Videopress MON
Praha

Generální ředitelství
TESLA IE
Praha

SBORNÍK PŘEDNÁŠEK PRO PGS

"SPOJOVACÍ SYSTÉMY S PROGRAMOVÝM ŘÍZENÍM"

Díl V.

Ing. Emanuel Prager, CSc.
Speciální mikroelektronické obvody

Ing. Jiří Chod, CSc.
Mikroprocesorová technika II.

PRAHA 1985

Ing. Emanuel Prager, CSc.

Speciální mikroelektronické obvody

PRAHA 1985

1. Rozvoj mikroelektroniky a její vliv na strukturu, vlastnosti i perspektivu elektronických ústředen

1.1. Úvod

Rychlý rozvoj mikroelektronických obvodů se obráží i v rozvoji všech elektronických zařízení, nevyjímaje i elektronické telefonní ústředny. Mikroelektronika umožňuje nejen podstatně zmenšit rozměry zařízení, snížit příkon elektrické enenrgie, ale dovoluje i využívat principy, které byly sice dříve známé, ale ekonomicky a technologicky nevyužitelné. Jako příklad je možno jmenovat, že rozvoj číslicové spojovací techniky byl umožněn rozvojem mikroelektroniky, přestože principy číslicového spojování byly známy již dříve.

Mikroelektronika se postupně rozšiřuje do všech dílčích obvodů elektronických ústředen a to jak se spojováním na prostorovém principu, tak i se spojováním číslicovým.

Mikroelektronické obvody však nepronikají jen do oblastí vlastního spojování hovorů, ale i do jiných částí ústředen a to v řadě případů v mnohem větším počtu než pro vlastní spojovací účely.

Jednou z velmi důležitých oblastí jsou obvody účastnického rozhraní, t.j. účastnické sady, v nichž mikroelektronika dovoluje realizovat všechny potřebné funkce. Důležitost této oblasti využití vyplývá z toho, že počet mikroelektronických obvodů pro účastnické rozhraní odpovídá počtu účastnických přípojek v ústředně a jejich celková potřeba a tím i výrobní serie proto budou mimořádně velké.

Samozřejmou oblastí aplikace mikroelektronických obvodů je také řízení ústředen, v nichž se využívají jak mikro-

procesory, tak i polovodičové paměti a ostatní obvody, potřebné pro sestavu mikropočítače.

Kromě těchto oblastí je možné mikroelektronické obvody postupně využívat i pro signální účely /vysílače a přijímače kódu/, generátory signálních kmitočtů apod.

S přechodem na integrované systémy a sítě se pak objevují další aplikace, jako mikroelektronické obvody pro číslicový přenos po účastnickém vedení, specielní obvody pro centralizovanou signalizaci apod.

Je zřejmé, že počet aplikací mikroelektronických obvodů v elektronických telefonních ústřednách dále poroste s rozvojem nových polovodičových technologií i se zvětšováním hustoty integrace.

Přehled dnešních aplikací hlavních mikroelektronických obvodů v elektronických ústřednách je uveden na tab. 1. V ní je také uvedena přibližná četnost těchto obvodů, vztažená na jednu účastnickou přípojku. Z tabulky vyplývá, že hlavní těžiště aplikací mikroelektroniky v elektronických ústřednách není ve standardních mikroprocesorech a polovodičových pamětech, ale spíše v oblasti obvodů pro účastnická rozhraní, u kterých potřeba v budoucnu poroste lineárně s nárůstem objemu výroby těchto ústředen.

Problematice telekomunikačních integrovaných obvodů dnes věnují pozornost všechny světové polovodičové firmy a ve svých předpovědích předpokládají velký nárůst výroby těchto specielních obvodů v porovnání se standardními, např. mikroprocesorovými.

1.2. Rozvoj základních logických obvodů

Trend dalšího rozvoje mikroelektroniky je všeobecně určován některými vztahy, vyplývajícími z extrapolace dosavadního rozvoje, charakterizovaného především trvalým zvětšováním hustoty prvků /hradel - tranzistorů/ v jednom integrovaném obvodu. Dosavadní trend např. polovodičových pamětí od původních 8 bitových až po dnešní 64 kbitové /případně 256 kbitů/ na jednom čípu tento trend nejnázorněji potvrzuje.

Promítneme-li si zvětšování hustoty integrace do různých dalších ukazatelů, dojdeme k tomu, že např. velikost základního polovodičového prvku v integrovaném obvodu /čípu/ se bude pravděpodobně dále zmenšovat podle grafu na obr.1.1, podle kterého byla v roce 1980 zvládnuta technologie 3 mikronová a do roku 1990 se očekává dosáhnutí 1 mikronové. Tomu odpovídá vzrůst počtu prvků na jednom čípu prakticky o 2 řády. Celkově je tento trend vyznačen na obr.1.2 , ukazující vzájemnou vazbu všech hlavních parametrů v průběhu vývoje a předpokládaný rozvoj do roku 2000.

Se stálým růstem hustoty integrace a novými technologiemi se mění i některé další parametry. Jedním z nejdůležitějších je snižování příkonu. Bez tohoto snižování příkonu na jeden prvek by totiž prakticky nebylo možné ani hustotu integrace zvyšovat, bereme-li v úvahu, že vyzářený výkon z jednoho integrovaného obvodu nemůže překročit určitou velikost. Při velikosti vyzářeného výkonu 1 W to znamená při hustotě 10^6 prvků, je možné připustit příkon na jeden prvek 10^{-6} W /to jsou hodnoty dosahované v roce 1980/. Při dalším růstu hustoty integrace musí pak úměrně klesat i příkon na jeden

prvek, aby obvody byly vůbec použitelné.

Trendy poklesu příkonu jsou charakterizovány např. u základní řady TTL logicky

Vývoj od normální technologie TTL šel nejprve cestou zvýšení rychlosti, ovšem za cenu zvýšení příkonu /řada S - Shottkyho TTL/. Dalším typem byla řada AS /Advanded Shottky/, vyznačující se dalším zvětšením rychlosti bez podstatného zvětšení příkonu.

Později následovala LS /Low Power Shottky/, u níž rychlost byla prakticky shodná jako u základní řady TTL, ale spotřeba klesla asi 5krát. Poslední z této řady logiky TTL je řada ALS /Advanced Low Power Shottky/, vyznačující se jak nízkou spotřebou /asi desetinovou proti základní řadě TTL/, tak i zvětšenou rychlostí /dvaapůlkrát/.

Současně se snižováním příkonu na jeden prvek v integrovaném obvodě musí klesat i cena, aby používání nových obvodů s vyšší hustotou integrace bylo ekonomicky zdůvodnitelné. Obecný trend, naznačený na obr.1.3 ukazuje tento poměrný pokles z hlediska snižování nákladů na realizaci jednoho prvku v integrovaném obvodu. Při uvažování skutečných cen lze tento pokles brát v úvahu při zaběhnuté výrobě a technologii, t.j. nikoliv na začátku zavedení nové technologie, nebo nové řady obvodů. Obr.1.4 ukazuje příklad, jak se měnila cena integrovaných pamětí v průběhu doby. Tento pokles platí samozřejmě pro obvody, vyráběné hromadnou technologií při velkých seriích. Cena pro maloseriové výrobky nebo při malých odběrech bývá obvykle vyšší.

Tento rychlý pokles cen zavedeného typu IO je na jedné straně vynucen tím, že na trh velmi rychle přicházejí IO

s větší hustotou integrace /nejlépe je tento vývoj vidět u polovodičových pamětí/, takže by dřívější výrobek na trhu při stálých cenách neobstál. Na druhé straně však tento trend má vliv i na konstrukci zařízení, kde konstruktér v každém okamžiku musí zvažovat použití lacinějšího obvodu s nižším stupněm integrace nebo použití prvku dražšího, ale prostorově a obvodově výhodnějšího s vyšším stupněm integrace. Toto hledisko má ovšem i negativní vliv na konstrukci zařízení. Ukazuje se, že rychlý trend rozvoje mikroelektronických obvodů vede k určité bezradnosti při vývoji zařízení u nichž doba vývoje je relativně dlouhá, např. 5 let do zavedení výroby. Při použití mikroelektronických prvků, které jsou při zahajování vývoje běžné, dostává se konstruktér při ukončení vývoje a zavádění výroby u takových zařízení často do situace, že zařízení je již normálně zastaralé, z hlediska použitých mikroelektronických prvků a že dokonce často ani výrobce mikroelektronických prvků nemůže zaručit dodávky obvodů ve vyvinutém zařízení použitých. I když samozřejmě přechod na modernější prvky je vždy dodatečně možný, vede často i ke strukturálním změnám zařízení a tím vlastně prodlužování vývoje.

Tato skutečnost je celosvětově známá a vede k určitým rozpakům při zahajování vývoje. Je nutné správně předpovídat trend rozvoje mikroelektroniky a při vývoji elektronických zařízení volit taková řešení, která dovolí postupné využívání obvodů vyšší integrace tak, aby bylo možné operativně v jednotlivých fázích vývoje a výroby přecházet od prvků jednodušších na složitější.

Tato situace je kritická především u telekomunikačních zařízení, u nichž vývoj trvá poměrně dlouho, vzhledem ke složitosti těchto zařízení i vzhledem k potřebě dlouhého provozního

ověřování a přizpůsobování se provozním potřebám a u kterých se očekává, že zařízení zavedená do výroby budou vyráběna nejméně 10 až 15 let a provozována 30 let. Z toho vyplývající požadavky na výrobu mikroelektronických obvodů z hlediska zajištění výroby telekomunikačních zařízení i z hlediska zajištění potřebného sortimentu náhradních součástí jsou dnes téměř nepředstavitelné .

I když lze předpokládat další zvyšování hustoty integrace podle uvedených zákonitostí, jsou zde i určité hranice z hlediska možností využití, které pravděpodobně povedou ke zpomalení dalšího rozvoje mikroelektroniky pro běžné použití. Přitom samozřejmě z hlediska různých specielních zařízení se bude asi tento vývojový trend dále uplatňovat a zpětně pak v budoucnu se změní i skokem možnosti využití i v běžných zařízeních.

1.3. Rozvoj mikroprocesorů

Mikropočítače, které se dnes běžně využívají pro konstrukci řídících obvodů elektronických telefonních ústředen procházejí trvalým vývojem, charakterizovaným jak zvětšením počtu bitů ve slově, které mikroprocesor paralelně zpracovává, tak i rychlostí, počtem a charakterem základních instrukcí a konečně i používáním dalších obvodů na čipu mikroprocesoru /tzv. jednočípový mikropočítač/.

V současné době se pro konstrukci řídících obvodů využívají převážně 8-bitové mikroprocesory, představované např. typy Intel 8080, Motorola 6800, Zilog 8080 a pod. Charakteristické parametry např. mikroprocesoru Intel 8080 /ekvivalent se vyrábí také v ČSSR/ jsou:

velikost slova 8 bitů

rychlost /doba cyklu/ 2 us

počet druhů instrukcí 78

možnost přímého adresování paměti do 64 K

Od konce sedmdesátých let se začaly objevovat mikroprocesory 16-bitové, představující nejen zvětšení hustoty integrace, ale i vyšší pracovní rychlosti. Typickými představiteli jsou např. Intel 8086, Motorola 68000a pod. Hlavní parametry typu Intel 8086 jsou :

velikost slova 16 bitů

rychlost 0,5 us

počet instrukcí 92

možnost přímého adresování pamětí do 1 Mbyte

Všechny uvedené mikroprocesory se vyrábějí v technologii MOS.

Vývoj pokračoval dále a v 80tých letech se začaly objevovat mikroprocesory 32-bitové, které svou výkonností již odpovídají procesorům dřívějších velkých počítačů např. 3. generace. Těžko lze odhadnout další vývoj , je však možné očekávat, že počet 32 bitů ve slově, zpracovaném paralelně v mikroprocesoru bude po delší dobu tvořit maximum a že další vývoj půjde spíše cestou dalšího zdokonalování technologie, snižování spotřeby, zvyšování počtu instrukcí a pod.

Paralelně s tímto vývojem mikroprocesorů probíhá i vývoj mikropočítačů na jednom čípu, dnes charakterizovaných typy např. 8035, 8048. Uvedené jednočípové mikropočítače obsahují tyto podsystémy:

- procesor
- programovou paměť /u typu 8048 - 1K, u typu 8035 není/
- paměť RAM s kapacitou 64 bytů

- tři programovatelné obvody I/O
- programovatelný generátor časových prodlev
- řadič přerušení a zdroj hodinových signálů

S těmito jednočípovými mikropočítači lze realizovat jednodušší řídící obvody, často plnící jen některé základní funkce, případně jako doplněk základních řídících mikroprocesorů. Realizují se tak např. tzv. periferní procesory pro předzpracování vstupních dat, komunikační procesory, zajišťující některé specifické funkce vstupního a výstupního zpracování dat a pod.

Každý typ mikroprocesoru má dále řadu podpůrných obvodů, které slouží ke kompletaci mikropočítače. Patří k nim jednak polovodičové paměti a jednak:

- systémové obvody /řadiče, zdroje hodinových impulzů/
- vazební obvody pro číslicové vstupy a výstupy
- vazební obvody pro přídavná zařízení /kodéry pro klávesnice, programovatelné vazební obvody, řadiče diskových pamětí apod/.
- vazební obvody pro komunikaci /asynchronní komunikační obvod, programovatelné komunikační obvody, komunikační řadiče a pod/.

Tyto integrované obvody, plnící specifické funkce pak umožňují, že lze základní sestavu řídící jednotky rozmístit např. na jedné desce /deskové mikropočítače/, standardně vyráběné a používané pro různé aplikace. Tyto deskové mikropočítače obsahují vedle mikroprocesoru, přídavných obvodů, také základní vybavení programových i datových pamětí. Při potřebě větší paměťové kapacity lze pak tyto deskové mikropočítače rozšířit o další paměťové jednotky.

Mikroprocesory mají význam především pro decentralizované řízení spojovacích systémů. Jednotlivé moduly systému mají své mikropočítačové jednotky, navzájem spolupracující. Jejich počet

může podle velikosti základních modulů dosáhnout až 100 pro ústřednu o kapacitě kolem 10000 přípojek, t.j. 0,01 mikropočítačů na přípojku.

Rozvoj mikroprocesorů z hlediska velikosti i zpracovávaného slova má vliv na stupeň desentralizace systému. Zatím co při použití mikropočítačů 8-bitových lze realizovat pouze malé ústředny, řádově sta přípojek nebo v systému větší kapacity moduly o velikosti např. 128 přípojek, je možné při použití 16-bitových mikroprocesorů již velikost základního modulu ústředny zvětšit řádově na 1000 přípojek. Vyplývá to z toho, že celková výkonnost 16-bitového mikroprocesoru /vycházející ze čtyřnásobné rychlosti a dvojnásobného slova/ je přibližně osminásobná.

Vedle vlivu na strukturu systému má vývoj mikroprocesorů směrem k většímu počtu bitů ve slově i vliv na celkovou velikost řídících jednotek a tedy i na velikost ústředny.

1.4. Polovodičové paměti

Snad nejvýrazněji se projevuje rozvoj mikroelektroniky v polovodičových pamětech, které pak svými parametry ovlivňují vlastnosti mikropočítačů, jak z hlediska objemu zařízení, pracovní rychlosti, spotřeby elektrické energie, tak i samozřejmě možnostmi zvětšení kapacity pamětí.

Polovodičové paměti obecně dělíme do několika skupin podle způsobu činnosti, druhu záznamu i podle druhu použité polovodičové technologie:

- paměri RAM /random acces memory/
- paměti ROM / read only memory/

s variantami PROM, EPROM, EAROM atd.

Z hlediska použité technologie existují paměti bipolární,

které jsou velmi rychlé a paměti vyráběné technologií MOS. Bipolární paměti mají většinou menší kapacitu a používají se v rychlých mikroprocesorových obvodech.

Z hlediska způsobu záznamu se pro paměti RAM používají dva hlavní způsoby:

- záznam statický
- záznam dynamický

Paměti statické umožňují záznam bez nutnosti jeho obnovování. Statické paměti jsou tvořeny v podstatě klopnými obvody /v technologii MOS i bipolární/. V důsledku toho je potřebná plocha čípu relativně velká, takže se obecně dosahuje při dané velikosti čípu menší paměťové kapacity, než u pamětí dynamických.

Paměti dynamické vyžadují trvalé obnovování záznamu /refreshing/. Záznam je na principu kapacitním, t.j. náboj kondenzátoru ovládá vlastní paměťový element, např. tranzistor. Vzhledem k tomu, že paměťové buňky jsou menší, vychází dynamické paměti výhodnější. Obnovování záznamu /např. 200 - 500 krát za s/ nepřináší žádné potíže, protože většina mikroprocesorů může zajistit refrešovací systém bez obtíží.

V současné době se nejvíce používá pro konstrukci řídících mikropočítačů /i ve spojovací technice/

pro datové paměti - paměti RAM dynamické

pro programové paměti - paměti EPROM

Z hlediska kapacity těchto pamětí se v roce 1983 dosahovalo

- u pamětí RAM kapacity 64 kbit na jednom čípu
- u pamětí EPROM kapacity 64 kbitů

Při tom je nutno mít na paměti, že kapacita pamětí se v minulých letech trvale zvětšovala tak, že přibližně každé

2 roky se zvětšila čtyřikrát. Tento vzrůst kapacity pamětí na čípu měl vliv nejen na konstrukci elektronických zařízení, ale i na jejich cenu, protože větší typ pamětí byl vždy levnější, vztaženo na cenu jednoho bitu paměti.

Z hlediska konstrukce spojovacích systémů je nutné se ještě zmínit o tzv. masových pamětech, t.j. pamětech s poměrně velkou kapacitou, u nichž doba přístupu k paměti může být relativně větší než u pamětí pracovních a programových. Jsou to většinou paměti určené pro nespřaženou činnost /off line/, t.j. pro záznam dat i programů, které jsou zapotřebí pouze výjimečně.

V současné době se pro tyto účely používají ve spojovacích systémech různé druhy paměti s magnetickým záznamem na pohyblivých mediích /páskové paměti, kazetové paměti, diskové paměti, diskotékové paměti, paměti typu Winchester a pod./. Všechny tyto paměti mají poměrně velkou kapacitu /např. diskové paměti mají kapacitu desítky Mbyte/. Jejich mechanický pohon však podstatně snižuje jejich životnost i střední dobu mezi poruchami. Přesto se dnes stále v elektronických systémech využívají, jejich záznam je však většinou zdvojen.

Podstatně nové řešení se očekává od nepohyblivých bublinových pamětí, které byly objeveny nedávno a u nichž se dnes již dosahuje kapacity 256 kbitů na jednom čípu, případně v hybridním provedení 1 mil. bitů v jednom pouzdru. Doba přístupu k těmto pamětem je srovnatelná např. s diskovými pamětmi a je možné proto očekávat, že v budoucnu nahradí uvedené mechanicky poháněné typy pamětí. Některé japonské firmy již dnes zavádějí bublinové paměti do spojovacích systémů. Celkový vývoj v této oblasti ještě není uzavřen a dokonce se objevují u některých výrobců názory na nerentabilnost jejich výroby.

2. Mikroelektronické obvody pro účastnické rozhraní v elektronických telefonních ústřednách

2.1. Všeobecně

Dřívější pokusy nahradit releovou účastnickou sadu jinými elektronickými obvody ztroskotávaly pochopitelně předem na ekonomické stránce problému. To byl také jeden z důvodů, který znevýhodňoval nástup perspektivních elektronických telefonních ústředen a vedl např. k tomu, že se začala v těchto sadách používat jazýčková relé, různé magnetické prvky /ferrody a pod./ s cílem zmenšení rozměrů.

V dnešní době, zvláště při zavádění číslicového spojování /na principu PCM/ do telefonní spojovací techniky, se ukazuje, že účastnická sada musí plnit další funkce, t.j. převod z dvoudrátové hovorové cesty v ústředně na čtyřdrátovou, t.j. v podstatě zahrnout do sebe i tzv. vidlici. Vidlicové zapojení, které až dosud bylo v přenosové technice realizováno transformátorem by vedlo k dalšímu zvětšení účastnické sady.

Uvažování všech těchto faktorů i nových možností mikroelektronické součástkové základny vedlo k principielně novému návrhu účastnického vstupu do ústředny. Pro nová řešení byla standardně zavedena zkratka BORSHT pro analogově pracující ústředny a pro číslicové spojování BORSCHT. Tyto zkratky jsou odvozeny z anglického názvu hlavních funkcí účastnické sady:

B - stejnosměrné napájení /battery/
O - ochrana proti přepětí na vedení /overvoltage protection/
R - vyzvánění /ringing/
S - dohled /supervision/
C - kodér /coding/
H - vidlicové zapojení /hybrid/ T - zkoušení /testing/

Kromě označení BORSHT se někdy pro účastnické sady elektronických ústředen používá označení SLIC /subscriber line interface circuit/.

Zatímco u analogových telefonních ústředen se obvod SLIC nebo BORSHT stal jediným rozhraním mezi účastnickým vedením a elektronickým spojovacím polem ústředny, je nutné v číslicových ústřednách zajistit v účastnické sadě ještě převod analogového signálu na číslicový, se kterým se pak vstupuje do spojovacího pole.

Účastnická sada pro číslicové spojování se tedy skládá ze tří základních mikroelektronických obvodů:

- SLIC
- filtr
- kodek

Tato základní sestava mikroelektronických obvodů pro účastnický vstup se pak v různých obměnách objevuje v moderních číslicových ústřednách. Podle provedení kodeku se někdy doplňuje o řídící obvod, přiřazující časové kanály /tzv. obvod TSAC/. V novějších provedeních se již objevuje filtr a kodek v jediném integrovaném mikroelektronickém obvodu /označovaném např. monočíp/.

V dalším bude podrobněji uvedena základní řada obvodů firmy Motorola.

Blokové schema účastnické sady pro číslicové ústředny s obvody této firmy je na obr. 2.1. Na účastnický obvod SLIC navazuje obousměrný filtr MC 14414, omezující kmitočtové pásmo pro kodek, dále kodek MC 14404, který pracuje současně jako kodér i dekodér /oba směry přenosu/. Potřebné časové kanály mu dodává řadič TSAC, typ MC 14418.

Kromě těchto základních integrovaných obvodů je nutné ještě pro činnost kodeku doplnit specielní obvod-zdroj referenčního na-

pětí typ MC 1403.

Jak je z obrázku zřejmé, rozděluje v tomto případě obvod SLIC hovorovou dvoudrátovou cestu z účastnického vedení na čtyřdrátovou směrem do ústředny. Na jeho výstupu jsou tedy dva oddělené směry přenosu, které pak jsou v navazujícím filtru a kodeku zpracovány odděleně.

V dalším je uveden podrobnější popis jednotlivých obvodů.

2.2. Integrovaný obvod SLIC

Mikroelektronické provedení obvodů SLIC pro účastnické sady má několik firem. Popišme si jedno z provedení firmy Motorola, označované MC 3419. Principielní zapojení funkční vychází z obr.2.2 . Vstupní obvod tohoto integrovaného obvodu zajišťuje jak sekundární ochranu proti přepětí, tak i vlastní napájení účastnické smyčky /přes vnější obvody/ a nastavení tohoto napájecího proudu i impedance přizpůsobovací.

Za vstupním obvodem následuje vidlicové zapojení pro převedení dvoudrátového účastnického vedení na čtyřdrátové nesymetrické zapojenní, t.j. rozdělení obou směrů přenosu hovorového signálu. Ve vysílací větvi je zapojen zesilovač s nastavitelným ziskem. Podobné nastavení zisku je i v přijímací větvi. Nastavení zisku v obou směrech přenosu lze provádět nezávisle.

Mezi oběma větvemi, přijímací i vysílací je zapojen obvod, sloužící k vyvážení vlastní vidlice /spolu s vnějším obvodem/. V něm se rovněž zajišťuje detekce změny napětí v účastnické smyčce pro zjištění stavu vyzvednutí mikroetelefonu, pro detekci volání, volby, přihlášení a pod.

S ohledem na napájecí napětí 48 V, které může dostoupit až 56 V se pro konstrukci tohoto mikroelektronického obvodu použilo bipolární technologie s vyšším napětím. Obvod je umístěn

v pouzdře s 18 vývody. Zapojení je navrženo tak, aby v případě, že je účastnické vedení v klidovém stavu, mohl být ztrátový příkon snížen na minimum /power down/. Při tom však zůstává účastnické vedení pod kontrolou je možné detekovat změnu stavu při sejmutí mikrotelefonu. Klidová spotřeba obvodu je 1 mW.

Na obr. 2.3 je naznačeno celé zapojení účastnické sady s tímto integrovaným obvodem. Vlastní obvod SLIC je doplněn ještě diodovým obvodem MAD 220, který zajišťuje ochranu proti přepětí /jako doplněk k ochraně výbojkou, která musí být zapojena na vstupu účastnického vedení do ústředny/. Protože při napájení účastnického vedení stejnosměrným proudem vzniká poměrně velký ztrátový výkon až 2 W, který nelze v běžném pouzdru integrovaného obvodu vyzářit, jsou výkonové části napájecího obvodu /tranzistory v Darlingtonově zapojení/ umístěny mimo pouzdru integrovaného obvodu. Jsou to přídavné obvody označované jako MJD 270 a 271.

Pro zmenšení ztrátového příkonu se často volí zapojení s konstantním napájecím proudem, dodávaným ze zdroje, který je tepelně kompenzován. Používá se k tomu specielní diody, zapojené mimo integrovaný obvod SLIC.

Jak je zřejmé z obrázku, i u tohoto integrovaného provedení obvodu SLIC se vyzváněcí proud vysílá přes zvláštní kontakty relé, mimo vlastní integrovaný obvod.

2.3. Vzorkovací filtr pro PCM

Pro zajištění převodu analogového signálu na číslicový je zapotřebí vedle vlastního převodníku - kodeku, ještě vstupní a výstupní filtry, které slouží jednak k omezení kmitočtového pásma ve vysílacím směru a jednak k obnově signálu po dekódování v přijímacím směru. Tuto činnost obvykle zajišťují kombino-

vané vysílací a přijímací filtry v jediném IO.

Jako příklad je uveden popis filtru PCM v provedení firmy Motorola, typ MC 14414. Je to filtr vyrobený na principu spínaných kondenzátorů v technologii CMOS. Vyznačuje se malou proudovou spotřebou, která v pracovním režimu činí asi 25 mW a v klidovém stavu jen 0,5 mW.

Blokové schema tohoto filtru je na obr. 2.3. Jsou to v podstatě dvě samostatné větve filtru, vysílací a přijímací, spolu se společnou řídící částí.

Vysílací filtr

Hlavním úkolem vysílacího filtru je omezit vstupní signál pro kodek na pásmo 4 kHz, které může být v kodeku vzorkováno kmitočtem 8 kHz. Je nutné při tom omezit pronikání jiných signálů o vyšších kmitočtech nad 4 kHz, které by působily rušivě při kódovacím procesu. /Někdy se tento filtr nazývá anti-alias filtr/. Rovněž je nutné potlačit kmitočty pod 60 Hz, t.j. rušivé kmitočty ze sítě, případně z vyzváněcího signálu a pod.

Vysílací filtr se skládá z dolní propusti, tvořené 5-pólovými eliptickými sekcemi pro omezení pásma nad 4 kHz a navazující horní propusti, tvořené Chebyševovým 3-pólovým filtrem pro omezení spodních kmitočtů pod 60 Hz.

Přijímací filtr

Úkolem přijímacího filtru je vyhladit stupňový průběh signálu vznikající po dekódování na výstupu kodeku, t.j. odstranit z něj vyšší kmitočtové složky. Z toho důvodu je hlavní částí přijímacího filtru dolní propust, která je podobná jako u vysílacího filtru. Je rovněž tvořena 5-pólovými eliptickými sekcemi, pracujícími se vzorkovacím kmitočtem 128 kHz.

Dále obsahuje přijímací filtr obvod pro kompenzaci zkreslení typu sin x/x, která normálně doprovází modulaci PAM.

Integrované provedení filtru PCM je dále doplněno dvěma operačními zesilovači. Operační zesilovač A na výstupu přijímacího filtru zajišťuje dostatečný výstupní výkon s nastavitelným ziskem. Je schopen dodávat dostatečný výkon do zátěže 600 až 900 Ω /t.j. na účastnické vedení/. Operační zesilovač B na vstupu vysílacího filtru zajišťuje vyrovnání ve čtyřdrátové cestě a dovoluje nezávislé nastavení zisku vysílací cesty.

Integrovaný filtr PCM, obsahující oba filtry je řízen z řídící logiky, na kterou jsou předávány jak základní vzorkovací kmitočet 128 kHz, tak i rámcová synchronizace 8 kHz. Z řídící logiky může být rovněž řízeno snížení příkonu celého obvodu v klidovém stavu /power down/.

2.4. Jednokanálový integrovaný kodek

Integrovaný kodek obsahuje obvody potřebné pro převod analogového signálu na číslicový PCM ve vysílacím směru a obráceně v přijímacím směru. Zastupuje tedy kodér i dekodér pro čtyřdrátový přenos. Pracuje se vzorkovacím kmitočtem 8 kHz a s 8 bitovým kódováním hovorového signálu. Obvykle bývá alternativně proveden pro oba způsoby komprese hovorového signálu a to podle zákona /t.j. podle americké normy/ a podle zákona A /podle CCITT a evropské normy/.

Jako příklad je uveden popis činnosti kodeku firmy Motorola, typ MC 14407. Činnost kodeků jiných firem je podobná, i když vnitřní provedení obvodové může být dosti rozdílné podle použité technologie, kterou používají různí výrobci mikroelektronických obvodů.

Hlavní charakteristiky popisovaného kodeku lze shrnout do těchto bodů:

- nezávisle volitelná rychlost od 64 kbit/s do 3,088 Mbit/s /umožňuje vytvářet rámce systémů pro 24 až 48 kanálů/
- kanálový plně duplexní provoz
- jednotné napájení 10 - 16 V
- nízká spotřeba asi 80 mW /v úsporném režimu 1 mW/

Blokové schema integrovaného kodeku je na obr. 2.4. Ve vysílacím směru přichází analogový signál na vstup vzorkovacího obvodu /vstup AD 1/. Výstup tohoto vzorkovacího obvodu přichází do analogového subsystému, pracujícího jako převodník analogového napětí na proud /s pomocí odporu R/, i jako obvod pro automatické nastavení nuly a obvod pro převod analogového vzorku do výstupního vzorkovacího obvodu při přijímacím směru pomocí kondenzátoru C.

Analogový subsystém je složen z operačních zesilovačů a analogových spínacích obvodů. Výstupní proud tohoto obvodu, úměrný velikosti amplitudy vzorku přichází pak do analogově číslicového převodníku.

V tomto převodníku se velikost proudu porovnává s velikostí referenčního proudu získaného z externího zdroje referenčního napětí. Porovnáváním a současně s předepsanou kompresí se v tomto obvodu vytváří 8-bitové slovo. Kompresní charakteristika se nastavuje vhodným napětím na vstupu u/A, ze kterého se přepínají vnitřní obvody převodníku.

Vytvořené 8-bitové slovo se pak přivádí do paralelního registru SAR a předchozí obvody se uvedou do klidového stavu, aby bylo možné zpracovávat nový vzorek analogového signálu.

Zaznamenané slovo v registru se pak ve vhodném okamžiku pře-

dá v paralelním tvaru do obvodů pro řízení vysílání dat, ze kterého se vyšle v seriovém tvaru ve zvolené časové poloze /výstup TDD/. Rychlost vysílání dat je řízena jednak vstupem datových hodin a jednak je rámcově synchronizována. TDC určuje rychlost přenosu a TDE určuje polohu slova v rámci /t.j. výstupní kanál/.

V přijímacím směru přechází ze společné sběrnice signální bity v seriovém tvaru po vodiči RDD na obvod pro řízení příjmu dat. Toto řízení je opět ovlivňováno vstupem datových hodin RDC a označením kanálu v rámci přenosu /vstup RDE/. Přijaté 8-bitové slovo se v paralelním tvaru předá do registru SAR a odtud se pomocí převodníku přenásí do analogového subsystému již jako amplituda vzorku přijímaného výstupního signálu. Tato amplituda se přenese do výstupního vzorkovacího obvodu s pamětí /kodenzátor C/, který tvoří regeneraci analogového signálu. Výstupní analogový signál v přijímacím směru se objeví na vývodu označeném ADO.

Činnost celého kodeku je řízena řídícími obvody, které zajišťují :

- základní synchronizaci /MSI /
- řízení posloupnosti převodu pro duplexní činnost CCI /vzorkovací kmitočet 128 kHz/
- uvedení obvodu do klidového stavu /power down/

Řídící obvody řídí i signální logiku, používanou při kanálové signalizaci /osmým bitem v každém kanálu/. Signální logika slouží ke vkládání signálního bitu do výstupního slova při vasílání a k vydělení tohoto bitu při příjmu. V našich spojovacích přenosových systémech podle doporučení CCITT se nepoužívá tohoto druhu signalizace, t.j. pracuje se s 8-bitovým kódováním a přenosem signalizace v jednom, k tomu účelu vyhrazeném kanálu společně pro všechny ostatní kanály.

