznych segmentoch môžu byť najviac 4 retranslátory. Prenosová rýchlosť je 10 Mb/s. Od normy DIX sa 10BASE5 líši formátom rámca.

Cheapernet(10BASE2) - Thin Ethernet používa ako médium lacný tenký koaxiálny kábel o impedancii 50 ohm a priemere 5 mm. Proti silnému káblu odpadá oddelený tranceiver, koaxiálny kábel je pripojený konektorom BNC (presnejšie T - rozbočkou) priamo na vstavaný tranceiver. Dĺžka segmentu je obmedzená na 185 m (niektorí výrobcovia dovoľujú až 450 m za predpokladu osadenia segmentu iba ich stanicami s presnejšie definovanými medzami indikátoru kolízie) a dá sa na neho pripojiť najviac 30 staníc vzdialených od seba najmenej 0.5 m. Medzi dvoma stanicami na rôznych segmentoch môžu byť najviac 4 retranslátory. Prenosová rýchlosť je 10 Mb/s.

Starlan(1BASE5) - používala ako médium dvojicu netienených točených dvojliniek, stanice sú na rozdiel od sietí Ethernet a Cheapernet pripojené hviezdicovo ku koncentrátoru. Prípojné vedenie má dĺžku do 250 m, rozsiahlejšie siete sa dajú vytvárať prepojením koncentrátorov hviezdicovo medzi sebou. Limitom je päť úrovní (najväčšia vzdialenosť medzi stanicami je 2500m). Nevýhodou je nízka prenosová rýchlosť 1 Mb/s a tým aj obtiažnosť kombinácie s inými variantmi Ethernetu.

Twisted Pair Ethernet(10BASE-T) - je nová varianta Ethernetu, má podobne ako Starlan hviezdicovú topológiu a opiera sa taktiež o dvojitú netienenú točenú dvojlinku (UTP). Prenosová rýchlosť je však štandardných 10Mb/s, vzdialenosť medzi stanicou a koncentrátorom (opakovačom) je však iba 100m. To obmedzuje najvyššiu vzdialenosť v sieti 10BASE-T na 500 m (pre najviac 4 opakovače medzi stanicami). Opakovače dovoľujú obvykle 8 až 16 staníc naviac a sú vybavené štandardným rozhraním tranceiveru prípadne rozhraním segmentu 10BASE2. Opakovače sa dajú teda prepojiť so zbernicovými segmentmi 10BASE5 a 10BASE2 alebo optickými vláknami na väčšiu vzdialenosť (do 1000m).

Širokopásmový Ethernet(10BROAD36) - je určený pre priemyslové prostredie a používa ako médium koaxiálny kábel s impedanciou 75 ohm pre káblovú TV. Modemy používajú diferenciálnu fázovú moduláciu, dátový signál NRZ je skramblovaný. Výsledné frekvenčné pásmo má šírku 14 MHZ. Detekcia kolízie sa opiera o počutie vlastného vysielania v spätnom kábli alebo pásme, kolízia je ostatným staniciam indikovaná po vyhradenom kanáli o šírke 4 MHZ. Výhodou technológie je dĺžka segmentu obmedzená až maximálnou vzdialenosťou stanice od koreňa (1800m).

Označenie technológií, ktoré sme uvádzali v zátvorkách odpovedajú konvencii noriem IEEE 802. Prvé číslo udáva prenosovú rýchlosť v Mb/s. Kód BASE hovorí, že sa jedná o sieť pracujúcu v základnom pásme, kód BROAD, že sa jedná o sieť širokopásmovú. Posledné číslo udáva dĺžku segmentu v stovkách metrov.

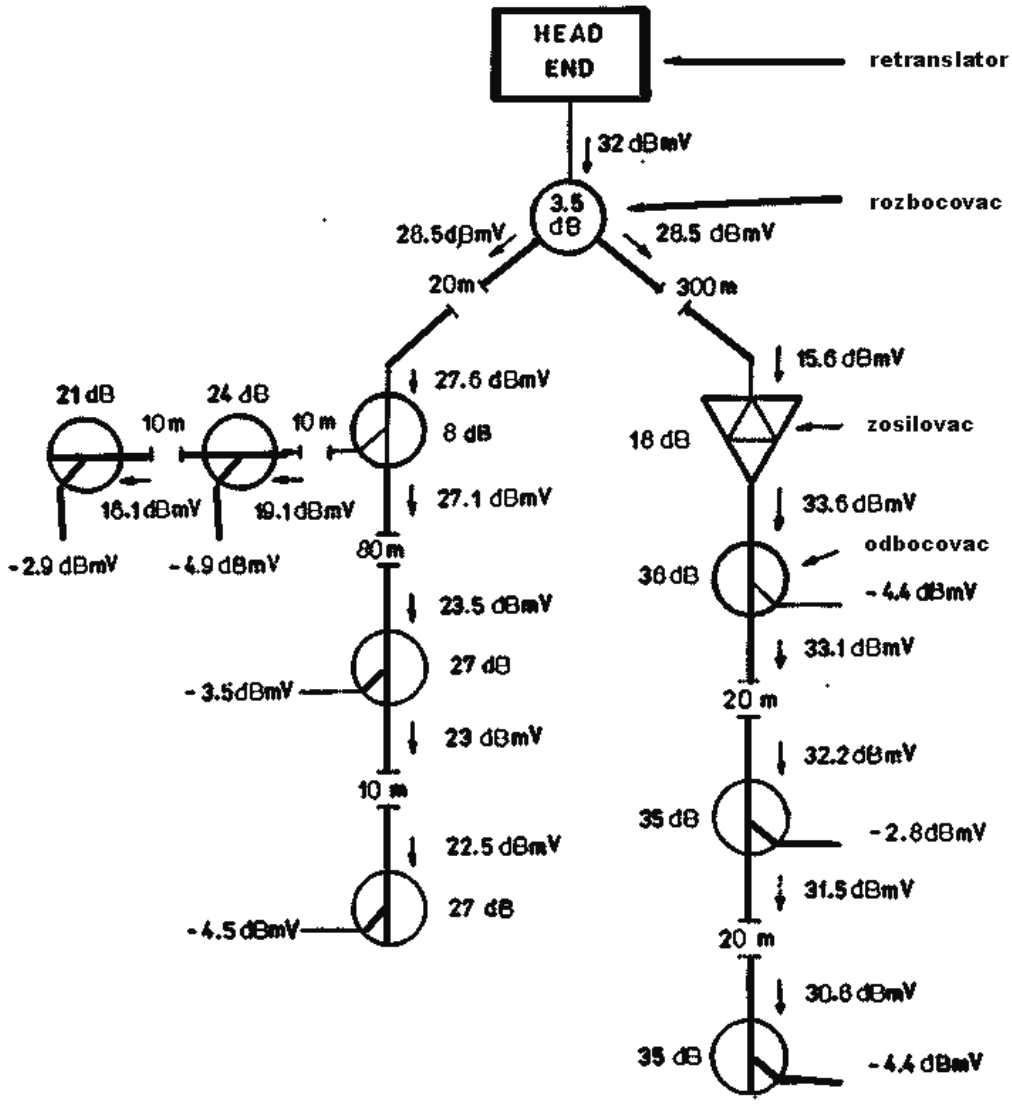
Optický Ethernet(10BASE-F) - dovoľuje v ostávajúcich sieťach prepojiť (a elektricky izolovať) prostredníctvom retranslátorov vzdialenejšie segmenty (bežne na 1 km). S využitím univerzálnych viacvstupových retranslátorov sa dajú ľahko realizovať optické siete s hviezdicovou topológiou.

5.3 Širokopásmové siete

Zaujímavou skupinou zbernicových lokálnych sietí sú siete využívajúce prenos v preloženom pásme. Toto pásmo je v prípade koaxialného kábla dostatočne široké, aby ho bolo možné rozdeliť na viac kanálov frekvenčného multiplexu.

5.3.1 Prenosové médium

Prenosovým médiom širokopásmových sietí je pravidelne koaxiálny kábel o priemere pol palca s charakteristickou impedanciou 75 ohm používaný pre rozvody káblovej televízie. Jeho výhodou je postačujúca kvalita a podstatne nižšia cena než je cena káblov sietí pracujúcich v základnom pásme (Ethernet). Súčasne sa dá využiť celá škála prvkov používaných pre inštaláciu káblovej televízie- rozbočovače, odbočovače a pásmové zosilňovače.



Obr. 5.3 Širokopásmová sieť

Logickou štruktúrou širokopásmových sietí je dvojica kanálov. Na jeden z nich sú pripojené vysielače staníc, na druhý sú pripojené prijímače. Oba kanály širokopásmovej siete sú prepojené v jedinom mieste zosilňovačom alebo retranslátorom. Zosilňovač

je používaný u sietí, ktoré pre vysielací a prijímací kanál používajú samostatné káble (Wangnet). Retranslátor používajú siete s jediným káblom pre prenos obidvoch kanálov v rôznych frekvenčných pásmach (Localnet,3MLan,IBM PCNet); retranslátor prevádza signály z pásma kanálu vysielacieho do pásma kanálu prijímacieho.

Rozbočovače (splitters) dovoľujú rozvetviť sieť, do všetkých vetiev vkladajú rovnaký útlm (obvykle 3.5 dB pre dvojcestný rozbočovač). Odbočovače (directional couplers) majú prechodný útlm oveľa menší ako (okolo 0.5 dB) útlm odbočky je voliteľný v rozsahu 10 až 40 dB. Odbočovače slúžiace k pripojeniu staníc k médiu sú označované ako „taps“ a bývajú často viacnásobné.

U rozsiahlejších sietí je nutné útlm káblov, rozbočovačov a odbočovačov kryť zosilnením linkových zosilňovačov so zosilnením v rozsahy 10 až 40 dB, rovnakú veľkosť máva aj zosilnenie retranslátorov. Dôsledkom nutnosti rešpektovať útlmy u širokopásmových sietí je nutnosť výpočtov káblových rozvodov pre konkrétne rozmiestnenie pracovných staníc. Príklad širokopásmovej siete je na obrázku Obr. 5.3.

5.3.2 Metódy riadenia

V širokopásmových lokálnych sieťach sú využívané všetky metódy riadenia, ktoré už poznáme. Metódy náhodného prístupu CSMA/CD sú použiteľné pre menšie lokálne siete, u väčších sietí z princípu klesá ich efektivita. Dôvodom je jednak dlhšia doba šírenia signálu medzi stanicami - signál je prenášaný cez retranslátor, jednako nižšia účinnosť detektora kolízie, ktorý musí pracovať na iných princípoch ako u sietí s prenosom v základnom pásme. Častejšie sa stretávame s deterministickým riadením (token-passing-bus), širokopásmová sieť s deterministickým riadením je základom doporučenia IEEE 802.4.

Širokopásmové siete sú vhodné pre aplikácie, na ktoré sú kladené väčšie požiadavky. Majú väčšiu prenosovú kapacitu, zaisťujú väčšiu odolnosť proti vonkajšiemu rušeniu a dovoľujú využiť prenosové médium pre ďalšie prídavné služby (telefón, TV signál). Prvky káblových rozvodov majú vysokú spoľahlivosť, sú overené z káblovej televízie a ľahko dostupné. Nevýhodou je vyššia zložitosť komunikačných staníc, ktoré obsahujú výrobne náročný modem.

5.3.3 Localnet

Jedna z najrozšírenejších lokálnych sietí Localnet firmy Sytek sa opiera o technológiu káblovej televízie (káble, odbočovače, rozbočovače). Využíva frekvenčného pásma 40 - 106 MHz pre vysielanie a pásma 196 - 262 MHz pre príjem. Signály pásma 40 - 106 MHz prevádza do pásma 196 - 262 MHz restranslátor (tverter) umiestnený v koreni stromovej siete. Sú dodávané dve vzájomne zlúčiteľné varianty siete označené ako Systém 20 a Systém 40.

Localnet Systém 20 využíva úseky 70 - 106 Mhz a 226 - 262 Mhz rozdelených do 120 kanálov frekvenčného multiplexu o šírke 300 kHz. Jednotlivé kanály sa dajú využiť ako zbernicové kanály s riadením CSMA/CD prenosovou rýchlosťou 2 Mb/s a vzdialenosťou až 6 km.

Localnet Systém 40 sa stal základom lokálnej IBM PCNet.Sieť Localnet je rozsiahlý systém, ktorý zahrňuje radu špeciálnych zariadení pre napojenie lokálnej siete, verejnej dátovej siete ap.

Technológia Localnet bola použitá firmou IBM pre prepojenie personálnych počítačov IBM PC a dodávaná pod označením IBM PCNet. Riadenie siete odpovedalo metóde CSMA/CD, prenosová rýchlosť je 2 Mb/s. Dáta v kóde NRZI sú vo vysielači stanice frekvenčne modulované na frekvenciu 50.75 MHz, prijímací kanál ma strednú frekvenciu

219 MHz. Pre ovládanie komunikačných staníc bol vytvorený programový ovládač známy ako NetBIOS.

5.3.4 Wangnet

Sieť Wangnet ma stromovú topológiu, pre prenos je využívaná dvojica koaxiálnych káblov - na jeden sú pripojené vysielače staníc, na druhý prijímače. Káble, rozbočovače, odbočovače a zosilňovače odpovedajú bežnej káblovej televízii.

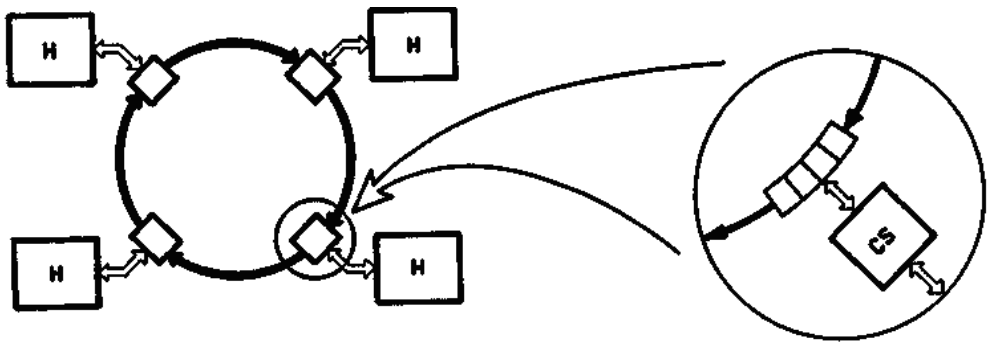
Pre prenos je využívané pásmo o šírke 340 MHz (10-350 Mhz) rozdelené do troch častí. Najdôležitejšou časťou spektra je Wangband - vytvára zbernicový kanál CSMA/CD s prenosovou rýchlosťou 12 Mb/s. Frekvencie medzi 10 a 82 MHz sú využité pre pomalé synchrónne a asynchrónne kanály. Prvá časť tohto pásma dovoľuje vytvoriť 32 pevných dvojbodových alebo viacbodových kanálov s rýchlosťou prenosu 9.6 kb/s, druhá časť 16 pevných dvojbodových alebo viacbodových kanálov s rýchlosťou prenosu do 64 kb/s. Kanály v tretej časti pásma sú prideľované na žiadosť, pre ich využitie je nutný preladiteľný modem (frequency agile modem). Do poslednej časti pásma (utility band) na frekvencii 174 - 216 MHz sa dá umiestniť až sedem televíznych kanálov využiteľných napríklad pre telekonferenciu alebo bezpečnostný systém.

5.4 Kruhové siete

Kruhové siete tvoria zvláštnu kategóriu lokálnych sietí, kruhová sieť je tvorená stanicami, ktoré sú vzájomne prepojené jednosmernými dvojdobými spojmi (Obr. 5.4).

Kruhové stanice obsahujú posuvný register(o dĺžke aspoň jedného bitu) celú sieť si môžeme predstaviť ako kanál, v ktorom sa signály šíria od vysielajúcej stanice jedným smerom a po prechode sieťou sa k nej opäť vracajú. Doba, ktorú signál k priechodu sieťou potrebuje je daná počtom komunikačných staníc a dĺžkou ich registrov a u sietí s vysokou prenosovou rýchlosťou a dlhými spojmi taktiež dobou prenosu signálu vlastným médiom.

Kruhové siete majú mnoho výhodných vlastností. Dovoľujú nasadenie metód distribuovaného riadenia prístupu a to aj v prípadoch keď stanice sú veľmi vzdialené (desiatky km).Také metódy zaisťujú ohraničenú dobu oneskorenia a vysoké využitie kapacity kanálu. Spoje sa dajú ľahko realizovať ako svetlovody, sieť je potom veľmi odolná voči vonkajšiemu rušeniu. Jedinou vážnou nevýhodou je ich závislosť na správnej činnosti všetkých komponentov, výpadok akéhokoľvek uzla alebo spoja preruší komunikačný kanál.

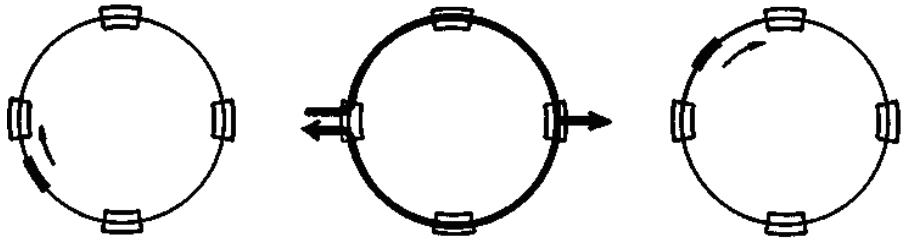


Obr. 5.4 Štruktúra kruhovej siete

Pre kruhové siete sa v praxi využívajú tri základné metódy riadenia, podľa použitej metódy riadenia hovoríme o sieťach Newhallovho typu, sieťach Piercovho typu a o sieťach s vkladaním rámcov.

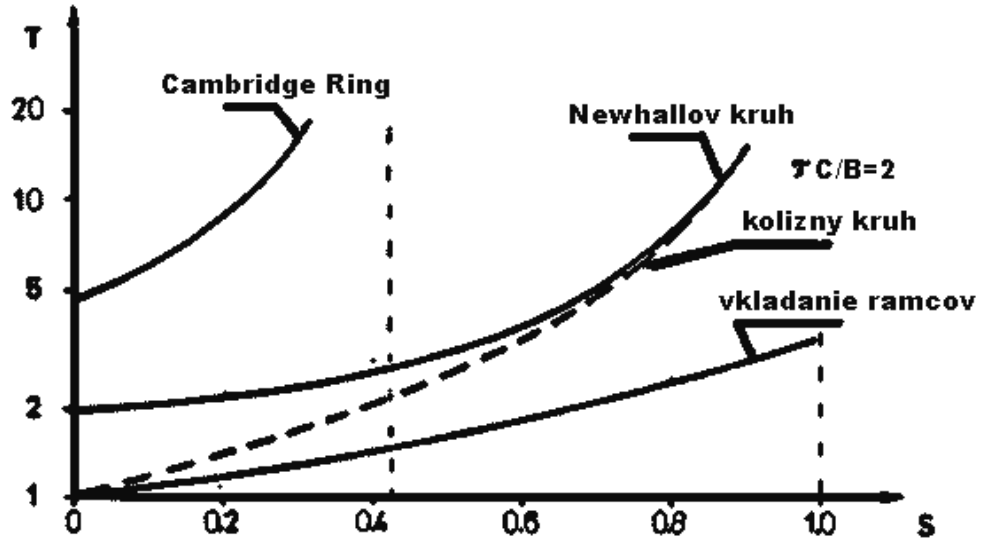
5.4.1 NewHallov kruh (Token Ring)

V sieti Newhallovho typu obieha v kľudovom stave keď žiadna stanica neprenáša správu iba takzvané poverenie (token). Stanica, ktorá má správu k vyslaniu a získa poverenie, môže správu do kruhu vyslať. Vyslanie správy spočíva v zmene poverenia na znak (postupnosť znakov) identifikujúci počiatok paketu a v odoslaní vlastného paketu. Po odvysielaní správy odovzdá stanica riadenie vyslaním poverenia svojmu susedovi. Prenos jedného paketu kruhovým spojom ilustruje obrázok Obr. 5.5:



Obr. 5.5 Prenos paketu Newhallovym kruhom

Oneskorenie paketu v sieti je pre malú záťaž dané dobou, ktorú musí paket čakať na obiehajúce poverenie a dobou vlastného prenosu. Doba obehu poverenia závisí na rýchlosti prenosu, počte staníc a dĺžke ich registrov, taktiež na dĺžke prepojovacích spojov. Pre veľké záťaže sa oneskorenie v sieti blíži ideálnemu prideľovaniu kanála, závislosť oneskorenia na počte staníc a dĺžke registrov uvádza obrázok Obr. 5.6:



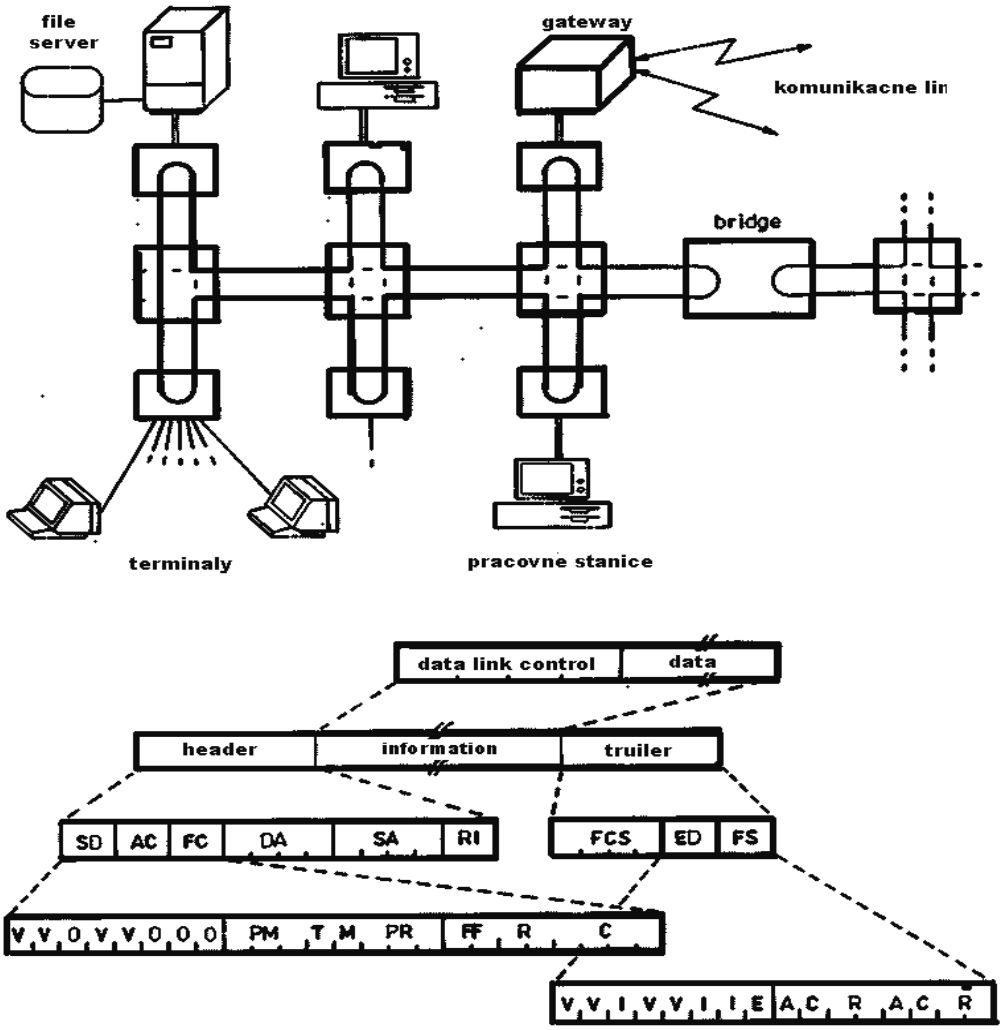
Obr. 5.6 Oneskorenie v kruhovej sieti

Správna funkcia NewHallovej siete je závislá na obiehaní jedného poverenia. Ak nechceme odštartovaním siete alebo jej znovuspustením po strate poverenia poveriť jedinú stanicu, môžeme ľahko realizovať distribuovaný algoritmus.

Sieť Newhallovho typu je pre svoje chovanie často používaná ako komunikačný prostriedok distribuovaných riadiacich prostriedkov. Na princípe siete Newhallovho typu je taktiež založená sieť IBM Token Ring.

IBM Token Ring (IEEE 802.5)

Kruhová sieť navrhnutá firmou IBM a odpovedajúca doporučeniu IEEE 802.5 má rad zaujímavých vlastností.



Obr. 5.7 Lokálna sieť IBM Token Ring

Jej elektrickou topológiou je kruh, spoje medzi jednotlivými stanicami sú však vedené takým spôsobom, že vytvárajú hviezdicovú sieť. Táto topológia je niekedy

označovaná ako laloková. Koncentrátory v uzloch siete dovoľujú odpojiť jednotlivé laloky alebo aj celé časti siete v prípade poruchy spoja alebo v prípade poruchy stanice.

Spoje medzi stanicami a koncentrátormi a spoje medzi koncentrátormi sú tvorené symetrickými káblami alebo svetlovodmi. Symetrickým káblom sa dajú prepojiť dve stanice na vzdialenosť až 770m pri väčšej prekonávanej vzdialenosti je nutné vyradiť do vedenia opakovač. Svetlovodné úseky siete môžu byť dlhé až 2 km, prevod medzi elektrickými signálmi staníc a koncentrátorov a optickým káblom zaisťujú optické opakovače.

Dáta sú prenášané v rámcoch odpovedajúcich obrázku. Informačné pole rámca má premenlivú dĺžku, hlavička rámca obsahuje riadiace pole AC, FC a adresy adresáta a odosielateľa. Pole AC riadi prístup staníc k prenosovému médiu, pole FC je riadiace pole rámca.

Jednotlivé bity poľa AC majú nasledujúci význam:

- *T*: odlišuje dátový rámec od poverenia (token),
- *PM*,*PR*: riadi prioritný prístup ku kanálu,
- *M*: slúži ako ochrana pred paketmi s poškodenou adresou.

Pole FC prenáša riadiace indikátory, Bity FF odlišujú riadiace rámce (FF=O1H) od rámcov informačných /FF=OOH).

Obsah rámca je chránený 32 bitovým cyklickým kódom. Ako počiatočný (SD) a koncový obmedzovač rámca (ED) slúžia špeciálne znaky, ktoré porušujú pravidlá kódovania typu DPSK (diferenciálny Manchester) použitého pre prenos káblom. Bity, v ktorých sú pravidlá porušené sú označené ako V. Bit E ukončujúceho obmedzovača môže nastaviť na hodnotu E=1 ľubovoľná stanica kruhu, ktorá rozpozná v prenášanom rámci chybu. Za ukončujúcim obmedzovačom je predávaný znak, v ktorom prijímač potvrdzuje rozpoznanie svojej adresy (bit A) a prevzatie správy (bit C). Táto časť rámca nie je cyklickým kódom, príznaky A a C sú zaistené opakovaním.

Poverenie (token) má dĺžku troch znakov a obsahuje iba počiatočný (SD) a koncový (ED) oddeľovač a pole riadenia prístupu (AC).

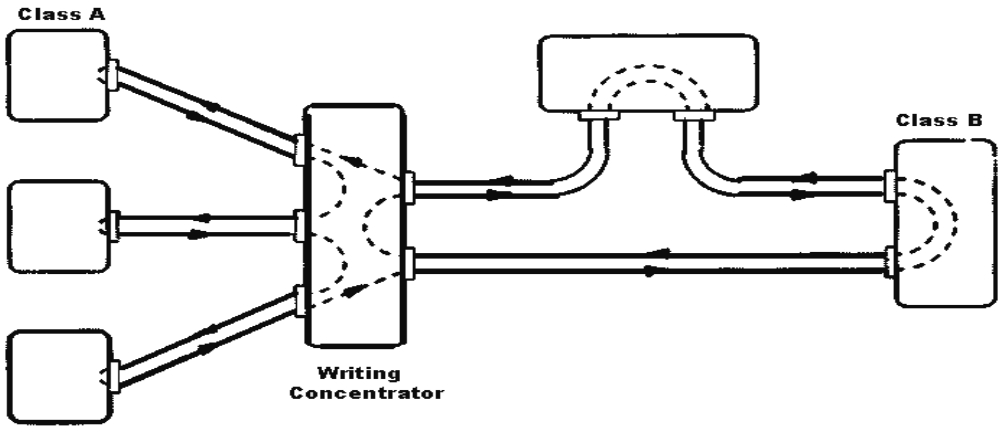
Sieť FDDI

Kruhová sieť FDDI (Fiber Distributed Data Interface) využíva pre prepojenie staníc optické káble. Prenosová rýchlosť je 100 Mb/s. Pre kódovanie dát je použitý efektívny kód 4 z 5, ktorý každé štyri bity dát transformuje na 5 bitov, čím zaistí okrem dobrej bitovej synchronizácie aj potrebné riadiace kombinácie.

Vzdialenosť medzi stanicami siete FDDI, ktorých môže byť v kruhu až tisíc je 2 km (bez použitia opakovačov). To dovoľuje postaviť sieť s celkovou dĺžkou až 200 km. Pre svoju vysokú kapacitu je sieť vhodná obzvlášť pre vytváranie rozsiahlých kostrových sietí prepojujúcich pomalšie lokálne siete a pre prepojovanie veľkých počítačov.

Základná konfigurácia siete (trieda A) je dvojitý kruh, jednotlivé kruhy majú opačný smer prenosu (a každý prenosovú rýchlosť 100 Mb/s). V bežnej prevádzke slúži primárny kruh pre prenos dát a sekundárny kruh pre riadenie a kontrolu funkcie. Pri detekovanom prerušení kruhu (napríklad prerušením alebo poruchou niektorej zo staníc) stanica susediaca s poruchou automaticky rekonfiguruje kruh a využije sekundárny kruh pre prenos dát. Pre menej náročné nasadenia sa dá použiť zjednodušená sieť (trieda B) s jediným kruhom, táto sieť však nie je schopná rekonfigurácie. Typickou štruktúrou siete FDDI je dvojitá kostra triedy A prepojujúca menšie siete triedy B.

Riadenie prenosu v sieti FDDI sa podobá riadeniu v sieti IBM Token Ring s tým rozdielom, že sieť je schopná prenášať rámce viacerých staníc súčasne.



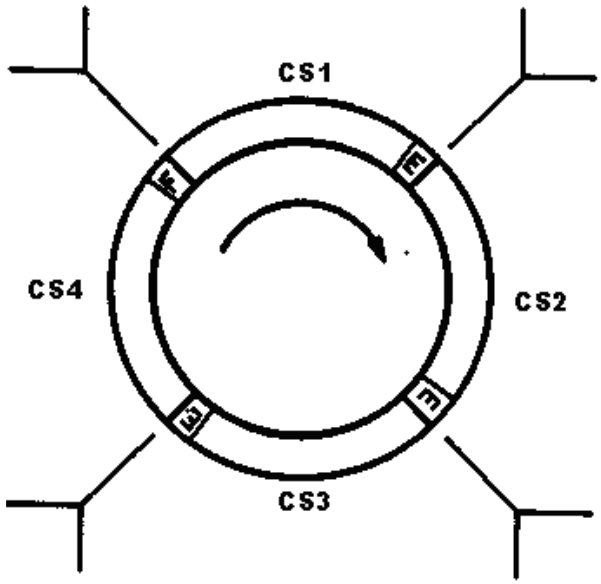
Obr. 5.8 Sieť FDDI

5.4.2 Pierceov kruh

Rozdelením pamäťovej kapacity kruhovej siete na krátke segmenty - minipakety dostávame sieť Pierceovho typu.

Multipakety prenášajú jediné šestnásťbitové slovo. Obsadenie minipaketu je indikované nastavením bitu E/F stanice ktorá ma dáta k vysielaniu a voľný minipaket vo svojom registri, minipaket obsadí a po jeho obehu sieťou (a potvrdení adresátom) ho opäť uvoľní.

Problémom siete je likvidácia minipaketov s poškodenou adresou odosielateľa, počiatočným naformátovaním segmentov a kontrola správnosti formátu. Tieto činnosti sú pravidelne realizované jednou zo staníc (centrálne), túto stanicu označujeme ako riadiaca stanica spoja.

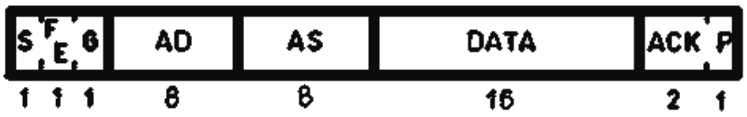


Obr. 5.9 Kruhová sieť Pierceovho typu

Siete Pierceovho typu patria k najdlhšie využívaným lokálnym sieťam a aj napriek malému využitiu kapacity kanálu a zlému chovaniu siete pri malých záťažiach sú často používané.

Cambridge Ring (Planet)

Kruhová lokálna sieť Cambridge ring bola vyvinutá na univerzite Cambridge v roku 1975 pre spojenie počítačov a koncentrátorov terminálov. Prenosovým médiom siete je dvojica symetrických vodičov (slúžia súčasne pre napájanie obovodov kruhového rozhrania staníc) dajú sa však použiť aj svetlovody. Pre symetrické vodiče je maximálna vzdialenosť medzi stanicami 100 m a prenosová rýchlosť je 10 Mb/s.



Obr. 5.10 Minipaket lokálnej siete Cambridge Ring

Dáta sú prenášané v minipaketoch o dĺžke 38 bitov, ich štruktúra odpovedá obrázku. Jednotlivé polia minipaketu majú nasledujúcu funkciu:

- *S*: synchronizácia, začiatok rámca,
- *F/F*: indikácia obsadenia rámca (Full/Empty),
- *G*: monitorovanie (Garbage bit),
- *AD*: adresa adresáta,
- *AS*: adresa odosielateľa,
- *DATA*: slovo dát,
- *ACK*: pole odpovede,
- *P*: paritná kontrola.

Bit G slúži riadiacej stanici siete k rozpoznaniu a likvidácii minipaketu s poškodenou adresou odosielateľa, ktorý by ináč blokoval obsadený slot. Tento bit je v odosielanom minipakete nastavovaný na hodnotu G=0. Riadiaca stanica ho prestaví na hodnotu G=1 a odosielateľ správy ho pri uvoľnení rámca opäť vráti na hodnotu G=0. Kombináciu odpovedajúcu obsadenému rámcu (F/E=1) a hodnote G=1 rozpozná riadiaca stanica ako chybu a rámec uvoľní.

Pole ACK slúži prijímaču k uloženiu potvrdenia, odosielateľ nastavuje pole na hodnotu ACK=11. Vrátené hodnoty poľa ACK majú potom nasledujúci význam:

- 00: dáta nemohol príjemca prevziať,
- 01: dáta boli prijaté,
- 10: dáta boli odmietnuté (indikácia chyby),
- 11: príjemca neodpovedá (chybná adresa).

Veľmi podobnú štruktúru paketu ako sieť Cambridge Ring má aj sieť Planet (Private Local Area Network) firmy Racal Milgo a sieť Domain firmy Apol 10 (dnes Hewlett - Packard). Sieť sa odlišuje prenosovým médiom, ktorým je koaxiálny kábel a štruktúrou paketu ktorý ma 42 bitov.

5.4.3 Vkladanie rámcov

Metóda vkladania rámcov bola vyvinutá firmou Hasler v roku 1974 a má dobre správanie v oblasti malých i veľkých záťaži. Jej nevýhodou je zložitejšia technická realizácia, prenos paketu sieťou ilustruje obrázok Obr. 5.11.

Obr. 5.11 Kruhová sieť s vkladaním rámcov

Stanica siete, ktorá chce vyslať paket, uloží paket do registra. Počká na koniec paketu, ktorý stanicou pravé prechádza a prepnutím prepínača predlží sieť o register. Odoslaný paket prejde sieťou, je prevzatý adresátom a vracia sa do registra stanice, ktorá register zo siete opäť odpojí.

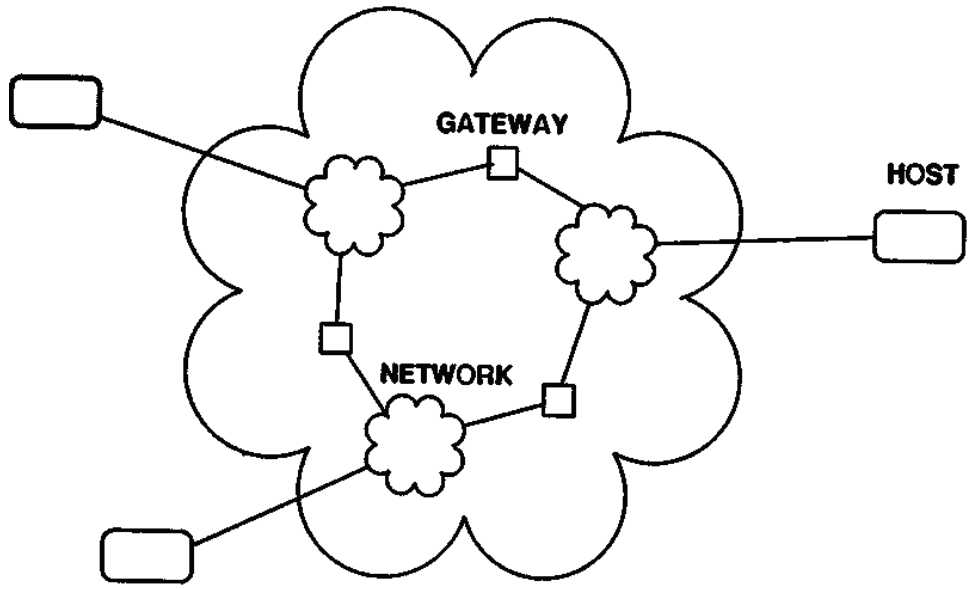
Kruhová sieť SILK, ktorá na tomto princípe pracuje, používa pre prenos rýchlosťou 16 Mb/s koaxiálny kábel 75 ohmový. Dátové pakety prenášajú šestnasťbitové slová.

5.5 Architekutúra Internetu a adresácia

V oblasti počítačových komunikácií hraje v súčasnej dobe primárnu úlohu technológia označovaná ako Internet alebo TCP/IP. Táto technológia prepojovania počítačových sieti (a to i s podstatne odlišnou architektúrou) vznikla v rámci projektu ARTPA.

5.5.1 Architektúra Internetu

Sieť Internet bola navrhnutá s cieľom prepojiť už existujúce počítačové siete jednotnou nadstavbou, ktorá by dovolila komunikáciu medzi ľubovoľnými prvkami vzájomne prepojených sietí (prepojovacích i lokálnych).



Obr. 5.12 Architektúra siete Internet

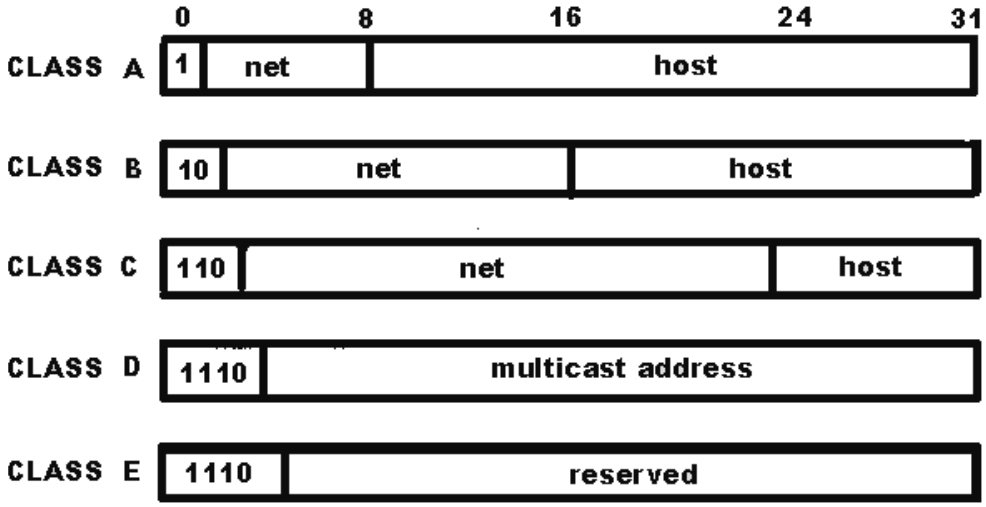
Základnými stavebnými prvkami Internetu sú brány (gateway) alebo smerovače (router), ich funkcia je obdobou funkcie prepojovacích uzlov v prepojovacích sieťach. Na rozdiel od prepojovacích sietí však Internet neprepojuje samostatné počítače, ale vzájomne i veľmi odlišné počítačové siete.
Podobne ako uzol v prepojovacej sieti prijíma brána pakety z pripojených sietí. Na základe internetovskej adresy (IP adresy), ktorú každý paket so sebou nesie, sa brána rozhoduje, či a ktorým smerom paket odošle. Na Internet ako celok sa môžeme dívať ako na datagramovú sieť.

5.5.2 Adresácia

Adresácia počítačových napojení, o ktorú sa Internet opiera je numerická. Každému napojeniu počítača do Internetu je pridelená jednoznačne 32-bitova internetovská adresa - *IP adresa*.

Pretože brány siete nerozlišujú pri smerovaní jednotlivé počítače, ale celé siete, musí mať adresácia hierarchický charakter. Z adresy počítača musí byť brána schopná ľahko adresu siete odvodiť. Súčasne je nutné dovoliť, aby adresy v konkrétnej sieti prideľoval správca siete a nie nejaká centrálna autorita Internetu.

Spôsob adresácie v Internete uvedené podmienky splňuje. Na IP adresu sa môžeme dívať ako na dvojicu (adresa-siete, adresa-počítača-v-sieti). Spôsob zakódovania adresy v 32-bitovej IP adrese uvádza obrázok Obr. 5.13.



Obr. 5.13 Štruktúra IP adresy

Pre adresáciu jednotlivých počítačov v sieti sú využívané prvé tri formáty adresy. Adresy triedy A umožňujú adresáciu v rozsiahlých sieťach s veľkým počtom počítačov (viac než 65536), pre adresu počítača v sieti je k dispozícii 24 bitov. Adresy triedy B dávajú pre identifikáciu počítača v sieti k dispozícii 16 bitov a sú určené pre siete majúce medzi 256 a 65355 pripojenými počítačmi. Konečne, trieda C slúži pre adresáciu v malých sieťach s menej než 256 pripojenými počítačmi. Pre zápis internetovských adries sa používa typická notácia, jednotlivé slabiky adresy, zapísané ako dekadické čísla, sú oddelené bodkami.

Poznamenajme, že IP adresa nie je adresa počítača, ale adresa jeho pripojenia k Internetu. Počítače, ktoré sú pripojene viacerými spojmi k viacerým sieťam, majú aj viac IP adries (to platí i pre brány).

Sieťovú časť IP adresy prideľuje centrálny správca internetu, prideľovanie časti adries identifikujúcich jednotlivé počítače v sieti je v kompetencii správcu tejto siete. K dispozícií má adresy v intervale 1.2n až 2, kde n je dĺžka poľa identifikujúceho počítač. Nulová hodnota poľa počítača slúži k označeniu siete ako takejto. Adresa obsahujúca v poli počítača samé jedničky definuje broadcast v danej sieti (*direct broadcast* - rozosielanie paketu všetkým počítačom danej siete).

Broadcast je často podporený schopnosťou siete (lokálnej siete) realizovať ho vyslaním jedinej správy adresovanej všetkým pripojeným staniciam. Vedľa metódy rozosielania, označovanej ako direct broadcast existuje ešte možnosť rozoslať paket všetkým staniciam vlastnej siete (*local broadcast*). IP adresa v tomto prípade obsahuje samé jedničky. Konečne, IP adresou so samými nulami identifikuje počítač sám seba, IP adresa s nulovým poľom siete identifikuje sieť do ktorej je pripojený.

Skupinová adresácia (multicasting) dovoľuje dynamicky vytvárať skupiny počítačov vo vnútry ktorých budú rozosielané pakety s danou adresou. Skupinová adresácia je väčšinou podporovaná podobnou schopnosťou lokálnych sietí (napr. Ethernetu) a zaistená schopnosťou brán smerovať pakety so skupinovou adresáciou. Táto prídavná služba je podporovaná zvláštnym protokolom spolupráce brán označovaným ako IGMP (Internet Group Management Protocol).

Ak si porovnáme jednoznačnú fyzickú adresáciu staníc v Ethernete s adresáciou v Internete zisťujeme podstatný rozdiel. Pokiaľ čo ethernetovská adresa je počítaču (jeho komunikačnému radiču) priradená napevno výrobcom, internetovská adresa musí odpovedať umiestneniu počítača v sieti a je priraďovaná správcom tejto siete. Pri odpojení počítača od jednej siete a jeho pripojenie k sieti inej musíme zmeniť IP adresu.

5.6 Prepájanie lokálnych sieti

Lokálne siete prepájajú úrady, technologické prevádzky, školy, obytné bytové domy a mnoho ďalších spoločností. Ich prenosový výkon je obmedzený kapacitou média, dovoľujú prepojiť obmedzený počet staníc a obmedzená je aj najväčšia vzdialenosť medzi stanicami. Pri väčších požiadavkách na rozľahlosť siete, na počet staníc alebo na kombináciu rôznych sieťových technológii, neostáva, len jednotlivé menšie siete medzi sebou prepojiť.

Lokálne siete prepojujeme pomocou špeciálnych staníc, pripojených ku dvom alebo viac prepojovaným sieťam, sústavu viac prepojených lokálnych sietí obvykle nazývame *internetwork*. Stanice, ktoré prepojujú lokálne siete označujeme ako *mosty (bridges)* a *smerovače (routers)*, pričom mosty sa už pri prepájní sieti typu Ethernet využívajú minimálne (skoro vôbec). Ich funkcia je podobná funkcii uzlov prepojovacej siete, a obvykle ju charakterizujeme termínom store-and-forward. Rámce prijate z pripojených sieti sú analyzované a podľa výsledku buď likvidované alebo uložené do pamäte a následne vyslane do niektorej siete.

Mosty a routere sa od seba líšia rozsahom informácie, ktorú pri analýze využívajú. Mosty sa opierajú iba o adresačné pole rámca (MAC adresy), routery analyzujú predávané dáta a využívajú informácii spojených s konkrétnym sieťovým alebo transportným protokolom. Existujú aj kombinovane prvky - *broutery*, ktoré pre niektorý sieťový alebo transportný protokol fungujú ako routere a pre iné protokoly ako mosty, a *viacprotokolové routere*, ktoré pre rôzne sieťové alebo transportne protokoly zaisťujú rôzne metódy smerovania.

Uzly v sieti Ethernet sa dávnejšie prepájali pomocou hub-ov (rozbočovačov), dnes sa prepájajú už iba pomocou switch-ov. Tak ako bol nahradený bridge za router, aj hub nahradil switch.

5.6.1 Bridge

Most (bridge) je najjednoduchším prvkom, ktorý dovoľuje prepojiť dve lokálne siete so zhodnou štruktúrou rámcov. Funkciu mostu si môžeme popísať takto:

- most príma všetky rámce zo všetkých pripojených lokálnych sietí,
- analyzuje adresy prijatých rámcov a rozhoduje sa, či ich odošle do niektorej lokálnej siete alebo ich zlikviduje,
- uloží prenášané rámce do svojej pamäte a nakoniec ich,
- odošle do vybranej siete.

Mosty môžu pracovať na niekoľkých princípoch, rozdiel spočíva v analýze prijatých rámcov. Doporučenie IEEE 802.2 definuje transparentne mosty, ďalšou možnosťou je explicitné smerovanie paketov (source routing) v sieti IBM Token Ring, do úvahy pripadá aj obecné smerovanie opierajúce sa o smerovacie tabuľky, získané analýzou topológie prepojovacej siete.

Prepojenie sieti mostami (mimo toho, že sa im v rozsiahlejších sieťach nevyhneme) zvyšuje *celkovú kapacitu siete* (komunikácia medzi dvoma stanicami v jednej lokálnej sieti nie je mostom prenášaná do inej siete), jej *spoľahlivosť* (porucha v jednej lokálnej sieti nenaruší chod v ostatných sieťach) a *bezpečnosť* (dáta prenášané medzi stanicami jednej siete sa nedajú opočúvať v iných sieťach).

Pomerne jednoduchou modifikáciou, rozdelením mostu na dve časti, vzájomne prepojené dvojbodovým spojom, sa dajú prepojiť i vzájomne geograficky oddelené lokálne siete. Kapacita prepájacieho spoja (prenajatý dátový spoj, spoj ISDN, kanál PCM) pochopiteľne ovplyvňuje priechodnosť mostu, a tým aj kvalitu služieb, ktoré takéto prepojenie aplikáciám poskytuje.

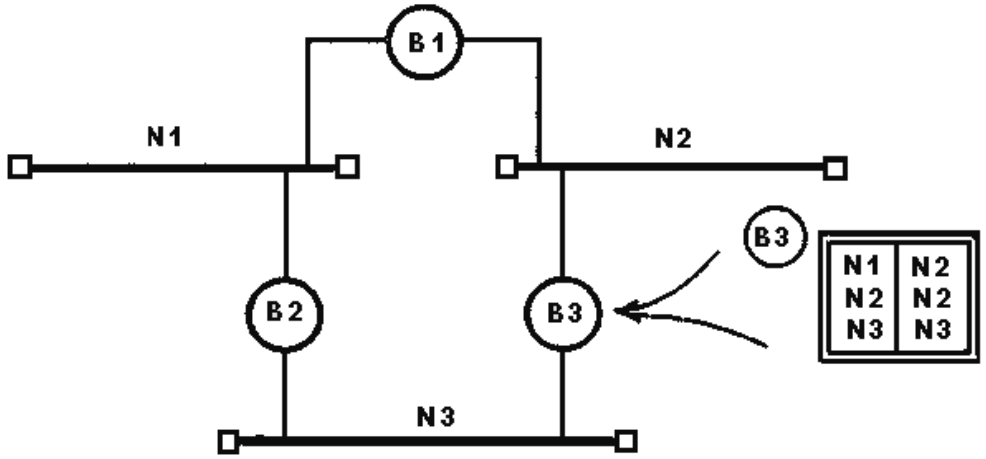
Hierarchické smerovanie

Hierarchické smerovanie je jednoduché, a dá sa použiť i v zložito prepojených lokálnych sieťach. Opiera sa o označenie jednotlivých prepojených sieti adresami, adresou každej stanice je potom dvojica (číslo lokálnej siete, MAC adresa stanice). Hierarchické smerovanie zaraďujeme do skupiny metód využívanej mostami, aj napriek tomu sa neopiera iba o MAC adresy staníc ale o prídavnú informáciu definovanú konkrétnym sieťovým alebo transportným protokolom.

Každý most ma k dispozícii pre každú pripojenú lokálnu sieť jednu smerovaciu tabuľku, v ktorej je pre každú z prepojených lokálnych sieti vyhradený jeden riadok. Ak prijme most rámec, potom na základe identifikácie cieľovej siete získa v tejto tabuľke informáciu, či ma rámec zlikvidovať, alebo či ho má previesť do niektorej z pripojených sieti. Príklad na obrázku Obr. 5.14 uvádza prepojenie troch sieti - N1, N2 a N3 tromi mostami - B1, B2 a B3 a odpovedajúce smerovacie tabuľky.

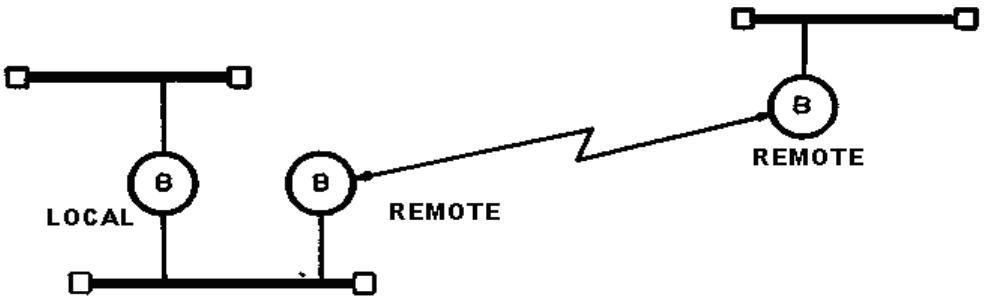
V skutočných sieťach vo smerovacích tabuľkách nájdeme namiesto identifikácie susednej siete identifikáciu komunikačnej dosky a adresu nasledujúceho uzla alebo priamo cieľovej stanice, identifikácia sieti je väčšinou pridaná.

Smerovacie tabuľky sú kľúčovým prvkom metódy. Dajú sa použiť fixne, predom definované smerovacie tabuľky, dajú sa vytvárať i tabuľky s alternatívnymi cestami dovoľujúce prekryť prípadné výpadky jednotlivých sietí a mostov. Okrem takýchto *statických tabuliek* sa dajú behom chodu siete vytvárať aj *tabuľky dynamické*.



Obr. 5.14 Most – hierarchycké smerovanie

S použitím hierarchického smerovania a s dynamickými tabuľkami sa môžeme stretnúť napríklad u mostov v sieti Netware firmy Novell. Sieť dovoľuje kombinovať rôzne sieťové technológie, Mac adresy hrajú do istej miery iba pomocnú úlohu a tak by asi bolo vhodnejšie hovoriť tu o routri.



Obr. 5.15 Miestny a vzdialený most

Transparentný most

Ak sú stanice prepojených lokálnych sieti jednoznačne adresované MAC adresami, dajú sa takéto siete prepojiť transparentným mostom. Funkcia mostu je definovaná normou IEEE 802.1 a most pracuje na nasledujúcom princípe:

1. Sleduje všetok chod v sieťach ktoré prepája.Vedie si evidenciu staníc, ktorých adresy sú uvedené ako adresy odosielateľa.Táto evidencia ma formu *smerovacej tabuľky (forwarding database).*Pre každú adresu, ktorá sa objavila v poli odosielateľa rámca je vo smerovacej tabuľke uvedená sieť, z ktorej správa s touto adresou prišla. Ukladanie do tabuľky je označované ako *učenie (bridge learning).*
2. Na každú správu, ktorá je prijatá mostom z niektorej pripojenej siete, most reaguje niektorým z troch spôsobov:
    - správa určená pre stanicu, o nej most vie, že leží v smere odkiaľ bola správa prijatá je likvidovaná,

- správa určená pre stanicu, o nej most vie, že leží v inej sieti, než z ktorej bola správa prijatá, je mostom prevedená do tejto siete,
- správa určená pre stanicu, ktorú most nepozná, je rozoslaná do všetkých smerov, okrem smeru, odkiaľ prišla.

Transparentný most pracuje iba v sieťach so stromovou štruktúrou, v ktorej uzly reprezentujú mosty a hrany reprezentujú prepojované lokálne siete.

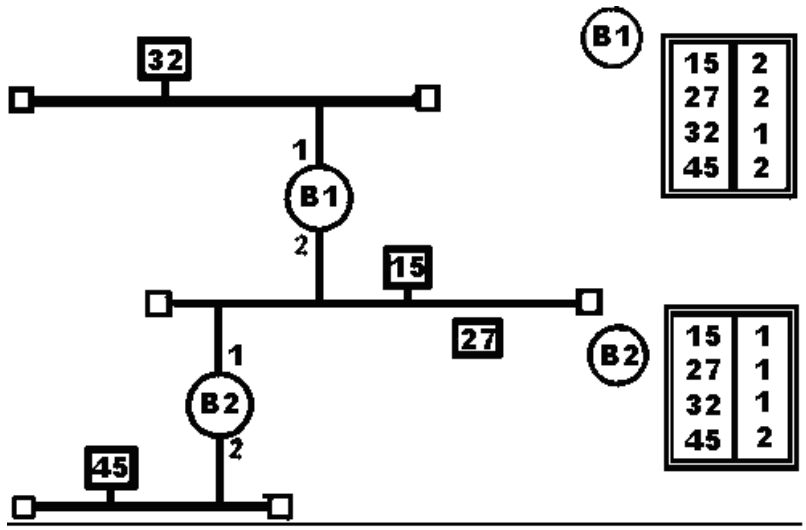
Príklad na obrázku Obr. 5.16 uvádza prepojenie troch sieti - N1, N2 a N3 dvoma mostami - B1 a B2. Správy stanice A určené stanici C sú prevádzané mostom B1, most B2 ich ignoruje. Správy medzi stanicami A a E sú prevádzané oboma mostami B1 a B2.

Algoritmus transparentného mostu funguje iba pre stromové topológie prepojenia sieti, ak sa v topológii objaví slučka, je nutne algoritmus doplniť o ďalšiu úroveň. V bežnej prevádzke sú blokované niektoré porty mostov tak, aby ostávajúce prepojenia tvorili strom. Blokované porty mostov ostávajú v zálohe pre prípad výpadku niektorého mostu alebo siete. Algoritmus výberu kostry (spanning tree algorithm) sa opiera o jednoznačnú číselnú identifikáciu mostov, distribuovaný vyber fungujúceho mostu s najnižšou identifikáciou a o nájdenie stromu najkratších ciest s vybraným uzlom ako koreňom.

Služobné rámce, ktoré si mosty medzi sebou vymieňajú pri konštrukcii kostry, majú zvláštny formát a sú označované ako BPDU (Bridge Protocol Data Unit). Pre ich predávanie sú využité všetky fungujúce porty mostov. Algoritmus zmeny topológie je periodicky spustený a dovolí prekryť výpadky lokálnych sieti a mostov.

Ako príklady transparentného mostu si uvedieme dvojportový HP 10:10 LAN Bridge.

(Hewlett-Packard), dovoľujúce prepojiť ľubovoľné z modifikácii Ethernetu, a vzdialený transparentný most HP Remote Bridge, dovoľujúci prepojiť siete Ethernet rýchlym dvojbodovým spojom (kanalom PCM 1.radu T1, rýchlym prenajatým spojom 64kb/s). Obidva mosty majú dostatočný výkon pre zapracovanie maximálneho toku v pripojených sieťach Ethernet a v prvom prípade schopnosť sieť Ethernet taktiež prenášaným tokom plne zaťažiť. Mosty samozrejme podporujú výber kostry podľa IEEE 802.1.



Obr. 5.16 Transportný most

V rade prípadov je možné funkciu transparentného mostu realizovať na osobnom počítači vybavenom dvoma rozhraniami Ethernetu (alebo Ethernetom a rýchlym sériovým rozhraním pre vzdialený most). Nevýhodou je nižší výkon a väčšinou chýbajúci algoritmus výberu kostry (to však v malých sieťach nevadí).

Explicitné smerovanie

V mnohých prípadoch je výhodné celkom vypustiť smerovacie tabuľky, všetku informáciu o ceste k cieľovej stanici potom musí niesť sama správa (vo forme zoznamu mostov a sietí, ktorými ma prechádzať).

Túto informáciu môžeme všetkým staniciam siete zadať staticky, predom, vo forme tabuliek. To je ale dosť nepružné a tak sa stretávame s metódami dynamického zisťovania najvýhodnejšej cesty. Odosielateľ potrebnú informáciu o ceste k adresátovi získa pri otváraní spojenia, kedy vyšle špeciálny rámec, ktorý je mostami rozoslaný do všetkých prepojených sietí. Tento rámec je cestou doplňovaný o adresy mostov a sietí, ktorými prechádza (ukladanie informácie o absolvovanej ceste do rámcov rozosielaných úplným broadcastom je nutné i pre rozhodnutie, či ma byť rámec ďalej rozosielaný alebo nie.) Adresát vracia odpoveď na každý rámec tohto typu, z viac odpovedí (u prepojenia so slučkami) si naša stanica vyberie najvýhodnejšiu cestu a tú uvádza v záhlaví dátových správ.

Uvedená metóda explicitnej adresácie je použitá v sieti IEEE 802.5 IBM Token Ring a označovaná ako source routing. Jednotlivé položky určujúce ďalší most a sieť majú dĺžku 16 bitov, z toho vždy 4 bity určujú ďalší most a 12 bitov ďalšej lokálnej siete na ceste k adresátovi. Použitie explicitnej adresácie je indikované jedným bitom v adrese adresáta.

5.6.2 Hub

Ethernetový rozbočovač (Hub) je aktívny prvok počítačovej siete, ktorý umožňuje jej vetvenie. Chová sa ako viacportový opakovač. To znamená, že všetky dáta, ktoré prichádzajú na jeden z portov skopíruje na všetky ostatné porty, bez ohľadu na to, na ktorom porte sa nachádza zariadenie, ktorému sú dáta adresované. Rozbočovač zväčšuje kolíznu doménu. To má za následok, že všetky zariadenia na danom segmente siete „vidia“ všetky rámce, aj tie, ktoré sú adresované iným zariadeniam a u väčších sietí to znamená zbytočné preťažovanie tých zariadení, ktorým dáta v skutočnosti nepatria. Rozbočovače pracujú na Fyzickej vrstve OSI modelu. Nástupcom sieťových rozbočovačov sú switch-e (prepínače), ktoré sieťovú prevádzku inteligentne prepínajú (pracujú na Linkovej vrstve OSI modelu).

5.6.3 Router

Ak sa požaduje, aby si jednotlivé segmenty zachovali relatívnu samostatnosť (napr. vlastnú sieťovú adresu, možnosť samostatnej správy a pod.), alebo keď treba vzájomne prepojiť lokálne siete rôznych typov, ak sa spájajú dve lokálne siete cez sieť rozľahlú alebo sa vytvára vzájomné prepojenie sieti so zložitejšou topológiou, treba na to použiť všeobecnejšie riešenie, než aké ponúkajú mosty.

Potrebné je prepojovacie zariadenie, ktoré už pracuje na úrovni sieťovej vrstvy a nazýva sa smerovač (router). Takéto zariadenie už vníma vlastný obsah jednotlivých rámcov (na úrovni linkovej vrstvy), dokáže správne rozpoznať formát jednotlivých paketov, ktoré sú v rámcoch prenášané a využiť informácie, ktoré sú v nich obsiahnuté (Obr. 5.17).

Hlavná úloha smerovačov je vlastne zhodná s úlohou sieťovej vrstvy - teda postarať sa o doručenie paketov od ich pôvodného odosielateľa až ku konečnému príjemcovi. Smerovače teda musia prijímať rozhodnutia o tom, kadiaľ majú ďalej odoslať každý

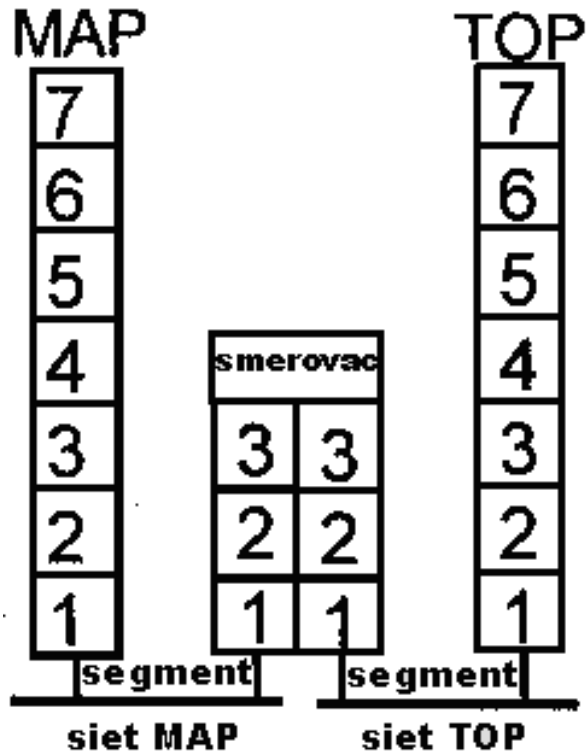
jednotlivý paket tak, aby sa dostal k svojmu cieľu - teda zaisťovať to, čo sa bežne nazýva *smerovanie* (routing). Musia nutne používať nejaký algoritmus smerovania, na ktorého základe svoje rozhodnutia prijímajú.

Tento algoritmus a z neho vychádzajúce smerovanie môže mat statickú povahu (t.j. byť nezávislé na okamžitom stave siete), alebo môže mať naopak dynamickú povahu (a reagovať na priebežnú situáciu). V tomto druhom prípade, ktorý je dnes najčastejší, potom ešte potrebuje vhodnú metódu, resp. protokol, ktorého prostredníctvom získava potrebné informácie o stave siete.

Smerovače sa od mostov ďalej odlišujú tým, že sú pre ostatné entity na úrovni sieťovej a linkovej vrstvy viditeľné. Majú svoje adresy a pakety, ktoré nimi majú prejsť, sú im explicitne adresované (zatiaľ čo mosty zachytávajú celú prevádzku v každom z pripojených segmentov). Preto tiež smerovače spracovávajú menej rámcov než mosty, ale ich spracovanie je zasa o to náročnejšie.

Treba si uvedomiť, že pre funkciu smerovačov je nutné, aby vzájomne prepojované siete používali rovnaký protokol na úrovni sieťovej vrstvy - podľa neho vlastne smerovač rozpoznáva odosielateľa aj adresáta jednotlivých paketov a rozhoduje o tom, kadiaľ ich má ďalej odoslať. Nie je však ale nutné, aby to iste platilo aj na úrovni linkovej a fyzickej vrstvy. Tu sa už konkrétne protokoly a prenosové technológie môžu líšiť.

Smerovače sú dnes obvykle konštruované tak, aby mali viac rôznych rozhraní (tzv.portov) a bolo ich možné vzájomne prepojiť napríklad pomocou pevných okruhov, verejných údajových sietí, optických prenosových ciest a pripojiť k nim rôzne lokálne siete podľa štandardu IEEE 802 a pod.



Obr. 5.17 Metóda prepojovania pomocou routra

Multiprotokolové smerovače

Požiadavka rovnakého (a teda jediného) protokolu v sieťovej vrstve je však veľmi obmedzujúca, zvlášť v dnešnej dobe, keď vedľa seba koexistuje rad sústav protokolov (okrem ISO/OSI tiež TCP/IP, DECnet a ďalšie) a užívatelia volajú po ich čo najtesnejšej integrácii v rámci tzv. *heterogénnych sietí* (t.j. sieti, ktorých uzly používajú rôzne sústavy protokolov).

Problém heterogénnych sieti možno riešiť v princípe dvoma spôsobmi - konverziou protokolov a smerovaním viacerých protokolov súčasne. Riešenie prostredníctvom konverzií sa ukázalo značne náročné a nespoľahlivé a preto sa presadila predovšetkým druha možnosť. Poprední výrobcovia dnes ponúkajú tzv. *multiprotokolové smerovače* (multiprotocol routers), schopné pracovať súčasne s viacerými rôznymi protokolmi. Multiprotokolový smerovač musí byť schopný rozpoznať typ paketu, ktorý dostane od linkovej vrstvy a podľa toho potom aplikovať ten smerovací algoritmus, ktorý k príslušnému sieťovému protokolu prislúcha.

Brouter

V dnešnej dobe, keď dochádza k stále tesnejšiemu prepojovaniu rozľahlých a lokálnych sietí, je použitie mostov i smerovačov veľmi rozšírené. Rozhodnutie medzi tým, či v určitej situácii použiť most alebo smerovač, nemusí byť vždy okamžite zrejmé, zvlášť potom pri lokálnych sieťach so zložitejšou topologiou a s väčším počtom používaných protokolov. V dnešnej dobe však už existujú tiež zariadenia, ktoré v sebe kombinujú funkcie oboch týchto zariadení. V angličtine sa pre ich označenie používa najčastejšie termín *bridge/router*, niekedy tiež *brouter*.

Ide o zariadenie, ktoré sa snaží fungovať ako smerovač a až v okamihu, keď pre nejaký paket nedokáže aplikovať smerovací algoritmus, odovzdá pôvodný rámec ďalej tak, ako by to urobil most. Výhodou takéhoto zariadenia je potom aj to, že sa dokáže vyrovnať s takými protokolmi, ktoré vôbec nemožno smerovať, lebo nepočítajú so sieťovou vrstvou. Ako napríklad protokoly DECLAT (DEC Local Area Transport), LU6.2 firmy IBM a protokoly NetBios.

5.6.4 Switch

Prepínač (switch) je aktívny prvok počítačovej siete, ktorý spája jej jednotlivé časti (uzly). Prepínač slúži ako centrálny prvok v sieťach hviezdicovej topológie. Prepínače sú v porovnaní s rozbočovačom inteligentnejšie, podľa MAC adresy dokážu rozpoznať, kam majú byť dáta doručené a tak prepínač dáta nepreposiela na všetky porty súčasne. Vďaka tomu dokáže prepínač radikálne znížiť tok zbytočných dát na sieti a je omnoho efektívnejší ako rozbočovače.

Prepínač v pamäti vytvorí a udržiava tabuľku MAC adries všetkých zariadení pripojených k jednotlivým portom. Po jeho štarte je však celá MAC tabuľka prázdna a preto sa správa ako rozbočovač. Rozposiela dátové rámce všetkým pripojeným zariadeniam, no prijíma ho len zariadenie, pre ktoré je rámec určený. Odpoveďou naň sa zariadenie zapíše do MAC tabuľky prepínača. Keď dátový rámec príde na port prepínača, prepínač porovná cieľovú MAC adresu v rámci so svojou tabuľkou MAC adries. Tak je prepínač schopný určiť, na ktorý port má prijatý dátový rámec preposlať.

Prepínače pracujú na Linkovej vrstve OSI modelu. Umožňujú segmentáciu siete na kolízne domény (ang. collision domain). Každý port prepínača predstavuje oddelenú kolíznu doménu a poskytuje celú šírku pásma sieťového média jednému alebo viacerým uzlom pripojeným k tomuto portu. Čím menej uzlov je v kolíznej doméne, tým väčšia šírka pásma pripadá na každý uzol a zmenšuje sa počet kolízií.

5.6.5 Gateway

Jedná se o zariadenie, ktoré prepája dva úplne odlišné komunikačné systémy, čiže siete úplne odlišných koncepcií. Prevádza konverziu komunikačných protokolov. Pracuje na všetkých úrovniach ISO/OSI modelu, na ktorých je nutné túto konverziu prevádzať – teda v niektorých prípadoch aj na aplikačnej vrstve.

6 Protokoly

Protokoly sa dajú rozdeliť podľa vrstvy v ISO/OSI modely, v ktorej sa používajú. Takže v tejto kapitole budú rozobraté protokoly od fyzickej vrstvy až po aplikačnú vrstvu.

6.1 Protokoly fyzickej a linkovej vrstvy

Fyzická vrstva definuje fyzické prepojenie medzi prvkami siete, jeho mechanické vlastnosti (konektory, typ média), elektrické vlastnosti (napäťové úrovne, spôsob kódovania a modulácie) a u lokálnych sietí aj spôsob prepojenia jednotlivých počítačov a metódu prístupu k prenosovému médiu. Príklady protokolov fyzickej vrstvy:

- Bluetooth (popis fyzickej vrstvy),
- CAN (popis fyzickej vrstvy),
- DSL,
- EIA RS-232, EIA-422, EIA-423, RS-449, RS-485,
- Etherloop,
- 10BASE-T, 10BASE2, 10BASE5, 100BASE-TX, 100BASE-FX, 100BASE-T, 1000BASE-T, 1000BASE-SX, atď. (Ethernet),
- GSM,
- rozhranie IEEE 1394 (FireWire),
- rozhranie IEEE 1284 (Centronics),
- ISDN,
- IRDA,
- OTN (Optical Transport Network) optická transportná sieť
- USB (popis fyzickej vrstvy),
- Wi-Fi IEEE 802.11 (popis fyzickej vrstvy),
- atď.

Linková vrstva definuje pravidlá pre predávanie správ. Správy sú sieťou prenášané v pevne definovaných rámcoch, rámce dovoľujú chrániť predávané dáta proti chybám. Štruktúra rámca často limituje dĺžku prenášaných blokov dát a hovoríme o takzvaných paketoch. Príklady protokolov linkovej vrstvy:

- CDP (Cisco Discovery Protocol),
- Ethernet,
- FDDI (Fiber Distributed Data Interface),
- Wi-Fi IEEE 802.11 (popis linkovej vrstvy),
- WiMAX IEEE 802.16 (popis linkovej vrstvy),
- LocalTalk,
- PAgP (Cisco Systems proprietary link aggregation protocol),
- PPP (Point-to-Point Protocol),
- PPTP (Point-to-Point Tunneling Protocol),
- RPR (Resilient Packet Ring) IEEE 802.17,
- StarLAN,
- STP (Spanning Tree Protocol),
- Token ring (protokol vyvinutý spoločnosťou IBM),
- VLAN (Virtual Local Area Network),
- atď.

6.2 Protokoly sieťovej vrstvy

Sieťová vrstva definuje spôsob akým sa pakety pohybujú po sieti ako si ich jednotlivé prvky siete predávajú na ich ceste od odosielateľa k adresátovi. Mechanizmy vrstvy sa konečne starajú aj o ochranu siete proti nadmernej záťaži. Príkladmi takýchto protokolov sú:

- ARP protokol,
- RARP protokol,
- IP (internetový protokol),
- protokol ICMP,
- protokol GGP,
- protokol EGP,
- protokol IGP,
- atď.

6.2.1 ARP protokol

S odlišnosťou IP adresácie a adresácie v konkrétnej sieti je spojený jeden problém. Ak potrebujeme adresovať konkrétny počítač (alebo bránu) v sieti pripojenej k Internetu, nestačí nám IP adresa, ale musíme poznať i fyzickú adresu tohto počítača v danej sieti (napr. fyzickú ethernetovskú adresu). Statické uloženie odpovedajúcich hodnôt do tabuliek je možné, ak sa väzba medzi IP adresou a adresou v sieti (napr. adresou v sieti X.25) nemení. Statická väzba adries zrejme nevyhovie v lokálnych sieťach, v ktorých môžeme počítače ľahko presúvať zo siete do siete.

Pre lokálne siete, v ktorých nemôžeme fyzickú adresu meniť (napr. Ethernet) je v Internete k dispozícii postup, ktorý dovolí každej stanici v lokálnej sieti zistiť fyzickú adresu inej stanice. Postup označovaný ako ARP protokol (Adress Resolution Protocol) a zahrňuje nasledujúce kroky:

1. Stanica, ktorá potrebuje zistiť fyzickú adresu pre danú IP adresu (pytajúci), rozošle fyzickým broadcastom otázku s touto IP adresou (*ARP Request*),
2. Stanica, ktorá ma pridelenú rozosielanú IP adresu odpovie pýtajucemu (*ARP Reply*), odpoveď obsahuje vedľa IP adresy i fyzickú adresu odosielateľa odpovede,
3. Pýtajúci uloží IP adresu a odpovedajúcu fyzickú adresu do tabuľky (*ARP Cache*).

Skutočný algoritmus je ďalej upravovaný tak, že i prvotnú otázku (prijatú všetkými stanicami) využijú stanice k zápisu do svojích tabuliek. V otázke je preto uložená i IP adresa a fyzická adresa pýtajucého.

6.2.2 RARP protokol

Ďalším typickým problémom lokálnych sietí je zahrnutie bezdiskových staníc a prenosných počítačov (portable, laptop) do Internetu. Tieto stanice nemajú pamäť potrebnú k trvalému uloženiu IP adresy, naviac je nutne pre ich zapojenie do rôznych sietí túto adresu často meniť.

Automatické pridelenie IP adresy pripojenému počítaču zaisťuje protokol RARP (Reverse Address Resolution Protocol).

1. Stanica po svojom pripojení k sieti požiada rozoslaním RARP žiadosti (*RARP Request*) o pridelenie adresy.
2. Prideľovanie IP adries zaisťuje jeden z počítačov v lokálnej sieti - primárny *RARP server*, ktorý na žiadosť odpovie pridelením IP adresy.

Proti výpadku primárneho servera, sieť môžu zaisťovať sekundárne RARP servery, ktoré sledujú RARP protokol a pri výpadku primárneho RARP servera ho nahradia.

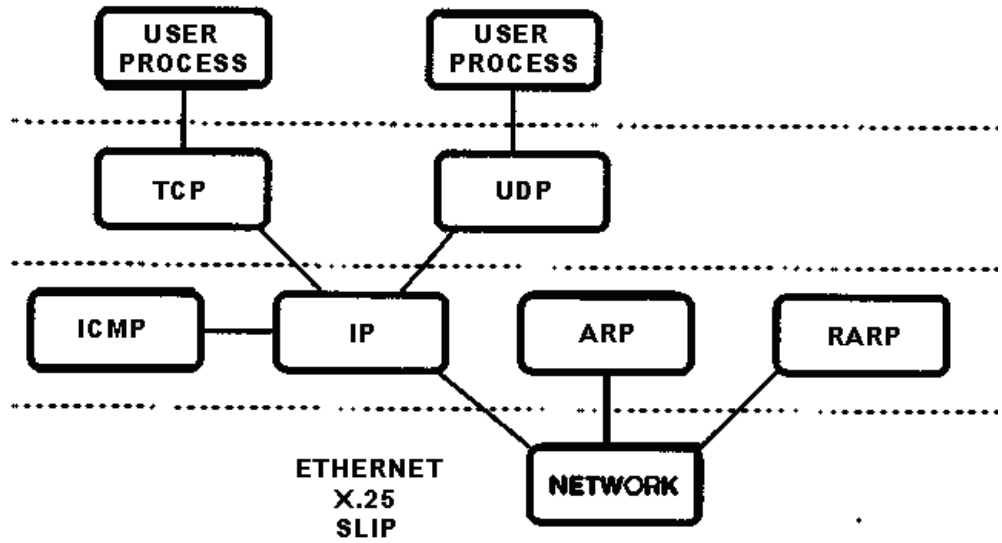
6.2.3 Internetový protokol (IP)

Ako sme si už skôr uviedli, Internet je tvorený prepojenými sieťami s odlišnou filozofiou. Dajú sa prepojovať pomerne spoľahlivé lokálne siete a virtuálne kanály verejných dátových alebo privátnych sieti X.25 CCITT, dajú sa prepojovať aj datagramové siete s omnoho horšími službami. Internet sa musí prispôsobiť vlastnostiam aj tých najnevýhodnejších sietí a je preto navrhnutý ako *sieť datagramová*.

Prenos datagramov medzi koncovými účastníkmi zaisťuje Internet Protocol (IP), ktorý definuje hlavne formáty. Internet Protokol sám nezaisťuje potvrdzovanie, spolieha sa na prípadné potvrdzovacie schémy v jednotlivých prepojovaných sieťach a na potvrdzovanie vo vyšších transportných protokoloch, ktorými sú v Internete *User Datagram Protocol* (UDP) a *Transmission Control Protocol* (TCP).

Kombinácia skratiek sieťového protokolu IP a transportného protokolu TCP je často používaná ako označenie internetovskej technológie (*TCP/IP*). Protokoly TCP/IP sú základnou podmienkou pre pripojenie počítača alebo siete počítačov k Internetu.

Architektúru základných protokolov Internetu si môžeme ilustrovať na obrázku Obr. 6.1:

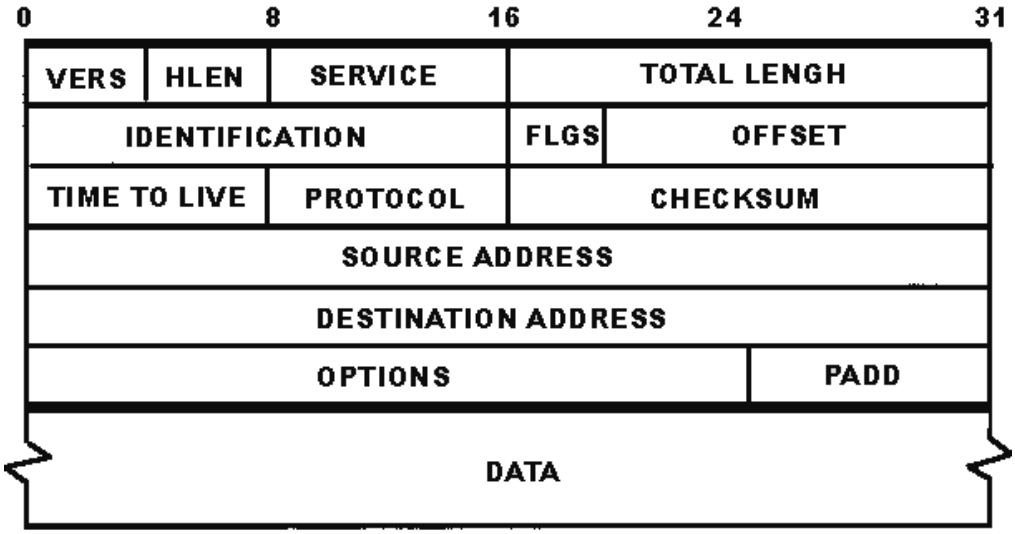


Obr. 6.1 Protokoly internetu

Protokol IP tvorí základňu pre vyššie prenosové protokoly a pre služobné protokoly Internetu. Sám sa opiera o protokoly sieti, ktoré prenášajú jeho pakety. Takouto sieťou je najčastejšie lokálny Ethernet, dátové spoje X.25 a rýchle kanály typu T1, ale aj vyhradene linky obsluhovanej špecializovaným protokolom PPP alebo primitívnym protokolom SLIP (Serial Line Internet Protocol).

Základným elementom protokolu IP je štruktúra prenášaného datagramu - IP paketu. IP paket je tvorený záhlavím a vlastnými prenášanými dátami. Záhlavie

je zabezpečené jednoduchým kontrolným súčtom, prenášané dáta nie sú zabezpečené. Štruktúru IP paketu uvádza obrázok Obr. 6.2.



Obr. 6.2 IP paket (fragment)

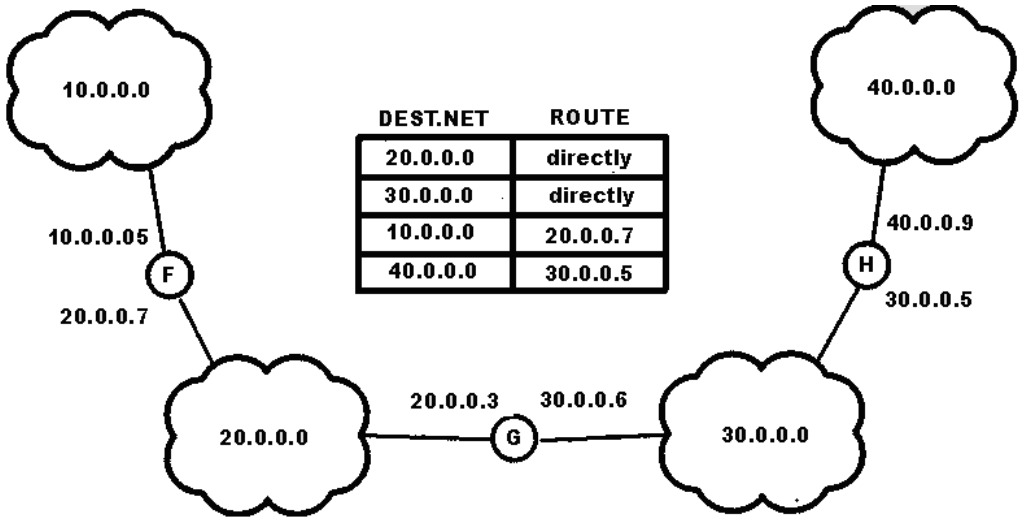
Protokol IP musí pracovať nad sieťami, ktoré sa líšia aj pokiaľ ide o taký parameter ako je dĺžka prenášaného paketu. Ak sieť neumožňuje, ktorou ma niektorá brána odoslať IP paket k adresátovi, preniesť IP paket vcelku, môže brána (ak to nie je zakázané) paket rozdeliť na samostatne smerované fragmenty. Zloženie fragmentu do IP paktu je úlohou adresáta, ak chýba niektorý fragment, je cely IP paket ignorovaný. Jednotlivé polia IP paketu (alebo fragmentu) majú nasledujúcu funkciu:

- *VERS* štvorbitové pole, definuje verziu IP protokolu, zatiaľ najpoužívanejšia verzia ma poradové čislo 4 (IPv4), ale už sa používa aj verzia 6 (IPv6),
- *HLEN* štvorbitové pole, udáva dĺžku záhlavia v 32 bytových slovách, nevyužité slabiky sú označované ako výplň (PADD),
- *SERVICE* osembitové pole, udáva požiadavku na prioritu IP paketu (0-7), požadované minimálne oneskorenie, požadovanú vysokú kapacitu a požadovanú vysokú spoľahlivosť. Tieto požiadavky dovolia vybrať najvýhodnejšiu cestu, Internet však ich splnenie negarantuje,
- *TOTAL LENGTH* šestnásťbitové pole udáva celkovú dĺžku IP paketu (fragmentu) v slabikách, zahrňuje záhlavie aj dáta, najväčšia možná dĺžka IP paketu je 65535 slabík,
- *IDENTIFICATION* šestnásťbitové pole, identifikuje IP paket a podporuje fragmentáciu,
- *FLAGS* tri jednobitové príznaky, podporujú fragmentáciu (zákaz fragmentacie, identifikácie posledného fragmentu v IP pakete),
- *OFFSET* trinásťbitová relatívna adresa fragmentu v Ip pakete v násobkoch ôsmich slabík,
- *TIME TO LIVE* osembitové pole udáva dobu života IP paketu v sekundách (Time to Live), každá brána znižuje hodnotu o jedničku, pri nule je IP paket (fragment) likvidovaný,

- *PROTOCOL* šestnásťbitová identifikácia protokolu vyššej vrstvy (TCP, UDP),
- *CHECKSUM* šestnásťbitový kontrolný súčet záhlavia paketu (fragmentu),
- *SOURCE ADDRESS* tridsaťdvabitová IP adresa odosielateľa IP paketu (IPv4, IPv6 používa 128 bitov),
- *DESTINATION ADDRESS* tridsaťdvabitová IP adresa adresáta IP paketu,
- *OPTIONS* pripadné riadiace a ladiace funkcie, meranie v sieti,
- *PADD* výplňové nulové slabiky,
- *DATA* prenášané dáta.