Činnostíkodeku se rovněž generuje tzv. analogová zem, což je

poloviční napájecí napětí. Toto referenční napětí se pak používá i v jiných obvodech účastnické sady.

Kodek firmy Motorola MC 14407 je zhotoven v technologii CMOS. Umožňuje spolupráci s ostatními obvody této firmy, především vstupním a výstupním filtrem na jedné straně a se standardními obvody sběrnice výstupního číslicového signálu PCM na straně druhé.

2.5. Řídící obvod TSAC

Řídící obvod TSAC /Time Slot Aligner circuit/ je zařízení, dovolující trvalé přiřazování časového kanálu pro kodek na podkladě řídících informací z mikroprocesorového řízení.

Řídící obvod obsahuje dvě samostatné části pro vysílací směr kodeku a pro přijímací. Blokové schema je na obr. 2.5.

Přiřazování určitého kanálu pro kodek je dáno naprogramováním této časové polohy. Naprogramování se děje předáním seriové informace o časovém kanálu, který má být v daném kodeku používán do posuvných registrů, sloužících jako paměť. V této paměti je zaznamenáno číslo kanálu tak, jak přijde z mikroprocesorového řízení. Úkolem obvodu TSAC je podle tohoto čísla kanálu vyslat v požadovaný časový okamžik uvolňovací signál jak pro přijímací, tak i vysílací část kodeku.

Toto generování uvolňovacího signálu se děje tak, že porovnávací obvod dostává z pevného generátoru adres postupně čísla jednotlivých kanálů a současně dostává číslo zvoleného kanálu z posuvného registru. V případě koincidence obou adres vyšle porovnávací obvod signál na výstupy RXE a TXE, který časově odpovídá zaznamenanému číslu kanálu.

Princip řízení tohoto obvodu z mikroprocesorového vstupu je znázorněn na obr. 2,6, kde je naznačen vstup informace z mikro-

procesorového řízení. Je zde naznačeno 128 obvodů TSAC, příslušejících 128 účastnickým sadám s kodeky. Všech 128 obvodů je rozděleno do 4 skupin po 32 a pro každou skupinu je jeden uvolňovací vstup CS 1 - 4. Vlastní data z mikroprocesorového řízení pak přichází po vstupech AD a DI. Vstup AD slouží k předání adresy příslušného obvodu TSAC ve skupině. K tomu je třeba 5 bitů /32 obvodů/. Zbývající 3 bity z 8-bitového slova slouží k předání vlastního čísla kanálu, vzhledem k tomu, že obvod TSAC umožňuje používat 64 časových poloh, je označení čísla kanálu 6-bitové. Zbývající 2 bity 8-bitového slova se rovněž využívají k pomocným účelům.

Informace, označující požadované číslo kanálu se zaznamená pouze v tom obvodu TSAC, který je adresován ze vstupu AD. Na blokovém schematu obvodu TSAC jsou zřejmé oba posuvné registry, sloužící k záznamu adresy i čísla kanálu.

Obvod TSAC je vyroben v technologii CMOS. Kromě základních funkcí přidělování časového kanálu umožňuje ještě některé další funkce potřebné pro činnost účastnické sady, jako např. řízení přenosu vyzváněcího proudu, předání povelu pro snížení příkonu ostatních obvodů účastnické sady v klidovém stavu a pod.

2.6. Integrovaný kodek a filtr

Vedle klasického uspořádání, používaného např. v popsané účastnické sadě Motorola, t.j. samostatných mikroelektronických obvodů pro filtr a kodek se postupem doby ukázaly možnosti vytvoření společného obvodu - kombinovaného filtru a kodeku v jediném mikroelektronickém obvodu /někdy nazývaném monočip/.

Příkladem může být obvod MC 14400 firmy Motorola, nebo podobné obvody jiných firem. Tento obvod vzniknul při zvětšování hustoty integrace v podstatě složením uvedených dvou dílčích ob-

vodů, filtru a kodeku, popsaných v předchozích kapitolách a objevují se v něm i stejné funkční celky.

Tento obvod, vytvořený sloučením obou funkcí na jednom čípu má také stejné vstupní a výstupní parametry, jako v předchozí kapitole popsané samostatné obvody filtr i kodek. Podobné typy obvodů mají i jiné firmy, např. AMI, typ S 3506. Tento integrovaný kodek obvod je uzpůsoben pro různá použití, mimo jiné i pro použití v telefonním přístroji.

Vedle normálních provedení filtrů PCM a kodeků se pro účastnické sady objevilo i principielně nové řešení, používající programovatelné signální procesory, které umožňují univerzálnější řešení skupiny filtr a kodek.

Jedním z takových je také integrovaný obvod, označovaný SLAC /subscriber line audio processing circuit/. Tento integrovaný obvod firmy AMD s typovým označením AM 7901 zajišťuje přímou přeměnu analogového signálu na signál PCM a naopak a navíc umožňuje přímé napojení na mikroprocesorový řídící obvod.

Blokové schema tohoto obvodu je na obr. 2.7. Skládá se ze tří základních sekcí, vysílacího procesoru, přijímacího procesoru a řídícího obvodu, který současně zajišťuje rozhraní k mikroprocesorovému řízení.

Vysílací sekce obsahuje tzv. "antialias filtr", t.j. pásmovou propust, interpolativní převodník analogočíslicový a signální procesor, který generuje buď lineární 16-bitový kód nebo 8-bitový s kompresí podle zákona u. Převodník je navržen tak, aby měl široký dynamický rozsah a velký poměr signálu k šumu. Signální procesor obsahuje aritmetickou jednotku, paměť RAM a ROM /t.j. ekvivalentní mikroprocesorovému systému/ a řídící logiku pro zajištění filtrových funkcí. Bloky označené B, X a GX jsou programovatelné filtry a jejich koeficienty jsou zaznamenány v paměti koeficien-

tů RAM. Filtr X je součástí korekčního obvodu pro korekci kmitočtové charakterjstiky. Filtr GX umožňuje uživateli programovat zisk v krocích po 0,1 dB ve vysílací větvi. Filtr B je řízen přijímacím signálním procesorem a zajišťuje při čtyřdrátovém přenosu vyrovnání. Nízkofrekvenční propust omezuje šířku pásma pro splnění přenosových parametrů. Horní propust potlačuje 15 Hz a kmitočty okolo 50 Hz.

Podobně je funkčně uspořádán i přijímací signální procesor. Jednotka řízení obsahuje již uvedené paměti RAM pro koeficienty vysílacích i přijímacích filtrů a potřebnou řídící logiku spojenou s přiřazováním kanálů. Dále obsahuje bistabilní obvody pro 5 výstupů C 1 až C 5, kterými je řízen připojený obvod SLIC, případně přes něj vyzváněcí relé nebo obecně 5 různých výstupů.

Tento nový typ kodeku, využívající principu signálního procesoru má výhodu v tom, že stejným obvodem lze nastavením příslušných koeficientů naprogramovat různé průběhy filtrů a přizpůsobit se tak podmínkám jednotlivých sítí. Tím se tento obvod stává univerzálním a nazývá se proto v komerčních prospektech "světovým čípem" /world chip/.

S podobným řešením přichází i jiné evropské firmy. Např. firma Eurotechnique přinesla na trh novou rodinu tzv. kodeků COMBO, vyznačujících se podobnými vlastnostmi, ale díky použité technologii P^2 CMOS mají malý ztrátový výkon /60 mW v pracovním režimu a 3 mW v klidovém stavu/. Tento typ kodeků se někdy nazývá CODEC a je použit v některých francouzských číslicových systémech.

2.7. Mikroelektronické obvody pro účastnickou sadu pro vyšší napětí

Při použití integrovaných obvodů standardních /např. řady

Motorola je nutné počítat s tím, že některé funkce /připojení zkušebních obvodů, připojení vyzváněcího signálu, přepólování smyčky a pod. a omezení přepětí vznikajícího na účastnickém vedení bude provedeno přídavnými obvody, nebo relé. Rovněž zvětšení napájecího napětí nad 48 V přináší u těchto obvodů potíže. Z toho důvodu se u většiny světových firem dnes sleduje intenzivně možnost vytvoření vstupních obvodů pro účastnickou sadu pro vyšší napětí.

Tuto otázku řeší různí zahraniční výrobci odlišně. Japonské firmy se soustředily na vývoj vstupní části účastnického stykového obvodu pro vyšší napětí /vyvinuly tzv. obvod RT/,který vedle elektronického připojení vyzváněcího proudu, testovacích obvodů a přepojování vodičů vytváří i základní přepěťovou ochranu. Toto řešení pak nevyžaduje u navazujících obvodů üpužití technologie pro vyšší napětí.

Nová koncepce účastnického rozhraní pro číslicové spojovací systémy /t.j. obsahující i kodek PCM/ vychází z použití obvodů pro vyyší napětí, které je možné použít na vstupu účastnického vedení bez zvláštního omezovače přepětí a tak nahradit dřívější zapojení s relé, omezovači napětí a pod. Do serie dnes již standardních obvodů tedy přistupuje další typ IO se specificky spínacími vlastnostmi.

Princip realizace jednotlivých funkcí BORSCHT pro nově navrhovanou řadu obvodů je na obr. 2.8, kde jsou rovněž naznačeny napěťové podmínky pro jednotlivé dílčí funkce a možnost soustředění kritických napěťových podmínek do jednoho ze vstupních obvodů, řešeného specielní technologií, což umožní, aby navazující obvody mohly být realizovány běžnou technologií.

Na vstupu účastnického vedení je zapojen obvod, označovaný RT pro připojení vyzváněcího signálu, zkušebního obvodu pro

připojení vyzváněcího signálu, zkušebního obvodu pro zkoušení ústředny a obvodů pro změnu polarity hovorových vodičů. Tento obvod tedy i podle názvu realizuje dvě hlavní funkce R a T z obvodu BORSCHT. Další funkce jsou pak soustředěny do navazujícího IO, ve kterém probíhá napájení účastnické smyčky, snímá se stav této smyčky a provádí se tramsformace dvoudrátového na čtyřdrátové spojení. Na tento obvod pak navazuje kodek /v provedení monočípu filtr a kodek/. Celou soupravu pak řídí obvod, označený CONT, který také zajišťuje rozhraní mezi účastnickou sadou a číslicovou datovou sběrnicí. Připojení zkušebních obvodů pro zkoušení vedení není zatím ještě integrováno na vstupním čípu a provádí se pomocí optočlenů, řízených z obvodu CONT.

Spínače typu PNPN snáší napětí až 400 V mezi anodou a katodou, Pro zajištění této odolnosti bylo použito technologie a dielektrickou izolací. Typická velikost jednotlivého spínače je 300 x 450 um, celková velikost integrovaného čípu s uvedenými 10 spínači a pomocnými obvody je 5 x 4,4 mm. Obvod je umístěn v pouzdru se 22 vývody. V propustném směru se u spínačů dosahuje hodnoty asi 5 Ω .

Rovněž jiné firmy se snaží dosáhnout vyšších napětí různými technologiemi, především pro zajištění přenosu vyzváněcího proudu /napětí do cca 150 V/ z obvodu SLIC. Např. firma Harris používá bipolární technologie s dielektrickou izolací pro konstrukci obvodu SLIC. Vyráběný obvod má orůrazné napětí 140 V.

Jiné firmy, např. ITT North Microsystems vyrábí hybridní provedení účastnické sady, obsahující všechny potřebné obvody včetně kodeku a sekundární ochrany pro 1,6 kV.

3. Integrované obvody pro spojovací pole elektronických telefonních ústředen

3.1. Úvod

Integrované obvody pro spojovací pole elektronických ústředen umožnily podstatné snížení objemu zařízení a přispěly k technické realizovatelnosti elektronických ústředen.

Integrované obvody pro analogově pracující elektronické ústředny se dnes používají v řadě mylých pobočkových ústředen a jsou to většinou matice 4x4, 8x4 nebo 3x8 v provedení jednovodičovém nebo dvouvodičovém, umístěné s příslušnými dekodéry adres na jednom čípu integrovaného obvodu. Přestože pokrok v této oblasti dosáhl relativně velkých úspěchů, nelze asi do budoucna očekávat větší rozšíření těchto obvodů s ohledem na očekávaný přechod na číslicové spojování.

Určitá perspektiva se očekává od konstrukce spínacích integrovaných matic, jejichž napěťová odolnost i parametry se přiblíží parametrům releových kontaktů a které bude možné potom připojovat přímo na vedení. To dovolí konstrukci různých hybridních provedení systémů /např. analogové koncentrační spojovací pole a číslicové spojování/. Podobné zapojení se dnes použilo již u moderního systému společnosti Bell, označovaného ESS 5, u něhož bylo použito spínacích matic pro napětí 400 V.

Rychlý rozvoj nastal i u spínacích obvodů pro číslicové spojovací systémy. Vyplývá to z charakteru tohoto spojovacího pole a z možnosti realizace jednotlivých částí /především časových článků T/ z polovodičových pamětí. Integrované spínací obvody jsou dnes již běžné pro kapacity 256 x 256 vstupů a výstupů a očekává se, že dalším zvyšováním hustoty integrace bude možné zvětšit v blízké době tuto kapacitu elementárního spojovacího pole

na dvojnásobek. Takové spínací obvody pak umožňují ekonomickou realizaci číslicového spojovacího pole i pro větší ústředny. Rozvoji takových obvodů je proto věnována ve světě velká pozornost.

3.2. Mikroelektronické spínací prvky pro číslicové spojování

Číslicové spojování je charakterizováno spojováním a přenosem přes ústřednu číslicových diskretních signálů. Jejich vzájemné propojování, t.j. změna časové polohy se děje přes pomocné paměti, do kterých se informace zapisuje a v jiném okamžiku čte.

Základní blokové schema časového článku realizovaného integrovanými pamětmi je naznačeno na obr. 3.1. Základem časového článku je paměť hovorových informací o velikosti 8 x n, kde n je počet vstupů, které tento článek zpracovává. U základního provedení pro spojování primárních multiplexů PCM 32 kanálových, bude n = 32. V praxi se však jeví výhodnější vytvářet tento článek o větší kapacitě,např. pro 512 vstupů. To znamená, že paměť hovorových informací bude pak mít kapacitu 4 kbitů. Velikost článku T a tím i velikost paměti hovorových informací závisí především na rychlosti, kterou může paměť pracovat. Při stoupajícím počtu vstupních kanálů rostou pochopitelně i požadavky na rychlost paměti. Jako příklad je možno uvést, že u normálního systému PCM 32 kanálového je doba potřebná pro zápis a náaledující čtení v paměti rovna době jednoho kanálu, t.j. cca 4 us. Jestliže bude mít článek T kapacitu 512 vstupních kanálů, což znamená, že tato doba bude muset být 16krát menší, t.j. cca 0,25 us. Pracovní cyklus paměti musí být však dvojnásobný, protože v této době musí proběhnout jak zápis informace, tak i následující čtení.

Z blokového schematu dále vyplývá, že v základním článku T je kromě paměti hovorových informací ještě prakticky stejně velká paměť řídící, v níž jsou zaznamenány vztahy mezi vstupními a vý-

stupními kanály v článku T, t.j. vztah, určující, do které časové polohy má být přesunut každý jednotlivý vstupní kanál. Tato paměť má kapacitu m x n, kde m označuje počet bitů potřebných pro generování adresy, označující řádek v paměti hovorových informací a n je opět počet kanálů, které daný časový článek propojuje. Pracujeme-li s počtem kanálů 512, znamená to, že pro adresování 512 řádků paměti hovorových informací je zapotřebí 9 bitů. Paměť tedy bude mít kapacitu 9 x 512, t.j. 4,6 kbitů.

Kromě těchto dvou hlavních částí obsahuje časový článek T ještě některé pomocné obvody pro úpravu vstupní a výstupní číslicové informace, pro adresování a pod.

V dalším jsou podrobněji popsána některá základní uspořádání mikroelektronických spínacích obvodů, použitých v moderních číslicových spojovacích systémech. Je zde uveden popis časového článku, používaného v moderních systémech kanadské firmy Mitel, kombinovaného časového článku, použitého v jednom z nejmodernějších číslicových systémů ITT 12.

3.3. Spínací obvod firmy Mitel

Tento časový článek o kapacitě 256 x 256 se vyrábí pod označením MT 8980 a je základem moderního číslicového spojovacího systému SX 2000. Časový článek je umístěn na čípu o velikosti 6,9 x 7 mm a je vyroben v technologii ISO CMOS. Z hlediska prostorového nahrazuje podle firemních údajů 3 desky s plošnými spoji, osazenými 100 integrovanými obvody TTL /velikost desek 204 x 382 mm/.

Tento časový článek tím, že spojuje jednotlivé kanály 8 vstupních multiplexů navzájem, tvoří tedy současně i základní prostorový článek pro spojení mezi jednotlivými multiplexy.

Pro další rozšíření spojovacího pole může být použito běžné-

ho řazení článků o struktuře T-T-T, nebo paralelního řazení.

Svou složitostí je tento integrovaný obvod ekvivalentní 36500 tranzistorům. Použitá technologie ISO CMOS umožňuje podstatné snížení potřebného příkonu, zvětšení odolnosti proti rušení a snížení doby šíření, t.j. zlepšení dynamických vlastností tohoto obvodu.

Blokové schema obvodu je na obr. 3.2. Je na něm vidět hlavní části, t.j. paměť hovorových informací, paměť řídící, vstupní a výstupní obvody a obvody rozhraní pro řídící účely. V obvodu jsou použity běžné statické paměti CMOS se 6 tranzistory na jednu paměťovou buňku. Paměti zabírají asi 40 % plochy celého čipu. Celkový příkon pamětí je 150 mW. Paměti pracují rychlostí 5 MHz /při tom maximální pracovní rychlost obvodu je 4 MHz/.

Vstupní signál PCM přichází na převodník serioparalelní a odtud do paměti hovorových informací o velikosti 8 x 256. Zápis je standardně řízen z generátoru cyklických adres, t.j. každá vstupní informace /vzorek/ má předem pevné místo v paměti.

Čtení v paměti a tím i přenos z paměti do výstupních obvodů zajišťuje řídící paměť o kapacitě 11 x 256 bitů /z čehož 8 bitů je určeno pro adresování paměti hovorových informací. Adresy pro čtení v paměti hovorových informací přecházejí přes adresový multiplexor. Zbývající 3 bity adresy jsou určeny pro různé řídící a kontrolní funkce, např. pro současné spojování hovorů a dat a průběžnou kontrolu čipu.

V integrovaném obvodu je také obvod pro řízení rozhraní mezi řídícími částmi systému. Přes toto rozhraní přecházejí do řídící paměti 8-bitové informace, vyznačující budoucí adresy pro čtení v paměti hovorových informací. Zápis do řídící paměti je řízen obvodem pro řízení zápisu /generátor cyklických adres/, čtení řídící paměti zajišťuje hodinový generátor ve spolupráci s adre-

sovým multiplexorem pro čtení.

Tento mikroelektronický obvod je určen převážně pro využití v číslicovém spojovacím systému SX 2000. Jeho funkční možnosti jej však předurčují i pro jiné aplikace, např. pro spojování přenosů dat až do rychlosti 2048 Mbit/s, např. pro meziprocesorovou komunikaci a pod.

Podobný typ spínacích obvodů pro 256 vstupů a 256 výstupů má i firma Thomson a používá jej ve svých spojovacích systémech MT 20/25 a MT 35. Jiný typ spínacího obvodu pro kapacitu 512 vstupů a 256 výstupů má také firma Siemens /v systému EWSD/.

3.4. Integrovaný spínací obvod firmy ITT

Na obr. 3.3 je naznačeno základní blokové schema integrovaného spínače. Je rozdělen na vstupní a výstupní část a obsahuje i společnou sběrnici pro propojení 16 těchto integrovaných spínačů mezi sebou.

Činnost tohoto obvodu je možno stručně popsat takto. Vstupní signál PCM /32 kanálový, 16 bitů ve slově, t.j. přenos rychlostí 4,196 Mbit/s/ přichází na vstupní synchronizační obvod, ve kterém se rozděluje přicházející signál na hovorový, který se přenásí na společnou datovou sběrnici a na řídící informace, přicházející do přijímací paměti RAM, do níž se zapisuje adresa spínacího bodu, adresa kanálu i stav. Tyto informace pak slouží k řízení spojení. Tyto informace se přenášejí na řídící sběrnice a přes ně do ostatních 15 spínacích obvodů dílčího bloku spojovacího pole.

Obvod pro hledání jiného volného spínacího obvodu pak dostává informace jednak ze sběrnice a jednak z obvodu pro porovnávání čísla spínacího obvodu ve vysílací části obvodu. Podle toho

pak obvod pro hledání volného spínače vybere vhodný spínač a jeho adresu uloží do paměti spínacího prvku v přijímací části obvodu. Při příchodu další informace na vstupním kanálu je pak již vždy adresován zvolený spínací obvod, zaznamenaný v této paměti spínacího bodu.

Při takto sestaveném spojení je hovorová informace předávána do vysílací části zvoleného spínacího obvodu do její vysílací paměti. Z ní je pak předávána přes synchronizační obvod na výstup. Řízení vysílací paměti, .tj. vysílání v určenou časovou polohu se děje z obvodu pro řízení vysílání, který dostává informaci o přiděleném kanálu po sběrnici. Volba kanálu se děje dvojím způsobem. Buď se jedná o nalezení libovolného volného kanálu nebo o určení určitého, předem daného kanálu.

V prvém případě se výběr děje v obvodu pro vyhledání 1. volného kanálu ve vysílací části. Vybraný volný kanál /t.j. jeho adresa/ se pak přenese do vstupní paměti kanálů, kde se zaznamená a při dalších přicházejících vzorcích se po sběrnici předává do vysílací části /do obvodu pro řízení vysílání/ toho spínacího obvodu, který byl pro spojení vybrán.

Tímto způsobem je sestaveno spojení a všechny parametry /spínacího bodu i kanálu/ jsou zaznamenány ve vstupní paměti. Sestavení spojení je tedy přímo řízeno podle informací přicházejících v kanálu PCM a souhrou spínacích obvodů přes sběrnicový systém.

Popisovaný integrovaný obvod firmy ITT je v technologii NMOS a je umístěn na čípu velikosti 5,9 x 5,9 mm . Je ekvivalentní asi 11500 tranzistorům. Největší objem zaujímá paměť RAM o kapacitě 1152 bitů /ekvivalent asi 6700 tranzistorů/.

Mikroelektronický obvod je v pouzdře se 64 vývody a maximální ztrátový výkon je 600 mW.

Kromě tohto spínacího prvku je v systému ITT 12 ve vstupních částech spojovacího pole použito ještě dvou podobných typů obvodů jednodušších, označovaných jako vstupní a výstupní brány. Jsou vyrobeny stejnou technologií, mají stejná pouzdra i ztrátový výkon. Složitostí činí asi dvě třetiny základního popsaného spínacího obvodu.

Firma ITT předpokládá, že s postupem vývoje obvodů VLSI bude možné realizovat na jednom čípu postupně 2, 4 a 8 těchto spínacích obvodů, případně v budoucnu i 16 /odpovídající asi 180000 tranzistorům/.

4. Integrované obvody v zařízeních signalizace

4.1. Úvod

Ve většině dosud probíraných aplikací mikroelektroniky v elektronických telefonních ústřednách se jednalo o použití speciálních obvodů, jejichž možnost využití v jiných oborech byla celkem malá. Kromě toho však v elektronických ústřednách nacházejí použití i některé speciální LSI obvody, které mají širší uplatnění.

Jsou to především obvody pro generování a příjem tónových signálů pro účely signalizace /např. pro multifrekvenční signalizaci R 2, pro tlačítkovou volbu a pod. /. S integrovanými obvody lze realizovat nejen vlastní vyhodnocení tónových signálů /přijímače/, ale i potřebné vstupní filtry. Používá se k tomu především principu tzv. fázového závěsu. S využitím takových obvodů je možné tedy vytvořit přijímače kmitočtové tlačítkové volby i přijímače multifrekvenční meziústřednové signalizace.

V řadě podobných nových aplikací se začíná prosazovat také princip signálního procesoru pro číslicové zpracování analogových

signálů. Výhodou tohoto řešení je především:

- při číslicovém zpracování se zmenšuje závislost přenosových parametrů na tolerancích součástek
- zapojení jsou v širokých mezích necitlivá na vnější vlivy
- stejným zařízením lze programově realizovat různé přenosové vlastnosti

4.2 Signální procesor

Signální procesor /obr. 4.1/ je v dnešním pojetí speciální mikroprocesor, vhodně přizpůsobený pro číslicové zpracování signálů /např. Intel 2920/.

Tento signální procesor obsahuje na čípu kromě základních mikropočítačových obvodů ještě potřebné obvody pro zpracování analogových signálů, t.j. vzorkovací vstupní a výstupní obvody a analogově číslicový převodník.a několik analogových vstupů a výstupů, které je možno použít jak pro analogový, tak pro číslicový signál.

Pro zpracování analogových signálů s pomocí signálního procesoru je nutné analogové vstupy a výstupy signálního procesoru ještě doplnit na vstupu omezovačem kmitočtového pásma vstupního signálu, který zabraňuje působení přídavných kmitočtových složek a na výstupu obvody pro rekonstrukci analogového signálu za stupňového průběhu /dolní propust/.

Vlastní činnost signálního procesoru je pak zjednodušeně možno rozvést do těchto funkcí:

- vstupní vzorkování
- převod velikosti analogového vzorku na číslicovou hodnotu
- číslicové zpracování
- převod číslicového signálu na analogový
- výstupní vzorkování s pamětí

Řešení signálních obvodů a především filtrů s pomocí signálního procesoru je jednou z možností, která může v budoucnu ovlivnit celou oblast číslicového zpracování analogových signálů.

Kromě toho existují integrovaná provedení filtrů na jiných principech. Jako příklad je uvedeno provedení filtru pro přijímač tlačítkové volby firmy Mitel, MT 8865, jehož princip je znázorněn na obr. 4.2. Oddělení obou skupin kmitočtů je docíleno tím, že přicházející dvoutónový signál je přiváděn na vstupy dvou pásmových propustí se spínanými kondenzátory šestého řádu. Šířka pásma obou propustí odpovídá pásmu, zahrnujícímu vždy příslušnou skupinu kmitočtů. Útlumové charakteristiky každého filtru obsahují pokles při 440 Hz, pro potlační vlivu oznamovacího tónu, který se na vedení může při volbě objevit. Výstup každé pásmové propusti je přiveden na sekci se spínanými kondenzátory prvního řádu, která pracuje jako interpolátor, vyhlazující signál před jeho omezením. Omezovací funkci přejímá komparátor s velkým ziskem "s hysterezí" pro zamezení detekce nežádoucích signálů o nízké úrovni a hluku. Výstup komparátoru je zesílen pro vybuzení výstupu FL /pro skupinu nízkých kmitočtů/ a FH /pro skupinu vyšších kmitočtů/.

Vstup tohoto integrovaného obvodu je jednovodičový, dovolující připojení jak na dekodér signálu PCM, nebo radiového přijímače, tak i přes diferenciální zapojení na telefonní vedení.

Na výstupu tohoto obvodu může být zapojen přímo integrovaný přijímač /např. MT 8860/.

Integrovaný filtr MT 8865 je vyroben technologií ISO CMOS o velké hustotě. Hodiny pro řízení filtru jsou odvozeny z oscilátoru umístěného na čípu, který pro synchronizaci vyžaduje přídavný krystal 3,58 MHz. Obvod pracuje v rozsahu napájecího napětí od + 5 do + 12 V. Obsahuje i zapojení pro snížení příkonu /power

down/ v době klidu. Je umístěn v pouzdře o 16 vývodech.

Tento filtr je použitelný nejen v obvodech přijímačů tlačítkové volby, ale i v mobilních radiových stanicích, různých měničích signálů a pod.

Postupem dalšího vývoje se objevily na trhu i kompletní přijímače tónové volby v jednom integrovaném obvodě.

4.3 Fázový závěr

Pro příjem tónové signlizace pro různé typy signalizace mfc /např. kódu R 2, R 1 a pod./, se dosud používalo převážně standardních filtrů LC pro oddělení jednotlivých kmitočtů /signalizace se děje 6 kmitočty v dopředném směru a 6 kmitočty ve zpětném směru/ s následující detekcí signálů. Nejsložitější částí přijímače byly právě tyto filtry, protože pracují v nízkofrekvenčním pásmu a jsou tedy poměrně objemné a složité ve výrobě i při nastavování. Později se přešlo na hybridní provedení aktivních filtrů, které umožnily zmenšit objem přijímačů tónové signalizace.

V dnešní době se ukazují možnosti využití mikroelektronických obvodů pro konstrukci těchto přijímačů mfc volby. Monolitické tónové dekodéry pracují většinou na principu tzv. fázového závěsu, který umožňuje selektivní příjem kmitočtu bez indukčností.

Princip fázového závěsu je znázorněn na obr. 4.3. Je to zpětnovazební systém skládající se z fázového detektoru /komparátoru/, dolní propusti, zesilovače a napěťově řízeného oscilátoru ve zpětnovazební smyčce. Napěťově řízený oscilátor má tu vlastnost, že jeho výstupní kmitočet je úměrný přiloženému vstupnímu napětí, což je v tomto zapojení odfiltrované a zesílené vstupní /tzv. chybové/napětí fázového detektoru.

Fázový závěs pracuje v principu takto. Jestliže není na vstupu žádný signál, je chybové napětí V_a na výstupu nulové. Napěťově řízený oscilátor pracuje na kmitočtu f_o, který je dán elektrickými parametry oscilátoru. Tentokmitočet se nazývá základní. Objeví-li se na vstupu signál, porovnává fázový detektor fázi i kmitočet přicházejícího kmitočtu a základního, vystupujícího z oscilátoru a generuje chybové napětí V_e, které je úměrné fázovému a kmitočtovému rozdílu mezi oběma signály. Toto chybové napětí se potom po odfiltrování vyšších kmitočtů a zesílení přivádín a vstup oscilátoru.

Jestliže je přicházející signál kmitočtově dosti rozdílný od základního kmitočtu oscilátoru, mají součtové a rozdílové složky vyšší kmitočet a jsou dolní propustí odfiltrovány, takže výstupní chybové napětí zůstává nulové, nebo-li obvod nereaguje na vstupní signál.

Má-li však vstupní signál kmitočet dosti blízký základnímu kmitočtu f_o a předpokládáme-li, že součtové složky s vyšším kmitočtem se po detekci odfiltrují, zbývají pouze rozdílové složky, které projdou filtrem bez tlumení. Chybové napětí V_a na výstupu zesilovače má pak sinusový průběh s kmitočtem rovným rozdílu základního i vstupního signálu. Toto chybové napětí pak ovlivňuje napěťově řízený oscilátor, jehož kmitočet se pak mění úměrně s velikostí chybového napětí.

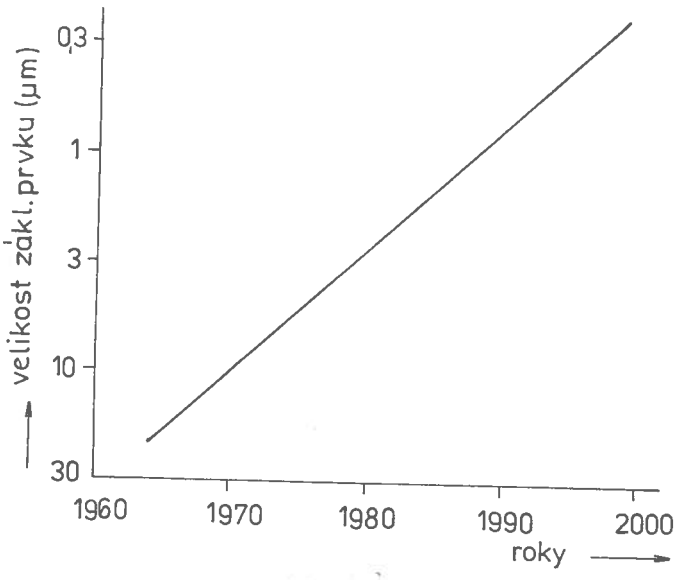
Kolísání kmitočtu oscilátoru způsobuje, že se jeho kmitočet přibližuje a vzdaluje vstupnímu kmitočtu. Stejnosměrná složka chybového napětí způsobuje trvale přibližování základního kmitočtu vstupnímu, takže jejich rozdíl rychle klesá. Jakmile se celý systém ustálí, je rozdíl kmitočtů nulový a zůstává na výstupu filtru pouze stejnosměrná složka, která nastavuje kmitočet oscilátoru na kmitočet vstupního signálu.

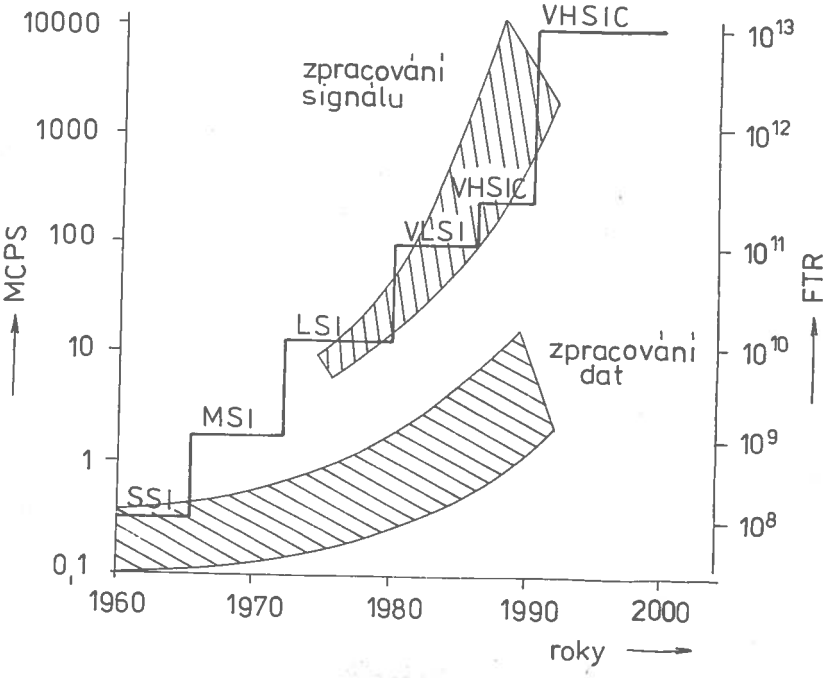
Na obr. 4.4 je naznačen princip typického zapojení celého fázového detektoru. První část tvoří vlastní obvod fázového závěsu a na něj navazuje přídavný kvadraturní detektor, který zajišťuje detekci stavu fázového závěsu. Tento kvadraturní detektor spolu s navazující dolní propustí umožňuje rozlišit stav systému bez nebo se vstupním signálem. Na ně pak navazuje potřebné zesílení pro buzení výstupních obvodů.

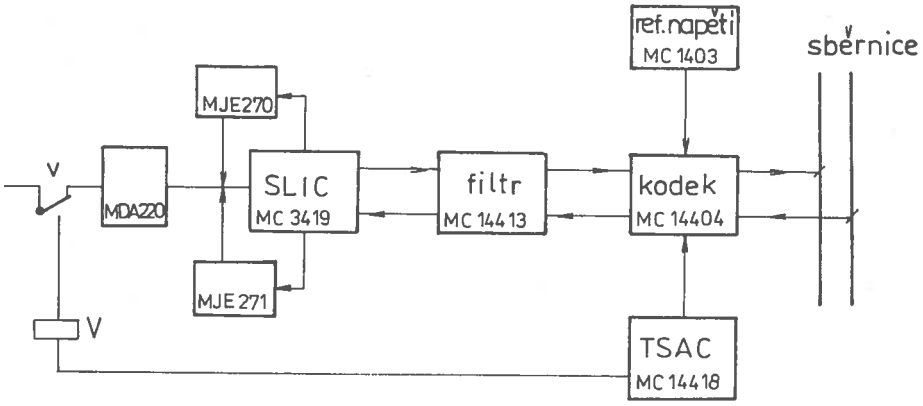
V moderních spojovacích systémech, především pro integrované sítě, se předpokládá použití soustředěné signalizace /signální systém č. 7 CCITT/. Pro hardwarovou realizaci koncových zařízení této signalizace pracující s přenosovou rychlostí 64 kbit/s bude rovněž možné využívat speciálních LSI obvodů.

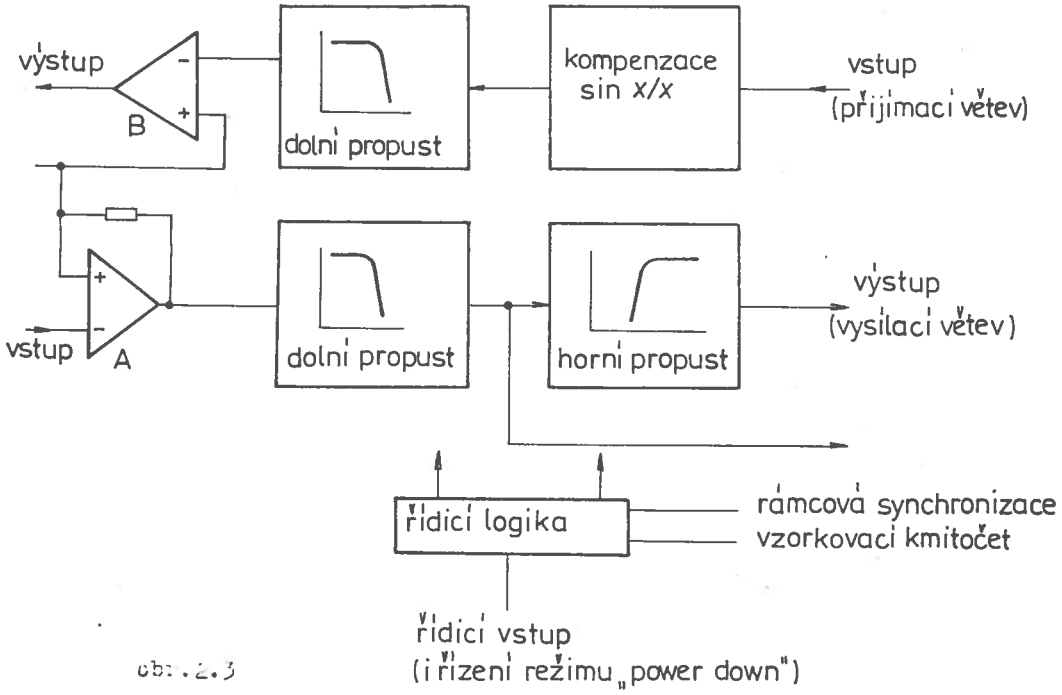
Tabulka 1 Přehled možností využití speciálních mikroelektronických obvodů v elektronických ústřednách a koncových zařízeních

		počet IO
1. Spojovací zařízení		
1.1 Účastnické rozhraní	- obvod SLIC	1 ks/příp.
	- kodek	1 ks/pp
	- filtr	1 ks/pp
	- řadič	1 ks/pp
1.2 Spojovací pole prostorové	- spínací matice 8x4, 4x4	1 ks/pp
1.3 Spojovací pole číslicové	- integrované paměti	0,1 ks/pp
	- speciální časové spínače 256x256	0,01 ks/pp
1.4 Přijímač tónové signalizace	- pro tlačítkovou volbu	0,01 ks/pp
	- pro mfc signalizaci R2	0,01 ks/pp
1.5 Společná signalizace	- modemy	0,001 ks/pp
2. Telefonní přístroj	- hovorový obvod	1 ks/pp
	- tlačítková volba dekadická	1 ks/pp
	- tlačítková volba frekvenční	1 ks/pp
	- přijímač vyzvánění	1 ks/pp

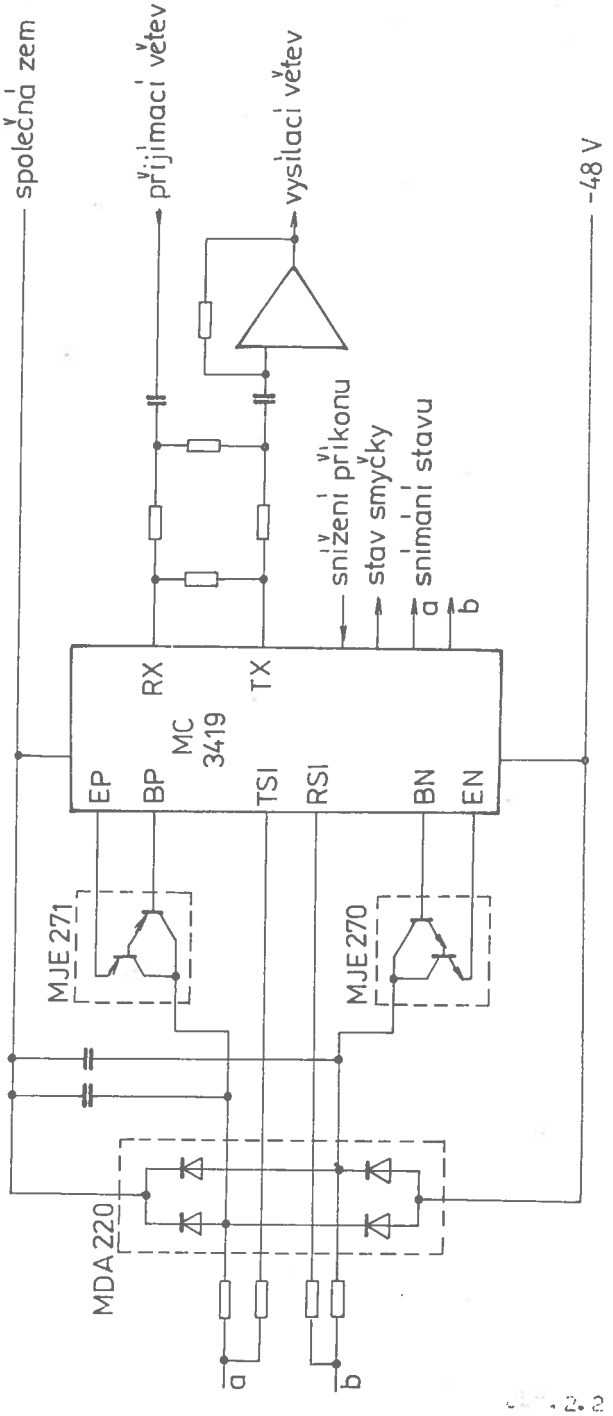

velikost zákl. prvku (μm)
0,3
1
3
10
30
1960
1970
1980
1990
2000
roky


MCPS
10000
1000
100
10
1
0,1
FTR
10^{13}
10^{12}
10^{11}
10^{10}
10^{9}
10^{8}
zpracování signálu
zpracování dat
VHSIC
VLSI
LSI
MSI
SSI
1960
1970
1980
1990
2000
roky

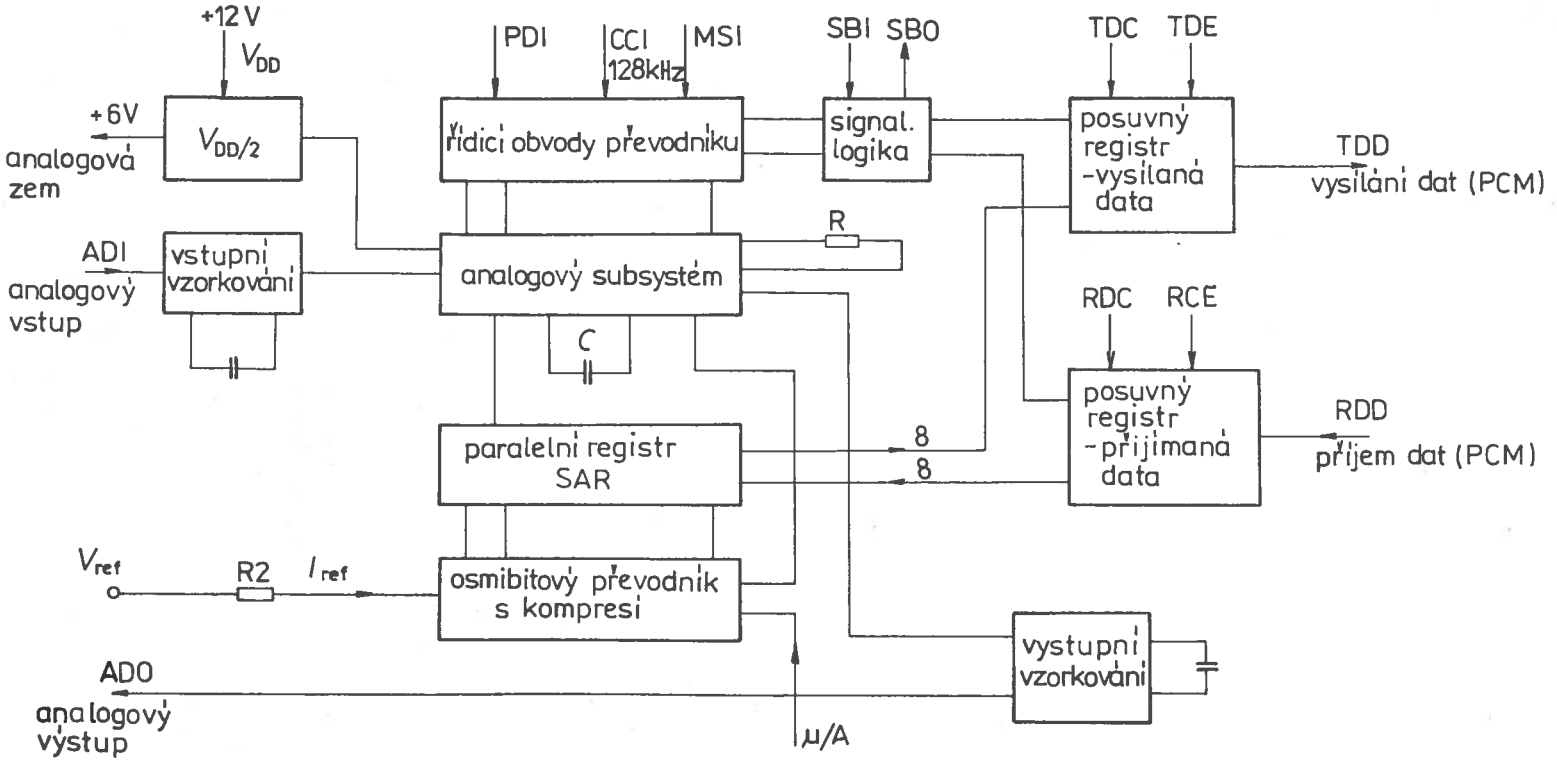

ref.napětí
MC 1403
sběrnice
MJE270
V
MDA220
SLIC
MC 3419
filtr
MC 14413
kodek
MC 14404
MJE 271
V
TSAC
MC 14418

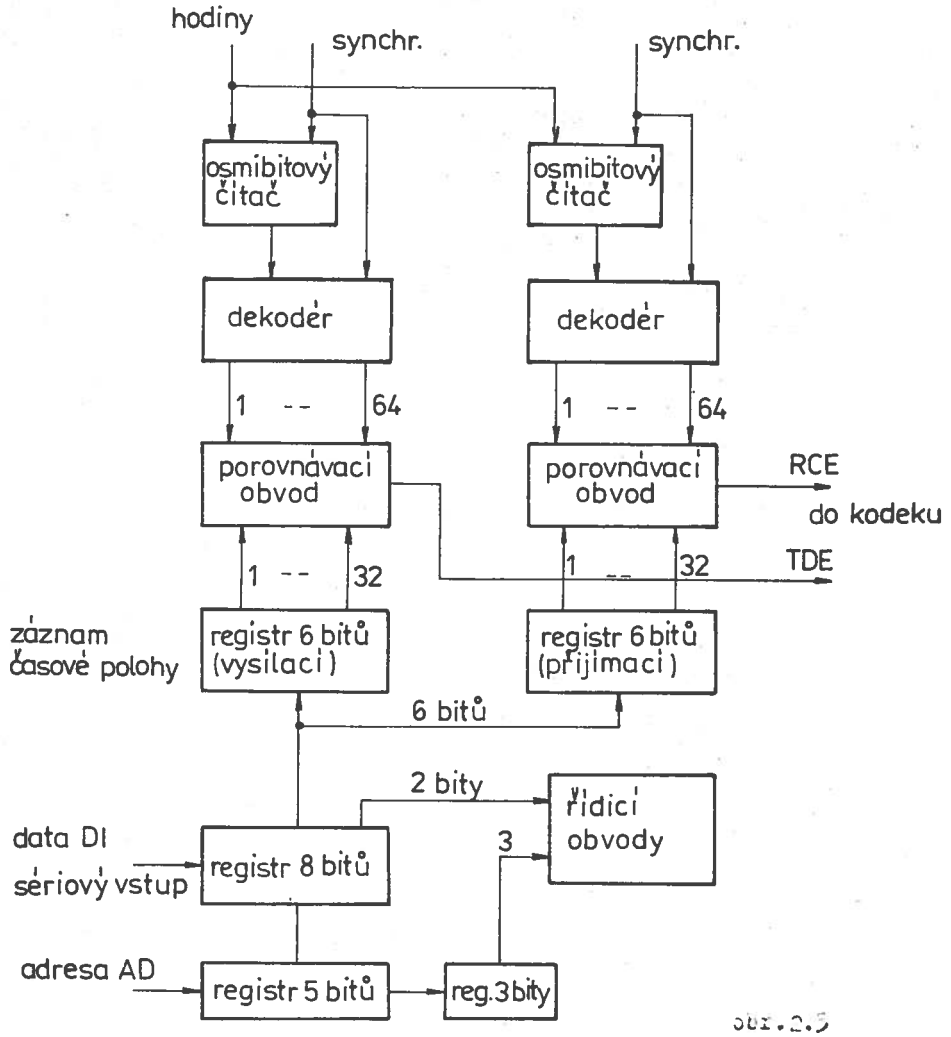

výstup
B
dolní propust
kompenzace
sin x/x
vstup
(přijímací větev)
vstup
A
dolní propust
horní propust
výstup
(vysílací větev)
řídicí logika
rámcová synchronizace
vzorkovací kmitočet
řídicí vstup
(i řízení režimu „power down")

obr. 2.3


společná zem
přijímací větev
vysílací větev
-48 V
snížení příkonu
stav smyčky
snímání stavu
a
b
RX
TX
MC
3419
EP
BP
TSI
RSI
BN
EN
MJE 271
MJE 270
MDA 220
a
b

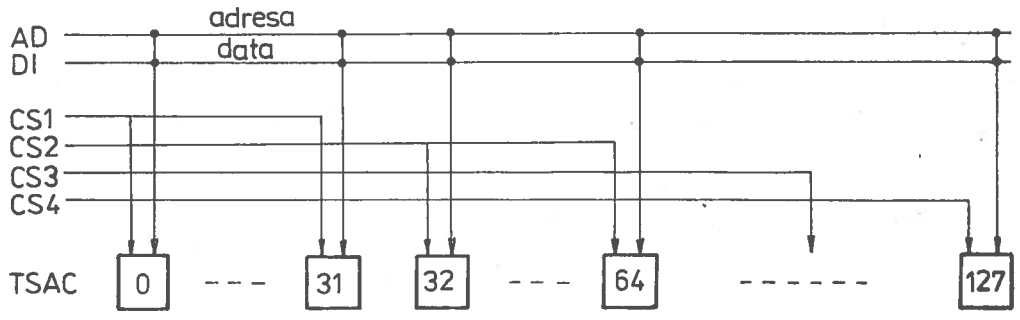
2.2

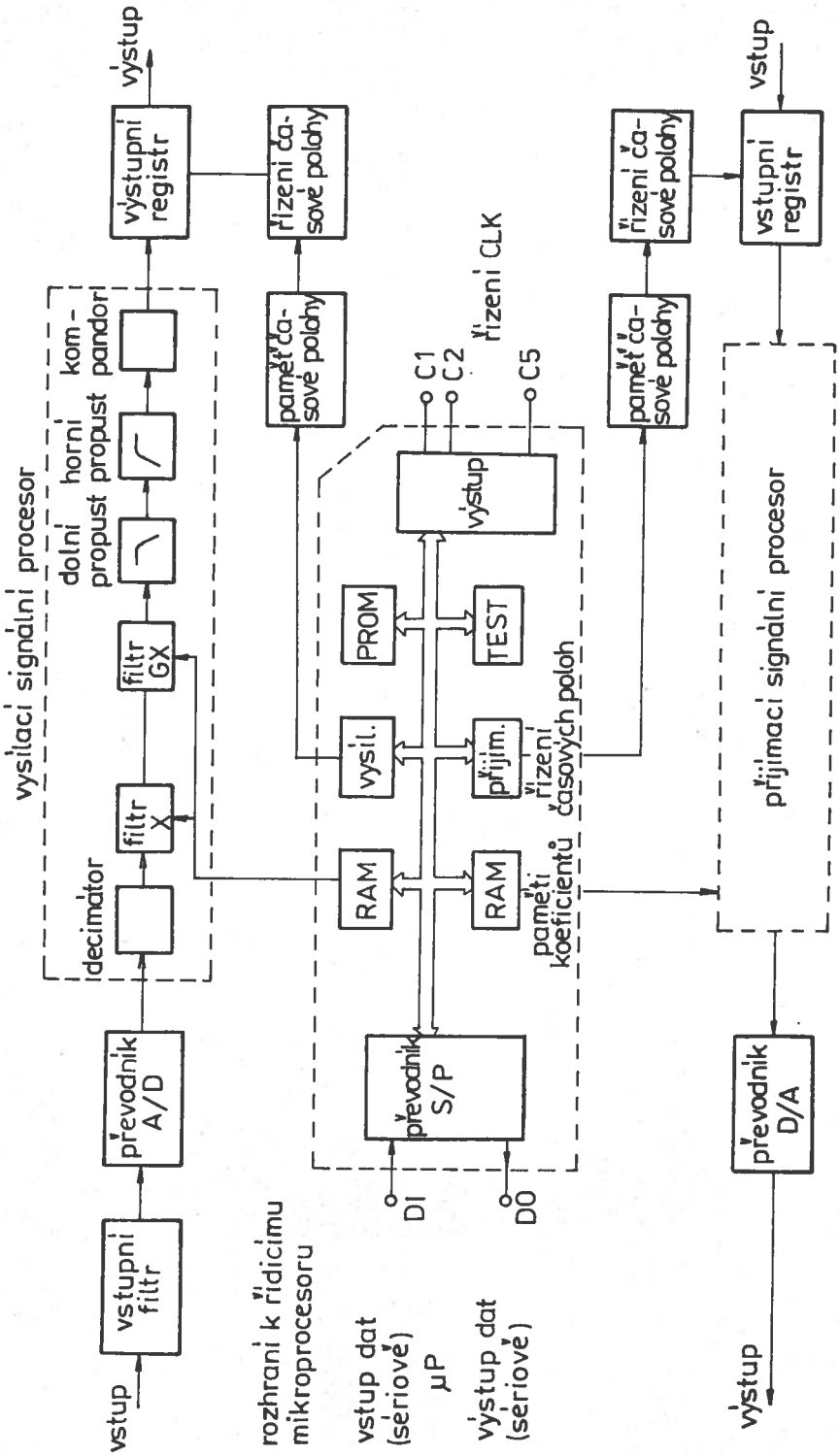

+12 V
V_{DD}
+6V
$V_{DD/2}$
analogová zem
ADI
analogový vstup
vstupní vzorkování
PDI
CCI 128kHz
MSI
řídicí obvody převodníku
SBI
SBO
signal. logika
TDC
TDE
posuvný registr -vysílaná data
TDD
vysílání dat (PCM)
R
analogový subsystém
C
RDC
RCE
posuvný registr -přijímaná data
RDD
příjem dat (PCM)
paralelní registr SAR
8
8
V_{ref}
R2
I_{ref}
osmibitový převodník s kompresí
vystupní vzorkování
ADO
analogový výstup
μ/A


hodiny
synchr.
synchr.
osmibitový čítač
osmibitový čítač
dekodér
dekodér
1 -- 64
1 -- 64
porovnávací obvod
porovnávací obvod
RCE
do kodeku
1 -- 32
1 -- 32
TDE
záznam časové polohy
registr 6 bitů (vysílací)
registr 6 bitů (přijímací)
6 bitů
2 bity
řídicí obvody
3
data DI
sériový vstup
registr 8 bitů
adresa AD
registr 5 bitů
reg. 3 bity

obr. 2.5

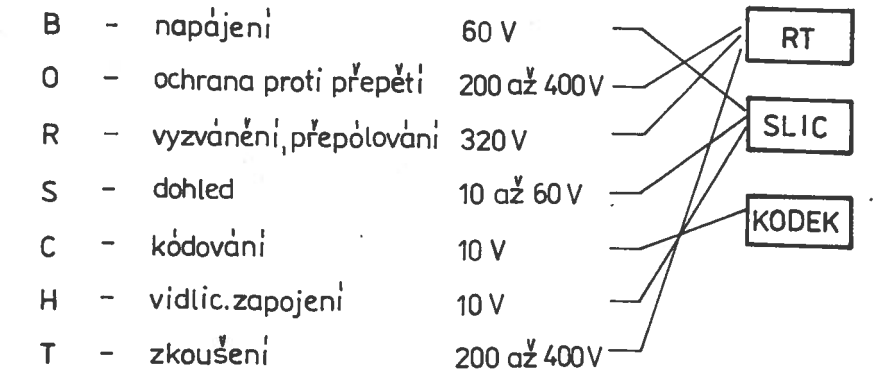
obr. 2.6

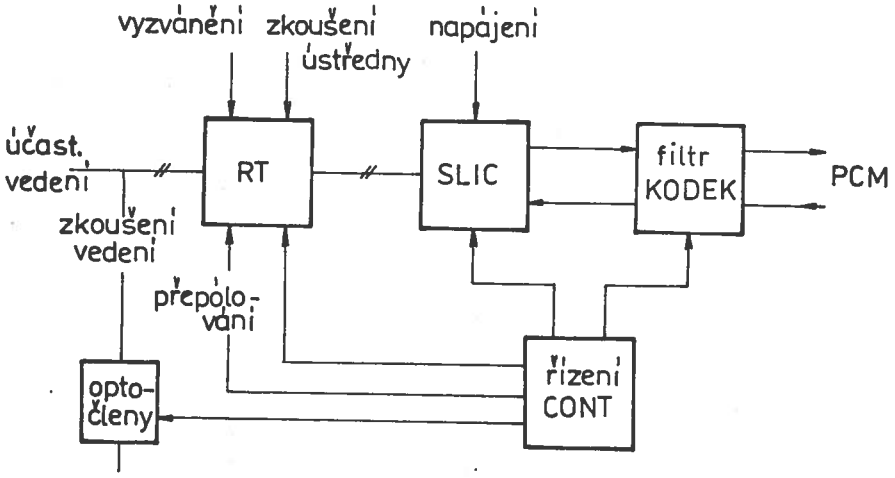

adresa
data
AD
DI
CS1
CS2
CS3
CS4
TSAC
0
31
32
64
127


vysílací signální procesor
decimátor
filtr X
filtr GX
dolní propust
horní propust
kompandor
převodník A/D
vstupní filtr
vstup
výstupní registr
výstup
řízení časové polohy
paměť časové polohy
rozhraní k řídícímu mikroprocesoru
vstup dat (sériově)
µP
výstup dat (sériově)
DI
DO
převodník S/P
RAM
RAM
paměti koeficientů
vysíl.
přijím.
řízení časových poloh
PROM
TEST
výstup
C1
C2
C5
řízení CLK
přijímací signální procesor
vstup
vstupní registr
řízení časové polohy
paměť časové polohy
převodník D/A
výstup

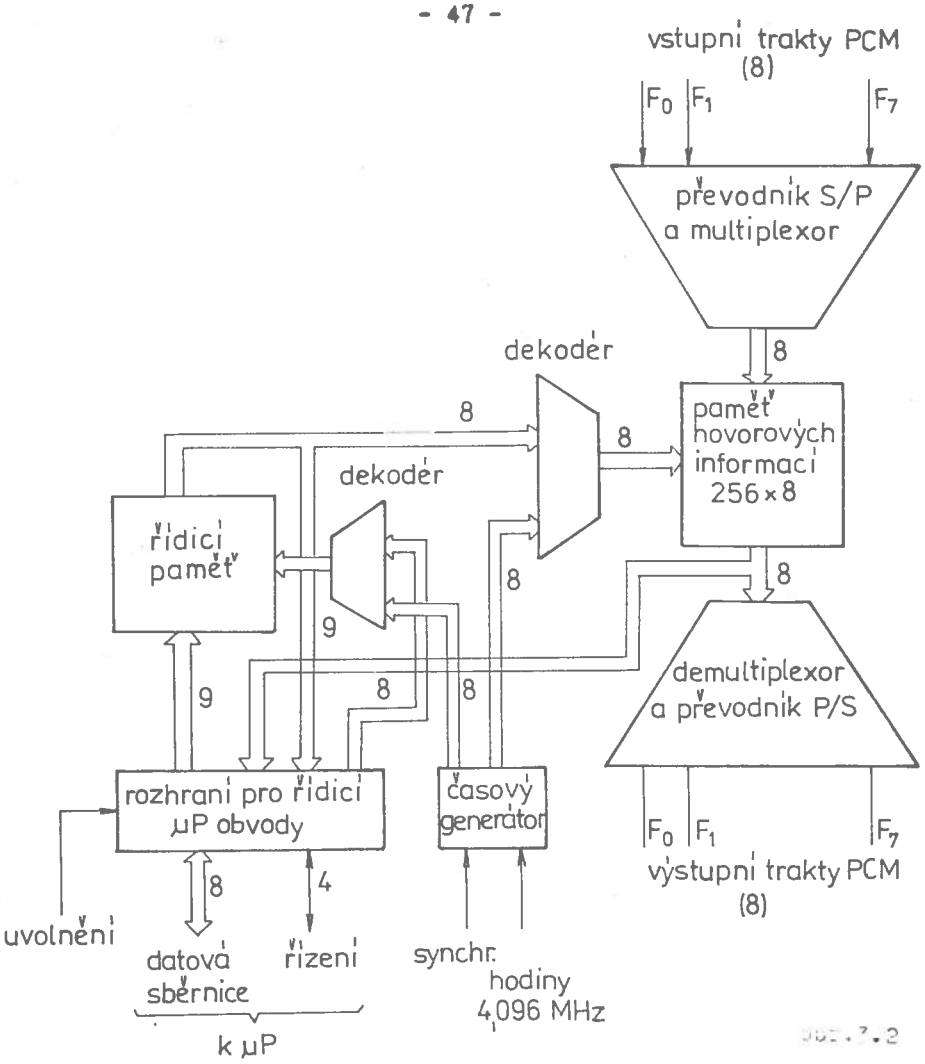
obr. 2.7

		Funkce	Napětí
B	-	napájení	60 V
O	-	ochrana proti přepětí	200 až 400 V
R	-	vyzvánění, přepólování	320 V
S	-	dohled	10 až 60 V
C	-	kódování	10 V
H	-	vidlic. zapojení	10 V
T	-	zkoušení	200 až 400 V