Smerovanie IP paketov

Brána pripojená k sieti analyzuje všetky prichádzajúce IP pakety. IP pakety, ktorých cieľová adresa odpovedá adrese iných počítačov v sieti, z ktorých prichádzajú, ignoruje. Zvyšné IP pakety sú buď určene priamo bráne, alebo majú byť smerované do iných sietí. Smerovacia tabuľka brány je tvorená položkami pre jednotlivé prepojene siete (a nie pre jednotlivé uzly), každá položka smerovacej tabuľky obsahuje pre danú cieľovú sieť IP adresu ďalšej brány na ceste k nej, pripadne informáciu, že sa jedna o sieť, ku ktorej je brána priamo pripojená. Príklad jednoduchej siete a údaje smerovacej tabuľky uvádza obrázok Obr. 6.3.



Obr. 6.3 Prepojenie sietí v Internete a smerovacie tabuľky

Implicitné cesty

Pre udržanie malého rozsahu smerovacích tabuliek, a vzhľadom k pomerne bežnej stromovej topológií, používa Internet zvláštnu položku v smerovacej tabuľke, označovanú ako implicitná cesta. Pre jednoduché pripojenie sieti k Internetu, aké ilustruje napr. sieť na obrázku Obr. 6.3 postačí mať v smerovacích tabuľkách údaje o smerovaní vnútri týchto sietí, smerovanie pre všetky ostatné siete Internetu definuje implicitná cesta.

Špecifikácia cesty

Aj napriek tomu, že Internet je koncipovaný ako sieť prepojujúca siete a nie jednotlivé počítače, programové vybavenie väčšinou dovoľuje definovať zvláštne smerovanie aj pre jednotlivé počítače.

6.2.4 Protokol ICMP

Protokol ICMP (Internet Control Message Protocol) prenáša v sieti Internet služobné informácie. Pakety ICMP protokolu sa prenášajú ako bežné datagramy, ich doručovanie sa riadi pravidlami pre doručovanie IP paketov s tým, že problémy pri doručovaní nie sú signalizované.

Pakety ICMP nám umožňujú realizovať radu potrebných funkcií. Medzi tieto funkcie patrí hlavne:

- vyžiadanie okamžitej odpovede od adresáta a tato odpoveď (služba ECHO),
- oznámenie o nedostupnosti adresáta,
- prekročenie doby života IP paketu,
- žiadosť o obmedzenie vysielania pri nezahltenej sieti,
- informácia o výhodnejšej ceste,
- distribúcia adresačných masiek,
- synchronizácia hodín.

Brána sa stará o informovanie počítačov v pripojenej sieti o výhodnejších cestách a dáva im tak možnosť vytvárať a upravovať ich smerovacie tabuľky. Distribúcia masky podsiete dovoľuje pripojeným počítačom rozpoznať, ktoré z adries odpovedajú počítačom v lokálnej sieti a ktoré adresy počítačom mimo lokálnu sieť (dostupným cez bránu) a správne fyzicky smerovať odchádzajúce pakety.

6.2.5 Autonómne systémy

Rozsiahlosť siete Internet, ktorá dnes pokrýva celý svet, si vyžiadala jej hierarchické rozčlenenie. A to nie len pokiaľ ide o adresáciu, ale aj pokiaľ ide o metódy riadenia v jednotlivých jej častiach.

Základom siete Internet je *chrbticová sieť* (core) tvorená bránami chrbticovej siete (Core Gateways). K týmto bránam sú pripojovane *autonómne systémy*, ktoré využívajú chrbticu iba pre vzájomnú komunikáciu.

6.2.6 Protokoly GGP

Úlohou chrbticovej siete je predovšetkým smerovať premávku medzi pripojenými autonómnymi systémami. Smerovanie medzi bránami chrbticovej siete sa v začiatkoch Internetu opieralo o metódu *GGP* (*Gateway-to-Gateway Protocol*), modifikáciou Ford-Fulkersonovho algoritmu. Neskôr bolo pre nevýhody tohto algoritmu (zlá stabilita pri zmenách v sieti) nahradené smerovaním podľa algoritmu označovaného ako *SPF algoritmus* (*Shorted Path First*). V algoritme SPF si brány vymieňajú informácie o zmenách topológie a na ich základe si vytvárajú obraz o aktuálnom stave *chrbticovej siete*. Zo získaných údajov si každá brána samostatne vytvára smerovaciu tabuľku podľa Dijkstrovho algoritmu.

6.2.7 Protokol EGP

Pre smerovanie toku do sieti v autonómnych systémoch musia byť brány chrbticovej siete a niektoré brány v ostatných autonómnych systémoch informované o adresách týchto sietí. Je úlohou brán v autonómnych systémoch tieto informácie poskytovať, príslušný protokol je označovaný ako EGP protokol - External Gateway Protocol.

6.2.8 Protokoly IGP

Smerovanie vo vnútri autonómnych systémov je na smerovanie v chrbtici značne nezávislé. Opiera sa o smerovacie tabuľky, ktoré si vytvárajú vnútorné brány autonómneho systému. Používajú pritom takmer univerzálne protokol označovaný ako *RIP* - Routing Information Protocol. Ten je založený na Ford- Fulkersonovom výpočte smerovacích tabuliek, pri výpočte sa uvažuje vzdialenosť susedov ako jednotková. Ďalším používaným protokolom je *HELLO*, pôvodne navrhnutý pre chrbticu siete NSFNET, ten sa podobne ako RIP opiera o Ford - Fulkersonov výpočet, ale využíva zámerne oneskorenie IP paketov. V budúcnosti sa dá očakávať rozšírenie protokolu OSPF-Open SPF, v ktorom si brány iba vymieňajú informácie o topológií a tabuľky počítajú lokálne.

6.3 Protokoly transportnej vrstvy

Transportná vrstva definuje adresáciu počítačov a aplikačných programov v sieti, zaisťuje vytváranie dočasných komunikačných spojení medzi nimi a konečne aj rozklad správ do paketov a skladanie paketov do správ. Príkladmi protokolov transportnej vstvy sú UDP a TCP protokol.

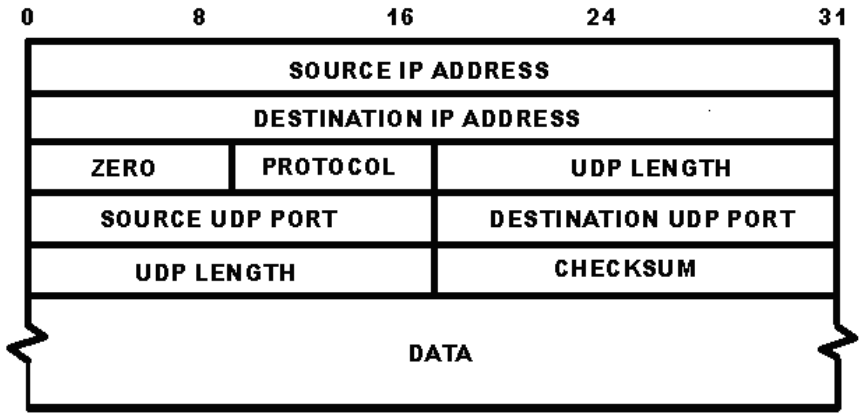
6.3.1 Protokol UDP

Datagramový protokol UDP (User Datagram Protocol) je jednoduchou nadstavbou IP protokolu. Zaisťuje prenos aplikácií definovaných blokov dát s nezaručenou spoľahlivosťou - datagramov internetom stratégiou best effort. Oproti protokolu IP pridáva schopnosť rozlíšiť viac adresátov vo vnútri cieľového adresáta - počítača.

Účel UDP protokolu :

- primárny mechanizmus pre posielanie datagramov medzi aplikačnými procesmi,
- rozlišuje medzi viac potenciálnymi adresátmi na danom počítači,
- poskytuje možnosť využiť takú istú kvalitu služieb ako ponúka IP.

Formát *UDP paketu* a spôsob ich zapuzdrenia do IP paketu uvádza obrázok Obr. 6.4:



Obr. 6.4 Štruktúra UDP paketu a jeho prenos

Krátke záhlavie UDP paketu definuje čísla socketu odosielateľa a adresáta, dĺžku UDP paketu (záhlavie a prenášanie dát vo slabikách) a voliteľne kontrolný súčet. Pri výpočte kontrolného súčtu berieme v úvahu nie len vlastné záhlavie UDP paketu, ale i niektoré informácie z IP záhlavia (pseudo - header).

Malou zaujímavosťou je spôsob akým je príjemca paketu informovaný o skutočnosti, že UDP paket je alebo nie je zabezpečený kontrolným súčtom. Pre výpočet kontrolného súčtu je použitý inverzný binárny kód, ktorý ma dvojitý obraz nuly. Samé jednotky v poli checksum indikujú nulový kontrolný súčet, samé nuly nezabezpečený UDP paket.

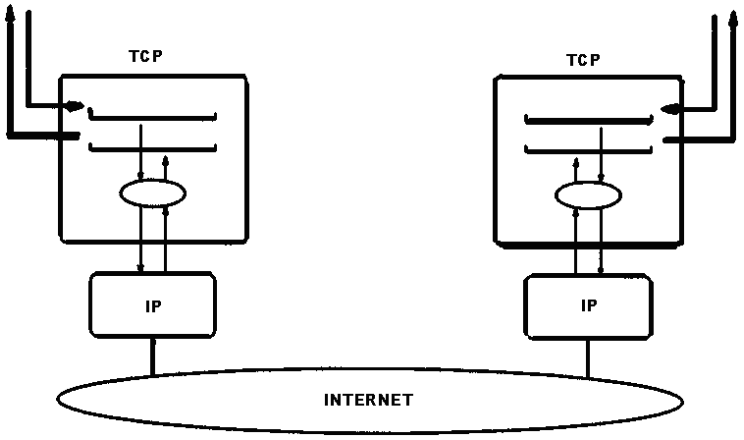
Čísla UDP socketu jednoznačne identifikujú aplikačné procesy predávajúce si medzi sebou UDP paket. Šestnásťbitová identifikácia UDP socketu dovoľuje aplikačný proces pripojiť k jednému z 65535 UDP socketov. Niektoré UDP sockety sú práve pridelené vnútorným službám Internetu a UNIXu, zoznam niektorých z nich uvádza nasledujúca tabuľka (Tab. 6.1):

Tab. 6.1 Čísla socketov niektorých vnútorných služieb Internetu pre protokol UDP

Číslo socketu	Služba	Popis
7	ECHO	Služba ECHO.
11	USERS	Užívatelia.
13	DAYTIME	Čas.
42	NAMESERVER	Host Name Server.
53	DOMAIN	Domain Name Server.
69	TFTP	Primitívny FTP protokol.
111	SUNRPC	Sun RPC protokol.

6.3.2 Protokol TCP

Protokol UDP poskytuje aplikáciam službu nezabezpečeného prenosu užívateľom definovaných dát. Pokiaľ je nutné prenos dát zabezpečiť musí potrebnú potvrdzovaciu schému zaistiť vlastná aplikácia. Pre aplikácie, ktoré vyžadujú spoľahlivý prenos dát a nemajú zahrňovať komplikované potvrďovanie je k dispozícii protokol TCP (Transmission Control Protocol).

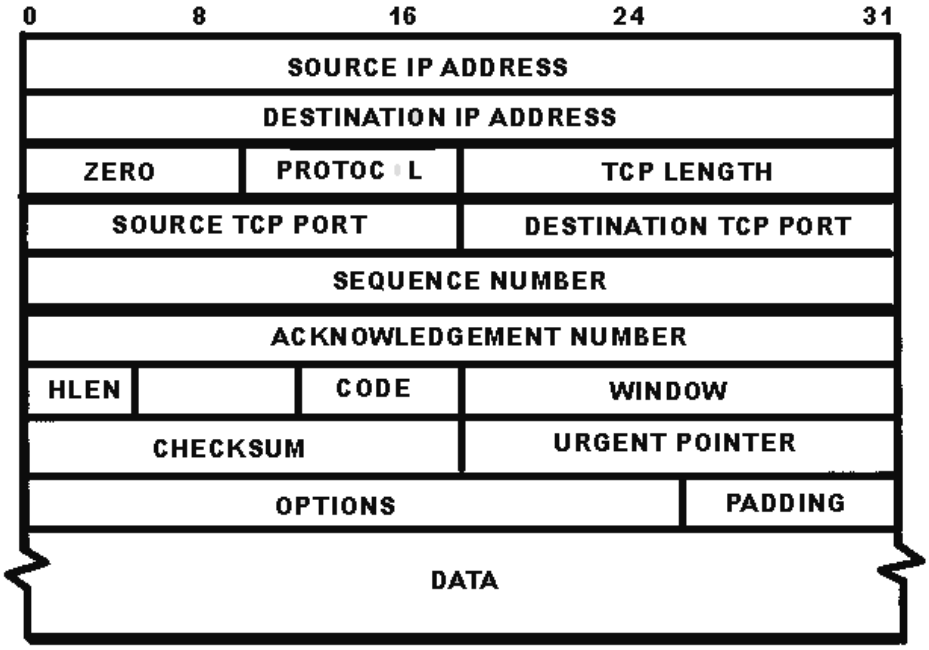


Obr. 6.5 Činnosť stanice TCP

Protokol TCP na rozdiel od protokolu UDP, ktorý prenáša aplikáciou definované bloky dát, poskytuje službu označovanú ako stream (názov zdedený z UNIXU). Dáta sú prenášané ako postupnosť slabík, prípadná štrukturalizácia je plnou záležitosťou aplikácie. Činnosť protokolovej stanice sa dá znázorniť obrázkom.

Jednotlivé znaky predávané aplikáciou k odoslaniu sú ukladané do vyrovnávacej pamäte. Hneď ako je v tejto pamäti dostatočné množstvo dát (alebo vyprší časový limit) stanica vytvorí *TCP segment* a odošle ho v IP pakete protistanici. Spojenie dvoch TCP staníc vytvára dve vzájomne nezávislé transportné kanály v opačných smeroch.

V hlavičke TCP segmentu je vedľa identifikácie TCP socketu odosielateľa a adresáta (podobne ako u UDP protokolu) miesto pre potvrďovanie a služobné informácie. Štruktúru TCP segmentu ilustruje obrázok Obr. 6.6.



Obr. 6.6 Štruktúra TCP segmentu

Potvrdzovanie

Potvrdzovacia schéma protokolu TCP sa opiera o tridsaťdvabitové čísla SEQNO a ACKNO. Neobvykle veľký interval potvrdzovacích čísel dovoľuje jednoducho zvládnuť nepríjemné vlastnosti siete (veľké a premenlivé oneskorenia, straty a zmenené poradie IP paketov) a to aj napriek tomu, že sa potvrdzovacie čísla vzťahujú k postupnosti znakov (a nie ako je to obvyklé u protokolov nižších vrstiev, k postupnosti rámcov alebo paketov).

Pole SEQNO udáva poradové číslo prvého znaku TCP segmentu, pole ACKNO udáva číslo znaku, ktorý očakáva prijímacia strana ako prvý znak ďalšieho TCP segmentu. Potvrdzovanie sa dá označiť ako skupinové potvrdenie segmentu, znamená aj potvrdenie segmentov predchádzajúcich (problémy spojené s modularitou potvrdzovacej schémy tu vzhľadom k veľkému modulu číslovania nehrozí).

Strata TCP segmentu (alebo ľubovoľného IP paketu v ňom) je riešená štandardne - časovým limitom. V takejto sieti ako je Internet je však nemožné nastaviť časový limit koncového staticky (ako sme napríklad zvyknutí u linkových protokolov) jeho nastavenie sa opiera a skutočné odmerané oneskorenia u predchádzajúcich TCP segmentov (metódou pohyblivého priemeru). V súčasnosti je meranie doplňované meraním charakteristickej

odchýlky, využitie tohto údaja pri nastavovaní časového limitu dovoľuje podstatne zvýšiť limit priechodnosti siete.

Jednotlivé polia záhlavia TCP segmentu majú nasledujúcu funkciu:

Tab. 6.2 Funkcie jednotlivých polí záhlavia TCP segmentu

Pole	Funkcia
SRCPORT	šestnásťbitová identifikácia socketu odosielateľa
DESTPORT	šestnásťbitová identifikácia socketu adresáta
SEQNO	tridsaťdvabitové poradové číslo (prvého znaku segmentu)
ACKNO	tridsaťdvabitové potvrdzovacie číslo (očakávaný znak)
HLEN	dĺžka hlavičky v tridsaťdvabitových slovách
CODE	riadiace bity URG, ACK, PSH, RST, SYN, FIN
WINDOW	kredit mechanizmu riadenia toku
CHECKSUM	kontrolný súčet TCP segmentu (vrátane pseudohlavičky IP)
URGPNT	označenie konca urgentných dát
OPTIONS	voliteľné funkcie
PADD	výplňové znaky posledného slova hlavičky
DATA	dáta TCP segmentu

Rovnako ako v protokole UDP aj v protokole TCP čísla socketov jednoznačne identifikujú aplikačné procesy predávajúce si TCP segment. Šestnásťbitová identifikácia socketu dovoľuje aplikačný proces pripojiť k jednému z 65536 TCP socketov. Sady TCP a UDP socketov sú vzájomne nezávislé. Podobne ako UDP sockety sú aj niektoré TCP sockety pevne pridelené vnútorným službám Internetu a UNIXu, výber zo zoznamov týchto socketov uvádza nasledujúca tabuľka.

Tab. 6.3 Čísla socketov niektorých vnútorných služieb Internetu pre protokol TCP

Číslo socketu	Služba	Popis
7	ECHO	Echo
11	USERS	Užívatelia
13	DAYTIME	Čas
20	FTP-DATA	File Transfer Protocol (Dáta)
21	FTP	File Transfer Protocol
23	TELNET	Terminal Emulation
25	SMTP	Simple Mail Transfer Protocol
37	TIME	Čas
42	NAMESERVER	Host Name Server
53	DOMAIN	Domain Name Server
79	FINGER	Finger
102	ISO - TSAP	ISO Transport Access Point
103	X400	X 400 Mail Service
106	X400 SND	X 400 Mail Service
111	SUNRPC	Sun RPC protokol

Riadenie spojenia

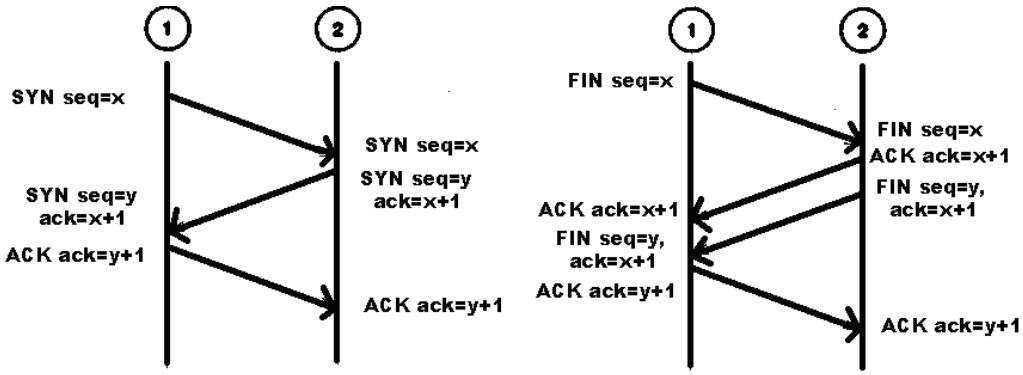
Protokol TCP používa na rozdiel od úplnej väčšiny protokolov linkových a sieťových a na rozdiel od transportných ISO TPx jediný formát transportných paketov - TCP segmentov. TCP segment môže prenášať dáta, potvrdzovať dáta prichádzajúce z opačného smeru, nadväzovať a ukončovať transportné spojenia.

Nevýhodou formátu TCP segmentu je veľká dĺžka záhlavia, ktorá je dôsledkom datagramovej adresácie (plná adresa v hlavičke IP paketu) a použitej metódy potvrdzovania. Pre efektívny prenos dát je potrebné zaistiť, aby pomer dĺžky prenášaného bloku dát k dĺžke záhlavia (40 znakov) bol čo najväčší. Na druhej strane voľba dlhých blokov dát nemusí byť najvhodnejšia. V sieťach , ktoré sú schopné preniesť aj blok o maximálnej dĺžke bez fragmentácie je dlhší blok viac zraniteľný chybami pri prenose a vyžaduje časté opakovanie. Rozdelenie IP paketu (obsahuje TCP segment) na fragmenty môže byť vynútené neschopnosťou siete IP paket o danej dĺžke preniesť alebo snahou znížiť pravdepodobnosť opakovania poškodených paketov na linkovej úrovni. Vzhľadom ku stratégii best-effort na úrovni siete však strata jediného fragmentu znamená nutnosť opakovaného prenosu celého TCP segmentu.

Protokol TCP dovoľuje koncovým užívateľom sa na maximálnej dĺžke TCP segmentu dohodnúť, v úvahu sú brané ich vlastné obmedzenia, ale taktiež spôsob prepojenia staníc. Pre stanice na jedinej lokálnej sieti je rozumné využiť tej dĺžky segmentu, ktorú lokálna sieť môže preniesť. Pre stanice spojené cez radu routerov je treba zvoliť vhodný kompromis, maximálna dĺžka TCP segmentu je tu súčasnými implementáciami obmedzená na 536 znakov dát.

Pre otvorenie spojenia je využívaný trojfázový protokol, jeho funkciu ilustruje obrázok. Pre otvorenie je využívaný bežný TCP segment s nastaveným príznakom SYN. Dvojité potvrdenie (súčasne so zmenou sekvenčných čísel, ktorá na rozdiel od linkových protokolov neštartujú pri otvorení od nuly) zabraňuje falošnému otvoreniu TCP spojení zblúdilými TCP segmentmi, ktoré boli po vypršaní limity opakovane vyslané.

Uzavretie spojenia TCP segmentom s nastaveným príznakom FIN odpovedá obrázku Obr. 6.7. Každý z oboch jednosmerných kanálov je potrebné uzavrieť samostatne. Strata niektorého TCP segmentu pri uzatváraní spojenia je riešená časovými limitmi, korektné algoritmické riešenie nie je možné.



Obr. 6.7 Otváranie a zatváranie transportného spojenia TCP

Riadenie Toku

Na pole WINDOW sa môžeme pozerať ako na okienko potvrdzovacej schémy, jeho základné použitie je však v riadení toku. Udáva množstvo pamäte, ktorú má prijímacia strana k dispozícii pre skladanie TCP segmentov. Ide o kredit, ktorý v spojení s potvrdzovacím

číslom udáva koľko dát smie vysielacia strana odoslať aby nespôsobila protistanici problémy s pamäťou.

Ďalšou z hľadiska chovania siete ako celku veľmi podstatnou funkciou riadenia toku je *ochrana* siete proti *zahlteniu* (podpora jej udržania v oblasti efektivity).

Mechanizmus sa opiera o informácie o stratách IP paketov, tie sú príznakom problemov v sieti. Časový limit pre opakovanie TCP segmentu je nastavovaný na násobky pôvodnej hodnoty pri opakovanej strate IP paketu, súčasne je však znižované množstvo dát, ktoré smie TCP protokol do siete odoslať (pod úroveň definovanú poľom WINDOW).

6.4 Protokoly relačnej vrstvy

Relačná vrstva vyvára logické rozhranie pre aplikačné programy, ktoré používajú služby siete. Definuje spôsob komunikácie programov a užívateľský pohľad na komunikačný kanál. Príklady protokolov relačnej vrstvy:

- NetBIOS,
- NetBEUI (NetBIOS Enhanced User Interface) – rozšírené užívateľské rozhranie NetBIOS-u,
- NFS Network File System,
- SMB Server Message Block,
- atď.

6.5 Protokoly prezentačnej vrstvy

Prezentačná vrstva transformuje prenášané dáta - zaisťuje prevody kódov a formátov dát pre nekompatibilné počítače, kompresiu prenášaných dát a konečne aj utajovanie prenášaných dát. Príklady protokolov prezentačnej vrstvy:

- TLS (Transport Layer Security),
- JPEG (Joint Photographic Experts Group),
- MP3 (Moving Picture Experts Group Audio Layer 3),
- MPEG (Moving Picture Experts Group),
- atď.

6.6 Protokoly aplikačnej vrstvy

Aplikačná vrstva je konečne vrstvou štandardných aplikačných programov, ktoré sieť využívajú.

6.6.1 Aplikačné rozhranie, Socket

Aplikácie v sieti Internet sa opierajú o filozofiu spolupráce označovanú ako klient - server. Server je proces ktorý poskytuje nejakú službu (anonymnému) klientovi. Klient o túto službu žiada zaslaním správy, žiadosti serveru, server prevedie požadovanú činnosť a odpovie správou, odpoveďou.

Aplikácie v operačnom systéme UNIX (ale stále aj viac i v iných operačných systémoch) sú dostupné prostredníctvom socketov. Mechanizmus bol pre UNIX navrhnutý a implementovaný na univerzite Berkeley je preto často označovaný ako Berkeley sockety. Okrem protokolovej architektúry Internetu podporuje i protokolovú archotektúru firmy Xerox (XNS - Xerox Internet Protocol).

Základným prvkom rozhrania je socket ktorý dovoľuje jednému aplikačnému programu v UNIXe spojiť sa s inými aplikačným programom v UNIXe prostredníctvom siete. Po jeho otvorení používajú aplikačný program rovnaké primitívy ako pre prácu so súbormi a perifériami teda operácia *write* pre vyslanie a *read* pre príjem dát.

Otvorenie socketu

Otvorenie socketu cez ktorý bude aplikačný program komunikovať zaisťuje funkcia:

```
result = socket(af, type, protocol);
```

Parameter *af* definuje triedu protokolu, preddefinované sú konštanty AF JNET - Internet, AF - PUP - Xerox XNS a AF - UNIX - vnútorné spojenie v UNIXe . Parameter *type* určuje typ protokolu, preddefinované sú konštanty SOCK - STREAM - pre protokol TCP, SOCK - DGRAM pre protokol UDP v Internete a SOCK - ROW pre priamy prístup k protokolu IP. Konečne posledný parameter *protokol* špecifikuje konkrétny protokol pre Internet. Sú tam preddefinované konštanty SOCK - STREAM pre protokol TCP v Internete, SOCK - DGRAM pre protokol UDP v Internete a SOCK ROW pre priamy prístup k protokolu IP.

Zadanie Lokálnej adresy

Predchádzajúce funkcie socket vytvoria potrebné dátové štruktúry , ale bez väzby na konkrétnu lokálnu a vzdialenú adresu. Väzbu na lokálnu adresu zaisťuje funkcia:

```
bind(socket, localaddr, addrlen);
```

ktorá má ako parameter adresu a dĺžku štruktúry obsahujúcu lokálny TCP alebo UDP adresu.

Žiadosť klienta o spojenie zo serverom

Adresu vzdialeného servera zadávame socketu volaním funkcie:

```
connect(socket, remoteaddr, addrlen);
```

ktorá má ako parameter adresu a dĺžku štruktúry obsahujúcu vzdialenú TCP alebo UDP adresu.

Určenie dĺžky frontu serveru

Server môže byť schopný obslúžiť postupne alebo aj súčasne (napr. odštiepením vlastnej obsluhy od procesu prijímacieho žiadosti) viac klientov. Potom je nutné zaistiť, aby sa žiadosti ďalších procesov nestratili zatiaľ čo jedna je obsluhovaná. Dĺžku fronty spojenú s príslušným socketom zadávame volaním funkcie:

```
listen(socket, qlength);
```

kde *qlength* je počet žiadostí, ktoré je fronta schopná uložiť.

Aktivácia serveru

Server očakáva volanie vzdialeného klienta volaním funkcie:

```
newsocket = accept(socket, addr, addrlen);
```

po prijatí volania funkcie vracia adresu a dĺžku štruktúry obsahujúcu vzdialenú TCP alebo UDP adresu a deskriptor nového socketu, na ktorom bude komunikácias klientom pokračovať. Starý socket zostáva k dispozícii pre príjem volania ďalšieho klienta.

Prenos dát

Vyslanie dát pre spojenie medzi klientom a serverom zaisťujú funkcie:

```
write(socket, buffer, length);
send(socket, buffer, length, flags);
sendto(socket, buffer, length, flags, destaddr, addrlength);
```

Všetky funkcie požadujú prenos bloku dát o zadanej dĺžke Funkcia send dovoľuje navia definovať príznaky, ktoré napríklad dovolia odoslať dáta ako prednostné (urgent). Funkcia send to dovoľuje odoslať správu špecifikovanému adresátovi po sockete, ktorý nebol s určitým náprotivkom spojený funkciou connect. K dispozícii sú navia c funkcie dovoľujúce odoslať postupnosti správ *writev* a *sendmsg*.

Príjem dát zo spojenia medzi klientom a serverom zaisťujú funkcie:

```
read(socket, buffer, length );
recv(socket, buffer, length, flags );
recvfrom(socket, buffer, length, flags, srcaddr, addrlength);
```

Všetky funkcie zadávajú požiadavku na prenos bloku dát do vyrovnávacej pamäte o zadanej adrese a dĺžke. Funkcia recv dovoľuje naviac definovať príznaky, ktoré napríklad dovolia odoslať a prehliadať dáta bez ich odoberania z kanálu. Funkcia recvfrom dovoľuje prijať správu zo socketu, ktorý nebol s určitým náprotivkom spojený funkciou connect. K dispozícii sú naviac funkcie dovoľujúce prijať postupnosti správ *readv* a *recvmsg*.

Uzavretie socketu

Uzavretie socketu zaisťuje funkcia:

```
close(socket);
```

6.6.2 Telnet

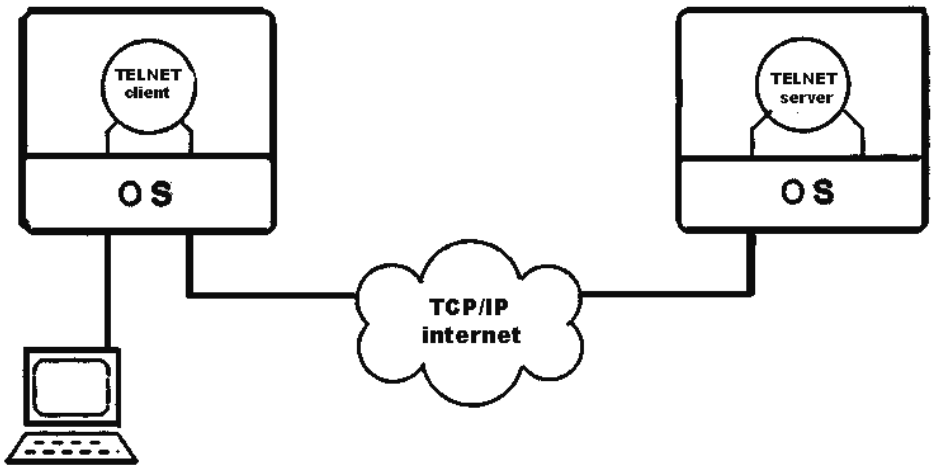
Protokol Telnet sprostredkuje možnosť prihlásiť sa zo vzdialeného počítača pre interaktívnu prácu na inom počítači. Telnet používa TCP spojenie tak, že pracuje na princípe klient- server : klient zriadi spojenie so serverom a posiela mu jednotlivé znaky tak, ako sú písané na klávesnici klienta, server ich potom spracováva akoby boli písane na jeho lokálnom termináli a vzniknuté výstupné znaky opäť pošle klientovi, ktorý ho potom lokálne zobrazí.

Táto služba je transparentná, lebo užívateľovi na strane klienta sa všetko javí, akoby pracoval na vzdialenom počítači - serveru.

Telnet ponuka tri základne služby :

- definuje *sieťový virtuálny terminál* (NVT : Network Virtual Terminal), ktorý poskytuje jednotné štandardné rozhranie pre prístup ku vzdialeným systémom.
- umožňuje klientovi i serveru vzájomne rovnocenne *vyjednávanie voliteľných módov* (option negotiation) z určitej preddefinovanej štandardnej množiny, takže sa dá používať viac služieb, ako ponúka základná definícia NVT

- považuje obidva konce spojenia za *symetrické*, nerozlišuje medzi terminálmi a procesmi, takže dovoľuje aby sa klientom stal akýkoľvek proces.



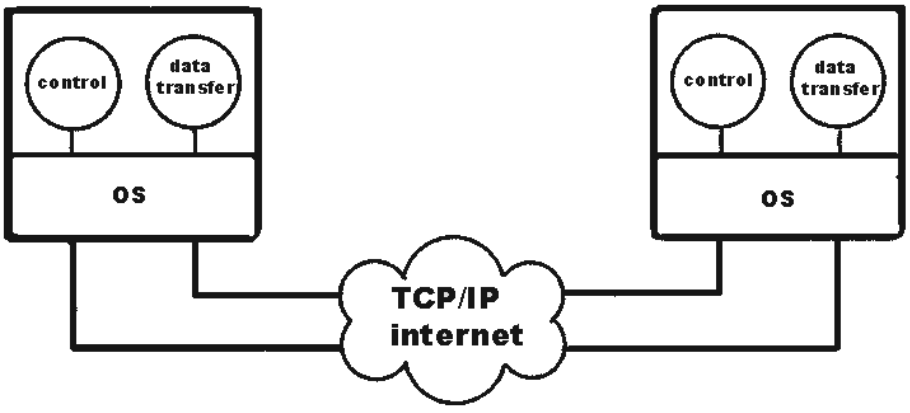
Obr. 6.8 Štruktúra virtuálneho terminálu TELNET

Virtuálny terminál TELNET pracuje po znakoch, dovoľuje prenášať na vzdialený systém riadiace znaky a zabezpečuje ich preklad (napr. znaky ukončujúce riadok, stránku, tabulátory, ap. Môžu byt implementované na rôznych systémoch rôzne, TELNET musí zaistiť potrebný preklad). Musí byt schopný prenášať ku vzdialenému počítaču riadiace znaky (napr.^ C, editačne znaky) vo forme, vyžadovanej týmto počítačom. Musí zvládnuť sedem i osembitový kod znakov.

Podobnú funkciu ako TELNET ma i rlogin dostupný v BSD UNIXu.

6.6.3 FTP

Prenosy súboru zo vzdialeného systému alebo na vzdialený systém prostredníctvom siete Internet zaisťujú procesy aplikácie FTP (File Transfer Protocol). Jej štruktúra odpovedá obrázok Obr. 6.9.



Obr. 6.9 Štruktúra prenosu súboru FTP

Základom protokolu je princíp klient - serve. Medzi základne vlastnosti protokolu patri :

- *interaktívny prístup*: väčšina implementácii poskytuje užívateľské rozhranie pre interaktívnu prácu so vzdialenými servermi.
- *špecifikácia formátu*: klient môže špecifikovať typ a formát uloženia dát (binárny, textový ASCII alebo EBCDIC, a pod).
- *riadenie prístupu*: klient musí prejsť prihlasovacou procedúrou na serveri (užívateľské meno a heslo, špeciálne užívateľské meno anonymous pre kohokoľvek

Aplikácia je tvorená riadiacimi procesmi, s ktorými komunikuje užívateľ a procesy zaisťujúce vlastný prenos. Užívateľ ma k dispozícii celu škálu príkazov dovoľujúcich sa pohybovať v adresároch lokálneho i vzdialeného počítača a špecifikovať jednotlivé i skupinové prenosy. Dajú sa prenášať textové súbory i binárne obrazy. Činnosť programu FTP sa opiera o protokol FTP.

TFTP (Trivial File Transfer Protocol)

Čo najjednoduchší protokol dovoľujúci prenos súborov, aby jeho implementačný software mohol byt umiestnený v ROM pamäti bezdiskových počítačov a typicky umožňoval dvojfázovú zavádzaciu procedúru : získanie základných (sieťových) informácii pomocou *BootP* a potom prenos vlastného nevyhnutného software pomocou *TFTP*.

Prenos súborov po blokoch pevnej dĺžky (512 bytov) nad UDP. Server čaká na potvrdenie každého vyslaného bloku pred vyslaním ďalšieho.

6.6.4 NFS

Alternatívou k prenosu súborov pred ich lokálnym použitím je sprístupnenie vzdialených súborov priamo aplikačnému programu, prevedenie každej žiadosti o prístup k súboru na správu predanú sieťou vzdialenému severu, ktorý žiadosť vykoná a vráti ďalšou správou výsledky svojej prace.

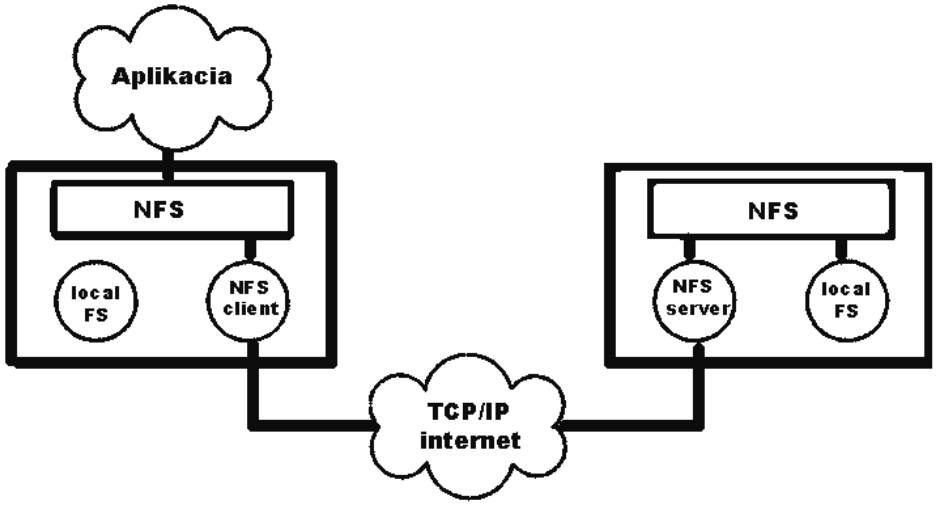
NFS je obvykle začlenené priamo do operačného systému a pracuje opäť v režime klient - server, server ponuka svoje lokálne súborové systémy k zdieľaniu cez sieť a klient ich využíva. Z hľadiska aplikačných programov i z hľadiska operátora je potom prístup ku vzdialenému súboru transparentný (nie je treba špecifikovať vzdialený počítač).

Takúto funkciu zaisťuje *NFS* (*Network File System*) firmy Sun Microsystems. Štruktúru NFS a spôsob prepojenia adresára ilustruje obrázok Obr. 6.10.

Systém NFS sa opiera o prídavné protokoly *RPC* (*Remote Procedure Call*) a *XDR* (*eXternal Data Representation*).

Prvý z protokolov zjednodušuje komunikáciu medzi procesmi systému (a je k dispozícii i pre programy užívateľa). Umožňuje, aby proces bežiaci na jednom počítači mohol zavolať vzdialenú procedúru, ktorá ich potom vykoná na vzdialenom počítači, a jej výsledok je opäť prenesený naspäť volajúcemu procesu.

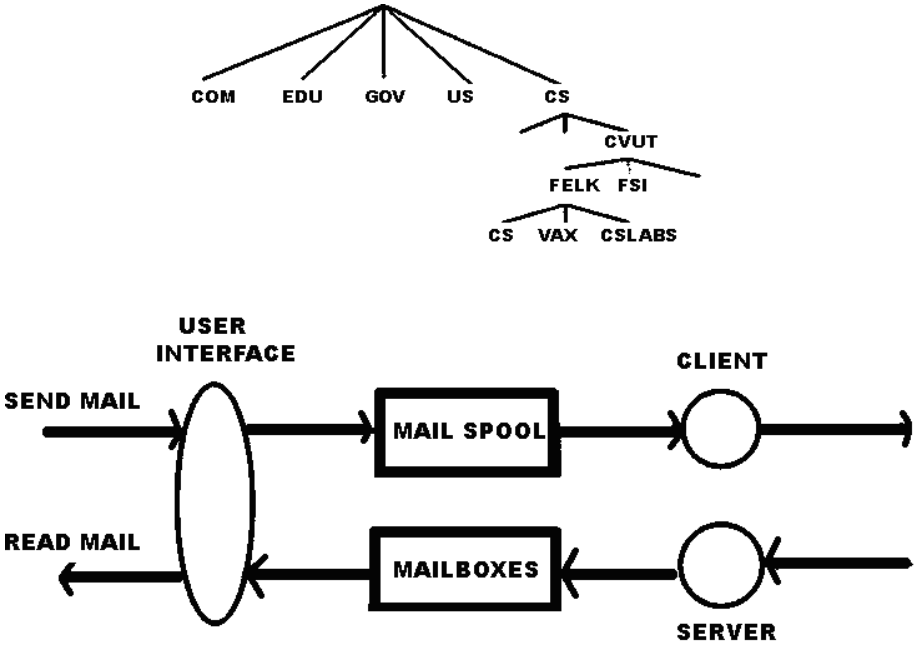
Druhy definuje na počítači nezávislú reprezentáciu dát, ktorá je používaná pre prenos dát po sieti medzi ľubovoľnými počítačmi (vysielajúci počítač urobí pred vysielaním dát ich konverziu do XDR a prijímací počítač opäť konvertuje dáta prijate XDR do svojho formátu) a uľahčuje tak komunikáciu v heterogénnych systémoch.



Obr. 6.10 Štruktúra NFS

6.6.5 SMTP

Elektronická pošta je snáď najširšou používanou aplikáciou Internetu. Štruktúra služby popisuje obrázok Obr. 6.11.



Obr. 6.11 Štruktúra služby Mail

Elektronická pošta zaisťuje priame dodávanie sprav medzi systémami v Internete (vysielajúci klient sa priamo spojuje so vzdialeným prijímajúcim serverom).

Adresát je pravidelne označovaný svojim užívateľským menom a adresou systému. Používať numerické adresy by bolo (a to nie len pre elektronickú poštu ale taktiež pre TELNET, FTP a ďalšie služby) veľmi nepraktické, Internet preto zahrňuje mechanizmus adresárov- počítača s programom Domain Name Server.

Obecný formát adresy elektronickej pošty je

```
lokalna_časť@doménove _meno
```

Resp. zjednodušene osoba@pocitac, kde osoba je užívateľské meno adresáta a počítač je doménové meno počítača na ktorom ma adresát konto.

Existencia brán medzi Internetom a inými sieťami dovoľuje doručovanie elektronickej pošty do a z iných sieti (ako napríklad EARN/Bitnet, UUCP, CompuServe, Fido a pod.).

Elektronická pošta sa opiera o spoľahlivý TCP protokol, nad ktorým je vystavaný protokol *SMTP* (*Simple Mail Transfer Protocol*) s plne textovými riadiacimi správami. Komunikácia SMTP procesov sa dá preto veľmi ľahko sledovať (čo je dôležité pre riešenie bežných prevádzkových problémov).

V prípade prenosových problémov (napr. nedá sa vytvoriť spojenie) sa periodicky vykonávajú opakovane pokusy a až potom po určitej dobe (napr.3 dni), je pošta vrátená odosielateľovi ako nedoručiteľná. Každá poštová sprava potom ostáva na odosielateľovom počítači, pokiaľ nie je úspešne doručená adresátovi.

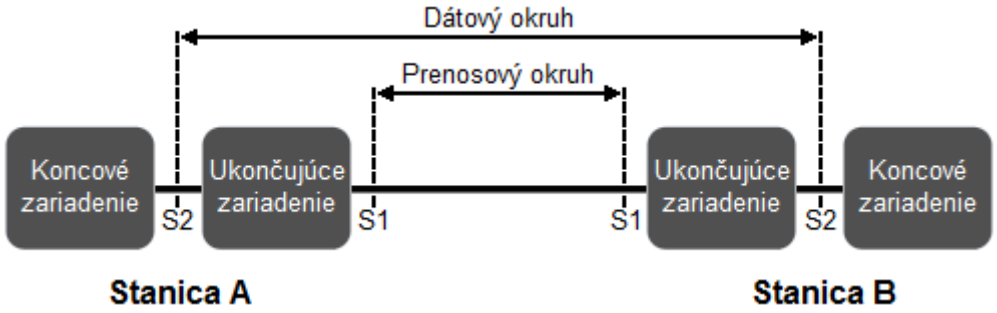
7 Rozhrania

Táto kapitola sa venuje niektorým počítačovým rozhraniam a to trom štandardným rozhraniam, technologickým rozhraniam a trom vstupno-výstupným obvodom. Ako štandardné rozhrania kapitola popisuje RS-232, Centronics a USB. Technologické rozhrania sa venujú niektorým vstupno-výstupným kartám od firmy Advantech a to kartám PCL-812, PCL818 a PCI-1730. V tejto kapitole sú rozobraté iba tri vstupno-výstupné obvody, ktorými sú čítač-časovač, digitálno-analógový a analógovo-digitálny prevodník.

7.1 Štandardné rozhranie RS-232C

Štandardné rozhranie RS-232C slúži na prepojenie 2 počítačov, alebo na prepojenie periférneho zariadenia s rozhraním RS-232C s počítačom. RS-232C je *asynchrónne sériové rozhranie*. Pojmom asynchrónne rozhranie sa rozumie to, že dané rozhranie neobsahuje synchronizačné hodiny a komunikácia beží pomocou potvrdzovacích signálov. Sériové rozhranie znamená, že každý jeden bit je poslaný za sebou (sériovo – jednotlivé informačné elementy informačnej jednotky sa prenášajú postupne), takže dáta sa vysielajú jedným vodičom (digitálne) a prijímajú taktiež jedným vodičom.

Ak sa spoja dve miesta dvojbodovým dátovým spojom a prenos dát medzi týmito bodmi sa deje sériovo, tak sa hovorí o *sériovom dátovom okruhu* (Obr. 7.1). Dátový spoj sa skladá z *koncových zariadení* prenosu dát. Dve stanice si navzájom vymieňajú dáta pomocou *dátového okruhu*. Dátový okruh tvorí *prenosový okruh* s dvoma *ukončujúcimi zariadeniami* prenosu dát. Vzdialenosť staníc nepriamo určuje typ ukončujúceho zariadenia a druh prenosového okruhu. Pre veľmi malé vzdialenosti (15 m) sa môže dátový okruh vynechať a obe koncové zariadenia pripojiť priamo (nulový modem). Mechanické prevedenie rozhrania používa konektor CANNON s 25 pinmi (DB-25), alebo CANNON s 9 pinmi (DB-9). Koncové zariadenie má konektor so špičkami (samec) a ukončujúce zariadenie konektor s dutinkami (samica).



Obr. 7.1 Sériový dátový okruh

7.1.1 Varianty

V praxi sa obvykle nepoužívajú všetky signály tohto rozhrania. Rozlišuje sa totiž malá, stredná a veľká skupina obvodov tohto rozhrania. Každá z nich môže zaisťovať obojsmerný prenos, ale aj striedavý (striedanie prijímača a vysielača) jednosmerný, alebo len jednosmerný. Vzájomne sa líšia počtom použitých signálov a hlavne využitím vedľajšieho sériového kanálu. Pri čítaní ďalších odstavcov sa odporúča sledovať aj tabuľka Tab. 7.1.

Malý variant sériového rozhrania RS-232 umožňuje jednokanálový asynchrónny prenos dát. Neumožňuje však zistiť stav zariadenia a neumožňuje technickými prostriedkami zaistiť kvitovanie (potvrdzovanie). To sa prevádza len programovo, na úrovní opakovania prenesených znakov, alebo potvrdením vybraným znakom. Používa sa väčšinou k jednosmernému alebo striedavému prenosu, a to v najjednoduchších aplikáciách pri prepojení medzi dvoma koncovými zariadeniami.

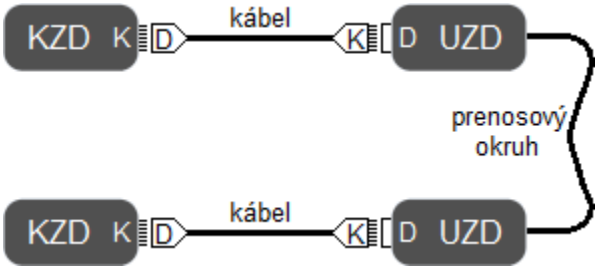
Najčastejšie sa toto rozhranie používa v *strednom variante*. Tá je vhodná pre jednokanálový prenos dát. Má prostriedky pre zistenie stavu prenosu aj ku kvitovaniu. Používa sa pre spojenie na väčšie vzdialenosti s modemom a na krátke vzdialenosti priamo medzi dvoma koncovými zariadeniami.

Veľký variant sériového rozhrania má všetky možnosti dané týmto štandardným rozhraním a používa ju koncové zariadenie typu komunikačný procesor pracujúci s pevným komunikačným protokolom.

Tab. 7.1 Rozloženie signálov rozhrania RS-232 na konektory a varianty

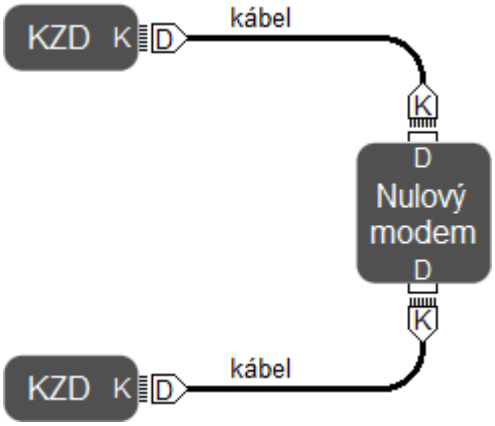
Označenie vývodov		**Konektor CANNON**		**Variant**		
V.24	**RS-232 C**	**9 pinov**	**25 pinov**	**malá**	**stredná**	**veľká**
101	FG	-	1	✓	✓	✓
102	SG	5	7	✓	✓	✓
103	TD	3	2	✓	✓	✓
104	RD	2	3	✓	✓	✓
105	RTS	7	4	×	✓	✓
106	CTS	8	5	×	✓	✓
107	DSR	6	6	×	✓	✓
108	DTR	4	20	×	✓	✓
109	DCD	1	8	×	✓	✓
114	TC	-	15	×	×	✓
115	RC	-	17	×	×	✓
118	STD	-	14	×	×	✓
119	SRD	-	16	×	×	✓
120	SRTS	-	19	×	×	✓
121	SCTS	-	13	×	×	✓
122	SDCD	-	12	×	×	✓
125	RI	9	22	×	✓	✓

Počet žíl kábla je závislý na použitej variante (skupine obvodov). Napríklad pre strednú skupinu je kábel 12-žilový s tienením. Dĺžka jedného kábla nemá presahovať 15 m. Pri jeho počítačovej aplikácií musíme najprv stanoviť charakter koncových a ukončovacích zariadení. Riadime sa zásadou, že zariadenie, ktoré prijíma alebo vysiela dáta prenosu je zariadením typu KZD (koncové zariadenie). Zariadenie, ktoré podľa dát mení parametre signálu prenášaných prenosovým okruhom, je zariadenie typu UZD (ukončovacie zariadenie). Charakteru zariadenia musí vyhovovať druh konektora (dutinky označené na obrázkoch ako D a kolíky označené ako K). Tento typ prepojenia koncových zariadení je na obrázku Obr. 7.2.



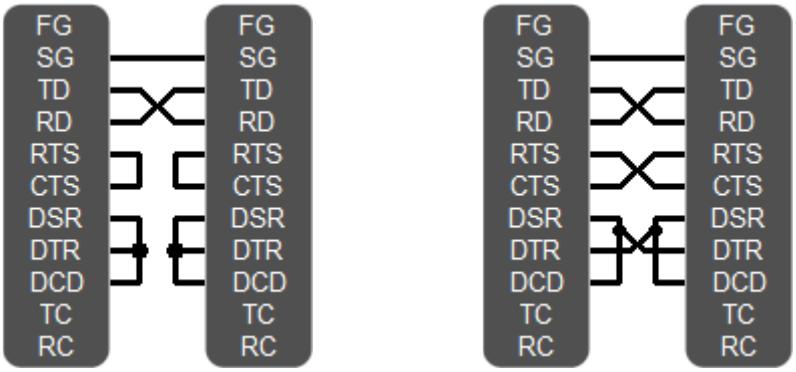
Obr. 7.2 Prevedenie sériového rozhrania dlhšieho ako 15 m

Pretože väčšina počítačových aplikácií sériového rozhrania vystačí s prepojovacím káblom do dĺžky 15 m, je zbytočné zostavovať celý dátový okruh. Obe zariadenia je možné spojiť priamo – opäť štandardnými káblami a tzv. nulovým modemom. Ten adaptuje dátové a riadiace obvody medzi dvoma KZD (Obr. 7.3).



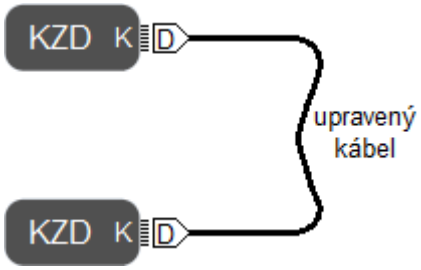
Obr. 7.3 Prevedenie sériového rozhrania kratšieho ako 15 m

Nulový modem má dve základné prevedenia. Troj-drátové zapojenie ukazuje obrázok Obr. 7.4 v ľavo, úplné zapojenie je na obrázku Obr. 7.4 v pravo.



Obr. 7.4 Základné prevedenia nulového modemu (v ľavo: troj-drátový nulový modem, v pravo: úplný nulový modem)

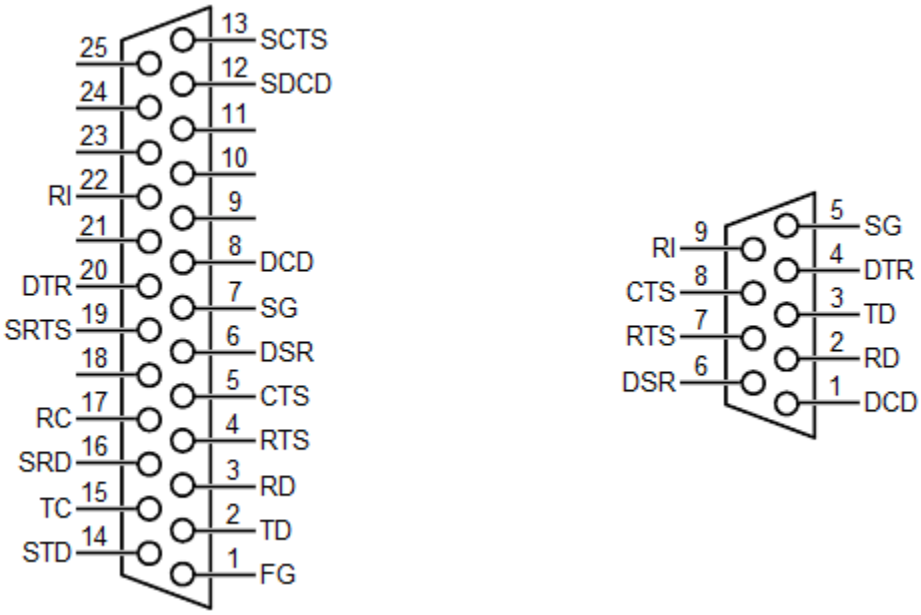
Podľa týchto obrázkov (Obr. 7.4) je zrejmé, že nulový modem sa zo zapojenia môže vynechať. A to takým spôsobom, že sa koncovky káblov prekrižujú presne podľa obrázka Obr. 7.4. V prípade, ak máme takýto špeciálny kábel môžeme prepojiť koncové zariadenia priamo (Obr. 7.5).



Obr. 7.5 Priamo prepojené koncové zariadenia pomocou upraveného kábla kratšieho ako 15 m

7.1.2 Konektory

Všetky varianty, čiže malý, stredný aj veľký môže využívať 25 pinový CANNON konektor. Vodiče (piny) ktoré daný variant nepotrebuje nebudú využité, alebo zapojené. Menší 9 pinový konektor sa dá použiť pri malej a strednej variante. Rozloženie signálov na špičkách 25 vodičového a 9 vodičového konektora CANNON uvádza obrázok Obr. 7.6.



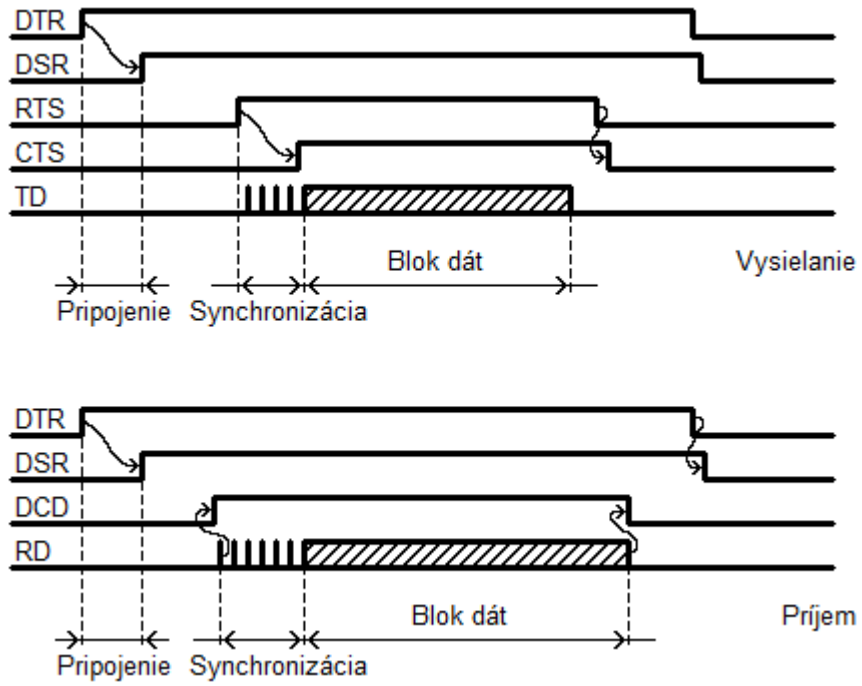
Obr. 7.6 Rozloženie pinov na konektoroch RS-232

Význam a popis jednotlivých pinov z obrázka Obr. 7.6 je uvedený v nasledujúcej tabuľke (Tab. 7.2).

Tab. 7.2 Význam pinov na konektoroch RS-232C

Signál	Význam	Popis
FG	Frame ground	ochranná zem, 9 pinová verzia ktorá obsahuje FG má na vonkajšej časti konektora (na ráme konektora) ochrannú zem.
SG	Signal ground	signálová zem
TD	Transmit data	vysielané dáta (používa sa aj označenie Tx)
RD	Receive data	prijímané dáta (používa sa aj označenie Rx)
RTS	Request to send	výzva k vysielaniu
CTS	Clear to send	pohotovosť (pripravenosť, čakanie) k vysielaniu
DSR	Data set ready	pohotovosť ukončujúceho zariadenia
DTR	Data terminal ready	pohotovosť koncového zariadenia
DCD	Data carrier detect	detektor nosného signálu
TC	Transmitted clock	vysielacia časová základňa
RC	Received clock	prijímacia časová základňa
STD	Secondary transmitted data	dáta vysielané vedľajším kanálom
SRD	Secondary received data	dáta prijímané z vedľajšieho kanálu
SRTS	Secondary request to send	výzva k vysielaniu vedľajšieho kanálu
SCTS	Secondary clear to send	pohotovosť k vysielaniu vedľajšieho kanálu
SDCD	Secondary data carrier detect	Detektor nosného signálu vedľajšieho kanálu
RI	Ring indicator	indikátor volania

Na obrázku Obr. 7.7 je zakreslený charakter vybraných signálov rozhrania RS-232C pri vysielaní a pri príjme dát.



Obr. 7.7 Charakter vybraných signálov pri vysielaní (hore) a pri príjme (dole) dát

7.1.3 Obvody

Prvým základným stavebným kameňom adaptéru sériového rozhrania v počítačoch bol obvod 8250 neskôr bol nahradený obvodmi 16450, 16550, 16650, 16750 a ďalšími. Tieto obvody sú spätne kompatibilné, takže obvod 16550 viete nastaviť a naprogramovať rovnako ako obvod 8250. Tento UART (Universal Asynchronous Receiver Transmitter) zaisťuje základné funkcie asynchrónneho komunikačného obvodu, ako sú serializácia a deserializácia dát, doplnenie bitov o asynchrónny rámec (štart bit, príslušný počet stop bitov a prípadný paritný bit).

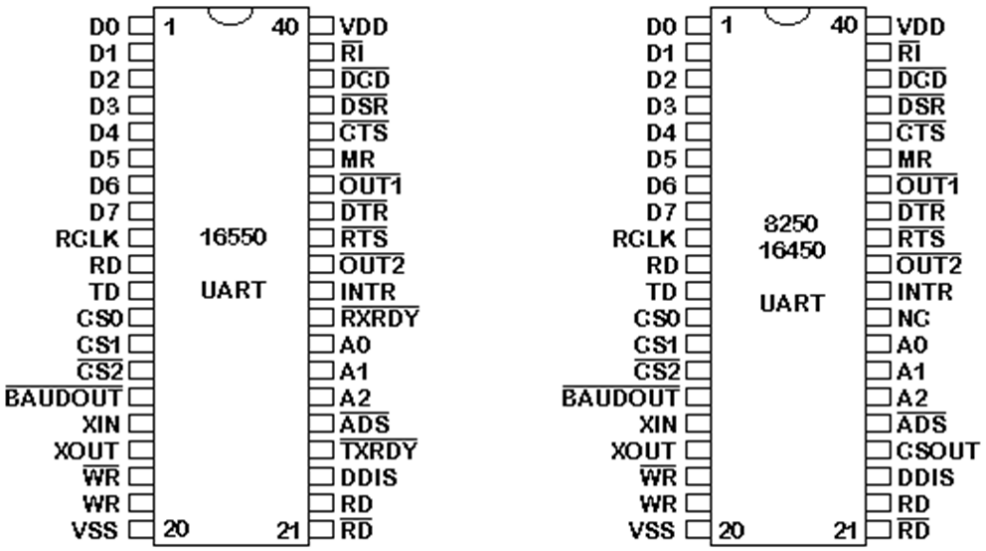
Pri nasadení sériového rozhrania UART 8250 sa v počítačoch nachádzali najčastejšie dva porty označené ako COM1 a COM2, alebo len jeden COM1. Dnes, ak počítač obsahuje sériové rozhranie, tak má iba jeden port COM1. Samozrejme priemyselné verzie počítačov môžu obsahovať týchto portov viac. Porty COM1 a COM2 mali pevne zadané bázové adresy.

Bázová adresa je adresa vstupno-výstupného zariadenia (viď kapitolu 2.2), konkrétne prvého registra periférie. Čiže ak COM1 má vstupno-výstupné registre (v úzkom zmysle zariadenia) od adresy 3F8H po adresu 3FEH, tak jeho bázová adresa je 3F8H. Bázová adresa COM2 je 2F8H.

Štart bit je označenie začiatku posielanej informácie (slova) po sériovej linke ktorá môže mať 5 až 8 bitov (podľa nastavenej dĺžky slova) po týchto bitoch nasleduje jeden alebo dva *stop bity* podľa nastavenia linky. Takže celá posielaná informácia pomocou obvodu UART 8250 môže mať od 7 bitov (1 štart bit + dáta 5 bitov + 1 stop bit) do 11 bitov (1 štart bit + dáta 8 bitov + 2 stop bity).

Dĺžka slova je veľkosť jednej informácie, ktorá sa posiela medzi štart bitom a stop bitom. Dĺžka slova sa udáva v bitoch. Inými slovami dĺžka slova je počet prenášaných bitov, ktoré prenášajú informáciu (bez štart bitu a stop bitov).

Význam signálov a ich rozloženie na obvodoch UART 8250, 16450 a 16550 uvádza obrázok Obr. 7.8.



Obr. 7.8 Rozloženie pinov na obvodoch UART 8250, 16450 a 16550

Význam jednotlivých signálov na kontaktoch obvodov UART 8250, 16450 a 16550 sú uvedené v tabuľke Tab. 7.3. Dva signály ktoré sú v spomenutej tabuľke (Tab. 7.3) kurzívou (šikmým písmom) popisujú signály, ktoré sú iba na obvode UART 16550.

Tab. 7.3 Význam signálov na kontaktoch obvodov UART 8250, 16405 a 16550

Kontakt (pin)	**Signál**	**Význam**
1 – 8	D0 – D7	Obojsmerná dátová zbernica (obojsmerná).
9	RCLK	Hodiny vstupného signálu (vstup).
10	RD (Rx, SIN)	Sériový vstup dát (vstup).
11	TD (Tx, SOUT)	Sériový výstup dát (výstup).
12 – 14	CS0 – CS2	Výber obvodu (vstup).
15	BAUDOUT	Hodinová frekvencia vysielaných dát (výstup).
16, 17	XIN, XOUT	Pripojenie kryštálového rezonátora alebo hodinového signálu (XIN – vstup a XOUT – výstup).
18, 19	WR	Zápis dát alebo stavového slova (vstup).
20	VSS	Pripojenie k uzemneniu (výstup).
21, 22	RD	Čítanie dát alebo stavového slova (vstup).
23	DDIS	Indikácia čítania dát (výstup).
24	CSOUT	Indikácia výberu obvodu (výstup).
24	*TXRDY*	Vysielanie pripravené (negovaný výstup).
25	ADS	Aktívna úroveň značí platnú adresu (vstup).
26 – 28	A0 – A2	Adresovanie vnútorných registrov obvodu (vstup).
29	NC	Nevyužitý kontakt (Not Connected).
29	*RXRDY*	Prijímanie pripravené (negovaný výstup).
30	INTR	Žiadosť o prerušenie (výstup).
31, 34	OUT2, OUT1	Programovateľné výstupy (výstup).
32	RTS	Pohotovosť (pripravenosť, čakanie) k vysielaniu (negovaný výstup).
33	DTR	Koncové zariadenie je pripravené (negovaný výstup).
35	MR	Nulovanie obvodu (vstup).
36	CTS	Pripravenosť k vysielaniu (negovaný vstup).
37	DSR	Indikácia pripravenosti k prevádzke (negovaný vstup).
38	DCD	Detektor nosného signálu (negovaný vstup).
39	RI	Prichádzajúce volanie (negovaný vstup).
40	VDD	Pripojenie k napájaniu + 5 V (vstup).

7.1.4 Registre

Obvod 8250 obsadzuje celkom 10 užívateľovi (programátorovi) prístupných 8-bitových registrov (16550 obsahuje navyše FIFO register). Niektoré z nich sú určené pre zápis, niektoré pre čítanie a niektoré pre zápis aj čítanie. Obvod využíva len 7 adries, ale sú rozdelené podľa toho či sa z nich číta alebo zapisuje, alebo podľa bitu DLAB. Zápisom hodnoty do riadiacich registrov môže užívateľ (programátor) konfigurovať obvod podľa konkrétnych požiadaviek aplikácie (počet prenášaných bitov, počet stop bitov, povolenie parity, nastavenie prenosovej rýchlosti, povolenie prerušovacieho zdroja). Čítaním stavových registrov sa zase zisťuje aktuálny stav prenosu (stav linky, prípadné chyby, stav modemu). Prehľad registrov je v tabuľke Tab. 7.4.

Tab. 7.4 Prehľad 11-tich registrov obvodu UART (8250, 16450, 16550)

COM1	COM2	DLAB	Význam vnútorných registrov pri	
			Čítaní	zápise
3F8H	2F8H	0	Prijaté dáta.	Vyslané dáta.
3F8H	2F8H	1	Deliteľ hodinovej frekvencie (LSB).	
3F9H	2F9H	0	Maska prerušenia.	
3F9H	2F9H	1	Deliteľ hodinovej frekvencie (MSB).	
3FAH	2FAH	×	Identifikácia prerušenia.	FIFO register (iba od 16550).
3FBH	2FBH	×	Riadenie linky.	
3FCH	2FCH	×	Riadenie modemu.	
3FDH	2FDH	×	Stav linky.	
3FEH	2FEH	×	Stav modemu.	

Baud je v telekomunikáciách a informatike, jednotka *signálnej rýchlosti*, čo predstavuje počet zmien stavu prenosového média za sekundu v modulovanom signáli. Termín baud sa niekedy používa na označenie *bitov za sekundu* čo je *prenosová rýchlosť*, táto zámena pojmov sa dá použiť v systémoch, kde sa signálna rýchlosť a prenosová rýchlosť rovnajú, keďže je možné, aby signálna udalosť niesla jeden bit.

Parita je jednoduchý kontrolný súčet určený pre ochranu integrity jedného dátového slova. Existujú dva rôzne druhy parity: *párna parita* a *nepárna parita*.

Význam jednotlivých bitov v 11-tich registroch:

1. register: Prijímací register (3F8H - čítanie):

- register obsahuje prijaté dáta,

2. register: Vysielací register (3F8H - zápis):

- do registra sa zapisujú dáta, ktoré sa majú poslať po sériovej linke,

3. register: Register deliteľa hodinovej frekvencie LSB (3F8H – zápis a čítanie pri DLAB):

- ak je aktívny bit DLAB (viď 8. register), tak sa do tohto registra zapisuje dolný bajt deliteľa hodín,

4. register: Register deliteľa hodinovej frekvencie MSB (3F9H – zápis a čítanie pri DLAB):

- ak je aktívny bit DLAB (viď 8. register), tak sa do tohto registra zapisuje horný bajt deliteľa hodín,
- dolný bajt spoločne s horným bajtom tvorí 16-bitovú hodnotu deliteľa, podľa tabuľky Tab. 7.5 sa dá nastaviť požadovaná prenosová rýchlosť:

Tab. 7.5 Nastavenie deliteľa (LSB, MSB) pri požadovanej prenosovej rýchlosti

Baud	Deliteľ	LSB	MSB		Baud	Deliteľ	LSB	MSB
2	57600	0	225		1 200	96	96	0
10	11520	0	45		2 400	48	48	0
55	2080	32	8		4 800	24	24	0
75	1536	0	6		9 600	12	12	0
110	1040	16	4		19 200	6	6	0
150	768	0	3		38400	3	3	0
300	384	128	1		76 800	2	2	0
600	192	192	0		115 200	1	1	0

- prenosová rýchlosť (B) sa dá na základe nastaveného deliteľa (D) vypočítať pomocou vzorca:

$$B = \frac{115200}{D} \qquad (7.1),$$

- deliteľ (D) sa dá tiež vypočítať podľa požadovanej prenosovej rýchlosti (B), ale treba dávať pozor na to, aby bol deliteľ celočíselný:

$$D = \frac{115200}{B} \qquad (7.2),$$

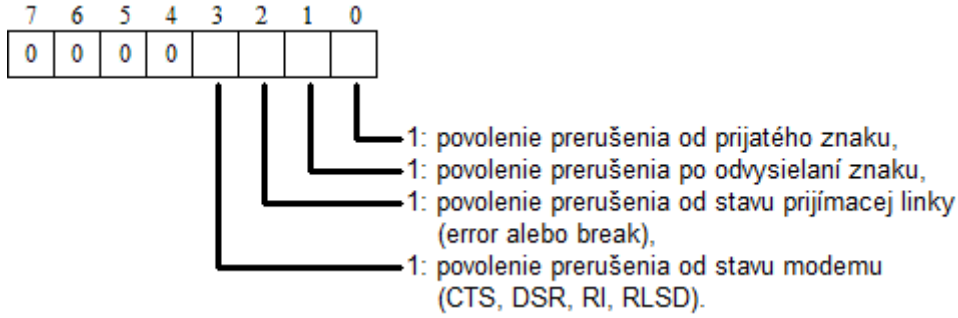
- najpoužívanejšia prenosová rýchlosť je 9 600 baudov, potom sa ešte často používajú 1200 baudov (myši) a 115 200 baudov,
- výpočet LSB a MSB:
  - MSB je výsledok celočíselného delenia deliteľa s číslom 256:

$$D \div 256 = MSB\ zv.\ LSB \qquad (7.3),$$

  - LSB je celočíselný zvyšok po tomto delení,

5. register: Register povolenia prerušenia – maska prerušenia (3F9H – zápis a čítanie):

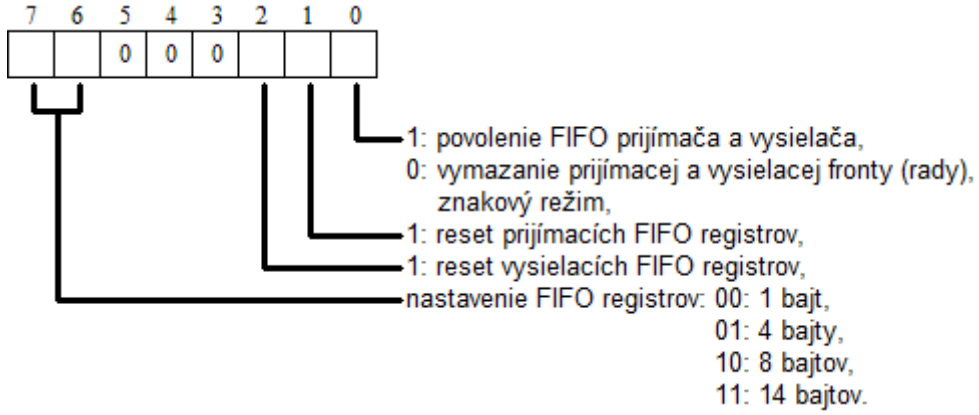
- význam jednotlivých bitov registra (Obr. 7.9):



Obr. 7.9 Význam bitov v registri povolenia prerušenia

6. register: Riadiaci register FIFO (3FAH – zápis, iba od obvodu UART 16550):

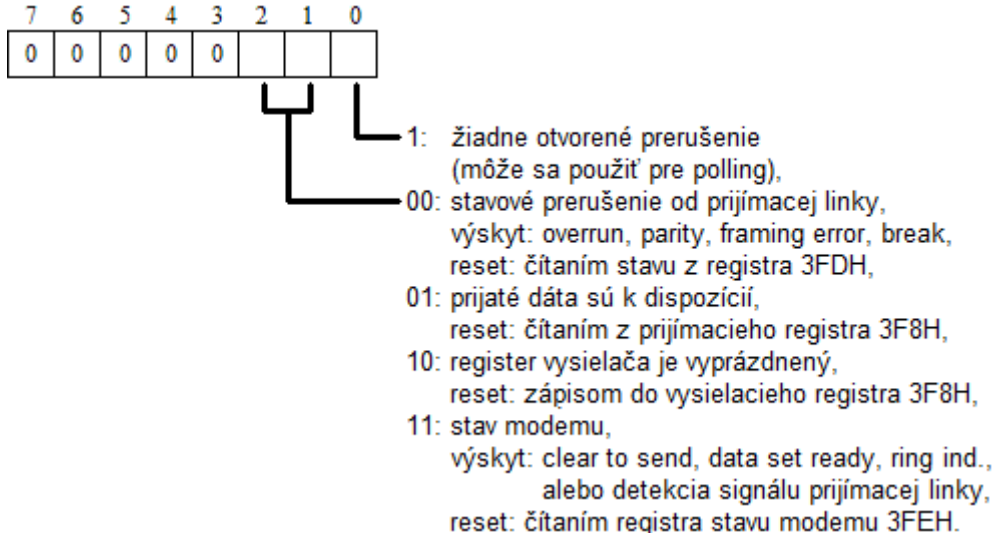
- skratka FIFO znamená: first in, first out (prvý dnu, prvý von)
- význam jednotlivých bitov registra (Obr. 7.10):



Obr. 7.10 Význam bitov v riadiacom registri FIFO

7. register: Register identifikácie prerušenia (3FAH - čítanie):

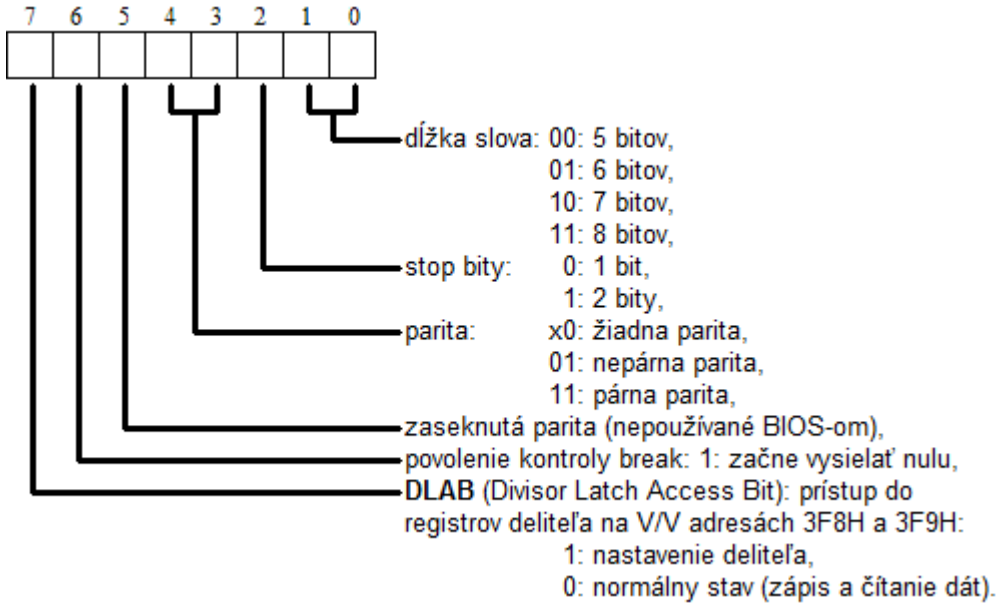
- po príchode prerušenia sa vyčíta tento register, čím sa zistí, čo prerušenie spôsobilo,
- význam jednotlivých bitov registra (Obr. 7.11):



Obr. 7.11 Význam bitov v registri identifikácie prerušenia

8. register: Register riadenia linky (3FBH – zápis a čítanie):

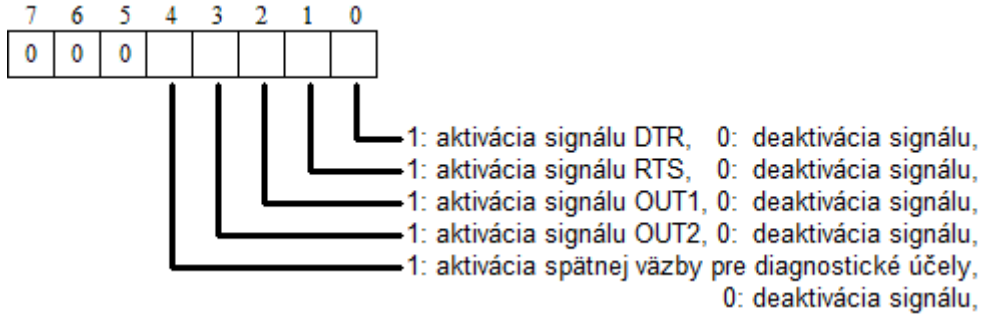
- význam jednotlivých bitov registra (Obr. 7.12):



Obr. 7.12 Význam bitov v registri riadenia linky

9. register: Register riadenia modemu (3FCH – zápis a čítanie):

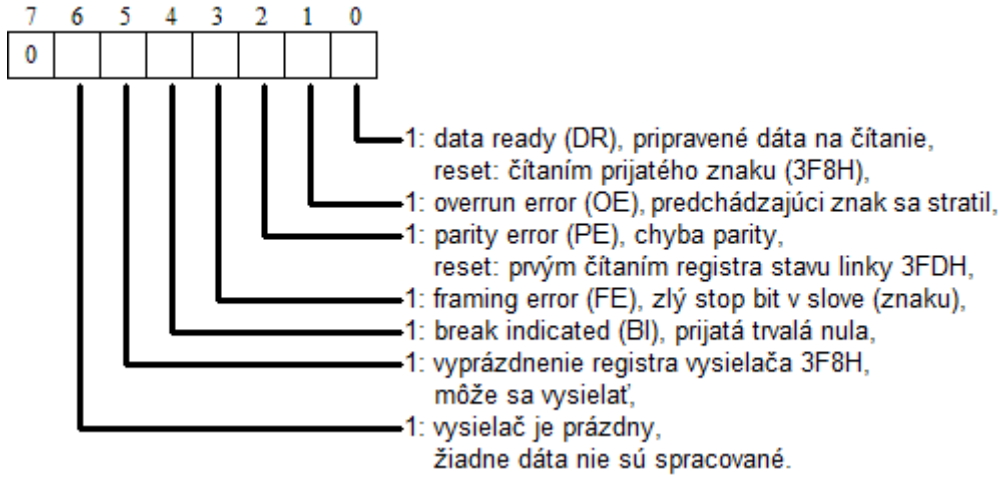
- význam jednotlivých bitov registra (Obr. 7.13):



Obr. 7.13 Význam bitov v registri riadenia linky

10. register: Register stavu linky (3FDH – čítanie):

- význam jednotlivých bitov registra (Obr. 7.14):

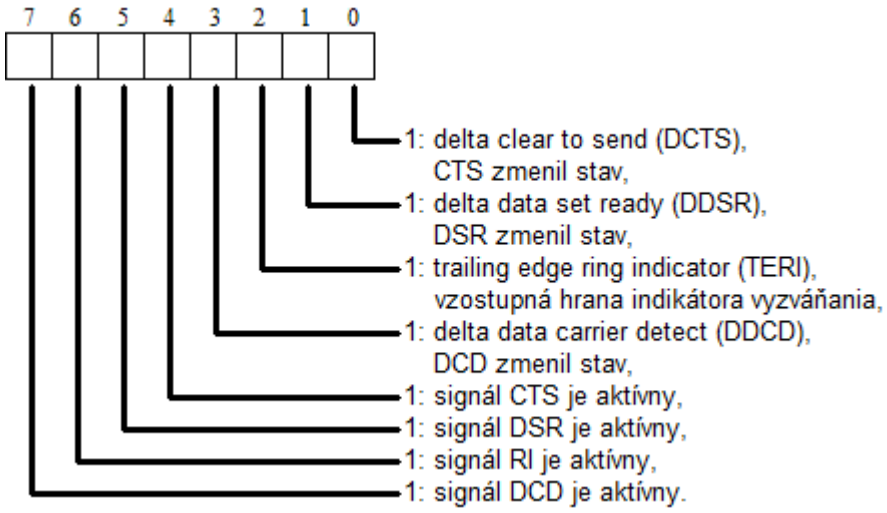


Obr. 7.14 Význam bitov v registri stavu linky

- bity 1 až 4 (OE, PE, FE a BI) spôsobujú prerušenie, ak je v registri 3F9H prerušenie povolené.

11. register: Register stavu modemu (3FEH – čítanie):

- význam jednotlivých bitov registra (Obr. 7.15):



Obr. 7.15 Význam bitov v registri stavu modemu

- bity 0 až 3 (DCTS, DDSR, TERI a DDCD) spôsobujú prerušenie, ak je v registri 3F9H prerušenie povolené.

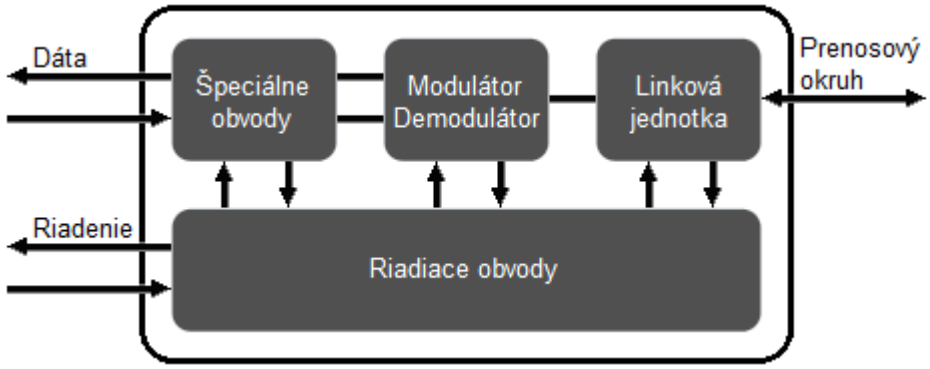
7.1.5 Modem

Pri spojení odľahlých zariadení sériovým rozhraním RS-232C sa nezaobišlo bez spomínaného UZD zariadenia nazvaného *modem*. Toto slovo vzniklo pôvodne spojením slov *mod*ulátor a *dem*odulátor. Základnou úlohou modemu (Obr. 7.16) je previesť dáta z koncového zariadenia do takej formy, ktorá vyhovuje dátovému okruhu. Modulácia znamená prevedenie dvoj-hodnotových digitálnych signálov na signál prenosný telefónnym kanálom. Tu nie je na mysli iba samotný telefónny okruh, ale aj jeho ďalšie možnosti, ako je šírka jeho frekvenčného pásma, chybovosť, atď. Frekvenčné pásmo telefónneho kanálu je obmedzené od 300 Hz do 3 400 Hz. Moduláciou sa teda signál z počítača prevedie na striedavý analógový signál, ktorého frekvencia spadá do uvedeného pásma. Demodulácia je proces presne opačný, kde z analógového signálu dostaneme digitálny signál. Modem sa neskôr začal používať aj pri prepájaní ethernetových sieti od dial-up až po dnešné ADSL.