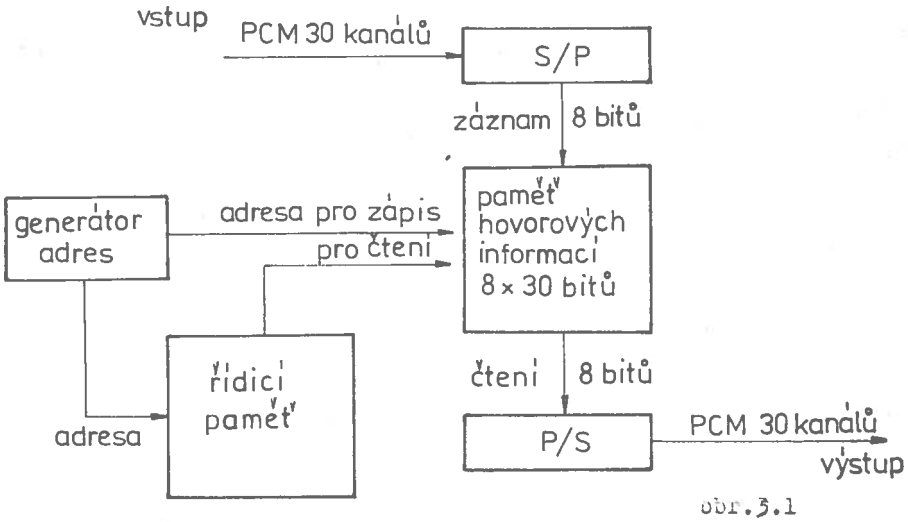




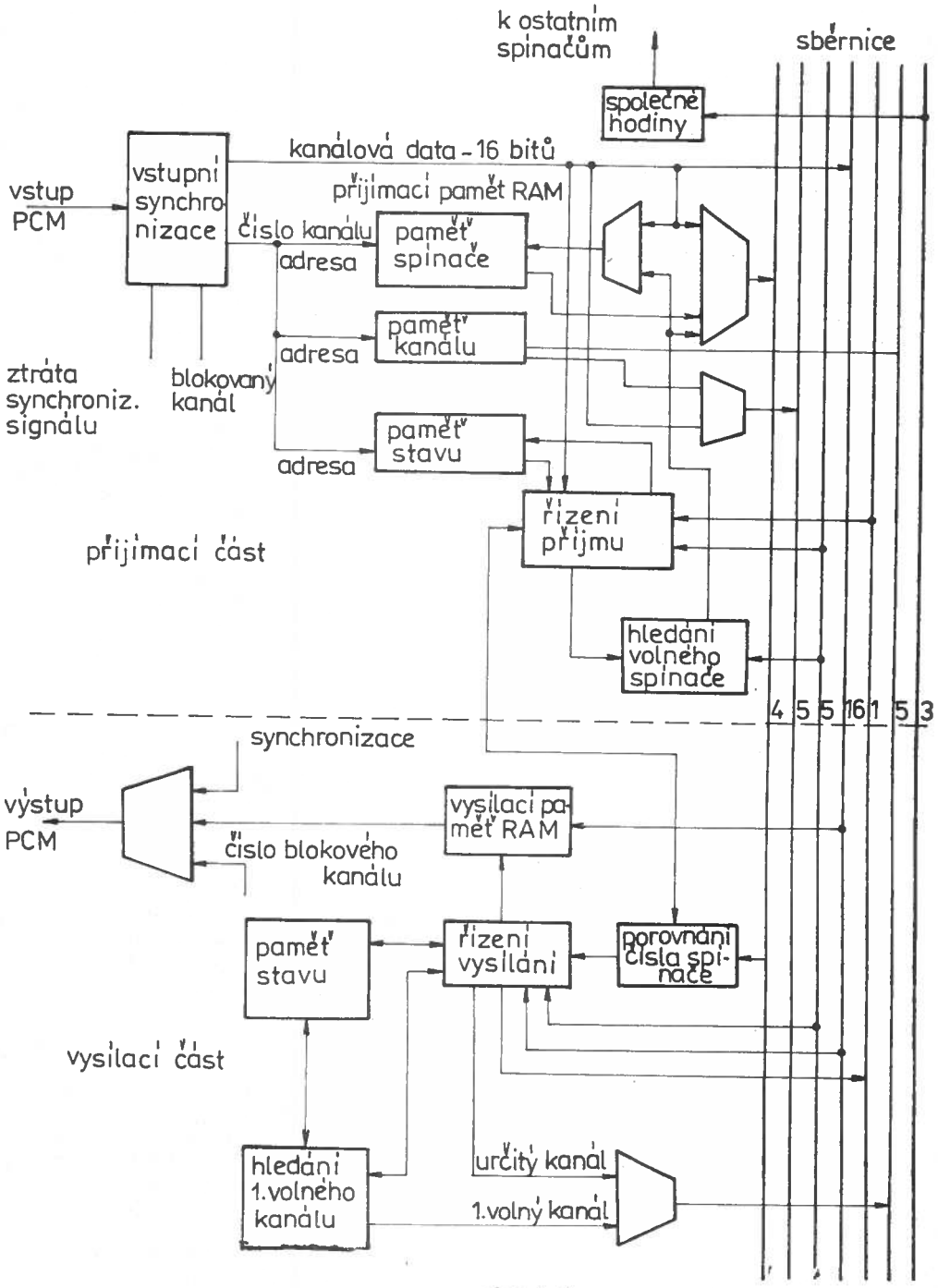
obr. 2.8


vstupní trakty PCM
(8)
F0
F1
F7
převodník S/P
a multiplexor
8
dekodér
8
8
paměť
hovorových
informací
256×8
dekodér
řídicí
paměť
8
9
9
8
8
8
demultiplexor
a převodník P/S
rozhraní pro řídicí
μP obvody
časový
generátor
F0
F1
F7
výstupní trakty PCM
(8)
8
4
uvolnění
datová
sběrnice
řízení
synchr.
hodiny
4,096 MHz
k μP

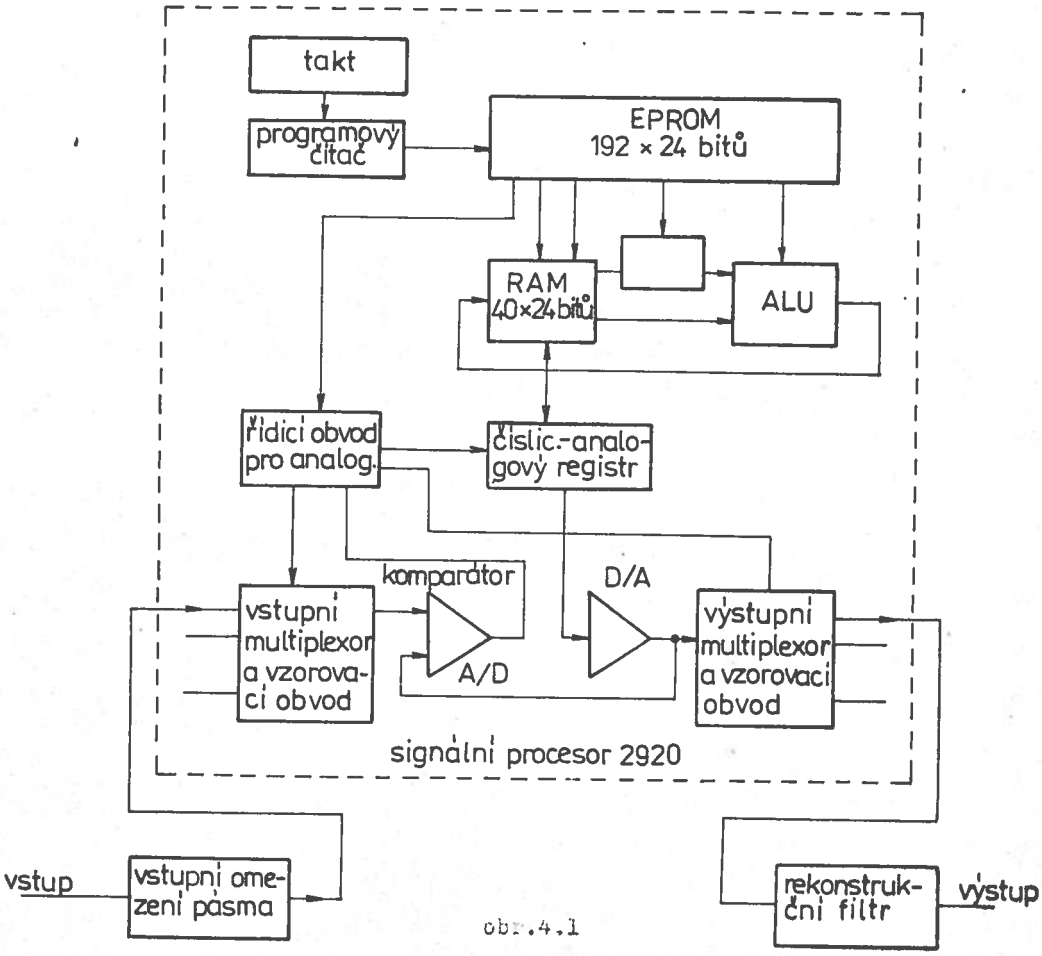
obr.3.2


vstup
PCM 30 kanálů
S/P
záznam
8 bitů
generátor
adres
adresa pro zápis
pro čtení
paměť
hovorových
informací
8 × 30 bitů
řídicí
paměť
čtení
8 bitů
adresa
P/S
PCM 30 kanálů
výstup

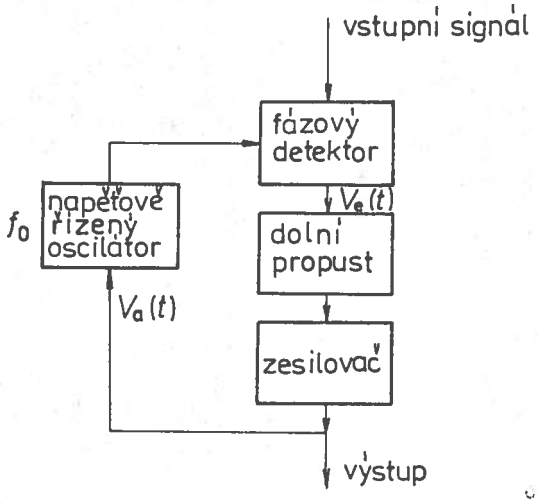
obr.3.1


k ostatním spínačům
sběrnice
společné hodiny
vstupní synchronizace
vstup PCM
kanálová data - 16 bitů
přijímací paměť RAM
číslo kanálu
adresa
paměť spínače
paměť kanálu
adresa
ztráta synchroniz. signálu
blokovaný kanál
paměť stavu
adresa
řízení příjmu
přijímací část
hledání volného spínače
4 5 5 16 1 5 3
synchronizace
výstup PCM
vysílací paměť RAM
číslo blokového kanálu
paměť stavu
řízení vysílání
porovnání čísla spínače
vysílací část
hledání 1. volného kanálu
určitý kanál
1. volný kanál

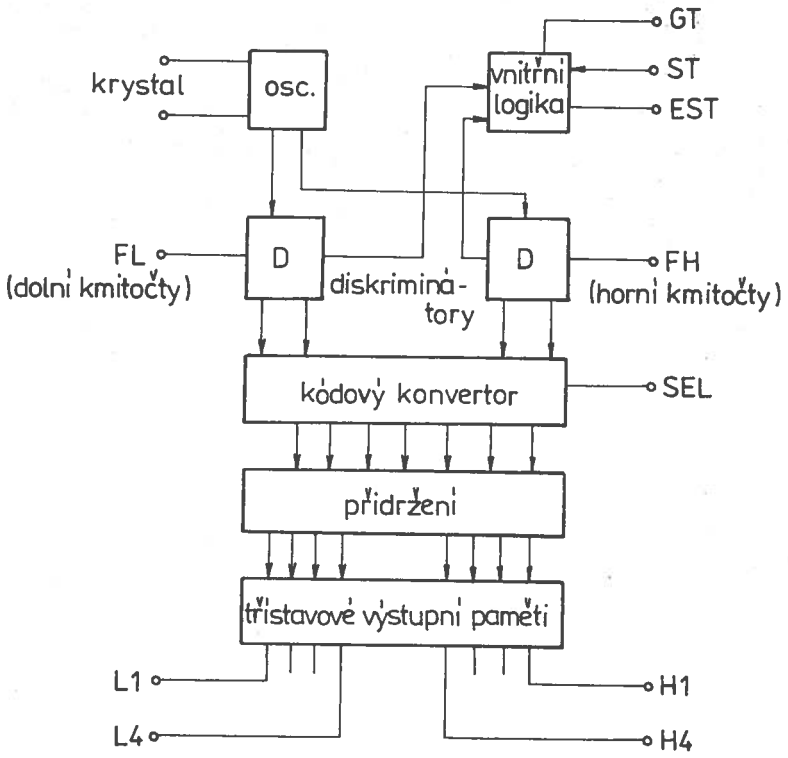
obr. 3.3


takt
programový čítač
EPROM 192 × 24 bitů
RAM 40×24bitů
ALU
řídicí obvod pro analog.
číslic.-analogový registr
komparátor
A/D
D/A
vstupní multiplexor a vzorovací obvod
výstupní multiplexor a vzorovací obvod
signální procesor 2920
vstup
vstupní omezení pásma
rekonstrukční filtr
výstup

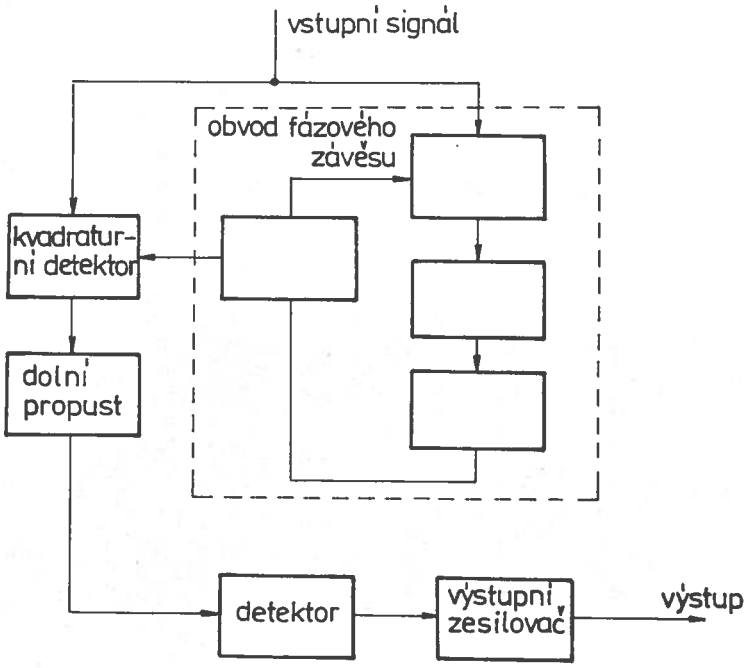
obr.4.1


vstupní signál
fázový detektor
$V_e(t)$
napěťově řízený oscilátor
f_0
dolní propust
$V_a(t)$
zesilovač
výstup

obr.4.3


GT
krystal
osc.
vnitřní logika
ST
EST
FL
(dolní kmitočty)
D
diskriminá-tory
D
FH
(horní kmitočty)
kódový konvertor
SEL
přidržení
tristavové výstupní paměti
L1
L4
H1
H4

obr. 4.3


vstupní signál
obvod fázového závěsu
kvadraturní detektor
dolní propust
detektor
výstupní zesilovač
výstup

obr. 4.4

Ing. Jiří Chod, CSc.

Mikroprocesorová technika II.

PRAHA 1985

Úvod

Následující kapitoly přímo navazují na učební text PGS Mikroprocesorová technika I. Návaznost očíslování jednotlivých kapitol je zachována, stejně tak i číslování obrázků.

2.8.5 Programovatelné obvody V/V

Programovatelný obvod 8255

8255 je programovatelný víceúčelový obvod pro připojení vstupů/výstupů k mikroprocesoru 8080A. Má 24 vstupů/výstupů, které mohou být naprogramované odděleně ve dvou skupinách po dvanácti a mohou pracovat ve třech režimech.

Při prvním druhu provozu (režim 0) může být každá skupina 12 vstupů/výstupů naprogramována ve skupinách po 4 jako vstup nebo výstup. Při druhém druhu provozu (režim 1) může být naprogramováno osm vstupů/výstupů v každé skupině jako vstup nebo výstup. Ze zbývajících 4 vývodů se použijí tři pro výměnu potvrzení přerušení a pro signály k řízení přerušení.

Třetí druh provozu (režim 2) můžeme označit jako obousměrný provoz sběrnice, při kterém se používá pro jednu obousměrnou sběrnici osm vývodů. Pět dalších vývodů, z nichž jeden patří už do druhé skupiny, se v tomto případě používá pro výměnu potvrzení přerušení.

Kromě toho je možné přímé nastavení a nulování jednotlivých bitů. Funkční vlastnosti obvodu 8255 jsou dány softwarem, takže pro připojení periferních zařízení nebo obvodů nejsou obvykle třeba žádné další logické obvody.

Schéma zapojení obvodu 8255 ukazuje obr. 2.9, pohled na rozmístění vývodů obr. 2.10. Označení vývodů udává tab. 2.4, a shrnutí základních druhů provozu tab. 2.5.

Označení	Funkce
$D_{\varnothing}$ - D_7	datová sběrnice (obousměrná
RESET	nulování (vstup)
$\overline{CS}$	aktivace obvodu
$\overline{RD}$	čtení
$\overline{WR}$	zápis
A_0, A_1	adresa brány a CWR
PA_0 - PA_7	brána A (bit 0 až 7)
PB_0 - PB_7	brána B (bit 0 až 7)
PC_0 - PC_7	brána C (bit 0 až 7)
V_{CC}	napájecí napětí + 5 V
GND	zem (0 V)

Tab. 2.4 Označení vývodů obvodu 8255

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Č T E N Í (vstupní operace)
0	0	0	1	0	brána A → Datová sběrnice
0	1	0	1	0	brána B → Datová sběrnice
1	0	0	1	0	brána C → Datová sběrnice
					Z Á P I S (výstupní operace)
0	0	1	0	0	Datová sběrnice → brána A
0	1	1	0	0	Datová sběrnice → brána B
1	0	1	0	0	Datová sběrnice → brána C
1	1	1	0	0	Datová sběrnice → CWR

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	P A S I V N Í F U N K C E
X	X	X	X	1	Datová sběrnice → 3. stav
1	1	0	1	0	bez funkce
X	X	1	1	0	Datová sběrnice → 3. stav

Tab. 2.5 Základní druhy provozu obvodu 8255

Všech 24 vstupů/výstupů obvodu 8255 je sdruženo do tří osmibitových bran (A, B, C). Podle způsobu naprogramování mohou splňovat různé funkce. Mají různé možnosti, které rozšiřují oblast použití a flexibilitu obvodů 8255. Funkční konfigurace každé brány se nastavuje programem tak, že mikroprocesor vyšle do 8255 vhodné řídicí slovo CW. Existují dva formáty CW, jeden pro definici druhu provozu (režimu, modu) a druhý pro řízení bitů brány C. Do registru CWR lze pouze zapisovat, nelze z něho číst. Brány A, B, C lze nastavit dle potřeby při respektování toho, že jednotlivé brány mohou pracovat jako:

Brána A: jeden osmibitový datový výstupní latch/buffer a jeden osmibitový datový vstupní latch.

Brána B: jeden osmibitový vstupní/výstupní datový latch/buffer a jeden osmibitový vstupní latch.

Brána C: jeden osmibitový datový výstupní latch/buffer a jeden osmibitový vstupní buffer.

Brána C může být rozdělena na dvě čtyřbitové brány. Každá obsahuje jeden čtyřbitový latch a může být použita pro výstupy ve spojení s branami A a B. Jak již bylo řečeno výše, naprogramováním systému jsou pevně dané tři základní druhy provozu (režimy):

Druh provozu 0 - jednoduché vstupy/výstupy,
druh provozu 1 - strobované vstupy/výstupy,
druh provozu 2 - obousměrná sběrnice.

Je-li na vstupu RESET úroveň log "1" (HIGH), nastaví se všechny brány do stavu pro vstup dat (tzn., že 24 vodičů je ve stavu s velkou impedancí). Po skončení signálu RESET zůstává 8255 v tomto výchozím stavu, aniž by bylo třeba nějakého dalšího nastavení. Během systémového programu (OUT) může být zvolen kterýkoliv z druhů provozu. To umožňuje, že jedním obvodem 8255 mohou být ovládána různá periferní zařízení s jednoduchým programem.

Druhy provozu bran A a B mohou být definované nezávisle na sobě, zatímco brána C je rozdělena podle požadavků bran A a B na dvě části. Změní-li se druh provozu, jsou všechny výstupní registry včetně stavových klopných obvodů resetovány. Druhy provozu lze kombinovat. Např. skupina B může být naprogramována na provoz Ø, zatímco skupina A může být naprogramována na provoz 1.

Definice řídicího slova CW pro volbu druhu provozu ukazuje obr. 2.11, typické propojení pro všechny druhy provozu obr. 2.12.

Nastavení jednotlivých bitů

Každý z 8bitů brány C může být instrukcí (OUT) nastaven na 1 (setován) nebo na Ø (resetován). Nastavení/nulování se uskuteční (viz obr. 2.13) zápisem do CWR (A_1 $A_Ø$ = 11) řídicího slova CW, které má D_7 = Ø a D_3 D_2 D_1 určují řízený bit.

Přerušovací funkce

Je-li 8255 naprogramován na druh provozu 1 nebo 2, máme k dispozici řídicí signály, které můžeme použít jako signály pro přerušení pro 8080A. Signály pro požadavek přerušení z brány C mohou být setováním nebo resetováním příslušného klopného obvodu INTE akceptovány nebo nikoli (použitím funkce "Bit set/reset" brány C).

Definice pro klopný obvod INTE:

Bit-SET - INTE je nastaven - přerušení umožněno.

Bit-RESET - INTE je nulován - přerušení není umožněno.

příklad

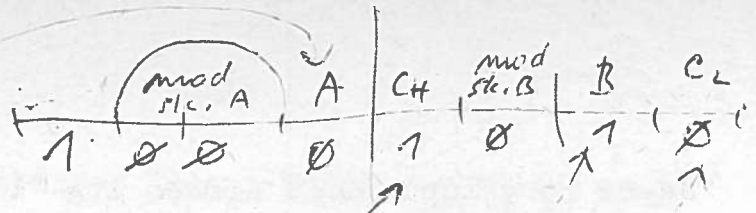
mód Ø

A = OUT

C_H = IN

B = IN

C_L = OUT



#8A = 138 dek.

---

potřeby:

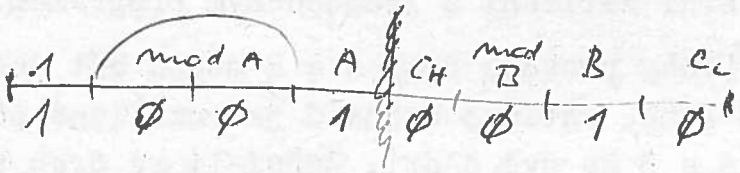
mód Ø

A = IN

C_H = OUT

B = IN

C_L = OUT



#92 = 146 dek.

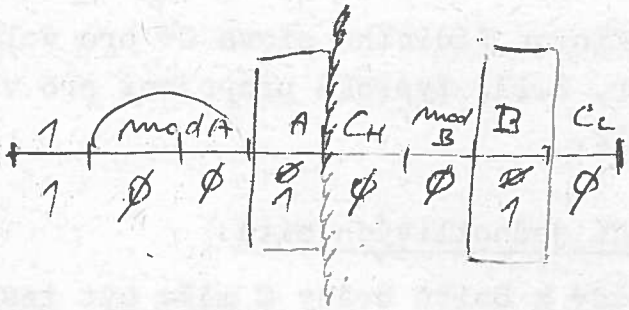
mód Ø

A = X

C_H = OUT

B = X

C_L = OUT



→ #8Ø = 128 mod Ø všechny brány OUT

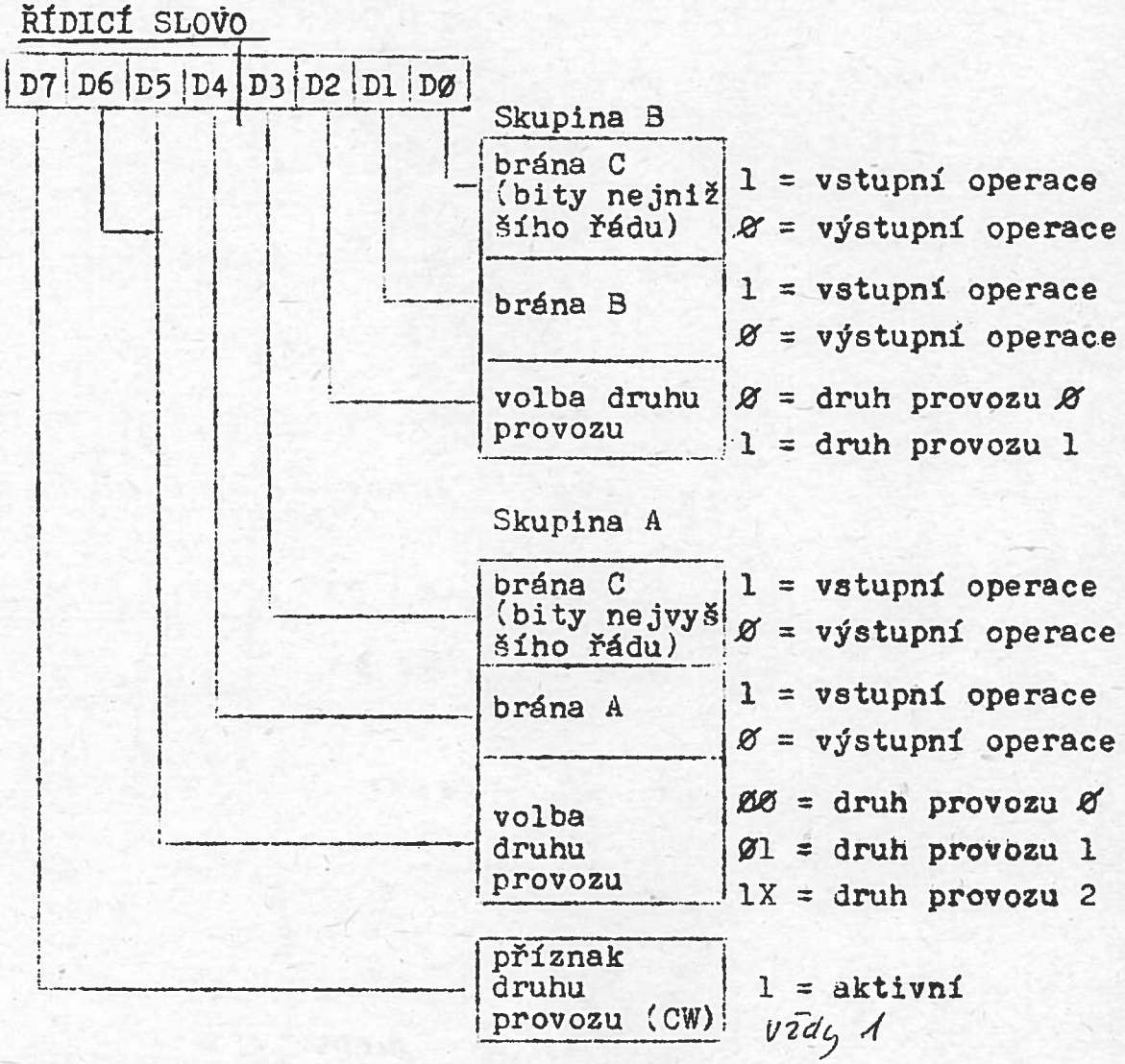
→ #82 = 130 mod Ø B = IN, jinak vše OUT

→ #9Ø = 144 mod Ø A = IN jinak vše OUT

FUNGUJE → #92 = 146 mod Ø A = IN, B = IN, jinak vše OUT

Poznámka: Všechny klopné obvody, programované maskou, se automaticky při výběru druhu provozu a při resetování obvodů (čipů) resetují.

V dalším budeme stručně charakterizovat všechny druhy provozu (režimy).



Obr. 2.11 Definice formátu řídicího slova 8255
pro volbu prac. módu (Ø, 1, 2) a režimu (IN/OUT) jednotlivých bran

při inicializaci 128 = #8Ø

D7	D6	D5	D4	D3	D2	D1	DO
1	0	0	1	0	0	Ø	Ø
128	64	32	16	8	4	2	1

128 + 16 = 144
128 + 18 = 146

přihlad

mód Ø

A = OUT

C_H = IN

B = IN

C_L = OUT

OUT 95, 255

PAUSE OUT 127, Ø RES DØ

PAUSE OUT 127, 1 SET DØ

START OUT 127, 4 RES D2

PAUSE 1

OUT 127, 5 SET D2

STOP OUT 127, 14 RES D7

PAUSE 1

OUT 127, 15 SET D7

<< OUT 127, 8 RES D4

PAUSE 1

OUT 127, 9 SET D4

>> OUT 127, 6 RES D3

PAUSE 1

OUT 127, 7 SET D3

CUE << << + START

CUE >> >> + START

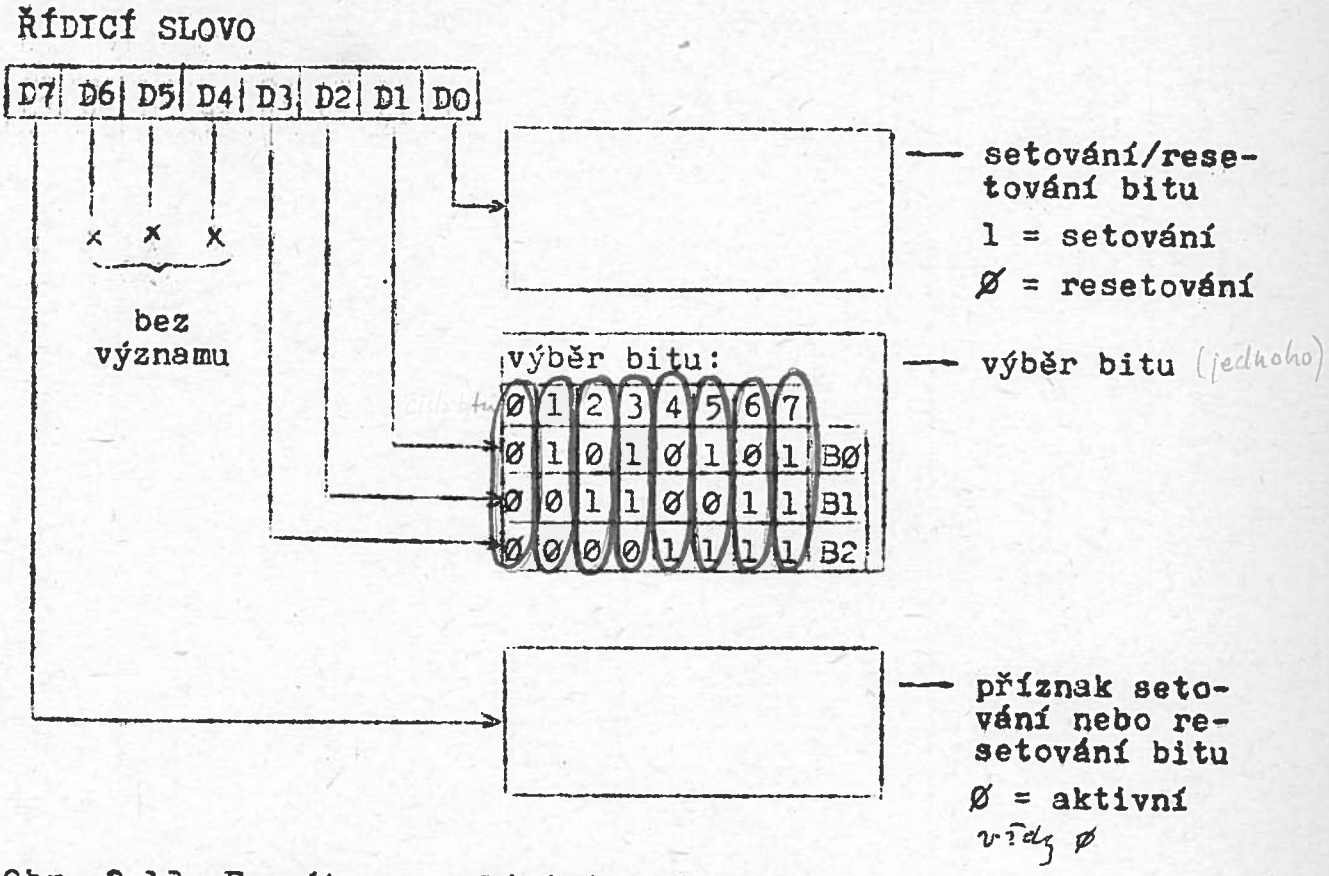
RECORD ~~DISC~~

OUT 127, 10 RES D5

PAUSE 1

OUT 127, 11 SET D5

	BIT							SET 1 RES Ø
	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	0	0	0	0	0	0	0	1
	0	0	0	0	0	1	0	Ø
	0	0	0	0	0	1	0	1
	0	0	0	0	1	1	1	0
	0	0	0	0	1	1	1	1
	0	0	0	0	1	0	0	0
	0	0	0	0	1	0	0	1
	0	0	0	0	0	1	1	0
	0	0	0	0	0	1	1	1
	0	0	0	0	1	0	1	0
					1	0	1	1



Obr. 2.13 Formát pro ovládání bitu brány C

jednoho

Druh provozu Ø (základní vstupní operace)

Funkční uspořádání umožňuje jednoduché vstupní a výstupní operace pro každou ze tří bran. Není nutná výměna potvrzení, neboť data jsou prostě do zvolené brány zapsána nebo z ní čtena. Definici bran v tomto případě zachycuje tabulka 2.6.

A		B		Skupina A			Skupina B	
D4	D3	D1	D0	brána A	brána C (bity D4, D5, D6, D7 vyššího řádu)	č.	brána B	brána C (bity D0, D1, D2, D3 nižšího řádu)
0	0	0	0	výstup	výstup	0	výstup	výstup
0	0	0	1	výstup	výstup	1	výstup	vstup
0	0	1	0	výstup	výstup	2	vstup	výstup
0	0	1	1	výstup	výstup	3	vstup	vstup
0	1	0	0	výstup	vstup	4	výstup	výstup
0	1	0	1	výstup	vstup	5	výstup	vstup
0	1	1	0	výstup	vstup	6	vstup	výstup

D4	D3	D1	D0	A			B	
0	1	1	1	výstup	vstup	7	vstup	vstup
1	0	0	0	vstup	výstup	8	výstup	výstup
1	0	0	1	vstup	výstup	9	výstup	vstup
1	0	1	0	vstup	výstup	10	vstup	výstup
1	0	1	1	vstup	výstup	11	vstup	vstup
1	1	0	0	vstup	vstup	12	výstup	výstup
1	1	0	1	vstup	vstup	13	výstup	vstup
1	1	1	0	vstup	vstup	14	vstup	výstup
1	1	1	1	vstup	vstup	15	vstup	vstup

Tab. 2.6 Definice bran pro druh provozu 0

Druh provozu 1 (strobovaný vstup/výstup)

Tato funkční konfigurace umožňuje vstup nebo výstup zvolenou bránou v korespondenčním režimu (handshaking), resp. se strobovacími signály. Tyto bitové signály jsou v režimu 1 pro bránu A nebo B generovány nebo přijímány vývody brány C. Definici režimu 1 odpovídají:

- dvě skupiny (A a B),
- každá skupina představuje jednu 8bitovou bránu a 4bitovou bránu,
- 8bitová brána může být vstupní i výstupní, vstup i výstup je s pamětí (latch),
- 4bitová brána se užívá pro řízení a kontrolu stavu 8bitové brány.