Modemom sa, inač povedané, prispôsobil výstup počítača tak, aby sa z hľadiska telefónnej siete javil ako telefónny prístroj. Nestačí splniť iba požiadavky vyplývajúce zo šírky frekvenčného pásma, ale prispôsobenie sa týka aj úrovne výstupných signálov a impedancie na rozhraní S1 (Obr. 7.1). Modem preto obsahuje takzvanú linkovú jednotku, ktorá toto zaisťuje. Obvody riadenia slúžia k spolupráci modemu s koncovým zariadením, v našom prípade najčastejšie s počítačom. To predáva s ich pomocou modemu požiadavku na prenos dát, modem ich prostredníctvom hlási pripravenosť k vysielaniu, indikuje kvalitu prijímaného signálu alebo indikuje prichádzajúce volanie.

Súčasťou modemu môžu byť aj *špeciálne obvody*, ktoré zaisťujú kompresiu (stlačenie dát za účelom zvýšenia účinnosti prenosu pri nezmenenej prenosovej rýchlosti dátového okruhu) a odolnosť voči chybám. Najjednoduchší typ kompresie dát sa spraví tak, že dlhšia postupnosť núl alebo jednotiek sa neprenášala celá, ale nahradila sa informáciou o stave

a dĺžke jeho trvania. Chybovú odolnosť zabezpečujú cyklické kódy majúce schopnosť detekcie a korekcie chýb.



Obr. 7.16 Bloková schéma modemu

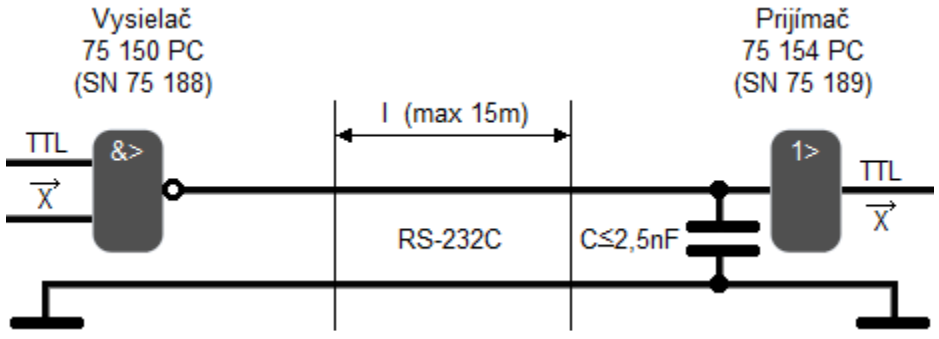
V bloku špeciálnych obvodov sa niekedy nájde aj *skrambler* (kóder), ktorý vstupné dáta kóduje do pseudonáhodnej postupnosti, a tak znemožňuje nepovolenej osobe zistiť obsah prenášanej správy. Okrem toho spôsobí rovnomernejšie rozdelenie výkonového spektra prenášaného signálu a zaistí aj jednoduchšiu synchronizáciu na prijímacej strane.

Modemy pre vyššiu rýchlosť prenosu dát obsahujú *korektor*. Ten umožňuje kompenzovať nedokonalosti telefónnych okruhov. Môže byť buď pevný, prípadne ručne nastaviteľný, alebo adaptívny. Pevný korektor sa používa pri prenajatých telefónnych okruhoch, ktorých parametre sa nemenia. Ak sa použije komutovaná telefónna sieť, nie je stále k dispozícií rovnaký okruh a tým pádom sa menia aj jeho parametre. Tu sa používa adaptívny korektor, ktorý je schopný nastaviť svoje parametre podľa konkrétnych vlastností.

Modem pre RS-232C musí samozrejme obsahovať aj obvody rozhrania S2 (Obr. 7.1), cez ne si počítač a modem vymieňajú dáta a riadiace informácie.

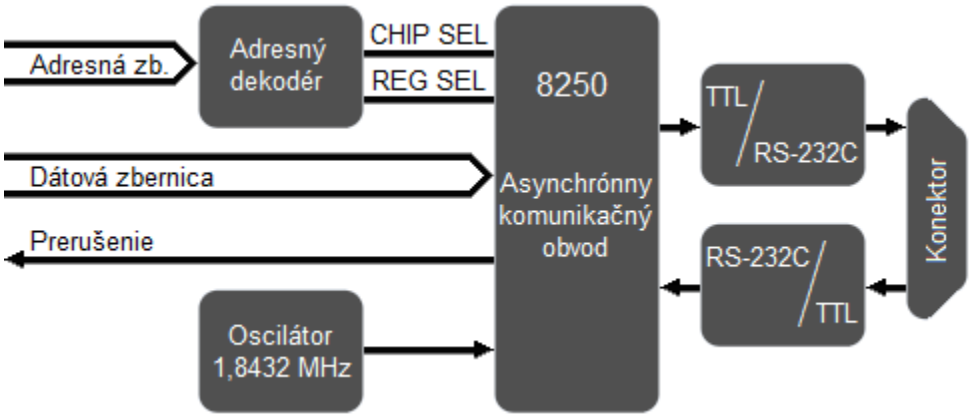
7.1.6 Zapojenie obvodu

Všetky vývody obvodu 8250 sú TTL kompatibilné. Vzhľadom k tomu, že sú normou RC-232C a V.28 stanovené napäťové úrovne pre zobrazovanie logických úrovní inak (úroveň H predstavuje napätie v rozmedzí -3V až -15V, úroveň L napätie v rozmedzí +3V až +15V), je súčasťou adaptéru konvertor medzi TTL a zobrazením na tomto rozhraní. Doporučené zapojenie je na obrázku Obr. 7.17.



Obr. 7.17 Schéma zapojenia obvodu rozhrania RS-232C

Bloková schéma adaptéru je na obrázku Obr. 7.18.



Obr. 7.18 Bloková schéma adaptéru (RS-232C)

7.1.7 Programovanie pomocou zápisu a čítania registrov

Oblasť premenných BIOS-u obsahuje zoznam až štyroch bázových adries COM portov. Behom testu pri štarte počítača BIOS testuje a inicializuje COM1 a COM2. Tieto porty sú tvorené obvodmi UART 8250 pri PC-XT, UART 16450 pri PC-AT, 16550 pri PC-386 alebo vyššou radov obvodu. Nevýhodou obvodov UART 8250 a UART 16450 je, že sú schopné si zapamätať iba jeden prijatý znak, takže môže hlavne pri vyšších prenosových rýchlostiach dôjsť k strate dát (overrun error), keď počítač nestihne včas odobrať prijatý znak pred príchodom ďalšieho. Z tohto dôvodu bol vyvinutý obvod UART 16550, opäť vývodovo aj programovo kompatibilný, ktorý má však 14-znakovú prijímaciu vyrovnávaciu pamäť (buffer). Táto pamäť však musí byť programovo zapnutá, inač sa obvod chová ako úplne štandardný obvod UART 16450.

Adresovanie vstupno-výstupných zariadení:

- adaptér COM1 obsadzuje porty 3F8H až 3FFH (bázová adresa 3F8H),
- adaptér COM2 obsadzuje porty 2F8H až 2FFH (bázová adresa 2F8H).

Programovanie pomocou zápisu a čítania registrov je riešené v nasledujúcich zadaniach pomocou funkcií `inportb()` a `outportb()`, ktoré už boli vysvetlené pri adresovaní vstupno-výstupných periférií (kapitola 2.2.7). Tieto zadania sú riešené pomocou programovacieho jazyka C pod operačným systémom MS-DOS.

Zadanie 1:

Naprogramujte aplikáciu v programovacom jazyku C pod OS MS-DOS, ktorá bude *prijímať* jednotlivé znaky zo sériovej linky RS-232C konkrétne z portu COM1. Podmienky prenosu: prenosová rýchlosť 9 600 baudov, dĺžka slova 7 bitov, 1 stop bit a párna parita. Pri programovaní využite zápis a čítanie registrov sériového rozhrania.

Analýza zadania:

V prvej časti programu je nutné inicializovať sériový port pomocou registra riadenia linky a registrov deliteľa hodinovej frekvencie. Tento krok sa nemôže spraviť, kým sa nenalinkuje knižnica *dos.h*, ktorá obsahuje funkcie `inportb()` a `outportb()`.

Pomocou bitu DLAB v registri riadenia linky sa sprístupnia registre deliča, ktoré sa nastavia na 9600 baudov, čiže LSB sa nastaví na hodnotu 12 a MSB na hodnotu 0 (viď Tab. 7.5). Potom sa v registri riadenia linky vypne bit DLAB a nastavia sa zvyšné podmienky prenosu, ktorými sú dĺžka slova 7 bitov, 1 stop bit a párna parita. Tieto podmienky sa nastavia pomocou hodnoty 1AH (viď Obr. 7.12). Týmito krokmi sa ukončí inicializácia sériového portu a pre lepšiu prehľadnosť kódu sa pre túto inicializáciu vytvorí vlastná funkcia.

Ďalším dôležitým krokom v programe je sledovanie, či sú dáta pripravené na čítanie (data ready), toto sa sleduje v registri stavu linky na nultom bite (viď Obr. 7.14). Pre zjednodušenie práce sa pre túto kontrolu zadefinuje makro, ktoré bude sledovať stav tohto bitu.

Teraz keď sú už všetky čiastočné úlohy splnené, tak sa môže prejsť k tomu, čo by mal obsahovať hlavný program. Najskôr sa vyčistí obrazovka, potom sa spustí navrhnutá funkcia inicializácie a tak sa môže čakať na prijímané znaky. Ak počítač príjme po sériovej linke nejaký znak, tak ho vypíše na obrazovku. Program by mal nejakú dobu bežať, tak sa spraví dohoda medzi prijímačom a vysielačom. Táto dohoda bude nasledovná, keď vysielač vyšle ASCII kód klávesa *ESC* (1BH), tak sa prijímač vypne. Táto úloha sa splní cyklom *while* s podmienkou na konci.

Riešenie:

```
1    #include<conio.h>
2    #include<stdio.h>
3    #include<dos.h>
4    #include<ctype.h>
5    #define PRISLO ( inportb(baza+5)& 1 )
6
7    int baza = 0x3f8;
8
9    void inic_sport(void)
10   {
11        outportb(baza+3,0x80);
12        outportb(baza,12);
13        outportb(baza+1,0);
14        outportb(baza+3,0x1a);
15   }
16
17   void main()
18   {
19        unsigned char prijmi;
20        clrscr();
21        inic_sport();
22        do
23        {
24             if ( PRISLO )
25             {
26                  prijmi = inportb(baza);
27                  printf("%c",prijmi);
28             }
29        }
30        while(prijmi != 0x1b);
31   }
```

Vysvetlenie:

1-4: Prilinkovanie dôležitých knižníc k programu.
5: Zadefinovanie makra sledujúceho, či sú dáta pripravené na čítanie (bit: data ready).
7: Deklarácia bázovej adresy sériového portu COM1.
9: Deklarácia a definovanie funkcie inicializácie sériového portu: `inic_sport()`.
11: Zopnutie bitu DLAB (viď Obr. 7.12).
12, 13: Nastavenie registra deliteľa hodinovej frekvencie (LSB, MSB – viď Tab. 7.5) – nastavenie prenosovej rýchlosti.
14: Nastavenie zvyšných podmienok prenosu: dĺžka slova 7 bitov, 1 stop bit a párna parita pomocou hodnoty 1AH (viď Obr. 7.12).
17: Deklarácia a definovanie hlavného programu.
19: Deklarácia premennej `prijmi` slúžiacej ku ukladaniu jednotlivých prijatých znakov.
20: Vyčistenie obrazovky.
21: Spustenie funkcie `inic_sport()` pre inicializáciu sériového portu.
22-30: Cyklus s podmienkou na konci, ktorý sa opakuje, kým v premennej `prijmi` nebude ASCII kód klávesa *ESC* (1BH).
24-28: Podmienka ktorá sa vykoná, ak sériová linka prijme znak (kontrola zadefinovaného makra).
26: Do premennej `prijmi` sa vloží prijatý znak sériovou linkou z registra 3F8H.
27: Znak v premennej `prijmi` sa vypíše na obrazovku.

Zadanie 2:

Naprogramujte aplikáciu v programovacom jazyku C pod OS MS-DOS, ktorá bude *vysielať* jednotlivé znaky po sériovej linke RS-232C konkrétne po porte COM1. Podmienky prenosu: prenosová rýchlosť 9 600 baudov, dĺžka slova 7 bitov, 1 stop bit a párna parita. Pri programovaní využite zápis a čítanie registrov sériového rozhrania.

Analýza zadania:

Prvá časť programu, čiže inicializácia sériovej linky a prilinkovanie knižnice *dos.h* je rovnaká ako v prvom zadaní, takže analýza je rovnaká.

V prvom zadaní sa sledovalo, či sú dáta pripravené na čítanie (data ready), v tomto zadaní sa sleduje vyprázdnenosť registra vysielača, toto sa sleduje pomocou piateho bitu v registri stavu linky (viď Obr. 7.14). V podstate sa sleduje, či vysielač môže vysielať dáta. Aj v tomto prípade sa pre zjednodušenie práce zadefinuje makro, ktoré bude sledovať stav tohto bitu.

Kvôli prehľadnosti programu je ešte vhodné vytvoriť ďalšiu funkciu, ktorej úlohou bude vysielanie znaku. V tejto funkcií sa v cykle kontroluje spomínané makro kým nie je možné vysielanie a keď je linka uvoľnená vyšle sa znak.

Hlavný program najskôr vyčistí obrazovku a následne zinicializuje sériový port. Potom v spomínanom dohodnutom cykle (viď analýzu prvého zadania), ktorý čaká na stlačenie klávesa *ESC* (1BH), sa ostatné stlačené klávesy posielajú pomocou sériovej linky. V tomto zadaní je ešte dôležité myslieť na to, že ak sa stlačí *Enter* (13) treba ju nahradiť znakom *nový riadok*.

Riešenie:

```
1    #include<conio.h>
2    #include<dos.h>
3    #include<stdio.h>
4    #include<ctype.h>
```

```
5    #define PRAZDNE ( inportb(baza+5) & 32 )
6
7    int baza = 0x3f8;
8
9    void vysli(char w)
10   {
11        while ( !PRAZDNE );
12        outportb((int)baza, (unsigned char) w);
13   }
14
15   void inic_sport(void)
16   {
17        outportb(baza+3,0x80);
18        outportb(baza,12);
19        outportb(baza+1,0);
20        outportb(baza+3,0x1a);
21   }
22
23   void main()
24   {
25        char ch;
26        inic_sport();
27        clrscr();
28        do
29        {
30             ch=getche();
31             vysli(ch);
32             if(ch==13)
33             {
34                  vysli(10);
35                  putch(10);
36             }
37        }
38        while(ch != 0x1b);
39   }
```

Vysvetlenie:

1-4: Prilinkovanie dôležitých knižníc k programu.

5: Zadefinovanie makra sledujúceho, či je register vysielača vyprázdnený (Obr. 7.14).

7: Deklarácia bázovej adresy sériového portu COM1.

9: Deklarácia a definovanie funkcie pre vyslanie znaku po sériovej linke: `vysli()`.

11: Prázdny cyklus čakajúci na vyprázdnenie registra vysielača.

12: Vyslanie znaku, ktorý je vstupným argumentom funkcie `vysli()` po sériovej linke pomocou registra 3F8H.

15: Deklarácia a definovanie funkcie inicializácie sériového portu: `inic_sport()`.

17: Zopnutie bitu DLAB (viď Obr. 7.12).

18, 19: Nastavenie registra deliteľa hodinovej frekvencie (LSB, MSB – viď Tab. 7.5) – nastavenie prenosovej rýchlosti.

20: Nastavenie zvyšných podmienok prenosu: dĺžka slova 7 bitov, 1 stop bit a párna parita pomocou hodnoty 1AH (viď Obr. 7.12).

23: Deklarácia a definovanie hlavného programu.

25: Deklarácia premennej `ch` slúžiacej k vyčítavaniu stlačených klávesov na klávesnici.
26: Spustenie funkcie `inic_sport()` pre inicializáciu sériového portu.
27: Vyčistenie obrazovky.
28-38: Cyklus s podmienkou na konci, ktorý sa opakuje, kým v premennej `ch` nebude ASCII kód klávesa *ESC* (1BH).
- 30: Čakanie na stlačenie klávesa a vloženie jej hodnoty do premennej `ch`.
- 31: Spustenie funkcie `vysli()` s argumentom `ch`.
- 32-36: Podmienka ktorá sa vykoná ak sa stlačí *Enter* na klávesnici.
  - 34: Vyšle sa znak *nového riadku* po sériovej linke.
  - 35: Vloží sa na obrazovku vysielača *nový riadok*, ktorý posunie kurzor na začiatok nového riadku.

7.1.8 Programovanie pomocou prerušení

Pre programovanie sériového rozhrania sa dá použiť obslužný program (funkcia) BIOS-u, ktorý je uložený na adrese INT 14H. V kapitole 2.3.7 bolo vysvetlené ako sa dá využiť obslužný program prerušenia aj s parametrami. Programátor okrem prepisovania programov obslúh prerušení, môže tieto programy aj využiť a to volaním obsluhy prerušenia s parametrom pomocou funkcie (príkazu): `int86()`, ktorá je súčasťou knižnice *dos.h* a presná syntax vyzerá následne:

```
union REGS r;
r.h.al = int hodnota;
r.h.ah = int parameter;
r.x.dx = int id;
int86(int adresa,&r,&r);
return(r.h.ah);
```

Vysvetlenie jednotlivých riadkov syntaxe:

- deklarácia parametrov pre funkciu `int86()`,
- parameter `r.h.al` vyjadruje hodnotu, ktorú chceme zapísať ako parameter obsluhy prerušenia,
- parameter `r.h.ah` vyjadruje poradie parametra obsluhy prerušenia,
- parameter `r.x.dx` vyjadruje identifikátor zariadenia (napr.: LPT1, LPT2, alebo COM1, COM2),
- volanie funkcie knižnice *dos.h* `int86()` jej prvým parametrom je adresa obslužného programu v tabuľke vektorov, druhým sú v podstate prednastavené parametre `r.h.al`, `r.h.ah` a `r.x.dx`, a tretím je výstup funkcie,
- ak má obsluha výstup tak je najčastejšie zapísaná v `r.h.ah`.

Obslužný program INT 14H bude fungovať so všetkými štyrmi portami, ak sú uložené adresy týchto portov v tabuľke COM portu začínajúcej na adrese 0:0400. Je potrebné, aby zadané dva adaptéry nezdieľali rovnakú adresu, pretože ani jeden z nich nebude fungovať. BIOS ale podporuje len pomerne primitívnu komunikáciu využívajúcu stavové dotazy (polling), čo je pre zložitejšie aplikácie takmer nepoužiteľné. Adaptér je však schopný vyvolať hardwarové prerušenie na základe mnohých rôznych podmienok v závislosti na hodnotách v registri povolení prerušení (3F9H, alebo 2F9H).

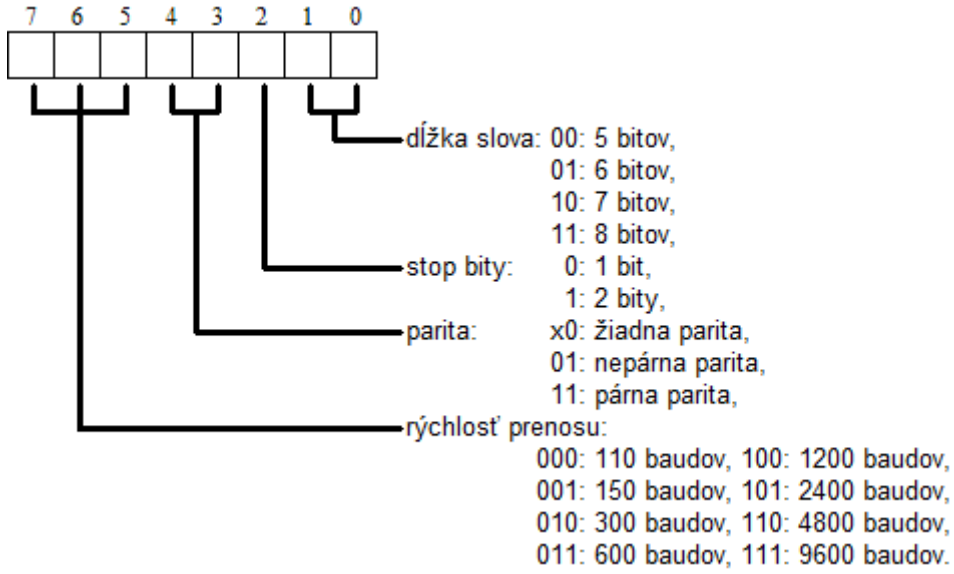
Prerušenie od sériovej linky:

- adaptér COM1 vyvoláva prerušenie úrovne 4 (IRQ 4 je spracovávaný obslužným programom na ktorý ukazuje vektor INT 0CH),
- adaptér COM2 vyvoláva prerušenie úrovne 3 (IRQ 3 je spracovávaný obslužným programom na ktorý ukazuje vektor INT 0BH).

Funkcie BIOS-u nevyužívajú prerušenia a preto je lepším variantom vlastný program, ako využívať ten, ktorý je na adrese INT 14H. Napriek spomínaným nedostatkom budú nižšie vysvetlené parametre obslužného programu sériovej linky (INT 14H):

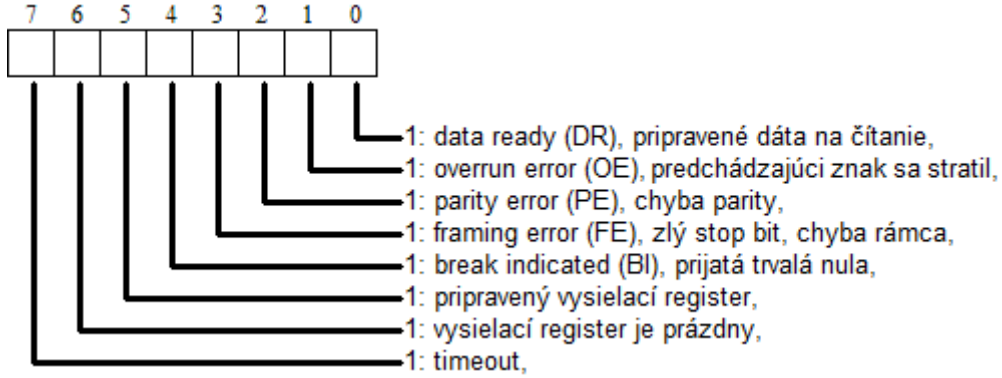
Služba (ah) 00H: Inicializácia komunikačného portu:

- vstup: dx – číslo portu (0, 1),
- vstup: al – bitová vlajka parametra inicializácie Obr. 7.19:



Obr. 7.19 Význam bitov vstupného parametra al služby 00H v programe INT 14H

- výstup: ah – stav riadiacej linky Obr. 7.20:



Obr. 7.20 Význam bitov výstupného parametra ah služby 00H v programe INT 14H

Služba (ah) 01H: Posielanie znaku na vybraný port RS-232C:

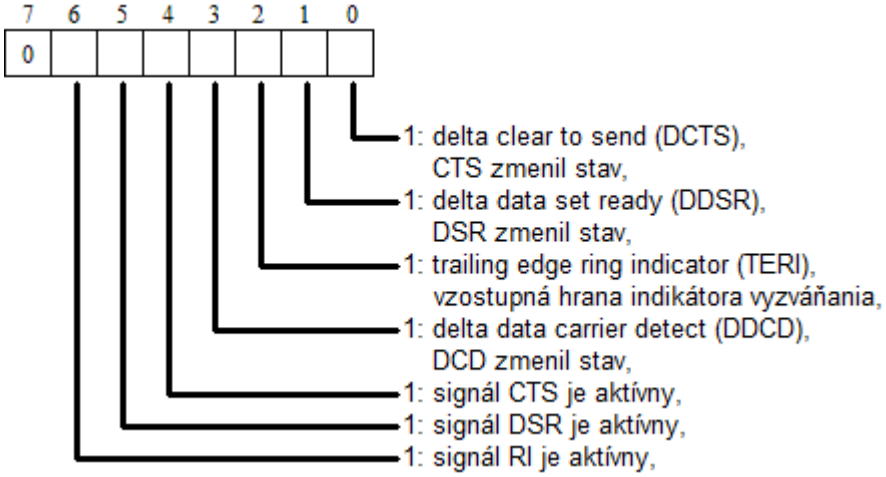
- vstup: dx – číslo portu (0, 1),
- vstup: al – vysielaný znak,
- výstup: al – je rezervovaný,
- výstup: ah – ak nastala chyba (timeout), tak v AH je stav riadiacej linky Obr. 7.20,

Služba (ah) 02H: Prijímanie znaku z vybraného portu RS-232C:

- vstup: dx – číslo portu (0, 1),
- výstup: al – obsahuje prijatý znak,
- výstup: ah – ak nastala chyba, tak v AH je stav riadiacej linky Obr. 7.20,

Služba (ah) 03H: Stav komunikačného portu:

- vstup: dx – číslo portu (0, 1),
- výstup: ax – stav komunikačného portu,
- výstup: ah – stav riadiacej linky Obr. 7.20,
- výstup: al – stav modemu Obr. 7.21:



Obr. 7.21 Význam bitov výstupného parametra al služby 00H v programe INT 14H

V tejto podkapitole (7.1.8) sú tri zadania, prvé dve zadania sú programovanie sériového rozhrania pomocou obsluhy prerušenia uloženej na adrese INT 14H (softvérové prerušenie) a tretie zadanie je pomocou novej obsluhy prerušenia IRQ 4 (hardvérové prerušenie).

Zadanie 1:

Naprogramujte aplikáciu v programovacom jazyku C pod OS MS-DOS, ktorá bude *prijímať* jednotlivé znaky zo sériovej linky RS-232C konkrétne z portu COM1. Podmienky prenosu: prenosová rýchlosť 9 600 baudov, dĺžka slova 8 bitov, 1 stop bit a bez kontroly parity. Pri programovaní využite obsluhu prerušenia vo vektore prerušení INT 14H (softvérové prerušenie).

Analýza zadania:

Pri programovaní tohto zadania sa využíva funkcia `int86()`, ktorá je v knižnici *dos.h*. Najprv je potrebné zinicializovať sériový port. Inicializáciu sériového portu má na starosti služba 00H v obsluhe prerušenia sériového portu (INT 14H). Tejto službe je potrebné nastaviť dva vstupné parametre:

- číslo portu (pre COM1 sa dx nastaví na 0) a
- podmienky prenosu (pre prenosovú rýchlosť 9 600 baudov, dĺžku slova 8 bitov, 1 stop bit a bez kontroly parity sa al nastaví na E3H, Obr. 7.19).

Inicializácia sériového rozhrania sa vloží do vlastnej funkcie pre lepšiu prehľadnosť kódu.

Hlavný program vyčistí obrazovku a spusti funkciu inicializácie sériového rozhrania. Potom sa môže čakať na prijímané znaky, informáciu o prijatom znaku zaobstaráva služba 03H v obsluhe prerušenia sériového portu (INT 14H). Táto služba sleduje stav komunikačného portu (Obr. 7.20), ak prvý bit výstupného parametra *ah* je aktívny (data ready), tak je znak pripravený na vyčítanie. Znak sa následne vyčíta pomocou služby 02H, ktorej úlohou je prijímanie znaku z vybraného portu RC-232C. Potom sa prijatý znak vypíše na obrazovku. Program by mal nejakú dobu bežať, tak sa spraví dohoda medzi prijímačom a vysielačom. Táto dohoda bude nasledovná, keď vysielač vyšle ASCII kód klávesa *ESC* (1BH), tak sa prijímač vypne. Táto úloha sa splní cyklom *while* s podmienkou na konci a v tomto cykle bude celý program od čakania na prijímaný znak až po jeho vypísanie na obrazovku počítača.

Riešenie:

```
1    #include<conio.h>
2    #include<stdio.h>
3    #include<dos.h>
4    #include<ctype.h>
5
6    void inic_sport(void)
7    {
8         union REGS in,out;
9         in.h.ah=0x00;
10        in.x.dx=0x00;
11        in.h.al=0xE3;
12        int86(0x14,&in,&out);
13   }
14
15   void main()
16   {
17        union REGS in,out;
18        unsigned char prijmi;
19        clrscr();
20        inic_sport();
21        do
22        {
23             in.h.ah=0x03;
24             in.x.dx=0x00;
25             int86(0x14,&in,&out);
26             if (out.h.ah & 0x01)
```

```
27              {
28                   in.h.ah=0x02;
29                   in.x.dx=0x00;
30                   int86(0x14,&in,&out);
31                   prijmi = out.h.al;
32                   printf("%c",prijmi);
33              }
34         }
35         while(prijmi!= 0x1b);
36    }
```

Vysvetlenie:

1-4: Prilinkovanie dôležitých knižníc k programu.

6: Deklarácia a definovanie funkcie inicializácie sériového portu: `inic_sport()`.

8: Deklarácia premenných `in` a `out` potrebných pre napĺňanie a vyčítavanie z funkcie `int86()`.

9: Nastavenie služby 00H (*Inicializácia komunikačného portu*) pre funkciu `int86()`.

10: Nastavenie portu COM1 pre funkciu `int86()`.

11: Nastavenie podmienok prenosu, konkrétne prenosová rýchlosť 9 600 baudov, dĺžka slova 8 bitov, 1 stop bit a bez kontroly parity, pomocou hodnoty E3H (Obr. 7.19) pre funkciu `int86()`.

12: Naplnenie funkcie `int86()` nastavenými parametrami v riadkoch 9, 10, 11.

15: Deklarácia a definovanie hlavného programu.

17: Deklarácia premenných `in` a `out` potrebných pre napĺňanie a vyčítavanie z funkcie `int86()`.

18: Deklarácia premennej `prijmi` slúžiacej ku ukladaniu jednotlivých prijatých znakov.

19: Vyčistenie obrazovky.

20: Spustenie funkcie `inic_sport()` pre inicializáciu sériového portu.

21-35: Cyklus s podmienkou na konci, ktorý sa opakuje, kým v premennej `prijmi` nebude ASCII kód klávesa *ESC* (1BH).

- 23: Nastavenie služby 03H (*Stav komunikačného portu*) pre funkciu `int86()`.
- 24: Nastavenie portu COM1 pre funkciu `int86()`.
- 25: Naplnenie funkcie `int86()` nastavenými parametrami v riadkoch 23 a 24.
- 26: Podmienka v ktorej sa kontroluje výstupný parameter funkcie `int86()`, podľa ktorého sa určuje, či sériová linka prijala znak (kontrola bitu data ready, viď Obr. 7.20).
  - 28: Nastavenie služby 02H (*Prijímanie znaku z vybraného portu RS-232C*) pre funkciu `int86()`.
  - 29: Nastavenie portu COM1 pre funkciu `int86()`.
  - 30: Naplnenie funkcie `int86()` nastavenými parametrami v riadkoch 28 a 29.
  - 31: Do premennej `prijmi` sa vloží prijatý znak sériovou linkou, tento znak je výstupným parametrom funkcie `int86()`.
  - 32: Znak v premennej `prijmi` sa vypíše na obrazovku.

Zadanie 2:

Naprogramujte aplikáciu v programovacom jazyku C pod OS MS-DOS, ktorá bude *vysielať* jednotlivé znaky po sériovej linke RS-232C konkrétne po porte COM1. Podmienky prenosu: prenosová rýchlosť 9 600 baudov, dĺžka slova 8 bitov, 1 stop bit a bez kontroly parity. Pri programovaní využite obsluhu prerušenia vo vektore prerušení INT 14H (softvérové prerušenie).

Analýza zadania:

Postup pri inicializácii sériového portu a dôvod na prilinkovanie knižnice *dos.h*, je rovnaký ako v prvom zadaní, takže analýza je rovnaká.

Toto zadanie potrebuje navyše funkciu pre vysielanie znaku. Funkcia využije dve služby obsluhy prerušenia sériového portu (INT 14H). Službu 03H (*Stav komunikačného portu*) kontrolujúcu stav vysielajúceho registra na šiestom bite (vysielací register je prázdny, viď Obr. 7.20) a službu 01H (*Posielanie znaku na vybraný port RS-232C*), ktorá posiela znak po sériovej linke.

Hlavný program vyčistí obrazovku, spustí funkciu inicializácie sériového portu a nakoniec posiela znaky stlačené na klávesnici, kým nie stlačený *ESC* na klávesnici. Vysielanie znaku ma na starosti funkcia navrhnutá a analyzovaná v predchádzajúcom odstavci tejto analýzy.

Riešenie:

```
1    #include<conio.h>
2    #include<dos.h>
3    #include<stdio.h>
4    #include<ctype.h>
5
6    void vysli(char znak)
7    {
8         union REGS in,out;
9         in.h.ah=0x03;
10        in.x.dx=0x00;
11        do
12        {
13             int86(0x14,&in,&out);
14        }
15        while(!(out.h.ah & 0x40));
16
17        in.h.ah=0x01;
18        in.x.dx=0x00;
19        in.h.al=znak;
20        int86(0x14,&in,&out);
21   }
22
23   void inic_sport(void)
24   {
25        union REGS in,out;
26        in.h.ah=0x00;
27        in.x.dx=0x00;
28        in.h.al=0xE3;
29        int86(0x14,&in,&out);
30   }
```

```
31
32  void main()
33  {
34       char ch;
35       inic_sport();
36       clrscr();
37       do
38       {
39            ch=getche();
40            vysli(ch);
41            if(ch==13)
42            {
43                 vysli(10);
44                 putch(10);
45            }
46       }
47       while(ch != 0x1b);
48  }
```

Vysvetlenie:

1-4: Prilinkovanie dôležitých knižníc k programu.

6: Deklarácia a definovanie funkcie pre vyslanie znaku po sériovej linke: `vysli()`.

8: Deklarácia premenných `in` a `out` potrebných pre napĺňanie a vyčítavanie z funkcie `int86()`.

9: Nastavenie služby 03H (*Stav komunikačného portu*) pre funkciu `int86()`.

10: Nastavenie portu COM1 pre funkciu `int86()`.

11-15: Cyklus, ktorý kontroluje výstup funkcie `int86()`, kým nie je sériová linka voľná (prázdny vysielací register, viď Obr. 7.20).

13: Naplnenie funkcie `int86()` nastavenými parametrami v riadkoch 9 a 10.

17: Nastavenie služby 01H (*Posielanie znaku na vybraný port RS-232C*) pre funkciu `int86()`.

18: Nastavenie portu COM1 pre funkciu `int86()`.

19: Nastavenie znaku, ktorý je potrebné vyslať, pre funkciu `int86()`.

20: Naplnenie funkcie `int86()` nastavenými parametrami v riadkoch 17, 18 a 19, ktorá vyšle požadovaný znak po sériovej linke.

23: Deklarácia a definovanie funkcie inicializácie sériového portu: `inic_sport()`.

25: Deklarácia premenných `in` a `out` potrebných pre napĺňanie a vyčítavanie z funkcie `int86()`.

26: Nastavenie služby 00H (*Inicializácia komunikačného portu*) pre funkciu `int86()`.

27: Nastavenie portu COM1 pre funkciu `int86()`.

28: Nastavenie podmienok prenosu, konkrétne prenosová rýchlosť 9 600 baudov, dĺžka slova 8 bitov, 1 stop bit a bez kontroly parity, pomocou hodnoty E3H (Obr. 7.19) pre funkciu `int86()`.

29: Naplnenie funkcie `int86()` nastavenými parametrami v riadkoch 26, 27 a 28.

32: Deklarácia a definovanie hlavného programu.

34: Deklarácia premennej `ch` slúžiacej k vyčítavaniu stlačených klávesov na klávesnici.

35: Spustenie funkcie `inic_sport()` pre inicializáciu sériového portu.

36: Vyčistenie obrazovky.

37-47: Cyklus s podmienkou na konci, ktorý sa opakuje, kým v premennej `ch` nebude ASCII kód klávesa *ESC* (1BH).
- 39: Čakanie na stlačenie klávesa a vloženie jej hodnoty do premennej `ch`.
- 40: Spustenie funkcie `vysli()` s argumentom `ch`.
- 41-45: Podmienka ktorá sa vykoná ak bola stlačený *Enter* na klávesnici.
  - 43: Vyšle sa znak *nového riadku* po sériovej linke.
  - 44: Vloží sa na obrazovku vysielača *nový riadok*, ktorý posunie kurzor na začiatok nového riadku.

Zadanie 3:

Naprogramujte aplikáciu v programovacom jazyku C pod OS MS-DOS, ktorá bude *prijímať* aj *vysielať* jednotlivé znaky po sériovej linke RS-232C konkrétne po porte COM1. Podmienky prenosu: prenosová rýchlosť 9 600 baudov, dĺžka slova 7 bitov, 1 stop bit a párna parita. Pri programovaní využite hardvérové prerušenie sériovej linky (IRQ4) pre prijímanie znakov zo sériovej linky a hardvérové prerušenie klávesnice (IRQ1) pre vysielanie znakov po sériovej linke.

Analýza zadania:

Najskôr je potrebné zadeklarovať premenné do ktorých sa vložia obsluhy jednotlivých prerušení. Tieto premenné budú typu *interrupt*, ktorý je zadefinovaný v knižnici *dos.h*. Je potrebné zadeklarovať 4 takéto premenné do ktorých sa neskôr vložia:

- stará obsluha prerušenia od sériovej linky (IRQ4), je potrebné vytvoriť smerník, pretože nepoznáme veľkosť programu,
- nová obsluha prerušenia od sériovej linky (IRQ4),
- stará obsluha prerušenia od klávesnice (IRQ1), je potrebné vytvoriť smerník, pretože nepoznáme veľkosť programu,
- nová obsluha prerušenia od klávesnice (IRQ1).

Ďalším dôležitým krokom v programe je sledovanie vyprázdnenosti registra vysielača, toto sa sleduje pomocou piateho bitu v registri stavu linky (viď Obr. 7.14). V podstate sa sleduje, či vysielač môže vysielať dáta. V tomto prípade sa pre zjednodušenie práce zadefinuje makro, ktoré bude sledovať stav tohto bitu.

Po zadeklarovaní premenných prerušenia a vytvorení makra je potrebné zadeklarovať ostatné premenné, ktoré sa budú globálne používať, ktorými sú bázová adresa sériového rozhrania, prijatý znak, kód stlačeného klávesa, pomocné premenné pre pohyb kurzora po obrazovke, mapa (pole) znakov reprezentujúca kódy klávesov. Bázová adresa sériového rozhrania je 3F8H, takže môže byť typu *int*. Prijatý znak by mal byť typu *unsigned char*. Kód stlačeného klávesa nadobúda hodnoty od 0 do 255, tento kód nekorešponduje s ASCII kódom stlačeného znaku, preto je zbytočné a zavádzajúce ho zadeklarovať ako typ *char*, a vhodnejšie je použiť typ *int*. Pre rozlišovanie prijímaného a vysielaného textu sa kurzor na obrazovke bude pohybovať v dvoch blokoch a preto je potrebné odpamätávanie polohy kurzora v týchto blokoch na to nám poslúžia 4 pomocné premenné (poloha x, y pre vysielaný text a poloha x, y pre prijímaný text) typu *int*. Mapa znakov je pole typu *char*, ktoré má prvky (jednotlivé znaky) usporiadané podľa umiestnenia klávesov na klávesnici (podľa kódu jednotlivých klávesov).

Kvôli prehľadnosti programu je ešte vhodné vytvoriť ďalšiu funkciu, ktorej úlohou bude vysielanie znaku. V tejto funkcií sa v cykle kontroluje spomínané makro kým nie je možné vysielanie a keď je linka uvoľnená vyšle sa znak.

Ďalej je nutné inicializovať sériový port pomocou registra riadenia linky, registrov deliteľa hodinovej frekvencie a registra povolenia prerušenia. Pomocou bitu DLAB v registri

riadenia linky sa sprístupnia registre deliča, ktoré sa nastavia na 9600 baudov, čiže LSB sa nastaví na hodnotu 12 a MSB na hodnotu 0 (viď Tab. 7.5). Potom sa v registri riadenia linky vypne bit DLAB a nastavia sa zvyšné podmienky prenosu, ktorými sú dĺžka slova 7 bitov, 1 stop bit a párna parita. Tieto podmienky sa nastavia pomocou hodnoty 1AH (viď Obr. 7.12). Posledným krokom pri inicializácií je povolenie prerušenia sériovej linky od prijatého znaku a to pomocou hodnoty 1 v registri povolenia prerušenia (Obr. 7.9). Týmito krokmi sa ukončí inicializácia sériového portu a pre lepšiu prehľadnosť kódu sa pre túto inicializáciu vytvorí vlastná funkcia.

V ďalšom kroku vytvoríme program novej obsluhy prerušenia od sériovej linky (IRQ4) kde sa pomocou prijímacieho registra (3F8H) príjme znak, kurzor obrazovky sa presunie na príslušné miesto a tento znak vypíše na obrazovku. Potom sa ešte v rámci prerušenia prepočíta nová pozícia kurzora pre prijímanie znaku.

Ďalším novým programom obsluhy prerušenia je určený pre prerušenie od klávesnice (IRQ1). V tomto programe sa vyčíta kód stlačeného klávesa a za pomoci vytvoreného poľa (mapy) sa vyšle tento kód ako samotný znak pomocou vyššie spomenutej funkcie pre vysielanie znakov po sériovej linke. Následne sa kurzor na obrazovke monitora presunie do bloku vysielača a vypíše sa tento znak. A tak ako to bolo aj pri predchádzajúcom prerušení aj tu sa prepočíta nová pozícia kurzora.

V hlavnom programe sa najskôr vyčistí obrazovka, aby sa textový kurzor mohol pohybovať po čistej obrazovke a potom sa inicializuje sériová linka. Do vytvorených premenných pre staré prerušenia sa vložia staré programy obslúh prerušení (0CH – adresa vektora prerušenia sériovej linky, 09H – adresa vektora prerušenia klávesnice) a následne sa pomocou adries vektora prerušení nahrajú novo naprogramované obsluhy prerušenia. Obe spomínané prerušenia sa programovo povolia. Potom sa spustí cyklus ktorý sa bude opakovať, až kým nebude stlačený prvý kláves na klávesnici, ktorou je *ESC*. Pre správne ukončenie programu je ešte nutné staré obsluhy prerušení nahrať späť, a tak nahradiť tie ktoré boli novo vytvorené.

Riešenie:

```
1    #include<conio.h>
2    #include<stdio.h>
3    #include<dos.h>
4    #include<ctype.h>
5    #define PRAZDNE (inportb(baza+5) & 32)
6
7    void interrupt(* stare_pr)(void);
8    void interrupt(nove_pr)(void);
9    void interrupt(* stare_prk)(void);
10   void interrupt(nove_prk)(void);
11
12   int baza = 0x3f8;
13   unsigned char prijmi;
14   int klavesa, rp = 12, sp = 2, rv = 2, sv = 2;
15   char znak[59]={' ',0x1b,'1','2','3','4','5','6','7',...};
16
17   void vysli(char w)
18   {
19        while(!PRAZDNE);
20        outportb((int)baza, (unsigned char) w);
21   }
22
```

```
23  void inic_sport(void)
24  {
25       outportb(baza+3,0x80);
26       outportb(baza,12);
27       outportb(baza+1,0);
28       outportb(baza+3,0x1a);
29       outportb(baza+1,0x01);
30  }
31
32  void interrupt nove_pr(void)
33  {
34       outportb(0x21, (inportb(0x21)|0x10));
35       prijmi = inportb(baza);
36       gotoxy(sp,rp);
37       printf("%c",prijmi);
38       sp++;
39       if ((sp==79) || (prijmi==10))
40       {
41            sp = 2;
42            rp++;
43       }
44       outportb(0x21,inportb(0x21)&~0x10);
45       outportb(0x20,0x20);
46  }
47
48  void interrupt nove_prk(void)
49  {
50       outportb(0x21, (inportb(0x21)|0x02));
51       klavesa = inportb(0x60);
52       if (klavesa<59)
53       {
54            vysli(znak[klavesa]);
55            gotoxy(sv,rv);
56            printf("%c",znak[klavesa]);
57            sv++;
58            if ((sv==79)||(znak[klavesa]==10))
59            {
60                 sv=2;
61                 rv++;
62            }
63       }
64       outportb(0x21,inportb(0x21)&~0x02);
65       outportb(0x20,0x20);
66  }
67
68  void main()
69  {
70       clrscr();
71       inic_sport();
72       stare_pr = getvect(0x0c);
73       stare_prk = getvect(0x09);
```

```
74          setvect(0x0c,nove_pr);
75          setvect(0x09,nove_prk);
76          outportb(0x21,inportb(0x21)&~0x12);
77          while(klavesa != 1);
78          setvect(0x0c,stare_pr);
79          setvect(0x0c,stare_prk);
80    }
```

Vysvetlenie:

1-4: Prilinkovanie dôležitých knižníc k programu.

5: Zadefinovanie makra sledujúceho, či je register vysielača vyprázdnený (Obr. 7.14).

7: Deklarácia premennej `stare_pr` typu *interrupt* určenej pre starú obsluhu prerušenia sériovej linky.

8: Deklarácia premennej `nove_pr` typu *interrupt* určenej pre novú obsluhu prerušenia sériovej linky.

9: Deklarácia premennej `stare_prk` typu *interrupt* určenej pre starú obsluhu prerušenia klávesnice.

10: Deklarácia premennej `nove_prk` typu *interrupt* určenej pre novú obsluhu prerušenia klávesnice.

12: Deklarácia bázovej adresy sériového portu COM1.

13: Deklarácia premennej `prijmi` slúžiacej ku ukladaniu jednotlivých prijatých znakov.

14: Deklarácia premenných `klavesa` (kód stlačeného klávesa), `rp` (riadok kurzora bloku prijímača), `sp` (stĺpec kurzora bloku prijímača), `rv` (riadok kurzora bloku vysielača), `sv` (stĺpec kurzora bloku vysielača) typu *int*.

15: Deklarácia poľa `znak` reprezentujúceho mapu znakov na klávesnici (nie je zmapovaná celá klávesnica ale iba 58 jej znakov) typu *char*, jednotlivé prvky poľa: *medzera*, *ESC*, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, +, *BackSpace*, *medzera*, Q, W, E, R, T, Y, U, I, O, P, [,], *nový riadok*, }, A, S, D, F, G, H, J, K, L, ;, ", !, <, |, Z, X, C, V, B, N, M, *čiarka*,. ,?, >,_ , *medzera*, *medzera* (klávesy ako *Shift*, *Alt*, *Ctrl* boli nahradené inými znakmi).

17: Deklarácia a definovanie funkcie pre vyslanie znaku po sériovej linke: `vysli()`.

19: Prázdny cyklus čakajúci na vyprázdnenie registra vysielača.

20: Vyslanie znaku, ktorý je vstupným argumentom funkcie `vysli()` po sériovej linke pomocou registra 3F8H.

23: Deklarácia a definovanie funkcie inicializácie sériového portu: `inic_sport()`.

25: Zopnutie bitu DLAB (viď Obr. 7.12).

26, 27: Nastavenie registra deliteľa hodinovej frekvencie (LSB, MSB – viď Tab. 7.5) – nastavenie prenosovej rýchlosti.

28: Nastavenie zvyšných podmienok prenosu: dĺžka slova 7 bitov, 1 stop bit a párna parita pomocou hodnoty 1AH (viď Obr. 7.12).

29: Nastavenie registra povolenia prerušení (viď Obr. 7.9) – povolenie prerušenia od prijatého znaku (hodnota: 1).

32: Definovanie novej obsluhy prerušenia sériovej linky: `nove_pr()`.

34: Zakázanie prerušenia od sériovej linky (OCW 1, viď Obr. 2.13), aby počas behu obsluhy nemohlo nastať prerušenie znovu.

35: Do premennej `prijmi` sa vloží prijatý znak sériovou linkou z registra 3F8H.

36: Presun kurzora obrazovky do bloku prijímača.

37: Vypísanie hodnoty v premennej `prijmi` na obrazovku.

38: Inkrementovanie premennej `sp` reprezentujúcu stĺpec kurzora bloku prijímača.
39-43: Ak je kurzor na konci riadku alebo prijatý znak nadobúda hodnotu *nový riadok*, tak sa inkrementuje premenná `sr` reprezentujúca riadok bloku prijímača a do premennej `sp` sa vloží jednotka, keďže je potrebné kurzor dostať na začiatok nového riadku.
44: Povolenie prerušenia od sériovej linky (OCW 1, viď Obr. 2.13).
45: Ukončenie práve obsluhovaného prerušenia (OCW2, viď Obr. 2.14).
48: Definovanie novej obsluhy prerušenia klávesnice: `nove_prk()`.
50: Zakázanie prerušenia od klávesnice (OCW 1, viď Obr. 2.13), aby počas behu obsluhy nemohlo nastať prerušenie znovu.
51: Do premennej `klavesa` sa vloží kód zatlačeného klávesa z dátového registra klávesnice 60H.
52-63: Podmienka ktorá kontroluje kód klávesa, ak je kláves mimo rozsahu poľa `znak`, tak sa podmienka nevykoná.
  54: Spustenie funkcie `vysli()` s argumentom `znak[klavesa]`, čiže argumentom je zatlačený znak a nie kód klávesa.
  55: Presun kurzora obrazovky do bloku vysielača.
  56: Vypísanie hodnoty `znak[klavesa]` na obrazovku.
  57: Inkrementovanie premennej `sv` reprezentujúcu stĺpec kurzora bloku vysielača.
  58-62: Ak je kurzor na konci riadku alebo znak zatlačeného klávesa nadobúda hodnotu *nový riadok*, tak sa inkrementuje premenná `rv` reprezentujúca riadok bloku prijímača a do premennej `sv` sa vloží jednotka, keďže je potrebné kurzor dostať na začiatok nového riadku.
64: Povolenie prerušenia od klávesnice (OCW 1, viď Obr. 2.13).
65: Ukončenie práve obsluhovaného prerušenia (OCW2, viď Obr. 2.14).
68: Deklarácia a definovanie hlavného programu.
70: Vyčistenie obrazovky.
71: Spustenie funkcie `inic_sport()` pre inicializáciu sériového portu.
72: Uloženie starej obsluhy prerušenia sériovej linky do premennej `stare_pr`.
73: Uloženie starej obsluhy prerušenia klávesnice do premennej `stare_prk`.
74: Vloženie novej obsluhy prerušenia sériovej linky z premennej `nove_pr`.
75: Vloženie novej obsluhy prerušenia klávesnice z premennej `nove_prk`.
76: Povolenie prerušenia od klávesnice a sériovej linky (OCW 1, viď Obr. 2.13).
77: Cyklus ktorý sa opakuje, kým nie je stlačený prvý kláves na klávesnici (*ESC*).
78: Vloženie starej obsluhy prerušenia sériovej linky z premennej `stare_pr`.
79: Vloženie starej obsluhy prerušenia klávesnice z premennej `stare_prk`.

7.1.9 Programovanie v programovacom jazyku C#

V programovacom jazyku C# je možné naprogramovať sériové rozhranie rovnakým spôsobom ako to bolo uvedené v kapitole 7.1.7, lenže je potrebné do programu vložiť funkcie dynamickej knižnice *inpout32.dll*, tak ako uvádza kapitola 2.2.7 a pred začatím nastavovania (programovania) vstupno-výstupných registrov otvoriť sériovú linku:

```
serialPort.Open();
```

Takže sériové rozhranie sa dá programovať pomocou zápisu a čítania vstupno-výstupných registrov. Samozrejme kódy uvedené v kapitole 7.1.7 je potrebné pozmeniť tak,

aby vyhovovali prostrediu v ktorom sú programované. Najmä si treba dať pozor na funkcie `inportb()` a `outportb()`, ktoré boli nahradené novými funkciami, zadeklarovanými pri vkladaní knižnice *inpout32.dll* do programu.

Programovanie pomocou prerušení, tak ako to bolo uvedené v kapitole 7.1.8 nie je možné, nakoľko využívajú funkcie knižnice *dos.h*. Funkcie tejto knižnice nie sú kompatibilné s operačnými systémami pre ktoré bol programovací jazyk C# vyvinutý (Windows XP a vyššie rady OS Windows).

Okrem zápisu a čítania vstupno-výstupných registrov sa dá sériové rozhranie v programovacom jazyku C# programovať pomocou objektu:

SerialPort

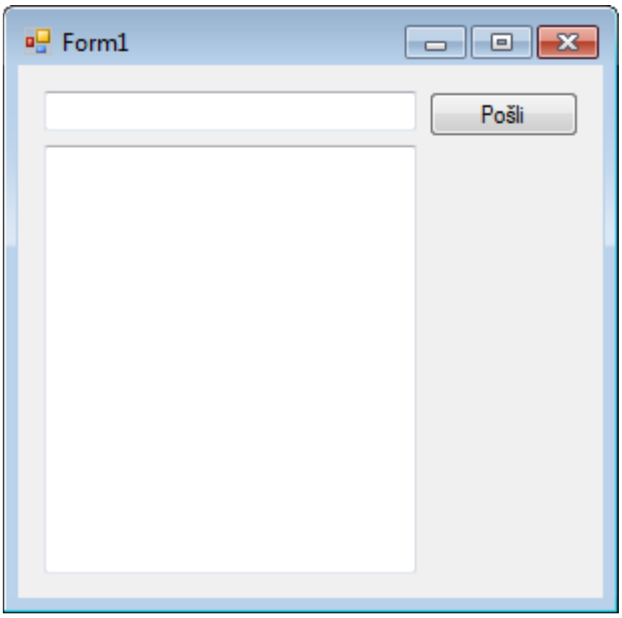
Tento objekt je súčasťou *Framework*-u *.NET* (dot NET), ktorý boli vyvinutý pre programovanie aplikácií v operačnom systéme Windows (Windows XP a vyššie rady OS Windows). V tejto podkapitole bude príklad jedného zadania, ktoré využíva objekt *SerialPort*.

Zadanie 1:

Pomocou rozhrania RS-232 vytvorte univerzálnu aplikáciu, ktorá sa bude chovať podľa potreby ako vysielač, alebo prijímač. Vo vrchnej časti okna aplikácie sa bude nachádzať textové pole, pomocou ktorého budete zapisovať text, ktorý sa bude zobrazovať v spoločnej komunikácií medzi dvoma počítačmi. Spoločná komunikácia medzi počítačmi sa bude nachádzať v spodnej časti okna aplikácie. Aplikáciu vytvorte pod OS Windows XP, pomocou programovacieho jazyka C#.

Analýza a návrh zadania:

V prvom rade je vhodné navrhnúť si vzhľad tejto aplikácie (Obr. 7.22):



Obr. 7.22 Vzhľad návrhu aplikácie

Podľa navrhnutého vzhľadu je viditeľné, že potrebujeme 3 objekty a to 2 textové polia a jedno tlačidlo. Okrem viditeľných objektov ešte potrebujeme spomínaný objekt *SerialPort*. Týmto objektom je vhodné zmeniť niektoré vlastnosti (*Properties*) pre lepší prehľad a prácu s nimi:

- textBox1: bez zmeny,
- textBox2: Multiline: true,
- button1: Name: button,
  Text: Pošli,
- serialPort1: Name: serialPort,
  zvyšne vlastnosti ostávajú bez zmeny (štandardne - default),
  pridať udalosť (*Event*): DataReceived.

Tieto zmeny sa vykonajú v možnostiach *Properties* vo vývojovom prostredí. Po týchto nastaveniach sa môže začať programovať.

V prvom rade je potrebne otvoriť (zapnúť) sériové rozhranie pomocou metódy `Open()` popri inicializácií všetkých komponentov:

```
serialPort.Open();
```

Potom je potrebné naprogramovať program, ktorý sa vykoná po stlačení tlačidla s textom „Pošli“. Najprv je potrebné zadeklarovať premennú do ktorej sa bude vkladať posielaný text, nakoľko by nemal byť typu *string* (ako je text v textBox-e), ale malo by to byť pole bajtov (*byte*). Po zadeklarovaní premennej sa skontroluje, či je sériové rozhranie stále otvorené, ak je stále otvorené, tak sa dokončí program pre tlačidlo. Do pripravenej premennej sa prekonvertuje a vloží text z *textBox1*, následne pomocou metódy `Write()` objektu *SerialPort* pošle dáta po sériovej linke, text ktorý sa poslal po sériovej linke sa ešte vloží do objektu *textbox2*, aby sa aplikácia správala ako „chat“.

Keďže vo vývojovom prostredí bola pridaná udalosť (*Event*), tak pre túto udalosť sa tiež napíše kód. Táto udalosť nastane ak sa po sériovej linke príjmu dáta. Aj v tomto prípade treba najskôr zadeklarovať pole typu *byte* tento raz pre prijímaný text. Pred prijímaním textu sa skontroluje, že či je stále linka otvorená a či sú stále dáta v buffer-y sériového rozhrania, ak tieto podmienky platia pokračuje sa ďalej. Pomocou metódy `Read()` objektu *SerialPort* sa vyčítajú dáta z buffer-a sériovej linky a vložia do spomínanej premennej, potom sa dáta v premennej prekonvertujú na text a vložia do objektu *textbox2*.

Okrem udalosti *DataReceived* sa mohol využiť objekt *Timer*, ktorý by po istých časových intervaloch kontroloval stav buffer-a sériovej linky, takže kód pre udalosť *Tick* tohto objektu by bol rovnaký, ako kód udalosti *DataReceived* objektu *SerialPort*.

Riešenie:

```
1    using System;
2    using System.Text;
3    using System.Windows.Forms;
4
5    namespace SerioveRozhranie
6    {
7      public partial class Form1 : Form
8      {
9        public Form1()
10       {
11           InitializeComponent();
```

```
12              serialPort.Open();
13          }
14          private void button_Click(object sender, EventArgs e)
15          {
16              byte[] buffer = new byte[255];
17              if (serialPort.IsOpen)
18              {
19                  buffer =
                    ASCIIEncoding.UTF8.GetBytes(textBox1.Text);
20                  serialPort.Write(buffer, 0,
                    textBox1.TextLength);
21                  textBox2.Text += "\r\n Tento PC: " +
                    textBox1.Text;
22              }
23          }
24          private void serialPort_DataReceived(object sender,
            System.IO.Ports.SerialDataReceivedEventArgs e)
25          {
26              byte[] buffer = new byte[255];
27              if(serialPort.IsOpen&&serialPort.BytesToRead>0)
28              {
29                  serialPort1.Read(buffer, 0,
                    serialPort1.BytesToRead);
30                  textBox2.Text += "\r\n Druhý PC: " +
                    ASCIIEncoding.UTF8.GetString(buffer);
31              }
32          }
33      }
34  }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.
5: Deklarácia a definovanie namespace-u tohto programu.
7: Deklarácia a definovanie hlavnej triedy programu.
9: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.
11: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).
12: Otvorenie sériového portu pomocou metódy `Open()`.
14: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla.
16: Deklarácia poľa `buffer` typu *byte*.
17-22: Podmienka ktorá sa splní, ak je sériový port otvorený.
- 19: Do premennej `buffer` sa vloží prekonvertovaný text z objektu *textBox1*.
- 20: Odoslanie textu po sériovej linke pomocou metódy `Write()`.
- 21: Vypísanie odoslaného textu do objektu *textBox2*.

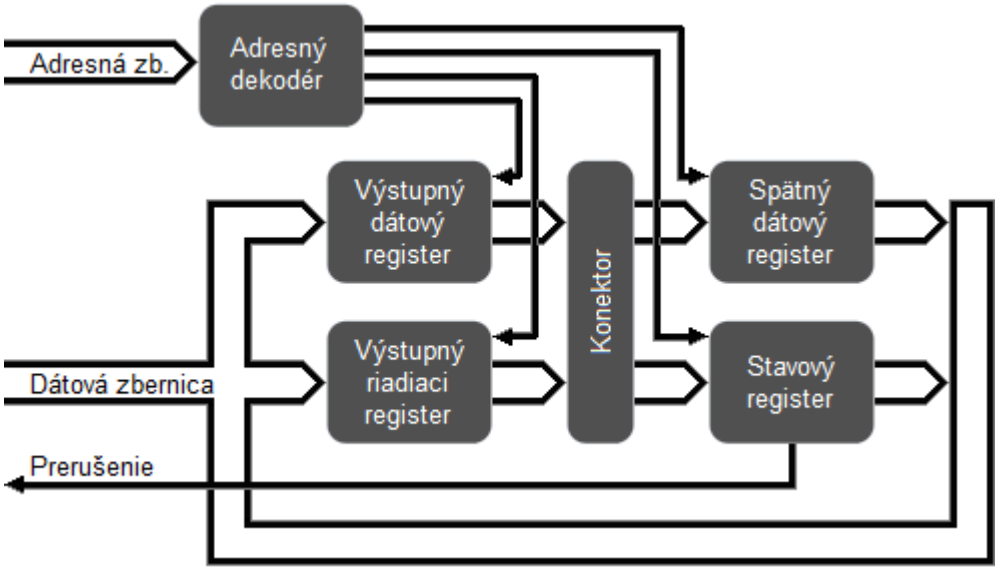
24: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po prijatí dát po sériovej linke.
26: Deklarácia poľa `buffer` typu *byte*.
27-31: Podmienka ktorá sa splní, ak je sériový port otvorený a zároveň existujú dáta prijaté sériovou linkou.
- 29: Vloženie prijatých dát sériovou linkou do premennej `buffer`.
- 30: Prekonvertovanie a vypísanie prijatého textu do objektu *textBox2*.

7.2 Štandardné rozhranie Centronics

Centronics je paralelné rozhranie, ktoré prioritne slúžilo ako rozhranie medzi počítačom a tlačiarňou, neskôr aj medzi počítačom a skenerom. Teraz ho už v týchto pozíciách (tlačiarne a skenery) nahradilo rozhranie USB. Centronics sa stále inštaluje do počítačov a je súčasťou SouthBridge-u. Toto rozhranie sa využíva najmä preto, lebo využíva logiku TTL a tým pádom sa dá používať ako digitálny vstup a digitálny výstup počítača. Toto rozhranie sa používa aj na nahrávanie programov do niektorých typov jednočipových mikropočítačov.

Tradične bola dátová zbernica paralelného portu jednosmerná – dáta sa dali prenášať len z počítača do tlačiarne. Napriek tomu boli na porte 4 riadiace signály obojsmerné, čo sa často využívalo najmä pri prepojení dvoch PC. V klasickom paralelnom porte je možnosť do dátového registra nielen zapisovať, ale aj čítať. Hardvér portu je však skonštruovaný tak, že jednotlivé bity dátového portu sú trvalo budené dvoj-činným výstupom, takže nie je možné bez poškodenia vnútiť portu stav zvonka, a čítanie len vrátilo hodnotu naposledy zapísanú do dátového registra. Pomerne jednoduchou úpravou sa dalo aj u pôvodného paralelného portu (ktorý bol skonštruovaný z bežných TTL logických obvodov) dosiahnuť, že pri nastavení jedného z bitov riadiaceho registra prešiel budič dátovej linky do vysoko impedančného stavu, čo umožňovalo prečítať priamo stav dátových pinov, t. j. sa port v tom okamihu správal ako vstupný. Takže pomocou spomínaného bitu v riadiacom registri sa môže paralelný port (jeho dátové piny) správať ako *vstupný*, *výstupný*, alebo *obojsmerný*. Tento režim implementovali aj viacerí výrobcovia rozširujúcich kariet, ale nikdy nebol formálne štandardizovaný.

Bloková schéma adaptéru je na nasledujúcom obrázku (Obr. 7.23):



Obr. 7.23 Bloková schéma adaptéru (Centronics)

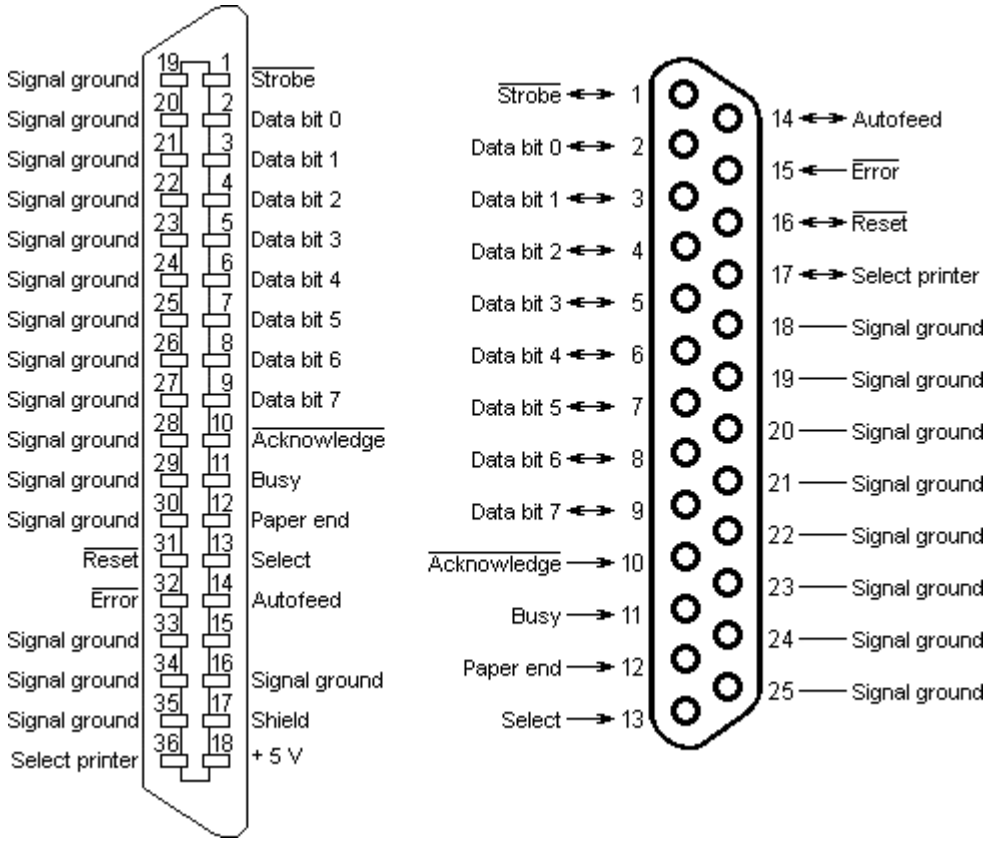
Tento adaptér bol len výnimočne realizovaný na samostatnej doske. Omnoho častejšie bol súčasťou iného adaptéru (napríklad staré grafické karty, viď kapitolu 2.5.6, konektor CANNON s 25 pinmi sa nazýval aj DB-25, alebo LPT). Aj keď sa jedná o veľmi jednoduché paralelne rozhranie s obojsmerným kvitovaním (Strobe - výstup, Busy - vstup), nehodí sa k jeho realizácií žiadny z kvitovacích (potvrdzovacích) režimov základných

paralelných prepojovacích obvodov. Model PC-XT obsahoval jedno paralelné rozhranie (port), a novšie modely mali väčšinou dve paralelné rozhrania, kým sa do počítačov nezačalo inštalovať rozhranie USB. Odvtedy sa do počítačov začalo inštalovať iba jedno paralelné rozhranie (prípadne ani jedno).

Rozhranie Centronics bolo štandardizované ako IEEE-1284, ale omnoho neskôr ako sa začalo používať v PC, tento štandard už podporoval obojsmerný prenos dát. Rozhranie Centronics bolo realizované rôznymi obvodmi od 74LS374, 74LS174, 74LS244 po 74VHC161284. Signály rozhrania pracujú s napäťovými úrovňami logiky TTL a otvoreným konektorom a preto dĺžka prepojovacieho vedenia nesmie presiahnuť 2 m.

7.2.1 Konektory a signály

Existujú 2 typy konektorov, pre toto rozhranie, 36 nožičkový Amphenol a 25 nožičkový CANNON. Rozmiestnenie pinov (vodičov) na jednotlivých typoch konektorov je na obrázku Obr. 7.24



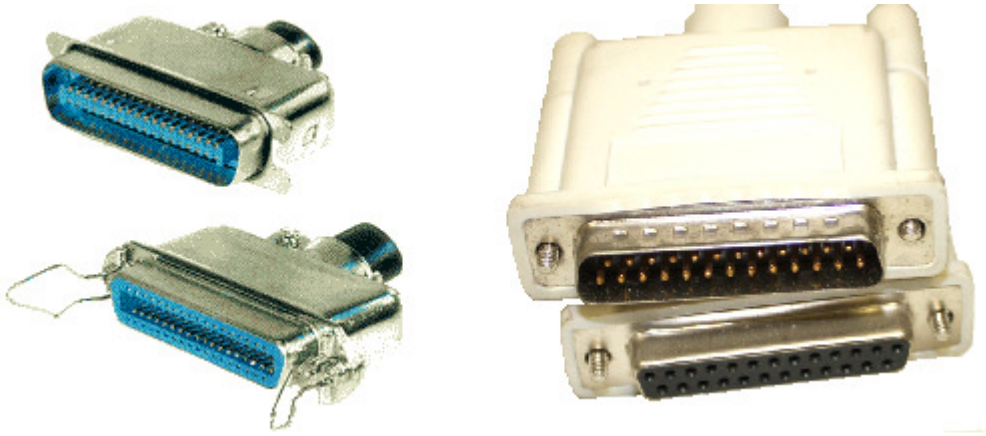
Obr. 7.24 Rozloženie pinov na konektoroch Amphenol (v ľavo) a CANNON (v pravo) rozhrania Centronics

Význam jednotlivých signálov (kontaktov) je vysvetlený v nasledujúcej tabuľke (Tab. 7.6).

Tab. 7.6 Význam pinov na konektoroch rozhrania Centronics

Kontakt (pin) Amphenol	Kontakt (pin) CANNON	Signál	Popis
1	1	Strobe	Riadiaci signál, aktívna úroveň je L.
2 – 9	2 – 9	Data bit 0 – 7	Dátové signály.
10	10	Acknowledge	Stavový signál, aktívna úroveň je L.
11	11	Busy	Stavový signál, aktívna úroveň je H a oznamuje, že zariadenie nie je pripravené k prenosu.
12	12	Paper end	Stavový signál, aktívna úroveň je H a oznamuje koniec papiera.
13	13	Select	Stavový signál, aktívna úroveň je H a oznamuje, že tlačiareň je on-line.
14	14	Autofeed	Riadiaci signál, aktívna úroveň je L a ak je signál aktívny automaticky tlačiareň po prijatom znaku <CR> doplňuje <LF>.
15	–	–	–
16, 19 – 30, 33 – 35	18 – 25	Signal ground (GND)	Signálová zem.
17	–	Shield	Tienenie.
18	–	+ 5 V	Napájanie +5V.
31	16	Reset	Riadiaci signál (inač INIT ako inicializácia), aktívna je úroveň L a uvádza tlačiareň do počiatočného stavu.
32	15	Error	Stavový signál, aktívna úroveň je L a oznamuje, že tlačiareň hlási chybu, alebo je odpojená.
36	17	Select printer	Riadiaci signál pre pripojené zariadenie, aktívnou úrovňou H sa žiada uvedenie do stavu ON-LINE.

Vzhľad konektorov Amphenol a CANNON je na obrázku Obr. 7.25. Tieto konektory využívajú rovnaké signály (viď Obr. 7.24 a Tab. 7.6) a preto sa môžu akokoľvek koncovky konektorov kombinovať (podľa potreby).



Obr. 7.25 Vzhľad konektorov Amphenol (v ľavo) a CANNON (v pravo) rozhrania Centronics

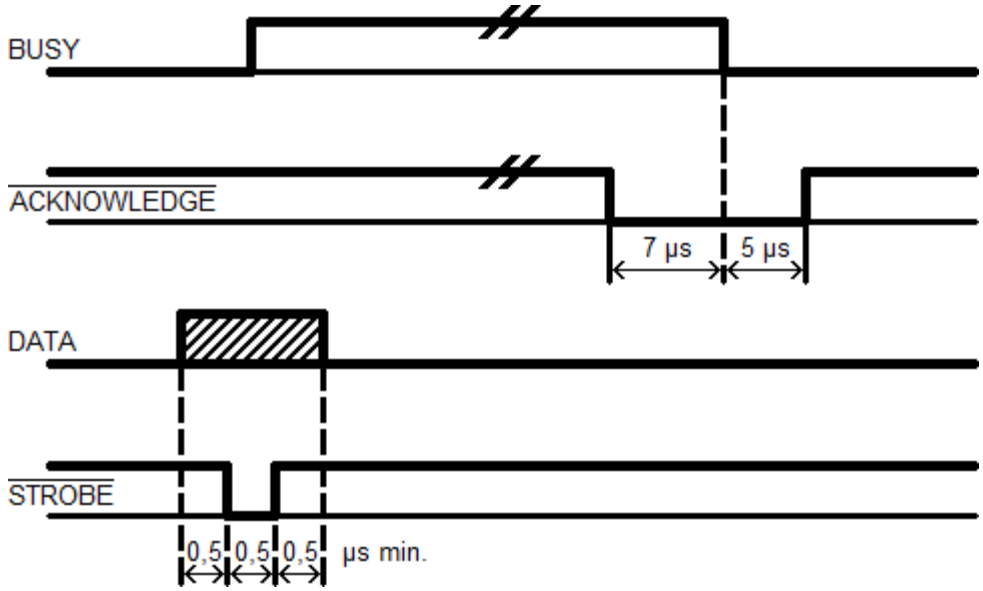
Prenos dát je na tomto rozhraní osembitové, asynchrónne, s obojsmerným kvitovaním. Funkčne môžeme jeho signály rozdeliť do troch skupín:

- dátové signály (Data bit 0 až 7),
- riadiace a stavové signály rozhrania (Strobe, Busy, Acknowledge),
- riadiace a stavové signály pre pripojené zariadenie (Autofeed, Reset (INIT), Select printer, Paper end, Error, Select).

Podľa smeru komunikácie, registrov a charakteru signálov môžeme signály taktiež rozdeliť do troch skupín:

- Dátové signály (Data bit 0 až 7),
- Riadiace signály (Strobe, Autofeed, Reset (INIT), Select printer),
- Stavové signály (Busy, Acknowledge, Paper end, Error, Select).

Na obrázku Obr. 7.26 je uvedený signálový sled pri prenose dát medzi tlačiarňou a počítačom.



Obr. 7.26 Signálový sled pri prenose dát medzi tlačiarňou a počítačom

7.2.2 Adresovanie a registre

Brány (adresy) adaptéru pre pripojenie periférnych zariadení (Tab. 7.7):

Tab. 7.7 Vstupno-výstupné adresy brán paralelného rozhrania

1. paralelný port	2. paralelný port	3. paralelný port	Význam pri	
			čítaní	zápise
378H	278H	3BCH	Dátový register	
379H	279H	3BDH	Stavový register	
37AH	27AH	3BEH	Riadiaci register	

Paralelné porty 1 a 2 sa najčastejšie používali na pripojenie tlačiarní a neskôr skenerov, ale samozrejme dajú sa využiť aj na pripojenie iných periférií, ako napríklad prepojenie dvoch PC, pripojenie jednočipového mikropočítača, atď. Paralelný port 3 sa využíval najmä na pripojenie obrazovky (MDA, Hercules), dnes sa táto bázová adresa dá použiť na pripojenie ďalšieho paralelného portu.

Aj toto rozhranie môže vyvolať prerušenie. Ak je pripojeným periférnym zariadením tlačiareň, tak k prerušeniu dôjde vtedy, keď je tlačiareň pripravená na vloženie ďalšieho znaku na tlačenie. V prípade, že pripojeným zariadením nie je tlačiareň a je potrebné, aby vyvolávalo prerušenie, tak dané zariadenie musí simulovať rovnaký stav (rovnaké nastavenie stavového registra) ako tlačiareň, ktorá je pripravená na vloženie ďalšieho znaku na tlačenie. Dokonca ani každá tlačiareň, ktorá sa pripájala k počítaču pomocou paralelného portu nedokázala toto prerušenie vyvolať, pretože nenastal potrebný stav stavového registra. Najčastejšie sa tieto tlačiarne programovali pomocou priameho zápisu do registrov (brány adaptéru paralelného rozhrania) a pomocou softvérového prerušenia BIOS-u. Tieto zariadenia vyvolávali tieto hardvérové prerušenia:

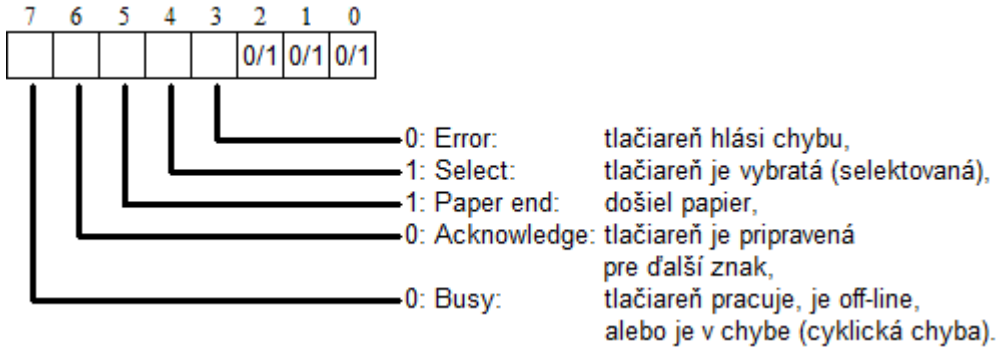
- LPT 1 (378H): IRQ 7,
- LPT 2 (278H): IRQ 5,
- Adaptér displeja (3BCH):
  - ak sa využíva ako paralelný port a je potrebné na obsluhu zariadenia prerušenie, treba ho namapovať na voľné IRQ,
  - ak sa využíva ako adaptér displeja, tak nastavovanie výstupných registrov má na starosti hardvér a vstupné sa nastavujú od adresy 3BAH po 3BCH podľa želaného nastavenia adaptéra.

Paralelné rozhranie využíva 3 registre spomenuté v Tab. 7.7. Význam jednotlivých bitov v registroch:

1. register: Dátový register (378H – zápis a čítanie):

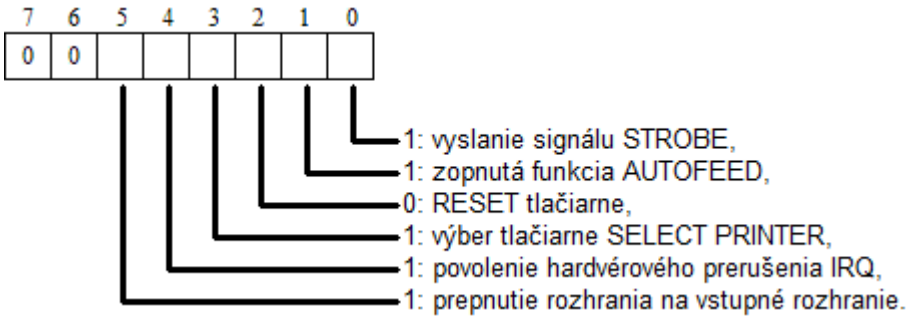
- ak je 5. bit riadiaceho registra 0:
  - zápis: poslať bajt tlačiarni, alebo inému periférnemu zariadeniu,
  - čítanie: prečítanie posledného vyslaného bajtu,
- ak je 5. bit riadiaceho registra 1:
  - zápis: zakázaný, pretože jednotlivé bity dátového portu sú trvalo budené dvoj-činným výstupom, takže nie je možné bez poškodenia vnútiť portu iný stav ako je prijímaný,
  - čítanie: prečítanie prijímaného bajtu.

2. register: Stavový register (379H – iba čítanie, Obr. 7.27):



Obr. 7.27 Stavový register paralelného rozhrania

3. register: Riadiaci register (ovládanie tlačiarne, 37AH – zápis a čítanie, Obr. 7.28):



Obr. 7.28 Riadiaci register paralelného rozhrania

7.2.3 Programovanie

Táto podkapitola obsahuje 6 zadaní. Prvé dve zadania sa venujú programovaniu rozhrania Centronics pomocou zápisu do registrov (vstupno-výstupné porty), prvé zadanie je programované v programovacom jazyku C a druhé je programované v programovacom jazyku C# v prostredí Visual Studio 2010. Popis registrov potrebných pre programovanie pomocou zápisu do registrov (portov) je v predchádzajúcej podkapitole 7.2.2 a popis prilinkovania dynamickej knižnice *inpout32.dll* do projektu programovaného jazykom C# je v kapitole 2.2.7. Tretie a štvrté zadanie využívajú prerušenia, pričom tretie využíva hardvérové prerušenie od tlačiarne (IRQ7) a štvrté využíva softvérové prerušenie, konkrétne službu BIOS-u INT 17H. Programovaniu hardvérových prerušení sa venovali už dve kapitoly (2.3.7 a 7.1.8), pri softvérových prerušeniach je potrebné popísať službu BIOS-u, ktorá sa venuje obsluhe tlačiarní. Spomínaná služba je popísaná v ďalších odstavcoch tejto podkapitoly. Piate a šieste zadanie poukazujú nato, že sa rozhranie Centronics dá využiť v riadení ako digitálny vstup a výstup, toto využitie je naznačené pomocou laboratórnych prípravkov (Obr. 7.29), tieto prípravky sú bližšie popísané v analýzach jednotlivých zadaní.

Obr. 7.29 Laboratórne prípravky pre prácu s rozhraním Centronics

Služba BIOS-u (INT 17H) zaisťuje plnú podporu až štyroch paralelných tlačiarní. Pre priradenie štvrtej tlačiarne je potrebné bázovú adresu zapísať do dátového slova BIOS-u na adrese 0:040E. Ak sa často tlačilo na dvoch LPT tlačiarňach naraz to isté a počítač mal dva paralelné porty, tak sa dala služba BIOS-u (INT 17H) zmiasť tým, že jednoducho sa prehodili bázové adresy v spodnej oblasti operačnej pamäte (napríklad prepis hodnoty z adresy pamäte 0:0408 na adresu pamäte 0:040A). Samozrejme je to možné spraviť aj teraz, ale už sa tlačiarne prepájajú častejšie rozhraním USB.

Bázové adresy portu tlačiarne sú uložené v operačnej pamäti na adresách 0:0408 (LPT1), 0:040A (LPT2), 0:040C (LPT3) a 0:040E (LPT4). Hodnoty timeout-u tlačiarne sú na adresách 0:0478 (LPT1), 0:0479 (LPT2), 0:047A (LPT3), 0:047B (LPT4).

.

V kapitole 2.3.7 a 7.1.8 bolo vysvetlené ako sa dá využiť obslužný program prerušenia aj s parametrami, takže sa priamo rozoberú parametre obslužného programu paralelného rozhrania na adrese INT 14H (softvérové prerušenie):

Služba (ah) 00H: Tlačenie znaku:

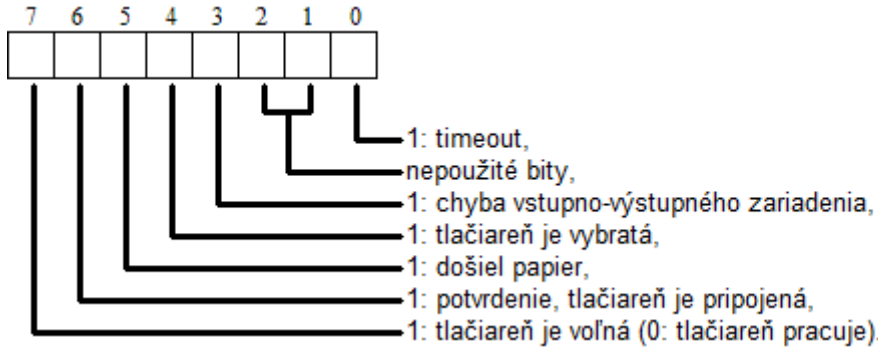
- vstup: al – znak v ASCII kóde, ktorý sa má vytlačiť,
- vstup: dx – číslo tlačiarne (0, 1, 2, alebo 3),
- výstup: ah – stavové vlajky tlačiarne (ak sa vyskytol timeout, tak je na výstupe len 01H, viď Obr. 7.30),

Služba (ah) 01H: Inicializácia portu tlačiarne:

- vstup: dx – číslo tlačiarne (0, 1, 2, alebo 3),
- výstup: ah – stavové vlajky tlačiarne (viď Obr. 7.30),

Služba (ah) 02H: Zistenie stavu tlačiarne:

- vstup: dx – číslo tlačiarne (0, 1, 2, alebo 3),
- výstup: ah – stavové vlajky tlačiarne (viď Obr. 7.30):



Obr. 7.30 Význam bitov výstupného parametra ah služieb 00H, 01H a 02H v obslužnom programe INT 17H

Pred začatím programovania je nutné podotknúť, že ak pripájate alebo odpájate tlačiareň (alebo iné aktívne zariadenie – zariadenie s vlastným napájaním) pri zapnutom počítači, môžete zničiť paralelne rozhranie a niekedy sa zároveň zničí aj sériové rozhranie, ktoré býva napájané z rovnakého zdroja.

Zadanie 1:

Naprogramujte program, ktorý pomocou rozhrania Centronics a ihličkovej tlačiarne vytlačí text, ktorý sa zadá ako vstupná premenná (pole) hneď po zapnutí programu. Program programujte pomocou zápisu do registrov (vstupno-výstupných portov) a v programovacom jazyku C pod operačným systémom MS-DOS.

Analýza zadania:

Keďže sa budú používať funkcie `inportb()` a `outportb()`, ktoré sú súčasťou knižnice *dos.h*, tak túto knižnicu je nutné prilinkovať k programu spolu so vstupno-výstupnou knižnicou *stdio.h*.

V prvom rade treba zadeklarovať pole znakov do ktorých sa vloží zvolený reťazec určený na tlačenie a prirodzené číslo, ktoré bude tento reťazec (pole) pri tlačení indexovať. Potom je vhodné vyčistiť obrazovku, aby bol program prehľadnejší, a vypísať text, ktorý užívateľa vyzve k zadaniu reťazca. Tento reťazec sa pomocou funkcie `fgets()` vloží do zadeklarovaného poľa.

Keď už je reťazec (pole) pripravené k vytlačeniu je vhodné zinicializovať tlačiareň pomocou riadiaceho slova 37AH (viď Obr. 7.28). Následne sa skontroluje či tlačiareň nehlási chybu, ak nehlási, tak sa začne s tlačením reťazca.

Reťazec sa bude tlačiť v cykle s podmienkou na konci, táto podmienka bude kontrolovať či tlačenie nedošlo ku koncu reťazca. V cykle sa pošle pomocou dátového registra 378H znak určený na tlačenie, potom sa na dĺžku jedného programového cyklu vyšle bit STROBE, program počká kým nie je tlačiareň pripravená na ďalší znak, ak je pripravená skontroluje sa či nedošlo k chybe a inkrementuje sa index reťazca.

Po ukončení cyklu sa ešte vyšle tlačiarni znak *nového riadku* dátovým registrom spolu s bitom STROBE na jeden programový cyklus.

V programe sa dvakrát zisťuje, že či nedošlo k chybe, ak náhodou k chybe došlo, spustí sa funkcia, ktorá na základe stavového registra 379H vypíše dané chyby (alebo chybu).

Riešenie:

```
1    #include<stdio.h>
2    #include<dos.h>
3
4    void chyba(int stav)
5    {
6         if (!(stav&0x08)) printf("\n Tlaciaren hlasy chybu");
7         if (!(stav&0x10)) printf("\n Tlac. nie je selekt.");
8         if ((stav&0x20))  printf("\n Dosiel papier");
9         if (!(stav&0x40)) printf("\n Tlac. nie je priprav.");
10        if (!(stav&0x80)) printf("\n Tlac. je zaneprazd.");
11        getchar();
12        exit();
13   }
14
15   void main()
16   {
17   char text[70];
18   int i=0;
19   clrscr();
20   printf("Zadajte text na tlačenie:\n");
21   fgets(text,sizeof(text),stdin);
```

```
22
23   outportb(0x37A,8);
24   if(inportb(0x379)!=223) chyba(inportb(0x379));
25
26   do
27   {
28        outportb(0x378,text[i]);
29        outportb(0x37A,5);
30        outportb(0x37A,4);
31        while(inportb(0x379)<192);
32        i++;
33        if(inportb(0x379)!=223) chyba(inportb(0x379));
34   }
35   while(text[i]!=0);
36
37   outportb(0x378,'\n');
38   outportb(0x37A,5);
39   outportb(0x37A,4);
40   }
```

Vysvetlenie:

1-2: Prilinkovanie dôležitých knižníc k programu.

4: Deklarácia a definovanie funkcie `chyba()`, ktorá sa spustí ak nastanie chyba pri tlačení reťazca.

6: Ak 3. bit vstupného argumentu funkcie `chyba()` je nulový vypíše sa hlásenie chyby tlačiarne.

7: Ak 4. bit vstupného argumentu funkcie `chyba()` je nulový vypíše sa hlásenie, že tlačiareň nie je selektovaná.

8: Ak 5. bit vstupného argumentu funkcie `chyba()` je jedna vypíše sa hlásenie o chýbajúcom papieri.

9: Ak 6. bit vstupného argumentu funkcie `chyba()` je nulový vypíše sa hlásenie, že tlačiareň nie je pripravená.

10: Ak 7. bit vstupného argumentu funkcie `chyba()` je nulový vypíše sa hlásenie o zaneprázdnenosti tlačiarne.

11: Funkcia čakajúca na stlačenie ľubovoľného klávesa.

12: Funkcia ukončujúca celý program.

15: Deklarácia a definovanie hlavného programu.

17: Deklarácia poľa znakov (reťazca) s názvom `text`.

18: Deklarácia premennej `i` typu *int* (index reťazca).

19: Vyčistenie obrazovky.

20: Vypísanie textu: „Zadajte text na tlačenie“, na obrazovku.

21: Vloženie vypísaného textu do reťazca `text`.

23: Inicializácia tlačiarne pomocou riadiaceho registra (37AH, viď Obr. 7.28) rozhrania Centronics.

24: Ak je tlačiareň v chybe, tak sa spustí funkcia `chyba()`.

26-35: Cyklus s podmienkou na konci, ktorý prejde všetky znaky reťazca až po znak ukončujúci reťazec (*null*).

- 28: Vloženie znaku v reťazci `text` s indexom `i` do dátového registra (378H) rozhrania Centronics.
- 29: Aktivácia bitu STROBE pomocou riadiaceho registra (37AH).

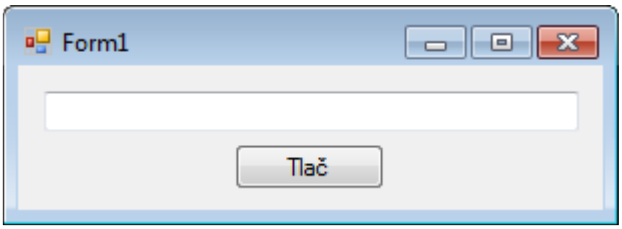
30: Deaktivácia bitu STROBE pomocou riadiaceho registra (37AH).
31: Čakanie na pripravenosť tlačiarne na ďalší znak.
32: Inkrementácia indexu `i`.
33: Ak je tlačiareň v chybe, tak sa spustí funkcia `chyba()`.
37: Vloženie znaku *nový riadok* do dátového registra (378H) rozhrania Centronics.
38: Aktivácia bitu STROBE pomocou riadiaceho registra (37AH).
39: Deaktivácia bitu STROBE pomocou riadiaceho registra (37AH).

Zadanie 2:

Naprogramujte program, ktorý pomocou rozhrania Centronics a ihličkovej tlačiarne vytlačí po stlačení tlačidla text v *textBox*-e. Program programujte pomocou zápisu do registrov (vstupno-výstupných portov) a v programovacom jazyku C# pod operačným systémom Windows XP (alebo pod vyššou verziou operačného systému Windows).

Analýza zadania:

Najprv je vhodné navrhnúť vzhľad aplikácie, ktorá bude obsahovať jedno tlačidlo (*button*) a jeden blok textu (*textBox*), návrh je na obrázku Obr. 7.31:



Obr. 7.31 Návrh vzhľadu aplikácie pre tlač textu

Do samotnej aplikácie je nutné prilinkovať funkcie `Out32()` a `Inp32()` z dynamickej knižnice inpout32.dll, postup a samotný kód pre vloženie funkcií z dynamickej knižnice je v kapitole 2.2.7, nakoľko sú to funkcie pre adresovanie periférnych zariadení. Nový názov týchto funkcií sa zvolí `outportb()` a `inportb()`, aby sa kódy predchádzajúceho a tohto zadania dali ľahko porovnať.

Ďalej je potrebné naprogramovať program, ktorý sa vykoná po stlačení tlačidla s textom „Tlač“ (*Event*). Najprv je vhodné zinicializovať tlačiareň pomocou riadiaceho slova 37AH (viď Obr. 7.28). Následne sa skontroluje či tlačiareň nehlási chybu, ak nehlási, tak sa začne s tlačením reťazca. Reťazec sa bude tlačiť v cykle s pevným počtom opakovaní (*for*), ktorý sa bude opakovať podľa dĺžky reťazca. V cykle sa pošle pomocou dátového registra 378H znak určený na tlačenie, potom sa na dĺžku 10 000 programových cyklov vyšle bit STROBE. Ak je procesor počítača dostatočne pomalý (programový cyklus je dlhší ako 0,5μs), tak postačuje jeden programový cyklus, tak ako to bolo v predchádzajúcom zadaní. Potom program počká, kým nie je tlačiareň pripravená na ďalší znak, ak je pripravená skontroluje sa či nedošlo k chybe.

Po ukončení cyklu sa ešte vyšle tlačiarni znak *nového riadku* dátovým registrom spolu s bitom STROBE na 10 000 programových cyklov (ak je procesor pomalší postačuje menej cyklov).

V programe sa dvakrát zisťuje, že či nedošlo k chybe, ak náhodou k chybe došlo, spustí sa funkcia, ktorá na základe stavového registra 379H vypíše dané chyby (alebo chybu) pomocou *MessageBox*-u a po stlačení tlačidla *OK* v *MessageBox*-e sa aplikácia vypne.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
5    namespace Tlaciaren
6    {
7        public partial class Form1 : Form
8        {
9            [DllImport("inpout32.dll", EntryPoint="Out32")]
             public static extern void outportb(intadd,intval);
10           [DllImport("inpout32.dll", EntryPoint="Inp32")]
             public static extern int inportb(int add);
11
12           public Form1()
13           {
14               InitializeComponent();
15           }
16
17           void chyba(int stav)
18           {
19               string text="";
20               if (((stav/8)%2)==0)
                 text += "\n Tlaciaren hlasy chybu";
21               if (((stav/16)%2)==0)
                 text += "\n Tlaciaren nie je selektovana";
22               if (((stav/32)%2)==1)
                 text += "\n Dosiel papier";
23               if (((stav/64)%2)==0)
                 text += "\n Tlaciaren nie je pripravena";
24               if (((stav/128)%2)==0)
                 text += "\n Tlaciaren je zaneprazdnena";
25               if (MessageBox.Show(text,"Chyby:")==
                 DialogResult.OK) Application.Exit();
26           }
27
28           private void button_Click
             (object sender, EventArgs e)
29           {
30               outportb(0x37A,8);
31               if(inportb(0x379)!=223)
                 chyba(inportb(0x379));
32               for (int i = 0; i<textBox.Text.Length; i++)
33               {
34                   outportb(0x378, textBox.Text[i]);
35                   outportb(0x37A, 5);
36                   for (int j = 0; j < 10000; j++);
37                   outportb(0x37A, 4);
38                   while (inportb(0x379) < 192) ;
39                   if (inportb(0x379) != 223)
                     chyba(inportb(0x379));
```

```
40                  }
41
42                  outportb(0x378,'\n');
43                  outportb(0x37A,5);
44                  for (int j = 0; j < 10000; j++);
45                  outportb(0x37A,4);
46              }
47          }
48      }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

5: Deklarácia a definovanie namespace-u tohto programu.

7: Deklarácia a definovanie hlavnej triedy programu.

9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `outportb()` (pre podobnosť s predchádzajúcim zadaním).

10: Vloženie funkcie `Inp32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `inportb()` (pre podobnosť s predchádzajúcim zadaním).

12: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

14: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).

17: Deklarácia a definovanie funkcie `chyba()`, ktorá sa spustí ak nastanie chyba pri tlačení reťazca.

19: Deklarácia prázdneho reťazca `text` do ktorého sa vkladajú jednotlivý chyby, ktoré nastali.

20: Ak 3. bit vstupného argumentu funkcie `chyba()` je nulový do reťazca `text` sa priloží hlásenie chyby tlačiarne.

21: Ak 4. bit vstupného argumentu funkcie `chyba()` je nulový do reťazca `text` sa priloží hlásenie, že tlačiareň nie je selektovaná.

22: Ak 5. bit vstupného argumentu funkcie `chyba()` je jedna do reťazca `text` sa priloží hlásenie o chýbajúcom papieri.

23: Ak 6. bit vstupného argumentu funkcie `chyba()` je nulový do reťazca `text` sa priloží hlásenie, že tlačiareň nie je pripravená.

24: Ak 7. bit vstupného argumentu funkcie `chyba()` je nulový do reťazca `text` sa priloží hlásenie o zaneprázdnenosti tlačiarne.

25: Otvorenie *MessageBox*-u do ktorého sa vypíše celý reťazec `text` (všetky chyby) a keď sa stlačí tlačidlo *OK* v *MessageBox*-e, tak sa vypne celá aplikácia.

28: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla.

30: Inicializácia tlačiarne pomocou riadiaceho registra (37AH, viď Obr. 7.28) rozhrania Centronics.

31: Ak je tlačiareň v chybe, tak sa spustí funkcia `chyba()`.

32-39: Cyklus s pevným počtom opakovaní s indexom `i`, ktorý prejde všetkými znakmi reťazca.

- 34: Vloženie znaku v reťazci text s indexom `i` do dátového registra (378H) rozhrania Centronics.
- 35: Aktivácia bitu STROBE pomocou riadiaceho registra (37AH).
- 36: Ponechaný aktívny bit STROBE na 10 000 programových cyklov (tento cyklus nie je potrebný, ak je procesor počítača pomalší).
- 37: Deaktivácia bitu STROBE pomocou riadiaceho registra (37AH).

38: Čakanie na pripravenosť tlačiarne na ďalší znak.
39: Ak je tlačiareň v chybe, tak sa spustí funkcia `chyba()`.
42: Vloženie znaku *nový riadok* do dátového registra (378H) rozhrania Centronics.
43: Aktivácia bitu STROBE pomocou riadiaceho registra (37AH).
44: Ponechaný aktívny bit STROBE na 10 000 programových cyklov.
45: Deaktivácia bitu STROBE pomocou riadiaceho registra (37AH).

Zadanie 3:

Naprogramujte program, ktorý pomocou rozhrania Centronics a ihličkovej tlačiarne vytlačí text, ktorý sa zadá ako vstupná premenná (pole) hneď po zapnutí programu. Program programujte pomocou hardvérového prerušenia od tlačiarne (IRQ7) a v programovacom jazyku C pod operačným systémom MS-DOS.

Analýza zadania:

Aj v tomto zadaní, tak ako v prvom zadaní tejto kapitoly sa budú používať funkcie `inportb()` a `outportb()`, ktoré sú súčasťou knižnice *dos.h*, tak túto knižnicu je nutné prilinkovať k programu spolu so vstupno-výstupnou knižnicou *stdio.h*.

V prvom rade treba zadeklarovať všetky globálne premenné ktoré sa budú používať:

- pole znakov (*char[]*) do ktorých sa vloží zvolený reťazec určený na tlačenie,
- celé číslo (*int*), ktoré bude tento reťazec (pole) pri tlačení indexovať, je dôležité, aby jeho inicializačná hodnota bola 0,
- celé číslo (*int*) do ktorého sa bude vkladať priebežne kód stavu tlačiarne,
- pomocnú premennú (napr.: *int*), ktorá bude kontrolovať ukončenie tlače,
- stará obsluha prerušenia IRQ7 (*interrupt*),
- nová obsluha prerušenia IRQ7 (*interrupt*).

V ďalšom kroku sa vytvorí program novej obsluhy prerušenia od paralelného rozhrania (IRQ7). V tejto obsluhe sa najskôr skontroluje či ide o posledný znak určený k tlačeniu. Ak sa nejedná o posledný znak, tak sa pomocou dátového registra 378H pošle znak určený na tlačenie (znak v poli s daným indexom), potom sa na dĺžku jedného programového cyklu vyšle bit STROBE a inkrementuje sa index poľa (reťazca). Ak sa jedná o posledný znak, tak sa pomocou dátového registra 378H pošle znak *nového riadku*, následne sa na dĺžku jedného programového cyklu vyšle bit STROBE a do pomocnej premennej sa zapíše príznak ukončenia tlače.

V rámci tejto obsluhy sa ešte skontroluje, či sa tlačiareň nedostala do chyby, ak áno, tieto chyby (alebo chyba) sa vypíšu na obrazovku a po zatlačení ľubovoľného klávesa sa do pomocnej premennej zapíše príznak ukončenia tlače.

Hlavný program najskôr vyčistí obrazovku, pre lepší vzhľad aplikácie, vyzve užívateľa k zadaniu reťazca. Tento reťazec sa pomocou funkcie `fgets()` vloží do zadeklarovaného poľa. Do pripravenej premennej sa vloží pomocou vektora obsluhy prerušenia stará obsluha prerušenia od paralelného portu (od tlačiarne), ktorá je na adrese 0FH. Následne sa pomocou tej istej adresy 0FH nahrá nová obsluha prerušenia, ktorá bola navrhnutá a analyzovaná vyššie. Potom sa tlačiareň zinicializuje a už sa len čaká na ukončenie tlače, nakoľko všetko obstaráva nová obsluha prerušenia. Posledným krokom hlavného programu je nahrať starú obsluhu prerušenia späť do operačnej pamäte pomocou vektora obsluhy prerušenia.

Riešenie:

```
1    #include<stdio.h>
2    #include<dos.h>
3    void interrupt(* stare_pr)(void);
4    void interrupt(nove_pr)(void);
5    int i = 0, koniec = 0, stav = 0;
6    char text[70];
7
8    void interrupt nove_pr (void)
9    {
10        outportb(0x21,(inportb(0x21)|0x80));
11        if (i==70)
12        {
13             outportb(0x378,'\n');
14             outportb(0x37A,5);
15             outportb(0x37A,4);
16             koniec = 1;
17        }
18        else
19        {
20             outportb(0x378,text[i]);
21             outportb(0x37A,5);
22             outportb(0x37A,4);
23             i++;
24        }
25        stav = inportb(0x379);
26        if (!(stav&0x08)) printf("\n Tlaciaren hlasy chybu");
27        if (!(stav&0x10)) printf("\n Tlac. nie je selekt.");
28        if ((stav&0x20))  printf("\n Dosiel papier");
29        if ((!(stav&0x08))||(!(stav&0x10))||((stav&0x20)))
30        {
31             getchar();
32             koniec = 1;
33        }
34        outportb(0x21,(inportb(0x21)&~0x80));
35        outportb(0x20,0x20);
36   }
37   void main()
38   {
39        clrscr();
40        printf("Zadajte text na tlacenie:\n");
41        fgets(text,sizeof(text),stdin);
42        stare_pr=getvect(0x0f);
43        setvect(0x0f,nove_pr);
44        outportb(0x21,(inportb(0x21)&~0x80));
45        outportb(0x37A,24);
46        while((text[i] != '\0')&&(koniec != 1));
47        i = 70;
48        while(koniec != 1);
49        setvect(0x0f,stare_pr);
50   }
```

Vysvetlenie:

1-2: Prilinkovanie dôležitých knižníc k programu.

3: Deklarácia premennej `stare_pr` typu *interrupt* určenej pre starú obsluhu prerušenia paralelného rozhrania (IRQ7).

4: Deklarácia premennej `nove_pr` typu *interrupt* určenej pre novú obsluhu prerušenia paralelného rozhrania (IRQ7).

5: Deklarácia premenných typu *int*: index tlačeného reťazca, príznak ukončenia tlače, kód stavu tlačiarne.

6: Deklarácia reťazca (poľa znakov), ktorý sa bude tlačiť.

8: Definovanie novej obsluhy prerušenia rozhrania Centronics: `nove_pr()`.

10: Zakázanie prerušenia od rozhrania Centronics (OCW 1, viď Obr. 2.13), aby počas behu obsluhy nemohlo nastať prerušenie znovu.

11-17: Ak sa tlačí posledný znak (*nový riadok*):

- 13: Vloženie znaku *nový riadok* do dátového registra (378H) rozhrania Centronics.
- 14: Aktivácia bitu STROBE pomocou riadiaceho registra (37AH).
- 15: Deaktivácia bitu STROBE pomocou riadiaceho registra (37AH).
- 16: Zápis príznaku ukončenia tlače.

18-24: Ak sa netlačí posledný znak:

- 20: Vloženie znaku v reťazci `text` s indexom `i` do dátového registra (378H) rozhrania Centronics.
- 21: Aktivácia bitu STROBE pomocou riadiaceho registra (37AH).
- 22: Deaktivácia bitu STROBE pomocou riadiaceho registra (37AH).
- 23: Inkrementácia indexu `i`.

25: Vloženie kódu stavu tlačiarne do premennej `stav`.

26: Ak 3. bit vstupného argumentu funkcie `chyba()` je nulový vypíše sa hlásenie chyby tlačiarne.

27: Ak 4. bit vstupného argumentu funkcie `chyba()` je nulový vypíše sa hlásenie, že tlačiareň nie je selektovaná.

28: Ak 5. bit vstupného argumentu funkcie `chyba()` je jedna vypíše sa hlásenie o chýbajúcom papieri.

29-33: Ak 3. bit je nulový, alebo 4. bit je nulový, alebo 5. bit je jedna vstupného argumentu funkcie `chyba()`, tak:

- 31: Čakanie na stlačenie ľubovoľného klávesa.
- 32: Zápis príznaku ukončenia tlače.

34: Povolenie prerušenia od paralelného rozhrania (OCW 1, viď Obr. 2.13).

35: Ukončenie práve obsluhovaného prerušenia (OCW2, viď Obr. 2.14).

37: Deklarácia a definovanie hlavného programu.

39: Vyčistenie obrazovky.

40: Vypísanie textu: „Zadajte text na tlačenie“, na obrazovku.

41: Vloženie vypísaného textu do reťazca `text`.

42: Uloženie starej obsluhy prerušenia rozhrania Centronics do premennej `stare_pr`.

43: Vloženie novej obsluhy prerušenia rozhrania Centronics z premennej `nove_pr`.

44: Povolenie prerušenia od rozhrania Centronics (OCW 1, viď Obr. 2.13).

45: Inicializácia tlačiarne pomocou riadiaceho registra (37AH, viď Obr. 7.28).

46: Cyklus ktorý sa opakuje, kým nie je nastavený príznak ukončenia tlače, alebo kým sa index reťazca nedostal k poslednému znaku.

47: Vloženie hodnoty 70 do indexu reťazca (index posledného znaku).

48: Cyklus ktorý sa opakuje, kým nie je nastavený príznak ukončenia tlače.

49: Vloženie starej obsluhy prerušenia rozhrania Centronics z premennej `stare_pr`.

Zadanie 4:

Naprogramujte program, ktorý pomocou rozhrania Centronics (LPT) a ihličkovej tlačiarne vytlačí text:
„Dobrý deň,

Srdečne Vás zdravím.“.

K naprogramovaniu použite obslužný program prerušenia od tlačiarne (paralelného portu) INT 17.

Analýza zadania:

Keďže v tomto programe je potrebné využívať funkciu `int86()`, tak je nutné prilinkovať k programu okrem knižnice *stdio.h* aj knižnicu *dos.h*. Pred programovaním funkcií a hlavného programu je potrebné zadeklarovať globálnu premennú typu *union REGS* pre prácu s funkciou `int86()`.

Program pre lepšiu prehľadnosť využije niekoľko funkcií:

- vyslanie znaku pre tlač,
- zistenie stavu tlačiarne,
- inicializácia tlačiarne,
- tlač reťazca.

Funkcia na vyslanie znaku pre tlač v podstate len nastaví parametre služby 00H (tlačenie znaku) v obsluhe prerušenia paralelného portu (INT 17H). Konkrétne nastaví znak určený pre tlač (*al*), číslo tlačiarne (*dx*) a poradie služby (*ah*), potom sa tieto parametre pošlú na spracovanie pomocou funkcie `int86()`.

Zistenie stavu tlačiarne bude tiež len jednoduchá funkcia, ktorá nastaví parametre služby 02H v obsluhe prerušenia INT 17H, ale táto funkcia bude mať aj návratovú hodnotu. V tomto prípade sa nastavia iba dva parametre a to číslo tlačiarne (*dx*) a poradie služby (*ah*). Parametre sa vložia do funkcie `int86()` a výstup bude v parametri *ah*, kódovanie (poradie bitov) je znázornené na obrázku Obr. 7.30. Tento výstup sa nastaví ako návratová hodnota funkcie.

Inicializáciu tlačiarne bude zabezpečovať funkcia, ktorá nastaví dva parametre číslo tlačiarne (*dx*) a poradie služby (*ah*). Následne už iba pomocou funkcie `int86()` sa táto služba spustí a zinicializuje tlačiareň.

Funkcia určená pre tlač reťazca už bude o čosi zložitejšia. Vstupným argumentom funkcie bude reťazec (pole znakov). Pomocou cyklu s podmienkou na začiatku sa prejde celým reťazcom. V cykle sa najprv zistí stav tlačiarne pomocou funkcie, ktorá bude nato určená. Ak došiel papier vypíše sa o tom hlásenie a program bude čakať na vloženie papiera. Keď sa počas čakania na papier stlačí ľubovoľný kláves, tak sa aplikácia vypne. V prípade, že sa vloží papier alebo papier vôbec nedošiel, tak sa budú vysielať jednotlivé znaky pre tlač každých 50ms v spomínanom cykle pomocou funkcie na vyslanie znaku pre tlač.

Hlavný program najskôr vyčistí obrazovku, zinicializuje tlačiareň, zistí stav tlačiarne. Ak je tlačiareň v chybe, tak sa daná chyba vypíše a to pomocou usporiadania bitov (kódu) v premennej stavu tlačiarne (Obr. 7.30). V prípade, že tlačiareň v chybe nie je spustí sa funkcia určená pre tlač reťazca.

Riešenie:

```
1    #include <dos.h>
2    #include <stdio.h>
3    union REGS r;
```

```
4
5    void vysli(int c)
6    {
7         r.h.al=c;
8         r.h.ah=0;
9         r.x.dx=0;
10        int86(0x17,&r,&r);
11   }
12
13   int stav_tlac(void)
14   {
15        r.h.ah=2;
16        r.x.dx=0;
17        int86(0x17,&r,&r);
18        return(r.h.ah);
19   }
20
21   void init(void)
22   {
23        r.h.ah=1;
24        r.x.dx=0;
25        int86(0x17,&r,&r);
26   }
27
28   void tlac_ret(char *str)
29   {
30        int stav;
31        stav = stav_tlac();
32        while(*str != 0)
33        if((stav = stav_tlac())&0x20)
34        {
35             printf("Dosiel papier, cakam nan !\n");
36             while(stav_tlac()&0x20) if(kbhit())
37             {
38                  getch();
39                  exit(1);
40             }
41        }
42        else
43        {
44             delay(50);
45             vysli(*str++);
46        }
47   }
48
49   void main(void)
50   {
51        int stav;
52        clrscr();
53        init();
54        stav = stav_tlac();
```

```
55        if(stav&0x40)
56        {
57             printf("\nTlaciaren nepripojena !\n");
58             exit(1);
59        }
60        if(stav&0x01)
61        {
62             printf("\nTime out");
63             exit(1);
64        }
65        if(stav&0x04)
66        {
67             printf("\nChyba vstupno-vystupneho zariadenia");
68             exit(1);
69        }
70        if(stav&0x20)
71        {
72             printf("\nTlaciaren nepripravena");
73             exit(1);
74        }
75        tlac_ret("Dobrý deň,\n\n\nSrdečne Vás zdravím.\n");
76   }
```

Vysvetlenie:

1-2: Prilinkovanie dôležitých knižníc k programu.
3: Deklarácia parametrov pre funkciu `int86()`.
5: Deklarácia a definovanie funkcie `vysli()`.
7: Vloženie znaku (vstupného parametra funkcie `vysli()`) do parametra obsluhy prerušenia.
8: Určenie nultého parametra (znak určený k tlačeniu), ako vstupného parametra obsluhy prerušenia.
9: Vybratie nultej tlačiarne (LPT1).
10: Zaslanie parametrov obsluhe prerušenia tlačiarne a spustenie tejto obsluhy s danými parametrami.
13: Deklarácia a definovanie funkcie `stav_tlac()`.
15: Určenie druhého parametra (žiadosť o návrat stavu tlačiarne), ako vstupného parametra obsluhy prerušenia.
16: Vybratie nultej tlačiarne (LPT1).
17: Zaslanie parametrov obsluhe prerušenia tlačiarne a spustenie tejto obsluhy s danými parametrami.
18: Vrátenie výstupnej premennej obsluhy prerušenia (vrátenie stavu tlačiarne).
21: Definovanie funkcie `init()`.
23: Určenie prvého parametra (žiadosť o inicializáciu tlačiarne), ako vstupného parametra obsluhy prerušenia.
24: Vybratie nultej tlačiarne (LPT1).
25: Zaslanie parametrov obsluhe prerušenia tlačiarne a spustenie tejto obsluhy s danými parametrami (inicializácia tlačiarne).
28: Deklarácia a definovanie funkcie `tlac_ret()`.
30: Deklarácia premennej `stav` (stav tlačiarne).
31: Do premennej `stav` sa vloží návratová hodnota funkcie `stav_tlac()`.

32-46: Cyklus ktorý prejde celým vstupným argumentom po jednom znaku.
- 33: Podmienka ktorá kontroluje stav papiera v tlačiarni (či v tlačiarni nedošiel papier).
- 35: Ak v tlačiarni došiel papier, tak sa na obrazovke zobrazí hlásenie o tej skutočnosti.
- 36-40: Počas vkladania papiera čaká program na stlačenie ktoréhokoľvek klávesa, ak sa kláves stlačí, tak sa program vypne, inač po vložení papiera bude hlavný cyklus pokračovať v svojej činnosti.
- 42-46: Ak je v tlačiarni papier, tak každých 50 ms sa spustí funkcia `vysli()` s argumentom jedného znaku, ktorý je práve predmetom hlavného cyklu.

49: Deklarácia a definovanie hlavného programu.
51: Deklarácia premennej `stav` (stav tlačiarne).
52: Vyčistenie obrazovky.
53: Spustenie funkcie `init()`.
54: Do premennej `stav` sa vloží návratová hodnota funkcie `stav_tlac()`.
55-59: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Tlačiareň nepripojená!*“ a následné vypnutie programu.
60-64: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Time out*“ a následné vypnutie programu.
65-69: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Chyba vstupno-výstupného zariadenia*“ a následné vypnutie programu.
70-74: Hlásenie chyby (podľa premennej `stav`, ak k chybe došlo): „*Tlačiareň nepripravená*“ a následné vypnutie programu.
75: Spustenie funkcie `tlac_ret()` s argumentom „Dobrý deň,\n\n\nSrdečne Vás zdravím.\n“.

Zadanie 5:

Naprogramujte program, ktorý na laboratórnom prípravku (Obr. 7.29 v pravo) bude rozsvecovať LED diódy pripojené k riadiacim a dátovým vodičom. Tieto diódy sa budú rozsvecovať po jednom každých 500ms. Na laboratórnom prípravku sú všetky riadiace a dátové vodiče rozhrania Centronics pripojené k LED diódam a sú uzemnené signálovou zemou (Tab. 7.6). Program naprogramujte pomocou programovacieho jazyka C v operačnom systéme MS-DOS.

Analýza zadania:

Aj v tomto zadaní je potrebné prilinkovať knižnice *stdio.h* a *dos.h* nakoľko budeme využívať ich funkcie.

Najprv sa budú rozsvecovať LED diódy pripojené k riadiacim vodičom. Nakoľko bity Strobe (0. bit), Autofeed (1. bit) a Select printer (3. bit) sú aktívne (H) v prípade, že nadobúdajú hodnotu 0 (*false*) a Reset (2. bit) je aktívny keď nadobúda hodnotu 1 (*true*), tak ak nemá svietiť žiadna dióda musí sa do riadiaceho slova zapísať hodnotu 11 (1x1 + 1x2 + 0x4 + 1x8). Ak má svietiť prvá dióda treba zapísať do registra hodnotu 10 (0x1 + 1x2 + 0x4 + 1x8). Ak má svietiť druhá dióda zapíše sa hodnota 9 (1x1 + 0x2 + 0x4 + 1x8). Ak má svietiť tretia dióda zapíše sa hodnota 15 (1x1 + 1x2 + 1x4 + 1x8). A pri poslednej treba nastaviť hodnotu 3 (1x1 + 1x2 + 0x4 + 0x8). Tieto hodnoty sa v cykle jednotlivo zapíšu do riadiaceho registra napríklad pomocou poľa, ktoré bude reprezentovať akúsi mapu týchto hodnôt.

V druhej časti programu sa budú rozsvecovať LED diódy pripojené k dátovým vodičom. V tomto prípade je to jednoduchšie ako pri riadiacom registri, nakoľko aktívna úroveň je H pri každom bite. Takže do dátového registra treba postupne zapísať hodnoty

1 (0000 0001b), 2 (0000 0010b), 4 (0000 0100b), 8 (0000 1000b), 16 (0001 0000b), 32 (0010 0000b), 64 (0100 0000b), 128 (1000 0000b) a hodnotu 256 (1 0000 0000b) na vyprázdnenie registra (mohla sa zapísať aj hodnota 0, ale týmto spôsobom si zjednodušíme cyklus). Tieto hodnoty zapíšeme do dátového registra pomocou jednoduchého cyklu s pevným počtom opakovaní, len v inkrementačnej časti príkazu pre cyklus nastavíme vzore `i=i*2` a inicializačná hodnota cyklu bude 1 (`i=1`) a do dátového registra sa bude zapisovať hodnota indexu `i`.

Riešenie:

```
1    #include<stdio.h>
2    #include<dos.h>
3
4    void main()
5    {
6         int i;
7         int j[6] = {11,10,9,15,3,11};
8         for(i=0;i<6;i++)
9         {
10             outportb(0x37A,j[i]);
11             delay(500);
12        }
13        for(i=1;i<257;i=i*2)
14        {
15             outportb(0x378,i);
16             delay(500);
17        }
18   }
```

Vysvetlenie:

1-2: Prilinkovanie dôležitých knižníc k programu.
4: Deklarácia a definovanie hlavného programu.
6: Deklarácia premennej `i` typu *int*, ktorá bude indexom cyklov.
7: Deklarácia poľa `j` typu *int[]*, ktoré mapuje LED diódy pripojené k riadiacemu registru rozhrania Centronics.
8-12: Cyklus s pevným počtom opakovaní (6), ktorý zasvieti jednotlivé LED diódy pripojené k riadiacim vodičom rozhrania Centronics na dobu 500ms.
13-17: Cyklus s pevným počtom opakovaní (8), ktorý zasvieti jednotlivé LED diódy pripojené k dátovým vodičom rozhrania Centronics na dobu 500ms.

Zadanie 6:

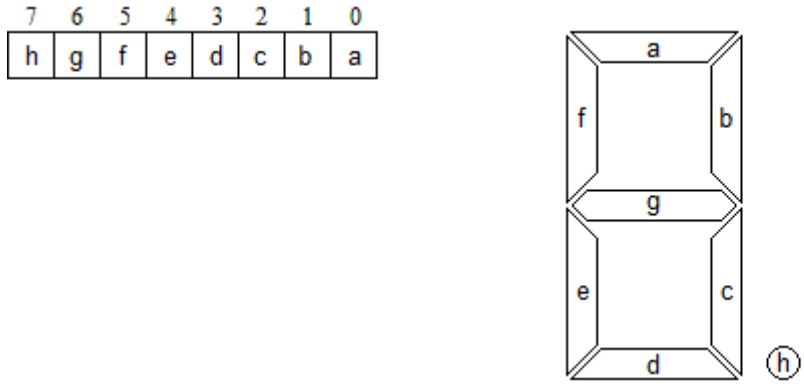
Naprogramujte program, ktorý na laboratórnom prípravku (Obr. 7.29 v ľavo) bude vypisovať pomocou displeja (sústava štyroch 7 segmentových displejov s bodkou) poradové číslo tlačidla ktoré je stlačené (ak nie stlačené žiadne vypíše sa 0) spolu so sprievodným textom TL (napr.: TL0). Laboratórny prípravok má dátové vodiče pripojené ku katódam displeja a riadiacimi vodičmi sa adresuje konkrétny 7 segmentový displej. Stavové vodiče sú pripojené k tlačidlám, ktoré sú uzemnené signálovou zemou. Program naprogramujte programovacím jazykom C# v prostredí Visual Studio 2005 (alebo vo vyššej verzií prostredia) pod operačným systémom Windows XP (alebo pod vyššou verziou operačného systému).

Analýza zadania:

Návrh vzhľadu aplikácie v tomto zadaní nie je dôležitý, nakoľko je dôležitejší výpis na displeji prípravku. V tomto prípade vložíme do aplikácie len dva objekty FrameWork-u .NET: časovač (*timer*) a text (*label*). Časovač sa vo vlastnostiach (*properties*) zapne a nastaví sa mu najnižšia perióda (1 ms).

Do samotnej aplikácie je nutné prilinkovať funkcie `Out32()` a `Inp32()` z dynamickej knižnice inpout32.dll, postup a samotný kód pre vloženie funkcií z dynamickej knižnice je v kapitole 2.2.7, nakoľko sú to funkcie pre adresovanie periférnych zariadení. Nový názov týchto funkcií sa môže zvoliť napríklad `Out()` a `In()`.

Keďže v zadaní sa žiada, aby na displeji bol aj sprievodný text bude nutné naprogramovať *multiplexné riadenie displeja*. Tomuto riadeniu sa bude venovať osobitná funkcia. Úlohou multiplexného riadenia displeja je preblikávanie jednotlivých sedem segmentových displejov a to tak, aby zobrazovaný text sa javil pre ľudské oko ako celistvý, čiže aby preblikávanie človek nevnímal. Takže periódu preblikávania treba určiť čo najnižšiu, ale aby sa dané údaje stihli zapísať do registrov a vysvietiť na displeji. Najvhodnejšia perióda bude približne 100 000 programových cyklov (experimentálne overená hodnota). Adresovanie displeja má na starosti riadiaci register paralelného portu. Takže pre adresovanie displeja bude platiť to isté ako pre adresovanie (rozsvecovanie) LED diód (viď analýzu predošlého zadania), takže adresa 1. segmentovky je 10, 2.segmentovky je 9, 3. segmentovky je 15 a 4. segmetovky je 3. Čo sa týka dátového registra je potrebné vedieť rozloženie LED diód (zapojenie katód) na displeji Obr. 7.32:



Obr. 7.32 Rozmiestnenie a zapojenie LED diód na 7-segmetovom displeji s bodkou k riadiacim vodičom

Takže keď je potrebné vysvietiť číslo jedna tak do dátového registra zapíšeme 6 (00000110b – bity b a c majú byť aktívne), takýmto spôsobom sa vypočítajú (prevedú z dvojkovej do desiatkovej sústavy) ostatné cifry a písmená: cifra 0: 63, cifra 1: 6, cifra 2: 91, cifra 3: 79, cifra 4: 102, cifra 5: 109, písmeno T: 7 a 49 – skladá sa z dvoch pozícií displeja, písmeno L: 56.

V programe pre časovač sa bude zisťovať, ktoré tlačidlo bolo stlačené pomocou stavového registra. Ak nie je žiadne tlačidlo stlačené, tak v stavovom registry je hodnota 127 (0111 1111b), pretože iba bit *Busy* nemá inverznú logiku. Takže ak aktivujeme bit *Error* (0111 0111b), tak v stavovom registry bude hodnota 119 (1. tlačidlo). Keď bude aktívny bit *Select* (0110 1111b), vtedy bude v stavovom registri hodnota 111 (2. tlačidlo). Pri bite *Paper end* (0101 1111b) bude v registri hodnota 95 (3. tlačidlo). Bit *Acknowledge* (0011 1111b) nastaví hodnotu registra na 63 (5. tlačidlo). Posledný bit *Busy* (1111 1111b) upraví hodnotu registra na 255 (4. tlačidlo). Jednoduchými podmienkami (*if*) sa zistí, ktoré tlačidlo bolo stlačené a podmienka spustí funkciu multiplexného riadenia displeja s argumentom tlačidla.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
5    namespace LPTpanel
6    {
7        public partial class Form1 : Form
8        {
9            [DllImport("inpout32.dll", EntryPoint="Out32")]
             public static extern void Out(int add, int val);
10           [DllImport("inpout32.dll", EntryPoint="Inp32")]
             public static extern int In(int add);
11           int[] cifra = new int[6] {63,6,91,79,102,109};
12           int i,j;
13
14           public Form1()
15           {
16               InitializeComponent();
17           }
18
19           void zobraz(int tlacidlo)
20           {
21               for (j = 0; j < 100; j++)
22               {
23                   Out(0x378, 0);
24                   Out(0x37A, 10);
25                   Out(0x378, 7);
26                   for (i = 0; i < 100000; i++);
27                   Out(0x378, 0);
28                   Out(0x37A, 9);
29                   Out(0x378, 49);
30                   for (i = 0; i < 100000; i++);
31                   Out(0x378, 0);
32                   Out(0x37A, 15);
33                   Out(0x378, 56);
34                   for (i = 0; i < 100000; i++);
35                   Out(0x378, 0);
36                   Out(0x37A, 3);
37                   Out(0x378, cifra[tlacidlo]);
38                   for (i = 0; i < 100000; i++);
39               }
40           }
41
42           private void timer1_Tick(object sender,
             EventArgs e)
43           {
44               if ((In(0x379)) == 127)
45               {
46                   zobraz(0);
47                   label1.Text = "Tlačidlo 0";
```

```
48                  }
49                  if ((In(0x379)) == 119)
50                  {
51                       zobraz(1);
52                       label1.Text = "Tlačidlo 1";
53                  }
54                  if ((In(0x379)) == 111)
55                  {
56                       zobraz(2);
57                       label1.Text = "Tlačidlo 2";
58                  }
59                  if ((In(0x379)) == 95)
60                  {
61                       zobraz(3);
62                       label1.Text = "Tlačidlo 3";
63                  }
64                  if ((In(0x379)) == 255)
65                  {
66                       zobraz(4);
67                       label1.Text = "Tlačidlo 4";
68                  }
69                  if ((In(0x379)) == 63)
70                  {
71                       zobraz(5);
72                       label1.Text = "Tlačidlo 5";
73                  }
74             }
75        }
76   }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

5: Deklarácia a definovanie namespace-u tohto programu.

7: Deklarácia a definovanie hlavnej triedy programu.

9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.

10: Vloženie funkcie `Inp32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `In()`.

11: Deklarácia poľa celých čísel (typu *int*) s názvom `cifra` reprezentujúceho mapu kódov prvých šiestich cifier (0, 1, 2, 3, 4, 5) v dátovom registri.

12: Deklarácia celých čísel (typu *int*) s názvom `i` a `j`, ktoré budú slúžiť ako indexy cyklov s pevným počtom opakovaní.

14: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

16: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).

19: Deklarácia a definovanie funkcie `zobraz()`, ktorá bude slúžiť na zobrazovanie textu na displeji laboratórneho prípravku.

21-36: Cyklus, ktorý zabezpečí, aby zobrazovanie trvalo dlhšie ako 1ms (takto je nastavený časovač (*timer*) a je to najnižšia možná perióda).
- 23: Vyprázdnenie dátového registra rozhrania Centronics.
- 24: Nastavenie 1. pozície displeja pomocou riadiaceho registra rozhrania Centronics, do ktorého sa zapíše hodnota 10.
- 25: Nastavenie prvej polovičky písmena T pomocou dátového registra rozhrania Centronics, do ktorého sa zapíše hodnota 7.
- 26: 100 000 prázdnych programových cyklov.
- 27: Vyprázdnenie dátového registra rozhrania Centronics.
- 28: Nastavenie 2. pozície displeja pomocou riadiaceho registra rozhrania Centronics, do ktorého sa zapíše hodnota 9.
- 29: Nastavenie druhej polovičky písmena T pomocou dátového registra rozhrania Centronics, do ktorého sa zapíše hodnota 49.
- 30: 100 000 prázdnych programových cyklov.
- 31: Vyprázdnenie dátového registra rozhrania Centronics.
- 32: Nastavenie 3. pozície displeja pomocou riadiaceho registra rozhrania Centronics, do ktorého sa zapíše hodnota 15.
- 33: Nastavenie písmena L pomocou dátového registra rozhrania Centronics, do ktorého sa zapíše hodnota 56.
- 34: 100 000 prázdnych programových cyklov.
- 35: Vyprázdnenie dátového registra rozhrania Centronics.
- 36: Nastavenie 4. pozície displeja pomocou riadiaceho registra rozhrania Centronics, do ktorého sa zapíše hodnota 3.
- 37: Nastavenie cifry, ktorá je vstupným argumentom tejto funkcie s kódovou mapou `cifra`, pomocou dátového registra rozhrania Centronics.
- 38: 100 000 prázdnych programových cyklov.

42: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná každú periódu časovača (*timer*-a).

44-48: Ak nie je stlačené žiadne tlačidlo na laboratórnom prípravku (stavový register rozhrania nadobúda hodnotu 127), tak sa spustí funkcia `zobraz()` s argumentom 0 a v aplikácií sa vypíše text: „Tlačidlo 0“.

49-53: Ak je stlačené prvé tlačidlo na laboratórnom prípravku (stavový register rozhrania nadobúda hodnotu 119), tak sa spustí funkcia `zobraz()` s argumentom 1 a v aplikácií sa vypíše text: „Tlačidlo 1“.

54-58: Ak je stlačené druhé tlačidlo na laboratórnom prípravku (stavový register rozhrania nadobúda hodnotu 111), tak sa spustí funkcia `zobraz()` s argumentom 2 a v aplikácií sa vypíše text: „Tlačidlo 2“.

59-63: Ak je stlačené tretie tlačidlo na laboratórnom prípravku (stavový register rozhrania nadobúda hodnotu 95), tak sa spustí funkcia `zobraz()` s argumentom 3 a v aplikácií sa vypíše text: „Tlačidlo 3“.

64-68: Ak je stlačené štvrté tlačidlo na laboratórnom prípravku (stavový register rozhrania nadobúda hodnotu 255), tak sa spustí funkcia `zobraz()` s argumentom 4 a v aplikácií sa vypíše text: „Tlačidlo 4“.

69-73: Ak je stlačené piate tlačidlo na laboratórnom prípravku (stavový register rozhrania nadobúda hodnotu 63), tak sa spustí funkcia `zobraz()` s argumentom 5 a v aplikácií sa vypíše text: „Tlačidlo 5“.

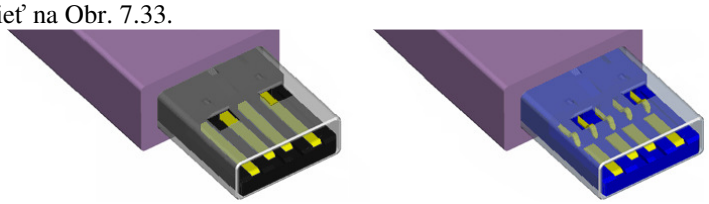
7.3 Štandardné rozhranie USB

USB vzniklo za spolupráce firiem Hewlett-Packard, Intel, Lucent, NEC, Microsoft a Philips. Jeho dizajn je štandardizovaný spoločnosťou USB Implementers Forum, ktorej členmi sú práve všetky menované spoločnosti. USB vzniklo ako alternatíva pomalých sériových a paralelných portov a prinieslo vyššiu prenosovú rýchlosť a kompatibilitu. Hlavným zámerom vyvinutia USB bolo práve zbaviť sa všetkých predtým používaných sériových a paralelných portov, pretože neboli dostatočne štandardizované a vyžadovali množstvo ovládačov.

Verzia 0.7 vyšla v novembri 1994. V januári 1996 vyšla USB verzia 1.0 s maximálnou prenosovou rýchlosťou 12 Mb/s. Revízia 1.1 vydaná v septembri 1998 napravila niektoré problémy verzie 1.0 spojené hlavne s konektormi. Dlho očakávaný nástupca USB 2.0 s novými výhodami prišiel v apríli 2000. USB Implementers Forum ho štandardizovalo na konci roku 2001. Oproti staršej verzii má badateľné zrýchlenie, pričom kompatibilita s USB 1.0 zostala zachovaná. To umožnilo USB preniknúť aj do oblastí, ktoré boli dovtedy z dôvodu nedostatočnej prenosovej rýchlosti pre neho uzavreté. Najvyššia rýchlosť je 480 Mb/s. Novšia verzia USB 3.0 bola predstavená v novembri 2008 a dosahuje prenosovú rýchlosť 5Gb/s. V júli 2013 predstavil USB Implementers Forum USB 3.1.

Verzia USB 1.1 a USB 2.0 využíva na prenos dát 4-vodičový kábel. Dva vodiče sú napájacie (5V a GND) a dokážu napájať zariadenia s odberom prúdu 0,5A (2,5W), ak sú tieto vodiče napájané priamo zo zdroja počítača, tak tento prúd môže byť aj vyšší. Ďalšie 2 vodiče slúžia na prenos dát a príkazov (D+ a D–). Vysoké napätie na D+ a nízke napätie na D– reprezentuje logický stav 1, vysoké napätie na D– a nízke napätie na D+ reprezentuje logický stav 0. Keďže signál je kódovaný iba dvoma vodičmi v jednom smere (half-duplex), tak pri USB 3.0 sa rozšíril počet týchto vodičov na 4 (TX+,TX–, RX+, RX–) a pripojil sa k nim aj GND pre dáta (full-duplex). Pri USB 2.0 malo napájanie externých zariadení veľký význam, takže USB 3.0 si túto výhodu ponechalo.

Aby bola zachovaná spätná kompatibilita, tak vodiče D+, D–, 5V a GND ponechali aj v tejto verzií, ako vyriešilo USB Implementers Forum kompatibilitu konektorov, ktoré je možné vidieť na Obr. 7.33.



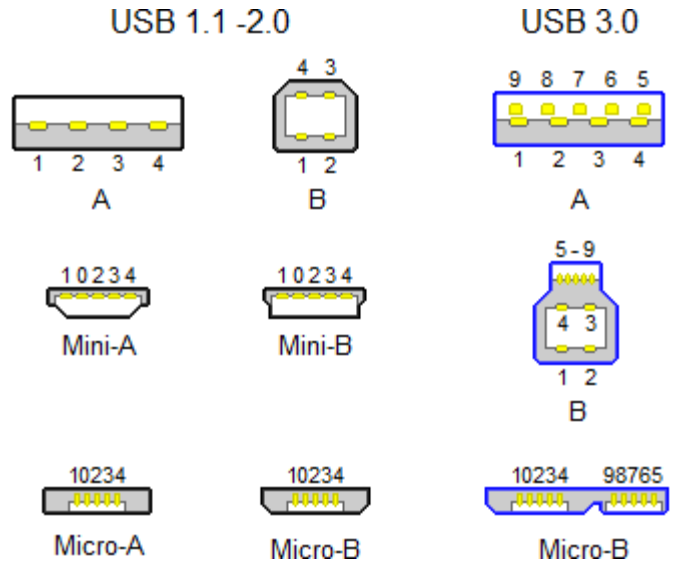
Obr. 7.33 USB 2.0 konektor vľavo a USB 3.0 konektor vpravo

Pri ponechaní napájacích vodičov zvýšil aj odber prúdu na 0,9 A (4,5W). USB 3.1 má odber prúdu zvýšený na 2A (10W) a možnosť napájania zariadení na 12V a 20V pri odbere prúdu 5A (60W a 100W). Základné parametre rozhrania USB sú v Tab. 7.8.

Tab. 7.8 Základné parametre rozhrania USB

Parameter	**Hodnota**
Maximálny počet uzlov	127 uzlov
Maximálna vzdialenosť medzi uzlami	5m (USB 2.0) 3m (USB 3.0 a 3.1)
Maximálna prenosová rýchlosť	10 Gb/s (USB 3.1)
Typ prenosového média	4-vodičový kábel (USB 2.0, USB 1.1) 9-vodičový kábel (USB 3.0)

Skratka USB znamená Universal Serial Bus (univerzálna sériová zbernica). Rozhranie USB má veľa typov konektorov, pretože najprv sa zaviedlo prepojenie medzi počítačom a periférnym zariadením (konektor A a B Obr. 7.34). Konektor pre periférne zariadenie (B, Obr. 7.34) bol veľmi veľký pre zariadenia ako videokamera, fotoaparát, atď. Preto sa zaviedli konektory Mini-A a Mini-B, ale neskôr najmä pre tenké inteligentné mobilné telefóny (*Smart phone*) boli aj tieto konektory veľmi hrubé, tak došlo k zúženiu na Micro-A a Micro-B konektor. Na obrázku Obr. 7.34 sa dá všimnúť, že USB 3.0 už konektory Mini-A, Mini-B a Micro-A ani nepoužíva (neexistujú), pretože rokmi používania sa zistilo, že z malých konektorov je najpopulárnejší práve Micro-B. Obrázok Obr. 7.34 znázorňuje vzhľady všetkých vstupných konektorov (samíc) aj s označením vodičov (pinov) okrem najnovšieho konektora typu C, pretože tam sa už spätná kompatibilita vytratila.



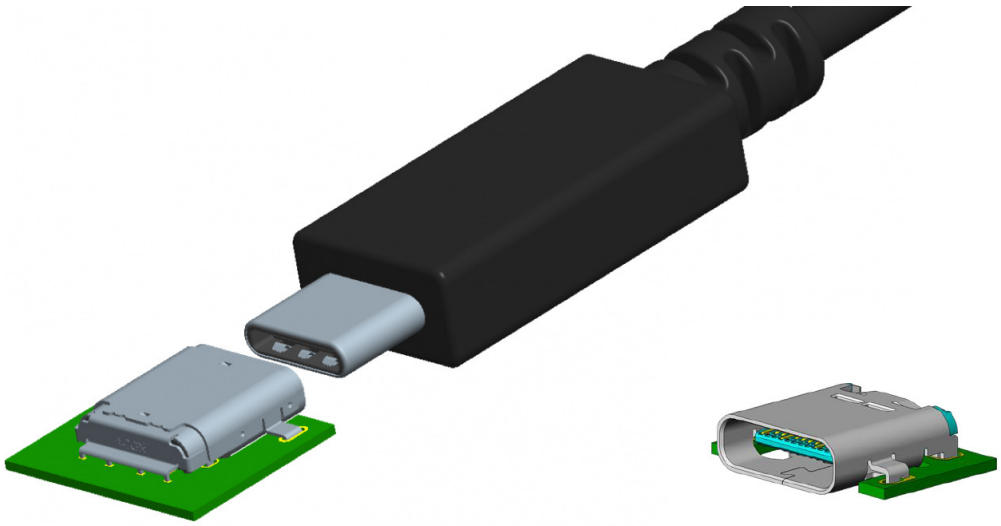
Obr. 7.34 Konektory USB s označením vodičov (pinov)

Popis pinov označených na obrázku Obr. 7.34 číslami je v tabuľke Tab. 7.9:

Tab. 7.9 Popis vodičov (kontaktov, pinov) rozhrania USB

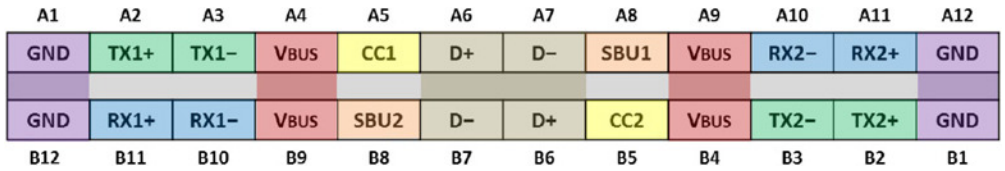
Kontakt (pin)	**Signál**	**Popis**
0	ID	Rozoznanie medzi *host* a *slave* zariadením: ak je tento vodič uzemnený jedná sa o *host* zariadenie, ak nie je uzemnený jedná sa o *slave* zariadenie.
1	GND	Uzemnenie napájania.
2	D+	Kladný dátový signál.
3	D-	Záporný dátový signál.
4	VCC	Napájanie +5V.
5	TX+	Kladný dátový signál pre odosielanie dát.
6	TX-	Záporný dátový signál pre odosielanie dát.
7	GND	Signálová zem.
8	RX+	Kladný dátový signál pre prijímanie dát.
9	RX-	Záporný dátový signál pre prijímanie dát.

USB konektor typu C je na obrázku Obr. 7.35:



Obr. 7.35 USB konektory typu C (samec a samica)

Tento konektor má 12 pinov vo vrchnej rade a ďalších 12 pinov v spodnej rade (Obr. 7.36):



Obr. 7.36 Rozmiestnenie pinov (kontaktov) USB konektora typu C

Popis pinov, ktorých rozmiestnenie je na obrázku Obr. 7.36, sa nachádza v tabuľke Tab. 7.10:

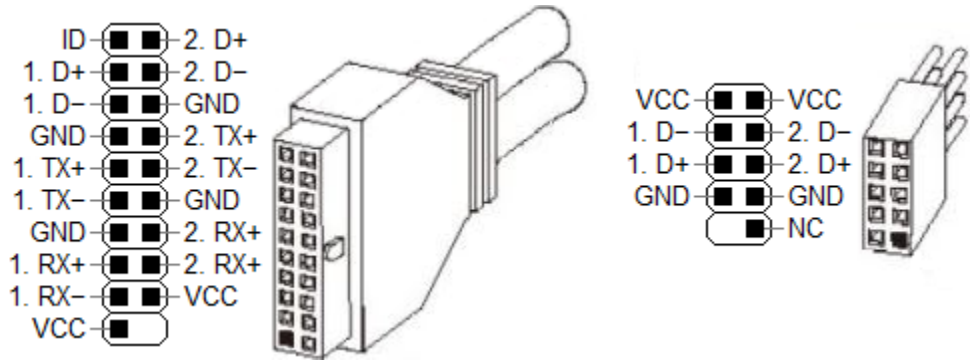
Tab. 7.10 Popis vodičov (kontaktov, pinov) rozhrania USB na konektore typu C

Kontakt (pin)	Signál	Popis
A1, A12, B1, B12	GND	Uzemnenie.
A2, A3, B10, B11	TX1+, TX1–, RX1+, RX1–	Dátové signály full-duplex (kompatibilné s USB3.0 a USB 3.1), prvá skupina.
A10, A11, B2, B3	TX2+, TX2–, RX2+, RX2–	Dátové signály full-duplex (kompatibilné s USB3.0 a USB 3.1), druhá skupina.
A4, A9, B4, B9	VBUS	Napájanie.
A8, B8	SBU1, SBU2	Sekundárna zbernica.
A5, B5	CC1, CC2	Detekcia konfigurácie
A6, A7, B6, B7	D+, D–, D+, D–	Signály kompatibilné s USB 2.0 (a nižšími verziami)

Rozhranie USB podporuje 5 dátových rýchlostí:

- *Low Speed (USB 1.1)*: rýchlosť 1.5 Mbit/s (187,5 kB/s, 128 kiB/s), ktoré sa najviac využíva na klávesnice, myši a joystick-y,
- *Full Speed (USB 1.1)*: rýchlosť 12 Mbit/s (1,5 MB/s, 1,43 kiB/s), ktorá bola najrýchlejšia pred uvedením USB 2.0,
- *High Speed (USB 2.0)*: rýchlosť 480 Mbit/s (60 MB/s, 57 MiB/s),
- *Super Speed (USB 3.0)*: rýchlosť 5 Gbit/s (625 MB/s, 596 MiB/s),
- *Super Speed USB 10Gbps (USB 3.1)*: rýchlosť 10 Gbit/s (1,25 GB/s, 1,16 GiB/s).

USB je množina špeciálnych čipov, ktoré fungujú ako rozhranie medzi softvérom a hardvérom. Aplikácie a rôzne ovládače, ktoré poskytujú informácie ako má zariadenie pracovať vysielajú dáta do rozbočovača (HUB), ktorý je na radič pripojený. Z rozbočovača vedie konektor na matičnej doske, ktorý je na obrázku Obr. 7.37. Okrem konektora na matičnej doske rozbočovač vedie klasické konektory (Obr. 7.34, konektor A) priamo na „zadnú“ časť matičnej dosky. Zadnou časťou matičnej dosky je myslené miesto, kde sú konektory všetkých rozhraní, čiže po pripojení dosky ku skrinke to bude zadná časť skrinky počítača. Konektory na matičnej (Obr. 7.37) doske sú určené pre predné USB konektory (Obr. 7.34, konektor A), čiže konektory, ktoré sú na prednej strane skrinky počítača, pretože všetky skrinky majú rôznu veľkosť a umiestnenie týchto konektorov. Na matičnej doske je zvyčajne (štandardne) jeden radič USB.



Obr. 7.37 USB konektor na matičnej doske (USB 3.0 s vyššie verzie v ľavo, USB 2.0 a nižšie verzie v pravo)

K radiču je možné pripojiť 127 periférnych zariadení, takže rozbočovač (HUB) je možné použiť aj za ďalšími rozbočovačmi. Ak sú rozbočovače pasívne (bez externého napájania), tak maximálna vzdialenosť medzi zariadením a počítačom je 5 m (3 m pri USB 3.0 a USB 3.1). Odporúčaný počet rozbočovačov medzi počítačom a zariadením je 5.

7.3.1 Kompatibilita všetkých verzií USB

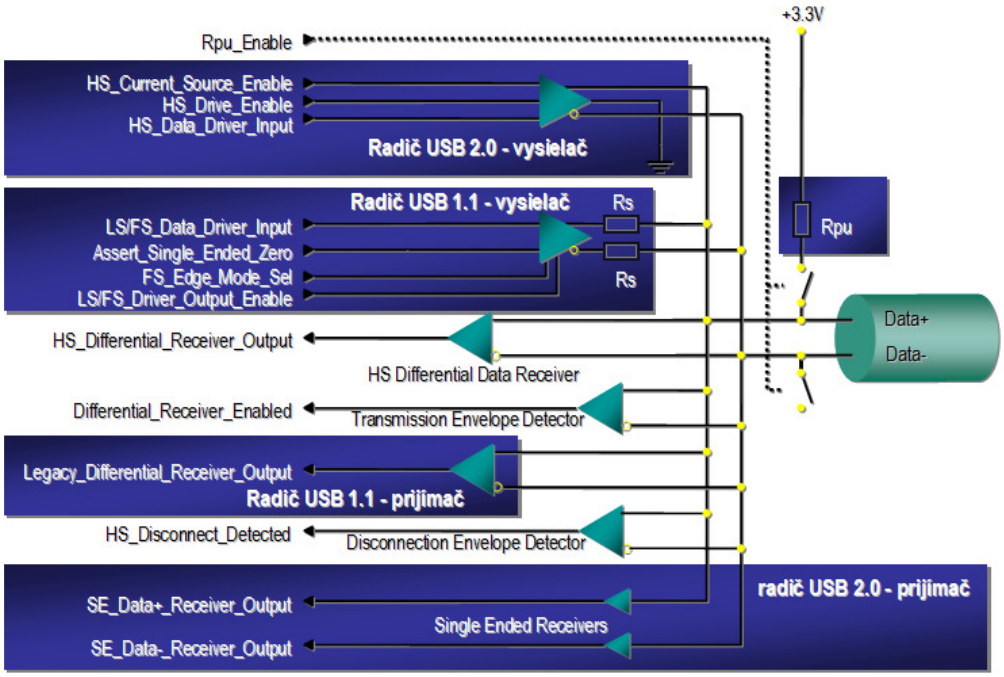
Na úrovni USB je podporovaný skoro každý typ periférií (zariadení). Rozhranie USB využíva technológiu *plug & play*. Ako náhle je pripojené nové zariadenie, tak to spôsobí zmenu napätia na oboch dátových vodičoch, vďaka tomu radič rozozná pripojenie nového zariadenia.

Rozlíšenie medzi *Low speed* a *Full speed* zariadením:

Ak po prihlásení zariadenia je na *D+* vyššie napätie, tak sa periférne zariadenie hlási s prenosovou rýchlosťou *Full speed*. Ak je po prihlásení periférie vyššie napätie na *D–*, tak ide o zariadenie, ktoré komunikuje prenosovou rýchlosťou *Low speed*.

Rozlíšenie medzi *Full speed* a *High speed* zariadením:

Toto rozlíšenie nemá na starosti radič, pretože by spomalil celú komunikáciu. Spomenuté prenosové rýchlosti rozlišuje rozbočovač (HUB), ktorý keď rozlíši zariadenie s prenosovou rýchlosťou *Full speed* alebo *Low speed*, tak pomocou služby *trancastion translator* (prekladač transakcie) prevedie signál na *High speed*. Takže radič USB 2.0 komunikuje prenosovou rýchlosťou *High speed* po celý čas. Ak by sa k radiču dostala komunikácia s prenosovou rýchlosťou *Full speed* alebo *Low speed*, tak by sa radič automaticky prepol z radiča USB 2.0 na radič USB 1.1, takže komunikácia by fungovala maximálnou prenosovou rýchlosťou 12 Mbit/s. Pre lepšiu predstavu fungovania radičov USB 2.0 a USB 1.1 na matičnej doske podporujúcej USB 2.0 je znázornený obrázok Obr. 7.38:



Obr. 7.38 Prepojenie radičov USB 1.1 a USB 2.0 v systéme podporujúcom USB 2.0

Rozlíšenie medzi *High speed* a *Super speed* zariadením:

Super speed (USB 3.0 a USB 3.1) zariadenie využíva pre komunikáciu nových 5 vodičov (viď Obr. 7.34) takže rozlíšenie spočíva iba v tom, ktoré vodiče periférne zariadenie využíva. Ak využíva signály (vodiče) *D+* a *D–*, tak riadenie preberá radič USB 2.0 (poprípade USB 1.1, tak ako je popísané vyššie). Keď zariadenie využíva signály *TX+*, *TX–*, *RX+* a *RX–*, tak sa riadenia ujíma radič USB 3.0.

7.3.2 Komunikácia po zbernici USB

Pokiaľ sa nové zariadenie stáva členom zbernice a zaujme svoje miesto, hlavný rozbočovač (HUB) zvolá zariadenia, aby im vydal príkazy. Zistí, či sú zariadenia pripravené posielať, alebo prijímať dáta a pridelí každému zariadeniu časť šírky pásma zbernice (kapacita pre prenos dát). Všetky správy začínajú poznávacím znamením (identifikátorom), ktorým sa identifikuje ktorej periférií je správa adresovaná, pretože každá správa ide po zbernici ku všetkým zariadeniam. Identifikátor má 7 bitov, takže dokáže zaadresovať 128 zariadení, pričom identifikátor 0 má master (host), takže v jednom čase sa môže pripojiť 127 periférnych zariadení. Zariadenia odosielajú údaje k rozbočovaču (HUB-u) len ak sú vyzvané od mastra (host-a).

Po zbernici USB je možné komunikovať jedným zo štyroch uvedených spôsobov:

- *riadiaci prenos* (control transfer) je využitý pre riadenie zariadenia, k tomu slúžia riadiace požiadavky (control requests), ktoré majú vysokú prioritu a obsahujú automatické sledovanie chýb. Jedna požiadavka môže obsahovať až 64 bajtov.
- *prenos pri prerušení* (interrupt transfer), systém tu periodicky posiela požiadavky na nové dáta, s názvom prerušenie to nemá nič spoločné, pretože pripojené zariadenie nemôže vyvolať prerušenie u hostiteľa (USB nemá k tomu príslušný vodič), typicky sa prenáša až osem bajtov, využíva sa pri klávesniciach alebo myšiach.
- *hromadný prenos* (bulk transfer) sa používa pre veľké množstvo dát, ktoré vyžaduje detekciu chýb, ale prenos nie je časovo kritický, nízka priorita, typická pre skener, tlačiareň, flash disky, atď.,
- *izochrónny prenos* (isochronous transfer) je tiež využívaný pre veľké množstvo dát, ale s definovanou prenosovou rýchlosťou bez korekcie chýb, uplatnenie si nájde všade tam, kde sa prenáša zvuk či video v reálnom čase, tu sú kladené vyššie nároky na časovú precíznosť než na vlastnú chybovosť prenosu, časové oneskorenie môže byť veľmi nepríjemným rušivým elementom.

Zbernica (rozhranie) USB dokáže pracovať štyrmi typmi prenosu údajov a prideľuje tri typy priority šírky pásma v nasledovnom poradí:

1. *najvyššia priorita* – izochrónna alebo používaná v reálnom čase, kde nemôže nastať prerušenie toku údajov týka sa to *riadiaceho prenosu* (control transfer) a *izochrónneho prenosu* (isochronous transfer), napr.: web-kamery, mikrofón, atď.,
2. *stredná priorita – prenos pri prerušení* (interrupt transfer) ku ktorému dochádza pri periodickom posielaní požiadavky na nové dáta a ešte k nemu dochádza ak dôjde k udalosti (k nepravému prerušeniu) pri ktorej sa dáta odovzdajú, napr.: klávesnica, myš, atď.,
3. *najnižšia priorita* – pri prenosoch veľkého objemu dát kde je potrebné previesť veľmi veľa údajov, avšak nie je nutné, aby sa prenášali kontinuálne jedná sa o *hromadný prenos* (bulk transfer), napr.: flash disky, tlačiarne, atď.

Z predošlých informácií musí byť zrejmé, že rozhranie (zbernica) USB využíva paketovú komunikáciu.

7.3.3 Programovanie rozhrania USB

Rozhranie USB ponúka veľa obvodov, ktoré spolu s driver-mi vedia nahradiť iné rozhrania. Asi najpoužívanejším obvodom je skupina FT-232 (FTDI), obvody tejto skupiny prepájajú USB a sériové rozhranie RS-232. Pričom driver pre USB vytvorí nový COM port, ktorý sa programuje totožne ako sériové rozhranie. V operačných systémoch Windows treba iba skontrolovať, pomocou nástroja Správca zariadení (Device manager), ktorý COM port bol zariadeniu USB pridelený. Z dôvodu rovnakého programovania sériového rozhrania a rozhrania USB pri použití spomenutého obvodu a driver-a zadania k tomuto typu komunikácie táto kapitola obsahovať nebude. Na obrázku Obr. 7.39 je jedno z takýchto zariadení.

Obr. 7.39 Konvertor signálu z USB na RS-232

Existuje aj mnoho ďalších podobných obvodov a driver-ov, ktoré prekonvertujú USB rozhranie na rozhrania ako Centronics, zbernicu ISA, atď.

Pre priame programovanie rozhrania USB sa využije dynamická knižnica *LibUsbDotNet.dll* táto knižnica obsahuje všetky dôležité metódy (funkcie) a triedy (deklarácie) pre komunikáciu s rozhraním USB. V zadaní sa využijú štyri triedy ako `UsbDevice`, `UsbDeviceFinder`, `UsbSetupPacket`, `UsbRequestType` a dve metódy `OpenUsbDevice()`, `ControlTransfer()` dynamickej knižnice. Trieda `UsbDevice` deklaruje samotné zariadenie. Trieda `UsbDeviceFinder` deklaruje adresu zariadenia pomocou sériového čísla zariadenia, GUID (globálny unikátny identifikátor – Global unique ID), VID (Vendor ID), PID (Product ID), alebo revízie (revízia nemôže byť samostatne iba v kombinácií s iným identifikátorom). Trieda `UsbSetupPacket` deklaruje samotný paket a trieda `UsbRequestType` deklaruje typ požiadavky (prenosu). Pomocou metódy `OpenUsbDevice()` otvoríme komunikáciu so zariadením a pomocou metódy `ControlTransfer()` pošleme paket.

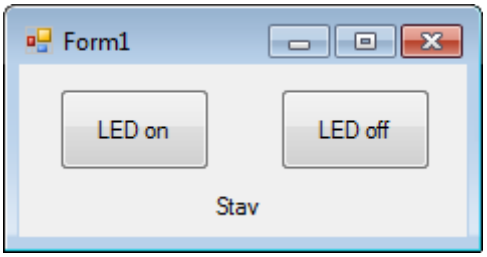
Zadanie 1:

Naprogramujte program, pomocou ktorého sa bude dať rozsvecovať a zhasínať LED diódu na robotickej ruke OWI (Obr. 7.40), ktorá sa nachádza medzi čeľusťami spomínanej ruky. Program naprogramujte pod operačným systémom Windows XP (alebo vyššou radou operačného systému Windows) pomocou programovacieho jazyka C# vo vývojovom prostredí Visual Studio 2005 (alebo vyššou radou vývojového prostredia Visual Studio).

Obr. 7.40 Robotická ruka OWI

Analýza zadania:

Najskôr je potrebné navrhnúť vzhľad aplikácie, keďže bude nutné rozsvecovať a zhasínať LED diódu najľahším riešení sú dve tlačidlá (*button*) a jedno textové pole (*label*), kde sa vypíše chyba pri komunikácií, ak náhodou nastane. Návrh vzhľadu aplikácie (Obr. 7.41):



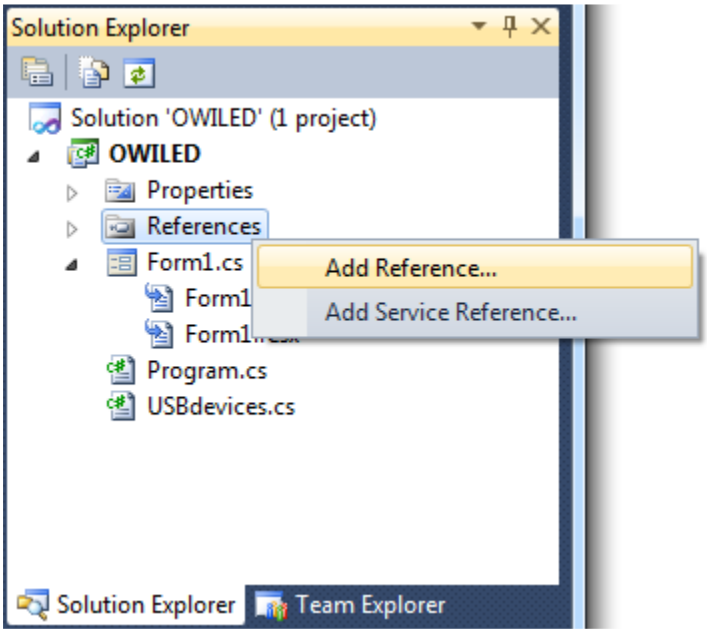
Obr. 7.41 Návrh vzhľadu aplikácie pre prácu s LED diódou

Pred začatím programovania je nutné pridať do referencií (*References*) dynamickú knižnicu *LibUsbDotNet.dll* (Obr. 7.42). Nakoľko sa nebudú využívať len zopár funkcií tejto knižnice, tak je nevhodné použiť `DllImport()` na importovanie jednotlivých funkcií.

K prilinkovaným systemovým triedam treba prilinkovať ešte ďalšie systémové triedy a samozrejme treba prilinkovať aj triedy z dynamickej knižnice, keďže sa bude používať:

```
using System.Runtime.InteropServices;
using LibUsbDotNet;
using LibUsbDotNet.Main;
```

Ďalším dôležitým krokom je zadeklarovať globálne premenné, ktorými sú objekt reprezentujúci samotné zariadenie, premenné určené k identifikácií zariadenia a dáta, ktoré sa budú vkladať do paketu.



Obr. 7.42 Pridanie referencií vo vývojovom prostredí Visual Studio 2010

Pri inicializácií potrebných komponentov netreba zabudnúť na otvorenie komunikácie so zariadením s ktorým sa bude komunikovať (robotická ruka OWI).

Pri stlačení tlačidla „LED on“ sa nastavia dáta, ktoré sa zadeklarovali pri globálnych premenných tak, aby dávali príkaz robotickej ruke na rozsvietenie LED diódy. Potom sa spustí funkcia, ktorá paket aj s dátami odošle. Ak sa stlačí tlačidlo „LED off“ dáta nastavia tak, aby poslali príkaz na zhasnutie LED diódy. Následne sa spustí funkcia, ktorá paket odošle.

Najdôležitejšou časťou programu je funkcia na odoslanie paketu. Najskôr treba vytvoriť smerník, ktorý ukazuje na dáta v USB pakete. Potom treba nastaviť paket pomocou triedy `UsbSetupPacket`. Nakoniec tento paket treba po zbernici odoslať pomocou metódy `ControlTransfer()`. Argumenty spomenutej triedy a metódy sú popísané vo vysvetlení zadania. Aby funkcia bola odolná, tak celý tento spomínaný algoritmus treba vložiť do testovania funkčnosti (*try*). Ak je všetko v poriadku vypíše sa v aplikácií o tom hlásenie, inač sa vypíše správa vyskytnutej chyby.

Riešenie:

```
1    using System;
2    using System.Collections.Generic;
3    using System.Text;
4    using System.Windows.Forms;
5    using System.Runtime.InteropServices;
6    using LibUsbDotNet;
7    using LibUsbDotNet.Main;
8    namespace OWILED
9    {
10       public partial class USBtest : Form
```