Definice signálů řízení vstupu:

- $\overline{STB}$ (Strobe Input) - $\overline{STB}$ = L strobuje vstup, t.j. převádí data ze vstupních svorek do vstupního registru,
- IBF (Input Buffer Full F/F) - IBF = H indikuje, že data již byla převzata do vstupního registru. Jinak řečeno IBF →L pro $\overline{STB}$ →H a opět IBF →L po vzestupné hraně signálu $\overline{RD}$.
- INTR (Interrupt Reguest) - INTR = H vyvolá v CPU přerušení, pokud to vstupní zařízení vyžaduje. INTR přejde na úroveň H při $\overline{STB}$ = H, IBF = H a INTE = H. Nuluje se sestupnou hranou $\overline{RD}$. Tím se umožňuje, aby vstupní zařízení si žádalo

o obsloužení mikroprocesorem pouhým strobováním dat do brány.

Působení signálů INTE A a INTE B je ovládáno nastavením a nulováním bitů PC_4 a PC_2. Dva bity PC_6 a PC_7, nevyužité pro řízení v režimu 1, mohou být naprogramovány na vstup nebo výstup a využity libovolně s kteroukoliv bránou nebo samostatně.

- $\overline{OBF}$ (Output Buffer Full F/F) - výstupní bit $\overline{OBF}$ přechodem na úroveň L indikuje, že CPU již předal data do příslušné brány. Klopný obvod OBF se nastaví vzestupnou hranou vstupu $\overline{WR}$ a nuluje se přechodem vstupu $\overline{ACK}$ na úroveň L.
- $\overline{ACK}$ (Acknowledge Input) - $\overline{ACK}$ = L indikuje, že data z brány A nebo B již byla převzata periferním zařízením.
- INTR (Interrupt Request) - úroveň H na tomto bitovém výstupu může být použita k přerušení CPU při převzetí dat periferním zařízením.
- INTR přejde na úroveň H při $\overline{ACK}$ = $\overline{OBF}$ = INTE = H. Nuluje se sestupnou hranou $\overline{WR}$.

Působení signálů INTE A a INTE B je ovládáno bity PC_6 a PC_2.

K volnému využití tentokráte zůstávají bity PC_4 a PC_5.

Jednotlivé kombinace zachycují obr. 2.14 až 2.16.

Druh pr vozu 2 (strobovaná obousměrná brána)

Tento režim se vztahuje pouze k bráně A a k pěti bitům brány C. Ostatní brány, fesp. bity, mohou být využity v režimu Ø nebo 1. Definici režimu 2 odpovídá:

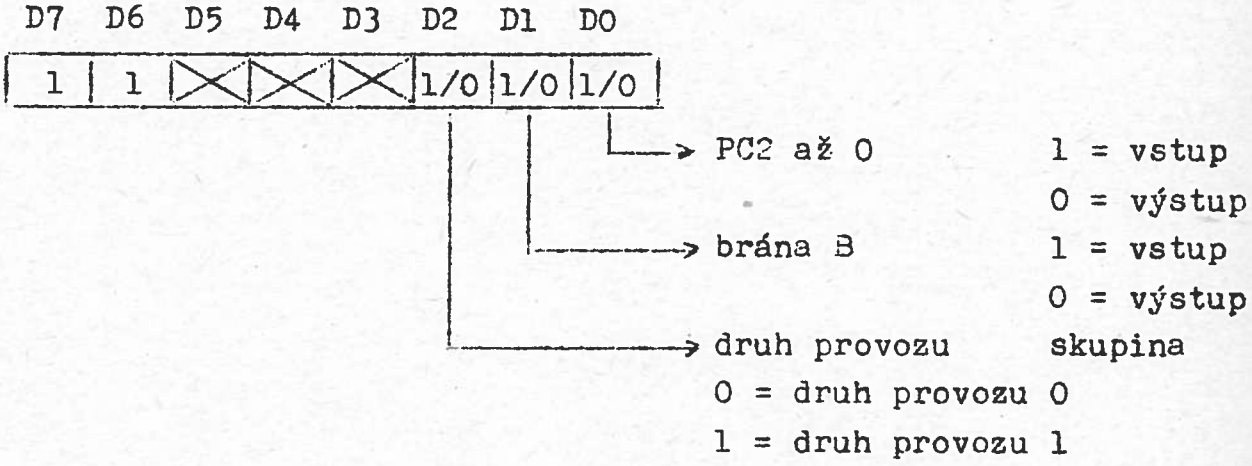
- práce jen se skupinou A,
- jedna 8bitová obousměrná brána A a pět bitů z brány C pro řízení a stav brány A,
- vstupy i výstupy jsou s pamětí.

Definice řídicích signálů obousměrné brány:

- INTR (Interrupt Request) - požadavek o přerušení CPU pro vstup i výstup bránou A,
- $\overline{OBF}$ (Output Buffer Full) - úroveň L oznamuje, že CPU již

předal data do registru brány A,
- $\overline{ACK}$ (Acknowledge) - úroveň L umožňuje třístavovému výstupu brány A předat data ven. V opačném případě výstup A bude ve 3. stavu.
- INTE 1 (klopný obvod INTE 1 přiřazený k $\overline{OBF}$) - nastavuje se bitem PC_6.
- $\overline{STB}$ (Strobe Input) - úroveň L převádí data z periferní strany do vstupního registru brány A.
- IBF (Input Buffer Full F/F) - úroveň H indikuje, že data již byla převzata do vstupního registru.
- INTE 2 (kolpný obvod INTE 2 přiřazený k IBF) - nastavuje se bitem PC_4.

Situaci opět zachycují obr. 2.17 až 2.22, možné kombinace zachycuje tabulka 2.7.



Obr. 2.17 Řídicí slovo pro druh provozu 2

Pokyny pro kombinace druhů provozu (režimů)

Jsou kombinace režimů, při kterých nejsou všechny bity brány využity pro řízení nebo pro informaci o stavu. Zbývající bity mohou být použity takto:

a) jsou-li naprogramovány jako vstupní - všechny vstupní vývody jsou přístupné při obyčejném čtení brány C;
b) jsou-li naprogramovány jako výstupní - bity brány C_U (PC_7

až PC_4) jsou individuálně přístupné bitovými operacemi (bit set/reset), bity brány C_L (PC_3 až $PC_\emptyset$) jsou individuálně přístupné bitovými operacemi nebo hromadně zápisem do brány C.

Zatižitelnost vývodů bran B a C:

Jakákoliv kombinace o s m i výstupů, náhodně vybraných z bran B a C může dodávat proudy až 1 mA při 1,5 V. To umožňuje z 8255 přímo napájet budiče, které tak velké proudy vyžadují.

Čtení stavu brány C

Brána C přenáší při pracovním režimu 0 data od nebo do periferních zařízení. Je-li 8255 naprogramován na pracovní režim 1 nebo 2, generuje nebo přijímá potvrzovací signály od periferních zařízení. Programátor může po vyhodnocení obsahu kanálu C otestovat nebo přezkoušet stav každého periferního zařízení, aby mohl případně změnit i průběh programu. Stavová informace z brány C se čte bez jakékoliv speciální instrukce.

	Druh provozu 0		Druh provozu 1		Druh provozu 2
	vstup	výstup	vstup	výstup	pouze skupina A
PAD	vstup	výstup	vstup	výstup	⟷
PA1	vstup	výstup	vstup	výstup	⟷
PA2	vstup	výstup	vstup	výstup	⟷
PA3	vstup	výstup	vstup	výstup	⟷
PA4	vstup	výstup	vstup	výstup	⟷
PA5	vstup	výstup	vstup	výstup	⟷
PA6	vstup	výstup	vstup	výstup	⟷
PA7	vstup	výstup	vstup	výstup	⟷
PB0	vstup	výstup	vstup	výstup	
PB1	vstup	výstup	vstup	výstup	pouze druhy
PB2	vstup	výstup	vstup	výstup	provozu 0 a 1
PB3	vstup	výstup	vstup	výstup	

PB4	vstup	výstup	vstup	výstup	
PB5	vstup	výstup	vstup	výstup	
PB6	vstup	výstup	vstup	výstup	
PB7	vstup	výstup	vstup	výstup	
PC0	vstup	výstup	$INTR_B$	$INTR_B$	E/A
PC1	vstup	výstup	IBF_B	$\overline{OBF}_B$	E/A
PC2	vstup	výstup	$\overline{STB}_B$	ACK_B	E/A
PC3	vstup	výstup	$INTR_A$	$INTR_A$	$INTR_A$
PC4	vstup	výstup	$\overline{STB}_A$	E/A	$\overline{STB}_A$
PC5	vstup	výstup	IBF_A	E/A	IBF_A
PC6	vstup	výstup	E/A	ACK_A	$\overline{ACK}_A$
PC7	vstup	výstup	E/A	OBF_A	$\overline{OBF}_A$

Tabulka 2.7 Přehledná tabulka definic druhů provozu

Uspořádání vstupů

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
		skupina A			skupina B		

Uspořádání výstupů

D7	D6	D5	D4	D3	D2	D1	D0
OBF_A	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	OBF_B	$INTR_B$
		skupina A			skupina B		

Formát stavového slova pro pracovní režim 2

D7	D6	D5	D4	D3	D2	D1	D0
OBF_A	$INTE_1$	OBF_A	$INTE_2$	$INTR_A$			
		skupina A			skupina B		

(Definováno výběrem pracovního režimu 0 nebo 1)

Obr. 2. 23 Formát stavového slova pro druh provozu 0 nebo 1

Programovatelný sériový interface 8251

8251 je univerzální obvod vysílač/přijímač (výkonový buffer) pro synchronní a asymchronní provoz pro přenos dat v mikropočítačových systémech. Je určen jako ostatní obvody pro periférie mikroprocesoru a je naprogramován tak, že může být použit prakticky pro všechny dnes používané druhy sériového přenosu dat. Výkonový buffer přijme paralelní znaky z mikroprocesoru a pro přenos je převede na spojitý sériový tok dat. Současně může data i přijímat a předávat je na paralelní znaky pro mikroprocesor. Výkonový buffer hlásí mikroprocesoru, kdy může nový znak pro přenos přijmout nebo předat na mikroprocesor. Mikroprocesor může v každém okamžiku "přečíst" stav výkonového bufferu včetně chyby při přenosu dat a včetně řídicích signálů, jako např. SYNDET a TxE. Obvod je vyráběn technologií n-kanálu na křemíku.

Vzhledem k tomu, že v cvičeních na školním počítači nebude tento obvod používán, odkazujeme zájemce na příslušnou literaturu.

Příklad propojení obvodů vstup/výstup.

Na obr. 2.24 je zapojení typického systému V/V v rozsáhlé kombinaci s jinými členy (8212, 8251 a 8255). Toto zapojení lze použít např. pro přizpůsobení interface na inteligentní terminál s obrazovkovým displejem, s tlačítkovou soupravou a komunikačním interface. Dále lze toto zapojení použít při řízení procesorů a přizpůsobením na senzory, relé nebo řídicí jednotky pro motory. Lze je aplikovat jak pro izolované, tak pro paměťové mapované obvody V/V.

2.8.5 Programovatelné pomocné obvody

K těmto obvodům řadíme programovatelné časovače 8253, řadič DMA 8257 a řadič přerušení 8259. Do této skupiny můžeme dále řadit obvod 8271 - řadič styku s pružnými disky; obvod 8273 - řadič SDLC protokolu; obvod 8275 - řadič styku s obrazovkovým terminálem atd. Z tohoto sortimentu si povšimneme pouze obvodu 8253, ostatní - s ohledem na jejich speci-

fické použití - nalezne čtenář v příslušné literatuře.

Programovatelný časovač 8253

Programovatelný obvod 8253 je mimořádně univerzální, neboť obsahuje tři 16 bitové čítače s předvolbou, které mohou být využity k nejrůznějším účelům, ale nejčastěji se uplatní při časování (Interval Timer).

Obvod charakterizující následující základní údaje:

- 3 nezávislé 16bitové čítače s předvolbou,
- frekvence čítání Ø až 2 MHz,
- čítání binární i dekadické,
- napájecí napětí + 5 V,
- pouzdro DIL o 24 vývodech,
- přímá slučitelnost s mikroprocesorem 8080A.

Obvod 8253 pomáhá řešit problémy generování časových intervalů, které vznikají téměř v každém mikropočítačovém systému. Namísto realizace programových smyček, programátor zajistí vhodnou inicializaci 8253 tak, aby jeden nebo více čítačů obsažených v 8253 nezávisle na CPU odpočítával impulsy a přerušil činnost CPU (INT) po uplynutí žádoucího intervalu. Existují přitom i jiné funkce, které lze snadno plnit obvodem 8253. Např. programovatelný generátor rychlosti, čítač impulsů, dělič, generování hodin pro reálný čas atd.

Schéma obvodu 8253 ukazuje obr. 2.25. Obvod 8253 spolupracuje s datovou sběrnicí 8080A přes vyrovnávací registr instrukcemi IN a OUT. Vyrovnávací registr zabezpečuje programování režimu, zavádění předvolby do čítačů i jejich čtení.

Logika čtení a záznamu má pět bitových vstupů z mikropočítače a generuje vmitřní řídicí signály pro 8253. Jsou to vstupy:

- $\overline{RD}$ - úroveň L oznamuje 8253, že CPU čte data, t.j. stavy čítačů
- $\overline{WR}$ - úroveň L oznamuje 8253, že CPU předává na sběrnici data, t.j. řídicí slovo CW nebo hodnotu předvolby čítačů,
- A_0, A_1 - tyto vstupy jsou obvykle připojeny k adresové sběrnici. Slouží k volbě jednoho ze tří čítačů a k adresování

registru řídicího slova CWR při určování režimu (módu),
- $\overline{CS}$ - úroveň L aktivuje 8253 pro operaci čtení nebo zápisu z CPU. Vstup $\overline{CS}$ však nemá žádný vliv na průběžnou činnost čítačů.

$\overline{CS}$	$\overline{DR}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	předvolba čítače č. Ø
0	1	0	0	1	předvolba čítače č. 1
0	1	0	1	0	předvolba čítače č. 2
0	1	0	1	1	zápis CW módu
0	0	1	0	0	čtení stavu čítače č. Ø
0	0	1	0	1	čtení stavu čítače č. 1
0	0	1	1	0	čtení stavu čítače č. 2
0	0	1	1	1	3. stav
1	-	-	-	-	desaktivace - 3. stav
0	1	1	-	-	3. stav

Tab. 2.8 Definice funkcí obvodu 8253

Registr řídicího slova CWR je adresován při $A_0 = A_1 = 1$, kdy přijímá a uchovává slabiku z vyrovnávacího registru datové sběrnice. Tato informace určuje operační režim každého čítače, výběr binárního nebo dekadického způsobu čítání a předvolbu registru. Do CWR lze pouze psát. Jeho obsah nelze zpětně zjišťovat čtením.

Čítače č. 0, 1 a 2 jsou funkčně identické a tak popíšeme jen jeden z nich. Každý čítač je 16bitový, s předvolbou a čítá dolů buď binárně nebo dekadicky. Funkce jeho vstupu (CLK), hradla (GATE), výstupu (OUT) jsou určeny naprogramovaným režimem (modem), uloženým v CWR.

Čítače jsou navzájem nezávislé a každý může mít odlišný režim způsobu čítání. Zjištění obsahu kteréhokoliv čítače může programátor zajistit jednoduchou operací čtení a také je k dispozici speciální povel a logika umožňující čtení stavu čítače "za chodu", aniž by se narušovalo čítání ze vstupu CLK.

Režim činnosti 8253 se volí programově prostou výstupní operací mikropočítače, přičemž každý z čítačů se nastavuje individuálně zápisem CW do CWR při A_0 A_1 = 11. Situaci zachycují tabulky 2.9 až 2.13.

Bit	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Symbol	SC1	SCØ	RL1	RLØ	M2	M1	MØ	BCD

Tab. 2.9 Formát CW 8253

SC1	SCØ	
0	0	čítač č. Ø
0	1	čítač č. 1
1	0	čítač č. 2
1	1	neplatný

Tab. 2.10 Volba čítače (SC-Select Counter)

M2	M1	MØ	režim
0	0	0	0
0	0	1	1
-	1	0	2
-	1	1	3
1	0	0	4
1	0	1	5

Tab. 2.11 Nastavení režimu (modu) čítače (M - Mode)

Režim Ø: Přerušení na konci čítání.

Výstup OUT čítače je po operaci volby režimu napřed na úrovni L. Poté, co je do čítače vložena hodnota předvolby, OUT zůstane na úrovni L a čítač zahájí čítání. Po dosažení dočítání přejde OUT na úroveň H a zůstane na ní, dokud registr uvažovaného čítače není znovu předvolen.

Změna předvolby v průběhu čítání vede na:

a) zavedení 1. slabiky předvolby zastaví probíhající čítání,
b) zavedení 2. slabiky předvolby odstartuje nové čítání.

Vstup GATE povoluje čítání, je-li na úrovni H a zabraňuje čítání při úrovni L.

RL1	RLØ	
0	0	Operace s pomocným registrem - viz operace čtení
0	1	Čtení/zápis jen vyšší slabiky
1	0	Čtení/zápis jen nižší slabiky
1	1	Čtení/zápis nižší slabiky a pak vyšší slabiky

Tab. 2.12 Čtení/předvolba (RL - Read/Load)

BCD	
0	Binární čítání - 16bitový čítač
1	Dekadické čítání - čítač o 4 dekádách

Tab. 2.13 Základ čítání (BCD)

Režim 1: Programovatelný monostabilní generátor

Výstup OUT přejde na úroveň L po čítání následujícím za vzestupnou hranou na vstupu GATE.

Výstup OUT přejde na úroveň H po dočítání. Jestliže byla vložena nová předvolba, zatímco OUT = L, až do dalšího přepnutí to nikterak neovlivní délku osamoceného impulsu (One-Shot). Průběžný stav čítače může být čten kdykoliv, aniž by to ovlivnilo generování impulsu.

Impuls je možné znovu spouštět i v d bě jeho trvání, neboť OUT zůstane na úrovni L po celou dobu čítání po vzestupné hraně vstupu GATE.

Režim 2: Dělení frekvence číslem N

Výstup OUT bude na úrovni L po jednu periodu průběhu na hodinovém vstupu CLK (Clock). Rozteč mezi výstupními impulsy se rovná číslu N zadaném do registru čítače (count register - - není totožný s čítačem). Jestliže byl obsah registru čítače znovu změněn mezi dvěma impulsy, probíhající výstupní perioda se nezmění. Následující perioda však už bude odpovídat nové předvolbě.

Vstup GATE, pokud je na úrovni L, převede OUT na úroveň H. Jakmile nastane GATE = H, zahájí čítač čítání z počáteční hodnoty. Vstup GATE proto může být využíván k synchronizaci čítače.

V případě nastavení režimu 2 zůstane OUT = H, dokud není registr čítače naplněn. Výstup proto může být synchronizován i programem.

Zavedení čísla N do registru čítače však musí probíhat přesně v pořadí naprogramovaném v CW modu (RLØ, RL1). Tato operace sice může nastat kdykoli po CW modu, ale jakmile začne být registr čítače plněn, musí být naplněn tím počtem slabik, které je určeno v CW modu.

Všechny čítače čítají dolů, t.j. jejich obsah se čítáním zmenšuje a čítání se končí dosažením nuly (terminal count). Zavedení nuly do registru čítače vede k maximálnímu počtu čítání (2^{16} binárně, 10^4 dekadicky). V režimu Ø nové čítání nezačne, dokud není dokončeno plnění registru počítače, který očekává jednu nebo dvě slabiky v závislosti na CW modu (RLØ, RL1). Pak se pokračuje operací nulování (restart).

V četných aplikacích je třeba zjišťovat průběžnou hodnotu v čítači za chodu a rozhodovat podle ní. Čítání událostí (impulsů) je patrně nejobvyklejší aplikací, kdy je to zapotřebí. 8253 zahrnuje i logiku, která dovoluje číst obsah kteréhokoliv ze tří čítačů bez narušování jeho činnosti.

Existují dvě metody. První metoda spočívá v použití prosté vstupní operace čtení CPU z čítače, kdy prostřednictvím adresových bitů A_1, A_0 programátor zvolí žádoucí čítač. Připomeňme zde však, že při A_1 A_0 = 11 (CW modu) žádné čtení nemůže nastat. Jediné omezení této metody spočívá v tom, že pro zajištění ustálené hodnoty v čítači se musí na dobu čtení zastavit čítání ve zvoleném čítači. Zajistí se to snadno s pomocí řídicího vstupu GATE nebo externí logikou, která zablokuje vstup CLK. Obsah vybraného čítače se čte nadvakrát. Jako první se čte LSB a jako druhý MSB.

Je zcela nezbytné uskutečnit obě čtení. Jestliže se naprogramuje čtení dvou slabik, pak obě se musí přečíst dříve, než se do téhož čítače opět zapisuje.

Druhým způsoben je čtení okamžitého obsahu čítače bez jakéhokoliv zásahu do operace čítání. 8253 je k tomu vybaven speciální logikou, která se zpřístupní zavedením jednoduchého povelu do CWR. Když si programátor přeje zjistit stav určitého čítače za chodu, pošle do příslušného CWR slabiku CW modu následujícího formátu:

D7	D6	D5	D4	D3	D2	D1	DØ
SC1	SCØ	Ø	Ø	-	-	-	-

kde specifikují:

- SC1 a SC2 - čítač, jehož stav má být zachycen do pomocného registru (latch),
- 00 v D5 a D4 označují operaci zachycení stavu,
- ostatní bity CW (D3 až DØ) jsou bez významu.

Jako u předešlé metody, i u tohoto způsobu je nutné uskutečnit úplnou operaci čtení, jak byla programována.

2.9 Zásady pro tvorbu programů a jejich ladění

Sestavení programu pro mikropočítač, jak již bylo uvedeno, znamená sestavení sledu kódovaných instrukcí. Při návrhu programu musíme přihlédnout k reálným možnostem mikropočítače (schopnost testování, způsoby adresování atd.). Řešení programového zabezpečení přitom bývá vhodné rozdělit na menší, dobře přehledné a snadno odladitelné části, tedy zpracovat ve formě podprogramů.

Tato forma je natolik výhodná, že mnohé programy pro mikroprocesorem řízené přístroje či zařízení jsou organizovány tak, že hlavní program tvoří pouze posloupnost volání podprogramů, každý z těchto podprogramů volá opět další podprogramy atd. Je ovšem pravda, že tato struktura programu má i své nevýhody, např. v kritických aplikacích v reálném čase může zpomalovat dobu výpočtu či odezvy.

Mezi hlavní výhody řešení programového vybavení pomocí podprogramů patří:

1) Každý z podprogramů můžeme vytvořit jako relativně krátký, dobře srozumitelný a relativně dobře testovatelný a odladitelný blok celkového programu.
2) Vazby mezi všemi částmi celkového programu (t.j. mezi volajícím a volaným podprogramem) můžeme vytvořit přehledné a dobře definovatelné.

Z těchto hledisek si v dalším všimneme základních vlastností mikropočítače a jejich použití při sestavování programů.

Režim 3: Generování obdélníkových průběhů

Činnost v režimu 3 je podobná režimu 2, až na to, že výstup OUT zůstane na úrovni L, dokud neproběhne polovina všech čítání (pro sudé N), a přejde na úroveň H na dobu druhé poloviny z celkového počtu čítání. V případě, že číslo N je liché, bude OUT = H po dobu (N + 1) /2 čítání a OUT = L po dobu (N - 1) /2 čítání.

Pokud však registru čítače zadáme novou hodnotu v průběhu čítání, nová hodnota se uplatní ihned po té změně výstupu OUT, která odpovídala předchozímu obsahu.

Režim 4: Programem ovládané strobování

Po nastavení režimu je OUT = H a po zavedení N začne čítač pracovat. Po dočítání jde výstup OUT na úroveň L na dobu jedné periody CLK a pak se vrací na H.

Jestliže je obsah registru čítače změněn v době mezi výstupními impulsy, průběžná perioda výstupu se tím neovlivní. Nová hodnota N se však uplatní ihned v další periodě. Čítání se zastaví, je-li vstup GATE = L.

Režim 5: Obvodově ovládané strobování

Čítač zahájí čítání po vzestupné hraně ovládacího vstupu GATE a po dočítání jde výstup OUT na úroveň L na dobu 1 periody CLK.

Výstup OUT neklesne na úroveň L, dokud po poslední vzestupné hraně GATE neodpočítá úplné číslo N.

Operace zápisu a čtení do 8253

Při zápisu musí program mikropočítače každý z čítačů 8253 inicializovat, t.j. zapsat řídicí slovo CW (režim) a hodnotu N (1 nebo 2 slabiky) pro registr čítače, dříve než přistoupí ke skutečnému použití vybraného čítače. Přitom zapsání CW je možné v libovolném pořadí. (Každý CWR má vlastní adresu).

2.9.1 Testování

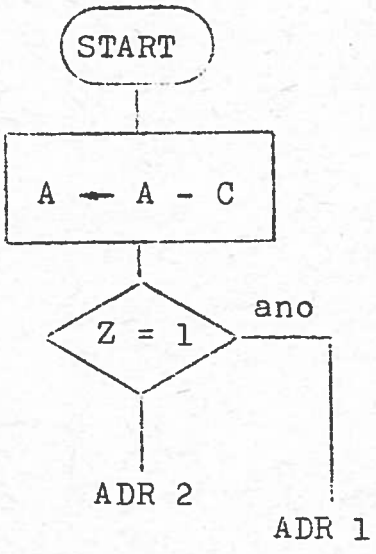
Schopnost testovat booleovské (t.j. dvouhodnotové) proměnné patří k velmi důležitým vlastnostem počítače. Pro testování obvykle používáme zvláštní skupinu registrů (viz odst. 2.1), u mikroprocesoru 8080A je to registr příznaků. Hodnoty bitů tohoto registru můžeme pomocí některých instrukcí testovat a podle výsledku testu větvit program. Přitom testy prováděné v programu můžeme rozdělit:

a) jednoduché testy řešící dvouhodnotová rozhodování,
b) testy řešící mnohohodnotová rozhodování

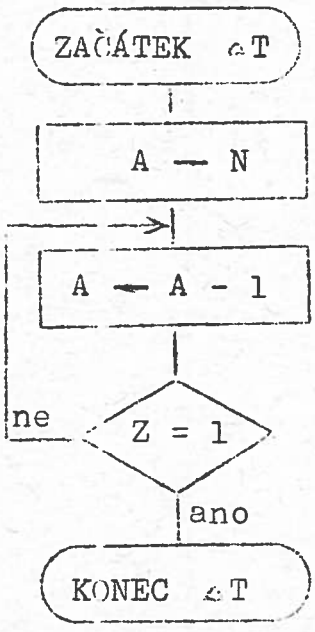
ad a) Jednoduché testy:

V těchto případech nás zajímá pouze hodnota jedné proměnné testované počítačem, např. hodnota příznakového bitu nulového výsledku (Z). Příkladem může být test na obsah střadače po provedení určité operace (např. po odečtení obsahu registru C). Příklad takového programu ukazuje obr. 2.26. Shodným způsobem můžeme použít i dalších příznakových bitů aplikací příslušné informace. Podobnou záležitostí jsou i programy realizující zpoždění určitou částí programu.

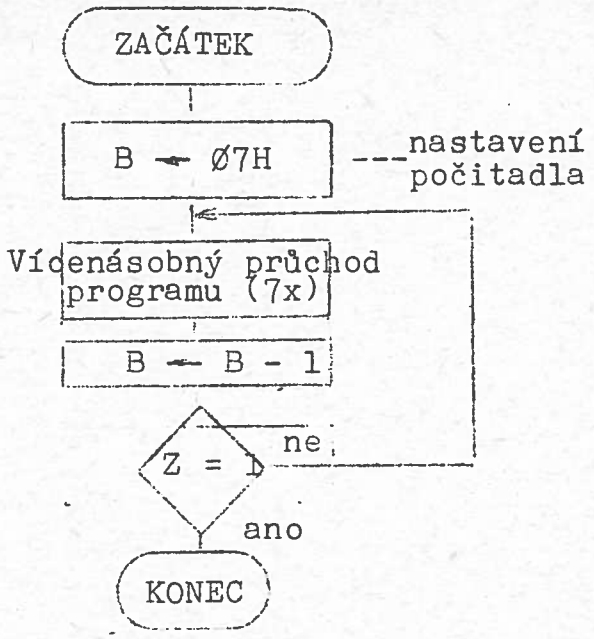
Situaci ukazují obr. 2.27 a 2.28



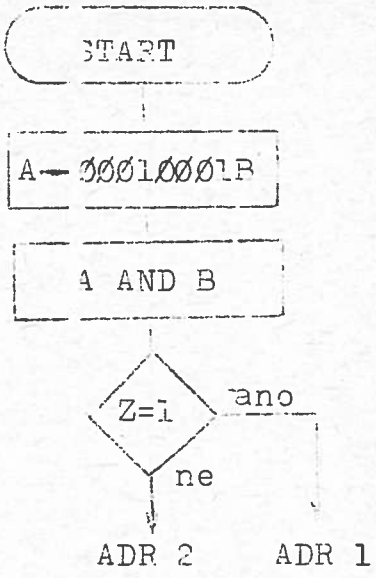
Obr. 2.26 Test nulového výsledku



Obr. 2.27 Realizace zpoždění pomocí cyklu



Obr. 2.28 Vícenásobný průchod programu skupinou instrukcí



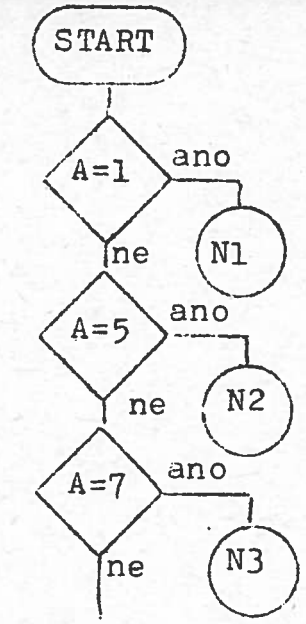
Obr. 2.29 Určení, zda alespoň jeden ze dvou bitů má hodnotu

V prvním případě je využit přímo střadač, v druhém případě použijeme některého z univerzálních registrů procesoru k uložení čísla udávajícího požadovaný počet průchodů cyklem. V případech, ve kterých máme za úkol testovat určitou množinu prvků, je vhodné použití tzv. "masky", t.j. vhodně zvoleného čísla, které při operaci AND vyloučí nepřebné bity. Příklad takového programu ukazuje obr. 2.29, kde máme za úkol zjistit, zda jsou bity b_0 a b_4 v registru B jedničkové. Do střadače vložíme masku, která vyloučí nepotřebné bity a pomocí příznaku Z větvíme program na ADR1 nebo ADR2.

ad b) Testy řešící mnohohodnotová rozhodování

Pokud posloupnost testů, řešících mnohohodnotová rozhodování není příliš dlouhá, lze použít kaskádu jednoduchých rozhodování obdobných principu z příkladu na obr. 2.29.

Příklad: Rozhodněte, zda se obsah střadače shoduje s některým z čísel 1, 5, 7. Pokud se rovná 1, pokračuje program na návěští N1, pokud 5 na N2, pokud 7 na N3.



Obr. 2.30 Mnohohodnotové rozhodování

Řešení úkolu ukazuje obr. 2.30. Pro porovnání je použita instrukce CPI, která porovnává obsah střadače s operandem v druhé slabice instrukce a v případě jejich shody nastaví Z = 1.

Obsah střadače zůstává po provedení instrukce nezměněn. Uvedený princip je vhodný pouze do určitého omezeného počtu, rozhodování, při větším počtu je vhodnější použití tabulek (viz dále). Principem je prohledávání tabulky referenčních znaků, kde podle pořadového čísla znaku znaku nalezeného v tabulce se v jiné tabulce vyhledá adresa příslušného skoku.

2.9.2 Zásobníková paměť

Při návrhu programu (viz odst. 2.9) často používáme podprogramů. Instrukce pro volání podprogramu (CALL) obsahuje adresu, která nahradí obsah čítače instrukcí. Po provedení podprogramu se však počítač musí vrátit na toto místo programu, odkud byl podprogram volán (resp. na místo následující). Znamená to, že dříve než nahradíme obsah čítače instrukcí počáteční adresou podprogramu, musíme stávající adresu z čítače instrukcí dočasně uložit na nějaké známé místo v paměti. Aby se pak počítač po provedení podprogramu mohl vrátit do volajícího programu, stačí, aby instrukce návrhu z podprogramu (RET), umístěná v podprogramu jako poslední, přesunula zpět do čítače instrukcí dříve uloženou návratovou adresu.