```
11        {
12            private UsbDevice device = null;
13            private UsbDeviceFinder USBF;
14            public const int device_VID = 0x1267;
15            public const int device_PID = 0x0000;
16            private byte[] controlPacket=new byte[] {0,0,0};
17
18            public USBtest()
19            {
20                InitializeComponent();
21                USBF = new UsbDeviceFinder(device_VID,
                  device_PID);
22                this.device = UsbDevice.OpenUsbDevice(USBF);
23            }
24            private void button_on_Click(object sender,
              EventArgs e)
25            {
26                controlPacket[2] = 1;
27                SendControlPacket();
28            }
29            private void button_off_Click(object sender,
              EventArgs e)
30            {
31                controlPacket[2] = 0;
32                SendControlPacket();
33            }
34            public void SendControlPacket()
35            {
36                try
37                {
38                    int transferred = 0;
39                    IntPtr pointer =
                      Marshal.AllocHGlobal(3);
40                    Marshal.Copy(this.controlPacket, 0,
                      pointer, 3);
41                    UsbSetupPacket USP =
                      new UsbSetupPacket(
                      (byte)UsbRequestType.TypeVendor, 6,
                      0x0100, 0, 0);
42                    device.ControlTransfer(ref USP,
                      pointer, controlPacket.Length, out
                      transferred);
43                    label.Text = "Poslané!";
44                }
45                catch (Exception ex)
46                {
47                    label.Text="Chyba: " + ex.Message;
48                }
49    }   }   }
```

Vysvetlenie:

1-5: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.
6-7: Prilinkovanie dôležitých tried dynamickej knižnice *LibUsbDotNet.dll*.
9: Deklarácia a definovanie namespace-u tohto programu.
11: Deklarácia a definovanie hlavnej triedy programu.
13: Globálna deklarácia prázdneho objektu `device` typu `UsbDevice` reprezentujúceho samotné zariadenie
14: Globálna deklarácia objektu `USBF` typu `UsbDeviceFinder` reprezentujúceho adresu zariadenia.
15: Globálna deklarácia konštantného celého čísla (*const int*) `device_VID`, ktoré nesie hodnotu identifikátora zariadenia, konkrétne Vendor ID.
16: Globálna deklarácia konštantného celého čísla (*const int*) `device_PID`, ktoré nesie hodnotu identifikátora zariadenia, konkrétne Product ID.
17: Deklarácia poľa `controlPacket` typu *byte* reprezentujúceho dáta v pakete.
19: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.
21: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).
22: Do objektu `USBF` sa vloží adresa zariadenia (`device_VID`, `device_PID`).
23: Za pomoci adresy zariadenia uloženej v objekte `USBF` sa otvorí komunikácia so zariadením, ktoré reprezentuje objekt `device`.
26: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla „LED on“.
28: Nastavenie dát (`controlPacket`) pre rozsvietenie LED diódy, nultý bit druhého bajtu je potrebné nastaviť na hodnotu 1 (*true*).
29: Spustenie funkcie `SendControlPacket()`.
32: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla „LED off“.
34: Nastavenie dát (`controlPacket`) pre zhasnutie LED diódy, nultý bit druhého bajtu je potrebné nastaviť na hodnotu 0 (*false*).
35: Spustenie funkcie `SendControlPacket()`.
38: Deklarácia a definovanie funkcie `SendControlPacket()`.
40: Ak pri vykonaní funkcie `SendControlPacket()` nedôjde k chybe, tak sa vykonajú nasledovné kroky (*try*):
    42: Deklarácia celého čísla (*int*) `transferred` s inicializačnou hodnotou 0, ktoré sa neskôr použije ako výstup metódy `ControlTransfer()`.
    43: Deklarácia smerníka `pointer`, ktorý alokuje 3 bajty pamäte.
    44: Skopírovanie *3*-och bajtov poľa `controlPacket` od pozície *0* na adresu smerníka `pointer` (argumenty metódy `Copy()` sú kurzívou).
    45: Nastavenie paketu pomocou objektu `USP`, význam jednotlivých argumentov triedy `UsbSetupPacket`: typ požiadavky (prenosu), kód požiadavky, maximálna veľkosť dátovej časti paketu, index paketu, dĺžka očakávanej odpovede.
    46: Poslanie paketu po zbernici USB pomocou metódy `ControlTransfer()`, argumentmi tejto metódy sú: nastavenie paketu, smerník ukazujúci na uložené dáta v pamäti, dĺžka dátovej časti paketu, výstup metódy (ak je výstup 1, tak boli dáta odoslané).
    47: Vypísanie textu: „Poslané!“ do textového poľa (*label*).
49-52: Ak pri vykonaní funkcie `SendControlPacket()` došlo k chybe, tak do textového poľa (*label*) sa vypíše hlásenie o chybe.

7.4 Technologické rozhrania

Táto časť učebnice sa venuje technologickým rozhraniam ktoré sú určené pre riadenie a zber dát. Tieto rozhrania sú špeciálnymi vstupno-výstupnými kartami ktoré obsahujú pri najmenšom digitálne vstupy a výstupy, ďalej môžu obsahovať analógové vstupy, analógové výstupy a čítače-časovače. Vďaka spomínaným vstupom a výstupom je možné zabezpečiť týmito kartami (technologickými rozhraniami) riadenie alebo zber dát.

Podkapitola sa venuje technologickým rozhraniam od spoločnosti Advantech, tieto karty sa nazývajú laboratórne karty, alebo skrátene Labkarty (*LabCards*). Konkrétne sa podkapitola bude venovať trom typom týchto laboratórnych kariet PCL-812, PCL-818 a PCI-1730. Tieto laboratórne karty sú súčasťou laboratória s názvom Laboratórium konštrukcie počítačových riadiacich systémov, ktoré je určené pre výučbu predmetu Počítačové systémy v riadení.

7.4.1 Laboratórne karty PCL-812 a PCL-818

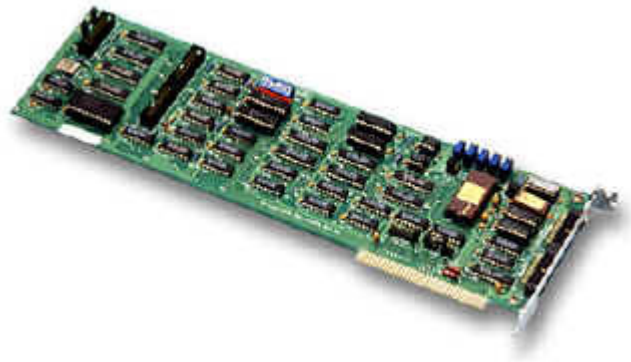
Laboratórne karty PCL-812 sú určené pre zbernicu ISA. Laboratórne karty sú viacfunkčné analógové a digitálne vstupno-výstupné karty, ktoré ponúkajú päť najviac požadovaných meraní a kontrolných funkcií pre počítače (počítačové systémy):

- analógový vstup a s ním spätý analógovo-digitálny prevod,
- analógový výstup a s ním spätý digitálno-analógový prevod,
- digitálny vstup,
- digitálny výstup,
- čítač – časovač.

Konkrétne pre tieto merania laboratórne karty obsahujú:

- šestnásť 12-bitových analógových vstupných kanálov,
- dva 12-bitové analógové výstupné kanály,
- šestnásť digitálnych vstupných kanálov,
- šestnásť digitálnych výstupných kanálov,
- programovateľný čítač – časovač.

Na obrázku Obr. 7.43 je vzhľad laboratórnej karty PCL-812PG:



Obr. 7.43 Laboratórna karta PCL-812PG

Okrem funkcií uvedených vyššie tieto laboratórne karty ponúkajú pohodlie programovateľnosti analógových vstupných rozsahov. Nastavenie analógových vstupných

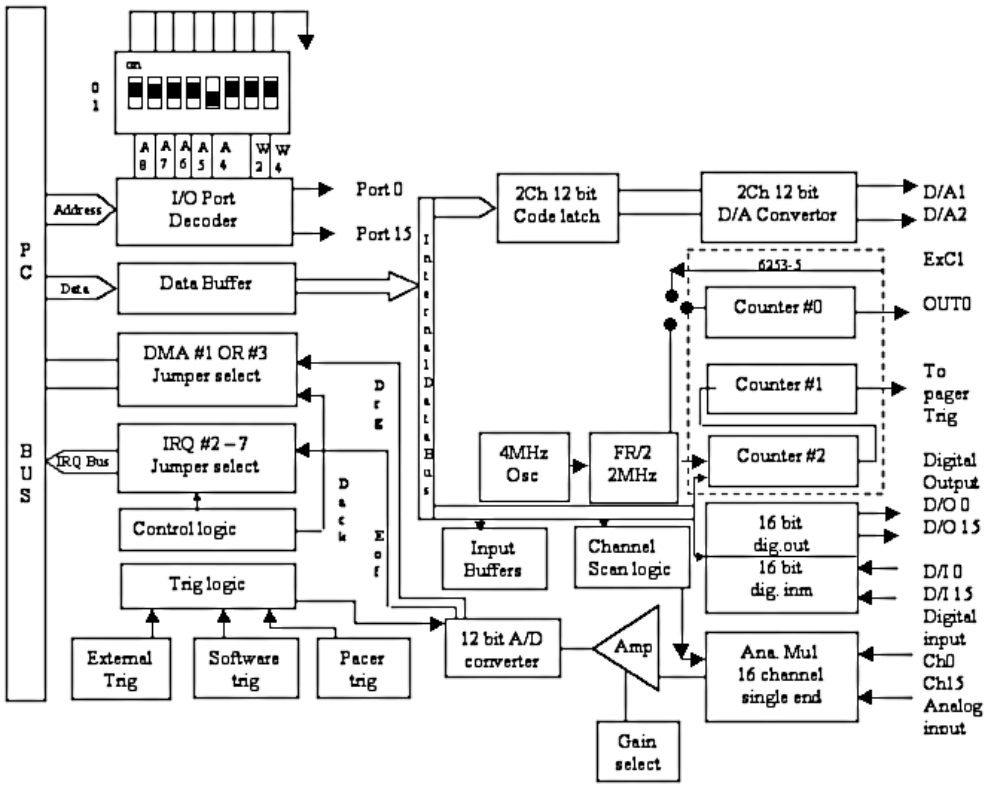
rozsahov sa nerobí pomocou DIP prepínačov, ale pomocou príkazov, ktoré sa zadávajú softwarovo. Pre aplikácie kde potrebujeme rozdielny rozsah pre rôzne kanály alebo rozdielny rozsah pre rozdielne úseky procesu, tieto karty ponúkajú pohodlie a maximálnu rozlišovaciu schopnosť.

Bohatá softwarová podpora a početné vstupno-výstupné nastavenia robí z týchto kariet ideálne zariadenia (periférie) pre priemyselné aplikácie, ktoré vychádzajú od kombinácie analógových a digitálnych vstupov a výstupov.

Príklady aplikácií týchto kariet:

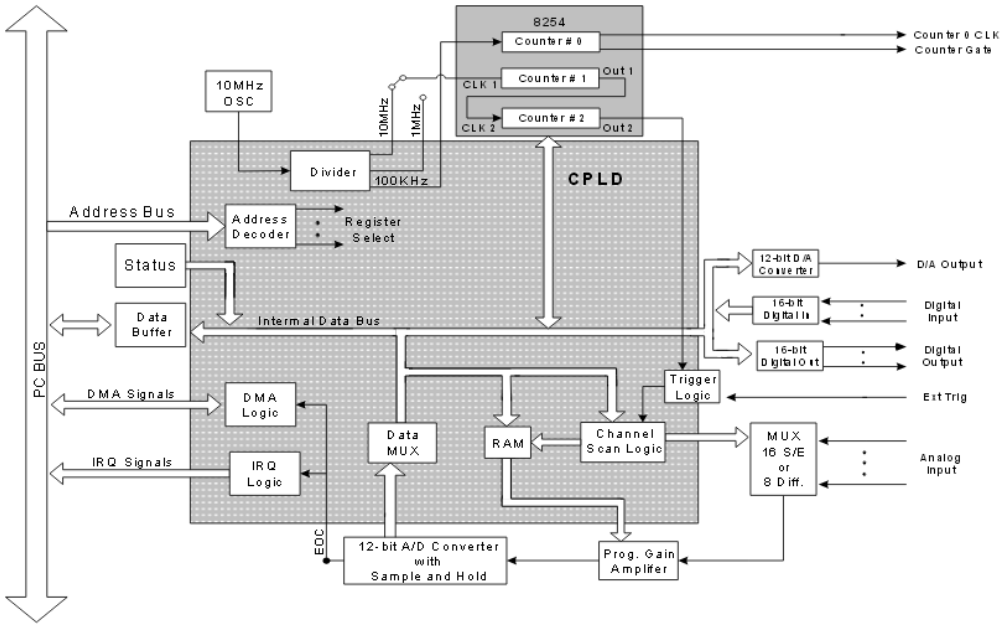
- meranie jednosmerného napätia,
- čidlo/snímač rozhranie,
- waveform analýza,
- riadenie procesov,
- programovateľný napäťový výstup,
- sledovanie spínania kontaktov,
- digitálny signál a BCD rozhranie,
- priemyselné ON/OFF riadenie,
- multiplexné a relé riadenie,
- meranie frekvencie, doby a šírky impulzu,
- generovanie čítacích impulzov,
- ať d...

Bloková schéma PCL-812 je na obrázku Obr. 7.44:



Obr. 7.44 Bloková schéma laboratórnej karty PCL-812

Bloková schéma PCL-818 je na obrázku Obr. 7.45:



Obr. 7.45 Bloková schéma laboratórnej karty PCL-818

Ďalšie dôležité vlastnosti:

- programovateľná vzorkovacia frekvencia až 30 kHz,
- A/D s DMA alebo prerušením,
- podporuje C/C++, PASCAL a Basic ovládače a obsahuje kalibračné a demonštračné programy,
- bohatá softwarová podpora,
- programovateľné A/D rozsahy.

7.4.2 Vstupno-výstupné funkcie kariet PCL-812 a PCL-818

Analógový vstup:

PCL-812 a PCL-818 používajú priemyselný štandard 12-bitovú postupnú aproximáciu A/D prevodníka (AD574) so vzorkovaním pre presné a rýchle A/D prevody. Typická doba prevodu týchto kariet je 25 μs. A/D prevod sa môže spustiť troma spôsobmi:

- pomocou programového riadenia,
- pomocou impulzu od čítača-časovača, ktorý je na doske (karte),
- pomocou vonkajšieho spúšťacieho impulzu.

Pri A/D prevode pomocou čítača-časovača sa používajú dve 16-bitové kanály mikroprocesora Intel 8253 (čítač-časovač) – PCL-812, alebo Intel 8254 – PCL-818. Kryštálový oscilátor Intel 8253 poskytuje časovú základňu 2 MHz. Oscilátor riadi generovanie spúšťacích impulzov s kmitočtovými rozsahmi od 500 kHz do 465 μHz (1 impulz každých 36 minút). Pri A/D prevode sa odporúča použiť maximálna frekvencia 40 kHz, keďže prevod trvá približne 25μs.

Vyčítanie dát (presun dát) sa dá spraviť taktiež troma spôsobmi:

- pomocou programového riadenia,
- pomocou obslužného programu prerušenia,
- pomocou DMA.

Ak použijete prerušenie na presun dát môžete jumper-om vybrať ľubovoľnú IRQ úroveň medzi IRQ2 až IRQ7. Pri použití DMA môžete jumper-om vybrať DMA kanál 1 alebo 3. Pomocou programového riadenia sa dáta vyčítajú priamo zo vstupno-výstupných registrov (portov) laboratórnej karty.

Analógový výstup:

Navyše k analógovým vstupom PCL-812 a PCL-818 poskytuje dve 12 bitové analógové výstupné kanály s dvojnásobnou vyrovnávacou pamäťou. Pomocou oboch laboratórnych kariet sa môžu obsluhovať D/A prevodníky s pevnou internou referenciou od 0 do 5 V výstupného rozsahu alebo s vonkajšou referenciou pre 0 až –10 V alebo 0 až +10 V na výstupe.

Digitálny vstup a výstup

PCL-812 a PCL-818 prišli so 16 digitálnymi vstupmi a 16 digitálnymi výstupmi, prístup cez dve 20-vývodové dual-in-line konektory. Konektory sú štandardizované na väčšine V/V kariet a základných doskách v PC-LabCard rodine od firmy Advantech. Digitálne výstupy sú normálne nastavená na úroveň signálu H (aktívny výstup je 1). Digitálny vstup je normálne nastavený na úroveň signálu L (aktívny vstup je 0).

Čítač-časovač

Tretí kanál čítača-časovača na mikroprocesoroch Intel 8253 a Intel 8254 je obsluhovaný pomocou internej alebo externej časovej základne, môže sa používať k zisťovaní frekvencie , doby a šírky impulzu.

Prístupová doba

Najskôr boli problémy s pomalými CPU, keď boli CPU už dostatočne rýchle (80486) nastal po istom čase problém s OS a architektúrou programovania (C# a .NET). Ak sa programuje v C# a prístupová doba je dlhá odporúča sa prejsť na C, alebo C++.

7.4.3 Technický popis PCL-812 a PCL-818

Analógový vstup:

- počet kanálov:
  - PCL-812 a PCL-812PG: 16 klasických kanálov,
  - PCL-818: 16 klasických kanálov, alebo 8 diferenciálnych kanálov (prepnutie medzi možnosťami sa robí pomocou jumper-ov),
- rozlíšenie: 12 bitov,
- menič: Honeywell HADC-574ACCJ alebo ekvivalentný,
- vstupný rozsah:
  - PCL-812: ±10V, ±5V, ±2V, ±1V,
  - PCL-812PG: ±10V, ±5V, ±2.5V, ±1.25V, ±0.625V, ±0.3125V,
  - PCL-818: ±10V, ±5V, ±2,5V, ±1,25V, ±0,625V
- výber rozsahu:
  - PCL-812 pomocou DIP,
  - PCL-812PG a PCL-818 pomocou programu,

- spúšťanie:
  - pomocou programu,
  - impulzu čítača-časovača,
  - externé spúšťanie,
- presun dát:
  - pomocou programového riadenia,
  - prerušenie (IRQ2 až IRQ7),
  - DMA (1 alebo 3) pre snímanie jediného kanála,
- presnosť a linearita: ± 1 bit,
- potlačenie čítaného napätia: 60 dB,
- vstupná impedancia: >10 MW,
- prepätie: ±30V maximálne jednosmerné napätie.

Analógový výstup:

- počet kanálov:
  - PCL-812 a PCL-812PG: 2 kanály,
  - PCL-818: 1 kanál,
- rozlíšenie: 12 bitov,
- D/A rozsah:
  - PCL-812: 0V až 5V , 0V až 10V,
  - PCL-812PG a PCL-818: 0V až 5V, 0V až 10V; dokáže pracovať aj s externým referenčným napätím maximálne však v rozsahu ±10V pričom referenčné napätie môže byť striedavé aj jednosmerné,
- vyrovnávací čas:
  - PCL-812 a PCL-812PG: 25 μs,
  - PCL-818: 5 μs,
- výstup:
  - PCL-812 a PCL-812PG: maximálne ±10 mA,
  - PCL-818: maximálne ±5 mA,
- D/A zariadenia: MP7623KN, AD7541AKN alebo ekvivalentné,
- linearita: ±½ bita.

Digitálny vstup (TTL):

- počet kanálov: 16,
- logická úroveň 0: 0 až 0,8 V,
- logická úroveň 1: 2,0 až 5,0 V,
- zavádzaný vstup: 0,4 mA max. pri 0,5 V (L); 0,05 mA max. pri 2,7 V (H).

Digitálny výstup (TTL):

- počet kanálov: 16,
- logická úroveň 0: 0 až 0,4 V,
- logickej úroveň 1: 2,4 až 5,0 V,
- napájanie: nízke: 8,0 mA pri 0,5 V; vysoké: 0,4 mA pri 2,4 V.

Čítač-časovač (Intel 8253):

- 32-bitový časovač s časovou základňou 2MHz,
- max. a . min frekvencia: 500 kHz do 465 μHz (vzorka každých 36 minút),
- čítač: jeden 16-bitový čítač s časovou základňou 2MHz.

Čítač-časovač (Intel 8254):

- počet kanálov: 3 kanály po 16 bitoch:
  - 2 kanály sú pernamentne nastavené ako programovateľné časovače, pričom druhým kanálom sa ešte nastavuje delič prvého,
  - 1 kanál je určený pre akékoľvek využitie čítača alebo časovača,
- 2x16-bitový časovač s nastaviteľnou časovou základňou 1 MHz a 10 MHz (pomocou jumper-a) pre prvé 2 kanály,
- max. a . min frekvencia: 2,5 MHz do 233 μHz (vzorka každých 72 minút),
- tretí kanál je 16 bitový, kde sa môže časová základňa nastaviť na 100 kHz, 1 MHz alebo 10 MHz.

Kanál prerušenia:

- kanál sa nastavuje v rozmedzí IRQ2 až IRQ7:
  - PCL-812 a PCL-812PG: pomocou jumper-ov,
  - PCL-818: softvérovo,
- povolenie prerušenia pomocou bitu INTE.

Kanál priameho prístupu do pamäte (DMA):

- kanál sa nastavuje v rozmedzí DMA1 alebo DMA3: pomocou jumper-a,
- povolenie priameho prístupu do pamäte pomocou bitu DMAE.

Prevádzkové podmienky:

- spotreba elektrickej energie (PCL-812 a PCL-812PG):
  - priemerne 500 mA pri +5 V (2,5 W), maximálne však 1 A (5 W),
  - priemerne 50 mA pri +12 V (0,6 W), max. však 100 mA (1,2W),
  - priemerne 14 mA pri -12 V (0,17 W), max. však 20 mA (0,24 W),
- spotreba elektrickej energie (PCL-818):
  - priemerne 210 mA pri +5 V (1 W), max. však 500 mA (2,5W),
  - priemerne 20 mA pri +12 V (0,24 W), max. 100 mA (1,2 W),
  - priemerne 20 mA pri -12 V (0,24 W), max. však 40 mA (0,48 W),
- prevádzková teplota: 0°C až 50°C,
- skladovacia teplota: -20°C až 65°C,
- V/V porty (registre): 16 postupných bajtov,
- konektory: všetky V/V kanály sú pripojené cez päť 20 pinových vývodov (dual-in-line konektory) na doske,
- základné adresovanie (bázová adresa):
  - DIP voliteľný,
  - PCL-812 a PCL-812PG: štandardné nastavenie je 220H, v laboratóriu sa však používa 260H kvôli kolízií s inými perifériami.
  - PCL-818: štandardné nastavenie je 300H, v laboratóriu sa však používa 220H kvôli kolízií s inými perifériami.

7.4.4 Vstupno-výstupné registre PCL-812

Adresovanie a zoznam vstupno-výstupných portov (registrov) laboratórnej karty PCL-812 (Tab. 7.11):

Tab. 7.11 Adresovanie registrov (vstupno-výstupných portov) laboratórnej karty PCL-812

Adresa	Čítanie	Zápis
Báza + 0H	Čítač-časovač 0	
Báza + 1H	Čítač-časovač 1	
Báza + 2H	Čítač-časovač 2	
Báza + 3H	–	Riadiace slovo čítača-časovača
Báza + 4H	A/D dolný bajt	D/A dolný bajt (1. kanál)
Báza + 5H	A/D horný bajt	D/A horný bajt (1. kanál)
Báza + 6H	Digitálny vstup dolný bajt	D/A dolný bajt (2. kanál)
Báza + 7H	Digitálny vstup horný bajt	D/A horný bajt (2. kanál)
Báza + 8H	–	Vymaž požiadavku na prerušenie
Báza + 9H	–	Rozsah analógového vstupu
Báza + AH	–	Nastavenie skenovacích kanálov (MUX)
Báza + BH	–	Riadiace slovo PCL-812
Báza + CH	–	Softvérový spúšťač A/D prevodu
Báza + DH	–	Digitálny výstup dolný bajt
Báza + EH	–	Digitálny výstup horný bajt
Báza + FH	–	–

Podľa tabuľky Tab. 7.11 je zrejmé že laboratórna karta PCL-812 používa 19 registrov.

1. až 3. register: Registre deličov čítača-časovača (Báza + 0H, Báza + 1H a Báza + 2H):

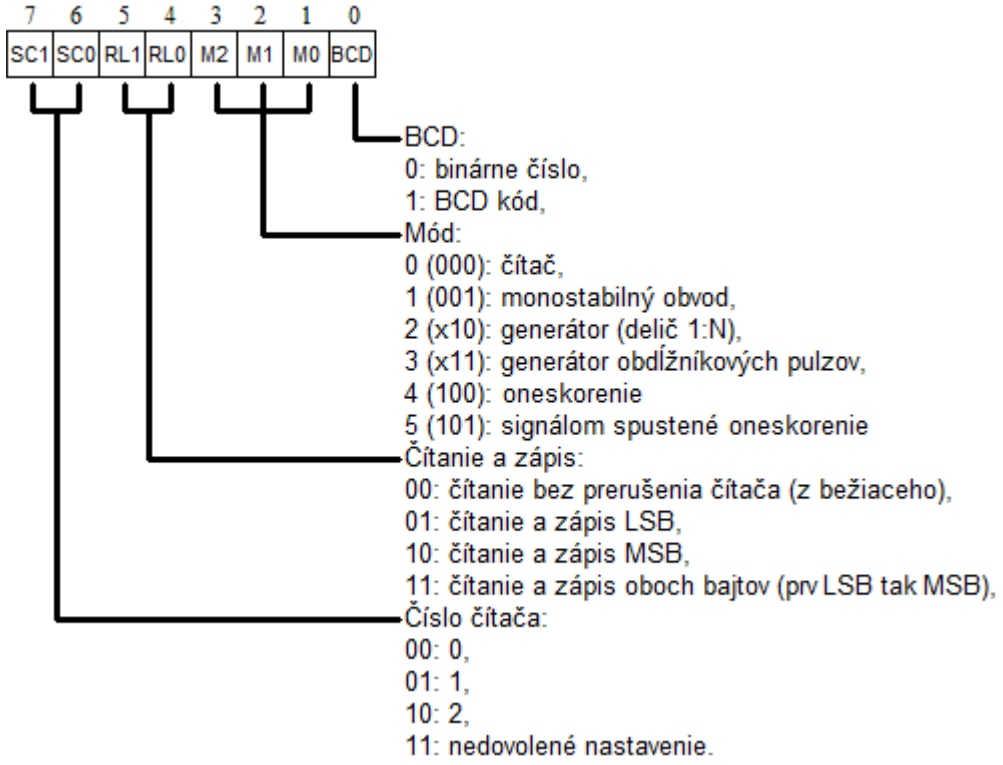
- jediné registre PCL-812 ktoré sú určené pre zápis aj čítanie, vyčítavať sa dá naposledy zapísaná hodnota,
- 3 registre 16 bitových deličov čítača-časovača,
- všetky registre sú 8 bitové, ale podľa nastavenia riadiaceho slova sa dá nastaviť LSB (8 bitov) aj MSB (8 bitov) pomocou jedného registra:
  - MSB je výsledok celočíselného delenia deliteľa s číslom 256:

$$D \div 256 = MSB\ zv.\ LSB \qquad (7.4),$$

  - LSB je celočíselný zvyšok po tomto delení,
- viac informácií o obvode Intel 8253 sú v podkapitole: 7.5.1.

4. register: Riadiace slovo čítača-časovača (Báza + 3H):

- register určený len ku zápisu,
- význam bitov riadiaceho slova čítača-časovača je na obrázku Obr. 7.46:

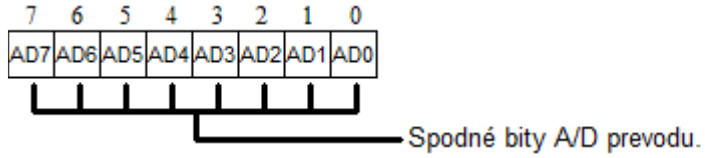


Obr. 7.46 Riadiace slovo čítača-časovača

- bližšie informácie k nastaveniu jednotlivých bitov tohto čítača-časovača sú v podkapitole: 7.5.1.

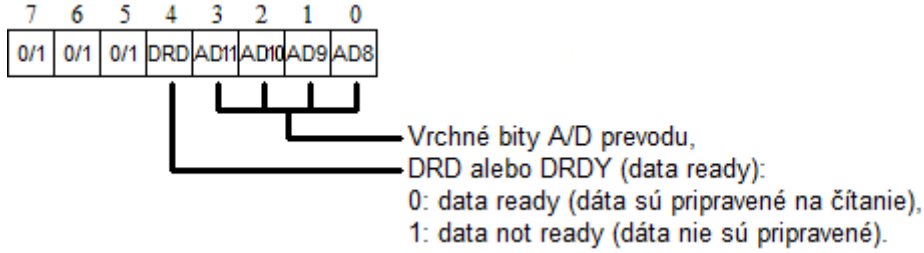
5. a 6. register: Dátové registre A/D prevodu (Báza + 4H a Báza + 5H):

- registre určené len k čítaniu,
- význam bitov prvého dátového registra A/D prevodu (báza + 4H) je na obrázku Obr. 7.47:



Obr. 7.47 Prvý dátový register A/D prevodu

- význam bitov druhého dátového registra A/D prevodu (báza + 5H) je na obrázku Obr. 7.48:



Obr. 7.48 Druhý dátový register A/D prevodu

- ak bit DRD je rovný 0 (*false*), tak sú dáta pripravené na čítanie,
- spodný bajt (*S*) stačí vyčítať z registra báza + 4H, ale vrchný bajt (*V*) treba ešte vypočítať:

$$Hodnota(Báza + 5H) \div 16 = X\ zv.\ \mathbf{V} \qquad (7.5),$$

- algoritmus v programovacom jazyku C/C++ (C#):

```
H_Byte = inportb(base + 0x5) % 16;
```

- výpočet inžinierskych jednotiek (*I*) z vrchného (*V*) a spodného (*S*) bajtu:

$$I = 256V + S \qquad (7.6),$$

- algoritmus v programovacom jazyku C/C++ (C#):

```
I_Unit = 256*H_Byte + L_Byte;
```

- výpočet napätia (*U*) pri referenčnom napätí U_{ref}:

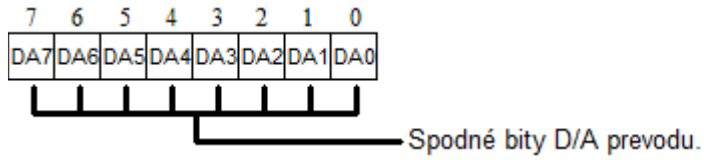
$$U = \frac{U_{ref}\,(I - 2048)}{2048} \qquad (7.7),$$

- algoritmus v programovacom jazyku C/C++ (C#):

```
Volts = (double)(I_Unit - 2048) / 204.8;
```

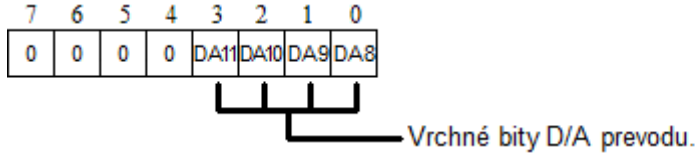
7. a 8. register: Dátové registre prvého kanálu D/A prevodu (Báza + 4H a Báza + 5H):

- registre určené len k zápisu,
- význam bitov prvého dátového registra D/A prevodu (báza + 4H) je na obrázku Obr. 7.49:



Obr. 7.49 Prvý dátový register D/A prevodu

- význam bitov druhého dátového registra D/A prevodu (báza + 5H) je na obrázku Obr. 7.50:



Obr. 7.50 Druhý dátový register D/A prevodu

- výpočet digitálnej hodnoty vrchného (*V*) a spodného (*S*) bajtu pri referenčnom napätí U_{ref} a požadovanom napätí *U*:

$$\frac{4096U}{U_{ref}} \div 256 = V, zv.S \qquad (7.8),$$

- algoritmus v programovacom jazyku C/C++ (C#):

```
H_Byte = (int)((4096*Volts) / Ref) / 256;
L_Byte = (int)((4096*Volts) / Ref) % 256;
```

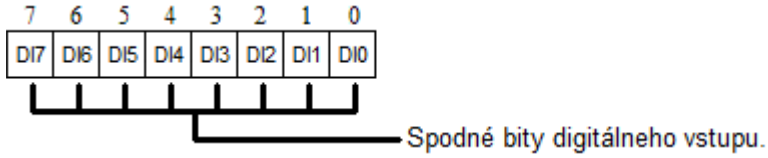
- zlomok vo vzťahu (7.8) zodpovedá inžinierskym jednotkám, ktoré je pred celočíselným delením nutné zaokrúhliť na celé číslo,
- nakoniec stačí už len do registra na adrese báza + 4H zapísať spodný bajt (spodné bity) a na adrese báza + 5H zapísať vrchný bajt (vrchné bity),
- obvod sa nastaví 30 μs po nastavení registra.

9. a 10. register: Dátové registre druhého kanálu D/A prevodu (Báza + 6H a Báza + 7H):

- registre určené len k zápisu,
- bity majú rovnaký význam ako 7. a 8. register len sú určené pre druhý kanál D/A prevodu (báza + 4H zodpovedá báze + 6H a báza + 5H zodpovedá báze + 7H).

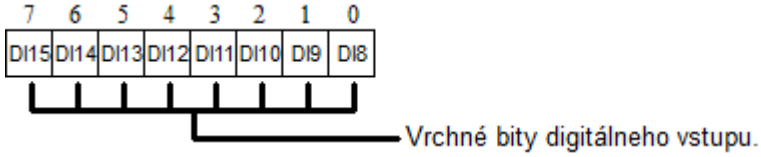
11. a 12. register: Dátové registre digitálneho vstupu (Báza + 6H a Báza + 7H):

- registre určené len k čítaniu,
- vodiče digitálneho vstupu sú uzemňované takže, ak je bit zopnutý nadobúda hodnotu 0 (*false*), inač nadobúda hodnotu 1 (*true*),
- význam bitov prvého dátového registra digitálneho vstupu (báza + 6H) je na obrázku Obr. 7.51 (prvých 8 kanálov digitálneho vstupu):



Obr. 7.51 Prvý dátový register digitálneho vstupu

- význam bitov druhého dátového registra digitálneho vstupu (báza + 7H) je na obrázku Obr. 7.52 (posledných 8 kanálov digitálneho vstupu):



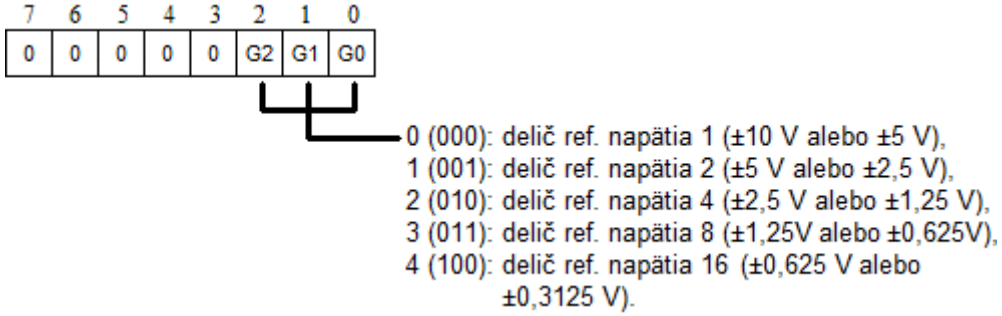
Obr. 7.52 Druhý dátový register digitálneho vstupu

13. register: Register vynulovania prerušenia (Báza + 8H):

- register určený len k zápisu,
- zápisom akejkoľvek hodnoty sa vymaže požiadavka o prerušenie a môže prerušenie nastať znovu.

14. register: Register nastavovania rozsahu analógového vstupu (Báza + 9H):

- register určený len k zápisu,
- význam bitov registra nastavenia rozsahu analógového vstupu (báza + 9H) je na obrázku Obr. 7.53:

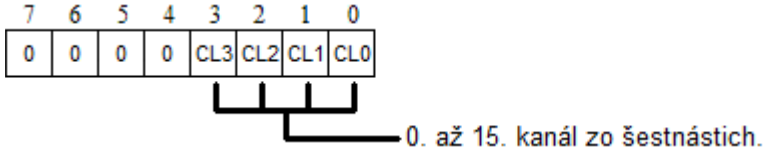


Obr. 7.53 Register nastavenia rozsahu analógového vstupu

- základné referenčné napätie bez deliča (10 V alebo 5 V) sa nastavuje pomocou *jumper*-ov.

15. register: Register nastavenia skenovacích kanálov – MUX (Báza + AH):

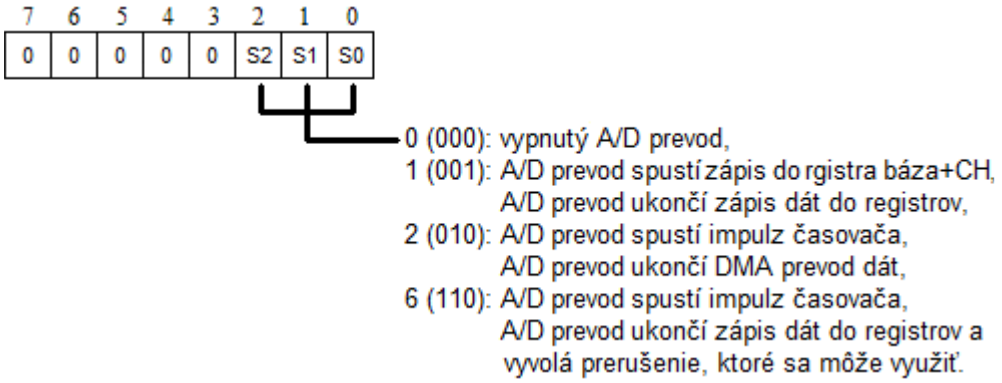
- register určený len k zápisu,
- pomocou tohto registra sa nastavuje kanál ktorý je potrebné merať pomocou A/D prevodníka,
- význam bitov registra nastavenia skenovacích kanálov – MUX (báza + AH) je na obrázku Obr. 7.54:



Obr. 7.54 Register nastavenia skenovacích kanálov - MUX

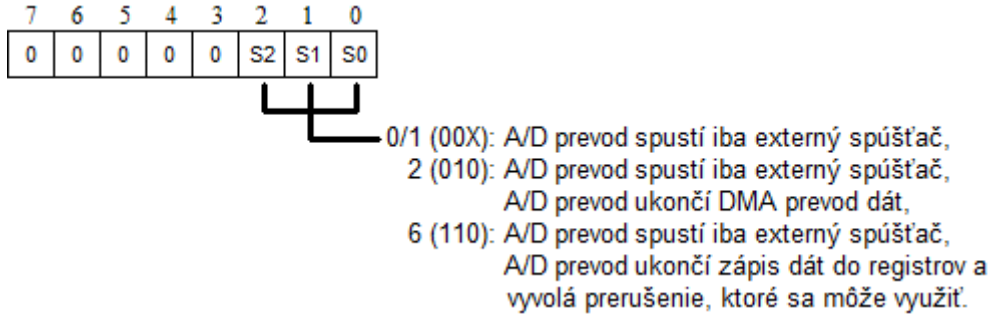
16. register: Riadiaci register PCL-812 (Báza + BH):

- register určený len k zápisu,
- význam bitov riadiaceho registra (báza + BH) v prípade, že nie je povolený externý spúšťač A/D prevodu (povoľuje sa pomocou *jumper*-a) je na obrázku Obr. 7.55:



Obr. 7.55 Riadiaci register PCL-812 pri zakázanom externom spúšťači

- prerušením môže byť aj externý signál (pri nastavení 110), ktorý sa nastaví pomocou jumper-a (iný jumper ako pri zopínaní externého spúšťača),
- význam bitov riadiaceho registra (báza + BH) v prípade, že je povolený externý spúšťač A/D prevodu je na obrázku Obr. 7.56:



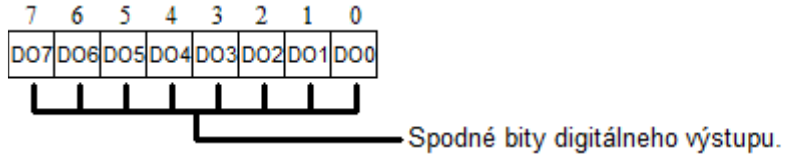
Obr. 7.56 Riadiaci register PCL-812 pri povolenom externom spúšťači

17. register: Register softvérového spúšťača A/D prevodu (Báza + CH):

- register určený len k zápisu,
- zápisom akejkoľvek hodnoty sa spustí softvérový A/D prevod (softvérový spúšťač).

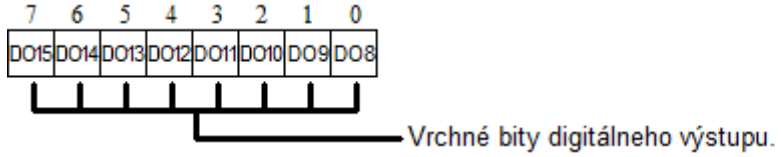
18. a 19. register: Dátové registre digitálneho výstupu (Báza + DH a Báza + EH):

- register určený len k zápisu,
- význam bitov prvého dátového registra digitálneho výstupu (báza + DH) je na obrázku Obr. 7.57 (prvých 8 kanálov digitálneho výstupu):



Obr. 7.57 Prvý dátový register digitálneho výstupu

- význam bitov druhého dátového registra digitálneho výstupu (báza + EH) je na obrázku Obr. 7.58 (posledných 8 kanálov digitálneho výstupu):



Obr. 7.58 Druhý dátový register digitálneho výstupu

7.4.5 Vstupno-výstupné registre PCL-818

Adresovanie a zoznam vstupno-výstupných registrov laboratórnej karty PCL-818 je v tabuľke Tab. 7.12:

Tab. 7.12 Adresovanie registrov (vstupno-výstupných portov) laboratórnej karty PCL-818

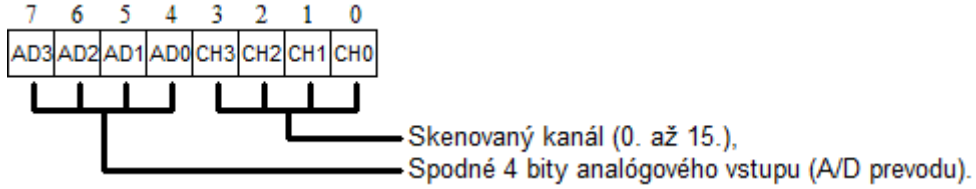
Adresa	Čítanie	Zápis
Báza + 0H	A/D dolný bajt a kanál	Softvérový spúšťač A/D prevodu
Báza + 1H	A/D horný bajt	Rozsah analógového vstupu
Báza + 2H	Nastavenie skenovacích kanálov (MUX)	
Báza + 3H	Digitálny vstup dolný bajt	Digitálny výstup dolný bajt
Báza + 4H	–	D/A dolný bajt
Báza + 5H	–	D/A horný bajt
Báza + 6H	–	–
Báza + 7H	–	–
Báza + 8H	Stavové slovo PCL-818	Vymaž požiadavku na prerušenie
Báza + 9H	Riadiace slovo PCL-818	
Báza + AH	–	Zapnutý čítač-časovač
Báza + BH	Digitálny vstup horný bajt	Digitálny výstup horný bajt
Báza + CH	Čítač-časovač 0	
Báza + DH	Čítač-časovač 1	
Báza + EH	Čítač-časovač 2	
Báza + FH	–	Riadiace slovo čítača-časovača

Podľa tabuľky Tab. 7.12 je zrejmé že aj laboratórna karta PCL-818 používa 19 registrov, ale iných a s inými adresami ako PCL-812.

1. a 2. register: Dátové registre A/D prevodu (Báza + 0H a Báza + 1H):

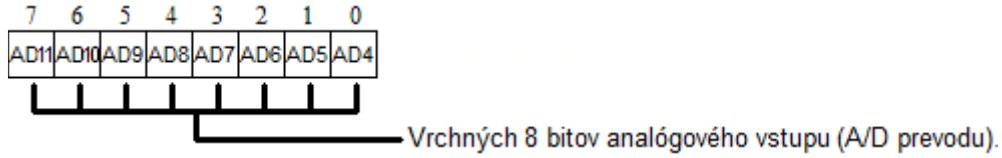
- registre určené len k čítaniu,

- význam bitov prvého dátového registra A/D prevodu (báza + 0H) je na obrázku Obr. 7.59:



Obr. 7.59 Prvý dátový register A/D prevodu

- význam bitov druhého dátového registra A/D prevodu (báza + 1H) je na obrázku Obr. 7.60:



Obr. 7.60 Druhý dátový register A/D prevodu

- vrchný (horný) bajt (*V*) stačí vyčítať z registra báza + 1H, ale spodný (dolný) bajt (*S*) treba ešte vypočítať (*K* - kanál):

$$Hodnota(Báza + 0H) \div 16 = S\ zv.\ K \qquad (7.9),$$

- algoritmus v programovacom jazyku C/C++ (C#):

```
L_Byte = inportb(base) / 16;
```

- výpočet inžinierskych jednotiek (*I*) z vrchného (*V*) a spodného (*S*) bajtu:

$$I = 16V + S \qquad (7.10),$$

- algoritmus v programovacom jazyku C/C++ (C#):

```
I_Unit = 16*H_Byte + L_Byte;
```

- výpočet napätia (*U*) pri referenčnom napätí U_{ref}:

$$U = \frac{U_{ref}\,(I - 2048)}{2048} \qquad (7.11),$$

- algoritmus v programovacom jazyku C/C++ (C#):

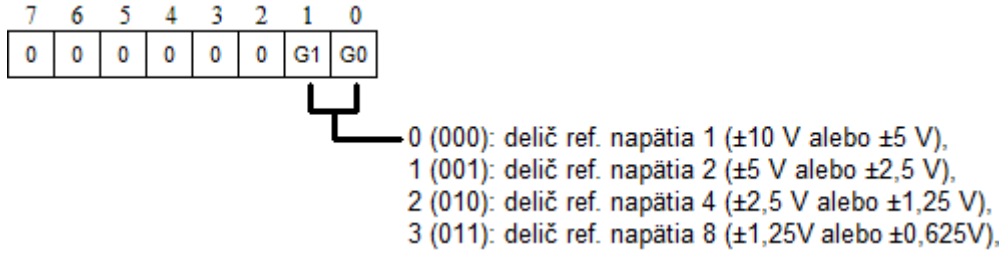
```
Volts = (double)(I_Unit - 2048) / 204.8;
```

3. register: Register softvérového spúšťača A/D prevodu (Báza + 0H):

- register určený len k zápisu,
- zápisom akejkoľvek hodnoty sa spustí softvérový A/D prevod (softvérový spúšťač).

4. register: Register nastavovania rozsahu analógového vstupu (Báza + 1H):

- register určený len k zápisu,
- význam bitov registra nastavenia rozsahu analógového vstupu (báza + 1H) je na obrázku Obr. 7.61:

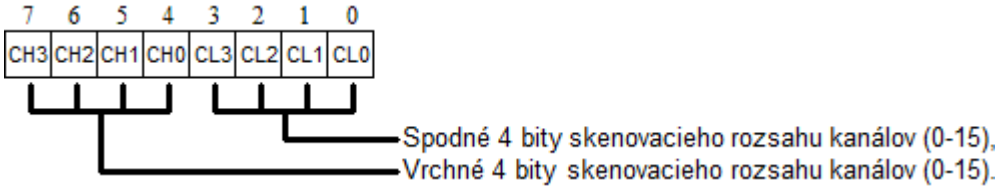


Obr. 7.61 Register nastavenia rozsahu analógového vstupu

- základné referenčné napätie bez deliča (10 V alebo 5 V) sa nastavuje pomocou *jumper*-ov.

5. register: Register nastavenia skenovacích kanálov – MUX (Báza + 2H):

- register určený k zápisu aj čítaniu,
- pomocou tohto registra sa nastavuje rozsah kanálov (napríklad od 3. po 6. kanál), ktorý je potrebné merať pomocou A/D prevodníka, pri čítaní sa vyčítava naposledy zapísaná hodnota do registra,
- význam bitov registra nastavenia skenovacích kanálov – MUX (báza + 2H) je na obrázku Obr. 7.62:

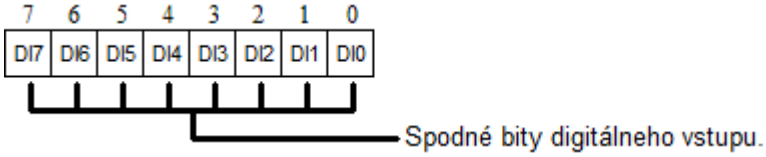


Obr. 7.62 Register nastavenia skenovacích kanálov - MUX

- na rozdiel od PCL-812 tú sa nastavuje rozsah skenovacích kanálov a nie konkrétny kanál, takže ak je potrebné merať len jeden kanál, tak treba zadať rovnaké spodné aj vrchné bity (16*K* + *K*, *K* - kanál).

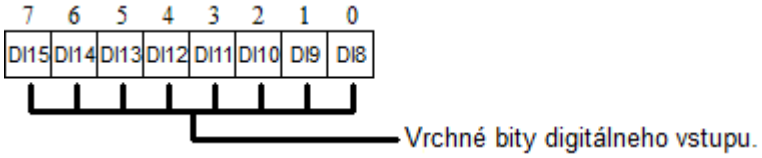
6. a 7. register: Dátové registre digitálneho vstupu (Báza + 3H a Báza + BH):

- registre určené len k čítaniu,
- význam bitov prvého dátového registra digitálneho vstupu (báza + 3H) je na obrázku Obr. 7.63 (prvých 8 kanálov digitálneho vstupu):



Obr. 7.63 Prvý dátový register digitálneho vstupu

- význam bitov druhého dátového registra digitálneho vstupu (báza + BH) je na obrázku Obr. 7.64 (posledných 8 kanálov digitálneho vstupu):

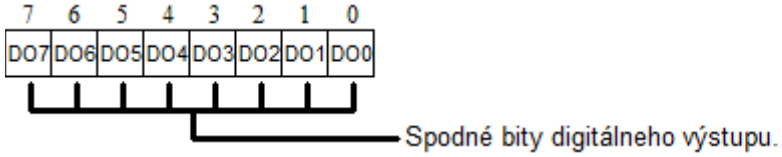


Obr. 7.64 Druhý dátový register digitálneho vstupu

- vodiče digitálneho vstupu sú uzemňované takže, ak je bit zopnutý nadobúda hodnotu 0 (*false*), inač nadobúda hodnotu 1 (*true*),

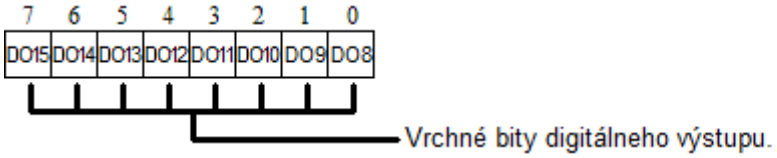
8. a 9. register: Dátové registre digitálneho výstupu (Báza + 3H a Báza + BH):

- register určený len k zápisu,
- význam bitov prvého dátového registra digitálneho výstupu (báza + 3H) je na obrázku Obr. 7.65 (prvých 8 kanálov digitálneho výstupu):



Obr. 7.65 Prvý dátový register digitálneho výstupu

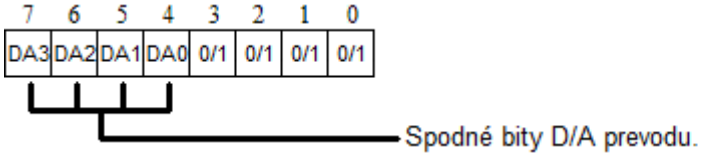
- význam bitov druhého dátového registra digitálneho výstupu (báza + BH) je na obrázku Obr. 7.66 (posledných 8 kanálov digitálneho výstupu):



Obr. 7.66 Druhý dátový register digitálneho výstupu

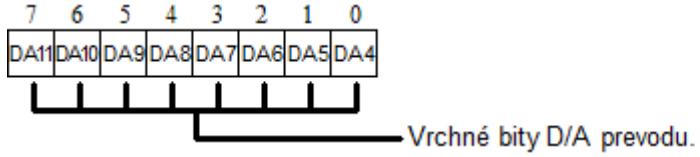
10. a 11. register: Dátové registre D/A prevodu (Báza + 4H a Báza + 5H):

- registre určené len k zápisu,
- význam bitov prvého dátového registra D/A prevodu (báza + 4H) je na obrázku Obr. 7.67:



Obr. 7.67 Prvý dátový register D/A prevodu

- význam bitov druhého dátového registra D/A prevodu (báza + 5H) je na obrázku Obr. 7.68:



Obr. 7.68 Druhý dátový register D/A prevodu

- výpočet digitálnej hodnoty vrchného (V) a spodného (S) bajtu pri referenčnom napätí U_{ref} a požadovanom napätí U:

$$\frac{4096U}{U_{ref}} \div 16 = V, zv.S_P \qquad (7.12),$$

$$S = 16S_P \qquad (7.13),$$

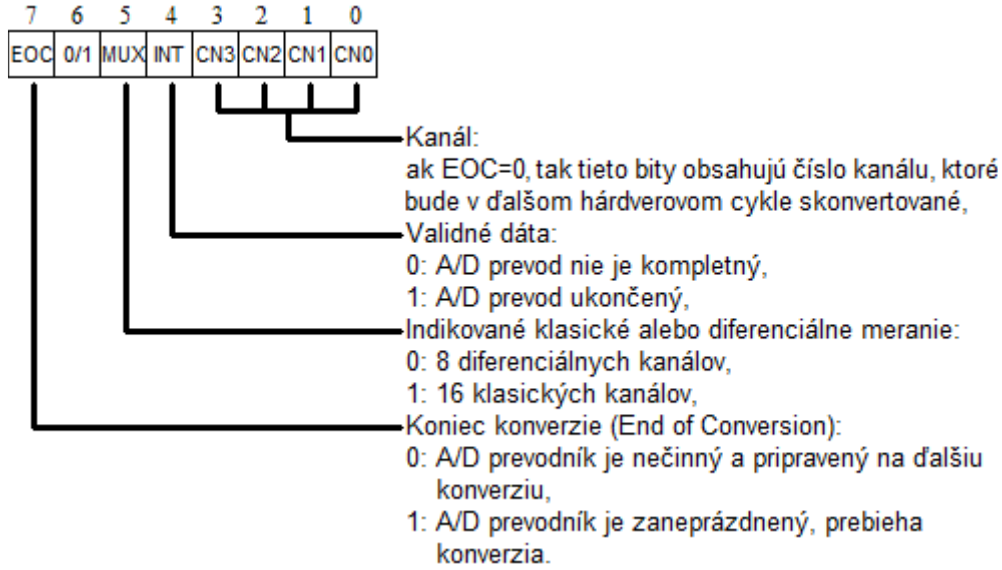
- algoritmus v programovacom jazyku C/C++ (C#):

```
    H_Byte = (int)((4096*Volts) / Ref) / 16;
 L_Byte = 16*((int)((4096*Volts) / Ref) % 16);
```

- zlomok vo vzťahu (7.12) zodpovedá inžinierskym jednotkám, ktoré je pred celočíselným delením nutné zaokrúhliť na celé číslo,
- nakoniec stačí už len do registra na adrese báza + 4H zapísať spodný bajt (spodné bity) a na adrese báza + 5H zapísať vrchný bajt (vrchné bity),
- obvod sa nastaví 5 μs po nastavení registra.

12. register: Stavový register A/D prevodu (Báza + 8H):

- register určený len k čítaniu,
- význam bitov stavového registra A/D prevodu (báza + 8H) je na Obr. 7.69:



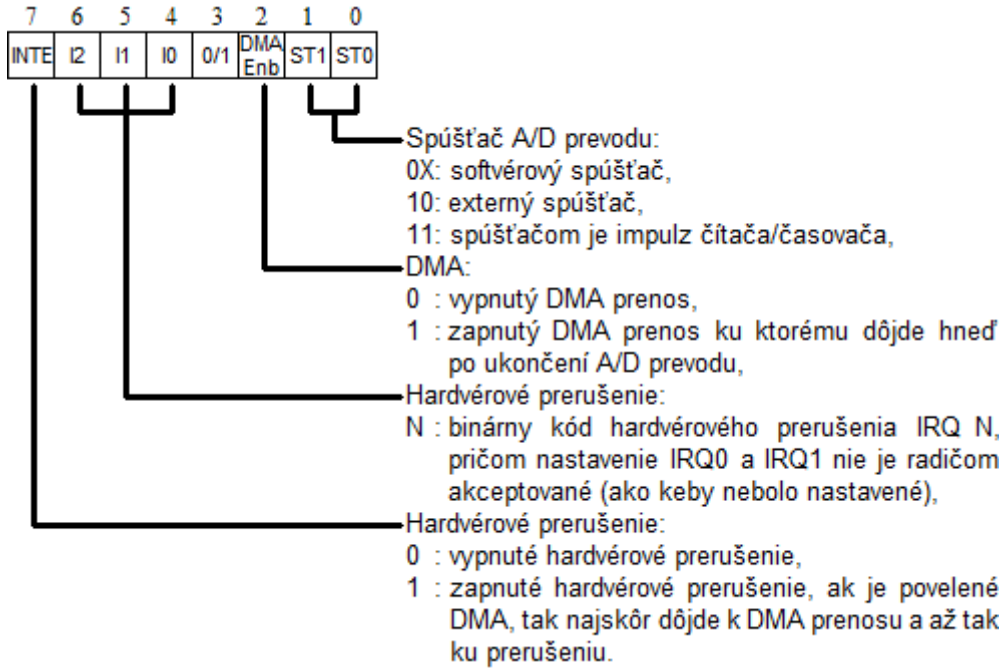
Obr. 7.69 Stavový register A/D prevodu

13. register: Register vynulovania prerušenia (Báza + 8H):

- register určený len k zápisu,
- zápisom akejkoľvek hodnoty sa vymaže požiadavka o prerušenie a môže prerušenie nastať znovu.

14. register: Riadiaci register PCL-818 (Báza + 9H):

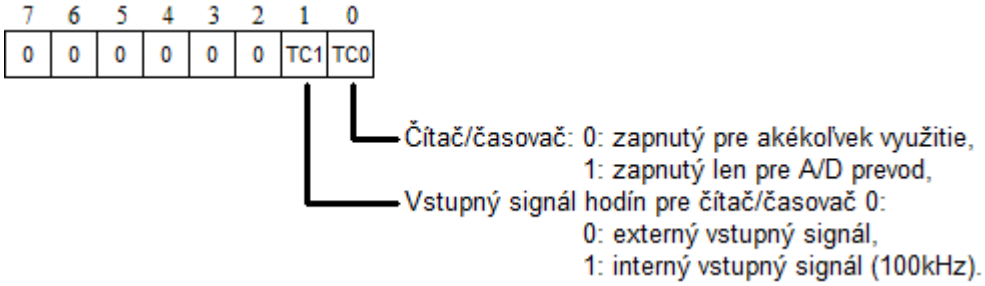
- register určený k zápisu aj čítaniu,
- riadiaci register PCL-818 slúži k nastaveniu A/D prevodu,
- význam bitov riadiaceho registra (slova) PCL-818 (báza + 9H) je na obrázku Obr. 7.70:



Obr. 7.70 Riadiaci register PCL-818 (A/D prevodu)

15. register: Register spúšťania čítača/časovača (Báza + AH):

- register určený k zápisu,
- význam bitov registra (báza + AH) je na obrázku Obr. 7.71:



Obr. 7.71 Register spúšťania čítača/časovača

16. až 18. register: Registre deličov čítača-časovača (Báza + CH, Báza + DH a Báza + EH):

- registre PCL-818 ktoré sú určené pre zápis aj čítanie, vyčítavať sa dá naposledy zapísaná hodnota,
- 3 registre 16 bitových deličov čítača časovača,
- všetky registre sú 8 bitové, ale podľa nastavenia riadiaceho slova sa dá nastaviť LSB (8 bitov) aj MSB (8 bitov) pomocou jedného registra:
  - MSB je výsledok celočíselného delenia deliteľa s číslom 256:

$$D \div 256 = MSB\ zv.\ LSB \qquad (7.4),$$

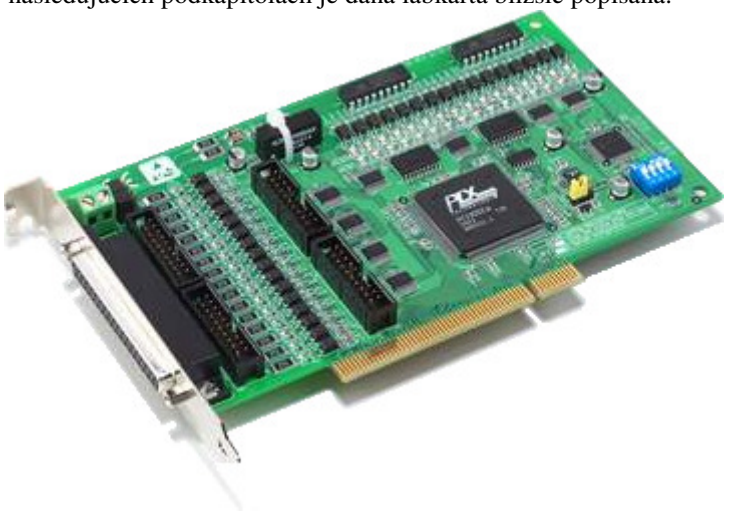
  - LSB je celočíselný zvyšok po tomto delení,
- viac informácií o obvode Intel 8254 sú v podkapitole: 7.5.1.

19. register: Riadiace slovo čítača-časovača (Báza + FH):

- register určený len ku zápisu,
- význam bitov riadiaceho slova čítača-časovača je na obrázku Obr. 7.46 pri popise registrov PCL-812,
- bližšie informácie k nastaveniu jednotlivých bitov tohto čítača-časovača sú v podkapitole: 7.5.1.

7.4.6 Laboratórna karta PCI-1730

Laboratórna karta PCI-1730 má iba digitálne vstupy a výstupy. Jej zjednodušenými variantmi sú labkarty PCI-1733 a PCI-1734. Labkarta PCI-1733 má iba digitálne vstupy a labkarta PCI-1734 má iba digitálne výstupy. Na obrázku Obr. 7.72 je zobrazená labkarta PCI-1730. V nasledujúcich podkapitolách je daná labkarta bližšie popísaná.



Obr. 7.72 Laboratórna karta Advantech PCI-1730

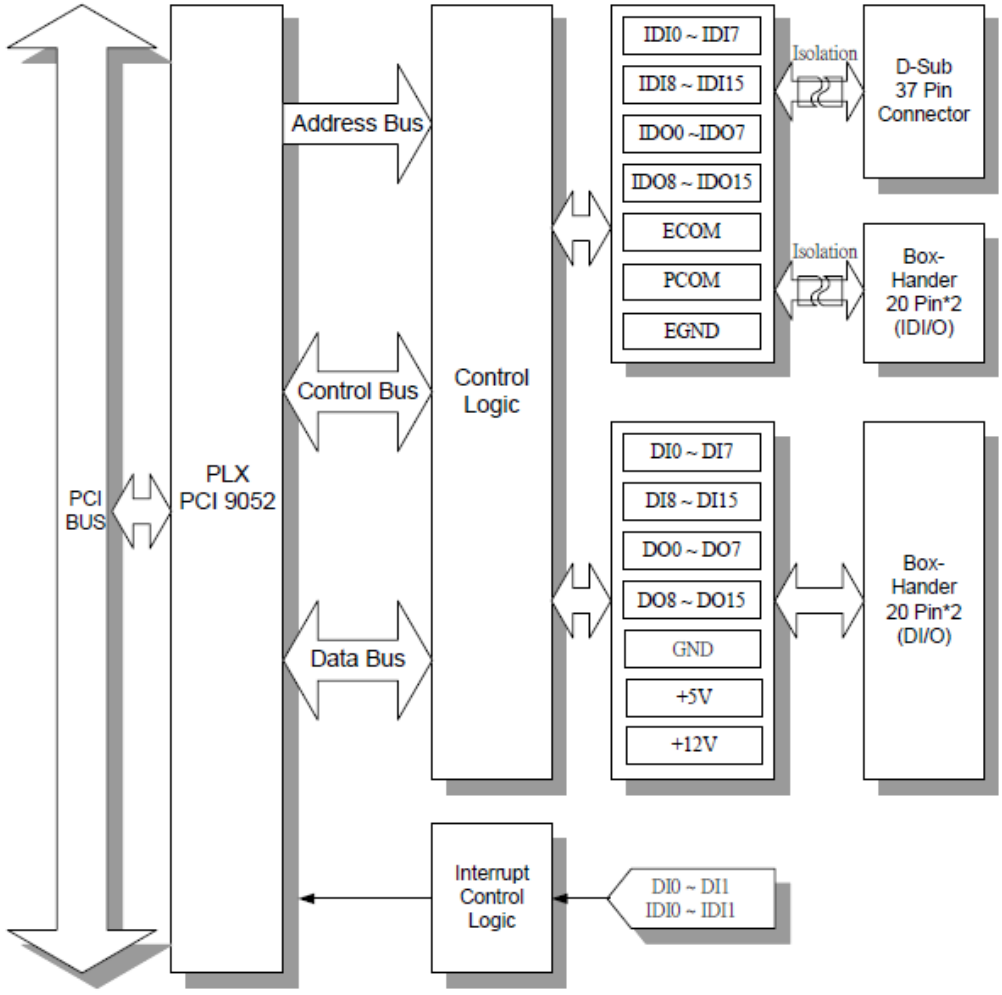
Vlastnosti laboratórnej karty PCI-1730:

- 32 samostatných digitálnych vstupno-výstupných kanálov (16 vstupov a 16 výstupov),
- 32 TTL digitálnych vstupno-výstupných kanálov (16 vstupov a 16 výstupov),

- kompatibilná ako PCL-730, ktorá bola určená pre zbernicu ISA,
- izolácia vyššieho napätia na izolovaných vstupno-výstupných kanáloch (2,5 V),
- zachytávanie prerušení,
- 2-krát 20 pinový konektor pre izolované vstupné a výstupné kanály a 2-krát 20 pinový konektor pre TTL vstupno-výstupné kanály,
- ochrana proti statickému napätiu (2 V),
- ochrana proti vysokému napätiu (70 V),
- rozsah napätia 5 a 30 V.

Jej použitie je možné v akýchkoľvek priemyselných podmienkach, kde je potrebné pracovať s digitálnymi vstupmi a výstupmi.

Na obrázku Obr. 7.73 je bloková schéma laboratórnej karty PCI-1730:



Obr. 7.73 Bloková schéma laboratórnej karty PCI-1730

7.4.7 Technický popis PCI-1730

Izolovaný digitálny vstup:

- počet kanálov: 16 (obojsmerné, bi-directional),
- optická izolácia: 2 500 V,
- čas do odpovede (reakcie) optickej izolácie: 25 μs,
- prepäťová ochrana: 70 V,
- logická úroveň 1: 5 V až 30 V,
- logická úroveň 0: 0V až 2 V,
- zavádzací vstup:
  - typický 1,4 mA pri 5 V (7 mW),
  - typický 3,9 mA pri 12 V (47 mW),
  - typický 8,2 mA pri 24 V (0,2 W),
  - typický 10,3 mA pri 30 V (0,3 W).

Izolovaný digitálny výstup:

- počet kanálov: 16,
- optická izolácia: 2 500 V,
- výstupné napätie: s otvoreným kolektorom od 5 V do 40 V,
- napájanie:
  - maximálne 200 mA na jeden kanál (5V: 1 W, 40V: 8 W),
  - maximálne však 2,4 A na všetky výstupy (5V: 12 W, 40V: 96 W), takže ak sú všetky zopnuté tak 150 mA na jeden kanál.

Neizolovaný digitálny vstup:

- logická úroveň 0: 0 až 0,8 V,
- logickej úroveň 1: 2,0 až 5,0 V,

Neizolovaný digitálny výstup:

- logická úroveň 0: maximálne 0,5 V,
- logickej úroveň 1: minimálne 2,4 V,
- napájanie:
  - pri logickej 0 (max. 0,5 V) je to typicky +24 mA (max.: 12 mW),
  - pri logickej 1 (max. 5 V) je to maximálne -15 mA (max.: 75 mW).

Prevádzkové podmienky:

- konektory: 37 pinový D-SUB samica a 4x 20 pinové konektory kompatibilné s predošlými laboratórnymi kartami (PCL-812 a PCL-818),
- rozmery: 175 mm x 100 mm,
- spotreba elektrickej energie:
  - priemerne 250 mA pre 5 V napájanie (1,25 W),
  - priemerne 35 mA pre 12 V napájanie (0,42 W),
  - maximálne však 400 mA pre 5 V napájanie (2 W),
- prevádzková teplota: 0°C až 60°C,
- skladovacia teplota: -20°C až 85°C,
- vlhkosť: 5% až 95% (nesmie dôjsť ku kondenzovaniu vody).

7.4.8 Vstupno-výstupné registre PCI-1730

Adresovanie a zoznam vstupno-výstupných registrov laboratórnej karty PCI-1730 je v tabuľke Tab. 7.13:

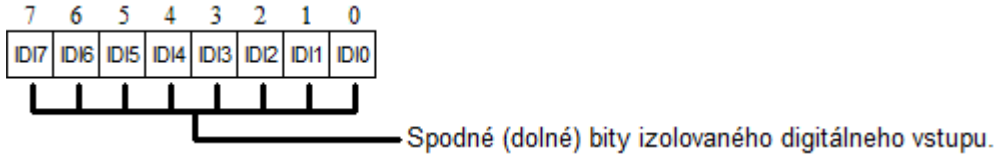
Tab. 7.13 Adresovanie registrov (vstupno-výstupných portov) laboratórnej karty PCI-1730

Adresa	Čítanie	Zápis
Báza + 00H	Izolovaný digitálny vstup (L bajt)	Izolovaný digitálny výstup (L bajt)
Báza + 01H	Izolovaný digitálny vstup (H bajt)	Izolovaný digitálny výstup (H bajt)
Báza + 02H	Digitálny vstup dolný bajt (L bajt)	Digitálny výstup dolný bajt (L bajt)
Báza + 03H	Digitálny vstup horný bajt (H bajt)	Digitálny výstup horný bajt (H bajt)
Báza + 04H	Register identifikátora dosky	–
Báza + 05H	–	–
Báza + 06H	–	–
Báza + 07H	–	–
Báza + 08H	Register povolenia prerušení	
Báza + 09H	–	–
Báza + 0AH	–	–
Báza + 0BH	–	–
Báza + 0CH	Register spúšťača prerušení	
Báza + 0DH	–	–
Báza + 0EH	–	–
Báza + 0FH	–	–
Báza + 10H	Register vlajky prerušenia	Register mazania prerušení

Podľa tabuľky Tab. 7.13 je možné vidieť, že laboratórna karta PCI-1730 používa 13 registrov.

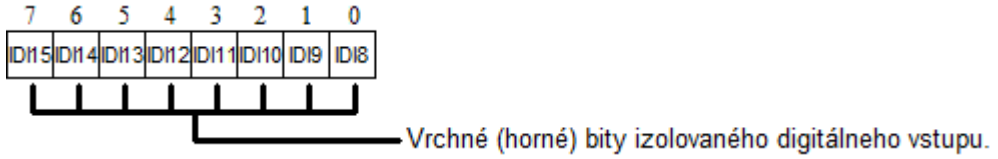
1. a 2. register: Dátové registre izolovaného digitálneho vstupu (Báza + 0H a Báza + 1H):

- registre určené len k čítaniu,
- význam bitov prvého dátového registra izolovaného digitálneho vstupu (báza + 0H) je na Obr. 7.74 (prvých 8 kanálov digitálneho vstupu):



Obr. 7.74 Prvý dátový register izolovaného digitálneho vstupu

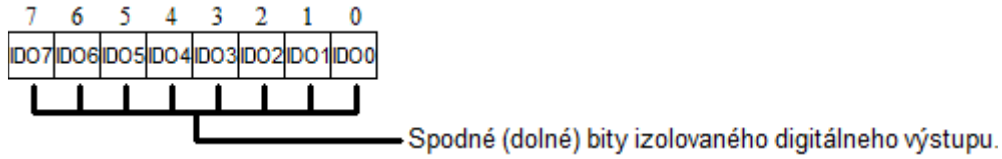
- význam bitov druhého dátového registra izolovaného digitálneho vstupu (báza + 1H) je na Obr. 7.75 (posledných 8 kanálov digitálneho vstupu):



Obr. 7.75 Druhý dátový register izolovaného digitálneho vstupu

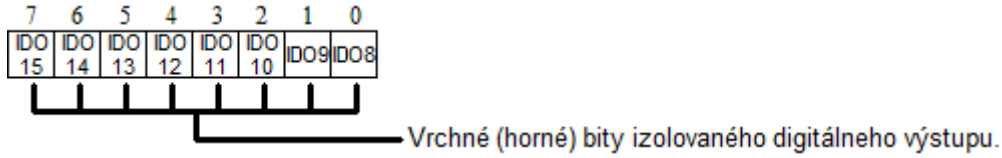
3. a 4. register: Dátové registre izolovaného digitálneho výstupu (Báza + 0H a Báza + 1H):

- register určený len k zápisu,
- význam bitov prvého dátového registra izolovaného digitálneho výstupu (báza + 0H) je na obrázku Obr. 7.76 (prvých 8 kanálov digitálneho výstupu):



Obr. 7.76 Prvý dátový register izolovaného digitálneho výstupu

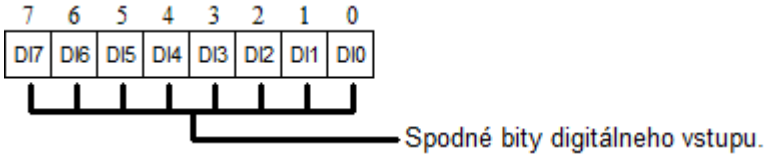
- význam bitov druhého dátového registra izolovaného digitálneho výstupu (báza + 1H) je na obrázku Obr. 7.77 (posledných 8 kanálov digitálneho výstupu):



Obr. 7.77 Druhý dátový register izolovaného digitálneho výstupu

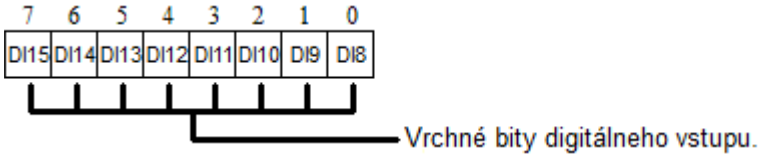
5. a 6. register: Dátové registre digitálneho vstupu (Báza + 2H a Báza + 3H):

- registre určené len k čítaniu,
- význam bitov prvého dátového registra digitálneho vstupu (báza + 2H) je na obrázku Obr. 7.78 (prvých 8 kanálov digitálneho vstupu):



Obr. 7.78 Prvý dátový register digitálneho vstupu

- význam bitov druhého dátového registra digitálneho vstupu (báza + 3H) je na obrázku Obr. 7.79 (posledných 8 kanálov digitálneho vstupu):

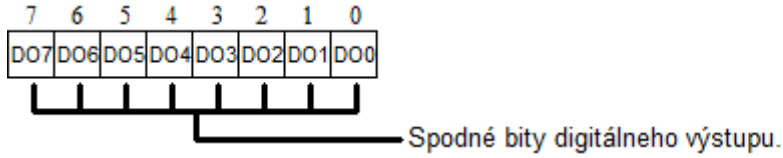


Obr. 7.79 Druhý dátový register digitálneho vstupu

- vodiče digitálneho vstupu sú uzemňované takže, ak je bit zopnutý nadobúda hodnotu 0 (*false*), inač nadobúda hodnotu 1 (*true*),

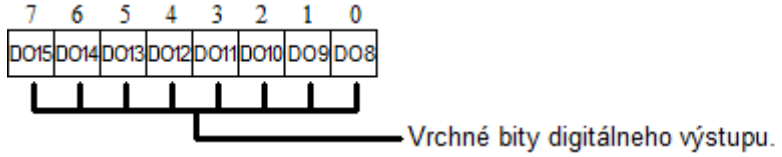
7. a 8. register: Dátové registre digitálneho výstupu (Báza + 2H a Báza + 3H):

- register určený len k zápisu,
- význam bitov prvého dátového registra digitálneho výstupu (báza + 2H) je na obrázku Obr. 7.80 (prvých 8 kanálov digitálneho výstupu):



Obr. 7.80 Prvý dátový register digitálneho výstupu

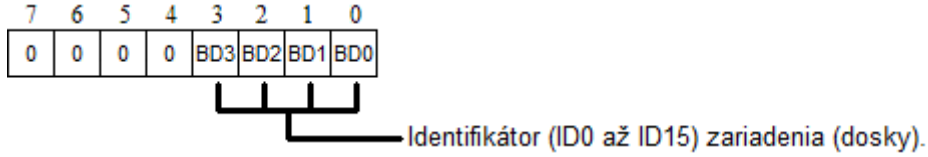
- význam bitov druhého dátového registra digitálneho výstupu (báza + 3H) je na obrázku Obr. 7.81 (posledných 8 kanálov digitálneho výstupu):



Obr. 7.81 Druhý dátový register digitálneho výstupu

9. register: Register identifikátora dosky (Báza + 4H):

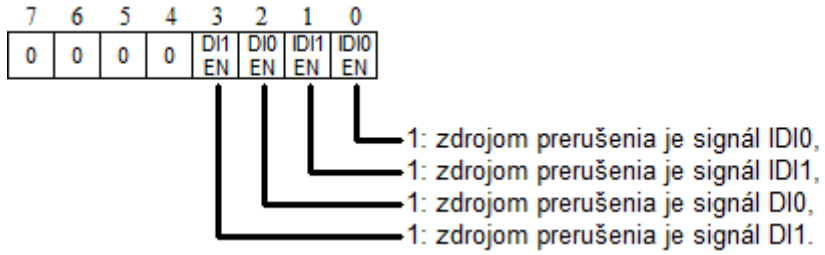
- register určený len k čítaniu,
- pri inštalácií vstupno-výstupného zariadenia (laboratórnej karty), ktoré je pripojené k počítaču pomocou zbernice PCI, sa bázová adresa priradí dynamicky,
- pre vyššie spomínaný dôvod je vhodné pri vyššom počte laboratórnych kariet nastaviť pomocou *jumper*-ov identifikátor (ID) zariadenia, ktoré je možné neskôr vyčítať pomocou tohto registra,
- význam bitov registra identifikátora dosky (báza + 4H) je na obrázku Obr. 7.82:



Obr. 7.82 Register identifikátora dosky

10. register: Register povolenia prerušení (Báza + 8H):

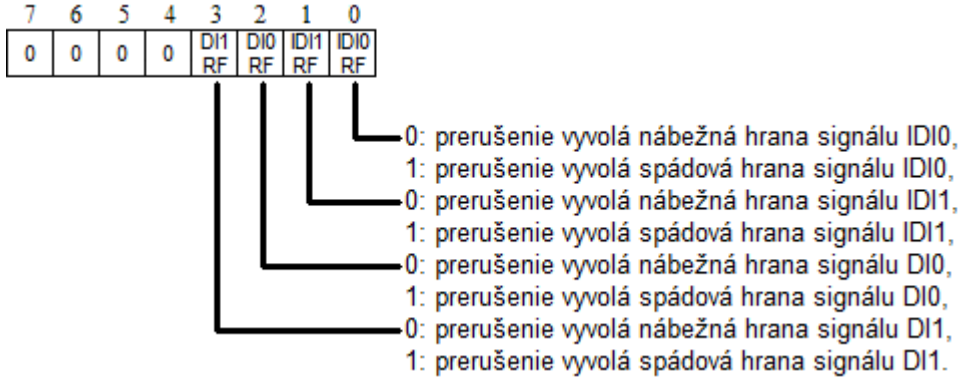
- register určený k čítaniu aj zápisu,
- v podstate pri čítaní sa jedná o stavový register povolenia prerušení a pri zapisovaní o riadiaci register povolenia prerušení,
- pomocou tohto registra sa nastavuje signál, ktorý vyvolá prerušenie,
- zdrojom prerušenia môžu byť digitálne vstupy DI0, DI1, IDI0 a IDI1,
- na Obr. 7.83 je popísaný význam bitov registra povolenia prerušení:



Obr. 7.83 Register povolenia prerušení

11. register: Register spúšťača prerušení (Báza + CH):

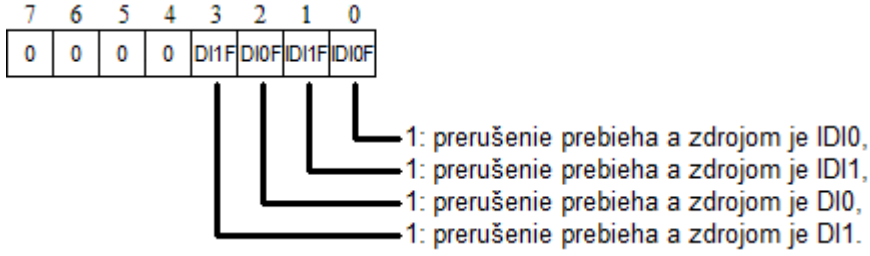
- register určený k čítaniu aj zápisu,
- v podstate pri čítaní sa jedná o stavový register spúšťača prerušení a pri zapisovaní o riadiaci register povolenia prerušení,
- pomocou tohto registra sa nastavuje spúšťač prerušenia daného signálu, ktorým môže byť nábežná hrana alebo spádová hrana signálu,
- význam bitov registra spúšťača prerušení (báza + CH) je na obrázku Obr. 7.84:



Obr. 7.84 Register spúšťača prerušení

12. register: Register vlajky prerušenia (Báza + 10H):

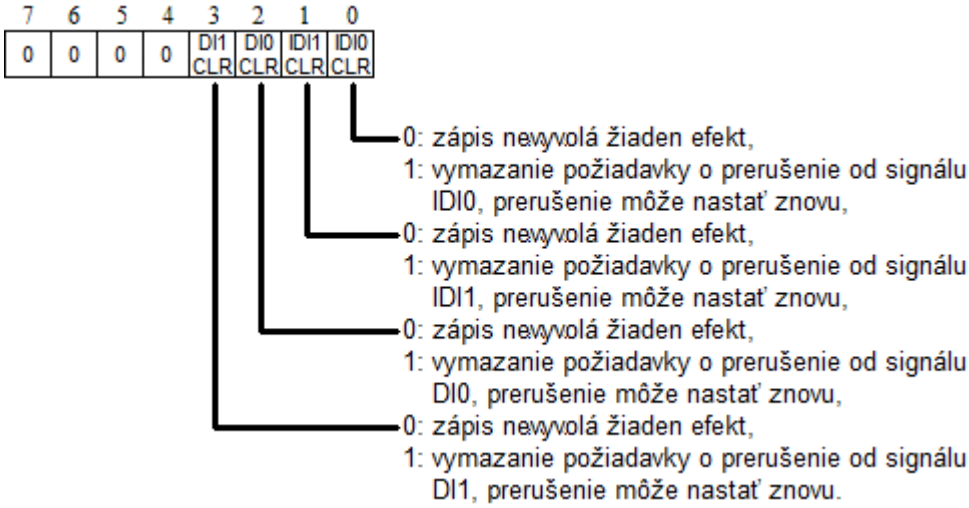
- register určený k čítaniu,
- pomocou tohto registra sa dá zistiť či práve prebieha prerušenie od daného zdroja,
- význam bitov registra vlajky prerušenia (báza + 10H) je na Obr. 7.85:



Obr. 7.85 Register povolenia prerušení

13. register: Register mazania prerušení (Báza + 10H):

- register určený k zápisu,
- tento register vymaže požiadavku o prerušenie a môže prerušenie nastať znovu,
- tieto bity je potrebné vymazať (zapísať hodnotu 1 - *true*) nato, aby mohlo nastať ďalšie prerušenie,
- význam bitov registra mazania požiadaviek o prerušenie (báza + 10H) je na obrázku Obr. 7.86:



Obr. 7.86 Register mazania prerušení

7.4.9 Programovanie laboratórnych kariet

Každá spomínaná laboratórna karta (PCL-812, PCL-818 a PCI-1730) je vložená do vstupno-výstupnej zbernice počítača a pripojená k panelu, ktorý ovláda (prípadne riadi, ak ku panelu pripojíme systém, ktorý je potrebné riadiť). Nad týmto panelom je vytlačený stručný popis pripojenia jednotlivých kanálov (digitálnych aj analógových) k laboratórnej karte (Obr. 7.87).

Obr. 7.87 Pripojenie laboratórnych kariet k panelom v laboratóriu V101b

V tejto podkapitole sa budeme venovať digitálnemu vstupu a digitálnemu výstupu. Analógovému vstupu, analógovému výstupu a čítaču/časovaču sa bude venovať ďalšia podkapitola 7.5 nakoľko pri týchto vstupno-výstupných obvodoch sú potrebné ešte ďalšie poznatky, ktoré sú v danej podkapitole vysvetlené. Ďalším dôvodom oddelenia týchto podkapitol je fakt, že tieto vstupno-výstupné obvody sa nenachádzajú iba na laboratórnych kartách.

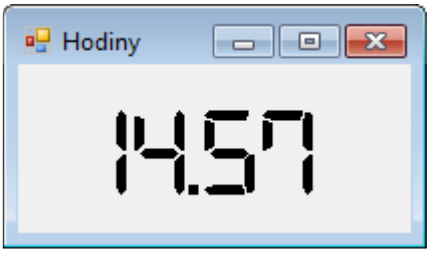
Táto podkapitola obsahuje tri zadania, pre každú laboratórnu kartu jedno zadanie a pre každú periférii taktiež jedno zadanie. Pomocou laboratórnej karty PCL-812 sa bude ovládať sústava 7-segmentových displejov s bodkou. Laboratórna karta PCL-818 bude ovládať sústavu ôsmich LED usporiadaných do kruhu a laboratórna karta PCI-1730 bude ovládať krokový motor.

Zadanie 1:

Naprogramujte program v programovacom jazyku C# v prostredí Visual Studio 2005 (alebo vyššou radou tohto vývojového prostredia), ktorý bude pomocou laboratórnej karty PCL-812 vysvecovať na pripojenom panely systémový čas počítača.

Analýza zadania:

Návrh vzhľadu aplikácie nie je tak dôležitý, nakoľko dôležitejšia je práca s perifémymi zariadeniami (panelom a laboratórnou kartou), ale napriek tomu si určíme jednoduchý vzhľad aplikácie (Obr. 7.88):

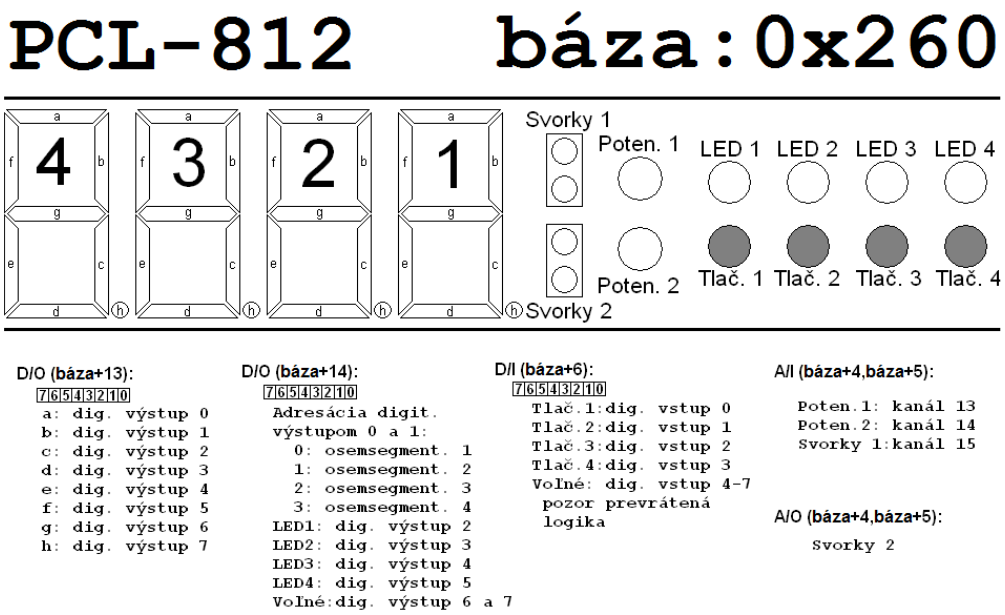


Obr. 7.88 Návrh vzhľadu aplikácie Hodiny

Textový blok je objekt *label* framework-u .NET. Okrem textového bloku je potrebný objekt *Timer*, ktorému je potrebné nastaviť vlastnosť (*properties*) *Enabled* na hodnotu *true*.

Keďže znovu sa bude pracovať so vstupno-výstupnými portami (registrami) je potrebné vložiť do projektu (programu) funkcie pre zápis do vstupno-výstupných registrov dynamickej knižnice *inpout32.dll.*

Tak ako v zadaní pre rozhranie Centronics bolo potrebné vytvoriť mapu kódov jednotlivých cifier aj v tomto zadaní je to potrebné. Pre vytvorenie takejto mapy potrebujeme vedieť pripojenie jednotlivých digitálnych vodičov ku 7-segmentovému displeju s bodkou. Ako už bolo spomenuté nad počítačom kde je pripojená laboratórna karta PCL-812 je toto pripojenie popísané (Obr. 7.89). Takže za pomoci obrázka Obr. 7.89 vypočítame jednotlivé kódy cifier. Keď je potrebné vysvietiť číslo jedna, tak do dátového registra zapíšeme 6 (00000110b – bity b a c majú byť aktívne), takýmto spôsobom sa vypočítajú (prevedú z dvojkovej do desiatkovej sústavy) ostatné cifry: cifra 0: 63, cifra 1: 6, cifra 2: 91, cifra 3: 79, cifra 4: 102, cifra 5: 109, cifra 6: 125, cifra 7: 7, cifra 8: 127, cifra 9: 111. Podobným spôsobom spravíme ďalšiu mapu kódov, kde ku každej cifre pridáme bodku. To znamená, že ku každému kódu cifry sa pripočíta 128, pretože bodku reprezentuje bit h (Obr. 7.89).



Obr. 7.89 Popis pripojenia jednotlivých vodičov vstupov a výstupov laboratórnej karty PCL-812 ku panelu

Spomínané mapy sa zadeklarujú ako polia celých čísel (*int*). Ďalšou premennou ktorú je potrebné zadeklarovať je typu *DateTime* do ktorej sa bude vkladať systémový čas. Tento systémový čas bude potrebné rozsekať na jednotlivé cifry, takže bude nutné zadeklarovať 4 celé čísla (*int*) do ktorých sa tieto cifry uložia. Pre krajší vzhľad aplikácie sa môže zadeklarovať aj premenná pre uloženie sekúnd, vďaka ktorej môže bodka oddeľujúca hodiny od minút blikať s periódou jednej sekundy. Poslednými dôležitými premennými sú pomocné premenné pre cykly (hlavný cyklus aplikácie a prázdne programové cykly).

Keďže v zadaní je potrebné mať vysvietený displej, bude nutné naprogramovať *multiplexné riadenie displeja*. Úlohou multiplexného riadenia displeja je preblikávanie jednotlivých sedem segmentových displejov a to tak, aby zobrazovaný text sa javil pre ľudské oko ako celistvý, čiže aby preblikávanie človek nevnímal. Takže periódu preblikávania treba určiť čo najnižšiu, ale aby sa dané údaje stihli zapísať do registrov a vysvietiť na displeji. Najvhodnejšia perióda bude približne 100 000 programových cyklov (experimentálne overená hodnota).

Pred začatím programovania *multiplexného riadenia displeja* sa vytvorí funkcia, ktorá bude pred každým spomínaným prebliknutím zisťovať systémový čas, tento systémový čas rozdelí do štyroch cifier (prvá cifra hodiny, druhá cifra hodiny, prvá cifra minúty, druha cifra minúty), uloží prebiehajúcu sekundu do premennej, vypíše systémový čas do okna aplikácie a počká spomínaných 100 000 programových cyklov (aby to nemuselo byť aplikované pri multiplexnom riadení displeja).

Program pre časovač najskôr spustí vyššie spomínanú funkciu, potom vynuluje register digitálneho výstupu pripojeného k LED diódam displeja, nastaví pozíciu displeja pomocou ďalšieho registra digitálneho výstupu a nastaví danú cifru na aktuálnej pozícií. Tento postup sa zopakuje štyri krát pre každú pozíciu displeja. Jediný malý rozdiel je pri druhej pozícií displeja, kde sa kontroluje či prebieha párna, alebo nepárna sekunda a podľa toho sa na displeji vysvieti, alebo nevysvieti bodka.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
5    namespace Hodiny
6    {
7         public partial class Form1 : Form
8         {
9              [DllImport("inpout32.dll", EntryPoint="Out32")]
               public static extern void Out(int add,int val);
10             int[] cifra = new int[10]
               { 63, 6, 91, 79, 102, 109, 125, 7, 127, 111};
11             int[] cifra1 = new int[10]
               {191,134,219,207,230,237,253,135,255,239};
12             DateTime cas;
13             int hod1, hod2, min1, min2, sec, i, j;
14
15             public Form1()
16             {
17                  InitializeComponent();
18             }
19
20             void aktualizuj()
21             {
22                  cas = DateTime.Now;
23                  hod1=cas.Hour / 10; hod2=cas.Hour % 10;
24                  min1=cas.Minute / 10; min2=cas.Minute % 10;
25                  sec = cas.Second;
26                  label.Text = cas.ToShortTimeString();
27                  for (j = 0; j < 100000; j++);
28             }
29
30             private void timer1_Tick(object sender,
               EventArgs e)
31             {
32                  for (i = 0; i < 1000; i++)
33                  {
34                       aktualizuj(); Out(0x26D, 0);
35                       Out(0x26E, 3); Out(0x26D, cifra[hod1]);
36                       aktualizuj(); Out(0x26D, 0);
37                       Out(0x26E, 2);
38                       if (sec % 2 == 0)
                         Out(0x26D, cifra1[hod2]);
39                       else Out(0x26D, cifra[hod2]);
40                       aktualizuj(); Out(0x26D, 0);
41                       Out(0x26E, 1); Out(0x26D, cifra[min1]);
42                       aktualizuj(); Out(0x26D, 0);
43                       Out(0x26E, 0); Out(0x26D, cifra[min2]);
44                  }
45   }    }    }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

5: Deklarácia a definovanie namespace-u tohto programu.

7: Deklarácia a definovanie hlavnej triedy programu.

9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.

10: Deklarácia poľa celých čísel (typu *int*) s názvom `cifra` reprezentujúceho mapu kódov všetkých cifier desiatkovej sústavy pre sústavu 7 – segmentových displejov na panely.

11: Deklarácia poľa celých čísel (typu *int*) s názvom `cifra1` reprezentujúceho mapu kódov všetkých cifier desiatkovej sústavy s bodkou pre sústavu 7 – segmentových displejov na panely.

12: Deklarácia premennej `cas` typu *DateTime* určenej pre vkladanie systémového času.

13: Deklarácia celých čísel (typu *int*) s názvami `hod1`, `hod2`, `min1`, `min2`, `sec`, `i` a `j`, premenná `hod1` bude slúžiť na vkladanie druhej cifry prebiehajúcej hodiny (desiatky), premenná `hod2` bude slúžiť na vkladanie prvej cifry prebiehajúcej hodiny (jednotky), premenná `min1` bude slúžiť na vkladanie druhej cifry prebiehajúcej minúty (desiatky), premenná `min2` bude slúžiť na vkladanie prvej cifry prebiehajúcej minúty (jednotky), do premennej `sec` sa budú vkladať prebiehajúce sekundy a premenné `i` a `j` budú slúžiť ako indexy cyklov s pevným počtom opakovaní.

15: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

17: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).

20: Deklarácia a definovanie funkcie `aktualizuj()`, ktorá bude slúžiť na aktualizáciu času v premennej `cas`, výpočet jednotlivých cifier času, vloženie prebiehajúcej sekundy do premennej `sec`, aktualizovanie času v okne aplikácie a pozastavenie programu na 100 000 programových cyklov.

22: Do premennej `cas` sa vloží systémový čas počítača.

23: Výpočet druhej a prvej cifry prebiehajúcej hodiny a vloženie výsledkov do premenných `hod1` a `hod2`.

24: Výpočet druhej a prvej cifry prebiehajúcej minúty a vloženie výsledkov do premenných `min1` a `min2`.

25: Vloženie prebiehajúcej (systémovej) sekundy do premennej `sec`.

26: Vypísanie systémového času (prebiehajúceho času) do okna aplikácie.

27: Cyklus, ktorý vykoná 100 000 prázdnych programových cyklov.

30: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná každú periódu časovača (*timer*-a).

32-44: Cyklus, ktorý zabezpečí, aby zobrazovanie trvalo dlhšie ako 100ms (takto je nastavený časovač (*timer*)).

- 34: Spustenie funkcie `aktualizuj()` a vyprázdnenie registra digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
- 35: Nastavenie 1. pozície displeja pomocou digitálneho výstupu laboratórnej karty PCL-812 báza + EH (viď Obr. 7.89) a cifry druhej pozície prebiehajúcej hodiny (`hod1`) pomocou kódovej mapy `cifra` prostredníctvom digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
- 36: Spustenie funkcie `aktualizuj()` a vyprázdnenie registra digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).

37: Nastavenie 2. pozície displeja pomocou digitálneho výstupu laboratórnej karty PCL-812 báza + EH (viď Obr. 7.89).
38: Ak prebieha párna sekunda, tak sa nastaví cifra prvej pozície prebiehajúcej hodiny (`hod2`) pomocou kódovej mapy `cifra1` (cifra s bodkou) prostredníctvom digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
39: Ak prebieha nepárna sekunda, tak sa nastaví cifra prvej pozície prebiehajúcej hodiny (`hod2`) pomocou kódovej mapy `cifra` (cifra bez bodky) prostredníctvom digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
40: Spustenie funkcie `aktualizuj()` a vyprázdnenie registra digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
41: Nastavenie 3. pozície displeja pomocou digitálneho výstupu laboratórnej karty PCL-812 báza + EH (viď Obr. 7.89) a cifry druhej pozície prebiehajúcej minúty (`min1`) pomocou kódovej mapy `cifra` prostredníctvom digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
42: Spustenie funkcie `aktualizuj()` a vyprázdnenie registra digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).
43: Nastavenie 4. pozície displeja pomocou digitálneho výstupu laboratórnej karty PCL-812 báza + EH (viď Obr. 7.89) a cifry prvej pozície prebiehajúcej minúty (`min2`) pomocou kódovej mapy `cifra` prostredníctvom digitálneho výstupu laboratórnej karty PCL-812 báza + DH (viď Obr. 7.89).

Zadanie 2:

Naprogramujte svetelného hada v programovacom jazyku C# v prostredí Visual Studio 2005 (alebo vyššou radou tohto vývojového prostredia), ktorý bude pomocou laboratórnej karty PCL-818 vysvecovať sústavu ôsmich LED diód na panely za sebou. Počet vysvietených diód sa bude nastavovať pomocou dvoch tlačidiel na panely. Ďalšou možnosťou aplikácie bude nastavovanie rýchlosti preblikávania (rýchlosti hada) pomocou zvyšných dvoch tlačidiel na panely.

Analýza zadania:

K tomuto zadaniu nebude potrebný návrh vzhľadu aplikácie, keďže aplikácia využije jediný objekt framework-u .NET, ktorým bude časovač (*timer*) a tento objekt neovplyvňuje vzhľad aplikácie.

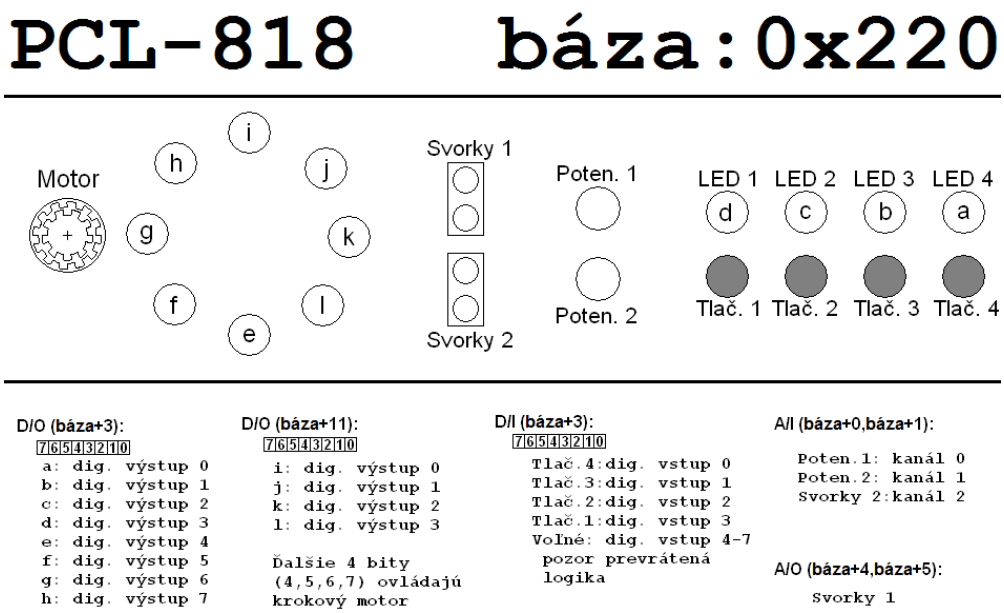
Znovu sa bude pracovať so vstupno-výstupnými portami (registrami), takže je potrebné vložiť do projektu (programu) funkcie pre zápis a čítanie zo vstupno-výstupných registrov dynamickej knižnice *inpout32.dll.*

Pre prácu s LED diódami a tlačidlami potrebujeme poznať zapojenie digitálnych vstupov a výstupov, toto zapojenie je popísané nad počítačom s laboratórnou kartou PCL-818 a tento popis je aj na ďalšej strane na obrázku Obr. 7.90.

V programe bude potrebné zadeklarovať 3 premenné: stav hada (stav vysvietených diód), krok v ktorom sa had nachádza (nadobúdajúci hodnoty od 0 po 7, keďže je 8 diód) a počet vysvietených LED diód (nadobúdajúci hodnoty od 1 po 7, keďže je 8 diód a keď nie je žiadna dióda vysvietená zadanie nemá zmysel, to isté platí aj keby boli vysvietené všetky diódy).

Funkcia časovača (*timer*-a) najprv zistí stavy stlačených tlačidiel (podľa Obr. 7.90) pomocou registra digitálnych vstupov (báza+3H), ak je niektoré stlačené, tak upraví vlastnosti hada (zvýši alebo zníži počet vysvietených LED diód, zmení periódu/interval

časovača). Tieto vlastnosti majú samozrejme intervaly v ktorom sa môžu nachádzať a preto sa pomocou podmienok hodnoty upravujú do daného intervalu.



Obr. 7.90 Popis pripojenia jednotlivých vodičov vstupov a výstupov laboratórnej karty PCL-818 ku panelu

Ovládanie hada (rozsvietených diód) sa spraví pomocou bitového posunu. Vytvorí sa 16 bitové slovo s ôsmimi platnými (*true*) bitmi (00FFH), ktoré sa posunie v pravo o toľko bitov aký je rozdiel medzi 8 a počtom rozsvietených diód (dĺžka hada) a následne sa posunie v ľavo podľa aktuálneho kroku. Potom všetky platné bity za ôsmim bitom sa pripočítajú k spodným ôsmim bitom.

Vysvetlenie spomínaného posunu bitov v predošlom odstavci:

- vytvorenie 16 bitového slova 00FFH: 0000 0000 1111 1111b,
- pri dĺžke hada 3 sa bity v slove posunú o 5 bitov: 0000 0000 0000 0111b,
- pri kroku 6 sa bity v slove posunú o 6 bitov: 0000 0001 1100 0000b,
- pripočítane vrchných bitov k spodným bitom: 000 0001 1100 0001b.

Podľa popisu zapojenia digitálnych výstupov na obrázku Obr. 7.90 sa dá všimnúť, že LED diódy sú pripojené k dvom registrom digitálneho výstupu. Pričom prvý register (báza+3) je posunutý o štyri bity a druhý register (báza + BH) má vrchné bity hada ako spodné bity registra. Pre spomenuté skutočnosti je pred zápisom stavu hada do registrov potrebný bitový posun o 4 bity. Nakoniec sa krok hada inkrementuje, ak už je krok väčší, alebo rovný ôsmim, tak sa krok vráti do nuly.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
```

```
5    namespace Had
6    {
7        public partial class Form1 : Form
8        {
9            [DllImport("inpout32.dll", EntryPoint="Out32")]
             public static extern void Out(int add, int val);
10           [DllImport("inpout32.dll", EntryPoint="Inp32")]
             public static extern byte In(int add);
11
12           UInt16 had = 0;
13           byte i = 0, poc = 3;
14
15           public Form1()
16           {
17               InitializeComponent();
18           }
19
20           private void timer1_Tick(object sender,
             EventArgs e)
21           {
22               if (In(0x223) == 254 && (poc < 7)) poc++;
23               if (In(0x223) == 253 && (poc > 1)) poc--;
24               if (In(0x223)==251 && timer1.Interval<2000)
                 timer1.Interval = timer1.Interval + 100;
25               if (In(0x223)==247 && timer1.Interval>100)
                 timer1.Interval = timer1.Interval - 100;
26
27               had = (UInt16)((0xFF >> (8 - poc)) << i);
28               had |= (UInt16)(had >> 8);
29               Out(0x223, (byte)(had << 4));
30               Out(0x22B, (byte)(had >> 4));
31               i++;
32               if (i >= 8) i = 0;
33           }
34       }
35   }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

5: Deklarácia a definovanie namespace-u tohto programu.

7: Deklarácia a definovanie hlavnej triedy programu.

9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.

10: Vloženie funkcie `Inp32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `In()`.

12: Deklarácia celého čísla typu *UInt16* s názvom `had` reprezentujúceho stav hada (stav rozsvietených LED diód), ktorého inicializačná hodnota je 0.

13: Deklarácia prirodzených čísel typu *byte* s názvami `i` a `poc`, pričom premenná `i` reprezentuje krok v ktorom sa had nachádza (inicializačná hodnota 0)

a premenná `poc` reprezentuje počet rozsvietených LED diód (inicializačná hodnota 3).

15: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

17: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).

20: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná každú periódu časovača (*timer*-a).

22: Ak je stlačené tlačidlo 4 a zároveň je dĺžka hada menšia ako 7, tak sa dĺžka hada inkrementuje.

23: Ak je stlačené tlačidlo 3 a zároveň je dĺžka hada väčšia ako 1, tak sa dĺžka hada dekrementuje.

24: Ak je stlačené tlačidlo 2 a zároveň je perióda časovača (*timer*-a) nižšia ako 2 000 ms, tak sa perióda zvýši o 100 ms.

25: Ak je stlačené tlačidlo 1 a zároveň je perióda časovača (*timer*-a) vyššia ako 100 ms, tak sa perióda zníži o 100 ms.

27: Vloženie do premennej `had` hodnotu ktorá je výsledkom nasledujúcich krokov: vytvorenie 16 bitového slova 00FFH (0000 0000 1111 1111b), posun bitov v pravo o hodnotu rozdielu (8 – `poc`), posun bitov v ľavo o hodnotu v premennej `i`.

28: Všetky platné bity v premennej `had` za ôsmim bitom sa pripočítajú k spodným ôsmim bitom.

29: Do registra digitálneho výstupu báza+3H sa zapíše stav hada (`had`) posunutý o 4 bity do ľavej strany.

30: Do registra digitálneho výstupu báza + BH sa zapíše stav hada (`had`) posunutý o 4 bity do pravej strany.

31: Inkrementovanie kroku hada (`i`).

32: Ak je krok (`i`) väčší alebo rovný ôsmim, tak sa vynuluje

Zadanie 3:

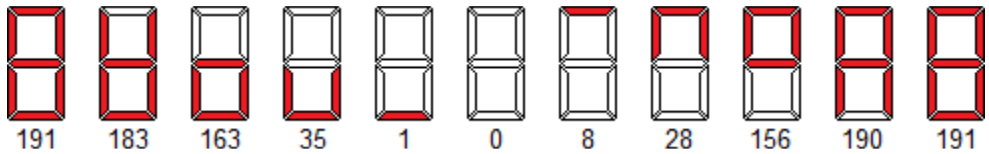
Naprogramujte program v programovacom jazyku C# v prostredí Visual Studio 2005 (alebo vyššou radou tohto vývojového prostredia), ktorý bude pomocou laboratórnej karty PCI-1730 ovládať krokový motor na panely. Pomocou tlačidiel na tomto panely sa bude nastavovať rýchlosť krokového motora. Motor bude môcť nadobúdať 11 rýchlostí, konkrétne 5 ľavo-točivých rýchlostí, státie motora a 5 pravo-točivých rýchlostí.

Analýza zadania:

Toto zadanie taktiež nepotrebuje návrh vzhľadu aplikácie, pretože využíva iba objekt časovač (*timer*) framework-u .NET. Tiež sa bude pracovať so vstupno-výstupnými portami (registrami), takže je potrebné vložiť do projektu (programu) funkcie pre zápis a čítanie zo vstupno-výstupných registrov dynamickej knižnice *inpout32.dll.*

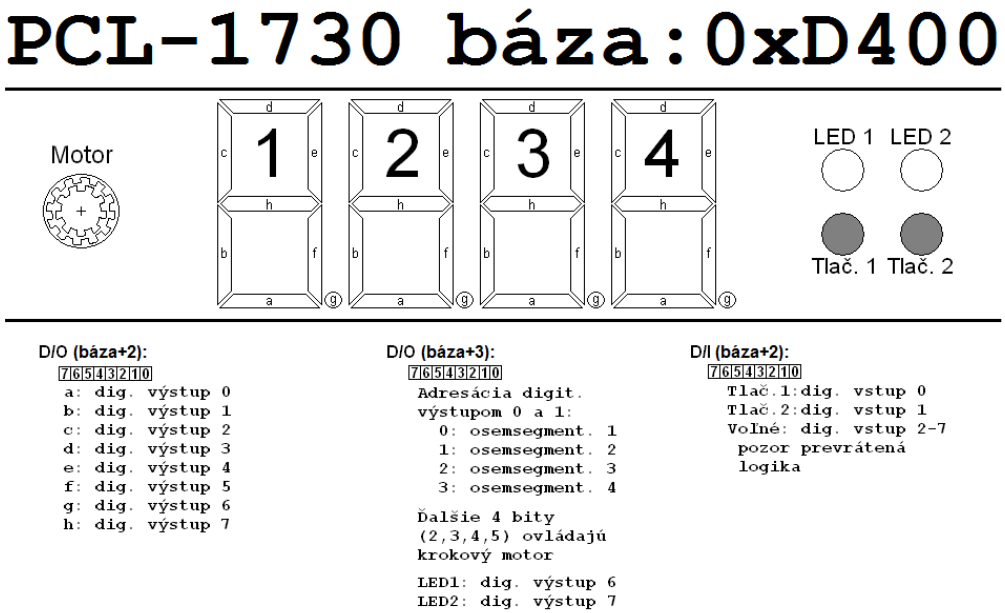
Pre prácu s motorom a tlačidlami potrebujeme poznať zapojenie digitálnych vstupov a výstupov, toto zapojenie je popísané nad počítačom s labortórnou kartou PCI-1730 a tento popis je aj na obrázku Obr. 7.92.

V tomto zadaní sa vytvoria 3 polia typu *int*, prvé pole bude mapa stavov digitálneho registra pre pohyb krokového motora, druhé pole bude mapa kódov ukazovateľa rýchlosti na displeji a tretie pole bude obsahovať časy v *tick*-och operačného systému, tieto časy budú reprezentovať dobu medzi prepnutiami cievok na krokovom motore pre jednotlivé rýchlosti. Ako ukazovateľ rýchlosti budú slúžiť dva 7-segmentové displeje (pri točení v ľavo bude svietiť ľavý displej a pri točení v pravo bude svietiť pravý displej) na ktorých bude ubúdať a pribúdať rýchlosť následne (Obr. 7.91, pod displejom je vypočítaný kód znaku v desiatkovej sústave podľa Obr. 7.92):



Obr. 7.91 Znázornenie jednotlivých rýchlosti na 7-segmentovom displeji

Po deklaráciách polí budú potrebné ďalšie deklarácie celých čísel (*int*) ako krok motora (bude nadobúdať hodnoty od 0 po 3), nový a starý stav vstupného digitálneho registra laboratórnej karty (stav tlačidiel), stupeň rýchlosti motora (bude nadobúdať hodnoty od 0 po 10) a pomocnú premennú cyklu s pevným počtom opakovaní. Poslednou nutnou premennou bude premenná slúžiaca na stopovanie času typu *Stopwatch*.



Obr. 7.92 Popis pripojenia jednotlivých vodičov vstupov a výstupov laboratórnej karty PCI-1730 ku panelu

Program časovača (*timer*-a) sa opakuje pri veľmi rýchlej frekvencií, preto sa nemôže meniť rýchlosť motora po každom načítaní stlačeného tlačidla. Preto sa tu spraví úprava, ktorá zabezpečí zmenu rýchlosti iba pri nábežnej hrane signálu stlačeného tlačidla, preto je potrebné odpamätávanie predošlého stavu tlačidla. Takže podmienka, ktorá zistí, že v predošlom stave nebolo stlačené tlačidlo a teraz je stlačené, zabezpečí detegovanie nábežnej hrany tlačidiel. V spomínanej podmienke sa zmení stupeň rýchlosti podľa stlačeného tlačidla. Následne program časovača začne stopovať čas, počas stopovania času sa nastavia registre digitálneho výstupu, ktoré zopnú cievku krokového motora a vysvietia rýchlosť motora na displeji. Inkrementuje alebo dekrementuje sa krok motora (podľa smeru motora), pričom krok motora musí byť obmedzený v intervale 0 až 3 (po kroku 3 nasleduje krok 0 a pred krokom 0 je krok 3). Po týchto krokoch sa bude čakať na okamih, kedy uplynie stopovaný čas, stopovaný čas je závislý na rýchlosti motora. Po uplynutí stopovaného času sa program časovača zopakuje.

Riešenie:

```
using System;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace Motor
{
    public partial class Form1 : Form
    {
        [DllImport("inpout32.dll", EntryPoint =
        "Out32")] public static extern void
        Out(int address, int value);
        [DllImport("inpout32.dll", EntryPoint =
        "Inp32")] public static extern int
        In(int address);
        int[] motor = new int[4] { 4, 8, 16, 32 };
        int[] rycd = new int[11]
        {3191,3183,3163,3035,3001,0,8,28,156,190,191};
        int[] rychm = new int[11]
        { 25000, 50000, 100000, 250000, 500000, 0,
        500000, 250000, 100000, 50000, 25000 };
        int k = 0, i, tl = 255, sttl = 255, rych = 0;
        Stopwatch cakat;

        public Form1()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender,
        EventArgs e)
        {
            for (i = 0; i < 10; i++)
            {
                tl = In(0xD402);
                if (tl != 255 && sttl == 255)
                {
                    if (tl == 254 && rych < 10) rych++;
                    if (tl == 253 && rych > 0)  rych--;
                }
                sttl=tl;

                cakat=Stopwatch.StartNew();
                Out(0xD403,motor[k]+(rycd[rych]/1000));
                Out(0xD402, rycd[rych] % 1000);
                if (rych < 5) k++; if (rych > 5) k--;
                if (k > 3) k = 0; if (k < 0) k = 3;
                while(cakat.ElapsedTicks<rychm[rych]);
                cakat.Stop();
}   }   }   }
```

Vysvetlenie:

1-4: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

6: Deklarácia a definovanie namespace-u tohto programu.

8: Deklarácia a definovanie hlavnej triedy programu.

10: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.

11: Vloženie funkcie `Inp32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `In()`.

12: Deklarácia poľa celých čísel (*int*) s názvom `motor`, ktoré reprezentuje mapu stavov krokového motora (mapu stavov zopnutých cievok).

13: Deklarácia poľa celých čísel (*int*) s názvom `rycd`, ktoré reprezentuje mapu kódov ukazovateľa rýchlosti na displeji (tisícky reprezentujú pozíciu displeja).

14: Deklarácia poľa celých čísel (*int*) s názvom `rychm`, ktoré obsahuje časy v *tick*-och operačného systému, tieto časy sú dobou medzi prepnutiami cievok na krokovom motore pre jednotlivé rýchlosti.

15: Deklarácia celých čísel (*int*) ako krok motora (`k`), pomocná premenná cyklu s pevným počtom opakovaní (`i`), aktuálny stav vstupného digitálneho registra laboratórnej karty (stav stlačených tlačidiel - `tl`), predošlý stav vstupného digitálneho registra (`sttl`) a stupeň rýchlosti motora (`rych`).

16: Deklarácia stopovacieho času typu *Stopwatch* s názvom `cakat`.

18: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

20: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).

23: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná každú periódu časovača (*timer*-a).

25-42: Cyklus, ktorý zabezpečí, aby vnútorný algoritmus cyklu trval dlhšie ako 1 ms (takto je nastavený časovač (*timer*)).

- 27: Zistenie stavu vstupného digitálneho registra (stlačených tlačidiel) a vloženie tohto stavu do premennej `tl`.
- 28-32: Ak je stlačené tlačidlo a zároveň v predchádzajúcom stave tlačidlo stlačené nebolo, tak (zabezpečenie detegovania nábežnej hrany):
  - 30: Ak je stlačené prvé tlačidlo a zároveň stupeň rýchlosti je nižší ako 10, tak sa stupeň rýchlosti inkrementuje.
  - 31: Ak je stlačené druhé tlačidlo a zároveň stupeň rýchlosti je vyšší ako 0, tak sa stupeň rýchlosti dekrementuje.
- 33: Odpamätanie stavu vstupného digitálneho registra (`tl`) ako predošlého stavu (`sttl`), keďže pri ďalšom porovnaní to už bude predošlí stav.
- 35: Začatie stopovania času pomocou premennej `cakat`.
- 36-37: Nastavenie výstupných digitálnych registrov pre chod motora a vysvietenie displeja na panely.
- 38: Ak je stupeň rýchlosti nižší ako 5, tak sa krok motora inkrementuje a ak je stupeň rýchlosti vyšší ako 5, tak sa krok motora dekrementuje, takže pri stupni rýchlosti 5 sa nič nestane a motor bude stáť.
- 39: Ak je krok motora vyšší ako 3, tak sa krok motora vráti do nuly a ak je krok motora nižší ako 0, tak sa krok motora nastaví na tretí krok.
- 40: Prázdny cyklus s podmienkou (*while*), ktorý čaká na uplynutie definovaného času pre konkrétny stupeň rýchlosti (Poznámka: 10 000 tikov operačného systému trvá 1 ms).
- 41: Skončenie stopovania času pomocou premennej `cakat`.

7.5 Vstupno-výstupné obvody

Táto podkapitola sa venuje trom vstupno-výstupným obvodom, konkrétne čítaču-časovaču (Intel 8253 a Intel 8254), aproximačnému analógovo-digitálnemu prevodníku a digitálno-analógovému prevodníku.

7.5.1 Čítač - časovač

Keďže obvod Intel 8253 je súčasťou počítačov, tak najskôr sa popíše toto začlenenie a potom sa popíšu ďalšie možnosti alternatívneho využitia. Obvod 8253 je programovateľný čítač-časovač, ktorý ma 3 šestnásťbitové kanály. Obvod 8254 je nadmnožinou obvodu 8253. Využitie kanálov časovača je pri PC nasledovný (Tab. 7.14):

Tab. 7.14 Využitie kanálov časovača počítača

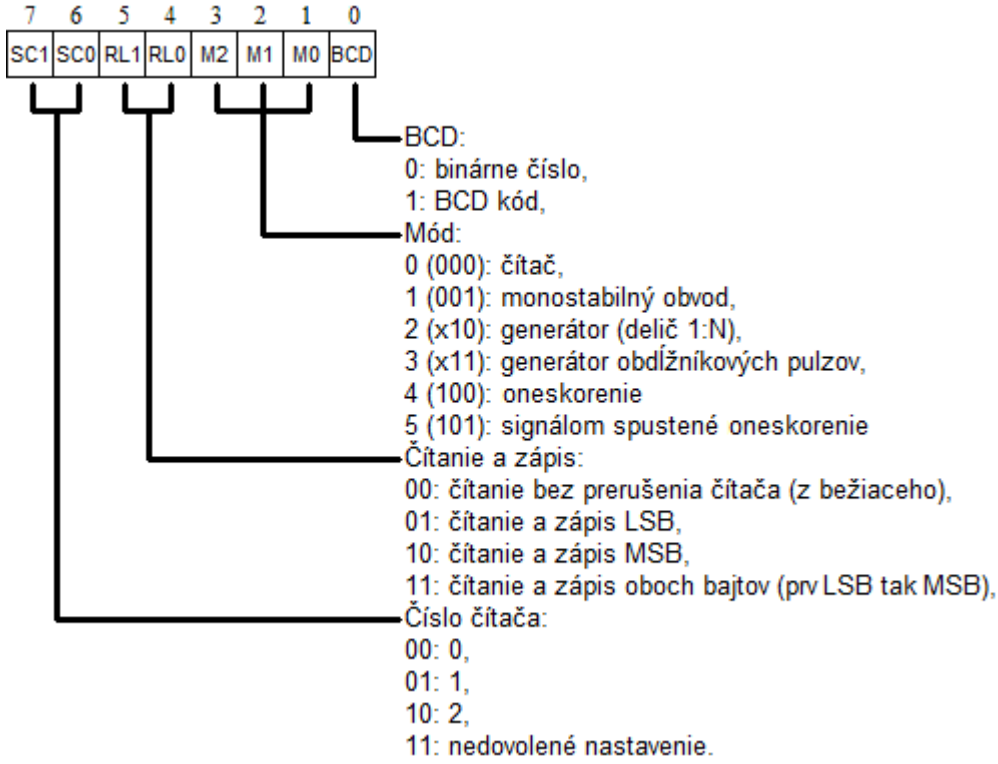
Kanál	Adresa	Popis využitia
0	040H	*Systémové hodiny* – 18,2 prerušení za sekundu (18,2 Hz), ošetrené obsluhou prerušenia INT 08H na vstupe je 1 193 180 Hz (1,2MHz). Táto frekvencia je po spustení počítača delená 65 536 (register je nastavený na plné 16-bitové slovo). Za normálnych okolností pracuje v móde 3 (generátor obdĺžnikových pulzov). Výstup sa používaj aj ku časovaniu diskových operácií, takže ak je nutná práca s diskom je potrebné ponechať v registri hodnotu 65 536.
1	041H	*Časovanie dynamickej pamäte* – Tento kanál sa nesmie využívať a najmä prestavovať, pretože v okamihu spadne počítač (výpadok pamäte a následná havária počítača). Kanál časuje plnenie dynamických pamätí.
2	042H	*Generovanie zvuku* – Výstup tohto kanálu je pripojený na reproduktor a dá sa využiť na generovanie zvukov (pokiaľ je reproduktor v počítači), alebo k časovaniu. Ak sa využije tento kanál k časovaniu, tak sa odporúča zakázať výstupu prístup k reproduktoru (aby počas časovania nevydával nepríjemné zvuky). Bitom 0 na porte 061H sa ovláda signál GATE tohto kanálu a to takto: 0 (*false*) povolenie signálu a 1 (*true*) zakázanie signálu.
Riadiace slovo	043H	*Riadiaci register časovača* – Postup pri programovaní časovača je nasledovný: • zápis riadiaceho slova na adresu 43H (viď Obr. 7.93), • zápis alebo čítanie jednej alebo dvoch časových konštánt kanálov 0 až 2, pozor je absolútne nutné zapísať alebo prečítať správny počet bajtov podľa nastaveného riadiaceho slova, inak obvod prestane poslúchať.

Inicializácia každého čítača spočíva v postupnom zápise módu konkrétneho registra (043H) a zápise jedného poprípade dvoch bajtov predvoľby (podľa bitov RL1 a RL0 - viď Obr. 7.93) na jednu z adries 040H 041H, alebo 042H (podľa nastavenia bitov SC1 a SC0 - viď Obr. 7.93).

Čítanie časovača je možné dvoma spôsobmi. Prvý spôsob je možné aplikovať jednoducho čítaním z čítača adresovaného bitmi SC1 a SC0 (viď Obr. 7.93). V tomto prípade je však nutné prerušiť čítanie buď vonkajšou logikou (zastaviť CLK), alebo hardvérovým vstupom GATE. Je nutné prečítať 1 alebo 2 bajty podľa toho, ako sú nastavené bity RL1 a RL0. Druhou možnosťou je čítanie čítača za chodu pomocou povelu spätného čítania.

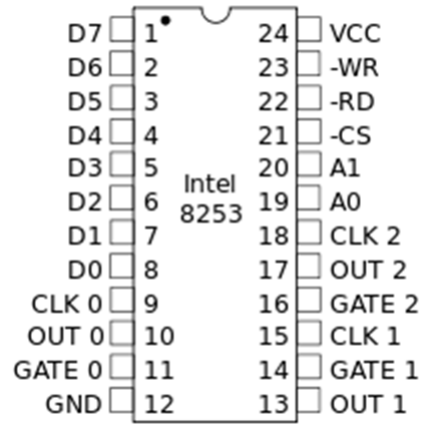
Povel spätného čítania: bitmi SC0 a SC1 sa nastaví čítač, RL1 s RL0 sa nastavia do nuly a na ostatných bitoch nezáleží.

Formát riadiaceho slova obvodov Intel 8253 a Intel 8254 je na obrázku Obr. 7.93:



Obr. 7.93 Riadiace slovo čítača-časovača

Obvod Intel 8253 je na obrázku Obr. 7.94 obvod Intel 8254 má rovnaký počet výstupov, líši sa v prehodených kontaktoch 13 a 14 a možnosťou zapojenia vyššej frekvencie na vstupe CLK.



Obr. 7.94 Obvod Intel 8253

V tabuľke Tab. 7.15 je popis všetkých kontaktov (pinov) z obrázka Obr. 7.94:

Tab. 7.15 Popis kontaktov (pinov) obvodu Intel 8253

Kontakt (pin)	Signál	Popis
1 až 8	D0 až D7	*Data* – dátové piny pre čítanie a zápis do vnútorných registrov.
9	CLK 0	*Clock* – vstupný signál hodín 0. kanálu čítača/časovača.
10	OUT 0	*Out* – výstupný signál 0. kanálu čítača/časovača.
11	GATE 0	*Gate* – hradlovací vstup 0. kanálu čítača časovača.
12	GND	*Ground* - uzemnenie obvodu.
13	OUT 1	*Out* – výstupný signál 1. kanálu čítača/časovača.
14	GATE 1	*Gate* – hradlovací vstup 1. kanálu čítača časovača.
15	CLK 1	*Clock* – vstupný signál hodín 1. kanálu čítača/časovača.
16	GATE 2	*Gate* – hradlovací vstup 2. kanálu čítača časovača.
17	OUT 2	*Out* – výstupný signál 2. kanálu čítača/časovača.
18	CLK 2	*Clock* – vstupný signál hodín 2. kanálu čítača/časovača.
19 a 20	A0 a A1	*Address* – adresné piny kanálu (výber kanálu).
21	CS	*Chip select* – výber obvodu.
22	RD	*Read* – čítanie z obvodu.
23	WR	*Write* – zápis do obvodu.
24	VCC	*Power supply* – Napájanie +5 V

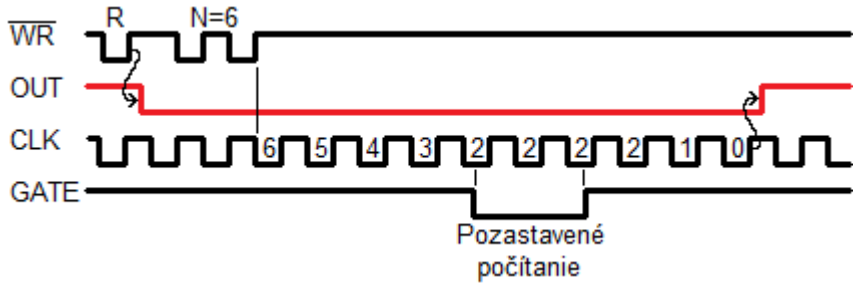
Dnes je tento obvod súčasťou *Southbridge*, ale adresovanie zostalo nezmenené. Adresovanie vstupných a výstupných signálov obvodu čítača-časovača:

Výstup nultého kanálu (OUT 0) je pripojený ku prerušovaciemu obvodu pomocou IRQ 0, takže každý vygenerovaný výstup vyvolá hardvérové prerušenie najvyššej priority. Nakoľko nultý kanál využívajú systémové hodiny a časovanie diskových operácií, tak hradlovací vstupný signál tohto kanálu (GATE 0) sa nemôže ovládať. Preto sa pri 0. kanály čítača/časovača nedá využiť 1. a 5. mód (módy sú vysvetlené nižšie). Výstup prvého kanálu (OUT 1) využíva dynamická pamäť, preto sa ku tomuto výstupu neodporúča pripájať a čítať ho. Z toho istého dôvodu je zakázané meniť aj hradlovací vstupný signál prvého kanálu (GATE 1). Výstup druhého kanálu (OUT 2) je vyvedený na vstupno-výstupnej adrese 062H ako 5. bit. Hradlovací vstup druhého kanálu (GATE 2) je na 0. bit na vstupno-výstupnej adrese 061H. Signály CLK sú priamo prepojené s generátorom hodinového signálu.

Nastavením čítačov (kanálov čítačov/časovačov) sa nastavujú časy, ktoré sa budú odpočítavať (počítať) s frekvenciou napojených hodín k vstupu CLK (pri počítačoch je to frekvencia 1 193 180 Hz). Závislosť výstupu (signál OUT) od spomínaného počítania je popísaný nižšie v jednotlivých módoch (režimoch) čítača/časovača.

Mód 0 – Čítač:

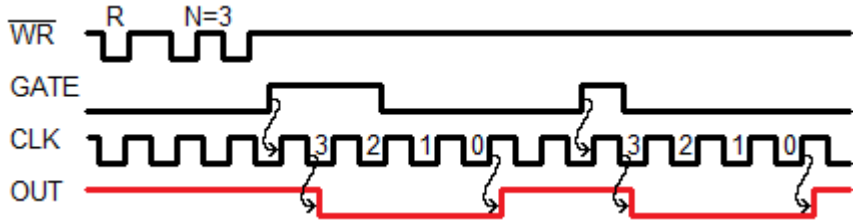
Výstup OUT od okamžiku zápisu riadiaceho slova sa prepne do 0. Po zápise hodnôt čítačov začne časovač počítať po dosiahnutí nuly sa nastaví výstup do 1. V jednotke zotrvá až kým nepríde nové riadiace slovo. Keď je GATE v 0, tak znemožňuje počítanie, ale nemá vplyv na výstup (OUT). Priebeh výstupného a vstupných signálov pri móde 0 je na obrázku Obr. 7.95.



Obr. 7.95 Priebeh signálov čítača/časovača pri móde 0

Mód 1 – Monostabilný obvod:

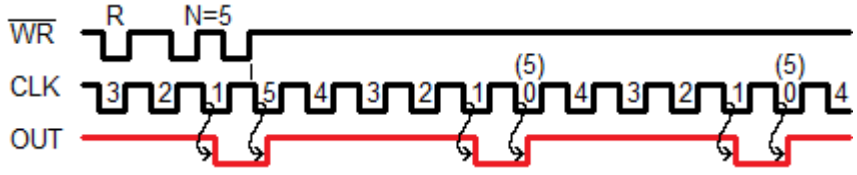
Signál OUT je na začiatku stále v jednotke (kým nebeží časovač). Nábežná hrana signálu GATE spúšťa časovač, pričom prvá spádová hrana hodín vypne signál OUT (0 - *false*) a počas celej doby počítania času je signál OUT v nule. Pri poslednej spádovej hrane hodín (po dopočítaní) sa signál OUT znovu zapne (1 - *true*). Každá nábežná hrana signálu GATE spúšťa počítanie od začiatku. Na obrázku Obr. 7.96 je znázornený priebeh výstupného a vstupných signálov pri využití prvého módu čítača/časovača.



Obr. 7.96 Priebeh signálov čítača/časovača pri móde 1

Mód 2 – Generátor:

Výstup OUT je v nule iba jeden takt hodín (CLK), inač je stále v jednotke. Interval medzi dvoma pulzmi signálu OUT je číslo uložené do čítača (*N*). Pokiaľ behom počítania časovača sa prepíše hodnota v čítači, tak to prebiehajúci pulz neovplyvní, až ten nasledujúci. Signál GATE sa dá v tomto móde použiť pre synchronizáciu čítača. Keď je v tomto móde signál GATE v nule signál OUT je v jednotke a po prepnutí signálu GATE do jednotky sa počítanie začne od začiatku. Obrázok Obr. 7.97 znázorňuje priebeh výstupného a vstupných signálov čítača/časovača pri využití druhého módu.

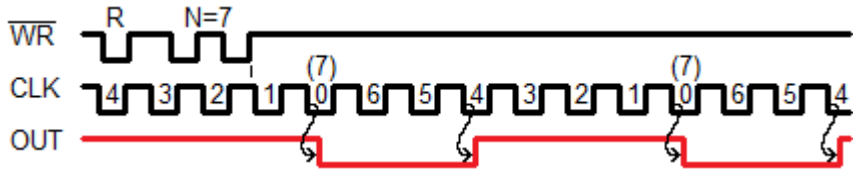


Obr. 7.97 Priebeh signálov čítača/časovača pri móde 2

Mód 3 – Generátor obdĺžnikových pulzov:

Generátor obdĺžnikových pulzov je podobný 2. módu. Výstup je v stave 0 po dobu *N / 2* a v stave 1 od doby *(N+1) / 2* hodinových pulzov (*N* – číslo uložené v čítači). V tomto prípade platí tiež to, čo pri druhom móde, keď prepíšeme *N*, tak sa to prejaví

až pri nasledujúcom pulze a signál GATE sa dá taktiež využiť na synchronizáciu, tak ako to bolo popísané vyššie (v druhom móde). Na obrázku Obr. 7.98 je priebeh výstupného a vstupných signálov pri využití tretieho módu čítača/časovača, pre tento príklad sa schválne vybralo nepárne nastavenie čítača (N = 7), aby bolo viditeľne akým spôsobom sa v tomto prípade rozdeľuje výstupný signál OUT (3 + 4, pri nepárnom nastavení čítača je stále prvý impulz signálu kratší, pri párnom sú samozrejme impulzy rovnako dlhé).



Obr. 7.98 Priebeh signálov čítača/časovača pri móde 3

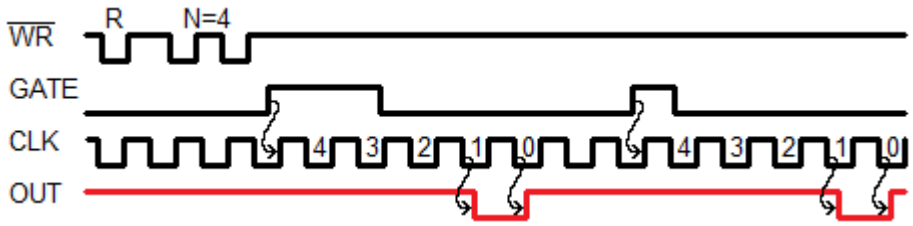
Mód 4 – Oneskorenie:

Výstup OUT je jedna a po zadaní N sa začne počítať po danom počte impulzov hodín (CLK) sa na jednu periódu prepne signál OUT do nuly, potom sa hneď vráti do jednotky. Ak sa počas počítania zapíše nová hodnota N, tak sa začne nové počítanie hneď po dokončení prebiehajúceho počítania (v prípade že sa počas jedného počítania prepíše hodnota N viac krát platí ta najnovšia hodnota – to platí pre všetky módy). Signál GATE nemení obsah čítača, ale zastaví počítanie po dobu, dokiaľ je v úrovni 0 (takže signál GATE v nule pozastaví časovač na dobu pokiaľ sa neprepne späť do jednotky). Obrázok Obr. 7.99 popisuje priebeh signálov WR, CLK a OUT v štvrtom móde čítača/časovača.

Obr. 7.99 Priebeh signálov čítača/časovača pri móde 4

Mód 5 – Signálom spustené oneskorenie:

Tento mód je kombináciou prvého a štvrtého módu. Počítanie začína od začiatku pri každej nábežnej hrane signálu GATE a výstupný signál OUT sa prepne do nuly na konci počítania po dobu jednej periódy hodín (CLK). Priebeh spomínaných signálov je znázornený na nasledujúcom obrázku Obr. 7.100:

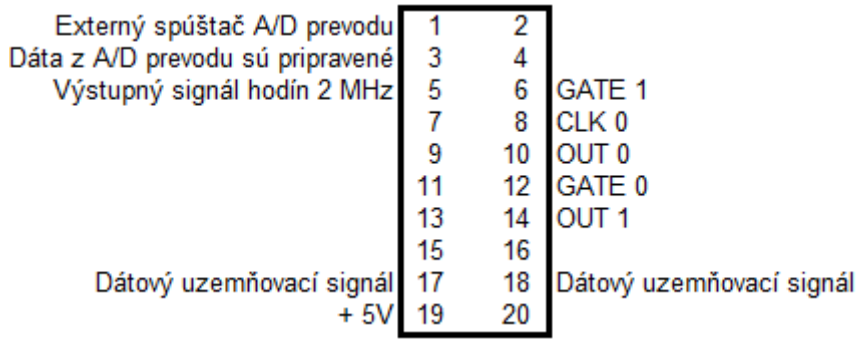


Obr. 7.100 Priebeh signálov čítača/časovača pri móde 5

Laboratórne karty využívajú čítač/časovač najmä ku synchrónnemu načítavaniu analógového vstupu. Pomocou signálu OUT sa spustí A/D prevod (následne A/D prevod trvá

30 µs pri PCL-812, alebo 5 µs pri PCL-818), pričom pre synchronizáciu je najvýhodnejší druhý a tretí mód čítača/časovača, pretože pri synchronizácií je potrebné aby medzi prevodmi uplynul rovnako dlhý čas. Takáto synchronizácia je veľmi výhodná pri regulovaní systému, kde sa pomocou čítača/časovača nastaví požadovaná vzorkovacia perióda (frekvencia) a táto perióda sa môže využiť napríklad pri výpočte akčného zásahu pomocou PSD algoritmu riadenia.

Okrem spomínaného využitia čítača/časovača ako spúšťača A/D prevodu sa pri laboratórnych kartách môže využiť čítač/časovač aj na nejaký typ vlastnej aplikácie. Pri vlastnej aplikácií treba prihliadať nato, že signály GATE a OUT sú externými signálmi laboratórnej karty (vyvedené na konektore). Preto je vhodné signál GATE napojiť napríklad na nejaký digitálny výstup laboratórnej karty, alebo na externé uzemňovacie tlačidlo (pri pomalšom systéme). Signál OUT sa napojí na niektorý digitálny vstup, alebo na LED diódu s malým odberom elektrickej energie. Laboratórna karta vie využiť aj externý zdroj hodín (CLK) pri 0. kanály čítača/časovača, ktorý sa napojí na konkrétny pin laboratórnej karty. Na obrázku Obr. 7.101 je schematicky zakreslený 5. konektor laboratórnej karty PCL-812 so spomenutými signálmi.



Obr. 7.101 Piaty konektor laboratórnej karty PCL-812

7.5.2 Programovanie čítača-časovača

V tejto podkapitole budú dve zadania, obe sa budú venovať čítaču-časovaču integrovaného do počítača. V prvom zadaní sa v programe použije 0. kanál čítača-časovača Intel 8253 a bude programovaný pomocou programovacieho jazyka C v operačnom systéme MS-DOS. Druhé zadanie využije 2. kanál čítača/časovača pod operačným systémom Windows XP a bude naprogramované programovacím jazykom C#.

Zadanie pre čítač/časovač integrovaný na laboratórnych kartách bude až v podkapitole 7.5.6, pretože bez znalostí A/D prevodu, by zadanie bolo skoro totožné so zadaniami pre čítač/časovač integrovaný v počítači, nakoľko je to ten istý obvod. Preto sú v tejto podkapitole iba dve zadania pre čítač/časovač počítača.

Zadanie 1:

Naprogramujte program stopky, ktorý po stlačení klávesa *Enter* bude stopovať čas s presnosťou na milisekundy po ďalšie stlačenie ľubovoľného klávesa. Program naprogramujte pomocou 0. kanálu čítača/časovača a hardvérového prerušenia ktoré generuje. Program naprogramujte v programovacom jazyku C pod operačným systémom MS-DOS.

Analýza zadania:

Samotné zadanie si vyžaduje prácu s prerušeniami, takže je potrebné prilinkovať knižnicu *dos.h*, ktorá definuje typ *inerrupt*.

Nakoľko je potrebné stopovať čas je nutné zadeklarovať premennú v ktorej sa tento čas bude inkrementovať. Nakoľko typ *int* môže nadobúdať maximálnu hodnotu 32 768, čo by predstavovalo v tisícinách sekúnd iba 32 sekúnd, tak je potrebné túto premennú zadeklarovať ako typ *long*. Zadeklarovať je potrebné aj premennú, do ktorej sa vloží algoritmus pôvodnej obsluhy prerušenia od čítača/časovača (typ *interrupt*).

Ďalej je vhodné navrhnúť algoritmus novej obsluhy prerušenia. Tento návrh je úplne jednoduchý, zavolá sa pôvodná obsluha prerušenia, ktorá bude uložená vo vyššie spomínanej premennej a potom sa bude už len inkrementovať počítadlo.

V hlavnom programe sa nastaví 0. kanál čítača/časovača do módu 3, pričom sa čítač bude nastavovať v binárnych číslach a hneď po dolnom bajte sa zadá vrchný bajt, tomu zodpovedá nastavenie 0011 0110b, čo je 36H (viď Obr. 7.93). Následne sa vloží vyrátaný dolný a horný bajt. Výpočet je jednoduchý frekvencia sa vynásobí presnosťou:

$$1\ 193\ 180\ s^{-1} \,.\, 0{,}001\ s = 1\ 193\ jednotiek \qquad (7.14),$$

a potom rozdelíme výsledok na vrchný a spodný bajt:

$$1\ 193 \div 256 = 4\ zv.\ 169\ (4H\ zv.\ A9H) \qquad (7.15),$$

Do premennej určenej k vloženiu starej obsluhy prerušenia čítača/časovača sa táto obsluha vloží pomocou funkcie `getvect()`, potom pomocou funkcie `setvect()` sa nahrá nová obsluha prerušenia. Obe tieto funkcie sú definované v knižnici *dos.h*. Následne sa vyčistí obrazovka a vypíše sprievodný text, ktorým bude užívateľ oboznámený náplňou programu a klávesom ktorým sa stopovanie začne a môže ukončiť. Program počká na stlačenie klávesa *Enter*, vynuluje počítadlo a počas čakania na stlačenie ľubovoľného klávesa pomocou cyklu s podmienkou bude program vypisovať uplynutý čas. Nakoniec sa ošetrí prípad vypnutia programu po ukončení cyklu a do pamäte sa vloží späť stará obsluha prerušenia.

Riešenie:

```
1    #include<stdio.h>
2    #include<conio.h>
3    #include<dos.h>
4
5    long poc=0;
6    void interrupt (*old_v)();
7
8    void interrupt obsluha()
9    {
10        (*old_v)();
11        poc++;
12   }
13
```

```
14  void main()
15  {
16       outportb(0x43,0x36);
17       outportb(0x40,0xA9);
18       outportb(0x40,0x04);
19       old_v = getvect(0x08);
20       setvect(0x08, obsluha);
21       clrscr();
22       printf("Začatie a skončenie stopovania: Enter");
23       getchar();
24       poc=0;
25       while(!kbhit())
26       {
27            printf("\r %.3f sekúnd", (float)poc/1000);
28       }
29       getchar();
30       getchar();
31       setvect(0x8,old_v);
32  }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých knižníc k programu.

5: Deklarácia premennej `poc` typu *long* (celé číslo) slúžiacej ako počítadlo.

6: Deklarácia premennej `old_v()` typu *interrupt* určenej pre starú obsluhu prerušenia čítača/časovača.

8: Deklarácia a definovanie novej obsluhy prerušenia: `obsluha()`.

10: Spustenie starej obsluhy prerušenia pomocou premennej `old_v()`.

11: Inkrementácia premennej `poc` slúžiacej ako počítadla uplynutého času.

14: Deklarácia a definovanie hlavného programu.

16: Nastavenie riadiaceho registra čítača/časovača: do čítača sa vloží binárne číslo, čítač/časovač bude v 3. móde, postupne sa zadá dolný a potom horný bajt a toto nastavenie sa aplikuje na nultý kanál čítača/časovača (viď Obr. 7.93).

17: Nastavenie dolného bajtu čítača.

18: Nastavenie horného bajtu čítača.

19: Uloženie starej obsluhy prerušenia čítača/časovača do premennej `old_v`.

20: Vloženie novej obsluhy prerušenia čítača/časovača z premennej `obsluha`.

21: Vyčistenie obrazovky.

22: Vypísanie textu: „Začatie a skončenie stopovania: Enter“, na obrazovku.

23: Čakanie na stlačenie klávesa *Enter* (prijatie znaku s potvrdením).

24: Vynulovanie počítadla (premennej `poc`).

25-28: Cyklus s podmienkou, ktorá sa opakuje kým sa nestlačí ľubovoľný kláves.

27: Vypísanie aktuálneho stavu počítadla v sekundách.

29-30: Dvojité čakanie na stlačenie klávesa *Enter* (prijatie znaku s potvrdením), aby program neskončil, ak by ľubovoľným klávesom, ktorý ukončil cyklus, bol kláves *Enter*.

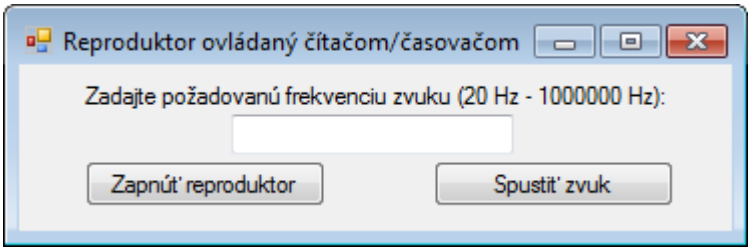
31: Vloženie starej obsluhy prerušenia čítača/časovača späť do pamäte z premennej `old_v`.

Zadanie 2:

Za použitia 2. kanálu čítača/časovača naprogramujte program, ktorý bude pomocou reproduktora v počítači generovať zvuk s frekvenciou, ktorú si užívateľ sám zvolí. Program naprogramujte v programovacom jazyku C# pod operačným systémom Windows XP (alebo vyššou radou operačného systému) vo vývojovom prostredí Visual Studio 2005 (alebo vo vyššej rade vývojového prostredia).

Analýza zadania:

Táto aplikácia si žiada nejaké vstupy z aplikácie, takže je potrebné navrhnúť vzhľad aplikácie (Obr. 7.102):



Obr. 7.102 Návrh aplikácie pre generovanie zvuku pomocou čítača/časovača

Tak ako pri každom zadaní v programovacom jazyku C# kde bolo potrebné pracovať so vstupno-výstupnými portami aj tu je potrebné prilinkovať funkcie `Out32()` a `Inp32()` z dynamickej knižnice *inpout32.dll*. V tejto aplikácií sa použijú iba dva premenné typu *int*, ktoré budú slúžiť ako hodnoty horného a dolného bajtu druhého kanála čítača.

Program prvého tlačidla za pomoci frekvencie zadanej do textového poľa vypočíta horný a dolný bajt. Hodnota čítača (*C*) sa vypočíta podielom frekvencie hodín čítača/časovača a požadovanej frekvencie zvuku (*f*):

$$C = \frac{1193180Hz}{f} \qquad (7.16),$$

Táto hodnota (*C*) sa už iba rozdelí na horný (*H*) a dolný (*D*) bajt:

$$C \div 256 = H\ zv.\ D \qquad (7.17),$$

Ak sa táto operácia nepodarí (textové pole neobsahuje hodnotu, alebo hodnotou nie je prirodzené číslo), tak sa vypíše hlásenie o chybe a program tlačidla sa ukončí. V prípade, že sa operácia podarila, program pokračuje a zistí či horný a dolný bajt nadobúda správne hodnoty. Ak nenadobúda správne hodnoty, tak sa vypíše o tom hlásenie a program tlačidla sa ukončí, inač program pokračuje ďalej. Poslednou časťou programu je nastavenie čítača/časovača. Nastaví sa riadiaci register čítača/časovača a to takto, že do čítača sa vloží binárne číslo, čítač/časovač bude v 3. móde, postupne sa zadá dolný a potom horný bajt a toto nastavenie sa aplikuje na druhý kanál čítača/časovača (viď Obr. 7.93). Po tomto nastavení sa vloží do čítača/časovača vypočítaný horný a dolný bajt.

Druhé tlačidlo je navrhnuté, aby bola možnosť softvérového vypnutia reproduktora. V prípade nepríjemného zvuku sa bude dať reproduktor okamžite vypnúť. Tlačidlo bude slúžiť ako vypínač na zapnutie a vypnutie reproduktora, takže v prvom rade sa v programe

zistí či je reproduktor zapnutý, alebo vypnutý. To sa skontroluje pomocou textu v tlačidle, ak je reproduktor zapnutý text bude „Vypnúť reproduktor“ a v opačnom prípade „Zapnúť reproduktor“. Jednoduchou podmienkou sa zistí stav reproduktora a pomocou vstupno-výstupného portu (registra) 61H sa prvým bitom zapne (alebo vypne) signál GATE a druhým bitom zapne (alebo vypne) reproduktor. V spomínanej podmienke sa ešte upraví text tlačidla tak, aby zodpovedal aktuálnemu stavu reproduktora a čítača/časovača.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
5    namespace Časovač
6    {
7         public partial class Form1 : Form
8         {
9              [DllImport("inpout32.dll", EntryPoint="Out32")]
               public static extern void Out(int add, int val);
10             [DllImport("inpout32.dll", EntryPoint="Inp32")]
               public static extern int In(int add);
11             int D, H;
12
13             public Form1()
14             {
15                  InitializeComponent();
16             }
17
18             private void button1_Click(object sender,
               EventArgs e)
19             {
20                  try
21                  {
22                   D=(1193180/Convert.ToInt16(txtB.Text))%256;
23                   H=(1193180/Convert.ToInt16(txtB.Text))/256;
24                  }
25                  catch
26                  {
27                   MessageBox.Show("Zlý formát", "Chyba");
28                   return;
29                  }
30                  if ((H > 255) || ((H==0) && (D==0)))
31                  {
32                   MessageBox.Show("Mimo rozsahu", "Chyba");
33                   return;
34                  }
35                  Out(0x43, 182);
36                  Out(0x42, D);
37                  Out(0x42, H);
38             }
39
```

```
40              private void button2_Click(object sender,
                EventArgs e)
41              {
42                   if (button2.Text == "Zapnúť reproduktor")
43                   {
44                        Out(0x61, 3);
45                        button2.Text = "Vypnúť reproduktor";
46                   }
47                   else
48                   {
49                        Out(0x61, 0);
50                        button2.Text = "Zapnúť reproduktor";
51                   }
52              }
53         }
54    }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

5: Deklarácia a definovanie namespace-u tohto programu.

7: Deklarácia a definovanie hlavnej triedy programu.

9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.

10: Vloženie funkcie `Inp32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `In()`.

11: Deklarácia celých čísel (*int*) s názvom `D` a `H`, ktoré budú nosiť hodnoty dolného a horného bajtu čítača.

13: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

15: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).

18: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla: ....

- 20-24: Vyskúšanie výpočtu horného a dolného bajtu čítača pomocou hodnoty zadanej v textovom poli.
- 25-29: Ak sa výpočet nepodarí (textové pole neobsahuje hodnotu, alebo hodnotou nie je prirodzené číslo), tak sa vypíše hlásenie o danej chybe a algoritmus sa ukončí.
- 30-34: Ak je hodnota v textovom poli mimo povoleného rozsahu, tak sa vypíše o danej skutočnosti hlásenie a algoritmus sa ukončí.
- 35: Nastavenie riadiaceho registra čítača/časovača: do čítača sa vloží binárne číslo, čítač/časovač bude v 3. móde, postupne sa zadá dolný a potom horný bajt a toto nastavenie sa aplikuje na druhý kanál čítača/časovača (viď Obr. 7.93).
- 36: Nastavenie dolného bajtu čítača.
- 37: Nastavenie horného bajtu čítača.

40: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná po stlačení tlačidla: ....

- 42-46: Ak text stlačeného tlačidla je „Zapnúť reproduktor“, tak sa reproduktor a signál GATE zapne (pomocou vstupno-výstupného portu 61H) a zmení sa text tlačidla na „Vypnúť reproduktor“.
- 47-51: Ak text stlačeného tlačidla je „Vypnúť reproduktor“, tak sa reproduktor a signál GATE vypne a zmení sa text tlačidla na „Zapnúť reproduktor“.

7.5.3 Digitálno-analógový prevodník

Digitálno-analógový prevodník je elektronické zariadenie na prevod digitálneho signálu na analógový signál. Jeho nasadenie je na výstupoch zo zdrojov digitálnej informácie do okolitého sveta, ktorý vnímame ako spojitý (analógový). Príkladom je zvuková karta, ktorá prijme digitálnu informáciu z procesora (napríklad dekódovanú MPEG pesničku) a pomocou digitálno-analógového prevodníku ju prenesie cez zosilňovač a ekvalizér do reproduktoru. Výsledok vnímame ako spojitý signál (zvuk, melódiu). Samozrejme obrovské uplatnenie má tento prevodník aj pri riadení. Regulátor (napríklad počítač) vypočíta akčný zásah a pomocou tohto prevodníka vyšle akčný zásah do systému.

D/A prevodník môžeme chápať ako dekódovací obvod, ktorý prevádza digitálne slovo pomocou vstupného referenčného prúdu alebo napätia do výstupného prúdu alebo napätia.

Digitálno-analógové prevodníky sa dajú rozdeliť podľa princípu fungovania:

- prúdový princíp,
- napäťový princíp,
- nábojový princíp.

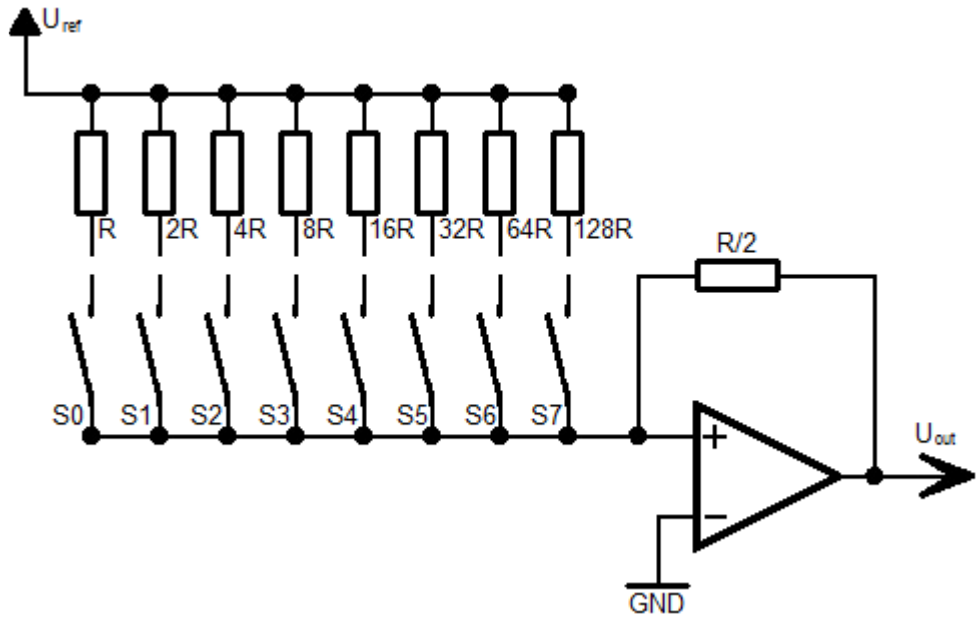
Prúdový princíp:

V tomto prípade sa konverzia dosahuje pripájaním binárne vážených interných prúdových zdrojov, ktoré sú nakoniec sčítané na operačnom zosilňovači.

Na obrázku (Obr. 7.103) je znázornený 8 bitový DAC (Digital-to-Analog Converter – D/A prevodník). Spínače S0 až S8 symbolicky reprezentujú pripájanie jednotlivých vetiev (prúdov) podľa riadiaceho slova. Pre názornosť je možné si predstaviť, že vysoká úroveň reprezentuje zopnutý vypínač a nízka rozopnutý. Veľkosť napätia na výstupe (U_{out}) potom môžeme vyjadriť vzťahom:

$$U_{out} = -I_{out} R_{out} = -U_{ref} \left(2^{-1} b_0 + 2^{-2} b_1 + \ldots + 2^{-(n+1)} b_n \right) \qquad (7.18),$$

kde R_{out} je spätnoväzobný odpor, I_{out} je výstupný prúd a b_i sú jednotlivé bity.

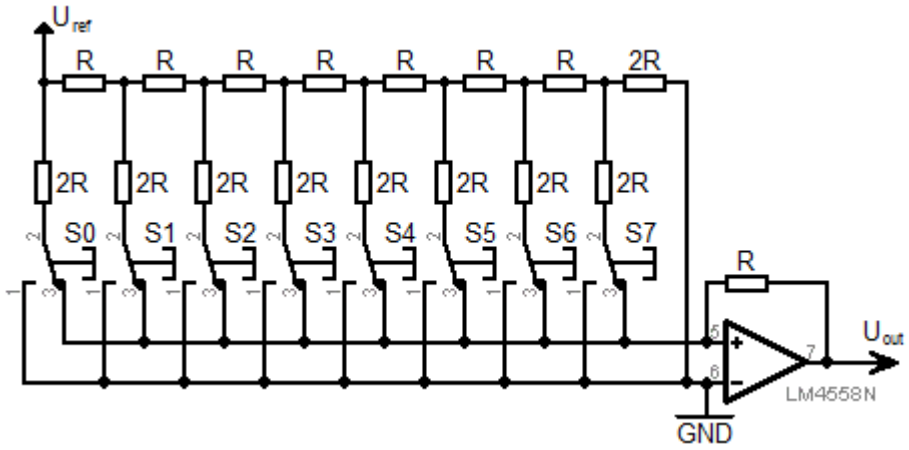


Obr. 7.103 D/A prevodník založený na prúdovom princípe

Po podrobnejšej analýze obvodu je možné vytknúť dva nedostatky:

- veľké parazitné kapacity spôsobené pripájaním jednotlivých vetiev,
- exponenciálny nárast hodnôt odporov.

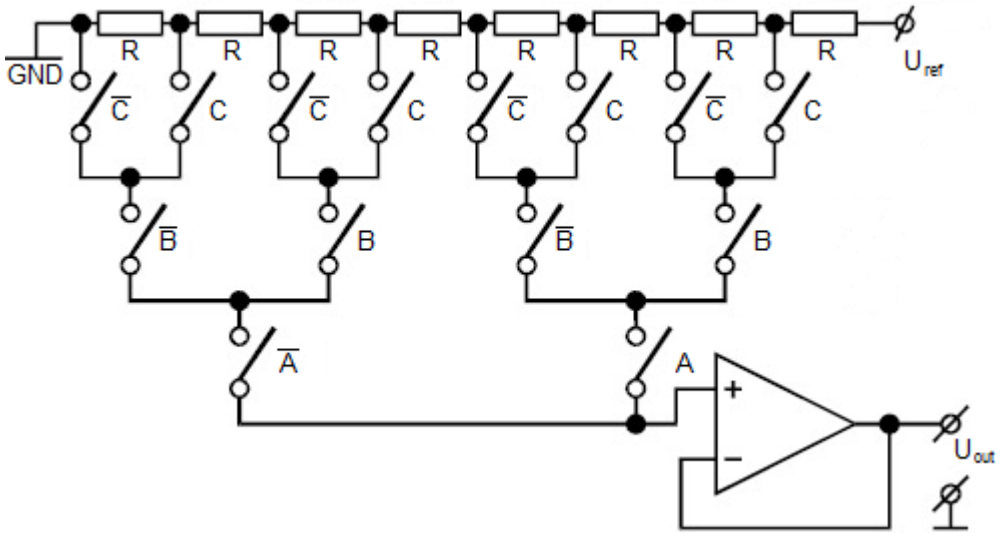
Na ďalšom obrázku (Obr. 7.104) je schéma zapojenia, kde sú tieto problémy vyriešené. Nevýhodou tohto zapojenia je zdvojnásobenie počtu odporov, čo však nie je taký veľký problém ako umiestnenie veľmi rozmanitých hodnôt do jedného integrovaného obvodu. Odpory sa tu vyskytujú už len so základnou a dvojnásobnou hodnotou. Binárne delenie prúdu je zabezpečené na každej vetve postupne na zložky v pomere ½.



Obr. 7.104 D/A prevodník založený na prúdovom princípe len s dvoma typmi odporov

Napäťový princíp:

D/A prevodníky pracujúce na napäťovom princípe používajú na vytvorenie napätia na výstupe kombináciu dielov referenčného napätia. Princíp je možné pochopiť z nasledujúceho obrázka (Obr. 7.105), kde je zobrazený troj-bitový D/A prevodník.

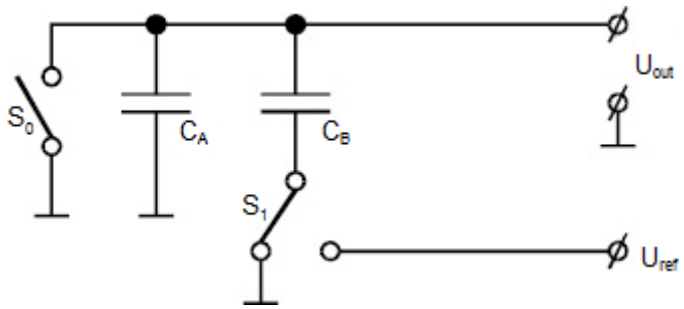


Obr. 7.105 D/A prevodník založený na napäťovom princípe

Podmienkou správnej funkcie tohto obvodu je zanedbateľná veľkosť prúdu tečúceho do sledovača. Každý bit digitálnej informácie zopína príslušný vypínač, vysoká úroveň na nultom bite zopne vypínač A nízka naopak zopne A s pruhom, takže slovo 110 zopne A, B a C s pruhom a na výstupe bude U_{ref} / 4. Nevýhodou tohto zapojenia je veľký počet ovládacích prvkov (2^{N+1}), odporov (2^N) a logických prepojení (2^N), pričom *N* je šírka digitálneho slova (šírka kódovania analógového signálu). Tieto vlastnosti ho obmedzujú len na použitie v rýchlych aplikáciách s malým rozlíšením.

Nábojový princíp:

DA prevodník pracujúci na tomto princípe vytvára napäťovú úroveň na výstupe pomocou predávania náboja sieťou kondenzátorov. Jeho činnosť je možne vysvetliť na nasledujúcom obrázku (Obr. 7.106):



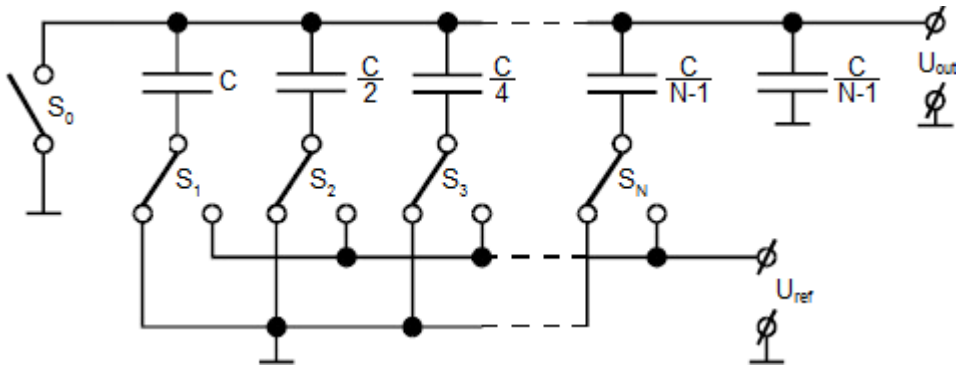
Obr. 7.106 Logika D/A prevodu založená na nábojovom princípe

Kondenzátor C_A pripojíme na zem a C_B budeme periodicky prepínať medzi referenčný napätím a zemou. Ak budú oba spínače na zemi, budú oba kondenzátory vybité (nulovaná fáza) a na výstupe (U_{out}) bude nulové napätie. Ak bude S_0 otvorený a S_1 na referenčnom napätí, bude platiť:

$$U_{out} = \frac{U_{ref} C_B}{C_A + C_B} \qquad (7.19),$$

čo je vzorkovacia fáza.

Usporiadaním kondenzátorov podobne ako pri využití prúdového princípu a binárnym vyvážením, ukazuje obrázok Obr. 7.107:



Obr. 7.107 D/A prevodník založený na nábojovom princípe

Toto zapojenie má však rovnakú chybu ako ekvivalentný odporový variant, a to je exponenciálne rastúci rozdiel najvyššej a najnižšej kapacity. Z tohto dôvodu je zapojenie ako integrovaný obvod realizovateľné maximálne do 8 bitov.

Chyby DA prevodníkov sú dvoch základných druhou:

1. Odstrániteľné chyby:
   - chyba ofsetu – možné odstrániť nastavením trimrov,
   - chyba rozsahu – možné odstrániť predradením deliča,
2. Neodstrániteľné chyby:
   - chyba spojitosti – chyba sa prejaví ako lokálny extrém na krivke výstupného napätia, ktorá ma byť v ideálnom prípade priamka
   - nelinearita – chyba prejavujúca sa ako odchýlky od skutočných výstupných hodnôt napätia od ideálnej priamky, bez funkčnej závislosti; rozlišujeme dva typy:
     - diferenciálna – chyba udávajúca rozdiel skoku hodnoty výstupného napätia od ideálneho skoku 1 LSB,
     - integrálna – udáva najväčšiu odchýlku reálnych hodnôt od ideálnej priamky, obyčajne sa vyjadruje v percentách.

Najpoužívanejší D/A prevod v praxi je založený na prúdovom princípe, ten používajú aj laboratórne karty, ktoré boli rozobraté v kapitole 7.4.

7.5.4 Programovanie digitálno-analógového prevodníka

Programovanie digitálno-analógového prevodníka je jednoduché, nakoľko stačí nastaviť digitálne výstupy obvodu a automaticky je na výstupe požadované napätie (viď Obr. 7.103 až Obr. 7.107).

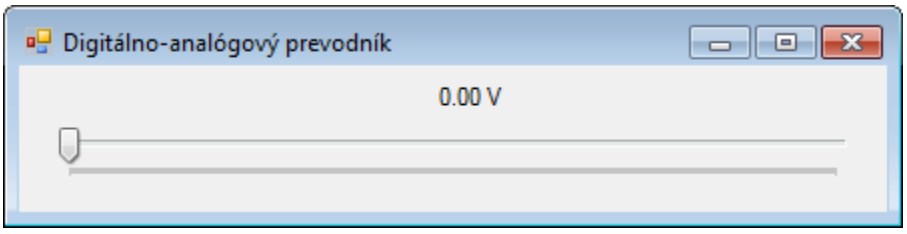
Podkapitola obsahuje dve zadania, v prvom sa nastavuje výstupné napätie podľa požiadavky užívateľa aplikácie a v druhom aplikácia generuje sínusový signál s požadovanou frekvenciou a amplitúdou.

Zadanie 1:

Naprogramujte program, ktorý pomocou D/A prevodníka na laboratórnej karte PCL-818 vygeneruje požadované napätie od 0 po 10 V. Program naprogramujte v programovacom jazyku C# pod operačným systémom Windows XP (alebo pod vyššou radou operačného systému) vo vývojovom prostredí Visual Studio 2005 (alebo vo vyššej rade vývojového prostredia).

Analýza zadania:

Návrh vzhľadu aplikácie sa prispôsobí podmienkam zadania (Obr. 7.108):



Obr. 7.108 Návrh vzhľadu aplikácie jednoduchého D/A prevodu

Pomocou posúvača (*Track Bar*) sa bude nastavovať požadované napätie, ktorého veľkosť sa zobrazí v textovom bloku (*Label*) nad týmto posúvačom. Poloha posúvača sa bude sledovať každý cyklus časovača (*Timer*) a na základe jeho polohy sa nastaví analógový výstup a hodnota v textovom bloku aplikácie. Posúvaču (*Track Bar*) je potrebné zmeniť vo vývojovom prostredí jednu vlastnosť (*properties*), tou vlastnosťou je maximálna hodnota (*maximum*), ktorá sa nastaví na 4095. Takže posúvač môže nadobúdať hodnoty od 0 po 4095, čo predstavuje celý rozsah prevodníka (12 bitov). Z toho vyplýva, že digitálnu hodnotu (inžinierske jednotky) potrebnú pre prevodník už nie je nutné vypočítať. V tomto prípade bude nutné z digitálnej hodnoty (inžinierskych jednotiek) vypočítať napätie, ktoré sa vypíše do textového bloku (*Label*). Poslednú zmenu pri návrhu vzhľadu je pridanie spomínaného časovača (*Timer*), ktorému je potrebné zmeniť vlastnosť *Enabled* na *true*.

V deklaračnej časti programu je nutné prilinkovať funkciu `Out32()` z dynamickej knižnice *inpout32.dll* do programu pomocou funkcie `DllImport()` a zadeklarovať premenné pre horný bajt, dolný bajt a výpočet napätia.

Program, ktorý sa vykoná pri udalosti (*Event*) tiku časovača (*timer tick*) prepočíta z hodnoty posúvača (*Track Bar*) veľkosti horného a dolného bajtu podľa obrázkov Obr. 7.67 a Obr. 7.68 a zapíše ich do registrov. Potom sa vypočíta hodnota napätia na analógovom výstupe a vypíše sa do textového bloku (*Label*). Výpočet je jednoduchý, nakoľko rozsahu 0 až 10 V zodpovedá digitálny rozsah od 0 do 4095 inžinierskych jednotiek, tak postačuje digitálnu hodnotu vydeliť číslom 409,5.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
5    namespace DAPrevodník
6    {
7         public partial class Form1 : Form
8         {
9              [DllImport("inpout32.dll", EntryPoint="Out32")]
               public static extern void Out(int add, int val);
10             int H, D;
11             double U;
12
13             public Form1()
14             {
15                  InitializeComponent();
16             }
17             private void timer1_Tick(object sender,
               EventArgs e)
18             {
19                  H = trackBar1.Value / 16;
20                  D = 16 * (trackBar1.Value % 16);
21                  Out(0x225, H);
22                  Out(0x224, D);
23                  U = trackBar1.Value / 409.5;
24                  label1.Text = U.ToString("0.00 V");
25             }
26        }
27   }
```

Vysvetlenie:

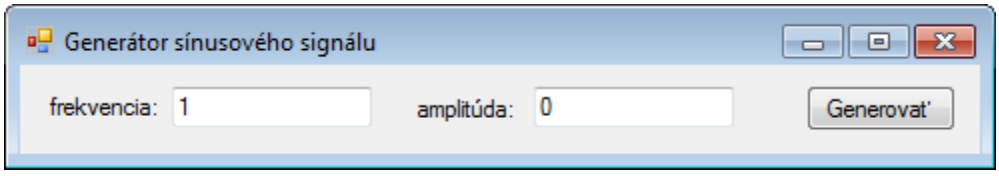
1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.
5: Deklarácia a definovanie namespace-u tohto programu.
7: Deklarácia a definovanie hlavnej triedy programu.
9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.
10: Deklarácia premenných `H` a `D` typu *int* (celé čísla), ktoré budú slúžiť na výpočet a zápis horného a dolného bajtu registra D/A prevodníka.
11: Deklarácia premennej `U` typu *double* (reálne číslo), táto premenná bude slúžiť na výpočet napätia z inžinierskych jednotiek (digitálnej hodnoty).
13: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.
15: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).
17: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná každú periódu (tik) časovača (*timer*-a).
19: Výpočet horného bajtu z digitálnej hodnoty (*Track Bar*-u), keďže horný bajt využíva celých 8 bitov, tak digitálnu hodnotu treba vydeliť číslom 16 (4095/16 = 255 zv. 15) a výsledok je hodnota horného bajtu (viď Obr. 7.68).
20: Výpočet dolného bajtu z digitálnej hodnoty (*Track Bar*-u), nakoľko dolný bajt využíva iba 4 bity, tak digitálnu hodnotu treba vydeliť 16 a výsledkom by mal byť zvyšok po tomto delení, ale dolný bajt je ešte o 4 bity posunutý, takže tento zvyšok treba ešte vynásobiť číslom 16 (viď Obr. 7.67).
21-22: Zápis horného a dolného bajtu do registrov D/A prevodníka laboratórnej karty.
23: Výpočet napätia z digitálnej hodnoty (*Track Bar*-u), hodnotu stačí vydeliť číslom 409,5, keďže rozsah 0 až 10 V zodpovedá digitálnemu rozsahu 0 až 4095.
24: Výpis vypočítaného napätia do textového bloku (*Label*) aplikácie.

Zadanie 2:

Naprogramujte program ktorý bude generovať sínusový signál pomocou D/A prevodníka laboratórnej karty PCL-812. V aplikácií si bude môcť užívateľ nastaviť frekvenciu a amplitúdu signálu. Program naprogramujte v programovacom jazyku C# pod operačným systémom Windows XP (alebo pod vyššou radou operačného systému) vo vývojovom prostredí Visual Studio 2005 (alebo vo vyššej rade vývojového prostredia).

Analýza zadania:

Vzhľad aplikácie môže vyzerať takto (Obr. 7.109):



Obr. 7.109 Návrh vzhľadu aplikácie generátora sínusového signálu

Hlavným objektmi obrazovky (okna) budú dve textové polia (*Text Box*) a jedno tlačidlo (*Button*). V tomto program sa ukáže práca s vláknami (vykonávacími tokmi).

Deklaračná časť programu bude obsahovať prilinkovanie funkcie `Out32()` z knižnice *inpout32.dll*, deklaráciu bázovej adresy laboratórnej karty, deklaráciu vlákna s konštruktorom, deklaráciu premenných na vkladanie frekvencie, amplitúdy, vrchného a spodného bajtu registra D/A prevodníka. V inicializačenj časti programu sa spustí vlákno.

Algoritmus vlákna bude v nekonečnom cykle (kým sa vlákno nevypne) generovať sínusový signál. V nekonečnom cykle vlákna sa spustí cyklus s pevným počtom opakovaní. Týchto opakovaní bude 628, čo bude predstavovať stotiny radiánov (6,28 rad je 360°). V tomto vnútornom cykle sa začne stopovať čas v tikoch systémových hodín operačného systému (1 tik = 100ns). Potom sa vypočíta hodnota sínusu s indexom cyklu a vynásobí amplitúdou nastavenou v textovom poli (*Text Box*). Keďže rozsah D/A prevodu nie je v záporných číslach je potrebné posunúť vypočítanú hodnotu do kladných čísiel (0 V sa posunie do stredu rozsahu na 5 V). Digitálnu hodnotu (inžinierske jednotky) vypočítame vynásobením vypočítanej hodnoty číslom 409,5, keďže 10 V prislúcha 4095 inžinierskych jednotiek (4 095 / 10 = 409,5). Inžinierske jednotky sa následne rozdelia na horný a dolný bajt (viď Obr. 7.49 a Obr. 7.50) a zapíšu sa do registrov D/A prevodu. Potom sa počká kým uplynie 1/628 frekvencie (periódy), ktorá sa nastavila v textovom poli (*Text Box*).

Po stlačení tlačidla v aplikácii sa premenná frekvencie a premenná amplitúdy aktualizujú. Pri vypnutí aplikácie sa vlákno ukončí a register D/A prevodu vynuluje.

Signál s nulovou amplitúdou (konštantný signál o veľkosti 5 V) a frekvenciou 1 Hz sa bude generovať hneď po spustení aplikácie, tlačidlo bude slúžiť len na aktualizáciu frekvencie a amplitúdy. Po vypnutí aplikácie sa generovanie vypne a na analógovom výstupe bude 0 V.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4    using System.Threading;
5    using System.Diagnostics;
6
7    namespace generator
8    {
9         public partial class Form1 : Form
10        {
11             [DllImport("inpout32.dll", EntryPoint="Out32")]
               public static extern void Out(int add, int val);
12
13             const int Baza = 0x260;
14             Thread gen=new Thread(new ThreadStart(generuj));
15             static int frek = 1, amp = 0, hodn1, hodn2;
16
17             public Form1()
18             {
19                  InitializeComponent();
20                  if (!gen.IsAlive) gen.Start();
21             }
22
23             static void generuj()
24             {
25                  while (true)
```

```
26                  {
27                       double sinus;
28                       for (int i = 0; i <= 628; i++)
29                       {
30                            Stopwatch cas=Stopwatch.StartNew();
31                            Int64 stop;
32                            sinus = Math.Sin(i * 0.01);
33                            sinus = amp * sinus;
34                            sinus = sinus + 5;
35                            double hodn3 = sinus * 409.5;
36                            hodn1 = (int)hodn3 / 256;
37                            hodn2 = (int)hodn3 % 256;
38                            Out(Baza + 4, hodn2);
39                            Out(Baza + 5, hodn1);
40                            stop = (10000000 / frek) / 629;
41                            while (cas.ElapsedTicks < stop) ;
42                            cas.Stop();
43                       }
44                  }
45             }
46
47             private void Form1_FormClosing(object sender,
               FormClosingEventArgs e)
48             {
49                  if (gen.IsAlive) gen.Abort();
50                  Out(Baza + 4, 0);
51                  Out(Baza + 5, 0);
52             }
53
54             private void button_Click(object sender,
               EventArgs e)
55             {
56                  frek = int.Parse(textBox6.Text);
57                  amp = int.Parse(textBox5.Text);
58             }
59        }
60   }
```

Vysvetlenie:

1-5: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.