Ve výše uvedeném případu jsme používali paměťové místo jako součást struktury a návratu z podprogramu. Struktura pro volání a návrat z podprogramu používá zásobníkové paměti (Stack). Zásobníková paměť je typu poslední dovnitř - prv-

ní ven (LIFO - Last In First Out).

U mikroprocesoru 8080A je zásobníková paměť vytvořena v paměti RWM a v procesoru je pouze registr zvaný ukazatel zásobníku (Stack Pointer - SP), který ukazuje na poslední obsazenou pozici zásobníku - tzv. vrchol zásobníku (viz odst. 2.1).

Při volání podprogramu je návratová adresa uložena na vrchol zásobníku a při návratu z podprogramu je adresa z vrcholu zásobníku přesunuta do čítače instrukcí. Tato struktura zásobníku (zásobníkové paměti) umožňuje, aby hlavní program volal podprogram a tento podprogram volal další podprogram atd. Toto vnořování podprogramů je zajištěno právě uložením každé návratové adresy na právě platný vrchol zásobníku. Adresování zásobníku, t.j. regulace ukazatele zásobníku (SP) tak, aby ukazoval vždy na vrchol zásobníku, je automatické při volání i při návratu z podprogramů.

Dokumentace objasňující vazby mezi podprogramem a některým z volajících programů musí obsahovat:

1) Ve kterých registrech hledá volaný podprogram hodnoty parametrů předávaných volajícím programem (vstupní parametry).
2) Ve kterých registrech předává volaný podprogram výsledky (výstupní parametry).
3) Které registry jsou ovlivněny volaným podprogramem.
4) Jména podprogramů, které volaný program sám volá.

Informace o podprogramu je vhodné zapisovat formou komentáře vždy před vlastní zápis v jazyce symbolických adres.

2.9.3 Tabulky

Tabulky se při programování mikropočítačů často používají a jejich vhodným použitím můžeme program zjednodušit nebo zkrátit dobu výpočtu. Při vytváření algoritmu programu rozvážíme, které aspekty problému jsou více náhodné než pravidelné a ty náhodnější se snažíme vyřešit použitím tabulek.

Abychom mohli vyzvednout správně určitou položku z tabulky musíme znát:

1) Počáteční adresu celé tabulky.
2) Posunutí adresy dané položky oproti počáteční adrese nebo pořadí položky v tabulce.
3) Velikost položky (tzn. kolika slabikami je tvořena).

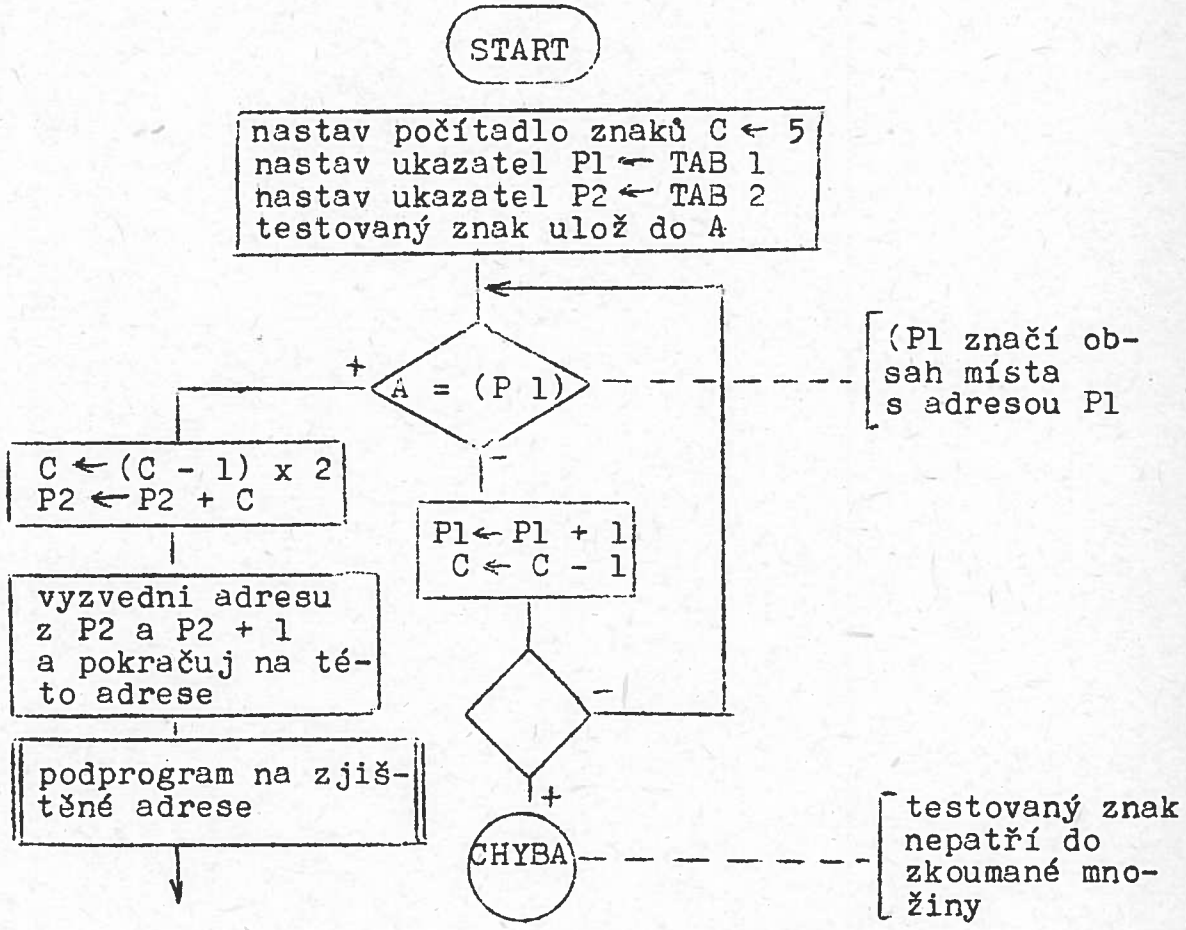
Pokud jsou všechny položky v tabulce stejné, mohou být za sebou uloženy bez oddělení a k určení vstupní adresy tabulky nám stačí znalost její počáteční adresy, pořadí položky v tabulce a velikost položky. V opačném případě oddělíme jejich konce koncovým znakem. V takovém případě musíme znát všechny počáteční adresy položek, nebo pouze počáteční adresu tabulky a posun jednotlivých položek proti této adrese.

Jako příklad použití tabulky můžeme uvést např. kódování adres podmíněných skoků při mnohonásobném rozhodování (srovnej s postupem v odst. 2.9.1). Bližší je zřejmé z následujícího příkladu (lit. /23/).

Příklad: Máme rozhodnout, zda určitý znak je shodný s některým ze znaků dané množiny. Každému znaku z této množiny je přiřazena počáteční adresa podprogramu. Pokud se testovaný znak shoduje s některým ze znaků této množiny, máme adresu odpovídající znaku vyzvednout z tabulky. Situaci ukazuje obr. 2.31. Testovaný znak uložíme do střadače. Program zjišťuje, zda-li je znak ve střadači shodný s některým ze znaků A, B, C, D, E. Počáteční adresa tabulky znaků je TAB1. Znakům z této tabulky jsou přiřazeny dvouslabikové adresy a počáteční adresa tabulky adres je TAB 2.

Nastavíme počítadlo znaků C, ukazatel P1 na počáteční adresu tabulky znaků a ukazatel P2 na počáteční adresu tabulky adres. Algoritmus prohledává tabulku znaků, nenajde-li žádnou shodu, končí a hlásí, že znak nepatří do povolené množiny. Najde-li shodu, vypočítá vstupní adresu tabulky adres a adresu přiřazenou nalezenému znaku vyzvedne z ta-

bulky adres. Program pak může pokračovat skokem na zjištěnou adresu, kde začíná podprogram volený právě testovaným znakem.



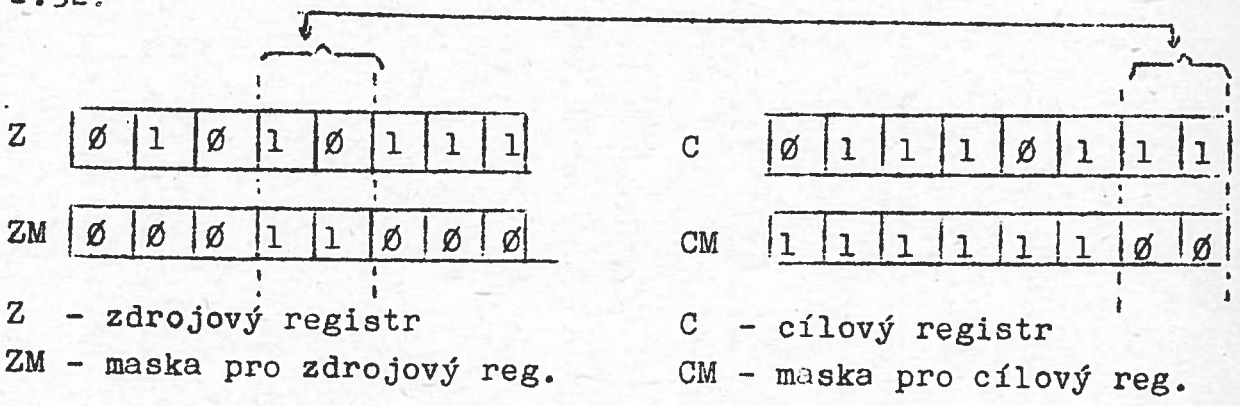
Obr. 2.31 Použití tabulek při mnohonásobném rozhodování

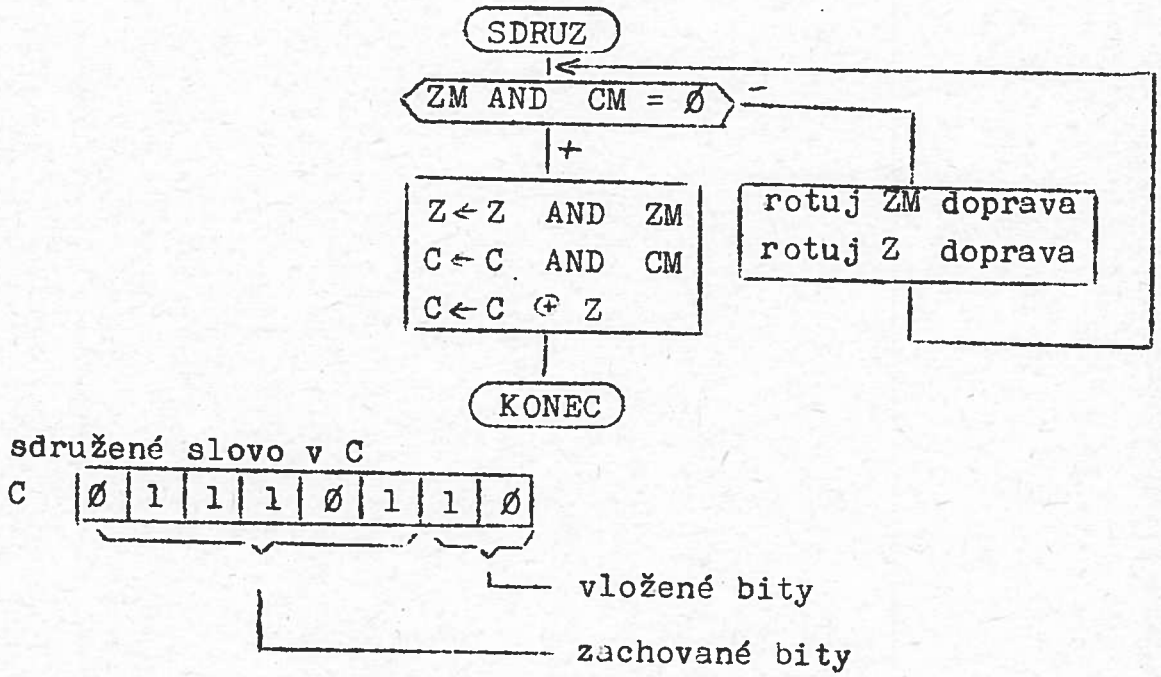
2.9.4 Bitové operace

Při realizaci určitého problému, resp. zařízení řízeného mikroprocesorem, se snažíme klást hlavní důraz na programové zabezpečení (software). Při manipulaci s obvody V/V často zjišťujeme, že slovo mikropočítače je buď příliš dlouhé nebo naopak nepostačující pro potřebný počet vstupních (výstupních) signálů. V takovém případě sdružujeme jednotlivé, vzájemně nesouvisející, signály do ucelených slov. Tím zároveň minimalizujeme i technické řešení (hardware), neboť se sníží počet vstupních/výstupních bran. Např. 20 vstupních signálů a 11 výstupních signálů sloučíme do 4 bran vhodně naprogramovaných obvodů 8255. Pro roztřídění jednotlivých signálů potom použijeme programové zabezpečení, stačí, když

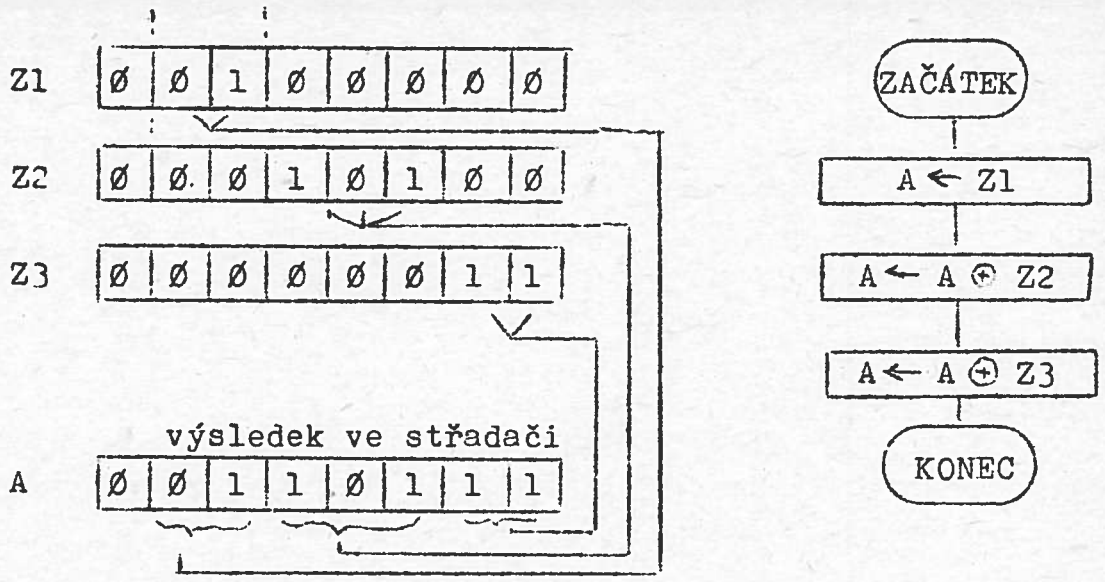
si zapamatujeme pozici signálů ve vstupní (výstupní) bráně. Pomocí programu tedy budeme bity buď sdružovat nebo naopak třídit.

Základní postup při sdužování bitů spočívá v odebrání určitého počtu bitů ze zdrojového slova Z, jejich posunutí do nové polohy uvnitř slova a uložení do cílového slova C bez ovlivnění jeho zbývajících bitů. Pro ilustraci použijeme program SDRUZ, převzatý z lit. /23/. Situaci ukazuje obr. 2.32.





Obr. 2.32 Sdružování bitů do jednoho slova



Obr. 2.33 Sdružování předem připravených slov

Před použitím podprogramu si připravíme zdrojové slovo i cílové slovo. Zdrojové slovo přesuneme do registru označeného Z, cílové slovo do registru C. Do dalších dvou registrů (ZM a CM) připravíme masku. V registru ZM budou jedničky pouze na pozicích, odkud odebíráme sdružované bity, v registrech CM budou nuly pouze na pozicích, kam bity ze zdrojového slova ukládáme. Podprogram SDRUZ nejprve zkoumá, zdali jsou pozice sdružovaných bitů ve zdrojovém i cílovém slově shodné. Test se provede pomocí operace AND mezi oběma maskami. Pokud nejsou pozice shodné, posouváme pomocí instrukce pro kruhový posun doprava o jeden bit obsah zdrojového registru i jeho masku, až dojde ke shodě. Pak operací Z AND ZM vynulujeme ve zdrojovém registru všechny bity, kterých se sdružování netýká a naopak v cílovém registru operací C AND CM vynulujeme ty pozice, na které přisouváme sdružované bity. Nakonec operací EX-OR sloučíme obě sdružené části do jednoho slova.

Výše popsaný algoritmus vyžaduje použití mnoha přesunů. Zjednodušení můžeme dosáhnout vynulováním cílového registru před zahájením sdružování a současně připravením bitů zdrojového registru do cílových pozic spolu s vynulováním nepotřebných bitů. Pak stačí pouze přesunout cílové slovo do střadače a pomocí EX-OR mezi střadačem a zdrojovým slovem slova sdružit. Tuto situaci zachycuje příklad programu na obr. 2.32.

Princip třídění bitů slova spočívá ve dvou krocích:

1) Všechny nepotřebné bity slova vynulujeme pomocí masky.

2) Zbylé bity posuneme vlevo nebo vpravo, jak to aplikace vyžaduje.

2.9.5 Pole a použití ukazatelů

Častým úkolem při realizaci určitého úkolu je manipulace s velkým množstvím slov (např. ukládání dat z A/D převodníku, textové řetězce atd.). Pole je charakteristické tím, že jeho prvky čteme nebo ukládáme jeden po druhém tak, jak jdou za sebou z jednoho konce pole na druhý a naopak. Přístup k jednotlivým prvkům zajišťuje ukazatel, obsahující adresu prvků pole a jehož dekrementací nebo inkrementací můžeme jednotlivé prvky vyzvednout. Bližší podrobnosti viz např. v lit. /23/.

2.9.6 Styk mikropočítače s vnějším prostředím

Pro styk mikropočítače s vnějším prostředím musí být k němu připojena příslušná periferní zařízení. K nim můžeme zařadit např. alfanumerickou nebo hexadecimální klávesnici, displej, tiskárnu, snímač, děrovač, převodníky D/A a A/D atd. Všechna tato periferní zařízení jsou připojena přes stykové obvody - interfejsy (Interface) a obvody vstupů/výstupů k mikropočítači.

Z hlediska jejich řízení můžeme rozdělit:

1) Řešení programovým řízením, kdy zařízení, které má aktivovat nějakou operaci, použije vhodný stavový bit (nebo jejich soubor) a mikropočítač, který periodicky tyto bity testuje rozhodne o další činnosti. Tímto způsobem vlastně periférie sděluje počítači svůj stav (připraven, porucha atd.).

2) Řešení pomocí systému přerušení, kdy zařízení oznámí, pomocí speciálního vstupního signálu, požadavek na přerušení hlavního programu. Procesor přeruší svoji činnost (dokončí právě prováděnou instrukci), provede požadovaný úkon

(obslužný program přerušení) a vrátí se zpět tam, kde byl přerušen a pokračuje ve vykonávání původního programu.

3) Řešení pomocí přímého přístupu do paměti - DMA (Direct Memory Access)

V tomto případě spočívá operace v přenosu bloků dat do paměti nebo naopak a vyloučením činnosti procesoru. Tohoto způsobu se používá zejména tehdy, jestliže situace vyžaduje velmi rychlou odezvu.

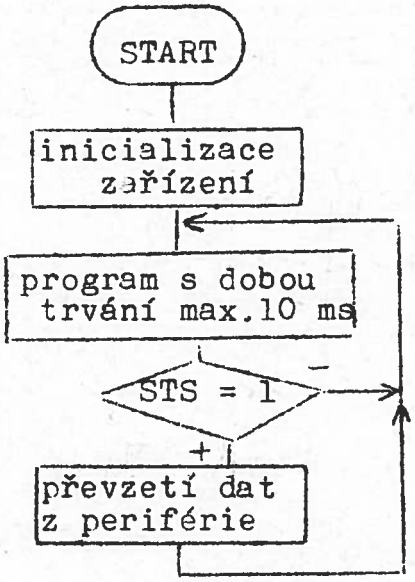
V dalším si všimneme zejména dvou prvních způsobů styku mikropočítače s periferními jednotkami, neboť třetí způsob (DMA) vyžaduje určitá technická zabezpečení a svým rozsahem se vymyká ze základních aplikací.

Programové řízení

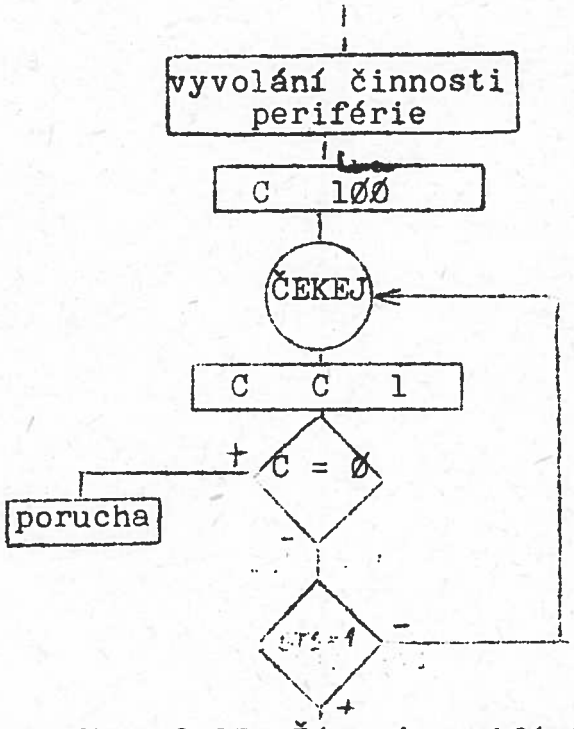
Použití stavových bitů, případně sdružených do stavového slova, představuje pružný a účinný přístup s minimálními požadavky na technické zabezpečení. Používáme jej zejména tehdy, jestliže periférie nevyžaduje (z hlediska rychlosti mikropočítače) častou obsluhu, nebo naopak obsluhu v určitých časových úsecích. Typickým příkladem je vstup dat z klávesnice. Je zřejmé, že zadávání dat operátorem bude podstatně pomalejší než rychlost mikroprocesoru a že postačí, když bude mikropočítač testovat stav klávesnice pouze v určitých časových úsecích (např. každých 10 ms) a ve zbývajícím čase bude vykonávat jinou činnost. Strukturu takovéhoto programu ukazuje obr. 2.34.

Obdobou této úlohy je např. ovládání motoru nebo jiného zařízení, např. ovládacích magnetů brzdy snímače děrné pásky. Tyto úlohy patří do skupiny úloh, kdy mikropočítač spustí určitou operaci v periferním zařízení a čeká na odezvu, kterou vyšle periférie nastavením určitého stavového bitu indikujícího ukončení činnosti. V tomto případě mikropočítač periodicky testuje stavový bit v tzv. čekací smyčce. Protože však očekávaná odezva také nemusí nastat (porucha) je nutné zabránit uváznutí programu ve smyčce např. způsobem naznačeným na obr. 2.35. Zde vycházíme ze znalosti časového okamžiku, který má

maximálně uplynout než se uskuteční požadovaný úkon. Do smyčky zařadíme počítadlo představené tak, aby program čekal ve smyčce jen o málo delší dobu než nastane očekávaná odezva. Pokud do této doby nedojde k nastavení stavového bitu, indikuje počítač poruchu, může spustit diagnostický test atd.



Obr. 2.34 Struktura programu při periodickém testování



Obr. 2.35 Řízení periférie

Přerušení

Princip přerušení byl popsán v úvodu této kapitoly. Procesor obsahuje klopný obvod (INTE), který lze programově nastavit nebo vynulovat (instrukce EI a DI) a jehož stav indikuje, zda v dané části programu je přerušení povoleno či nikoliv. Mimo ovládání programového je tento obvod nulován po zapnutí napájení a dále po každém přijetí požadavku na přerušení. Pokud je vynulován, je požadavek na přerušení ignorován. Po přijetí žádosti o přerušení procesor automaticky zablokuje další možnost přerušení (vynulováním klopného obvodu), uloží stávající obsah čítače instrukcí do zásobníkové paměti a nastaví čítač instrukcí na počátační adresu obslužného programu na které potom program pokračuje. Jde vlastně o obdobu vo-

lání podprogramu s tím rozdílem, že instrukce volání je vyvolána technickými prostředky a nikoliv programem. Návrat z obslužného programu je pak řízen obvyklým způsobem, t.j. zařazením instrukce RET na konec obslužného programu. Vlastní obslužný program pak může umožnit další přerušení (ještě naléhavější požadavek) zařazením instrukce EI. Po ukončení obslužného programu je třeba obnovit původní obsahy registrů procesoru (přečtením ze zásobníku). Bližší podrobnosti vyplývají přímo z kap. 2., odst. 2.3 a 2.7.

Připomeňme si, že při přerušení přikládá instrukci volání obslužného podprogramu (obvykle RST n, méně často CALL) na společnou sběrnici přerušující periferní zařízení (není tedy čtena z paměti). Řídicí obvod sběrnice 8228 má schopnost samostatně uložit na sběrnici instrukci RST 7 (připojíme-li výstup INTA přes 1 kΩ na + 12 V). Požadujeme-li víceúrovňové přerušení, můžeme použít řadič přerušení 3214 nebo 8259.

Dále si připomeňme, že instrukce DI (zákaz přerušení) účinkuje ihned, jakmile se provede. Instrukce EI (povolení přerušení) je účinná až po první následující instrukci, to proto, aby byl zajištěn před dalším přerušením správný návrat do hlavního programu (a nedošlo k přeplnění zásobníku návratovými adresami). Pro obsluhu většího počtu periférií, využívajících přerušení musíme zjistit, které zařízení žádá o přerušení. Toto můžeme uskutečnit nejsnáze technickými prostředky ve spolupráci s programovým zebezpečením. Nejvhodnější řešení představuje použití prioritních kodérů (např. 3214 nebo 8259). Bližší podrobnosti nalezne čtenář v lit. /17-, /19/, /23/, /24/.

3. Školní mikropočítač ŠMS - VÚVT

Školní mikropočítač ŠMS - VÚVT představuje prostředek pro výuku základních poznatků o mikroprocesoru 8080A. Svojí konstrukcí představuje standardní zapojení mikroprocesoru 8080A rozšířené tak, aby byla možná, pro školní účely, realizace maximálního počtu typických úloh. Blokové schéma mikropočítače je uvedeno na obr. 3.1, pohled na desku mikropočítače ukazuje obr. 3.2, mapu zachycuje obr. 3.3.

Školní mikropočítač ŠMS - VÚVT obsahuje základní paměť PROM (max. 4 kB) s monitorem, paměť RWM (max. 2 kB), vstupní/výstupní obvody (28 volných V/V), časovač (8253), A/D a D/A převodník (8 bitů), modem pro kazetový magnetofon (100 - 300 Baud), interfejs RS 232 C (V 24), klávesnici a displej. Další možnosti rozšíření systému jsou zřejmé z obr. 3.1 Konstrukčně je systém uspořádán ve dvou kufřících, z nichž jeden obsahuje vlastní mikropočítač se zdrojem (viz obr. 3.2), druhý uživatelské periferie (megnetofon MK27, motorek, reproduktor). Spojení s okolím systému lze fyzicky provést přes devět 16-pólových kontaktových polí, dva 5-pólové konektory (MK27 a SM7202) a tři 16-pólové konektory IO pro V/V. Systém - viz dále - je doplněn o interfejs pro dálnopis T 100 a doplněn pomocným monitorem rozšiřujícím možnosti použití.

Jak je zřejmé z mapy paměti (obr. 3.3) má použitá konfigurace ŠMS - VÚVT osazeny celkem 3 kB paměti PROM a 1 kB paměti RWM.

Pro ovládání mikropočítače slouží 9 příkazových kláves (červené) a 16 hexadecimálních kláves, které jsou zároveň určeny pro vyvolávání obsahu registrových párů na displej.

Monitor (základní) zabezpečuje obsluhu uživatelských přerušení, zápis a čtení dat do paměti atd. V použité verzi je vestavěn rozšiřující (pomocný) monitor, který zabezpečuje obsluhu dálnopisu T 100 použitého ve funkci tiskárny a děrovače a využívající jeho klávesnice. Start pomocného monitoru se provádí z adresy 0800 H.

Význam příkazových kláves je následující:

[A] (Adr) ... Přivolává uživatelský čítač instrukcí a zobrazuje jej spolu s adresou na displeji. Následuje-li po stlačení "A" vstup hexadecimálních kláves, zobrazí se nová adresa. Vstup končí stiskem některé z příkazových kláves.

[M] (Mem) ... Přivolá režim vstupu dat na právě zobrazenou adresu, v případě opakování se adresa dekrementuje. (Druhá

možnost ukládání dat do paměti je představována klávesou "N", která však adresu inkrementuje !).
Režim zadávání dat indikuje tečka v 6-té pozici displeje. Klávesu "M" lze použít pro prohlížení a vkládání dat ve směru klesajících adres. Dále ji lze použít pro ukončení žádosti o zobrazení registrů, vrchu zásobníku a ukazatele zásobníku.

[N] (NEXT) ... Zápis dat do paměti a její inkrementování. Při zobrazení registrů vybírá další registr.

[C] (CLR) ... Ruší data vložená při předcházejícím stlačení příkazové klávesy. Je-li stlačena při zadávání adresy, znovu zobrazí programový čítač. Jde-li o vstup dat do paměti nebo registru, obnoví jejich předcházející obsah.

[R] (REG) ... Stlačení "R" následované hexadecimální klávesou zobrazí programový čítač spolu s názvem a obsahem registru.

[S] (STEP) ... V režimu krokování (přepínač režimu v poloze STEP) umožní vykonání uživatelského režimu po instrukcích. Změnu PC možno dosáhnout klávesou "A" a vstupem hexadecimálních kláves ukončeném klávesou "S".

[G] (GO) ... Start programu

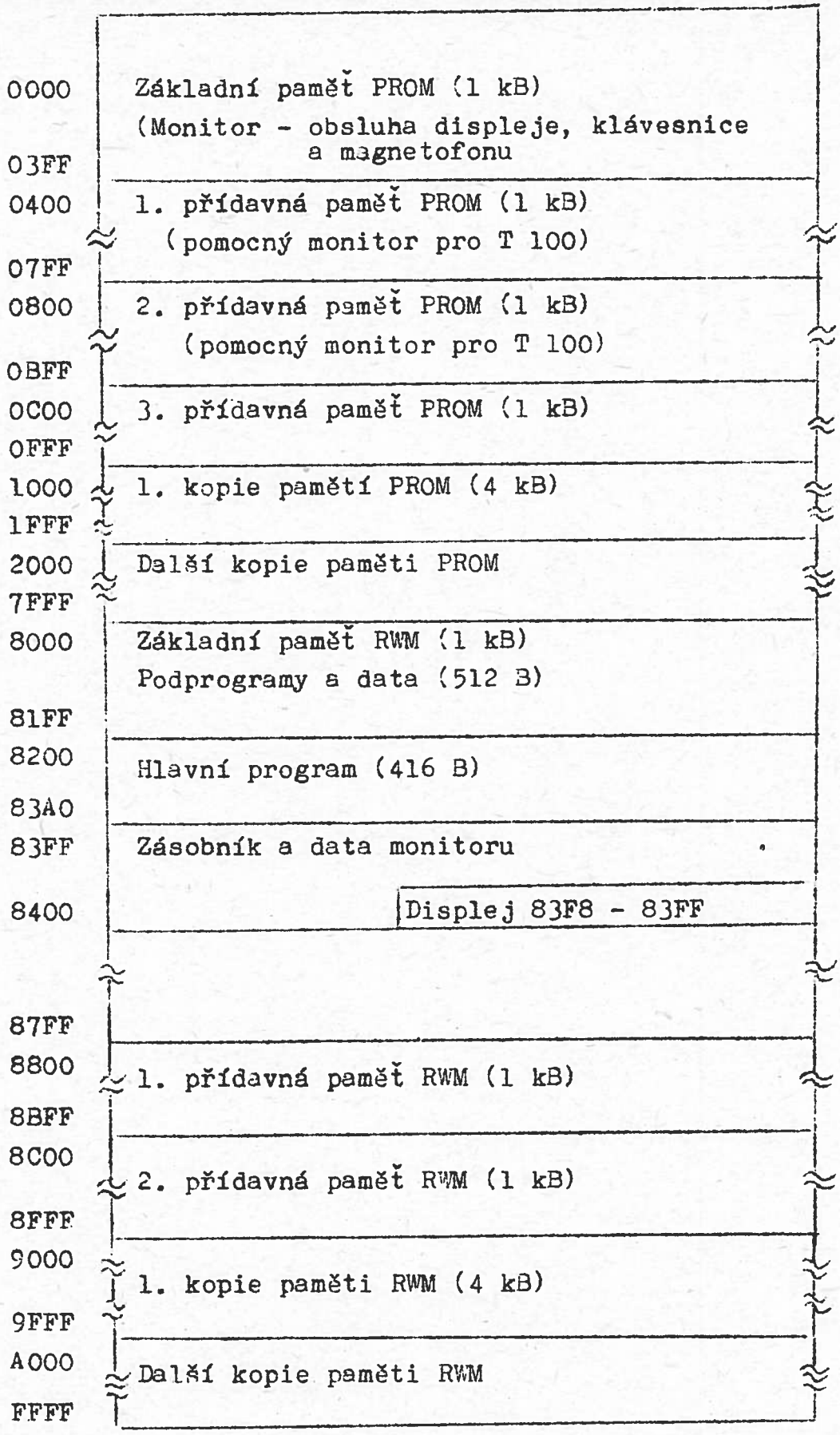
[B] (BRK) ... Zobrazí adresu naposledy aktivovaného ladicího bodu

[RESET] . "Studený start" monitoru na adrese 0000 H

3.1 Popis činnosti ŠMS - VÚVT

3.1.1 Monitor - základní funkce

V první etapě, po přeložení programu do strojového tvaru, je nutno uložit program do příslušné paměťové buňky. Nejprve oznámíme adresu začátku ukládání klávesou "A" a poté následuje čtyřmístná, hexadecimálně vyjádřená, adresa pomocí



Obr. 3.3 Mapa paměti ŠMS - VÚVT

hexa-kláves. Dalším krokem je povolení změny obsahu paměti - klávesa "M". Adresovaná buňka paměti (je současně zobrazena v levé části displeje) se naplní hodnotou zadanou z klávesnice. Zvýšení adresy se dosáhne stlačením klávesy "N". Lze též ukládat do paměti i v opačném pořadí (i když je to nezvyklé) tzn. od vyšších adres směrem k nižším, pomocí krokování klávesou "M". Ukládání přitom kontroluje monitor a je možné:

- při plnění paměťové buňky lze zadat více údajů (chyba při zadání), ale významné jsou pouze poslední dva;
- po zadání nového obsahu paměťové buňky je možné původní obsah obnovit klávesou "C" za předpokladu, že nebyla stisknuta žádná řídicí klávesa;
- monitor kontroluje, zda jsou zadané údaje uloženy, v opačném případě vyvolá na displeji text "Err" (např. při zadání neexistující paměťové buňky).