7: Deklarácia a definovanie namespace-u tohto programu.

9: Deklarácia a definovanie hlavnej triedy programu.

11: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.

13: Deklarácia bázovej adresy `Baza` laboratórnej karty PCL-812 typu *const int*

14: Deklarácia vlákna `gen` ktorého algoritmus bude vo funkcií `generuj()`.

15: Deklarácia premenných `frek` a `amp` s inicializačnou hodnotou a premenných `hodn1` a `hodn2` bez inicializačných hodnôt typu int (celé čísla).

17: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.

19: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).
20: Ak vlákno `gen` nie je spustené, tak sa spustí.
23: Deklarácia a definovanie funkcie `generuj()`, ktorá je algoritmom vlákna.
25: Nekonečný cyklus.
27: Deklarácia premennej `sinus` typu *double* (reálne číslo).
28: Cyklus s pevným počtom opakovaní:
- 30: Začatie stopovania času pomocou premennej `cas`.
- 31: Deklarácia premennej `stop` typu *Int64*.
- 32: Výpočet hodnoty sínusu v hodnote jednej stotiny indexu (inkrementálnej premennej) cyklu a vloženie tejto hodnoty do premennej `sinus`.
- 33: Vynásobenie vypočítanej hodnoty (`sinus`) amplitúdou (`amp`) a vloženie výsledku do premennej `sinus`.
- 34: Posunutie hodnoty (premennej) `sinus` o 5 voltov (kvôli rozsahu).
- 35: Výpočet inžinierskych jednotiek s vypočítaného signálu (`sinus`) a vloženie výsledku do premennej `hodn3`.
- 36: Výpočet horného bajtu (`hodn1`), nakoľko horný bajt využíva prvé 4 bity registra (Obr. 7.50) postačuje inžinierske jednotky vydeliť číslom 256.
- 37: Výpočet dolného bajtu (`hodn2`), keďže dolný bajt využíva celých 8 bitov registra (Obr. 7.49) postačuje inžinierske jednotky vydeliť číslom 256 a výsledkom bude zvyšok tohto delenia.
- 38-39: Zápis horného a dolného bajtu do registrov D/A prevodníka.
- 40: Výpočet stopovacieho času a vloženie vypočítanej hodnoty do premennej `stop` (10 000 000 tikov je 1 s a 629 cyklov je jedná perióda signálu).
- 41: Čakanie na uplynutie stopovaného času pomocou cyklu s podmienkou.
- 42: Zastavenie stopovania času.

47: Deklarácia a definovanie funkcie (event-u), ktorá nastane po vypnutí programu.
49: Ak vlákno `gen` je spustené, tak sa zastaví.
50-51: Vynulovanie registrov D/A prevodu.
54: Deklarácia a definovanie funkcie (event-u), ktorá nastane po stlačení tlačidla.
56: Vloženie frekvencie s textového poľa do premennej `frek` s konverziou na typ *int*.
57: Vloženie amplitúdy s textového poľa do premennej `amp` s konverziou na typ *int*.

7.5.5 Analógovo-digitálny prevodník

Analógovo-digitálny prevodník alebo analógovo-číslicový prevodník (ADC z anglického Analog-to-Digital Converter) je elektronické zariadenie na prevod analógového signálu na digitálny signál. Prevod analógového signálu na digitálny je využívaný pomerne často, keďže signály sa skoro výlučne analyzujú a spracovávajú číslicovo (digitálne).

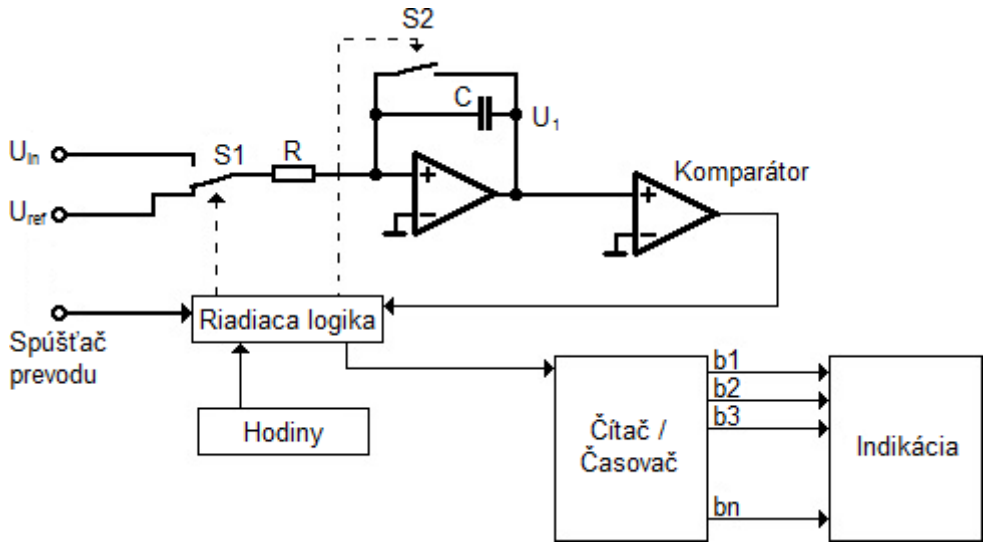
Príkladom konkrétnych aplikácií analógovo-digitálneho prevodu sú elektronické diktafóny, mobilné telefóny, automatizované zberače dát v laboratóriách s následným digitálnym vyhodnotením a archiváciou nameraných dát, či plynový pedál v moderných automobiloch, ktorý už nie je spojený lankom s klapkou na prívod vzduchu, ale uhol jeho stlačenia je meraný potenciometrom, ďalej zdigitalizovaný – s následnou vypočítanou akciou klapky prívodu vzduchu, ktorá je ovládaná motorovým akčným členom. Samozrejme aj pri riadení je tento prevodník dôležitý, pri spätnej väzbe sa do regulátora (napríklad počítača) musí dostať analógový signál zo systému, ktorý je potrebne previesť na digitálny signál, aby mohol byť vypočítaný akčný zásah. Tento prevodník má mnoho využití.

Pri digitalizácii analógovej informácie existuje nespočetné množstvo metód. V nasledujúcom texte sú zhrnuté najjednoduchšie alebo naopak zložité, ale osvedčené, sú nimi tieto metódy:

- integračná metóda,
- metóda využitia medziprevodu napätia na frekvenciu,
- aproximačná metóda,
- metóda založená na sledovaní vstupného signálu,
- metóda paralelného vzorkovania.

Integračná metóda:

Prevodníky postavené na dvojstupňovej integrácii sú najčastejšie využívané v digitálnych meracích prístrojoch. Dôvodom je ich presnosť a odolnosť voči starnutiu prvkov. Ich najchúlostivejšou časťou je integračný článok, ktorý sa však dnes dá vyrobiť veľmi presne (Obr. 7.110):



Obr. 7.110 A/D prevodník založený na integračnej metóde

Meranie je rozdelené do dvoch fáz, ktoré trvajú istý čas (T_1 a T_2). V dobe T_1 je pripojené U_{in}, táto doba je konštantná a je daná časom zaplnenia čítača. Potom (v dobe T_2) sa integrátor pripojí k U_{ref} (má opačnú polaritu ako U_{in}), nastáva pokles a táto doba končí, keď výstupné napätie prechádza nulou a komparátor dá povel na ukončenie prevodu. Doba T2 je meradlom veľkosti signálu U_{in} a meria sa počtom impulzov, ktoré čítač načíta v tejto dobe.

Na začiatku konverzie je spínač S1 krátko zopnutý a na výstupe integrátoru je nulové napätie, hneď ako sa otvorí na výstupe narastá napätie so strmosťou:

$$\left[\frac{dU_1}{dt}\right]_{T_1} = \frac{U_{in}}{RC} \qquad (7.20).$$

Po vynulovaní čítača (po dopočítaní) sa spínač S1 prepne na U_{ref} a na výstupe integrátora bude napätie klesať so strmosťou:

$$\left[\frac{dU_1}{dt}\right]_{T_2} = -\frac{U_{ref}}{RC} \qquad (7.21).$$

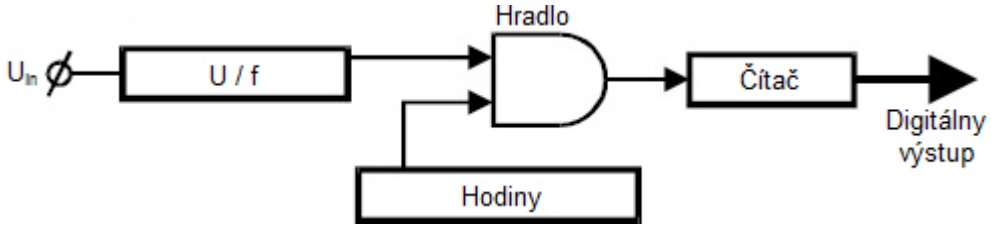
Pritom sa čítajú hodinové impulzy až do nulového napätia na výstupe integrátora, ich počet je daný vzťahom:

$$M = \left|-U_{in}\right| \frac{2^n}{U_{ref}} \qquad (7.22).$$

Metóda využitia medziprevodu napätia na frekvenciu:

Prevodníky pracujúce na princípe prevodu napätia *U* na frekvenciu *f* sa skladajú zo štyroch základných častí (Obr. 7.111):

- prevodník napätia na frekvenciu - jednoduchým napätím preladiteľný oscilátor, obvod využívajúci integrátor,
- generátor hodinového signálu,
- komparačné AND hradlo,
- čítač frekvencie.



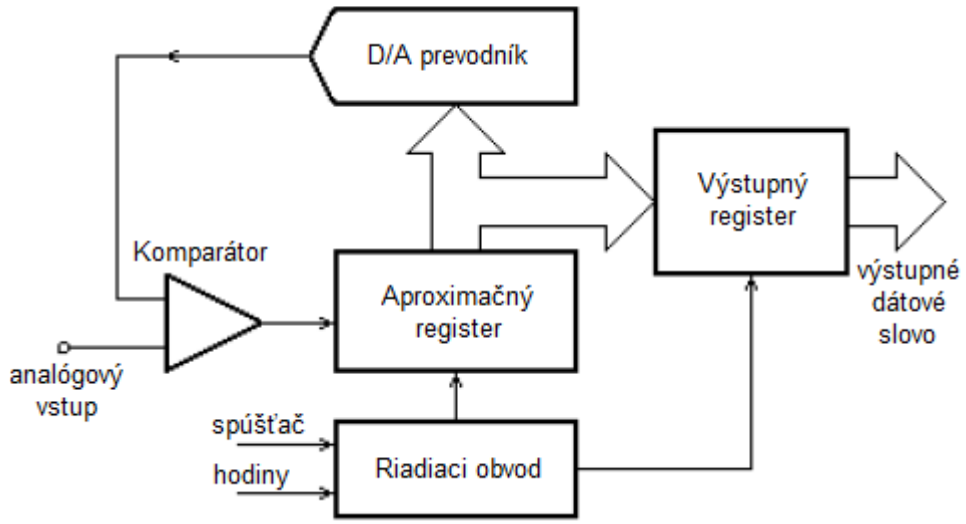
Obr. 7.111 A/D prevodník založený na metóde využitia medziprevodu napätia na frekvenciu

Pri meraní sa najprv prevedie napätie na frekvenciu, potom sa signál s frekvenciou závislou na vstupnom napätí privedie na porovnávacie hradlo AND. Na druhý vývod tohto hradla sa privedie hodinový signál a na výstupe sa objaví výsledok, ktorý čítač navzorkuje a podá ďalej. Pri prevode napätia na frekvenciu sa najčastejšie využívajú integrátory, preto by bolo možné zaradiť aj tento prevodník medzi integračné.

Aproximačný metóda:

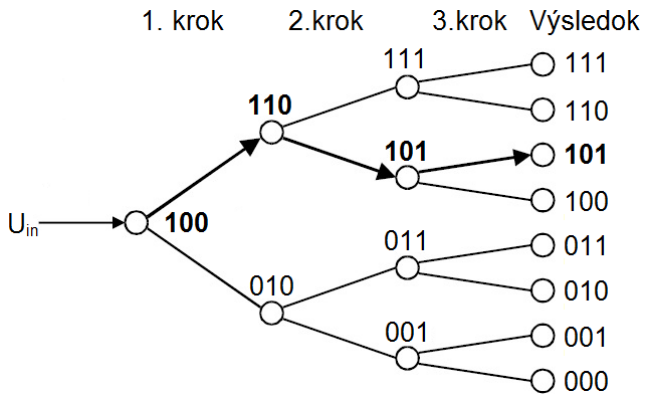
Aproximačné prevodníky (Obr. 7.112), v iných literatúrach tiež ako kompenzačné alebo prevodníky s postupnou aproximáciou, sú v podstate spätnoväzbové systémy, ktoré autonómne generujú signál a ten následne porovnávajú so vstupným. Pohybujú sa v binárnom strome, odkiaľ čerpajú digitálne kombinácie potrebné ako vzory pre generovanie analógového signálu.

Na začiatku je register postupných aproximácií vynulovaný. V prvom kroku je do neho zapísaná 1 ako najvyšší bit a ostatné sú nulové, vygeneruje sa signál a ten sa porovná so vstupným signálom. Podľa výsledku porovnania sa rozhodne o platnosti 1 alebo nahradení 0, ak je vstupný signál nižšej úrovne. Tento proces pokračuje pokiaľ sa nenavzorkuje daný počet bitov. Takže dĺžka merania je závislá od hodinovej frekvencií a digitálnej dĺžke prevodníka (12 bitovému prevodníku bude prevod trvať 12 hodinových impulzov).



Obr. 7.112 A/D prevodník založený na aproximačnej metóde

Vzorkovanie sa v podstate stáva pohybom po binárnom strome úrovne rovnej rozlíšeniu aproximácie (Obr. 7.113).



Obr. 7.113 Stromová štruktúra A/D prevodu

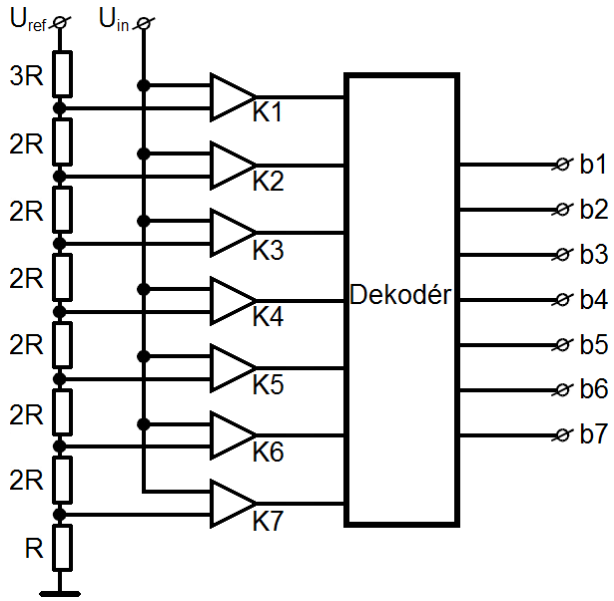
Metóda založená na sledovaní vstupného signálu:

Prevodníky založené na tejto technológii sú podobné aproximačným prevodníkom, ale nepoužívajú aproximačný register. Register je nahradený vratným čítačom. Prevodník sa skladá z troch častí: komparátor, vratný čítač, n-bit D/A prevodník. Signál z D/A prevodníka a vstup sú privedené na vstup komparátora. Výstup komparátora je pripojený na vratný čítač, ktorý sa podľa neho inkrementuje alebo dekrementuje. Jeho hodnota je výstupom a zároveň opravným signálom, ktorý vstupuje do D/A a následne do komparátora.

Metóda paralelného vzorkovania:

Prevodníky využívajúce túto metódu sú najrýchlejšími prevodníkmi, pretože dokážu navzorkovať celé binárne slovo naraz. Ich funkcia je založená na rozložení úrovne vstupného signálu na odporovom rade. Napätie je snímané na každom spoji dvoch odporov

a je v podstate opakom paralelného A/D prevodníka. Z toho vyplýva aj jeho najväčšia nevýhoda, čo je konštrukčná zložitosť. Na n-bitov totiž treba 2^n-1 komparátorov a 2^n odporov. Aj napriek tomu sa kvôli svojej rýchlosti často používajú. Nasledujúci obrázok Obr. 7.114 je hrubá schéma jeho stavby:



Obr. 7.114 A/D prevodník založený na metóde paralelného vzorkovania

7.5.6 Programovanie aproximačného A/D prevodníka

Ako sa dalo všimnúť v predchádzajúcej podkapitole (7.5.5) každá metóda A/D prevodu, okrem metódy paralelného vzorkovania, potrebuje pre svoj chod spúšťač a hodiny. Na laboratórnych kartách PCL-812 a PCL-818 pred začatím prevodu treba nastaviť zdroj, ktorý bude tento prevod spúšťať. Spúšťačom môžu byť 3 signály a to externý signál, výstupný signál čítača/časovača, alebo softvérový signál vyvolaný zápisom do registra. Ešte pred prvým spúšťacím signálom je potrebné nastaviť rozsah napätia a kanál. Nastavenie rozsahu merania pomocou programu zabezpečuje delič napätia (viď Obr. 7.53 a Obr. 7.61), ktorý delí vstupné napätie a to posiela ako referenčné napätie A/D prevodu. Obe laboratórne karty s analógovými vstupmi a výstupmi obsahujú iba jeden aproximačný A/D prevodník, preto je potrebné nastavovanie kanálu (viď Obr. 7.54 a Obr. 7.62).

Ukončenie prevodu je zapísané v registri ako bit DRD alebo INT (viď Obr. 7.48 a Obr. 7.69), v okamihu prepísania bitu (z 0 na 1) môže nastať prerušenie, zápis do pamäte pomocou DMA, alebo nemusí nastať nič a dáta sa vyčítajú priamo z registrov. Udalosť ktorá nastane po prevode sa nastavuje pomocou riadiaceho registra laboratórnej karty (viď Obr. 7.55 a Obr. 7.70).

Ak je spúšťačom A/D prevodu výstup čítača/časovača, tak je potrebné ho pred meraním nastaviť (viď podkapitolu 7.5.1). Keď je spúšťačom externý signál, tak ho treba k laboratórnej karte napojiť (viď Obr. 7.101).

Táto podkapitola obsahuje dve zadania. Prvé zadanie meria napätie na dvoch potenciometroch, spúšťačom je zápis do registra (softvérový spúšťač) a po ukončení merania sa hodnoty vyčítavajú z registra. Druhé zadanie je komplexnejšie, zadaním je vytvoriť

diskrétny PID regulátor (PSD regulátor), konkrétne PI regulátor. Takže v tomto zadaní je potrebná práca s A/D prevodom, D/A prevodom a čítačom časovačom.

Algoritmizácia A/D prevodu má 9 možných konbinácií (3 spúšťače, 3 typy ukončenia prevodu, 3x3), ale nakoľko sa dajú pomocou predchádzajúcich kapitol (prerušenia, technologické rozhrania) skombinovať a navrhnúť, tak preto sú na tomto mieste zhrnuté len tie, ktoré sú pre spomínané dve zadania potrebné. Celý potrebný postup pre nasledujúce zadania, ktorý bol popísaný vyššie je zhrnutý v nasledujúcich dvoch tabuľkách (Tab. 7.16 a Tab. 7.17):

Tab. 7.16 Postup algoritmizácie A/D prevodu pri softvérovom meraní

Krok	Úkon	PCL-812	PCL-818
1	Nastavenie softvérového spúšťača	Báza + BH = 1	Báza + 9H = 0
2	Nastavenie rozsahu napätia	Báza + 9H = 0	Báza + 1H = 0
3	Nastavenie kanála	Báza + AH = k	Báza + 2 = k + k * 16
4	Spustenie merania	Báza + CH = 0	Báza + 0H = 0
5	Čakanie na ukončenie prevodu	Ak 4. bit báza+5 je 1	Ak 4. bit báza+8 je 1
6	Načítanie dát	Báza + 4, Báza + 5	Báza + 0, Báza + 1
7	Konverzia	Podľa potreby	Podľa potreby

Tab. 7.17 Postup algoritmizácie A/D prevodu pri využití výstupu čítača/časovača ako spúšťača prevodu

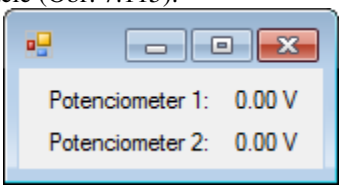
Krok	Úkon	PCL-812	PCL-818
1	Nastavenie čítača/časovača	Báza + 3 = 0x74 Báza+1 LSB a MSB Báza + 3 = 0xb4 Báza+2 LSB a MSB	Báza + FH = 0x74 Báza+DH LSB a MSB Báza + FH = 0xb4 Báza+EH LSB a MSB
2	Nastavenie výstupného signálu čítača/časovača ako spúšťača	Báza + BH = 6	Báza + 9H = 3
3	Nastavenie rozsahu napätia	Báza + 9H = 0	Báza + 1H = 0
4	Nastavenie kanála	Báza + AH = k	Báza + 2H = k + k * 16
5	Čakanie na ukončenie prevodu	Ak 4. bit báza+5 je 1	Ak 4. bit báza+8 je 1
6	Načítanie dát	Báza + 4, Báza + 5	Báza + 0, Báza + 1
7	Konverzia	Podľa potreby	Podľa potreby

Zadanie 1:

Naprogramujte program, ktorý zosníme napätia na potenciometroch panela pripojených k laboratórnej karte PCL-818 a vypíše zosnímané napätia do okna aplikácie. A/D prevodník využije softvérové spúšťanie prevodu a prevod sa ukončí iba zápisom bitu s oznámením o ukončení prevodu (softvérové meranie). Program naprogramujte v programovacom jazyku C# pod operačným systémom Windows XP (alebo pod vyššou radou OS) vo vývojovom prostredí Visual Studio 2005 (alebo vo vyššej rade prostredia).

Analýza zadania:

Návrh vzhľadu aplikácie (Obr. 7.115):



Obr. 7.115 Návrh vzhľadu aplikácie pre snímanie napätí na potenciometroch

Aplikácia bude obsahovať 2 textové bloky (*Label*) pre vypísanie aktuálnej hodnoty napätia a jeden časovač (*Timer*), ktorému sa vo vlastnostiach (*properties*) hneď nastaví vlastnosť *Enabled* na hodnotu *true*.

V deklaračnej časti sa prilinkujú funkcie `Out32()` a `Inp32()` z dynamickej knižnice *inpout32.dll* a zadeklarujú premenné pre bázovú adresu laboratórnej karty, inžinierske jednotky a obe napätia.

Inicializačná časť programu bude okrem funkcie `InitializeComponent()` obsahovať aj nastavenie softvérového spúšťača (Obr. 7.70) a nastavenie rozsahu napätia (Obr. 7.61) A/D prevodu na 10 V.

V programe časovača (*Timer*-a), ktorý sa bude opakovať každých 100 ms, prebehnú dve merania (A/D) prevody. Najskôr sa nastaví kanál (Obr. 7.62), ktorého hodnota je na obrázku Obr. 7.90, ktorý je aj nad počítačom ku ktorému je laboratórna karta pripojená. Potom sa spustí pomocou softvérového spúšťača A/D prevod, zápisom akejkoľvek hodnoty do registra s adresou báza + 0. Počká sa na ukončenie prevodu, kým 4. bit (INT, Obr. 7.69) v registri na adrese báza + 8 nie je rovný jednej. Ak je prevod ukončený, tak sa z hodnôt v registroch A/D prevodu (báza + 0 a báza + 1) vypočítajú inžinierske jednotky (viď Obr. 7.59 a Obr. 7.60). Potom sa tieto jednotky prevedú na napätie a hodnota sa vypíše do okna aplikácie. Rovnaký postup sa uskutoční aj pre ďalší potenciometer, jedinou zmenou bude kanál A/D prevodu.

Riešenie:

```
1    using System;
2    using System.Windows.Forms;
3    using System.Runtime.InteropServices;
4
5    namespace Potenciometer
6    {
7        public partial class Form1 : Form
8        {
9            [DllImport("inpout32.dll", EntryPoint="Out32")]
             public static extern void Out(int add, int val);
10           [DllImport("inpout32.dll", EntryPoint="Inp32")]
             public static extern byte In(int addr);
11           int baza=0x220, IJ;
12           double volt1, volt2;
13
14           public Form1()
15           {
16               InitializeComponent();
17               Out(baza + 9, 0);
18               Out(baza + 1, 0);
19           }
20           private void timer1_Tick(object sender,
             EventArgs e)
21           {
22               Out(baza + 2, 0);
23               Out(baza + 0, 0);
24               while (((In(baza + 8) / 16) % 2) != 1);
25               IJ = (In(baza + 1)*16) + (In(baza + 0)/16);
26               volt1 = (double)(IJ - 2048) / 204.8;
27               label1.Text=volt1.ToString("0.00 V");
```

```
28
29                  Out(baza + 2, 17);
30                  Out(baza + 0, 0);
31                  while (((In(baza + 8) / 16) % 2) != 1) ;
32                  IJ = (In(baza + 1)*16) + (In(baza + 0)/16);
33                  volt2 = (double)(IJ - 2048) / 204.8;
34                  label2.Text = volt2.ToString("0.00 V");
35              }
36          }
37      }
```

Vysvetlenie:

1-3: Prilinkovanie dôležitých (potrebných pre tento program) systémových tried framework-u .NET.
5: Deklarácia a definovanie namespace-u tohto programu.
7: Deklarácia a definovanie hlavnej triedy programu.
9: Vloženie funkcie `Out32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `Out()`.
10: Vloženie funkcie `Inp32()` z dynamickej knižnice *inpout32.dll* do tohto programu ako funkciu `In()`.
11: Deklarácia bázovej adresy laboratórnej karty (`baza`) a inžinierskych jednotiek (`IJ`) ako typ *int* (celé čísla).
12: Deklarácia meraných napätí (`volt1` a `volt2`) ako typ *double* (reálne čísla).
14: Deklarácia a definovanie inicializačnej funkcie samotnej aplikácie.
16: Inicializácia všetkých dôležitých komponentov pre zobrazenie okna (aplikácie).
17: Nastavenie softvérového spúšťača A/D prevodu (Obr. 7.70).
18: Nastavenie rozsahu A/D prevodu na 10 V (Obr. 7.61).
20: Deklarácia a definovanie funkcie (event-u), ktorá sa vykoná každú periódu časovača (*timer*-a).
22: Nastavenie kanálu A/D prevodu na 0. kanál (Obr. 7.62).
23: Spustenie A/D prevodu pomocou zápisu ľubovoľnej hodnoty do registra báza + 0.
24: Cyklus s podmienkou, ktorý čaká na ukončenie prevodu (Obr. 7.69).
25: Výpočet inžinierskych jednotiek z výstupných registrov A/D prevodu (Obr. 7.59 a Obr. 7.60) a ich zápis do premennej `IJ`.
26: Do premennej `volt1` sa vloží výpočet napätia z inžinierskych jednotiek.
27: Vypísanie hodnoty `volt1` do okna aplikácie pomocou objektu `label1`.
29: Nastavenie kanálu A/D prevodu na 1. kanál (k + 16 * k, viď Obr. 7.62).
30: Spustenie A/D prevodu pomocou zápisu ľubovoľnej hodnoty do registra báza + 0.
31: Cyklus s podmienkou, ktorý čaká na ukončenie prevodu (Obr. 7.69).
32: Výpočet inžinierskych jednotiek z výstupných registrov A/D prevodu (Obr. 7.59 a Obr. 7.60) a ich zápis do premennej `IJ`.
33: Do premennej `volt2` sa vloží výpočet napätia z inžinierskych jednotiek.
34: Vypísanie hodnoty `volt2` do okna aplikácie pomocou objektu `label2`.

Zadanie 2:

Naprogramujte diskrétny PI regulátor, ktorý na základe výstupu zo systému vypočíta akčný zásah pre vstup do systému. Vstupnými parametrami regulátora budú

proporcionálna zložka, integračná zložka, vzorkovacia frekvencia, požadovaná hodnota regulácie a názov súboru do ktorého sa budú ukladať merané hodnoty. Systém bude regulovaný pokiaľ sa nestlačí ľubovoľný kláves na klávesnici. Pre rýchle riadenie sa využije programovací jazyk C pod operačným systémom MS-DOS.

Analýza zadania:

Nakoľko bude potrebné pracovať so vstupno-výstupnými portami (registrami) je nutné prilinkovať knižnicu *dos.h*. Taktiež sa bude pracovať so štandardnými vstupmi a výstupmi jazyka C, čiže je potrebná aj knižnica *stdio.h*.

V deklaračnej časti bude nutné zadeklarovať množstvo premenných ako proporcionálnu zložku, integračnú zložku, požadovaná hodnota regulácie, vzorkovacia frekvencia, prepočet vzorkovacej frekvencie na vstupné parametre čítača/časovača, chyba regulácia a množstvo ďalších.

Po deklarácií premenných sa vyčisti obrazovka, zadefinuje bázová adresa laboratórnej karty, vypne A/D prevod a do registrov D/A prevodu sa zapíšu nuly, aby sa nepoškodil systém, ak tam boli iné hodnoty. Následne sa vytvorí cyklus s platnou podmienkou (nekonečný cyklus) do ktorého sa vloží celý program. Najskôr sa program opýta, že či chce užívateľ vykonať reguláciu. Ak nechce vykonať reguláciu, program sa vypne v opačnom prípade program bude pokračovať ďalej. Postupne si program vypýta všetky vstupné parametre regulátora: názov súboru do ktorého sa budú ukladať merania, požadovaná hodnota regulácie, integračná zložka, proporcionálna zložka a vzorkovacia frekvencia. Vzorkovacia frekvencia sa prepočíta na vstup čítača a nastaví sa čítač/časovač na požadovanú frekvenciu. Ešte pred začiatkom regulácie sa nastaví kanál A/D prevodu, kanál sa nastaví podľa toho, kam je pripojený výstup zo systému, v tomto prípade je to kanál 10. Otvorí sa súbor pre zápis nameraných dát a spustí sa cyklus regulácie. Tento cyklus sa bude opakovať kým sa nestlačí ľubovoľný kláves na klávesnici.

Počas regulácie sa najskôr nastaví spúšťač. Spúšťačom bude výstup časovača, aby bola dodržaná vzorkovacia frekvencia. Následne sa vyčíta hodnota napätia z A/D prevodu, ale je nutné počkať kým nie je A/D prevod ukončený, to zabezpečí požadovanú vzorkovaciu frekvenciu. Vyčítaná hodnota sa prepočíta na volty, zakáže sa A/D prevod a napätie vstupu sa prepočítajú na luxy (tento prípad reguluje svetelnú sústavu, tento prepočet sa dá nahradiť iným, poprípade len premennej lux priradiť priamo vypočítané napätie). Potom sa vypočíta chyba regulácie, diferencia chýb a čas medzi regulačnými zásahmi. Ak všetky parametre sú známe, tak sa môže vypočítať akčný zásah pomocou rýchlostného algoritmu:

$$\Delta v(i) = K\left(\Delta e(i) + \frac{\Delta t}{T_I}\cdot e(i)\right) \qquad (7.23).$$

Potom sa akčný zásah obmedzí na hodnoty od 0,7 po 1,5 V (podľa charakteristiky modelu svetelnej sústavy) a vypočítajú inžinierske jednotky, ktoré sa rozložia na vrchný a spodný bajt. Následne sa tieto hodnoty zapíšu do registrov D/A prevodu a do súboru sa vložia informácie o výstupe systému a akčnom zásahu (vstupe do systému). Tento cyklus regulácie sa opakuje, kým sa nestlačí ľubovoľné tlačidlo na klávesnici.

Po stlačení klávesa sa súbor uloží a zatvorí, potom sa do registrov D/A prevodu zapíšu nuly, aby systém nebol pod napätím. Nakoniec sa program vráti do hlavného cyklu, kde sa opýta či užívateľ chce vykonať ďalšiu reguláciu.

Riešenie:

```
1    #include<stdio.h>
2    #include<dos.h>
3    main()
4    {
5         float data,lux,chyba,dt,ti,p,vstup,dvstup,stch,dch;
6         int port,h_byte,l_byte,rlux,frek,frekc,vol,volh,vold;
7         char y, sub[12];
8         FILE *f;
9         clrscr();
10        port=0x220;
11        outportb(port+11,0);
12        outportb(port+4,0);
13        outportb(port+5,0);
14        while (1==1)
15        {
16             do
17             {
18                  printf("Uskut. novu reg. (a-ano, n-nie):");
19                  scanf("%s",&y);
20             }
21             while ((y!='a') && (y!='n'));
22             if (y=='n') return 0;
23             printf("Názov sub. do kt. sa bude ukl. mer.:");
24             scanf("%s",&sub);
25             printf("Požadovaná hodnota regulácie (luxy):");
26             scanf("%d",&rlux);
27             printf("Integračná zložka PI regulátora:");
28             scanf("%f",&ti);
29             printf("Proporcionálna zložka PI regulátora:");
30             scanf("%f",&p);
31             do
32             {
33                  printf("Vzorkovacia frekvencia riadenia:");
34                  scanf("%d",&frek);
35             }
36             while ((frek<150) || (frek>300));
37             frekc=10000/frek;
38             outportb(port+3,0x74);
39             outportb(port+1,frekc);
40             outportb(port+1,0);
41             outportb(port+3,0xb4);
42             outportb(port+2,200);
43             outportb(port+2,0);
44             outportb(port+10,10);
45             f=fopen(sub,"w");
46             printf("\n Prebieha regulácia \n");
47             do
48             {
49                  outportb(port+11,6);
50                  do
```

```
51                  {
52                       h_byte=inportb(port+5);
53                  }
54                  while (h_byte > 15 );
55                  l_byte=inportb(port+4);
56                  data=(h_byte*256+l_byte-2048)/204.8;
57                  outportb(port+11,0);
58                  lux=-127.581*data+899.4462;
59                  chyba=lux-rlux;
60                  dch=chyba-stch;
61                  stch=chyba;
62                  dt=1/frek;
63                  dvstup=p*ti*dt*chyba+p*dch;
64                  vstup=vstup-dvstup;
65                  if (vstup>1.5) vstup=1.5;
66                  if (vstup<0.7) vstup=0.7;
67                  vol=vstup*819.1;
68                  volh=vol/256;
69                  vold=vol%256;
70                  outportb(port+4,vold);
71                  outportb(port+5,volh);
72                  fprintf(f,"%.1f %.3f\n",lux,vstup);
73             }
74             while (!kbhit());
75             printf("\n Regulácia ukončená \n \n ");
76             fclose(f);
77             outportb(port+4,0);
78             outportb(port+5,0);
79        }
80   }
```

Vysvetlenie:

1-2: Prilinkovanie dôležitých knižníc k programu.

3: Deklarácia a definovanie hlavného programu aplikácie.

5-8: Deklarácia premenných:

`data` – vstup do regulátora vo voltoch (*u(t)*),

`lux` – prepočet *u(t)* na luxy,

`chyba` – aktuálna chyba regulácie (*e(i)*),

`dt` – prepočet frekvencie na čas jedného kroku (Δt),

`ti` – člen I, PI regulátora,

`p` – člen P, PI regulátora,

`vstup` – vypočítaný nový vstup (výstup reg.) do systému (*y(t)*),

`dvstup` – rozdiel medzi vstupmi ($\Delta y(t)$),

`stch` – predošlá chyba regulácie (*e(i-1)*),

`dch` – rozdiel chýb ($\Delta e(i)$),

`port` – port na ktorom je napojená Labkarta,

`h_byte` – horný bajt prepočtu vstupu do regulátora (A/D prevodu),

`l_byte` – dolný bajt prepočtu vstupu do regulátora (A/D prevodu),

`rlux` – konečná hodnota regulácie,

`frek` – frekvencia riadenia,

`frekc` – prepočítaná frekvencia riadenia pre vstup do časovača,
`vol` – inžinierske jednotky napätia,
`volh` – horný bajt výstupu regulátora (D/A prevodu),
`vold` – dolný bajt výstupu regulátora (D/A prevodu),
`y` – premena slúžiaca na vypnutie programu,
`sub` – meno súboru,
`f` – súbor kde sa ukladajú informácie o regulácií,

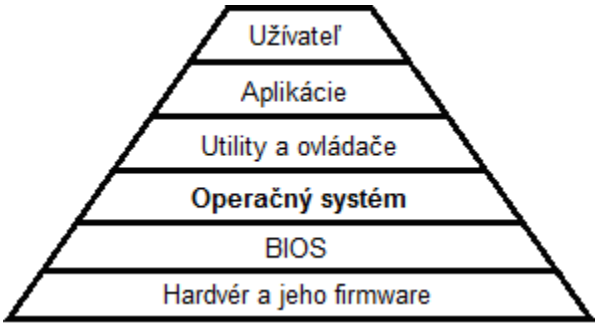
9: Vyčistenie obrazovky.
10: Vloženie bázovej adresy do premennej port.
11: Vypnutie A/D prevodu.
12-13: Nastavene 0V na výstup laboratórnej karty y(t)=0.
14-79: Zavedenie nekonečného cyklu ktorý trvá, až kým užívateľ nepožiada o ukončenie cyklu.
- 16-22: Zistenie či užívateľ chce, aby cyklus pokračoval (samotný program).
- 23-24: Načítanie mena súboru.
- 25-26: Načítanie konečnej hodnoty regulácie (v luxoch).
- 27-30: Načítanie členov PI.
- 31-36: Ošetrené načítanie frekvencie riadenia (iba hodnoty v rozmedzí 150 Hz až 300 Hz).
- 37: Prepočet frekvencie riadenia pre vstup do časovača.
- 38-43: Nastavenie časovača laboratórnej karty.
- 44: Nastavenie 10. kanálu A/D prevodu.
- 45: Otvorenie súboru na ukladanie informácií o regulácií.
- 46: Oznámenie o začatí regulácie.
- 47-74: Samotná regulácia (cyklus regulácie), ktorá trvá pokiaľ sa nestlačí akýkoľvek kláves na klávesnici.
  - 49: Nastavenie výstupu čítača/časovača ako spúšťača A/D prevodu.
  - 50-54: Načítanie horného bajtu A/D prevodu aj z ošetrením zistenia ukončenia prevodu.
  - 55: Načítanie dolného bajtu A/D prevodu.
  - 56: Prepočet `h_byte` a `l_byte` na volty.
  - 57: Vypnutie A/D prevodu.
  - 58: Prepočet voltov na luxy.
  - 59: Výpočet chyby (*e(i)*).
  - 60: Výpočet *Δe(i)*.
  - 61: Odpamätanie aktuálne chyby do budúceho cyklu.
  - 62: Výpočet *Δt*.
  - 63: Výpočet *Δy(t)* podľa rýchlostného algoritmu.
  - 64: Výpočet akčného zásahu *y(t)*.
  - 65-66: Ošetrenie maximálneho a minimálneho akčného zásahu *y(t)*.
  - 67: Prepočet *y(t)* na inžinierske jednotky pre laboratórnu kartu pri referenčnom napätí 5 V.
  - 68-69: Výpočet horného a dolného bajtu pre D/A prevod.
  - 70-71: Zapísanie *y(t)* na reálny výstup z laboratórnej karty.
  - 72: Zapísanie luxov na vstupe do regulátora (v luxoch) a výsledného výstupu (vo voltoch) do súboru.
- 75: Oznámenie o ukončení regulácie.
- 76: Uloženie a zatvorenie súboru.
- 77-78: Nastavene 0V na výstup laboratórnej karty (*y(t)=0*).

8 Operačné systémy

Operačný systém je skupina programov a mnohých súborov, ktoré riadia a kontrolujú činnosť hardvéru a softvéru tvoriaceho počítač, takže operačný systém:

- riadi a spravuje technické prostriedky,
- spravuje dáta,
- riadia spracovanie úloh,
- podporuje komunikáciu s užívateľom,
- podporuje bezpečnosť a spoľahlivosť výpočtového systému.

Pozícia operačného systému v softvérovej architektúre počítača je na nasledujúcom obrázku (Obr. 8.1):



Obr. 8.1 Pozícia operačného systému v softvérovej architektúre počítača

Operačné systémy môžeme rozdeliť takto:

- jedno-užívateľské a jedno-procesné (napr.: MS-DOS),
- jedno-užívateľské a viac-procesné (napr.: Windows 95, Windows 98),
- viac-užívateľské a viac-procesné (napr.: UNIX, Windows 7),
- operačné systémy reálneho času (napr.: RTOS).

Operačný systém vie sprostredkovať komunikáciu s užívateľom pomocou *textu* (príkazový riadok, command line), alebo *graficky* (grafické rozhranie: ikony, okná, atď.). Najpoužívanejším vstupným zariadením textového módu operačného systému je klávesnica, a grafického módu myš, ale myš už pomaly začínajú nahrádzať dotykové obrazovky. Dokonca už existujú operačné systémy, kde je prioritným vstupno-výstupným zariadením dotyková obrazovka (Android).

Operačné systémy sú diskovo orientované, operačný systém je uložený na pevnom disku a pracuje na tomto disku a aj s ním.

Jadro operačného systému (OS kernel) je súhrn základných funkcií systému, ako je prístup k prostriedkom výpočtového systému vrátane pamätí a procesoru, správa súborov a podobne. Jadro zaisťuje beh procesu v systéme a prideľuje im prostriedky. Ostatné časti operačného systému obyčajne pri svojej činnosti iba využívajú služby jadra. Práve štruktúra a naprogramovanie jadra má zásadný vplyv na bezpečnosť celého systému.

Štruktúra operačných systémov:

- *Monolitická štruktúra:* jadro operačného systému a rozhrania,
- *Hierarchická štruktúra:* usporiadaná do vrstiev od najnižšej po najvyššiu,

- *Virtuálne počítače:* systém je rozdelený do modulov, ktoré sú komunikačne prepojené (výmena dát),
- *Abstraktný počítač:* systém je rozdelený do modulov, pričom každý modul plní svoju vlastnú funkciu,
- *Model klient-server:* systém má malé jadro, ktoré plní základné funkcie, ostatné funkcie vykonávajú procesy uložené na vzdialenom serveri, výhodou tejto štruktúry je vysoká stabilita,
- *Stavebná štruktúra:* systém má malé jadro, ostatné časti jadra sú prilinkované iba ak ich nejaká klientska (užívateľská) aplikácia potrebuje, niektoré operačné systémy reálneho času túto štruktúru využívajú.

8.1 Základné pojmy

Táto podkapitola priblíži a zadefinuje niekoľko pojmov, s ktorými sa pri operačných systémoch často stretáva.

Ovládač (driver) je rozhranie medzi rôznymi perifériami (zariadeniami), alebo medzi perifériami a softvérom.

Zavádzanie (bootovanie) – zavádzanie operačného systému do operačnej pamäte počítača.

Multitasking je metóda, kde viacero procesov (úloh) je vykonávaných v rovnakom časovom úseku. Multitasking sa dá rozdeliť na kooperatívny a preemptívny multitasking:

- *Kooperatívny multitasking* prideľuje jednotlivým procesom procesorový čas (CPU) na takú dobu, na akú ju daná aplikácia potrebuje.
- *Preemptívny multitasking* – prideľovanie času riadi operačný systém, ktorý prepína prideľovanie CPU medzi procesmi.

Proces je bežiaci program, ktorý obsahuje (bližšie informácie …):

- spustiteľný program,
- dátovú tabuľku,
- postupnosť algoritmu (programu).

Microkernel architecture – zahŕňa iba niekoľko základných funkcií ako napríklad definíciu adresného priestoru, základnú postupnosť a podobne.

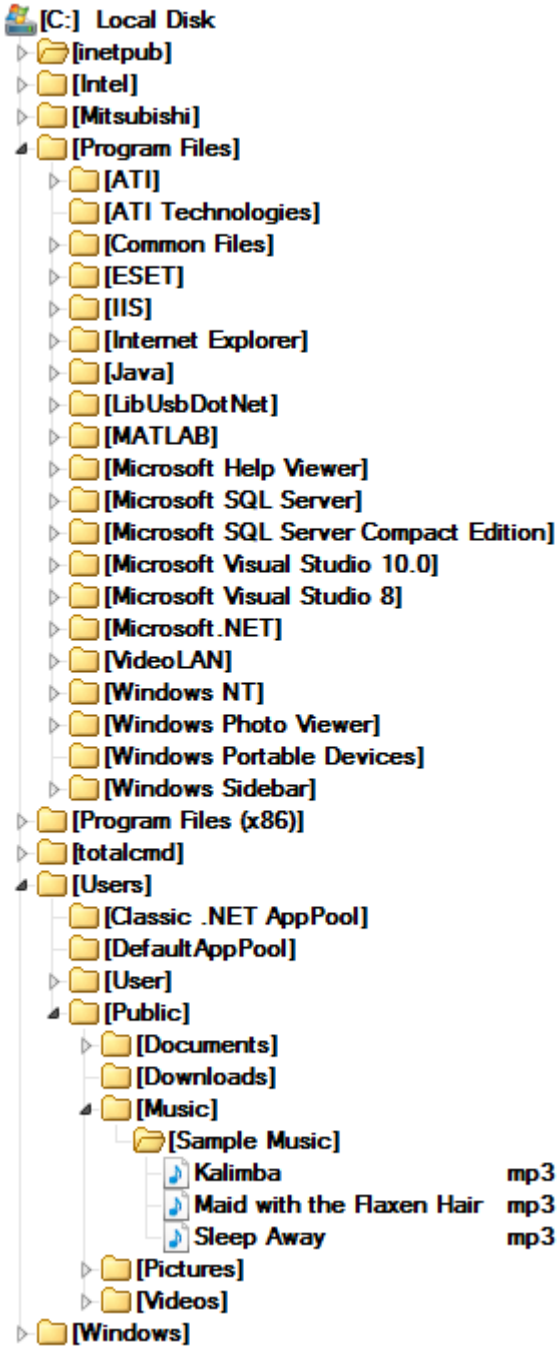
Multithreading je rozdelenie jedného procesu na časti, ktoré bežia v rovnakom časovom úseku. Tieto časti sa nazývajú *vlákna* (threads, vykonávacie toky). Vlákna sa môžu vykonávať paralelne pri dnešnej architektúre počítačov (viac jadrové / procesorové počítače), ale bežiacich vlákien je väčšinou viac ako procesorov v počítači. Takže operačný systém vymedzuje každému vláknu obmedzený čas na CPU, ale tento čas je taký krátky, že užívateľ to vníma ako paralelne vykonávané vlákna.

Organizácia dát na disku – adresáre:

- stromová štruktúra
- cesta (Path):
  - súbor je presne určený cestou,
  - príklad takejto cesty: C:\Windows\System\sys.dll,

- typy ciest:
  - relatívna cesta (čiastočná cesta): \sys.dll
  - absolútna cesta (celá cesta): C:\Windows\System\sys.dll

- stromová štruktúra (Obr. 8.2):



Obr. 8.2 Stromová štruktúra organizácie dát na disku operačných systémov Windows

Súbor je množina súvisiacich prvkov. Formát súborov je *.*, prvá časť formátu súboru je jeho názov a druhá časť je typ súboru. V názve a type súboru sú zakázané znaky: „/“, „\“, „:“, „*“, „?“, „<“, „>“, „|“, „““ (lomky, dvojbodka, hviezdička/násobenie, otáznik, väčší, menší, čiara, úvodzovky). Základné typy (prípony) súborov:

- *bin* – binárny súbor (strojový kód, procesný kód),
- *exe* – samostatne spustiteľný súbor (staré *EXE* súbory boli zároveň binárnymi kódmi procesu, dnes to nie je pravidlom),
- *dll* – dynamická knižnica (najčastejšie je to binárny kód, ktorý nie je samostatne spustiteľný),
- *msi* – inštalačný súbor operačných systémov Windows,
- *bmp* – bitmap: rastrový obrázok (bez kompresie),
- *jpg* – joint photographic experts group – rastrový obrázok s kompresiou,
- *gif* – graphic interchange format – obrázok alebo animácia,
- *svg* – scalable vector graphics – vektorový obrázok,
- *zip* – komprimované súbory alebo adresáre so súbormi,
- *rar* – komprimované súbory alebo adresáre so súbormi,
- atď., …

Kompresia dát na disku používa rôzne metódy redukcie (zmenšenia) objemu dát ako napríklad *RAR*, *ZIP*, *7Z*, *ARJ*, *CAB*, atď.

Kláster (data cluster) je najmenšia logická dátová jednotka na pevnom disku, ktorá zahŕňa niekoľko sektorov. Veľkosť klástra závisí od veľkosti pevného disku a použitého systému súborov. Napríklad HDD ktorý má kapacitu od 0,5 GiB po 1 GiB a používa súborový typ FAT 16, tak veľkosť klástra je 16kiB (FAT16 dokáže adresovať 16-timi bitmi: 2^{16}.16kiB = 1GiB). Maximálna veľkosť disku podporovaná FAT16 bola 2GiB (kláster mal 32 kiB). Potom sa začal pri operačných systémoch Windows používať systém súborov FAT32 a NTFS. Dnes je to najmä NTFS, najmenší zadefinovateľný kláster má 4 kiB a pri tejto veľkosti dokáže zaadresovať 16 TiB (max. veľkosť klástra 64 kiB zaadresuje 256 TiB).

Defragmentácia má na starosti usporiadanie dát na disku tak, aby boli zaplnené klástre v čo najbližšom slede. Defragmentácia je v podstate proces, ktorý redukuje počet voľných fragmentov pamäte. Táto práca sa vykonáva pri každom ukladaní nových dát na disk. Ak sa ukladá nový súbor, tak sa v pamäti hľadajú za sebou idúce voľné klástre s kapacitou veľkosti ukladaného súboru. Keď sa pracuje s veľkými súbormi môže nastať prípad, že je veľa fragmentov pamäte voľných. Dávnejšie, keď boli kapacity pevných diskov rádovo menšie používal sa nástroj pre defragmentáciu, dnes je výhodnejšie disk formátovať.

Master boot record je krátky program spúšťaný BIOS-om pri štarte počítača, jeho úlohou je nájsť a načítať aktívnu oblasť pomocou *tabuľky partícií* (partition table). Pomocou tohto programu a tabuľky sa spúšťa (zavádza) operačný systém.

Virtuálna pamäť slúži k rozšíreniu operačnej pamäte. Princíp virtuálnej pamäte spočíva v tom, že nie celý program alebo všetky údaje sa nachádzajú naraz v hlavnej pamäti počítača. Nachádza sa tam iba tá časť, ktorá sa práve používa, zvyšná časť programu alebo údajov je uložená vo vonkajšej pamäti, napr. na pevnom disku. Najpoužívanejší spôsob realizácie virtuálnej pamäte je segmentovanie a stránkovanie. Samozrejme prístup k dátam je omnoho pomalší ako pri operačnej pamäti.

Vyrovnávacia pamäť (Cache, kešovacia pamäť, rýchla prístupová pamäť) slúži na preklenutie rádového rozdielu medzi prístupovou dobou registrov procesora a hlavnej pamäte. Vyrovnávacia pamäť je rýchla pamäť, rádovo menšej kapacity ako hlavná pamäť, umiestnená medzi procesor a hlavnú pamäť. Do vyrovnávacej pamäte sa presunie časť obsahu hlavnej pamäte a procesor sprístupňuje informácie z vyrovnávacej pamäte vyššou rýchlosťou.

8.2 Prehľad operačných systémov

V tejto podkapitole sa rozoberú operačné systémy od firmy Microsoft a základ operačného systému UNIX. Operačne systémy sú v tejto podkapitole rozdelené takto:

- MS-DOS,
- Windows:
  - Windows s jadrom operačného systému MS-DOS (Windows verzií od 1.0 po 4.9, okrem Windows NT 3.1, 3.5, 3.51 a 4.0),
  - Windows bez jadra operačného systému MS-DOS (Windows NT 3.1, 3.5, 3.51, 4.0 a verzie 5 až do dnes),
- UNIX

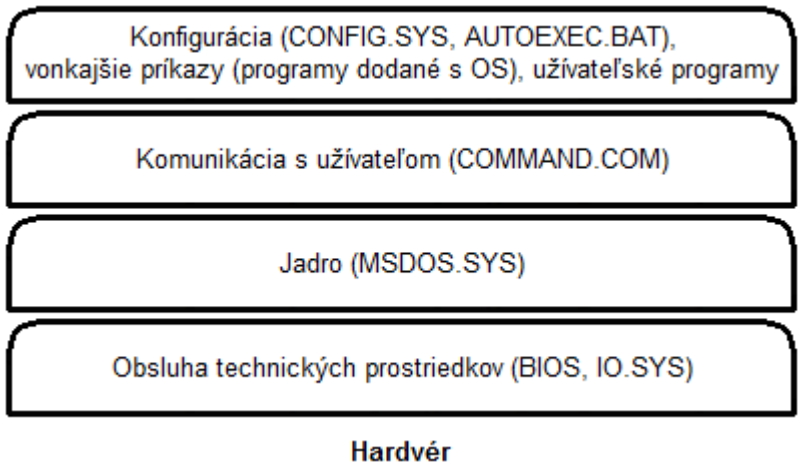
Prehľad operačných systémov Windows od spoločnosti Microsoft:

- Windows 1.0 (1985),
- Windows 2.0 (1987),
- Windows 2.1 (1988),
- Windows 2.11 (1989),
- Windows 3.0 (1990),
- Windows 3.1 (1991),
- Windows 3.11 (1992),
- Windows NT 3.1 (1993),
- Windows NT 3.5 (1994),
- Windows NT 3.51 (1995),
- Windows 4.0 – 4.9:
  - Windows 95 (1995) – verzia Windows 4.00.950,
  - Windows 95 A (1996) – verzia Windows 4.00.951,
  - Windows 95 B (1996) – verzia Windows 4.03.1212,
  - Windows 95 B USB (1997) – verzia Windows 4.03.1214,
  - Windows 95 C (1997) – verzia Windows 4.03.1216,
  - Windows 98 (1998) – verzia Windows 4.10.1998,
  - Windows 98 Sec. Edition (1999) – verzia Windows 4.10.2222,
  - Windows Millennium Edition (2000) – verzia Windows 4.9,
- Windows NT 4.0 (1996),
- Windows NT 5.0 – 5.9:
  - Windows 2000 (2000) – verzia Windows 5.0
  - Windows Server 2000 (2000) – verzia Windows 5.0
  - Windows XP 32bit (2001) – verzia Windows 5.1
  - Windows Server 2003 (2003) – verzia Windows 5.2
  - Windows XP 64bit (2003) – verzia Windows 5.2
- Windows NT 6.0 – 6.9:
  - Windows Vista (2005) – verzia Windows 6.0
  - Windows Server 2008 (2008) – verzia Windows 6.0
  - Windows 7 (2009) – verzia Windows 6.1
  - Windows 8 (2012) – verzia Windows 6.2
  - Windows Server 2012 (2012) – verzia Windows 6.2
  - Windows 8.1 (2013) – verzia Windows 6.3
  - Windows 10 (2014) – verzia Windows 10 (marketingový ťah, inač to mala byť verzia 6.4)

8.2.1 MS-DOS

MS-DOS je jedno-procesný, jedno-užívateľský a lokálny univerzálny operačný systém. Samotný MS-DOS má veľmi jednoduchú vrstvenú štruktúru (Obr. 8.3).

Najbližšie k hardvéru má BIOS a po ňom súbor *IO.SYS*, ktorý sa stará o obsluhu periférií. BIOS poskytuje programátorom základné ovládanie hardvéru (napr. klávesnice, monitora) cez hardvérové a softvérové prerušenia.



Obr. 8.3 Štruktúra operačného systému MS-DOS

Nad vrstvou pre ovládanie hardvéru je vrstva samotného jadra systému predstavovaná súborom *MSDOS.SYS*. Tento systém má monolitické jadro zložené z jediného súboru. Jadro poskytuje ďalšie softvérové prerušenia (prerušenia DOS-u), napríklad pre prístup k súborom, alebo pokročilejšiu prácu s grafikou.

Nasledujúca vrstva tvorená súborom *COMMAND.COM* je textové rozhranie medzi užívateľom a systémom. Tento program je spustený po celú dobu práce systému a komunikuje s užívateľom (spustené programy komunikujú s nižšími vrstvami). Užívateľ zadáva príkazy a rozhranie na ne reaguje a vypisuje výsledky, alebo chybové hlásenia. Samotný *COMMAND.COM* obsahuje sadu vnútorných príkazov. Ostatné príkazy sa nazývajú vonkajšie a sú implementované ako programy s príponou *EXE*, *COM*, alebo *BIN*. Vonkajšie príkazy môžu mať (ako aj vnútorné) aj argumenty.

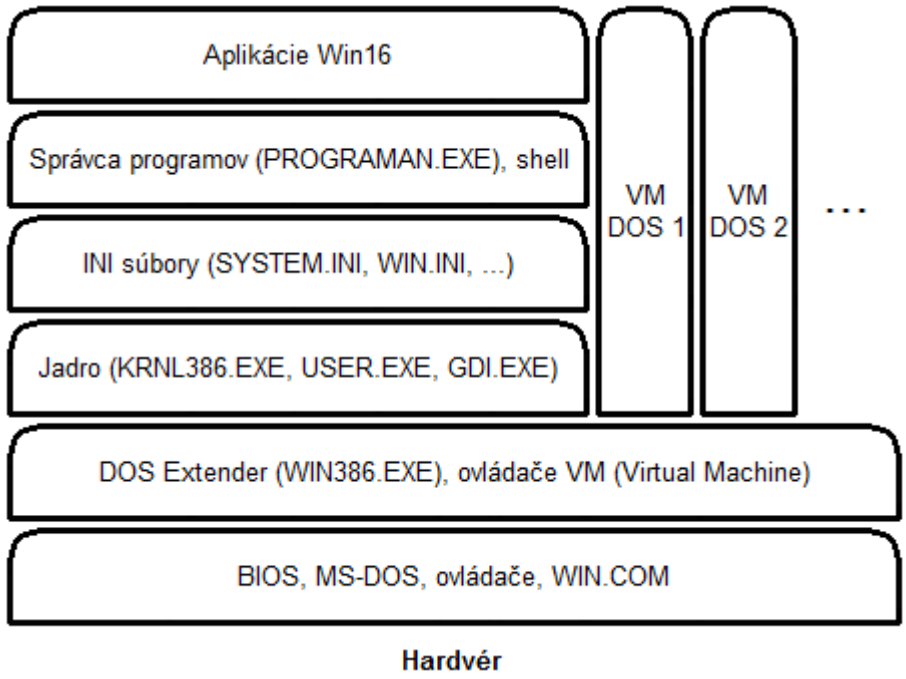
Posledná vrstva je určená k zjednodušeniu práce užívateľa. Okrem užívateľom spustených programov tu bežia tiež programy predstavujúce vnútorné príkazy a radíme tu tiež konfiguračné súbory, v ktorých si užívateľ môže určiť, ako má systém reagovať. Základné konfiguračné súbory sú dva a to *CONFIG.SYS* pre nastavenie hardvéru (napríklad spustenie určitých ovládačov pre monitor s určením znakovej sady pre slovenčinu) a *AUTOEXEC.BAT* pre nastavenie softvéru (tu sa určuje, ktoré programy alebo príkazy sa majú spustiť po štarte operačného systému).

Niektoré interné príkazy COMMAND.COM: BREAK, CHCP, CHDIR, CD, CLS, COPY, CTTY, DATE, DEL, ERASE, DIR, ECHO, EXIT, MKDIR, MD, PATH, PROMPT, REN, RENAME, RMDIR, RD, SET, TIME, TRUENAME, TYPE, VER, VERIFY, VOL.

8.2.2 Windows s jadrom operačného systému MS-DOS

Architektúra Windows 3.x:

Keď sa v operačnom systéme MS-DOS spustí Windows 3.x, štruktúra celého systému sa v hornej časti zmení. Na obrázku Obr. 8.4 je spodná časť trochu zhrnutá (BIOS, *MSDOS.SYS*). K nim je pridaný súbor WIN.COM, ktorý slúži k spusteniu systému Windows (tento súbor je vo všetkých Windows-och s jadrom MS-DOS), a ovládače. Windows už používa multitasking, 16-bitové knižnice (MS-DOS je 8-bitový) a vo verzií Windows 3.11 for Workgroups základnú podporu siete (iba sieť peer-to-peer).



Obr. 8.4 Štruktúra operačného systému Windows 3.x

Ovládače (driver-e) ovládajú periférne zariadenia pre Windows, ovládače priamo pre Windows sú spustené v súbore *SYSTEM.INI* pomocou príkazu DEVICE. Niektoré nové ovládače, ktoré sa dajú využívať aj v operačnom systéme MS-DOS sú v súbore *CONFIG.SYS*.

DOS Extender je modul pre podporu využitia rozšírenej pamäte (Extended Memory, viď 2.1), predstavuje ho súbor WIN386.EXE. Súčasťou tohto súboru je aj *Správca virtuálnych zariadení* (VMM – Virtual Machine Manager), ktorý ovláda možnosť pre súbeh Windows-u a DOS-u. *Ovládače virtuálnych zariadení* (VxD) sú ovládače, ktoré správca virtuálnych zariadení potrebuje pre manipuláciu so vstupno-výstupnými zariadeniami pre programy DOS-u.

V ďalšej vrstve je jadro Windows-u (pozor, jadrom operačného systému stále ostáva *MSDOS.SYS*), ktoré tu pracuje ako správca prostriedkov vzhľadom k programom bežiacim pod Windows-om. Skladá sa z troch súborov (častí):

- *KRNL386.EXE* – plní predovšetkým úlohu správcu pamäte a správcu procesov (riadenie prideľovania pamäte procesom, prideľovanie prostriedkov systému procesom,…),
- *GDI.EXE* – rozhranie grafického zariadenia (Graphic Device Interface), obsahuje funkcie pre vytváranie kurzoru, ikony, písma,…),

- *USER.EXE* – užívateľské ovládanie rozhrania, zdroje, ktoré nepatria *GDI.EXE* (dialógové okná, menu, okná,…).

Nasledujúca vrstva obsahuje konfiguračné súbory s príponou *INI*. Z nich sú najdôležitejšie *WIN.INI* (konfigurácia softvéru a nastavenia pre určitého užívateľa) a *SYSTEM.INI* (konfigurácia hardvéru). Súbory *INI* môžu mať aj rôzne programy systému Windows.

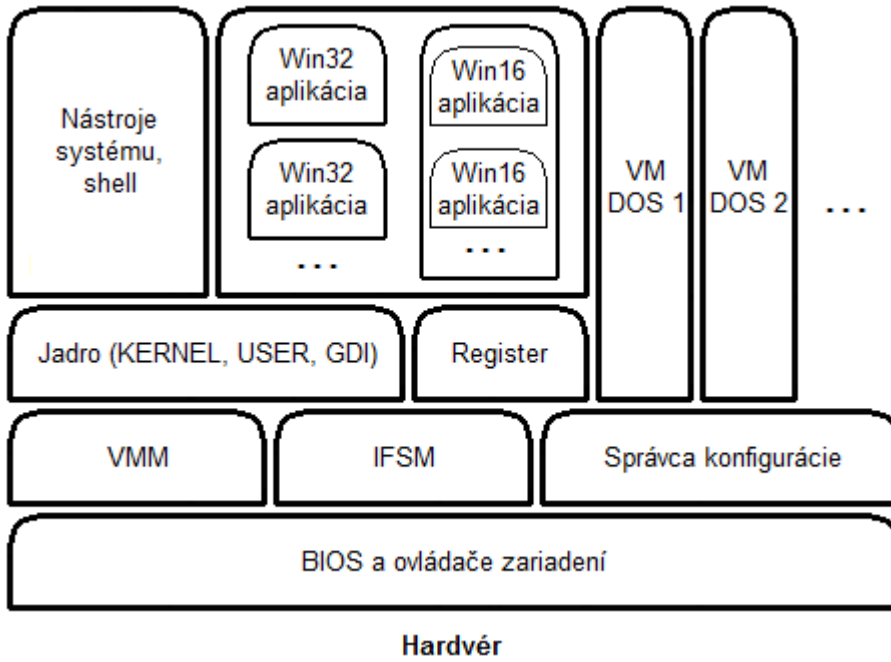
Nasleduje vrstva, ktorá je rozhraním medzi užívateľom, programami a samotným systémom. Súbor *PROGRAMAN.EXE* je správcom programov. *Shell* je grafické a textové rozhranie medzi užívateľom a systémom.

Najvyššiu vrstvu zastrešujú 16 bitové aplikácie (Win16), ktoré fungujú vďaka *API rozhraniu* (Application Programming Interface). Toto rozhranie je reprezentované *dynamicky linkovanými knižnicami* (obsahujú funkcie, objekty a podobne) a využívané procesmi pre prístup k systému. Knižnice celého *API* rozhrania sú však rozmiestnené v rôznych vrstvách, patria k nim aj súbory jadra (*KRNL386.EXE*, *USER.EXE* a *GDI.EXE*). Väčšina knižníc má však príponu *DLL*, ale niektoré majú aj príponu *EXE*, prípona knižníc fontov (tvar písma) zase závisí na type fontu (napríklad *TTF* pre true-type font), atď.

Všetky vrstvy platia pre 16-bitové aplikácie písané pre Windows 3.x, *MS-DOS programy* ani netušia o existencií jadra Windows a INI súboroch, preto vrchné vrstvy vôbec nevyužívajú. Keďže je samotný MS-DOS jedno-procesný systém (okrem ovládačov a rezidentných programov je spustený vždy iba jeden program), tieto programy sú napísané bez akýchkoľvek ohľadov na možnosti zdieľania pamäte s inými procesmi (keďže to ani nie je pri tejto architektúre možné). Preto je nutné separovať ich do virtuálnych počítačov, ktoré programom vytvoria ilúziu výlučnej existencie v systéme a znemožní im zásahy do prostriedkov iných procesov.

Architektúra Windows 4.x (Windows 95, 98, ME):

Na obrázku Obr. 8.5 je naznačená zjednodušená štruktúra tohto systému:



Obr. 8.5 Štruktúra operačného systému Windows 4.x (Windows 95, 98, ME)

Spodná vrstva opäť slúži k prístupu systému k zariadeniam. Nasledujúca vrstva sa taktiež vzťahuje k hardvéru, ale už na abstraktnej úrovni. Skladá sa z troch základných modulov:

- *VMM* je správca virtuálnych zariadení (Virtual Machine Manager), vytvára a udržuje prostredie virtuálnych zariadení,
- *IFSM* je správca inštalovateľných súborových systémov (Installable File System Manager), spravuje rôzne typy súborových systémov, ktoré sa dajú inštalovať, napr.: FAT16, FAT32, CDFS (systém súborov pre CD-ROM), atď.,
- *Správca konfigurácie* spravuje ovládače hardvéru na vyššej úrovni, predovšetkým zariadenia typu Plug&Play.

Jadro sa skladá z troch modulov, každý z nich má dve dynamicky linkované knižnice (jedna pre 16-bitové aplikácie s príponou *EXE*, druhá pre 32-bitové aplikácie s príponou *DLL*):

- *KERNEL* má na starosti multithreding, multitasking, správu pamäte, synchronizáciu objektov, vstupov a výstupov pri súboroch, atď.,
- *GDI* (Graphics Device Interface) spravuje tlač, spracovanie grafiky, základné grafické objekty,… (funkcie podobné ako pri Windows-e 3.x),
- *USER* spracováva vstupy z klávesnice, myši a podobne (riadené preruseniami), výstupy do užívateľského grafického rozhrania (okná, menu, ikony,…), prácu s čítačom/časovačom, atď.

Register je centrálna informačná databáza systému, nájde sa tu väčšinou to, čo vo Windows 3.x bolo v súboroch *INI* (tie sú však zachované pre spätnú kompatibilitu), ale samozrejme pre operačné systémy Windows 4.x a 32-bitové aplikácie. Fyzicky je register uložený v súboroch *SYSTEM.DAT* a *USER.DAT*.

Nástroje systému a *shell* majú podobný význam ako vrstva Správca programov a shell pri architektúre Windows-u 3.x (rozhranie medzi užívateľom, programami a samotným systémom, grafické a textové rozhranie medzi užívateľom a systémom, atď.).

Aplikácie Win32 (čiže pre systémy Windows 95 a vyššie) a *Win16* (pre nižšie verzie) bežia v systémovom virtuálnom počítači, každá *Win32* aplikácia ma vyhradený svoj vlastný adresný priestor, aplikácie *Win16* majú jeden spoločný priestor, ale platí pre ne to, čo vo Windows-e 3.x. *DOS aplikácie* majú rovnako ako vo Windows-e 3.x každá svoj virtuálny počítač (a preto má každá DOS aplikácia svoj vlastný adresný priestor, v rámci virtuálneho počítača).

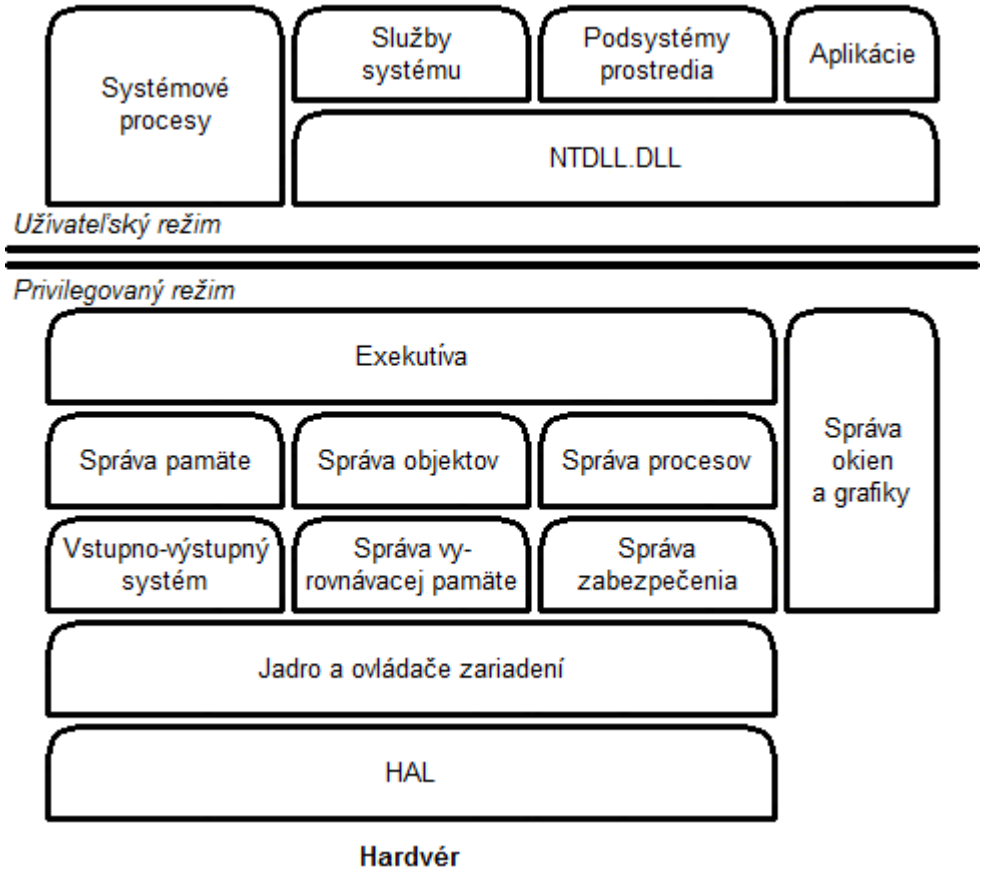
Windows s jadrom operačného systému MS-DOS je jedno-užívateľský a viac-procesný operačný systém (platí to pre obe architektúry Windows 3.x a 4.x).

8.2.3 Windows bez jadra operačného systému MS-DOS

Jadro operačných systémov rady NT (Windows NT, 2000, XP, Vista, 7, 8, 10) vznikalo nezávisle od systému MS-DOS, už pri návrhu bolo brané v úvahu typické použitie tohto systému ako servera alebo klienta v sieti, a taktiež hlavným hľadiskom bola stabilita a možnosti zabezpečenia. Systém bol navrhnutý ako viac-procesný (SMP – symetrický multiprocessing), viac-užívateľský multitaskový univerzálny sieťový systém. Nasledujúca štruktúra platí pre Windows NT 3.x, Windows NT 4.x, Windows NT 5.x (Windows 2000 a XP) a hlavná štruktúra ostala aj pri Windows 6.x (Windows Vista, 7, 8, 10).

Zjednodušená štruktúra systému je naznačená na obrázku Obr. 8.6. Niektoré prvky sú podobné častiam štruktúry Windows-u s jadrom MS-DOS-u, ale vnútorne pracujú inač.

Dôležité je predovšetkým rozdelenie do dvoch základných častí – časti bežiacej v privilegovanom režime (režime jadra) a časti bežiacej v užívateľskom režime.



Obr. 8.6 Štruktúra operačného systému Windows NT 3.x, 4.x, 5.x a 6.x (NT, 2000, XP, Vista, 7, 8, 10)

Privilegovaný režim:

HAL je vrstva abstrakcie hardvéru (Hardware Abstraction Layer), rozhranie medzi hardvérom a zvyškom jadra systému. Je riadená súborom *HAL.DLL*. Je oddelená od ostatných častí systému z dôvodu ľahšej prenositeľnosti systému (premiestnenie HDD s OS medzi rôznymi počítačmi). Ovládače komunikujú so zariadeniami iba sprostredkovane cez túto vrstvu.

Jadro je súčasťou súboru *NTOSKRNL.EXE* (zároveň s exekutívou). Tu sú obsluhované prerušenia, prevádza sa správa procesov (synchronizácia prideľovania procesorov), a podobne.

Exekutíva je riadiaci program operačného systému, má na starosti riadenie celého jadra bežiaceho v privilegovanom režime. Fyzicky je zároveň s jadrom súčasťou súboru *NTOSKRNL.EXE*.

Podsystémy (subsystémy) prostredia sú rozhrania zaisťujúce správny a bezpečný beh rôznych typov aplikácií. V týchto podsystémoch bežia aplikácie, ktoré tak nemusia byť kompatibilné s Windows NT. Podsystémy poskytujú aplikáciám rozhranie, ktoré prekladá komunikáciu (požiadavky na informácie, zdroje, prevedenie určitej akcie, a podobne) medzi aplikáciou a operačným systémom tak, aby si obe strany „rozumeli“. Sú tu napríklad podsystémy pre aplikácie písané pre Win32, Win16, DOS, OS/2, POSIX, atď.

Podsystém pre Win32 (vrátane NT) je predstavovaný súborom *CSRSS.EXE*, pre POSIX je to predovšetkým súbor *PXSS.EXE* (je to server podsystému). Podsystém Win32 je potrebný pre beh operačného podsystému, preto sa ako jediný spustí hneď po štarte počítača, ostatné podsystémy sú spustené až na žiadosť pri spustení aplikácie patriacej tomuto podsystému. Každý podsystém má okrem svojho riadiaceho programu (napríklad *CSRSS.EXE* pre Win32) tiež knižnice, v ktorých sú uložené funkcie a objekty, obsahujú API daného podsystému. Napríklad ku knižniciam Win32 patria taktiež knižnice *KERNEL32.DLL*, *USER32.DLL* a *GDI32.DLL*. Ich funkcie sú podobné ako v iných variantoch Windows-u, vnútorne však majú odlišnú štruktúru.

Modul pre *správu okien a grafiky* (GUI, Win32 User a GDI) je kód užívateľského rozhrania a rozhrania grafických zariadení pre podsystém Win32. Vo Windows-och NT od tretej verzie bola táto časť kódu presunutá do režimu jadra z dôvodov urýchlenia práce aplikáciám, ktoré veľmi často využívajú grafické zariadenia. Tento modul je určený pre podsystém Win32, ale aby nebolo nutné tieto funkcie implementovať v každom podsystéme zvlášť, je prekladané volanie grafických funkcií iných podsystémov na volanie v podsystéme Win32.

Umiestnenie kódu grafického rozhrania do režimu jadra je neobvyklé. Nevýhodou tohto postupu je však väčšie bezpečnostné riziko a riziko porušenia stability systému pri chybnej práci tohto modulu (pracuje v režime jadra, preto má prístup do pamäte systémových procesov). Ďalšou nevýhodou je náročnejší postup výmeny užívateľského rozhrania za alternatívny.

Užívateľský režim:

Súbor *NTDLL.DLL* predstavuje rozhranie medzi bežnými procesmi a systémom. Pokiaľ niektorý proces v užívateľskom režime volá službu bežiacu v privilegovanom režime, volanie ide vždy cez tento súbor, aby sa vylúčila možnosť modifikácie systémových knižníc a ďalších systémových zdrojov. *NTDLL* predstavuje tzv. *dokumentované rozhranie* systému, ktoré každému procesu sprostredkováva komunikáciu s daným podsystémom.

Systémové procesy sú procesy, ktoré spúšťa systém, napríklad procesy zaisťujúce užívateľské prostredie. V zobrazení *Správcu úloh* v karte *Procesy*, je pri zobrazovaní užívateľov hodnota SYSTEM, LOCAL SERVICE, NETWORK SERVICE a podobne. Pracujú v užívateľskom režime z dôvodu bezpečnosti, k častiam systému pracujúcim v privilegovanom režime však majú trochu jednoduchší prístup oproti ostatným procesom v užívateľskom režime.

Služby systému (serverové procesy) sú služby poskytované systémom, zoznam sa dá nájsť napríklad v nástroji *Nástroje pre správu – Služby*. Služby sú systémové procesy bežiace často aj bez prihlásenia užívateľa, je to obdoba rezidentných programov v operačnom systéme MS-DOS. Beh služieb zaisťuje proces ovládača služieb predstavovaný súborom *SERVICES.EXE*.

Popis vrstvenia architektúry:

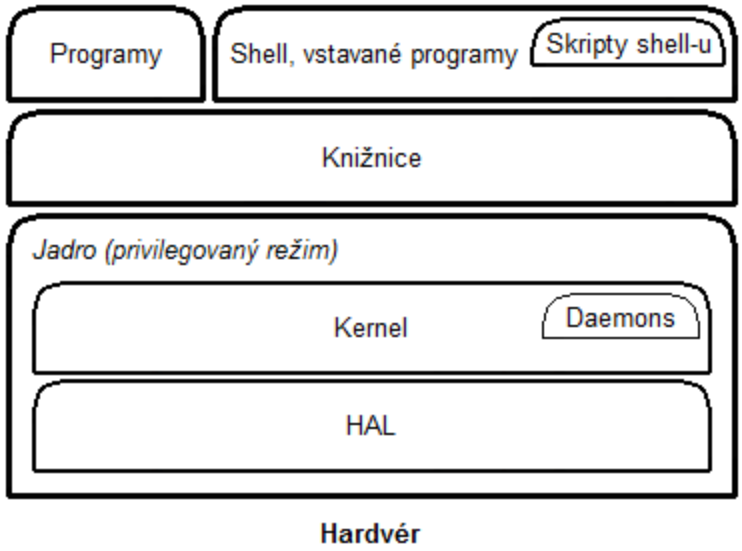
Ako je vidno na obrázku Obr. 8.6, Windows NT nie je prísne vrstvený systém, ale kombinuje viac rôznych architektúr pre svoje rôzne časti. Sú to tieto architektúry:

1. *Vrstvená architektúra* sa uplatňuje v jadre vo vrstve *HAL* a *I/O systéme*.
2. *Modulárna architektúra* zahŕňa uzavreté moduly, vnútorne kompaktné, ktoré poskytujú služby cez nadefinované rozhranie, komunikácia prebieha voľne medzi rôznymi modulmi, túto architektúru využíva *exekutíva* pri riadení *správy procesov*, *správy pamätí*, *I/O systému*, atď. (modulov bežiacich v privilegovanom režime).
3. *Architektúra klient-server* sa uplatňuje v *API*, čo je množina dynamicky linkovaných knižníc, tu považovaných za servery, procesy z vyšších vrstiev (klienti) využívajú ich služieb (cez knižnicu *NTDLL.DLL*).

8.2.4 UNIX

Väčšina Unixových systémov má veľmi podobnú štruktúru (okrem tých, ktoré boli upravené pre použitie v reálnom čase – operačné systémy reálneho času). Jadro beží v privilegovanom režime (v režime jadra), je často tvorený jediným súborom. Preto sa nazýva monolitické, aj keď jeho vnútorná štruktúra je presne rozvrhnutá, napríklad pri Linuxe je to súbor */boot/vmlinuz*.

Unixové systémy sú viac-procesné viac-užívateľské maltitaskové univerzálne sieťové systémy už od svojho počiatku. Na to bol braný zreteľ už pri navrhovaní ich štruktúry (Obr. 8.7), preto za dlhé desaťročia existencie Unixov ani nebolo treba ich výrazne meniť. Táto štruktúra tiež zrejme poslúžila ako jeden zo vzorov pri návrhu štruktúry Windows NT.



Obr. 8.7 Štruktúra operačných systémov UNIX

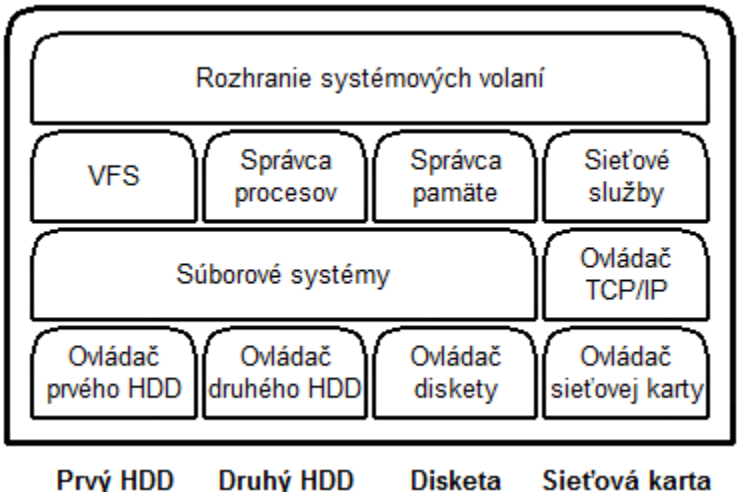
Jadro systému beží v privilegovanom režime (kernel mode, všetko ostatné beží v užívateľskom režime – user mode) a skladá sa z dvoch oddelených častí:

- *HAL* je časť jadra závislá na hardvéry, sú tu predovšetkým ovládače zariadení,
- *Kernel* je jadro nezávislé na hardvéry, bežia tu tiež takzvaný *démoni* (deamons) – systémové procesy obdobné službám vo Windows-e NT, bežia na pozadí často bez ohľadu na beh užívateľských procesov a prihlasovanie či odhlasovanie užívateľov.

Knižnice v Unixe majú podobnú rolu ako DLL vo Windows-e, teda obsahujú objekty a rôzne rutiny. Mnohé z nich sú súčasťou grafického subsystému (napr.: X Window).

Shell je rozhranie pre komunikáciu s užívateľom. Unixové systémy obvykle ponúkajú viac rôznych shell-ov. Komunikácia prebieha v textovej forme (užívateľ zadá príkazy a systém reaguje textovými výpismi), ale súčasné Unixové systémy väčšinou majú tiež veľmi prepracované grafické rozhrania a bežný užívateľ s textovým shell-om ani nemusí prísť do styku. Rovnako ako vo Windows-e príkazy sa môžu zbierať do textových súborov, ktoré sa nazývajú skripty.

Vrstva *HAL* je tvorená predovšetkým ovládačmi periférnych zariadení. Nasleduje vrstva, ktorá pristupuje k zariadeniam na abstraktnej úrovni – sú tu ovládače súborových systémov na pamäťových médiách a ovládač TCP/IP komunikujúci s ovládačom sieťovej karty. Štruktúra jadra Unixových systémov je na nasledujúcom obrázku (Obr. 8.8):



Obr. 8.8 Štruktúra jadra operačných systémov UNIX

Súborový systém je rozhranie medzi ovládačom vonkajšieho pamäťového média a vyššími vrstvami jadra. V Unixových systémoch sú súborové systémy implementované priamo v jadre vrátane podpory pridávania nových súborových systémov. Pretože v Unixoch platí, že všetko je súbor, sú tu nie len súborové systémy pre vonkajšie pamäťové média, ale aj ďalšie, abstraktné, sprostredkujúce prístup k informáciám o momentálnom stave systému, konfigurácií a podobne (napr. Linux s jadrom 2.4 má tieto informácie v */proc*). Takže v Unixe netreba pristupovať na niektoré porty vstupno-výstupných zariadení, ale prepísať hodnotu vo virtuálnom súbore a ta sa prejaví na vstupno-výstupnom zariadení. Taktiež dokážu tieto súborové systémy združiť iné súborové systémy (v spomínanom Linuxe sú združené systémy v */etc*).

VFS (Virtual File System, virtuálny súborový systém) predstavuje rozhranie pre podobný spôsob prístupu k rôznym súborovým systémom. Všetky súborové systémy združuje v jedinej stromovej štruktúre. Pokiaľ užívateľ chce s konkrétnym súborovým systémom pracovať, pripojí ho na stanovené miesto do tejto štruktúry a tým ho sprístupní (pripojovanie je zvyčajne automatizované, užívateľ sa o to nemusí starať). Dôležitou funkciou VFS je zaistenie rovnakého spôsobu zaobchádzania s dátami, či už sa nachádzajú v akomkoľvek súborovom systéme, teda užívateľ sa nemusí starať o fyzické umiestnenie súboru, atribúty (napr.: nastavenie času posledného prístupu k súboru), konvencie pri práci so súborom (ako oznámiť súborovému systému, že chce otvoriť určitý súbor, a podobne).

Rozhranie systémových volaní je rozhranie medzi jadrom a čímkoľvek, čo môže priamo ovplyvniť užívateľ (programy, príkaz shell-u, skripty). S touto vrstvou sa dá komunikovať cez knižnice obsahujúce definície API funkcií (obdoba WinAPI a Win32API vo Windows-och). Hlavnou úlohou je zaistenie bezpečnosti, znemožnenie zásahu užívateľa do jadra (tu sa Microsoft inšpiroval pri programovaní knižnice *NTDLL.DLL*, ktorá má podobnú funkciu). Je to zjednodušenie aj pre programátora, ktorý sa nemusí zaoberať jednotlivými systémovými volaniami a môže priamo používať API funkcie.

8.3 Vytváranie aplikácií pod operačným systémom Windows NT

Dnes sa pri programovaní aplikácií pod operačnými systémami Windows NT najčastejšie používa framework .NET, ktorý všetku tu ťažkú prácu pri programovaní WinAPI (Win32API) spraví za programátora. Programovanie v jazyku C# si vyžaduje prácu so spomínaný framework-om, ale pri programovaní v C++ si programátor môže vybrať, či použije framework, alebo sa pustí do programovania priamo pomocou WinAPI. Framework .NET je súčasťou každého Windows-u NT od verzie 5.0 (v základnom balíku Windows 2000 nie je, ale je v aktualizáciách systému), s týmto framework-om vyšiel aj nový programovací jazyk C#. Takže aj väčšina programátorov C++ si radšej vyberie .NET, keď ide programovať aplikáciu pre Windows NT. Lenže tento framework dosť spomaľuje výkonnosť naprogramovanej aplikácie, pretože prilinkuje a používa veľké množstvo funkcií, ktoré by možno aplikácia ani nepotrebovala. Programátori framework-u .NET chceli uľahčiť prácu programátorovi, tak jednému objektu pridali nespočetne množstvo vlastností (stačí sa pozrieť do vlastností/properties jednotlivých objektov framework-u), aby nemuseli napríklad pre textové pole vytvárať 10 či 20 objektov, tak vytvorili len *Text Box*, *Rich Text Box* a všetky vlastnosti sa musia pomocou WinAPI nastaviť (a tie čo by programátor pre svoju aplikáciu nepotreboval). Práve tento prípad ešte nezahadzuje programovanie pomocou WinAPI, ak programátor potrebuje rýchlu aplikáciu (napr.: regulácie, merania pri vysokých vzorkovacích frekvenciách).

Ďalší zásadný rozdiel medzi programovacími jazykmi C# a C++ je ten, že kompiláciou kódu C++ dostaneme strojový kód, pokiaľ pri kompilácií kódu C# iba medziprekladový jazyk MSIL (Microsoft Intermediate Language), ktorý musí framework ešte za behu programu prekladať do strojového kódu (ďalšie spomalenie kódu). Táto skutočnosť sa dá všimnúť pri kompilácií rovnakého programu v C++ a v C# pri využití framework-u .NET. Program naprogramovaný v C++ ma rádovo väčšiu veľkosť ako program v C#, pretože kompilátor C++ prilinkoval kódu všetky potrebné veci z framework-u, aby mohol fungovať samostatne (len pomocou svojho strojového kódu), pričom kompilátor C# len prepísal kód do MSIL a až za behu programu bude volať potrebné funkcie.

Programátori, ktorý sa do programovania WinAPI nehrnú, tak sa v tejto podkapitole dozvedia ako sa pristupuje a pracuje s objektmi jadra operačného systému a tých ktorých to zaujíma sa dozvedia aspoň nejaké tie základy.

Hrubý popis Win32 API je v predchádzajúcej podkapitole. Win32 je názov aplikačného programového rozhrania (API). Rozhranie Win32 API obsahuje množinu funkcií, ktoré sa môžu v programovanej aplikácií volať. Keď sa vytvára aplikácia pre rozhranie Win32, tak sa volajú funkcie tohto rozhrania. Rozhranie Win32 API definuje množinu funkcií, ktoré aplikácie môžu volať a zároveň definuje chovanie týchto funkcií.

Pri programovaní pomocou WinAPI v programovacom jazyku C++ je potrebné poznať jadro operačného systému a hlavne jeho objekty. Programátor v rozhraní Win32 bude objekty jadra pravidelne vytvárať, rušiť a pracovať s nimi. Systém vytvára a spravuje niekoľko typov objektov jadra, medzi ne patria aj tieto:

1. Procesy,
2. Mapovanie súborov,
3. Semafory,
4. Mutexy,
5. Vykonávacie toky,
6. Súbory,
7. Poštové priehradky,
8. Rúry,
9. Udalosti.

8.3.1 Procesy

Proces je definovaný ako inštancia bežiaceho programu. V rozhraní Win32 vlastní každý proces svoj adresný priestor o veľkosti 4 GiB. Procesy rozhrania Win32 sú inertné to znamená, že proces v rozhraní Win32 nič nevykonáva, iba vlastní svoj 4 GiB adresný priestor, v ňom je uložený kód a dáta *EXE* súboru aplikácie. Všetky dynamické knižnice, ktoré *EXE* súbor potrebuje, majú svoje dáta a svoj kód uložený taktiež v tomto adresnom priestore. Okrem adresného priestoru môže proces vlastniť aj rôzne prostriedky, ako sú súbory, dynamicky alokovaná pamäť alebo vykonávacie toky. Všetky možné prostriedky zabrané v dobe „života“ procesu systém zaručene automatický uvoľní po skončení procesu.

Keďže proces je inertný, tak aby mohol proces čokoľvek vykonávať, musí vlastniť vykonávací tok. Tento tok je zodpovedný za vykonávanie kódu, uloženého v adresnom priestore procesu. V skutočnosti môže jeden proces obsahovať viac vykonávacích tokov, ktoré všetky „súčasne“ vykonávajú kód v adresnom priestore procesu. Za týmto účelom má každý tok svoju vlastnú sadu registrov procesora (procesorov) a svoj vlastný zásobník. Každý proces má najmenej jeden vykonávací tok, ktorý vykonáva jeho kód. Pokiaľ by neexistoval žiaden tok vykonávajúci kód v adresnom priestore procesu, nebol by žiaden dôvod ospravedlňujúci existenciu procesu a systém by takýto proces automaticky zrušil, vrátane jeho adresného priestoru.

Aby mohli všetky vykonávacie toky pracovať, operačný systém prideľuje čas procesoru jednotlivým tokom. Operačný systém vytvára ilúziu súčasného behu všetkých procesov tým, že krátke časové úseky procesoru (tzv. *kvantá*) prideľuje jednotlivým tokom neustále dookola.

Keď sa v rozhraní Win32 vytvára proces, vytvorí systém automaticky aj jeden vykonávací tok, nazývaný primárny tok. Primárny tok potom môže vytvárať ďalšie vykonávacie toky a tie môžu ďalej vytvárať ďalšie toky.

Windows-y NT (3.1, 3.5, 3.51, 4.0, XP, ..., 10) sú samozrejme schopné pracovať aj s počítačmi, ktoré obsahujú viacero procesorov, takže ak používate procesor s 8 vláknami (Intel i7), tak 8 vykonávacích tokov sa ozaj budú vykonávať naraz, ale samozrejme v operačnom systéme beží rádovo viac vykonávacích tokov takže platí spomenuté prideľovanie procesorového času. Prideľovanie jednotlivých procesorov vykonávacím tokom má na starosti jadro operačného systému, programátor nemusí nijak upravovať kód. Prvé verzie Windows-ov až po Windows ME (okrem Windows NT) neboli schopné pracovať s viacerými procesormi naraz, takže ak boli nainštalované na počítači s viacerými procesormi (jadrami), tak fungoval iba jeden procesor ostatné sa nevyužívali.

Proces sa vytvára volaním funkcie `CreateProcess()`, ktorá aj vytvorí primárny vykonávací tok.

8.3.2 Vykonávacie toky

Vykonávací tok vykonáva spracovanie kódu procesu. Pri inicializácií procesu mu systém vždy pridelí primárny tok. Vykonávací tok začína v spúšťacom kóde runtime-ovej knižnice jazyka C, ktorá volá funkciu `WinMain()`.

Tok ďalej spracováva túto funkciu až do jej ukončenia, a potom sa vracia do runtime-ového kódu, ktorý volá funkciu:

```
ExitProcess().
```

Pri mnohých aplikáciách je primárny vykonávací tok zároveň aj jediným vykonávacím tokom aplikácie. Procesy však môžu vytvárať aj dodatočné vykonávacie toky,

pokiaľ ich potrebujú pri svojej práci. Celý zmysel vytvárania dodatočných tokov spočíva v tom, aby bol čo najviac (čo najefektívnejšie) využitý procesorový čas.

Funkcia, ktorou sa vytvorí primárny vykonávací tok, bola spomenutá na konci predchádzajúce podkapitoly, dodatočný vykonávací tok sa vytvára volaním funkcie:

```
CreateThread().
```

Príklady využitia sekundárnych tokov v operačnom systéme:

Príkladom je keď Microsoft Office Excel potrebuje po každej zmene dát spraviť prepočet tabuľky. Pretože prepočet celej tabuľky môže trvať pomerne dlho (užívateľ by si toho mohol všimnúť), tak aplikácia v primárnom toku nebude prepočítavať tabuľku stále keď do nej užívateľ zasiahne, na to využije vykonávací tok s nižšou prioritou. Tento tok s nižšou prioritou sa spustí iba vtedy, ak užívateľ spravil nejakú zmenu v tabuľke.

Podobným príkladom je Microsoft Office Word, ktorý využíva sekundárny tok na prestránkovanie dokumentu. Word potrebuje prestránkovať dokument stále, keď do neho užívateľ zasiahne. Primárny tok bude zodpovedný za spracovanie užívateľských vstupov a sekundárny tok bude na pozadí riadiť rozdelenie dokumentu na stránky.

Ďalším príkladom je kopírovanie súborov, kde užívateľ vidí dialógové okno priebehu kopírovania s možnosťou zrušenia tohto kopírovania. Pokiaľ užívateľ v priebehu kopírovania klikne na tlačidlo „Zrušiť“ operácia sa preruší, vďaka vykonávaciemu toku, ktorý na túto udalosťou zareaguje.

Samozrejme existuje mnoho ďalších príkladov, ale pre základnú predstavu to stačí.

8.3.3 Zdieľanie objektov jadra medzi procesmi

Často potrebujú vykonávacie toky rôznych procesov zdieľať objekty jadra medzi sebou. Môžu k tomu mať napríklad niektorý z nasledujúcich dôvodov:

- objekty mapovania súborov dovoľujú zdieľať bloky dát medzi dvoma procesmi, bežiacim na rovnakom počítači,
- poštové priehradky a rúry umožňujú posielať po sieti bloky dát medzi procesmi, ktoré bežia na rôznych počítačoch v sieti,
- mutexy, semafory a udalosti umožňujú vykonávacím tokom rôznych procesov vzájomne synchronizovať svoj beh, jedna aplikácia napríklad potrebuje oznámiť inej aplikácii, že už svoju prácu ukončila.

Objekty jadra podliehajú mechanizmom zabezpečenia a skôr než môže proces s objektom manipulovať, musí k tomu získať povolenie. Ten, kto objekt vytvoril, môže ľahko zabrániť neautorizovanému užívateľovi v prístupe k tomuto objektu, stačí jednoducho prístup zakázať.

Exitujú tri rôzne mechanizmy, ktoré umožňujú zdieľať objekty jadra medzi rôznymi procesmi:

- dedenie objektu `HANDLE`,
- pomenovanie objektov,
- duplikácia objektu `HANDLE`.

8.3.4 Synchronizácia vykonávacích tokov

Synchronizácia vykonávacích tokov by mala vyzerať tak, že jeden vykonávací tok synchronizuje svoj beh s iným tokom tým, že sám seba „uspí“. V dobe, keď vykonávací

tok spí, mu operačný systém neprideľuje procesorový čas, takže beh toku je pozastavený. Tesne pred uspatím však vykonávací tok oznamuje, čo sa musí stať, aby systém prebudil tok.

Operačný systém túto požiadavku toku zaregistruje a sleduje, kedy sa stane to, na čo tok čaká. Ako náhle očakávaná udalosť nastane, bude možné toku normálne prideľovať čas procesora. Časom teda dôjde k prideleniu procesora prebudenému toku a tok pokračuje vo svojej činnosti – jeho beh bol synchronizovaný s výskytom nejakej špeciálnej udalosti.

Pokiaľ by v systéme neexistovali synchronizačné objekty a systém by nepodporoval sledovanie špeciálnych udalostí, tak by sa tok musel o svoju synchronizáciu postarať sám. Napríklad tak, že by sa vôbec neuspal, ale čakal by informáciu od iného toku, že môže pokračovať, napríklad pomocou globálnej premennej. Nakoľko synchronizačné objekty existujú, tak sa táto metóda neodporúča, pretože spomaľuje výpočtový čas procesora, nakoľko vykonávaciemu toku systém stále prideľuje procesorový čas (hoc len čaká na nejakú podmienku).

Vykonávací tok sa môže synchronizovať pomocou:

- kritických sekcií,
- mutexov,
- semaforov,
- udalostí,
- oneskorenia časovača,
- asynchrónny súborový vstup a výstup.

8.3.5 Kritické sekcie

Kritické sekcie sú malé časti kódu, ktoré pred svojím prebehnutím vyžadujú výhradný prístup k nejakým zdieľaným dátam. Kritické sekcie sa používajú zo všetkých synchronizačných objektov najľahšie, umožňujú však synchronizovať vykonávacie toky iba v rámci jedného procesu. Kritické sekcie dovoľujú, aby k danej oblasti dát mal v danom okamžiku prístup iba jeden vykonávací tok.

8.3.6 Mutex

Mutexy sa veľmi podobajú kritickým sekciám s tou výnimkou, že sa dajú použiť k synchronizácií dátového prístupu vo viacerých procesoch. Aby bolo možné mutex použiť, musí ho najprv jeden proces vytvoriť volaním funkcie `CreateMutex()`:

```
HANDLE CreateMutex(LPSECURITY_ATTRIBUTES lpsa,
  BOOL fInitialOwner, LPTSTR lpszMutexName);
```

Parameter `lpsa` je ukazovateľ na štruktúru `SECURITY_ATTRIBUTES`. Parameter `fInitialOwner` udáva, či vykonávací tok, ktorý mutex vytvára, má byť zároveň počiatočným vlastníkom mutexu. Hodnota *true* znamená, že vykonávací tok sa stáva majiteľom mutexu a mutex sa teda bude nachádzať v nesignalizovanom stave. Akýkoľvek tok čakajúci na tento mutex bude pozastavený až do doby, keď vytvárajúci tok mutex uvoľní. Pokiaľ parametrom `fInitialOwner` bude hodnota *false*, znamená to, že vytvorený mutex nebude vlastnený žiadnym vykonávacím tokom, a preto bude od počiatku v signalizovanom stave. Prvému toku, ktorý na tento mutex čaká, bude mutex okamžite pridelený a bude môcť pokračovať v práci.

Parameter `lpszMutexName` môže mať buď hodnotu *null*, alebo môže obsahovať adresu ukončenú nulovým znakom (ukončený reťazec) s menom mutexu. Keď aplikácia volá

funkciu `CreateMutex()`, systém vytvorí objekt jadra typu mutex a pridelí mu meno, zadané spomínaným parametrom. Pomocou tohto mena je možné mutex zdieľať medzi viacerými procesmi. Funkcia `CreateMutex()` vráti hodnotu `HANDLE` volajúcemu procesu, ktorá identifikuje novo vytvorený mutex.

Jeden z najvýznamnejších rozdielov medzi mutexami a kritickými sekciami spočíva v tom, že mutexy dovoľujú synchronizovať vykonávacie toky vo viacerých nezávislých procesoch. Za týmto účelom musí každý takýto proces získať vlastný `HANDLE` tohto mutexového objektu. Proces môže `HANDLE` získať niekoľkými spôsobmi. Najjednoduchší spôsob je, že vykonávacie toky všetkých procesov, ktoré sa chcú synchronizovať zavolajú funkciu `CreateMutex()` a ako parameter `lpszMutexName` zadajú všetky toky rovnaké meno. Prvý tok, ktorý funkciu zavolá, spôsobí, že systém vytvorí nový objekt jadra typu mutex. Keď funkciu `CreateMutex()` zavolajú ďalšie toky, systém zistí, že mutex zadaného mena už existuje. Výsledkom bude, že systém nevytvorí nový objekt, ale vráti hodnotu `HANDLE` už existujúceho mutexu, ktorý sa vzťahuje k volajúcemu procesu.

Vykonávací tok môže zistiť, či funkcia `CreateMutex()` skutočne vytvorila nový objekt tak, že hneď po návrate hodnoty `HANDLE` zavolá funkciu `GetLastError()`. Pokiaľ funkcia vráti hodnotu *ERROR_ALREADY_EXISTS*, znamená to, že nebol vytvorený nový objekt. Pokiaľ bude mutex zdieľaný s inými procesmi je táto kontrola bezpredmetná.

Ďalší spôsob, ako získať HANDLE existujúceho mutexu, spočíva vo volaní funkcie `OpenMutex()`:

```
HANDLE OpenMutex(DWORD fdwAccess, BOOL fInherit,
                 LPTSTR lpszName);
```

Parameter `fdwAccess` môže nadobúdať buď hodnotu *SYNCHRONIZE*, alebo *MUTEX_ALL_ACCESS*. Parameter `fInherit` udáva, či môžu procesy, ktoré budú volať túto funkciu neskôr získať hodnotu `HANDLE` tohto mutexu. Parameter `lpszName` obsahuje meno mutexu (s ukončovacím nulovým znakom), ktorého `HANDLE` sa tok pokúša získať.

Po volaní funkcie `OpenMutex()` systém najprv prehľadá všetky existujúce mutexy, ak systém nájde mutex so zadaným menom nastaví jeho parametre podľa parametrov funkcie (`fdwAccess`, `fInherit`) a vráti hodnotu `HANDLE` pre daný objekt mutexu. Ak systém nenájde mutex so zadaným menom vráti `HANDLE` s hodnotou *null*.

Obe spomenuté metódy predpokladajú, že sa pracuje s pomenovaným mutexom. Okrem toho existujú ešte dve ďalšie metódy, ktoré nevyžadujú pomenovanie objektu. Jedna používa funkciu `DuplicateHandle()` a druhá využíva dedenie `HANDLE`.

Mutex zatvoríme pomocou funkcie `CloseHandle()`.

8.3.7 Semafor

Objekty jadra typu semafor sa typický používajú k sledovaniu počtu voľných systémových prostriedkov. Semafor dáva vykonávaciemu toku možnosť zistiť počet dostupných systémových prostriedkov. Pokiaľ je požadovaný systémový prostriedok voľný, automaticky dôjde k dekrementácii počítadla voľných prostriedkov. Táto akcia testovania a dekrementácie je v rámci semaforu atomická, čo znamená, že ako náhle je semafor požiadaný o nejaký systémový prostriedok, operačný systém zistí, či je prostriedok dostupný a dekrementuje počítadlo voľných prostriedkov bez toho, aby do situácie mohol iný vykonávací tok zasiahnuť. Takže až po znížení hodnoty počítadla systém umožní, aby o rovnaký prostriedok mohol žiadať iný vykonávací tok.

Napríklad v počítači sú tri sériové porty. V danom okamžiku môžu so sériovým portom pracovať maximálne tri vykonávacie toky. Je to presne situácia, ktorá sa priamo ponúka k použitiu semaforu. Použitie sériových portov sa bude sledovať pomocou semaforu, jeho počítadlo bude mať na začiatku hodnotu 3 – jedna pre každý sériový port. Pokiaľ bude počítadlo prostriedkov väčšie ako nula, semafor bude signalizovaný. Ako náhle bude počítadlo rovné nule, semafor nebude signalizovaný. Hodnota počítadla nemôže nikdy klesnúť pod nulu. Stále, keď vykonávací tok volá funkciu `WaitForSingleObject()` a predáva hodnotu `HANDLE` semaforu, systém zistí, či je počítadlo voľných prostriedkov tohto semaforu väčší než nula. Pokiaľ je, systém dekrementuje počítadlo prostriedkov a umožní volajúcemu toku pokračovať v práci. Pokiaľ má počítadlo v okamihu volania funkcie `WaitForSingleObject()` nulovú hodnotu, systém uspí vykonávací tok až do doby, keď iný tok uvoľní semafor (inkrementuje tým počítadlo voľných prostriedkov).

Semafor sa vytvorí volaním funkcie `CreateSemaphore()`:

```
HANDLE CreateSemaphore (LPSECURITY_ATTRIBUTES lpsa,
LONG cSemInitial, LONG cSemMax, LPTSTR lspzSemName);
```

Funkcia vytvorí semafor, ktorého maximálna hodnota počítadla voľných prostriedkov je zadaná parametrom `cSemMax`. Pre predchádzajúci príklad by to bola hodnota 3. Parameter `cSemInitial` dovoľuje určiť inicializačnú hodnotu voľných prostriedkov. V predchádzajúcom príklade by to bola opäť hodnota 3, pretože pri spustení systému sú všetky porty voľné. Posledný parameter `lpszSemName` umožňuje vytvoriť pomenovaný semafor. Meno semaforu je potom možné použiť v iných procesoch k získaniu hodnoty `HANDLE` tohto semaforu pomocou funkcie `CreateSemaphore()` alebo `OpenSemaphore()`:

```
HANDLE OpenSemaphore(DWORD fdwAccesss, BOOL fInherit,
                     LPTSTR lpszName);
```

Činnosť funkcie je rovnaká ako pri funkcií `OpenMutex()` (viď podkapitolu 8.3.6).

Semafor uvoľňuje (inkrementuje počítadlo voľných prostriedkov) volaním funkcie `ReleaseSemaphore()`:

```
BOOL ReleaseSemaphore(HANDLE hSemaphore, LONG cRelease,
                      LPLONG lplPrevious);
```

Parameter `hSemaphore` je hodnota `HANDLE` dané semaforu. O koľko sa má hodnota počítadla zvýšiť určuje parameter `cRelease`. Parameter `lplPrevious` je ukazovateľ na premennú typu *long*, do ktorej funkcia `ReleaseSemaphore()` uloží hodnotu počítadla voľných prostriedkov pred zvýšením o hodnotu cRelease. Pokiaľ je pre programátora tento údaj bezpredmetný, môže zadať na mieste parametra `lplPrevious` hodnotu null.

Semafor sa zatvára pomocou funkcie: `CloseHandle(HANDLE hSemaphore).`

8.3.8 Udalosť

Udalosti (Events) sú zo všetkých synchronizačných objektov najjednoduchšie a dosť sa líšia od mutexov a semaforov. Mutexy a semafory sa obvykle používajú k riadeniu prístupu k nejakým dátam, udalosti slúžia k oznámeniu ukončenia nejakej činnosti. Existujú

dva rôzne druhy objektov udalostí: ručne resetované udalosti a automatický resetované udalosti. Ručne resetované udalosti sa používajú vtedy, keď je nutné skončenie nejakej operácie oznámiť viacerým vykonávacím tokom súčasne. Automaticky resetované udalosti slúžia k upovedomeniu iba jedného vykonávacieho toku.

Najčastejšie sa udalosti používajú v situáciách, keď jeden tok robí nejakú inicializáciu a po jej skončení oznámi inému toku, že môže spraviť svoju vlastnú prácu. Inicializačný tok nastaví udalosť do nesignalizovaného stavu a zaháji inicializáciu. Po skončení inicializácie tok prepne udalosť do signalizovaného stavu. Výkonný vykonávací tok je hneď po svojom spustení pozastavený a čaká na signalizáciu udalostí. Až inicializačný vykonávací tok udalosť signalizuje, výkonný vykonávací tok sa prebudí a môže spraviť potrebné operácie.

Proces môže napríklad obsahovať dva vykonávacie toky. Prvý vykonávací tok načíta dáta z disku do nejakého pamäťového buffer-a. Po načítaní dát bude prvý tok signalizovať druhému, že môže začať spracovávať dáta. Ak druhý tok spracovanie dokončí, môže signalizovať o tom informáciu späť prvému toku a tým žiadať o ďalší blok dát.

Udalosť sa vytvára volaním funkcie `CreateEvent()`:

```
HANDLE CreateEvent(LPSECURITY_ATTRIBUTES lpsa,
BOOL fManualReset, BOOL fInitialState, LPTSTR lpszEventName);
```

Parameter `fManualReset` je hodnota typu *bool*, ktorá hovorí, či vytvorený objekt bude ručne resetovaná udalosť (*true*), alebo automatický resetovaná udalosť (*false*). Parameter `fInitialState` určuje počiatočný stav udalosti. Signalizovaný stav má hodnotu *true* a nesignalizovaný stav hodnotu *false*. Po vytvorení objektu udalosti funkcia `CreateEvent()` vráti `HANDLE` pre vytvorený objekt udalosť. Vykonávacie toky iných procesov môžu `HANDLE` vytvorenej udalosti získať volaním funkcie `CreateEvent()` s rovnakou hodnotou parametra `lpszEventName`, pomocou dedenia, pomocou funkcie `DuplicateHandle()`, alebo pomocou funkcie `OpenEvent()`, ktorej parameter `lpszName` sa bude zhodovať s menom, zadaným pri volaní funkcie `CreateEvent()` v inom toku:

```
HANDLE OpenEvent(DWORD fdwAccess, BOOL fInherit,
                 LPTSTR lpszName);
```

Tak ako aj pred tým, aj udalosti sa zatvárajú pomocou funkcie `CloseHandle()`.

8.3.9 Rúry a poštové priehrady

Tieto objekty slúžia na prenos dát medzi procesmi. Z pohľadu programátora sa tieto objekty správajú ako vstupno-výstupné zariadenia (viď 8.6). Príkazy k vytváraniu týchto objektov sú v podkapitole 8.6.

Rúry zabezpečujú obojsmerný prenos dát medzi procesmi, ktoré môžu bežať na rôznych počítačoch s operačným systémom Windows. Prenos dát je typu „od jedného k jednému“. Rúry sa dajú rozdeliť na pomenované rúry a nepomenované rúry. Pomenované rúry sa dajú použiť v sieti medzi operačnými systémami Windows, ale nepomenované rúry sa dajú použiť iba v rámci jedného počítača.

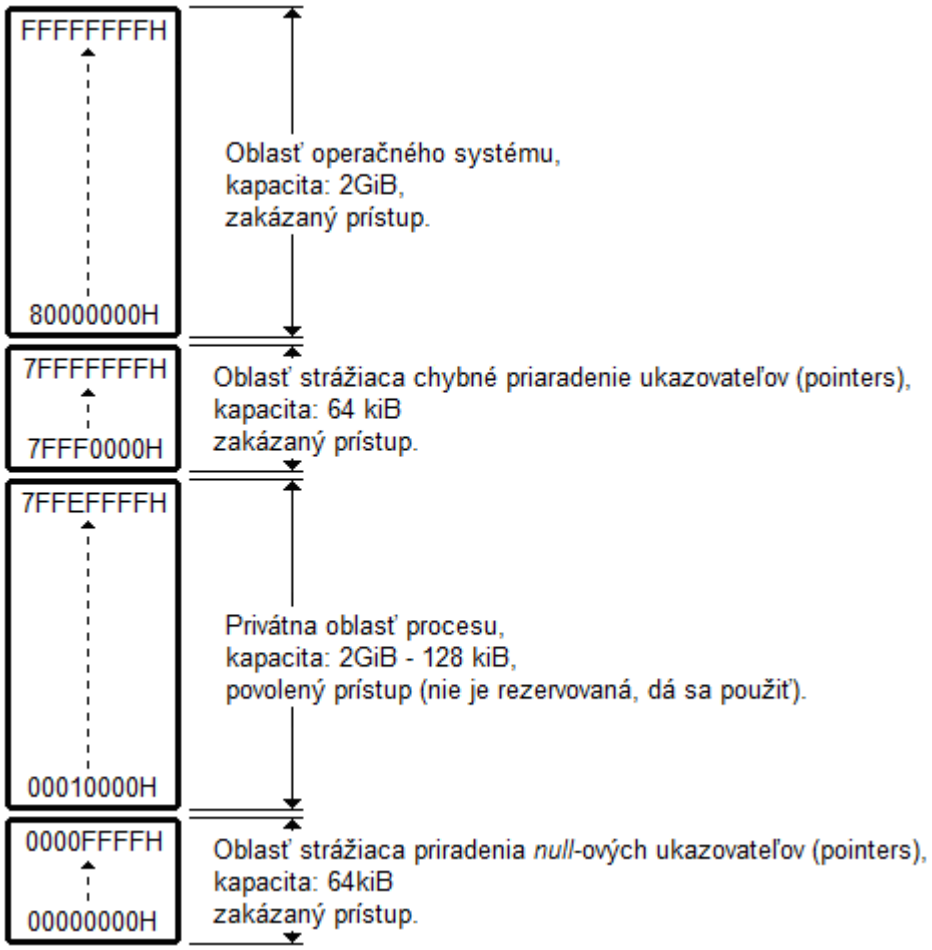
Poštové priehradky zabezpečujú prenos dát medzi procesmi jedným smerom a taktiež tieto procesy môžu byť na rôznych počítačoch s operačným systémom Windows. Prenos dát je typu „od jedného k viacerým“.

8.4 Pamäťová architektúra operačných systémov Windows NT

V prostredí rozhrania Win32 má každý proces pridelený vlastný virtuálny adresný priestor o veľkosti 4 GiB, 32-bitový ukazovateľ (pointer) môže nadobúdať akúkoľvek hodnotu z intervalu 00000000H až FFFFFFFFH. Ukazovateľ tak môže nadobúdať 4 294 967 296 rôznych hodnôt, čo presne odpovedá oblasti o veľkosti 4GiB.

Vykonávací tok akéhokoľvek procesu môže pristupovať výhradne k pamäti, ktorá procesu patrí. Pamäťový priestor ostatných procesov je pred vykonávacím tokom skrytý a nedá sa k nemu pristupovať.

Na obrázku Obr. 8.9 je znázornené rozdelenie adresného priestoru procesu vo Windows-och NT (Windows NT 3.1 až Windows 10):



Obr. 8.9 Virtuálny adresný priestor procesu vo Windows-och NT

Oblasť pamäte od 0000 0000 H do 0000 FFFF H:

Túto 64kiB oblasť v dolnej časti pamäte rezervuje Windows NT pre potreby zachytenia chýb priradenia *null*-ových ukazovateľov. Akýkoľvek pokus o čítanie alebo zápis v tejto oblasti spôsobí chybu prístupu do pamäte.

Oblasť pamäte od 0001 0000 H do 7FFE FFFF H:

Táto oblasť o veľkosti 2GiB bez 128 kiB je privátnou nezdieľanou pamäťovou oblasťou procesu. Pri spustení vyžaduje proces prístup k systémovým dynamickým knižniciam *KERNEL32.DLL*, *USER32.DLL*, *GDI32.DLL* a *ADVAPI32.DLL*. Kód týchto dynamických knižníc, rovnako ako všetkých ostatných dynamických knižníc používaných procesom, sa tiež s procesom nahrávajú do tejto oblasti pamäte. Každý proces môže tieto dynamické knižnice nahrať na inú adresu v tejto oblasti (je to však veľmi nepravdepodobné, keďže sa prilinkujú ako prvé). Do tejto oblasti systém taktiež mapuje pamäťovo mapované súbory, ku ktorým ma proces prístup.

Oblasť pamäte od 7FFF 0000 H do 7FFF FFFF H:

Táto oblasť o veľkosti 64 kiB tesne pod hranicou 2GiB sa chová podobne, ako oblasť na adresách 0000 0000 H až 0000 FFFF H. Je to oblasť so zakázaným prístupom a akýkoľvek pokus o prístup do tejto oblasti spôsobí chybu prístupu do pamäte. Rezervovaním tejto oblasti pamäte si Microsoft iba uľahčil implementáciu operačného systému. Ak nejakej API funkcií sa predá adresa bloku dát pamäte a jeho dĺžka, pred prevedením operácie funkcie overuje platnosť zadaného bloku pamäte. Dá sa to ľahko predstaviť na nasledujúcej ukážke kódu:

```
BYTE bBuf[7000];
DWORD dwNumBytesWritten;
BOOL WriteProcessMemory(hProcess, 0x7FFEEE90, bBuf,
     sizeof(bBuf), &dwNumBytesWritten);
```

Pri funkciách ako je práve `WriteProcessMemory()` sa oblasť pamäte do ktorej sa má zapisovať najprv overuje kódom v režime jadra, ktorý môže jednoducho pristupovať k oblastiam nad adresou 8000 0000 H. Pokiaľ by sa malo zapisovať do pamäte nad adresou 8000 0000 H, volanie funkcie by úspešne zapísalo dáta do oblasti, ktorá má byť prístupná iba kódu jadra. Aby sa tomu zabránilo a zároveň bola overená platnosť dostatočne rýchlo, Microsoft sa rozhodol zakázať prístup k oblasti 7FFF 0000 H až 7FFF FFFF H, takže akýkoľvek pokus o prístup do tejto oblasti spôsobí chybu prístupu pamäte.

Oblasť pamäte od 8000 0000 H do FFFF FFFF H:

V tejto oblasti o veľkosti 2GiB zdieľa výkonný kód Windows NT, kód jadra a ovládače zariadení. Pokiaľ sa aplikácia pokúsi o prístup do tejto oblasti, dôjde k chybe prístupu, systém zobrazí dialógové okno a Windows NT aplikáciu ukončí.

8.4.1 Pridelenie fyzického priestoru oblastiam

Procesu alokovania oblasti pamäte sa hovorí *rezervovanie* oblasti. Keď sa rezervuje oblasť pamäte, systém zaistí, že rezervovaná oblasť bude vždy začínať na celých násobkoch *alokačného kroku*. Alokačný krok systém používa, aby sa mu ľahšie spravovali interné záznamy o rezervovaných regiónoch adresného priestoru a taktiež ku zníženiu miery fragmentácie segmentov adresného priestoru, ku ktorej môže v adresnom priestore dochádzať.

Pri rezervovaní oblasti adresného priestoru systém zaistí, že veľkosť rezervovaného priestoru bude vždy celočíselným násobkom veľkosti systémovej *stránky*. Stránka je pamäťová jednotka konštantnej veľkosti, ktorú systém používa pri správe pamäte. Samozrejme keď je nutné rezervovať oblasť pamäte o nejakej veľkosti, stále sa musí použiť cela stránka, hoc by sa v nej zapísal iba jeden bajt (podobne ako pri klástroch).

Aby mohla byť rezervovaná oblasť pamäťového priestoru použitá, musí sa alokovať odpovedajúca časť fyzického pamäťového priestoru a táto časť sa musí mapovať na rezervovanú oblasť. Tomuto postupu sa hovorí *pridelenie* (commiting) fyzického priestoru. Fyzický pamäťový priestor sa vždy prideľuje po stránkach. Pridelenie fyzického priestoru rezervovanej oblasti sa prevádza volaním funkcie:

```
VirtualAlloc().
```

Keď sa prideľuje rezervovanej oblasti fyzický pamäťový priestor, nemusí sa nutne pokryť celá rezervovaná oblasť. Fyzický pamäťový priestor sa prideľuje po stránkach.

Ako náhle program ďalej nepotrebuje pristupovať k fyzickému pamäťovému priestoru rezervovanej oblasti, mal by pridelený fyzický priestor *uvoľniť* (decommiting). Uvoľnenie fyzického pamäťového priestoru sa prevádza pomocou funkcie:

```
VirtualFree().
```

8.4.2 Fyzický pamäťový priestor

V 16-bitových Windows-och 3.x sa za fyzický pamäťový priestor považoval objem pamäte RAM. Správa pamäte vo Windows-och 4.x (95, 98, ME) a Windows-och NT (Windows NT až Windows 10) je implementovaná úplne odlišne od mechanizmu používaných vo Windows-och 3.x. V týchto sytémoch (Windows NT a Windows 4.x) je pamäť RAM spravovaná výhradne operačným systémom a s jej obsahom nemôže priamo manipulovať žiadna aplikácia.

Pri práci s rozhraním Win32 je najlepšie si fyzicky pamäťový priestor predstavovať ako dáta, uložené v stránkovacom súbore na diskovom zariadení (obvykle na HDD). Keď teda aplikácia volá funkciu `VirtualAlloc()` a prideľuje fyzický pamäťový priestor nejakej oblasti adresného priestoru, fyzicky priestor sa v skutočnosti alokuje zo súboru na pevnom disku. Najdôležitejší údaj, ktorý ovplyvňuje veľkosť fyzického pamäťového priestoru dostupného aplikáciám je práve veľkosť tohto stránkovacieho súboru na HDD. Stránkovací súbor je v podstate virtuálna pamäť spomínaná v podkapitole 8.1.

Pokiaľ chce vykonávací tok procesu pristupovať k oblasti v adresnom priestore procesu, môže sa stať jedna z dvoch vecí:

1. Prvá možnosť je tá, že dáta, ku ktorým chce vykonávací tok pristupovať, sa nachádzajú priamo v pamäti RAM. V takomto prípade uskutoční procesor mapovanie virtuálnej adresy dát na fyzickú adresu v RAM. Po tomto je možné realizovať požadovaný prístup.
2. Druhá možnosť nastáva vtedy, pokiaľ dáta, ku ktorým tok pristupuje, nie sú síce v pamäti RAM, ale nachádzajú sa niekde v stránkovacom súbore. V takomto prípade dochádza k takzvanému výpadku stránky (*page fault*) a procesor oznámi operačnému systému požiadavku na dáta. Operačný systém nájde v pamäti RAM voľnú stránku, ak nie je žiadna stránka voľná, operačný systém musí nejakú uvoľniť. Pokiaľ však systém uvoľňuje stránku, ktorá bola modifikovaná musí najskôr obsah stránky prekopírovať z pamäte RAM do stránkovacieho súboru. Potom systém vezme mapovaný súbor, nájde v ňom požadovaný blok dát procesorom a tieto dáta nahrá do voľne stránky v pamäti RAM. Nakoniec systém prevedie mapovanie virtuálnej adresy dát na odpovedajúcu fyzickú adresu v pamäti RAM.

Čím častejšie potrebuje systém kopírovať stránky zo stránkovacieho súboru do pamäte RAM a späť, tím sa viac pracuje s HDD a systém pracuje pomalšie (HDD má rádovo pomalší prístup ako RAM). Najhorším prípadom tak môže byť, že operačný systém celý svoj čas trávi *swapovaním* stránok do a z pamäte a aplikácia beží veľmi pomaly, alebo vôbec. Dodaním ďalšej pamäte RAM do počítača sa obmedzia diskové manipulácie nutné pre beh aplikácií, čo pochopiteľne výrazne zvýši výkon systému.

8.4.3 Práca s virtuálnou pamäťou

Informácie o správe pamäte a o virtuálnom adresnom priestore procesov sa dá zistiť požiadaním o systémové informácie, vrátenie stavu virtuálnej pamäte a vrátenie stavu adresného priestoru.

Systémové informácie:

Systémové informácie sa zisťujú pomocou funkcie `GetSystemInfo()`, ktorá získava informácie o implementácií rozhrania Win32 API vrátane informácií o virtuálnej pamäti. Funkcia sa volá takto:

```
VOID GetSystemInfo(LPSYSTEM_INFO lpSystemInfo);
```

Tejto funkcii sa predáva adresa štruktúry *SYSTEM_INFO*. Funkcia nastaví jednotlivé položky tejto štruktúry a vráti sa.

Stav virtuálne pamäte:

Stav virtuálne pamäte sa zisťuje pomocou funkcie `GlobalMemoryStatus()`, ktorá poskytuje dynamické informácie o súčasnom stave pamäte, funkcia sa volá takto:

```
VOID GlobalMemoryStatus(LPMEMORYSTATUS lpmstMemeoryStatus);
```

Keď sa volá spomínaná funkcia, tak sa jej predáva adresa štruktúry *MEMORYSTATUS*.

Zistenie stavu adresného priestoru:

Rozhranie Win32 obsahuje funkciu, ktorá dovoľuje získať niektoré informácie o adresách v adresnom priestore procesu (napríklad veľkosť, fyzický typ uloženia a príznaky ochrany). Zistenie stavu adresného priestoru voláme funkciou `VirtualQuery()`:

```
DWORD VirtualQuery (LPVOID plAddress,
PMEMORY_BASIC_INFORMATION lpBuffer, DWORD dwLength);
```

Pri volaní funkcie `VirtualQuery()` musí prvý parameter `lpAddress` obsahovať adresu virtuálneho pamäťového priestoru, o ktorom sa zisťujú informácie. Parameter `lpBuffer` obsahuje adresu štruktúry MEMORY_BASIC_INFORMATION, ktorú je potrebné pred tým alokovať.

Kódy jednotlivých spomenutých štruktúr sa nájdu v manuáloch Win32 API pre C++ a sú aj na servery MSDN (http://msdn.microsoft.com/).

8.5 Pamäťovo mapované súbory

Pracovať so súbormi potrebuje takmer každá aplikácia (otváranie, čítanie, uloženie, zatvorenie súboru). V rozhraní Win32 je implementovaný spôsob *pamäťovo mapovaných súborov*.

Rovnako ako pri virtuálnej pamäti, pomocou pamäťovo mapovaných súborov sa môže rezervovať časť adresného priestoru a tejto časti prideliť fyzický pamäťový priestor. Rozdiel je iba v tom, že fyzický pamäťový priestor nie je reprezentovaný systémový stránkovacím súborom, ale použije sa súbor, ktorý už na disku existuje a pridelí sa mu adresa vo virtuálnej pamäti. Ako náhle bude súbor namapovaný, je možné k nemu pristupovať rovnako, ako keby bol celý nahraný v pamäti.

Pamäťovo mapované súbory sa využívajú k trom rôznym účelom:

- pamäťovo mapované súbory používa systém k nahraniu a spusteniu *EXE* a *DLL* súborov, tým sa jednak výrazne šetrí miesto v systémovom stránkovacom súbore a jednak sa skracuje čas, potrebný pre spustenie aplikácie,
- pamäťovo mapované súbory je možné využiť pre prístup k dátam na disku, tým sa zabaví nutnosti prevádzať všelijaké vstupno-výstupné operácie a pracovať s rôznymi buffer-ami pre uchovanie časti súboru,
- pamäťovo mapované súbory sa dajú tiež využiť k zdieľaniu dát medzi rôznymi procesmi na jednom počítači (rozhranie Win32 obsahuje aj iné metódy pre vzájomnú komunikáciu procesov, všetky tieto metódy sú však implementované pomocou pamäťovo mapovaných súborov).

V tejto podkapitole sa píše o všetkých troch spôsoboch použitia pamäťovo mapovaných súborov.

8.5.1 Pamäťovo mapované EXE a DLL súbory

Keď vykonávací tok volá funkciu `CreateProcess()`, systém spraví nasledujúce kroky:

1. Funkcia nájde *EXE* súbor, zadaný pri jej volaní. Pokiaľ sa tento súbor nedá nájsť, proces nebude vytvorený a funkcia `CreateProcess()` vráti hodnotu *false*.
2. Systém vytvorí nový objekt jadra typu proces.
3. Pre tento nový proces vytvorí systém virtuálny adresný priestor o veľkosti 4 GiB.
4. V adresnom priestore systém rezervuje dostatočne veľkú oblasť pre uloženie *EXE* súboru. Požadované umiestnenie oblasti je zadané priamo v *EXE* súbore. Implicitne sa *EXE* súbory ukladajú na adresu 0040 0000 H. Implicitné nastavenie je však možno zmeniť pomocou parametra */BASE* pri linkovaní aplikácie.
5. Systém si poznačí, že fyzicky adresný priestor pridelený oblasti je zrkadlom v *EXE* súbore, v žiadnom prípade v systémovom stránkovacom súbore.

Ak dôjde k namapovaniu *EXE* súboru do adresného priestoru procesu, systém v tomto súbore nájde úsek, ktorý obsahuje zoznam všetkých *DLL* knižníc, ktoré kód *EXE*

súboru volá. Potom systém nahrá jednotlivé knižnice volaním funkcie `LoadLibrary()`, pokiaľ niektorá z knižníc používa ďalšie dynamické knižnice, systém opäť funkciou `LoadLibrary()` nahrá tieto ďalšie knižnice. Pri každom volaní funkcie `LoadLibrary()` spraví systém kroky, zodpovedajúce vyššie popísaným krokom 4 a 5:

1. V adresnom priestore procesu systém rezervuje pre dynamickú knižnicu dostatočne veľkú oblasť. Požadovaná adresa oblasti ja zadaná priamo v dynamickej knižnici. Implicitne nastavuje prekladač Visual Studio C++ bázovú adresu dynamických knižníc na hodnotu 1000 0000 H. Toto nastavenie je možno prepísať parametrom /BUILD pri linkovaní aplikácie. Všetky štandardné systémové knižnice dodávané s Windows-om NT majú nastavené rôzne bázové adresy.
2. Pokiaľ sa systému nepodarí rezervovať oblasť podľa požiadaviek dynamickej knižnice (buď je v tejto oblasti *EXE* súbor, alebo iná dynamická knižnica, alebo nie je voľná oblasť dostatočne veľká) systém hľadá akúkoľvek inú oblasť v adresnom priestore, ktorú by mohol pre dynamickú knižnicu rezervovať. Z dvoch dôvodov však nie je dobré, keď sa dynamickú knižnicu nepodarí nahrať na ňou požadovanú adresu. Za prvé sa systému nepodarí knižnicu nahrať inde, pokiaľ neobsahuje relokačné informácie (relokačné informácie je možno z knižnice odstrániť pomocou parametra */FIXED* pri linkovaní aplikácie, výsledný *DLL* súbor bude menší, ale je nutné, aby bol vždy nahraný na požadovanú bázovú adresu). Za druhé, pri nahrávaní knižnice na inú adresu musí systém vykonať relokáciu knižnice. Kvôli týmto relokáciám je nutné zabrať dodatočný priestor zo systémového stránkovacieho súboru, rovnako sa tým aj predlžuje doba, potrebná pre nahranie dynamickej knižnice.
3. Systém si poznačí, že fyzický pamäťový priestor, ktorý zrkadľuje rezervovanú oblasť, sa nachádza v *DLL* súbore a nie v systémovom stránkovacom súbore. Pokiaľ operačný systém musel spraviť relokáciu knižnice, pretože ju nebolo možné uložiť na požadovanú bázovú adresu, systém si ďalej poznačí, že niektoré časti knižnice sú uložené v stránkovacom súbore.

Pokiaľ sa systému z akýchkoľvek dôvodov nepodarí namapovať *EXE* súbor a všetky ním požadované dynamické knižnice, zobrazí sa dialógové okno s touto správou a adresný priestor procesu aj samotný objekt procesu budú uvoľnený. Funkcia `CreateProcess()` vráti volajúcemu procesu hodnotu *false*. Volajúci proces tak môže použiť funkciu `GetLastError()` a pokúsiť sa presne zistiť, prečo vlastne nebol proces vytvorený.

Keď systém namapuje *EXE* súbor aj všetky *DLL* súbory do adresného priestoru procesu, môže systém začať vykonávať kód *EXE* súboru. Ako náhle dôjde k namapovaniu *EXE* súboru, všetko stránkovanie, buffer-ovanie a cache-ovanie obstaráva systém vlastnými silami. Pokiaľ napríklad kód *EXE* súboru spraví skok na inštrukciu na adrese, ktorá nie je nahraná v pamäti, dôjde k výpadku stránky. Systém výpadok zaregistruje, a automaticky do pamäte požadovanú stránku z *EXE* súboru nahrá. Potom systém nahratú stránku namapuje na správnu adresu v adresovanom priestore procesu a vykonávací tok môže pokračovať v činnosti, ako keby bola cieľová inštrukcia skoku v pamäti prítomná hneď od začiatku. Všetko toto dianie je pre aplikáciu samozrejme neviditeľné. Postup sa opakuje zakaždým, keď sa ktorýkoľvek vykonávací tok v procese pokúsi pristupovať ku kódu alebo dátam, ktoré nie sú v pamäti momentálne nahraté.

8.5.2 Pamäťovo mapované dátové súbory

Pri nahrávaní EXE alebo DLL súborov používa systém vyššie popísaný postup automaticky. Do adresného priestoru procesu je však možné mapovať aj ľubovoľný dátový súbor. Vďaka tomu sa dajú pohodlne spracovávať aj veľké objemy dát. Pri spracovaní dát sa môžu použiť napríklad takéto metódy (samozrejme je to na programátorovi):

- jeden súbor, jeden buffer:
  - táto metóda naalokuje blok pamäte dostatočne veľký nato, aby mohol obsahovať celý dátový súbor; súbor sa otvorí, jeho obsah bude načítaný do pamäte a súbor sa zase zatvorí; dáta sa spracujú a znovu nahrajú do súboru (nevýhoda: maximálna veľkosť súboru so strojovým kódom jeho spracovania musí mať menej ako 2GiB),
- dva súbory, jeden buffer:
  - otvorí sa súbor, skopíruje sa z neho do pamäte jeho určitá časť, ktorá sa spracuje a vloží do nového súboru, toto sa opakuje kým sa nespracuje celý prvý súbor a v druhom súbore budú spracovane dáta,
- jeden súbor, dva buffer-e:
  - v prípade keď pri spracovaní jednej informácie v súbore je nutné pracovať s rôznymi dátami súboru, tak sa tieto dáta načítajú do dvoch buffer-ov (ak to postačuje) a načítané informácie sa zo súboru vymažú a nahradia novými (novou),
- jeden súbor, žiaden buffer:
  - pomocou pamäťového mapovania sa otvorí súbor a požiada sa systém o rezervovanie oblasti vo virtuálnom adresnom priestore, prvý bajt súboru sa namapuje na prvý bajt rezervovanej oblasti; potom pomocou prístupu k virtuálnej pamäti procesu sa spracuje cely súbor (súbor sa bude spracovávať ako dáta v pamäti procesu).

Aby sa dal pamäťovo mapovaný súbor použiť musia sa splniť tri kroky:

1. Vytvoriť alebo otvoriť objekt jadra typu súbor, ktorý bude identifikovať diskový súbor, ktorý sa bude mapovať do pamäte.
2. Vytvoriť objekt jadra typu mapovanie súboru, ktorým sa systému oznamuje veľkosť súboru a spôsob, akým sa k nemu bude pristupovať.
3. Oznámiť systému, aby celý obsah alebo len časť súboru mapoval do adresného priestoru procesu.

Po skončení práce s pamäťovo mapovaným súborom je nutné vykonať uvoľnenie zabraných prostriedkov:

1. Oznámiť systému, aby zrušil mapovanie súboru do adresného priestoru procesu.
2. Zavrieť objekt jadra typu mapovanie súboru.
3. Zavrieť objekt jadra typu súbor.

Vytvorenie alebo otvorenie objektu typu súbor:

```
HANDLE CreateFile(LPCSTR lpFileName, DWORD dwDesiredAccess,
      DWORD dwShareMode, LPSECURITY_ATTRIBUTES lpsa,
   DWORD dwCreationOptions, DWORD dwFlagsAndAttributes,
                  HANDLE hTemplateFile);
```

Parameter `lpFileName` je názov súboru prípadne aj s absolútnou, alebo relatívnou cestou. Parameter `dwDesiredAcces` špecifikuje prístup ku súboru a môže nadobúdať hodnoty: „0“, „GENERIC_READ“, „GENERIC_WRITE“, „GENERIC_READ|GENERIC_WRITE“. Ďalší parameter dwShareMode oznamuje systému ako sa bude súbor zdieľať (možné hodnoty: „0“, „FILE_SHARE_READ“, „FILE_SHARE_WRITE“, „FILE_SHARE_READ | FILE_SHARE_WRITE“).

Vytvorenie objektu mapovania súboru:

```
HANDLE CreateFileMapping(HANDLE hFile,
LPSECURITY_ATTRIBUTES lpsa, DWORD fdwProtect,
          DWORD dwMaximumSizeHigh,
DWORD dwMaximumSizeLow, LPSTR lpName);
```

Parameter `hFile` je hodnota `HANDLE` vytvoreného objektu súboru a parameter `lpName` je meno vytvoreného objektu. Parameter `fdwProtect` dovoľuje zadať príznaky ochrany („PAGE_READONLY“, „PAGE_READWRITE“, „PAGE_WRITECOPY“). Ďalšími dvoma parametrami sa nastavuje maximálna veľkosť súboru, aby systém vedel koľko pamäte potrebuje na zaadresovanie, nakoľko súbor môže mať viac ako 4 GiB (maximálne však 18 EiB), tak musia byť dva parametre (vrchných 32 bitov a spodných 32 bitov veľkosti súboru). Posledným parametrom je pomenovanie objektu.

Mapovanie dát súboru do adresného priestoru procesu:

```
LPVOID MapViewOfFile(HANDLE hFileMappingObject,
 DWORD dwDesiredAccess, DWORD dwFileOffsetHigh,
DWORD dwFileOffsetLow, DWORD dwNumberOfBytesToMap);
```

Prvý parameter `hFileMappingObject` je `HANDLE` objektu mapovania súboru (`HANDLE CreateFileMapping()`), ďalší parameter určuje prístup k dátam („FILE_MAP_READ“, „FILE_MAP_WRITE“, „FILE_MAP_ALL_ACCES“). Nastavenie prvého bajtu súboru od jeho počiatku, ktorý sa má premapovať do adresného priestoru, sa nastavuje pomocou ďalších dvoch parametrov (znova horných a dolných 32 bitov, nakoľko súbor môže mať cez 4GiB). Posledným parametrom `dwNumberOfBytesToMap` sa nastavuje koľko bajtov sa má z dátového súboru premapovať (na to stačí jeden parameter, keďže adresný priestor má len 4GiB).

Zrušenie mapovania dát v adresnom priestore, objektu mapovania súboru a objektu súboru:

```
BOOL UnmapViewOfFile(LPVOID lpBaseAddress);
        CloseHandle(hFileMapping);
            CloseHandle(hFile);
```

8.5.3 Pamäťovo mapované súbory v implementáciách rozhrania Win32 API

Táto časť podkapitoly vysvetlí ako môžu viaceré procesy zdieľať dáta s pamäťovo mapovaným dátovým súborom. Keď bude mať proces už zaadresované dáta súboru, musí zavolať funkciu `MapViewOfFile()`. Ak funkciu `MapViewOfFile()` zavolá jeden proces, tak operačný systém rezervuje oblasť pamäte pre pohľad do súboru iba v adresnom priestore tohto procesu (žiadny iný proces nemôže pohľad vidieť). Pokiaľ by ďalší proces chcel vidieť dáta z rovnakého mapovaného objektu, potom by musel funkciu `MapViewOfFile()` zavolať aj vykonávací tok tohto druhého procesu. Systém by vykonal rezerváciu oblasti pre pohľad do súboru aj v adresnom priestore druhého procesu.

Treba si zapamätať veľmi dôležitú vec: adresa, ktorú získa prvý proces volaním funkcie `MapViewOfFile()`, sa bude s veľmi veľkou pravdepodobnosťou líšiť od adresy, ktorú získa volaním rovnakej funkcie druhý proces. Bude tomu tak aj keď oba procesy mapujú pohľad do rovnakého objektu mapovania súboru (virtuálna adresa bude iná).

Ďalším spôsobom ako zdieľať mapovanie súboru medzi procesmi je ten, že so súborom sa môže pracovať ako s inými už spomínanými objektmi a to volaním funkcie `CreateFileMapping()`:

```
HANDLE CreateFileMapping(HANDLE hFile,
LPSECURITY_ATTRIBUTES lpsa, DWORD fdwProtect,
        DWORD dwMaximumSizeHigh,
DWORD dwMaximumSizeLow, LPSTR lpName);
```

Dôležitými parametrami sú `hFile` a `lpName`. Parameter `hFile` je hodnota `HANDLE` objektu súbor a parameter `lpName` je meno vytvoreného objektu.

Pokiaľ chce pristupovať k objektu ten istý proces stačí mu poznať jeho hodnotu `HANDLE`, ale ak k objektu mapovaného súboru chce pristupovať iný proces musí to spraviť pomocou funkcie `CreateFileMapping()` s rovnakým parametrom `lpName`, pomocou dedenia, pomocou funkcie `DuplicateHandle()`, alebo pomocou funkcie `OpenFileMapping()`, ktorej parameter `lpName` sa bude zhodovať s menom, zadaným pri volaní funkcie `CreateFielMapping()` v inom toku:

```
HANDLE OpenFileMapping(DWORD dwDesiredAccess, BOOL
        fInheritHandle, LPTSTR lpName);
```

Tak ako aj pred tým, aj mapovaný súbor sa zatvára pomocou funkcie `CloseHandle()`.

8.6 Vstupno-výstupné operácie

Jednou z najsilnejších stránok Windows-och NT je veľký počet podporovaných zariadení. V tejto súvislosti definujeme zariadenia ako čokoľvek, čo umožňuje komunikáciu.

V nasledujúcej tabuľke (Tab. 8.1) sú uvedené niektoré zariadenia a ich najobvyklejšie použitie:

Tab. 8.1 Výber vstupno-výstupných zariadení a ich použitie

Zariadenie	**Úkon**
Súbor	Trvalé uloženie ľubovoľných dát.
Adresár	Atribúty a komprimácia súborov.
Logický disk	Formátovanie.
Fyzický disk	Prístup k tabuľke partícií.
Sériový port	Prenos dát.
Paralelný port	Prenos dát (tlačiareň, skener).
Poštová priehradka	Prenos dát typu „od jedného k viacerým“, obvykle po sieti medzi počítačmi s operačným systémom Windows.
Pomenovaná rúra	Prenos dát typu „od jedného k jednému“, obvykle po sieti medzi počítačmi s operačným systémom Windows.
Nepomenovaná rúra	Prenos dát typu „od jedného k jednému“, na jednom počítači (nikdy nie po sieti).
Socket	Datagramový alebo prúdový prenos dát, obvykle po sieti medzi ľubovoľnými počítačmi (na počítači nemusí bežať operačný systém Windows).
Konzola	Obrazovkový buffer textového okna.

Aby sa mohla vykonať akákoľvek vstupno-výstupná operácia, musí sa najprv požadované zariadenie otvoriť a tým získať jeho `HANDLE`. Nasledujúca tabuľka (Tab. 8.2) obsahuje súpis rôznych zariadení spolu s funkciami, ktorými sa každé zariadenie otvára.

Tab. 8.2 Otváranie vstupno-výstupných zariadení

Zariadenie	**Funkcia pre otvorenie zariadenia**
Súbor	`CreateFile(pszName);`
Adresár	`CreateFile(pszName);` s príznakom *FILE_FLAG_BACKUP_SEMANTICS*
Logický disk	`CreateFile(pszName);` `pszName` je „\\.*x*:“
Fyzický disk	`CreateFile(pszName);` `pszName` je „\\.\PHYSICALDRIVE*x*“
Sériový port	`CreateFile(pszName);` `pszName` je „COM*x*“
Paralelný port	`CreateFile(pszName);` `pszName` je „LPT*x*“
Server poštovej priehradky	`CreateMailslot(pszName);` `pszName` je „\\.\mail-slot*meno priehradky*“
Klient poštovej priehradky	`CreateFile(pszName);` `pszName` je „*meno servera*\mail-slot*meno priehradky*“
Server pomenovanej rúry	`CreateNamedPipe(pszName);` `pszName` je „\\.\pipe*meno rúry*“
Klient pomenovanej rúry	`CreateFile(pszName);` `pszName` je „*meno servera*\pipe*meno rúry*“
Klient a server nepomenovanej rúry	`CreatePipe();`
Socket	`Socket(); accept(); AcceptEx();`
Konzola	`CreateConsoleScreenBuffer(); GetStdHandle();`

Každá z funkcií v tabuľke Tab. 8.2 vracia 32-bitový `HANDLE`, ktorý identifikuje otvorené zariadenie. Pri komunikácií so zariadením sa potom tento `HANDLE` predáva rôznym funkciám.

Rozhranie Win32 sa pred programátorom snaží čo najviac skryť rozdiely medzi rôznymi zariadeniami. Znamená to, že ako náhle sa raz otvorí zariadenie, k čítaniu a zápisu dát potom sa už používajú vždy rovnaké funkcie rozhrania Win32 bez ohľadu nato, s akým zariadením sa práve komunikuje. Hoci pre čítanie a zápis dát bez ohľadu na konkrétne použité zariadenie existuje v rozhraní Win32 len niekoľko málo funkcií, jednotlivé zariadenia sa od seba líšia. Napríklad pri sériovej linke má význam nastavovanie prenosovej rýchlosti, toto nastavenie však nemá žiaden zmysel pri komunikácií po sieti pomocou pomenovaných rúr.

Zoznam použitých skratiek

A – Ampér, základná jednotka prúdu v sústave SI, vedľajšia jednotka: mA
AC – Alternating Current (striedavý prúd)
AGP – počítačová zbernica: Accelerated Graphics Port (zrýchlený grafický port)
AHCI – Advanced Host Controller Interface (pokročilý hostiteľsky radič rozhrania)
ALU – časť procesora: Arithmetic-Logic Unit (aritmeticko-logická jednotka)
AMD – spoločnosť Advanced Micro Devices
AMOLED – zobrazovacia technológia: Active-Matrix Organic Light-Emitting Diode (organická luminiscenčná dióda s aktívnou maticou)
ANSI – súbor štandardov: American National Standards Institute (Národný inštitút amerických štandardov)
API – Application Programming Interface (Programátorské rozhranie aplikácií)
APIC – radič prerušení: Advanced Programmable Interrupt Controller (pokročilý programovateľný radič prerušení)
ARCNET – počítačová sieť: Attached Resource Computer NETwork (sieť prepojených počítačových prostriedkov)
ARP – protokol: Address Resolution Protocol (protokol rozpoznávania adries)
ASA – spoločnosť: Advertising Standards Autority
ASCII – American Standard Code for Information Interchange (Americký štandardný kód pre výmenu informácií)
ATA – počítačové rozhranie: Advanced Technology Attachment (zdokonalená prepojovacia technológia)
ATDMA – Advanced Time Division Multiple Access (asynchrónny časový multiplex)
ATX – formát matičných dosiek Advanced Technology eXtended (zdokonalená predĺžená technológia)
AUI – rozhranie: Attachment Unit Interface
AUX – rozhranie určené pre zvuk: AUdio syntaX
A/D – Analog to Digital (analógová hodnota do digitálnej hodnoty)
b – bit, základná jednotka informácie, vedľajšie jednotky: B, kb, kB, kiB, Mb, MB, MiB, Gb, GB, GiB, TB, TiB, EB, EiB
b/s – bit za sekundu, informačná prenosová rýchlosť, vedľajšie jednotky: B/s, kb/s, kB/s, kiB/s, Mb/s, MB/s, MiB/s, Gb/s, GB/s, GiB/s.
BCD – kódovanie: Binary Code Decimal (binárne kódovanie desiatkovej sústavy)
BD – optický disk: Blue Ray Disc
BIOS – Basic Input Output System (základný systém pre vstupy a výstupy)
BPDU – protokol: Bridge Protocol Data Unit (protokol dátovej jednotky mostu)
BRD – optický disk: Blue-Ray Disk
BSD – Berkeley Software Distribution (distribúcia softvéru univerzity v Berkeley)
CAM – pamäť: Content Access Memory (pamäť adresovaná obsahom)
CAN – sieťové rozhranie: Controller Area Network
CCD – Charge-Coupled Device (obvod s nábojovou väzbou)
CD – optický disk: Compact Disc
CD-R – optický disk: Compact Disc – Recordable (s možnosťou zápisu)
CD-ROM – optický disk: Compact Disc Read-Only Memory (iba na čítanie)
CDP – protokol: Cisco Discovery Protocol
CF – flash pamäť: Compact Flash
CGA – grafická karta: Color Graphics Adapter (farebný grafický adaptér)
CISC – inštrukčná sada procesora: Complex Instruction Set Computer (komplexná inštrukčná sada počítača)

CLK – Clock (hodiny)
CMOS – Complementary Metal Oxide Semiconductor (Komplementárny polovodič kov-kyslík)
CMY – kódovanie farieb: Cyan, Magenta, Yellow (azúrová, purpurová, žltá)
CPU – procesor: Central Processing Unit (centrálna procesná jednotka)
CRT – Cathode Ray Tube (katódová trubica)
CSMA – Carrier Sense Multiple Access (nosné metódami náhodného prístupu)
ČSSR – Česko-Slovenská Socialistická Republika
D – dutinka
D/A – Digital to Analog (digitálna hodnota do analógovej hodnoty)
dB – jednotka intenzity zvuku
DACK – Direct memory access Acknowledge (potvrdenie priameho prístupu do pamäte)
DC – Direct Current (jednosmerný prúd)
DDR – technológia RAM: Double Data Rate (dvojitý dátový zápis počas jedného hodinového signálu)
DEC – korporácia (spoločnosť): Digital Equipment Corporation
DECLAT – protokol: DEC Local Area Transport (protokol korporácie DEC pre transport po lokálnej sieti)
DIP – prepínač: Dual In-line Package
DIX – DEC/Intel/Xerox (norma zavedená spoločnosťami DEC, Intel a Xerox)
DLL – Dynamic Link Library (dynamický linkovaná knižnica)
DMA – Direct Memory Access (priamy prístup do pamäte)
DMAC – Direct Memory Access Controller (radič priameho prístupu do pamäte)
DMI – počítačová zbernica: Direct media interface (rozhranie priamo pre média)
DOS – operačný systém: Disk Operating System (diskový operačný systém)
DPCP – DisplayPort Content Protection (DisplayPort s ochranou)
DPI – údaj určujúci koľko obrazových bodov (pixelov) sa vojde do dĺžky jedného palca (inch): Dots Per Inch
DRAM – technológia RAM: Dynamic Random Access Memory (dynamická RAM)
DRQ – Direct Memory Access Request (žiadosť o priamy prístup do pamäte)
DSL – Digital Subscriber Line (digitálne účastnícke vedenie)
DTP – Desktop publishing (publikovanie celej pracovnej plochy)
DVD – optický disk: Digital Versatile Disc, Digital Video Disc
DVI – grafické rozhranie: Digital Visual Interface (digitálne a vizuálne rozhranie)
EBC – radič: EISA bus controller (radič zbernice EISA)
EBCDIC – kódovanie: Extended BCD Interchange Code (rozšírený BCD kód)
EEPROM – Electrically Erasable Programmable ROM (Elektrický mazateľná pamäť ROM)
EGA – grafická karta: Enhanced Graphic Adapter (Rozšírený grafický adaptér)
EGP – protokol: Exterior Gateway Protocol (protokol vonkajšej brány)
EISA – počítačová zbernica: Extended Industry Standard Architecture (predĺžená architektúra priemyselného štandardu)
EIST – procesorová technológia: Enhanced Intel SpeedStep Technology (rozšírená technológia Intel SpeedStep)
EMS – Expanded Memory Specification (rozšírená pamäťová špecifikácia)
ENIAC – počítač Electronic Numerical Integrator And Computer (elektronický číselný integrátor a počítač)
EOI – End Of Interrupt (koniec prerušenia)
EPROM – Erasable Programmable ROM (mazateľná a programovateľná pamäť ROM)
FAT – tabuľka súborového systému: File Allocation Table (tabuľka alokácií súborov)
FDD – disketová jednotka: Floppy Disk Drive (pružná disketová jednotka)

FDDI – počítačová sieť: Fiber Distributed Data Interface (Vláknom distribuované dátové rozhranie)
FDMA – Frequency Division Multiple Access (frekvenčný multiplex)
FIFO – First In – First Out (prvý dnu – prvý von)
FSB – počítačová zbernica: Front Side Bus (zbernica prednej strany)
FTDI – Future Technology Devices International
FTP – protokol: File Transfer Protocol (protokol určený k prenosu súborov)
GGP – protokol: Gateway-to-Gateway Protocol (protokol medzi bránami)
HAL – Hardware Abstraction Layer (abstraktná hardvérová vrstva)
HD – High Definiton (vysoké rozlíšenie)
HDD – disk: Hard Disk Drive (pevný disk)
HDMI – grafické rozhranie: High-Definition Multimedia Interface (Multimediálne rozhranie s vysokým rozlíšením)
HDTV – High-definition television (televízia s vysokým rozlíšením)
HGC – grafická karta: Hercules graphic card (grafická karta Hercules)
Hz – Hertz, základná jednotka frekvencie, vedľajšie jednotky: mHz, kHz, MHz, GHz
I/O – Input/Output (vstup/výstup)
IBM – korporácia (spoločnosť): International Business Machines Corporation
ICH – Input/Output Controler Hub (vstupno-výstupný radič a rozdeľovač)
ICMP – protokol: Internet Control Message Protocol (internetový protokol kontrolnej správy)
ICW – Instruction Command Word (inštrukčné príkazové slovo)
IDE – Integrated Development Environment (integrované vývojové prostredie)
IEEE – Institute of Electrical and Electronics Engineers (Inštitút pre elektrotechnické a elektronické inžinierstvo)
IFSM – Installable File System Manager (správca inštalovateľných súborových systémov)
IGP – protokol: Interior Gateway Protocol (protokol vnútornej brány)
IP – protokol: Internet Protocol (internetový protokol)
IPS – Instructions Per Second, počet inštrukcií za sekundu, vedľajšie jednotky: MIPS, GIPS
IRDA – Infrared Data Association (asociácia prenosu dát infračerveným lúčom)
IRQ – Interrupt Request (požiadavka o prerušenie)
ISA – počítačová zbernica: Industry Standard Architecture (architektúra priemyselného štandardu)
ISDN – Integrated Services Digital Network (Digitálna sieť integrovaných služieb)
ISO/OSI – International Standard Organization / Open System Interconnection (medzinárodná organizácia štandardov / otvorený prepojovací systém)
ITO – Indium Tin Oxide, pevná zlúčenina oxidu inditého a oxidu ciničitého
JPEG – rastrový formát digitálnych obrázkov: Joint Photographic Experts Group
K – kolík
KZD – koncové zariadenie
l – liter, jednotka objemu, ďalšie jednotky: pl, ml
LAN – sieť: Local Area Network (lokálna sieť)
LAPIC – Local component of APIC (lokálny komponent APIC-u)
LB – počítačová zbernica: Local Bus (lokálna zbernica)
LCD – displej: Liquid Crystal Display (displej s kvapalnými kryštálmi)
LD – optický disk: Laser Disc (laserový disk)
LEC – Light-Emitting Cell (luminiscenčná bunka)
LED – dióda: Light-Emitting Diode (luminiscenčná dióda)
LPC – počítačová zbernica: Low Pin Count bus (zbernica s malým počtom pinov)

LSB – Least Significant Byte (spodný bajt)
LSI – hustota tranzistorov na procesore: Large Scale Integration (veľká integrácia)
m – meter, základná jednotka dĺžky v sústave SI, vedľajšie jednotky: pm, nm, µm, mm, cm, dm, km
MAC – Media Access Control address (riadiaca adresa pripojeného média, fyzická adresa)
MAP – počítačová sieť: Manufacturing Automation Protocol (protokol v automatizovanej priemyselnej výrobe)
MCGA – grafická karta: Multi-Color Graphic Adapter (viac farebný grafický adaptér)
MCH – Memory Controler Hub (pamäťový radič a rozdeľovač)
MDA – grafická karta: Monochrome display adapter (jednofarebný adaptér displeja)
MMC – pamäťová karta: Multimedia Card (multimediálna karta)
MOS – polovodič: Metal-Oxide Semiconductor
MP3 – záznam digitálneho zvuku: MPEG Audio Layer 3
MPEG – digitálny záznam videa: Moving Picture Experts Group
MS – pamäťová karta: Memory Stick
MS-DOS – diskový operačný systém: Microsoft – Disk Operating System
MSB – Most Significant Byte (vrchný bajt)
MSI – hustota tranzistorov na procesore: Medium Scale Integration (malá integrácia)
MSIL – Microsoft Intermediate Language (medzi jazyk spoločnosti Microsoft)
NFS – protokol: Network File System (systém súborov na sieti)
NRZ – kódovanie: Non return to zero (bez návratu k nule)
NRZI – kódovanie: Non return to zero, inverted (bez návratu k nule, invertované)
NTFS – súborový systém: New Technology File System (nová technológia súborového systému)
NVT – Network Virtual Terminal (sieťový virtuálny terminál)
OC – Operate Code (operačný kód)
OCW – Operate Command Word (operačné príkazové slovo)
OLED – dióda: Organic Light-Emitting Diode (organická luminiscenčná dióda)
OP – operačná pamäť
OS – operačný systém, Operating System
OTN – Optical Transport Network (optická transportná sieť)
PATA – počítačové rozhranie: Parallel ATA, paralelná ATA
PC – počítač: Personal Computer (osobný počítač)
PCI – počítačová zbernica: Peripheral Component Interconnect (prepájanie periférnych komponentov)
PCI-e – počítačová zbernica: Peripheral Component Interconnect – Express (expresné prepájanie periférnych komponentov)
PCL – PC Labcard (laboratórna karta pre osobný počítač)
PCU – časť procesora: Program Control Unit, riadiaca jednotka
PIC – radič prerušení: Programmable Interrupt Controller (Programovateľný radič prerušení)
PMOLED – zobrazovacia technológia: Passive-Matrix Organic Light-Emitting Diode (organická luminiscenčná dióda s pasívnou maticou)
PPP – protokol: Point-to-Point Protocol (protokol bod k bodu),
PPTP – protokol: Point-to-Point Tunneling Protocol (protokol bod k bodu pomocou tunela),
PROM – Programmable ROM (Programovateľná pamäť ROM)
PSK – kódovanie: Phase-Shift Keying (fázová modulácia)
RAID – Redundant Array of Independent Disks (redundantné pole nezávislých diskov)
RAM – operačná pamäť: Random Access Memory (pamäť s náhodným prístupom)

RARP – protokol: Reverse Address Resolution Protocol (reverzný protokol rozpoznávania adries)
RCA – analógové grafické rozhranie: Radio Corporation of America
RGB – kódovanie farieb: Red, Green, Blue (červená, zelená, modrá)
RISC – inštrukčný súbor: Reduced Instruction Set Computer (redukovaný počítačový inštrukčný súbor)
ROM – pamäť: Read Only Memory (pamäť určená iba k čítaniu)
RPC – Remote Procedure Call (vzdialené volanie procedúry)
RPR – protokol: Resilient Packet Ring (pružný paket kruhovej siete)
RTOS – Real Time Operating System (operačný systém reálneho času)
RWM – pamäť: Read/Write Memory (pamäť určená k zápisu a aj čítaniu)
s – sekunda, základná jednotka času v sústave SI, vedľajšie jednotky: (ps, ns, μs, ms)
SAS – počítačová zbernica: Serial Attached SCSI (sériovo zameraná zbernica SCSI)
SATA – počítačové rozhranie: Serial ATA, sériová ATA
SCSI – počítačová zbernica: Small Computer System Interface (rozhranie pre malý počítačový systém)
SD – pamäťová karta: Secure Digital
SDRAM – operačná pamäť: Synchronous Dynamic Random Access Memory (dynamická a synchronizovaná pamäť s náhodným prístupom)
SI – systém fyzykálnych jednotiek, Systѐme International (medzinárodný systém)
SIMD – počítačová archytektúra: Single Instruction, Multiple Data (jedna inštrukcia, viacero dát)
SIMM – operačná pamäť: Single In-line Memory Module (pamäťový modul radený v jednej rade)
SIPP – operačná pamäť: Single In-line Pin Package (puzdro pinov radených v jednej rade)
SMB – protokol: Server Message Block (sever blokových správ)
SMTP – protokol: Simple Mail Transfer Protocol (protokol jednoduchého prenosu pošty)
SPF – algoritmus: Shorted Path First (najskôr kratšia cesta)
SRAM – operačná pamäť: Static Random Access Memory (statická pamäť s náhodným prístupom)
SSA – počítačová zbernica: Serial Storage Architecture (architektúra sériového ukladania dát)
SSI – hustota tranzistorov na procesore: Small Scale Integration (malá integrácia)
SSD – disk: Solid State Drive (mechanika s nepohyblivým médiom)
STP – Shielded Twisted Pair (tienená dvojlinka), alebo protokol Spanning Tree Protocol
SXGA – grafické rozhranie (karta): Super eXtended VGA
TCP – protokol: Transmission Control Protocol (protokol riadeného prenosu)
TDMA – Time Division Multiple Access (časový multiplex)
TFTP – protokol: Trivial File Transfer Protocol (primitívny FTP protokol)
TLS – Transport Layer Security (zabezpečenie transportnej vrstvy)
TTF – typ fontu: True-Type Font
TTL – Transistor-Transistor-Logic (tranzistorovo-tranzistorová logika)
UART – Universal Asynchronous Receiver and Transmitter (univerzálny asynchrónny prijímač a vysielač)
UDP – protokol: user datagram protocol (užívateľský datagramový protokol)
UFS – súborový systém: Unix File System (súborový systém OS Unix)
ULSI – hustota tranzistorov na procesore: Ultra Large Scale Integration (ultra veľká integrácia)

USA – United States of America (Spojené štáty americké)
USART – Universal Synchronous / Asynchronous Receiver and Transmitter (univerzálny synchrónno-asynchrónny prijímač a vysielač)
USB – počítačové rozhranie: Universal Serial Bus (univerzálna sériová zbernica)
UTP – Unshielded Twisted Pair (netienená dvojlinka)
UUCP – Unix-to-Unix Copy (kopírovanie medzi dvoma operačnými systémami Unix)
UXGA – grafické rozhranie (karta): Ultra eXtended Graphics Array
UZD – ukončovacie zariadenie
V – volt, základná jednotka napätia, vedľajšie jednotky: mV, kV
V/V – vstup/výstup
VESA – Video Electronics Standards Association (asociácia štandardov pre video elektroniku)
VFS – súborový systém: Virtual File System (virtuálny súborový systém)
VGA – grafické rozhranie (karta): Video Graphics Array
VL BUS – počítačová zbernica: VESA Local Bus (lokálna zbernica VESA)
VLAN – Virtual Local Area Network (virtuálna lokálna sieť)
VLSI – hustota tranzistorov na procesore: Very Large Scale Integration (veľmi veľká integrácia)
VMM – Virtual Machine Manager (správca virtuálnych zariadení)
W – watt, základná jednotka výkonu, vedľajšie jednotky: kW, MW
XDR – eXternal Data Representation (externá reprezentácia dát)
XGA – grafické rozhranie (karta): eXtended Graphics Array
XMS – eXtended Memory Specification (predĺžená pamäťová špecifikácia)
XNS – protokol: Xerox Internet Protocol (internetový protokol od spoločnosti Xerox)
ZSSR – Zväz sovietskych socialistických republík

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitej literatúry

[1] ŠNOREK, M. 1992. Standardní rozhraní PC, 241 strán, ISBN 80-85424-80-0

[2] RICHTER, J. 1997. Windows pro pokročilé a experty, Architektúra 32bitových systémov Windows 95 a Windows NT. Prel. J. Veselský. 1. vydanie, Praha: Computer Press, 1997, 988 s., preklad z angličtiny, originálne: Advanced Windows, ISBN 80-85896-89-3

[3] BURGER, I. 1990. Stykové obvody mikropočítačov. Bratislava: Alfa, 1990. 312 s. ISBN 80-05-00615-2

[4] JADLOVSKÝ, J. 1994. Distribuované systémy riadenia s heterogénnou štruktúrou. Habilitačná práca. Košice: Technická univerzita, 1994.

[5] PINKER, J. 2004. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN, 2004. 160 s. ISBN 80-7300-110-1

[6] SKALICKÝ, P. 2000. Mikroprocesory řady 8051. Praha: BEN, 2000. 164 s. ISBN 80-86056-39-2

[7] WHITE, R. 2002. Jak fungují počítače. Praha: SoftPress s.r.o., 2002. 404 s. ISBN 80-86497-48-8

[8] KOZIOREK, J. – KOCIÁN J. – CHROMČÁK, L. – LÁRYŠ, T.: Distribuované systémy řízení, Ostrava: Technická univerzita Ostrava, 2011, 264 strán, ISBN 978-80-248-2599-1

[9] Advantech. 1990. PCL-812 enhanced multi-lab card user´s manual. Tajvan, 1990, 81 s., číslo vydania: 2003812000 Rev. A3

[10] Advantech. 1990. PCL-818 High-performance DAS card with programmable gain. Tajvan, 1995, 132 s., číslo vydania: 2003818082 4th Edition

[11] Advantech. PCI-1730/33/34 Simultaneous 4-CH Analog Input Card for the PCI bus, User Manual. Tajvan, 2002, 56 s., číslo vydania: 2003173000 1st Edition

[12] Advantech. 2012. PCIE-1730. Tajvan, 2012. 19 s.

[13] Advantech. 2012. UNO-4673A/4683 Series. Tajvan, 2012. 43 s.

[14] Philips Semiconductors. 1994. 80C51-Based 8-Bit Microcontrollers. U.S.A., 1994. 1408 strán

[15] BALOGH, R. – BÉLAI, I. – DORNER, J. – Drahoš, P. 2001. Priemyselné komunikácie. Bratislava: Vydavateľstvo STU, 2001. 166 s. ISBN 80-227-1600-6

[16] JADLOVSKÁ, A. – JADLOVSKÁ, S. 2013 Moderné metódy modelovania a riadenia nelineárnych systémov, FEI, TU Košice, elfa Ltd., Košice 2013, 279 s. ISBN 978-80-8086-228

[17] KOVÁČ, F. 1998. Distribuované riadiace systémy: 1.vydanie. Bratislava: Vydavateľstvo STU, 1998. 211 s. ISBN 80-227-1082-2

[18] FRANK, J. – DERFLER, J. – LES, F.: Jak pracují sítě, Ziff-Davis Press, 1993, Emeryville, Kalifornia, český preklad - UNIS Brno, 1994.

[19] JADLOVSKÝ, J. - ČOPÍK, M. - PAPCUN, P.: Distribuované systémy riadenia. 1. vydanie, Košice: elfa, 2013, 215 s., ISBN 978-80-8086-227-5.

[20] JANEČEK, J.: Distribuované systémy, Vydavatelství ČVUT Praha, 1993.

[21] VAVREČKOVÁ, Š.: Operační systémy, Ústav informatiky, Filozoficko-prírodovedecká fakulta, Slezská univerzita Opava, 2006.

[22] VAVREČKOVÁ, Š.: Operační systémy, Texty k prednáškam. Ústav informatiky, Filozoficko-prírodovedecká fakulta, Slezská univerzita Opava, 2007.

[23] 82C50A CMOS Asynchronous Communications Element Datasheet, March 1997.

[24] ŠNOREK, M. – SLAVÍK, P.: Programovaní obrazových adaptéru PC, Vydavateľstvo Grada, 1993.
[25] KRUGLINSKÝ, D. J. - SHEPHERD, G. – WINGO, S.: Programujeme v Microsoft Visual C++, Computer Pree, Praha, 2000.

doc. Ing. Ján Jadlovský, CSc., Ing. Peter Papcun

Počítačové systémy v riadení

Vydala Technická univerzita v Košiciach, Letná 9, 042 00 Košice
1. vydanie
Rok vydania 2015
Náklad 70 kusov
Počet strán 415
ISBN 978-80-553-2102-8