3.1.2 Ladění programu

Po uložení programu do paměti můžeme kontrolovat správnou funkci programu - program ladit. Při ladění je vhodné nechat program probíhat po instrukcích a kontrolovat obsah paměti a obsah registrů. Tuto funkci dosáhneme zadáním startovací klávesy ("A"), následované 4místnou hexaadresou a postupným krokováním programem klávesou "S". Přitom musí být přepínač STEP-AUTO v poloze STEP. Na displeji se zobrazí adresa následující instrukce a její operační kód. Po každé instrukci je možné si prohlédnout obsah paměti a registrů (A, B, C, D, E, H, L, SP a vrchol zásobníku). Kromě toho je možno průběžně pozorovat, na diodách LED, hodnotu příznaku C a Z. Naposled vybraný registr se zobrazuje po provedení každé instrukce na displeji. Posledně vybranou buňku paměti zobrazíme klávesou "M". Vzhledem k tomu, že tyto funkce jsou dosaženy pomocí přerušení procesoru, je nezbytné dát do každého podprogramu instrukci EI (povolení přerušení). V režimu STEP je možno měnit jak obsahy registrů nebo paměti, tak i je možná změna startovací adresy.

Zobrazení obsahu registru je možné klávesou "R" následo-

vanou klávesou se jménem registru (A, B, C, D, E, F, H, L), klávesou "N" zobrazíme další registr. Párové registry zobrazujeme následovně:

BC	"A", "B", "M"
DE	"A", "D", "M"
HL	"A", "8", "M"
SP	"A", "1", "M"
vrchol zásobníku	"A", "2", "M".

Současně se zobrazením registrového páru se zobrazí obsah paměťové buňky na kterou registr ukazuje. Obsah této buňky je možno též měnit.

Po odladění programu můžeme přistoupit k jeho skutečnému běhu. Režim dosáhneme v poloze AUTO přepínače STEP/AUTO shodným způsobem jako v předchozí části s tím rozdílem, že místo klávesy "S" stiskneme klávesu "G".

Procesor v tomto případě bude trvale vykonávat uživatelský program. Uživatelský program se ukončí při skoku do monitoru na adresu 0020 H (instrukce RST 4) - tzv. "teplý start" nebo skokem na adresu 0000 H - tzv. "studený start". V prvém případě zůstanou uchovány obsahy registrů, stejně tak body přerušení (viz dále), hodnota SP atd. a je možno si je prohlédnout.

V druhém případě (adresa 0000 H) je tento skok ekvivalentní stisknutí klávesy "RESET" a výše uvedené informace nejsou uchovány.

Nevýhodou výše zmíněného ladění programu je skutečnost, že nedůležité části programu musíme pracně překrokovat klávesou "S". Proto ŠMS - VÚVT umožňuje zadání "bodů přerušení" k některým instrukcím. Jestliže potom dojde v programu k jejich aktivaci, přejde ŠMS - VÚVT na teplý start monitoru. Zde je možno uskutečnit podobnou činnost jako v režimu STEP a poté stiskem "S" nebo "G" (podle požadovaného režimu) pokračovat ve vykonávání programu. Přitom je možno zadat počet průchodů bodem přerušení než se činnost programu přeruší.

Zadání bodů přerušení se řídí následujícími pravidly:

- bod zastavení zadáme posloupností "A", 4místná hexa adresa instrukce na které je bod přerušení, "B", dvojmístné číslo v kódu hexa = počet povolených opakovaných průchodů daným bodem;
- při každém průchodu se počet povolených opakování zmenší o jeden (není-li již nula);
- je-li počet průchodů nula, přechází se na teplý start monitoru, bod přerušení se tím však neruší;
- je možno zadat maximálně osm bodů přerušení, každý s maximálně 255 průchody;
- bod přerušení možno zobrazit (adresa a aktuální počet povolených průchodů) stiskem klávesy "B" a další body prohlížet klávesou "N" (jen pokud neběží uživatelský program);
- zrušení bodu přerušení dosáhneme jeho zobrazením a stiskem klávesy "C";
- klávesa RESET, resp. skok na 0000 H, ruší všechny body přerušení;
- vzhledem k tomu, že způsob realizace je obdobný jako u režimu STET, musí být přepínač STEP/AUTO v poloze STEP. Je tedy zřejmé, že při funkci "G" se bude program realizovat podstatně pomaleji než pro případ, kdy by byl přepínač STEP/AUTO v poloze AUTO;
- s ohledem na technickou realizaci mikropočítače ŠMS - VÚVT je použití bodu přerušení účinné až po víceslabikové instrukci, tzn. že neúčinná bude funkce bodu přerušení ukazující na jednoslabikovou instrukci.

3.1.3 Simulace činnosti procesoru pro RST5 a RST6

Skutečný mikropočítač s mikroprocesorem 8080A přechází po RST 5 a RST 6 na instrukci na adrese 0028 H, resp. 0030 H. V ŠMS - VÚVT je tato část paměti (viz obr. 3.3) obsazena pamětí EPROM. Proto monitor zabezpečí přechod na jiné adresy: pro RST 5 se přechází na adresu, která je uložena na adresy

83ECH, 83EDH (dvě slabiky). Při studeném startu sem monitor ukládá hodnotu 8228 H. Pro RST 6 se použije adresa 83EA H a 83EB H, na které je při studeném startu uložena hodnota 8230 H.

3.1.4 Archivace programu na megnetofonové pásce

ŠMS - VÚVT umožňuje přenášet obsah paměti na megnetofonovou pásku běžného magnetofonu, resp. megnetofonu MK 27, tvořícího příslušenství školního mikropočítače (viz obr. 8.2 . Určená oblast paměti se přenese postupně, po jednotlivých bitech, do magnetofonu, namodulována na akustické kmitočty. Před nahráním nahrajeme krátký komentář (název, funkce programu, datum, atd.). Poté propojíme ŠMS - VÚVT a MK 27.

Posloupnost ukládání je následující:

"A", adresa začátku přenosu, "M", "A", adresa konce přenosu, "B", "A", "0371". V této fázi generuje mikropočítač nemodulovaný tón určený pro nastavení úrovně nahrání. Poté stiskneme klávesu "G". Přenos je indikován zhasnutím displeje, ukončení přenosu jeho rozsvícením.

Při zpětném čtení dat z magnetofonu nejprve ručně nastavíme začátek záznamu na magnetofon a poté aktivujeme příjem dat posloupností: "A", adresa začátku přenosu, "M", "A", "03AE" a zapneme magnetofon. Po skončení slovního komentáře, při přehrávání modulačního tónu, stiskneme klávesu "G". Mikropočítač kontroluje bezchybnost přenosu dat pomocí střadače, ve kterém musí být po ukončení přenosu nula. V opačném případě rozsvítí nápis "Err".

Při přenosu dat musí být přepínač STEP/AUTO v poloze AUTO !

3.1.5 Monitor - rozšiřující funkce

Jak již bylo uvedeno dříve, je použitá verze školního mikropočítače ŠMS - VÚVT doplněna dalším pomocným monitorem.

Jeho úkolem je rozšíření základních funkcí o ovládání dálnopisu T 100 a operace s bloky dat.

Pomocný monitor uvedeme v činnost, po propojení dálnopisu přes konektor a sběrnici laboratoře dle pokynů vedoucího cvičení, vyvoláním startovací adresy 0800 H. (Posloupnost "A", 0800 H, "G").

Monitor se ohlásí (na T 100):

pomocný monitor pro sms s tty
příkazy: d, f, m, s, w, r, b

Po vytisknutí znaku "." očekává zadání příkazu z klávesnice T 100.

Při vstupu do monitoru se uchová obsah všech registrů, vyjma uživatelského PC.

3.1.6 Příkazy pomocného monitoru

Příkaz D ⟨adr.1⟩ ␣ ⟨adr.2⟩ ⟨CR⟩ ⟨LF⟩

Vytiskneme obsah paměti od adresy ⟨adr.1⟩ do adresy ⟨adr.2⟩. Obě adresy musí být zadány jako dvě hexadecimální čísla, oddělená mezerou a zakončena návratem vozíku a novým řádkem.

Příkaz F ⟨adr.1⟩␣⟨adr.2⟩␣⟨konst.⟩ ⟨CR⟩⟨LF⟩

Obsah paměti RWM od adresy ⟨adr.1⟩ do adresy ⟨adr.2⟩ se naplní konstantou ⟨konst⟩

Příkaz M ⟨adr.1⟩␣⟨adr.2⟩␣⟨adr.3⟩⟨CR⟩⟨LF⟩

Oblast paměti od ⟨adr.1⟩ do ⟨adr.2⟩ se přesune do paměti RWM počínaje adresou ⟨adr.3⟩

Příkaz S ⟨adr.⟩ ␣ xx-xx ... ␣⟨CR⟩⟨LF⟩

Zobrazuje a mění se obsah paměti od zadané adresy. Po zapsání S ⟨adr⟩ se po stisknutí klávesy "mezera" vytiskne obsah adresy s nápovědným znakem "-". Zadáme-li z klávesnice (T 100) nový obsah zobrazené paměťové buň-

ky, bude automaticky uložen při přechodu na další buňku paměti stiskem klávesy "mezera". Při stisknutí klávesy "mezera" se vždy vytiskne obsah následující adresy. Činnost příkazu zakončuje <CR><LF>

Příkaz W <adr.1> u<adr.2><CR><LF>

Přenese obsah paměti od adr.1 do adr.2 na děrnou pásku. Při jeho aktivaci postupujeme dle nápovědního textu, který vyšle monitor na T 100.

Příkaz R

Čte a zavádí do paměti data z děrné pásky vyděrované dříve příkazem W.
Zapnutí a vypnutí čtečky se provede dle nápovědného textu, který monitor vytiskne při aktivaci příkazu.

Příkaz B

Provede návrat do základního monitoru, obnoví obsah uživatelských registrů, s výjimkou PC, vynuluje se displej a očekávají se příkazy základního monitoru z klávesnice ŠMS - VÚVT.

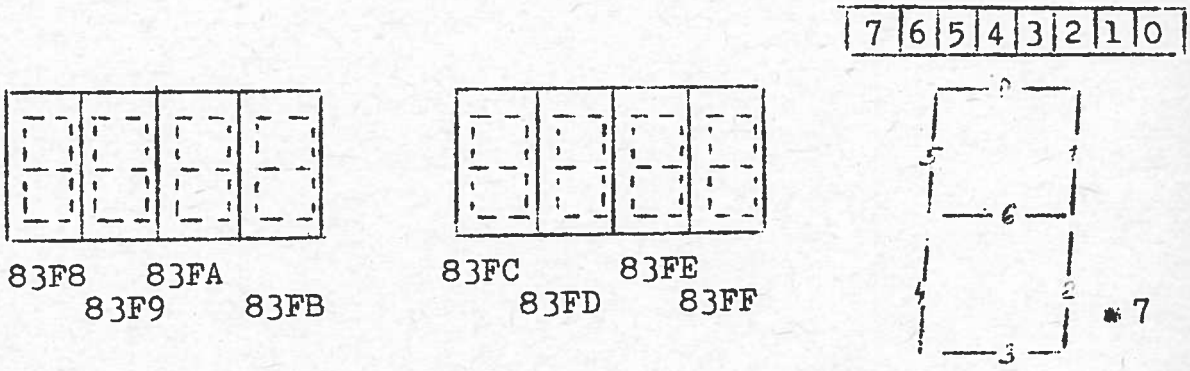
Poznámka: Pokud se monitor ohlásí černě vytisknutým znakem "zvonek" před nápovědným znakem ".", došlo při činnosti k nějaké chybě (např. neznámý znak, chyba na zaváděcí pásce). Při nalezení chyby na pásce při příkazu "R" se zavádění programu do paměti okamžitě přeruší, čeká se na konec pásky a po dalším spuštění se monitor ohlásí znakem "zvonek" před znakem ".".

3.2 Technické prostředky ŠMS - VÚVT

Základní údaje o technickém uspořádání (hardware) ŠMS - VÚVT byly uvedeny v úvodu kap. 3. V dalším si všimneme jednotlivých částí mikropočítače.

3.2.1 Displej

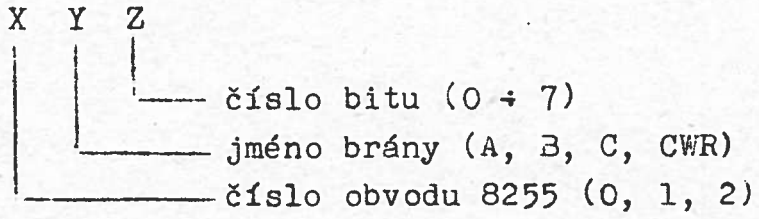
Školní mikropočítač používá 8-místný sedmisegmentový displej zobrazující v hexadecimálním kódu. Pro zobrazení dat se používá přímý přístup do paměti (DMA). Data, která se mají vyslat na displej, jsou zapsána do příslušných paměťových buněk (viz obr. 3.4) a zobrazena.



Obr. 3.4 Adresy a pozice bitů v displeji ŠMS - VÚVT

3.2.2 Vstupní/výstupní obvody

Školní mikropočítač používá 3 obvody 8255. Technické řešení je provedeno dekodéry pro výběr jednotlivých obvodů 8255. Brány 8255 se adresují přímo bity A1 a A0 adresy. Pro označení čísla V/V obvodu, jeho brány a jednotlivých bitů je použita následující symbolika:



Příklad: 2C2 ... bit 2 brány C druhého obvodu 8255.

Adresy a přidělení bran V/V obvodu jsou uvedeny v tab. 3.1 a tab. 3.2, programování obvodů 8255 je uvedeno v kap. 2., odst 2.8.4.

Při stisknutí klávesy "RESET" se všechny brány obvodů 8255 nastaví jako vstupní v režimu 0. Monitor inicializuje

poté obvod 8255 č. 0, (0A, 0B - vstup, 0C - výstup, režim 0), a tím umožmívstup z klávesnice, výstup na displej atd. Druhé dva obvody 8255 monitor neprogramuje a pokud nepotřebujeme změnit jejich režim, nemusíme je programovat.

Pro správnou činnost přerušovacího systému se však musí brána 2B naprogramovat pro vstup a brána 2C pro výstup (v režimu 0). Brána 1B, vyhrazená pro analogové vstupy/výstupy, se může použít v režimu 0 a 1; pouze u automatického režimu (viz dále) A/D se použije režim 0. Druh provozu (režim) 1 a 2 jsou vhodné pro bránu 1A. Přerušení, generované v těchto režimech, je přímo použité v přerušovacím systému ŠMS - VÚVT. Konfigurace pro druh provozu 0 jsou uvedeny v tab. 3.2. Tyto konfigurace je možno kombinovat s obvodem 8255 1 režimu 1 a 2.

Příklad programování obvodů 8255:

```
3E80  MVI A, 80H  ; řídicí slabika 8255
                  ; A, B, C = výstup
D307  OUT  CNT1   ; vyslání řídicí slabiky
                  ; na CNT1
3E92  MVI A  92H  ; řídicí slabika;
                  ; A, B = vstup
                  ; C = výstup
D30F  OUT  CNT2   ; vyslání řídicí slabiky
                  ; na CNT2
```

Adresa	Jméno	Funkce	Bity bran 0B, 0C, 1C
00	0A	Vstup klávesnice	0B0 - vstup k modemu magnetofonu
01	0B	Nepřidělené s výjimkou 0B0	0C0 - výstup na modem megnetofonu
02	0C	Monitor - přerušení - viz sloupec napravo	0C1 - povolení monitor. přerušení
03	CNTØ	Řídicí brána 8255 0	0C2 - výstup indikace CY
04	1A	Budič a indikace LED	0C3 - výstup indikace Z
05	1B	A/D a D/A převodník	0C4 - povolení kláves 0 ÷ 7

Adresa	Jméno	Funkce	Bity bran 0B, 0C, 1C
06	1C	Klávesy - viz sloupec napravo	0C5 - povolení kláves 8 ÷ F
07	CNT1	Řídicí brána 8255 1	0C6 - povolení příkazových kláves
0C	2A	Nepřiděleno	0C7 - povolení zobrazení
0D	2B	Stavová slabika přerušení	1C0 - řízení A/D ("1" = aut.A/D)
0E	2C	Povolovací slabika přerušení	1C1 - povolení buzení motoru
0F	CNT2	Řídicí brána 8255 2	1C2 - nepřiděleno
14	TIMØ	Časovač 0	1C3 - přerušení (povolované 2C6)
15	TIM1	Časovač 1	1C4 ÷ 7 nepřiděleno
16	TIM2	Časovač 2	
17	TIMCT	Řídicí registr časovače	

Tabulka 3.1 Adresy a přidělení jednotlivých bran obvodů V/V

Poznámka: V tabulce 3.1 je ponecháno označení řídicího registru obvodů 8255 a 8253 shodné s dokumentací ŠMS - - VÚVT. Porovnej s odst. 2.8.4.

Po naprogramování obvodů 8255 je možno z jednotlivých bran číst nebo do nich zapisovat instrukcemi IN a OUT.

Příklad:

```
DB0C    IN 2A          ; čtení z brány 2A
3F80    MVI A, 80H     ; zápis konst 80H
                       ; do brány 1A
D3      OUT 1A         ;
3E01    MVI A, 01H     ; nastavení 1C0 = 1
D307    OUT CNT1       ;
```

Řídicí slabika	Brána A	Brána B	Brána C0-C3	Brána C4-C7	8255 0	8255 1	8255 2
80	OUT	OUT	OUT	OUT		D/A	
81	OUT	OUT	IN	OUT		+	
82	OUT	IN	OUT	OUT		A/D	*
83	OUT	IN	IN	OUT		A/D	
88	OUT	OUT	OUT	IN		D/A	
89	OUT	OUT	IN	IN		+	
8A	OUT	IN	OUT	IN		A/D	
8B	OUT	IN	IN	IN		A/D	
90	IN	OUT	OUT	OUT		D/A	
91	IN	OUT	IN	OUT		+	
92	IN	IN	OUT	OUT	*	A/D	*
93	IN	IN	IN	OUT		A/D	
98	IN	OUT	OUT	IN		D/A	
99	IN	OUT	IN	IN		+	
9A	IN	IN	OUT	IN		A/D	
9B	IN	IN	IN	IN		A/D	

Poznámka: + ... zakázaná konfigurace

* ... standardní operace (použijte pouze tuto operaci)

Tabulka 3.2 Řídicí slabiky pro 8255 v ŠMS - VÚVT (druh provozu 0)

3.2.3 Časovač

Školní mikropočítač obsahuje programovatelný časovač 8253. Programování obvodu 8253 bylo uvedeno v kap.2., odst. 2.8.5. V ŠMS - VÚVT je použita adresace dle tab.3.3, přehled řízení v tab.3.4.

Časovač	adresa
T0	14
T1	15
T2	16
TIMCT (řídicí registr CWR)	17

Tab. 3.3 Adresy 8253 v ŠMS - VÚVT

Typickou činnost časovače ukazuje následující příklad programu:

```
MVI  A, 34H   ; programuj čítač 0 pro
OUT  TIMCT    ; režim 2, s dvouslabikovým
MVI  A, 20H   ; čtením/nastavením registru
OUT  T0       ; počáteční hodnoty a binárním
MVI  A, 0FH   ; počítáním čítače T0
OUT  T0       ; Nastav počítací interval = 0F20H
```

Poznámka: 1) Řídicí slabiky v tabulkách platí pro binární počítání

2) Řídicí slabiky z tab.3.4 zapisujeme do řídicího registru TIMCT na adrese 17 (viz.tab.3.3).

Čítač	Činnost	Režim					
		0	1	2	3	4	5
0	Vzorkuj	00	00	00	00	00	00
	Čti/nastav pouze LSB	10	12	14	16	18	1A
	Čti/nastav pouze MSB	20	22	24	26	28	2B
	Čti/nastav obě slabiky (LSB jako první)	30	32	34	36	38	3A
1	Vzorkuj	40	40	40	40	40	40
	Čti/nastav pouze LSB	50	52	54	56	58	5A
	Čti/nastav pouze MSB	60	62	64	66	68	6A
	Čti/nastav obě slabiky	70	72	74	76	78	7A
2	Vzorkuj	80	80	80	80	80	80
	Čti/nastav pouze LSB	90	92	94	96	98	9A
	Čti/nastav pouze MSB	A0	A2	A4	A6	A8	AA
	Čti/nastav obě slabiky	B0	B2	B4	B6	B8	BA

Tab. 3.4 Řídicí slabiky 8253 v ŠMS - VÚVT

Čítače v ŠMS - VÚVT může použít úplně univerzálně neboť nemají v systému pevně vymezenou funkci. Jejich jednotlivé vstupy jsou napojeny na hodiny procesoru (2 MHz) Ø 2 TTL, ale je možno je pomocí propojek přepojit jinam, případně i na externí taktovací pulsy (max. 2 MHz). Hradlovací vstupy čítačů 0 a 1 jsou připojeny na úroveň log. 1 (povolení), lze je však též řídit externě, neboť jsou vyvedeny na kontaktové pole. Čítač 2 má hradlovací vstup zapojený v obvodech převodníku A/D. Chceme-li je použít pro jiné účely, musíme dosáhnout povolení nastavením hradlovacího vstupu na log.1. Toho dosáhneme nulováním brány 1C0. Výstupy čítačů jsou jinak vyvedené na kontaktová pole, zároveň však vstupují i do přerušovacího systému, kde jsou zdrojem žádostí o přerušení. Čítač 0 a 1 mají v přerušovacím systému záchytné registry žádostí o přerušení schopné uchovat i velmi krátce trvající žádost o přerušení (0,5 /us). Čítač 2 nemá záchytný registr; jeho výstup je přímo hradlovaný vlastním povolovacím signálem 2B3 a dále sečten do společné přerušovací žádosti INT. Zde se žádost o přerušení od čítače 2 může uplatnit jen po dobu jejího trvání.

3.2.4 Převodníky A/D a D/A

Převodník ve školním mikropočítači používá osmibitový integrovaný převodník D/A typu ZN425E ve spojení s bránou 1B. Výstupní napětí převodníku D/A nabývá (v souladu s hodnotou čísla na bráně 1B) příslušné hodnoty z 256 diskrétních hodnot napětí v rozsahu (0 mV - 2550 mV). Kalibraci umožňuje potenciometr ANALOG OUT (obr.3.2), zapojený ve zpětné vazbě výstupního operačního zesilovače typu LM 324. Celkový výstupní rozsah analogového napětí je v rozmezí 0,731 - 3,778 V (cca 3 - 15 mV/bit).

Při realizaci převodníku A/D se výstupní napětí převodníku D/A (ZN425E) porovnává s napětím vstupním, které je (po oddělení vstupním operačním zesilovačem) úměrné nastavení potenciometru ANALOG IN. Rozsah vstupního napětí je limitován napájecím napětím operačního zesilovače a je

v rozsahu 0 - 3,8 V. Blokové schéma je znázorněno na obr. 3.5.

3.2.5 Výkonové výstupy

Výstupy připojené přes bránu 1C1 (viz obr.3.6) jsou určené pro buzení výkonových zátěží typu stejnosměrný motorek, relé atd., které vyžadují proudy řádu max. jednotek A. V ŠMS - VÚVT je výkonový výstup oddělen optočlenem a pro regulaci jsou určeny signály MOT CTL+, MOT CTL-, MOT SUP+. Signály určují buzení optočlenu ovládajícího vlastní výkonový prvek (tranzistor KU 612). Signál MOT CTL- lze ovládat buď programově (1C1), nebo externě z kontaktového pole (viz obr.3.2 a obr.3.5). Pro signál MOT CTL+ lze použít napětí v rozsahu 0 - 5 V. Pro napájení tranzistoru (MOT SUP+) je možné použít vnitřního napájecího napětí + 5 V nebo + 12 V nebo externí zdroj.

Ostatní výkonové výstupy (budič 7406 brány 1A) se budí signály přes tuto bránu. Stavy jsou indikovány diodami LED (D3 - D10). Budiče 7406 nemají zapojeny kolektorové odpory, a proto chceme-li je použít pro buzení nějaké zátěže, musíme odpory připojit a připojit je na napětí. Maximální napětí je 30 V, zatížitelnost 40 mA.

Poznámka: V použité konfiguraci ŠMS - VÚVT je vstup 1C1 propojen na interfejs dálnopisu T 100 a proto před aplikací je nutno zrušit příslušnou spojku.

3.2.6 Přerušovací systém

a) Přerušení monitoru

Úkolem monitorových přerušení je umožnit uživateli prohlížení paměti a registrů po vykonání každé instrukce. Povolení monitorových přerušení lze ovládat ručně (přepínač AUTO/STEP) nebo programově (V/V signálem OC1). Zákaz monitorových přerušení je indikován zhasnutou diodou LED ozn. v obr.3.2 MON. Vlastností použitého monitoru je, že po povolení přerušení (EI) se monitorové pře-

rušení nemůže indikovat dříve než se vykoná víceslabiková instrukce (viz odst.3.1.2).

b) Uživatelská přerušení

Tato přerušení lze rozdělit na systémová a externí. Zdrojem systémových přerušení jsou signály generované ŠMS - VÚVT (časovač atd.). Zdroje externích přerušení jsou obvykle mimo ŠMS - VÚVT a je nutno je do příslušného vstupního konektoru přivést.

O přerušení generovaných časovačem již pojednává odst.3.2.3. S výjimkou 1C3 se všechny zdroje přerušení vedou na bránu 2B jako vstupy, a proto je je možno považovat za stavovou slabiku přerušení. Jestliže dojde k přerušení, může program tuto bránu číst a určit zdroj přerušení. Obvykle je nežádoucí, aby se naráz generovalo více přerušení, a proto se všechny přerušení hradlují povolovacími signály, které je možno programově ovládat na výstupní bráně 2C (povolovací slabika přerušení). Procesor bude akceptovat přerušení, jestliže dojde k přerušovací události a zároveň bude její povolovací bit na bráně 2C nastaven ("1"). Bližší viz tab.3.5:

Zdroje přerušení	Povolovací bit	Stavový bit
Čítač 0 (kl.obvod)	2C0	2B0
Čítač 1 (kl.obvod)	2C1	2B1
Čítač 2 (bez kl.obvodu)	2C2	2B2
A/D komparátor	2C3	2B3
EXT 4 (kl.obvod)	2C4	2B4
EXT 5 (kl.obvod)	2C5	2B5
Brána 1C3	2C6	1C3
Všeobecné nepovolení	2C7	
EXT 4 (přímé - nepřímé)		2B6
EXT 5 (přímé - nepřímé)		2B7

Tab.3.5 Přerušení ŠMS - VÚVT

Na bráně 2C7 je signál všeobecného nepovolení pro všechna uživatelská přerušení. Bude-li nastaven ("1"), může být přerušení vyvoláno pouze monitorem. Při systémovém nulování (RESET) se nastaví všechny ovcody (viz odst.3.2.2) jako vstupní v režimu 0. Signálové vodiče však budou ve třetím stavu a logika přerušení vyhodnotí 2C7 jako "1", a proto všechna uživatelská přerušení zablokuje. Jestliže se brána 2C naprogramuje (v režimu 0) jako výstupní, automaticky se nulují všechny bity. Všeobecný zákaz přerušení přestane platit, ale jednotlivá přerušení jsou nyní blokována individuálně na 2C0 a 2C6. Pro jejich povolení je třeba setovat všechny odpovídající bity v bráně 2C.

Žádné uživatelské přerušení tedy nemůže vzniknout, dokud není 8255 č. 2 naprogramován a nastaveny bity brány 2C.

Po přijetí přerušení procesorem se uskuteční obsluha přerušení, která zahrnuje i příkaz na nulování záchytného klopného obvodu zdroje přerušení. Pro tento účel se používají instrukce nastavení/nulování individuálního bitu brány 2C. Při zápisu slabiky 2C se stav klopného obvodu nemění. Přehled podává tab.3.6.

Zdroj přerušení	Stavová slabika přerušení na bráně 2B (viz poznámka 2)	Příkazová slabika pro instr. OUT CNT2 (viz poznámka 1)		
	Binárně	Hex	Zákaz	Povolení
Čítač 0	0 X X X X X X 1	01	00	01
Čítač 1	0 X X X X X 1 X	02	02	03
Čítač 2	0 X X X X 1 X X	04	04	05
A/D kompar.	0 X X X 1 X X X	08	06	07
EXT 4	0 X X 1 X X X X	10	08	09
EXT 5	0 X 1 X X X X X	20	0A	0B
Brána 1C3	(viz poznámka 3)		0C	0D

Tab.3.6 Stavové slabiky a příkazové slabiky pro povolení/ /zákaz přerušení

Pozn. 1 Příkazová slabika pro povolení/zákaz přerušení se musí zapsat do CNT2, aby se jí vynulovaly klopné obvody počítačů 0, 1, EXT 4 nebo EXT 5. Příkaz pro povolení/zákaz přerušení pro A/D komparátor nuluje přerušení pouze při automatickém převodu A/D.

Pozn. 2 V hexadecimálních hodnotách stavových slabik jsou ostatní bity (kromě určujícího) doplněny nulami, což ve skutečnosti nemusí být pravda. Používáme je však na testování individuálního zdroje přerušení. Výsledek pro instr. ANI je vždy nula, jestliže testované přerušení nebylo přijato.

Pozn. 3 Přerušení od 1C3 není zahrnuto do stavové slabiky. Čteme je jako

X X X X 1 X X X

pomocí instrukce IN PORT 1C. Nulujeme jej čtením brány 1A ve strobovaném výstupním režimu (režim 1 a 2). Bránu 1C můžeme též nastavit (nulovat) zapsáním konstanty 06H (07H) do CNT1. Povolení přerušení pro bránu 1C3 nulujeme (nastavujeme) zápisem 0CH (0DH) do CNT2; data na bráně 1C3 se přitom nemění.

3.2.7 Modem pro magnetofon

Modem pro megnetofon je určen pro snímání dat nebo záznam dat na běžný magnetofon (viz odst. 3.1). Pro archivaci je použita frekvenční modulace, při které je úroveň log. 1 zaznamenána několika periodami frekvence 2 400 Hz a pro záznam log. 0 je použita frekvence 1 200 Hz. Modem pracuje až do rychlosti 500 Bd, ale s ohledem na megnetofon je použito nastavení přenosové rychlosti na 300 Bd. Na kazetu C60 je možno zaznamenat cca 90 kB.

3.2.8 Modem pro připojení dálnopisu T 100

V použité konfiguraci ŠMS - VÚVT je vestavěna přídavná proudová smyčka - modem pro T 100. Oddělení T 100 od ŠMS - - VÚVT je zajištěno optočlenem. Pro vysílání dat je použit

1C1, data jsou snímána na 1A0. Dálnopis se připojuje do zásuvky dle obr. 3.2. Potřebujeme-li použít 1C1 a 1A0 pro jiné účely, musíme zrušit příslušné spojky na desce.

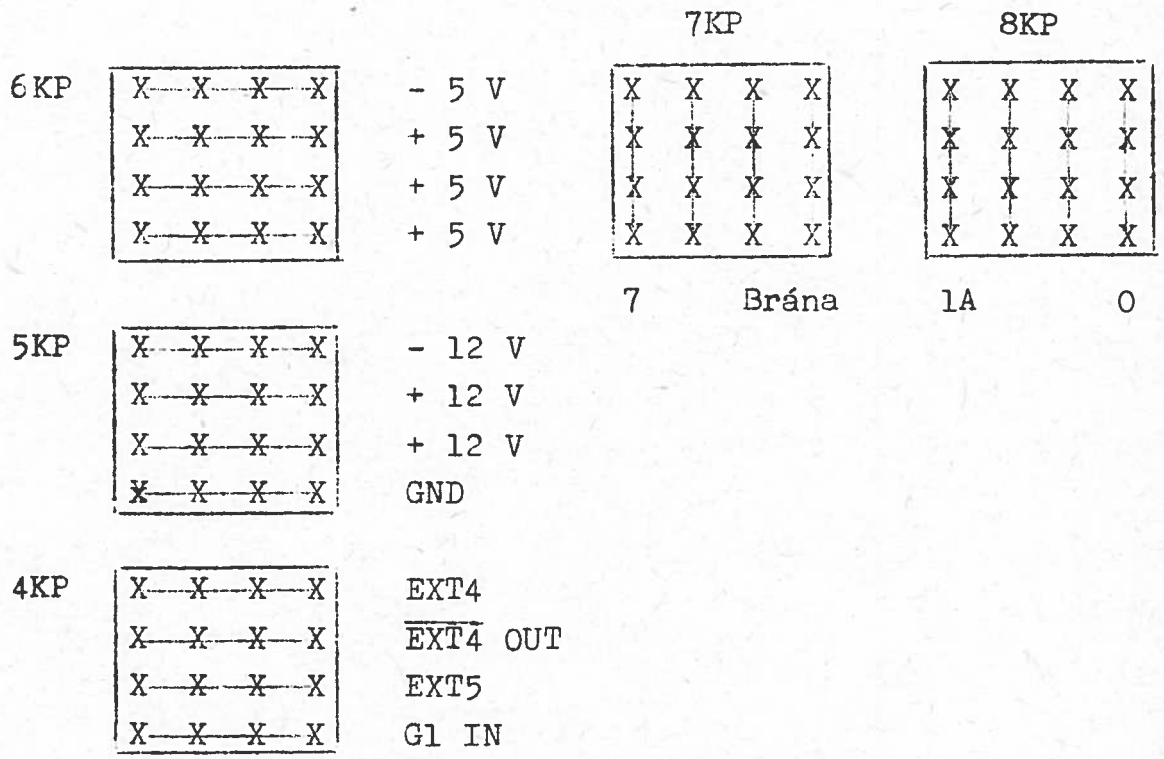
3.2.9 Interfejs RS 232C

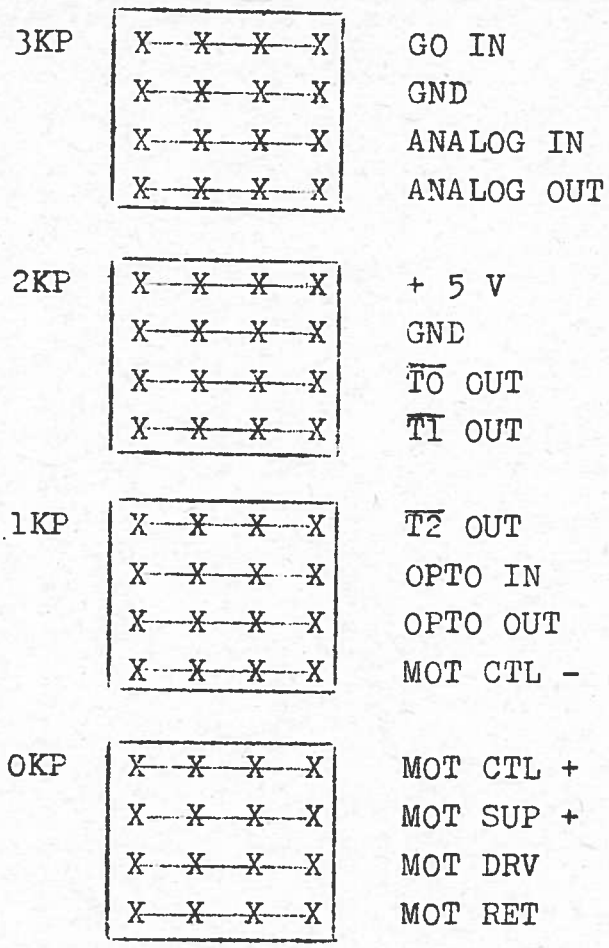
Interfejs RS 232C je určen pro připojení terminálů typu SM 7202. Bližší viz lit. /21/.

3.2.10 Systémový konektor

Na systémový 62 pólový konektor jsou vyvedeny systémové sběrnice a další signály umožňující rozšiřování systému. Ve cvičeních bude systémový konektor zapojen na interfejs umožňující spolupráci ŠMS - VÚVT s řídicím počítačem laboratoře. Při jeho použití postupujeme dle pokynů vedoucího cvičení.

3.2.11 Přiřazení konektorů ŠMS - VÚVT





Obr. 3.7 Rozložení signálů v kontaktových polích 0KP - 8KP na ŠMS - VÚVT

4. Příklady programů

Při návrhu programů postupujeme dle zásad uvedených v odst. 1.5, 1.6 a 2.9. Při zavádění jednotlivých programů do ŠMS - VÚVT využíváme poznatků z kap. 3. Základní programové moduly jsou uvedeny v odst. 4.7. Vyčerpávající přehled základních programů je uveden v lit. /26/, překlad jejich podstatné části v kap. 5 lit. /23/.

V dalším se zaměříme na realizaci programů vycházejících přímo z možností školního mikropočítače ŠMS - VÚVT (lit. /22/) a používajícího, co možná nejvíce, vhodných podprogramů monitoru školního mikropočítače a zejména jeho periferií.

4.1 Přehled podprogramů monitoru ŠMS - VÚVT

Využívání vhodných podprogramů monitoru můžeme značně zjednodušit aplikační programy, které budeme na školním mikropočítači řešit. Soubor podprogramů monitoru můžeme rozdělit do čtyř skupin:

- podprogramy pro ovládání vstupu z klávesnice
- podprogramy pro ovládání výstupu na displej
- podprogramy pro realizaci časového zpoždění
- pomocné podprogramy.

Přehled použitelných podprogramů podává tabulka 4.1:

Jméno	Funkce	Volací adresa
CLEAR	Mazání celého displeje	0287H
CLRGT	Mazání poloviny displeje	0282H
CLRLP	Mazání paměti	028CH
DELAY	Milisekundové zpoždění	0236H
DELYA	Nastavitelné zpoždění	0238H
DBYTE	Zobrazení slabiky	0295H
DBY2	Zobrazení slabiky	0298H
DISPR	Zobrazení číslice	02A6H
DMEN	Zobrazení slabiky	0294H
DMWD	Zobrazení adresy a dat	02CEH
DWDZ	Zobrazení obsahu HL	02D4H
DWORD	Zobrazení obsahu HL	02D1H
DYPC	Zobrazení PC a instrukce	02CBH
ENTBY	Vstup a zobrazení slabiky	0336H
ENTWD	Vstup a zobrazení dvou slabik	0346H
ERROR	Výpis hlášení Err na displeji	00BCH
GETKY	Vstup klávesy	023DH
MENAB	Povolení monitoru a zobrazení	0222H
SCAN	Jednorázový test klávesnice	0257H
SHLRC	Posuv HL a CY	022FH
SHLRT	Posuv HL	022DH
SHLRZ	Posuv HL včetně Z	022EH

Tab. 4.1 Přehled podprogramů monitoru ŠMS - VÚVT

4.1.1 Podprogramy pro klávesnici a displej

Podprogramy pro vstup z klávesnice jsou: SCAN, GETKY, ENTBY, ENTWD. P o d p r o g r a m SCAM prohlédne všechny klávesy a zjistí, zda byla některá stiskruta. Pokud ano, uloží její kód do střadače. Klávesy 0 ÷ F mají kód 0 ÷ FH, řídicí (červené) klávesy kód 10H ÷ 17H. Jestliže nebyla stisknuta žádná klávesa, nastaví se CY = 0. P o d p r o g r a m CETKY pracuje obdobně jako SCAN s tím rozdílem, že se čeká na stisknutí některé klávesy. Tehdy uloží její kód do střadače a příznak CY = 0 znamená, že byla stisknuta některá z řídicích kláves a ne hexadecimální. Oba podprogramy jsou vhodné pro kód hexa z klávesnice, pro vstup binární hodnoty jsou vhodné podprogramy ENTBY a ENTWD. P o d p r o g r a m ENTBY přejímá jednu slabiku z klávesnice (při stisku více hexa kláves jsou platné pouze poslední dvě !). Vstup se ukončí stisknutím některé řídicí klávesy. Zadaná slabika je uložena v registru L, kód řídicí klávesy ve střadači a v registru H je slabika odpovídající stisknutým dvěma klávesám, které předcházely před posledními dvěma klávesami. Stisknuté klávesy jsou průběžně zobrazeny ve dvou pravých pozicích displeje. Příznak Z = 1 znamená, že nebyla stisknuta žádná hexa klávesa, tedy, že byla zadána pouze klávesa řídicí. P o d p r o g r a m ENTWD je určen pro vstup dvouslabikového slova. Je obdobou podprogramu ENTBY a liší se pouze způsobem zobrazení. Po stlačení čtyř kláves se v pravé části displeje nezobrazí zadaná hodnota, ale obsah paměťové buňky, která je zadanými klávesami adresována. Zadané slovo (dvě slabiky) se průběžně zobrazuje v levé části displeje. Ostatní je shodné s ENTBY. Podprogramy ENTBY a ENTWD ukládají v registru D současně číslo odpovídající počtu stisknutých hexa kláves. P o d p r o g r a m DISPR je určen pro zobrazení jedné hexa číslice. Kód číslice má být uložen v nižších čtyřech bitech střadače, v registrech DE má být uložena žádaná pozice displeje (viz obr. 3.4). Obsah registrů DE se v podprogramu DISPR dekrementuje, a proto opakované použití podprogramu aktivuje další levé pozice displeje. P o d p r o g r a m DBY2 zobrazuje slabiku ulože-

nou ve střadači na pozici displeje, kterou určuje obsah registrů DE. Od podprogramu DBY2 jsou odvozeny DBYTE (obsah střadače se zobrazí na dvou pravých pozicích displeje) a DMEM (na posledních dvou pozicích displeje se zobrazí obsah paměťové buňky adresované registry HL). P o d p r o g r a m DWD2 zobrazuje obsah registrů HL na pozicích displeje shodně s DBY2. Na tyto základní podprogramy navazují další:

```
02CB  2AE683  DYPC:   LHLD 83E6H     ; naplň HL z paměti
                                        83E6H
02CE  CD9402  DMWD:   CALL DMEM      ; volej DMEM
02DI  11FB83  DWORD:  LXI  D,83FBH   ; nastav displej
02D4  7D      DWD2:   MOV  A,L       ; začíná DWD2
02D5  CD9802          CALL DBY2      ; zobraz nižší slabiku
                                       :
                                       :
                                       :
                      RET            ; návrat
```

Z výše uvedeného je zřejmé, že DWORD zobrazuje obsah HL v levých čtyřech pozicích displeje, DMWD navíc zobrazí obsah paměťové buňky adresované HL a DYPC zobrazí dvě slabiky z adresy 83E6H a též obsah buňky adresované těmito dvěma slabikami. Stejně jako v předchozím se obsah DE dekrementuje o 4.

Zde opět připomeneme, že displej je aktivován pomocí DMA a je uložen jako paměťová buňky na adresách dle obr. 3.4.

Příklad 4.1: Využití podprogramů SCAN a DBYTE k zobrazení kódu právě stisknuté klávesy.

```
                      ORG 8200H
8200  CD5702  CP1:    CALL SCAN      ;začátek
82Ø3  CD9502          CALL DBYTE     ;zobrazení kódu klávesy
82Ø6  C30082          JMP  CP1       ;návrat
```

Příklad 4.2: Využití podprogramů SCAN a DWORD pro zobrazení kódu stisknuté klávesy včetně zobrazení příznaků. V levé části displeje se zobrazuje obsah

střadače a příznaku.

```
                        ORG 8200H
8200  CD5702  CP2:   CALL SCAN     ;začátek
8203  F5             PUSH PSW     ;úschova střadače a
                                   ;příznaků
8204  E1             POP  H       ;naplnění HL uschova-
                                   ;nými daty
8205  CDD102         CALL DWORD   ;zobrazení HL
8208  C30082         JMP  CP2     ;návrat
```

Příklad 4.3: Sestavte program pro postupné rozsvěcení segmentů displeje směrem zleva doprava.

```
                        ORG 8200H
8200  21F883  CP4:   LXI  H,83F8H ;nastavení levé pozi-
                                   ;ce displeje
8203  3E01            MVI  A,1H    ;nastavení výchozího
                                   ;bitu
8205  77        ULOZ: MOV  M,A     ;uložení dat
8206  37              STC          ;nastavení CY = 1
8207  17              RAL          ;posuv včetně CY
8205  F5              PUSH PSW     ;uchování dat
8209  CD3D02          CALL GETKY   ;čekání na stlačení
                                   ;klávesy
820C  F1              POP  PSW     ;obnovení dat
820D  23              INX  H       ;další pozice displeje
820E  C30582          JMP  ULOZ    ;návrat na úschovu dat
                                   ;střadače
```

Příklad 4.4: Zobrazení písmena "U" na displeji

```
                        ORG 8200H
8200  3E3E    CP5:    MVI  A,3EH   ;naplnění střadače
                                   ;konstantou kodu U
8202  32FF83          STA  83FFH   ;vyslání dat na displej
8205  76              HLT          ;zastavení procesoru
```

Pro mazání obsahu displeje lze použít podprogramy CLRGT, CLEAR a CLRP. Podprogram CLRLP nuluje paměťové buňky od adresy uložené v HL směrem k nižším adresám, přitom v registru B

je počet nulovaných buněk (max. 256). Podprogram CLRLP lze použít všeobecně pro mazání obsahu paměti. Podprogram CLEAR nuluje celý displej a podprogram CLRGT pouze čtyři pravé pozice. Bližší je zřejmé z komentovaného výpisu monitoru:

```
0282  0604      CLRGT: MVI  B,4H      ;nastavení N = 4
0284  C38902             JMP  0289H   ;přeskok další
                                       ;instrukce
0287  0608      CLEAR: MVI  D,8       ;nastavení N = 8
0289  21FF83             LXI  H,83FFA ;pravá pozice displeje
028C  3600      CLRLP: MVI  M,0       ;nulování buňky paměti
028E  2B                  DCX  H       ;snížení ukazatele
                                       ;paměti
028F  05                  DCR  B       ;parametr cyklu dekre-
                                       ;mentace
0209  C28C02              LNZ  CLRLP   ;opakování cyklu dle
                                       ;reg. B
0293  C9                  RET          ;návrat
```

4.1.2 Podprogramy pro realizaci časového zpoždění

Základní podprogramy pro časové zapojení jsou dva - DELAY a DELYA. Podprogramy DELAY plní obsah střadače konstantou pro zpoždění 1 ms (83H). Na něj navazuje podprogram DELYA tvořící cyklus:

```
0236  3E83      DELAY: MVI  A,83H     ;konstanta pro 1 ms
                                       ;do střadače
8238  3D        DELYA: DCR  A         ;dekrementace střadače
0239  C23802              JNZ  DELYA   ;je obsah střadače
                                       ;nulový ?
023C  C9                  RET          ;návrat
```

Pokud potřebujeme další časová zpoždění, můžeme volat podprogram DELAY vícekrát v cyklu.

Příklad 4.5: Příklad programu pro realizaci časového zpoždění v závislosti na obsahu registrů H, L.

```
                          ORG 8000H
8000  CD3602    DELYH: CALL DELAY     ;základní zpoždění 1 ms
```

pokračování

pokračování

```
8003  CD3602    DELYH: CALL DELAY     ;dekrementace paramet-
                                      ;ru cyklu
8004  7C               MOV  A,H       ;je obsah HL nulový ?
8005  B5               ORA  L         ;
8006  C20080           JNZ  DELYH     ;jestliže ano, opako-
                                      ; vání
8009  C9               RET            ;návrat
```

4.1.3 Pomocné podprogramy

K pomocným podprogramům patří SHLRT, SHLRZ a SHLRC, které posouvají obsah registrů HL o jeden bit doprava, přitom DHLRC naplní uvolněný nejvýhnamnější bit registru H hodnotou CY; podprogram SHLRZ jej naplní nulou. Podprogram SHLRT navíc nastaví příznak Z = 1 je-li HL = 0. V registru CY vždy zůstává nejméně významný bit registru L.

K pomocným podprogramům řadíme dále podprogram COPY, který přemístí obsah specifikované části paměti do jiné oblasti a dále podprogram ERROR sloužící pro chybová hlášení (Err) na displeji.

Podprogram ERROR současně nuluje celý displej. Po jeho volání je nutno zastavit procesor, aby se opět nezobrazilo další hlášení monitoru:

```
                         ORG 8200H
8200  CDBC00    CP6:   CALL ERROR     ;chybové hlášení
8203  76               HLT            ;zastavení procesoru
```

4.2 Programování obvodů 8255 v ŠMS - VÚVT

Jak již bylo uvedeno v odst. 3.2.2 má školní mikropočítač ve své základní sestavě 3 programovatelné obvody 8255. Ve výše zmíněném odstavci jsou shrnuty i základní údaje o volacích adresách a režimech použitých obvodů 8255.

Ve školním mikropočítači lze bez omezení ve všech režimech použít pouze obvod 8255 č. 1, a proto je většina následujících programů na tento obvod zaměřena.

Příklad 4.6: Nastavte obvod 8255 č. 1 následovně:

brána P1A - režim 0, výstup
brána P1B - režim 1, výstup
brána P1C - P1C4 až P1C7 výstup

Poznámka: 1AØ je současně použit pro proudovou smyčku T 100 (viz odst. 3.2.8).

Dle zásad uvedených v odst. 2.8.4 je nutno vyslat do řídicího registru 8255 slovo 1 0 0 0 0 1 1 1 B = 87H. V souladu s tab. 3.1 a tab. 3.2 můžeme napsat následující program:

```
                        ORG 8200H
8200  3E87    CP7:      MVI  A,87H     ;řídicí slovo 8255
8202  D307              OUT  7         ;řídicí slovo do CWR
8204  E7                RST  4         ;návrat do monitoru
```

Po odstartování programu zhasnou všechny výstupní diody brány P1A na desce školního mikropočítače.

Příklad 4.7: Navrhněte program procyklický posuv rozsvícené diody na bráně P1A.

```
                        ORG 8200H
8200  3E80    CP16:     MVI  A,80H     ;řídicí slovo pro 8255
                                       ;č. 1
                                       ;brána A = výstup
8202  D307              OUT  7         ;výstup do 8255 č. 1
8204  D304    N:        OUT  4         ;rozsvícení jednoho
                                       ;bitu na P1A
8206  CD9502            CALL  DBYTE    ;zobrazení výstupu P1A
                                       ;na displeji
8209  21E803            LXI  H,1000    ;nastavení parametru
                                       ;zpoždění
820C  F5                PUSH PSW       ;úschova výstupu P1A
820D  CD0080            CALL DELYH     ;podprogram zpoždění
8210  F1                POP  PSW       ;obnovení stavu P1A
8211  0F                RRC            ; posun jedničky ve
                                       ; střadači doprava
                                                pokračování
```

pokračování

```
8212  C30482          JMP  N        ;opakování cyklu
```

Poznámka: Před odstartovnáním programu CP16 musíme uložit do počítače i program DELYH z příkladu 4.5.

Příklad 4.8: S použitím obvodu 8255 generujte na reproduktoru tón o frekvenci 500 Hz se střídou 1 : 1. Výstup na bráně P1A - druhý bit (b_1).

```
                        ORG 8200H
8200  3E80    CP19:     MVI  A,80H    ;řídicí slovo 8255
                                      ;brána A = výstup
8202  D307              OUT  7        ;výstup na obvod
                                      ;8255 č. 1
8204  DB04    N:        IN   4        ;čtení stavu brány P1A
8206  EE02              XRI  02       ;změna bitu b1 na opač-
                                      ;nou hodnotu
8208  D304              OUT  4        ;výstup negovaného bitu
820A  CD3602            CALL DELAY    ;čekání po dobu 1 ms
820D  CD0482            JMP  N        ;opakování cyklu
```

Reproduktor připojíme jedním výstupem na kontaktové pole P1A, bit b_1, druhým kontaktem na + 5 V nebo na GND (nižší intenzita tónu).

Příklad 4.9: Navrhněte program pro generování tónu jako v příkladu 9.8, ale s měnící se frekvencí.

```
                        ORG 8200H
8200  3E80    CP20:     MVI  A,80H    ;řídicí slovo pro
                                      ;8255 č. 1
8202  D307              OUT  7        ;výstup do obvodu 8255
8204  06FF              MVI  B,0FFH   ;nastavení parametru
                                      ;zpoždění
8206  DB04    N:        IN   4        ;čtení stavu brány P1A
8208  EE02              XRI  02H      ;negace prvního bitu
                                      ;brány P1A1
820A  D304              OUT  4        ;výstup negovaného bitu
```

pokračování

pokračování

```
820C  78        MOV  A,B     ;příprava parametru
                             ;zpoždění
820D  CD3802    CALL DELYA   ;čekání na stav
                             ;střadače
8210  05        DCR  B       ;snížení parametru
                             ;zpoždění
8211  C30682    JMP  N       ;opakování cyklu.
```

4.3 Programy využívající přerušení

Školní mikropočítač nemá, jak je obvyklé u mikroprocesoru 8080A, systém přerušení založen na použití instrukcí RST. Vznik přerušení (signál INT procesoru) lze vyjádřit funkcí:

$$INT = P_1 \cdot U_1 + P_2 \cdot U_2 + P_4 \cdot U_4 + P_5 \cdot U_5 + P_6 \cdot U_6 + P_7 \cdot U_7$$

kde P_i ... povolení i-tého přerušení
U_i ... výskyt i-té přerušovací události

Princip vytvoření přerušovacích signálů již byl popsán v odst. 3.2.2 ÷ 3.2.4 spolu s popisem obvodů 8255 a 8253 a obvody A/D a D/A. Zde pouze zrekapitulujeme přehled přerušovacích událostí - viz tab. 4.2:

Číslo události	Přerušovací zdroj
1	čítač z 8253 č. 0
2	čítač z 8253 č. 1
3	čítač z 8253 č. 2
4	obvody D/A převodníku
5	externí vstup na konektoru ExT4
6	externí vstup na konektoru EXT5
7	bit č. 3 brány 1C

Tabulka 4.2 Přehled přerušení ŠMS - VÚVT

Bližší podrobnosti použití zdrojů přerušení naleznete v lit. /22/. Zde pouze připomeneme, že signál U_1 způsobí přerušení typu RST5, ostatní signály RST6. Monitor používá

pro svoje účely (režim STEP) přerušení RST7. Jejich činnost je popsána v odst. 3.1.3.

4.4 Programování obvodů 8253 v ŠMS - VÚVT

Základní údaje jsou uvedeny v odst. 3.2.3 v tab. 3.3 a 3.4.

Příklad 4.10: Pomocí obvodu 8253 generujte signál 500 Hz (t = 2 ms). Výstup signálu indikujte reproduktorem připojeným na svorky T0 a GND.

```
                        ORG 8200H
8200  3E36    CP12: MVI  A,36H     ;nastavení 8253 do re-
                                   ;žimu 3
8202  D317          OUT  17H       ;vyslání řídicího slova
8204  3E00          MVI  A,0       ;nastavení LSB
8206  D314          OUT  14H       ;výstup LSB na T0
8208  3E10          MVI  A,10H     ;nastavení MSB
820A  D314          OUT  14H       ;výstup MSB na T0
820C  E7            RST4           ;návrat do monitoru
```

Poznámka 1. Při generování využíváme signálu Ø 2TTL s frekvencí 2048 kHz. Tuto frekvenci musíme vydělit 4096, t.j. 1000H pro t = 2 ms. Pro číslo předvolby tedy platí: předvolba = $2048 \cdot 10^3/f$ x /Hz-.

2. Tón se generuje trvale až do zavedení dalšího řídicího slova (klávesa RESET je neúčinná, neboť 8253 nemá nulovací vstup')

Příklad 4.11: Navrhněte program "hudebního nástroje" ovládaného v rozsahu 2 oktáv klávesami 0 ÷ F.

```
                        ORG 8200H
8200  3E36    CP23: MVI  A,36      ;režim 3, 2 slabiky,
                                   ;T0, binární čítání
8202  D317          OUT  17H       ;výstup na 8253
8204  CD5702  N1:  CALL SCAN       ;je stisknuta klávesa ?
```

pokračování

pokračování

```
8207  D20482          JNC   N1        ;jestliže ne, opakování
                                       ;testu ano, vynásobení
                                       ;kodu klávesy 4 nebot
                                       ;krok položek je 4
820B  87              ADD   A         ;
820C  210080          LXI   H,TAB     ;začátek tabulky
820F  0600            MVI   B,0       ;nulový MSB relativní
                                       ;adresy
8211  4F              MOV   C,A       ;relativní adresa tab.
8212  09              DAD   B         ;skutečná adresa položky
8213  7E              MOV   A,M       ;vybrání LSB předvolby
8214  D314            OUT   14H       ;výstup LSP na 8253, T0
8216  23              INX   H         ;další část položky
8217  7E              MOV   A,M       ;výběr MSB předvolby
8218  D314            OUT   14H       ;výstup MSB do T0
821A  23              INX   H         ;další část položky
821B  7E              MOV   A,M       ;výběr názvu tónu
821C  32F883          STA   83F8H     ;jeho zobrazení
821F  23              INX   H         ;další část položky
8220  7E              MOV   A,M       ;výběr druhé části
                                       ;názvu tonu
8221  32F983          STA   83F9H     ;jeho zobrazení
8224  CD5702    N2:   CALL  SCAN      ;je ještě stisknutá
                                       ;klávesa ?
8227  DA2482          JC    N2        ;jestliže ano, opak.testu
822A  3E00            MVI   A,0       ;není, smazání displeje
822C  32F883          STA   83F8H     ;
822F  32F983          STA   83F9H     ;
8232  C30082          JMP   CP23      ;opakování cyklu

                ;následuje tabulka dat

8000  E8167100  TAB:  DB    0E8H, 16H, 71H, 0
8004  68147D00        DB    68H, 14H, 7DH, 0
8008  2F127700        DB    2FH, 12H, 77H, 0
800C  2911767C        DB    29H, 11H, 76H, 7CH
8010  4A0F3900        DB    4AH, 0FH, 39H, 0
8014  9F0D5E00        DB    9FH, 0DH, 5EH, 0
```

pokračování

pokračování

```
8018  230C7900        DB   23H, OCH, 79H, 0
801C  740B7100        DB   74H, OBH, 71H, 0
8020  340A7D00        DB   34H, OAH, 7DH, 0
8024  17097700        DB   17H, 9, 77H, 0
8028  9508767C        DB   95H, 8, 76H, 7CH
802C  A5073900        DB   0A5H, 7, 39H, 0
8030  CF065E00        DB   0CFH, 6, 5EH, 0
8034  11067900        DB   11H, 6, 79H, 0
8038  BA057100        DB   0BAH, 5, 71H, 0
803C  1A057D00        DB   1AH, 5, 7DH, 0
8040  02000000        DB   2, 0, 0, 0
8044  02000000        DB   2, 0, 0, 0
8048  02000000        DB   2, 0, 0, 0
804C  02000000        DB   2, 0, 0, 0
8050  02000000        DB   2, 0, 0, 0
8054  02000000        DB   2, 0, 0, 0
8058  02000000        DB   2, 0, 0, 0
805C  02000000        DB   2, 0, 0, 0
```

Program napřed nastaví obvod 8253 a čeká na předvolbu. Testuje klávesnici pomocí podprogramu SCAN. Jestliže byla stisknuta klávesa, vytvoří čtyřnásobek jejího kódu - vzdálenost položek v tabulce je právě 4 slabiky. Data vyzvedne z tabulky; první dvě slabiky jsou předvolba a druhé kód názvu tónu. Další volání SCAN zjišťuje, zda je klávesa stisknuta. Jestliže ano, vysílání tónu pokračuje, v opačném případě se nuluje displej, nastaví znovu řídicí slovo 8253 a čeká se na stisknutí další klávesy. Změnu tónu můžeme dosáhnout změnou dat v tabulce.

4.5 Programování převodníků A/D a D/A v ŠMS - VÚVT

Popis převodníků je uveden v odst. 3.2.4, blokové schéma na obr. 3.5. Připomeneme, že vstup (8 bitů) převodníku D/A je připojen na bránu P1B a osmibitový spínač je řízen nultým bitem brány P1C. Analogový výstup aktivujeme rozepnutím spínače (P1C0 = 0) a na bránu P1B programem vyšleme potřebný údaj.

Rozsah výstupního napětí nastavíme ručně potenciometrem. Při aplikaci převodníku A/D můžeme použít buď řízení převodu procesorem, nebo automatickým režimem. V obou případech zůstává princip shodný - porovnávání vstupního napětí s napětím generovaným převodníkem D/A.

Programové řízení A/D převodu

Spínač nastavíme do rozepnutého stavu (P1C0 = 0) a programem, dle určitého algoritmu, generujeme číslo pro převodník D/A. Výstup komparátoru (třetí bit brány P2B) nám potom udá jejich shodu.

Lze použít dvě metody převodu - metodu postupných přírůstků a metodu postupné aproximace.

V prvém případu generujeme postupně čísla (8 bitů) pro D/A od nižší hodnoty (0 0 0 0 0 0 0 0 B) inkrementací po nejvyšší hodnotu (1 1 1 1 1 1 1 1 B). Převod je ukončen po hlášení komparátoru o shodě nebo nejbližší vyšší hodnotě. Tento způsob je sice jednoduchý, ale zdlouhavý, neboť v nejhorším případu musíme generovat až 255 čísel.

Vstupní analogový rozsah závisí na zesílení vstupního zasilovače (trimr ANALOG IN). Zesílení lze měnit v rozsahu 0,277 až 1 a tím dosáhnout vstupních napětí 0V až 2,55 V/0,277.

Druhý způsob, metoda postupných aproximací, je založen na principu půlení intervalu, kdy rozhodujeme v které polovině rozsahu bude hledaná hodnota. Nejprve necháme rozhodnout, v které polovině rozsahu je vstupní napětí, poté v které čtvrtině atd. Přitom se binární tvoří směrem od nejvyšších bitů k nejnižším (po každém kroku je znám jeden bit).

Automatické řízení převodu A/D

Zde je použita metoda přírůstková. V tomto případě nastavíme spínač v převodníku D/A do sepnuté polohy a bránu P1B do vstupního režimu. Na začátku převodu se čítač v převodníku D/A nuluje signálem R3. Čítač T2 z obvodu 8253 generuje pravidelné impulsy a čítač postupně inkrementuje. Děj se opakuje

až do vyhodnocení komparátorem, že vstupní napětí je nižší než napětí generované převodníkem D/A. Tím se nastaví P1C0 = 1 a převod je ukončen. Kód napětí, uložený v D/A přečteme pomocí brány P1B.

Automatický převod je výhodný zejména tehdy, jestliže necháme procesor zpracovávat jiný program a skončení převodu oznámíme pomocí přerušení nebo, v průběhu hlavního programu, periodicky testujeme bit P2B3, kde hodnota log. 1 indikuje ukončení A/D převodu (v případě přerušení čteme stav bran P1B). Pro tento účel je třeba naprogramovat T2 v obvodu 8253 na vhodnou frekvenci. Nulovací signál R3 lze generovat programově vysláním 0 0 0 0 0 1 1 X B na adresu OFH. Tím zároveň povolíme, nebo zakážeme, přerušení od ukončení převodu A/D.

Při programování nelze připustit, aby byl P1B ve výstupním režimu a současně P1C0 = 1. Proto jsou nepřípustná i všechna řídicí slova obvodu 8255 č. 1, která nastavují P1B do výstupního režimu současně P1C (0 - 3) do režimu vstupního (t.j. 81H, 89H, 91H, 99H).

Příklad 4.12: Navrhněte program převádějící číslo, zadané z klávesnice, na analogové napětí.

```
                              ORG 8200H
8220   3E90     CP29:  MVI  A,90H     ;řídicí slabika, brá-
                                      ;na B = výstup, brána
                                      ;C = vstup
8202   D307            OUT  7         ;vyslání řídicí sla-
                                      ;biky na 8255 č. 1
8204   CD3603   CYKL:  CALL ENTBY     ;vstup slabiky z kláves.
8207   7D              MOV  A,L       ;přesun zadané slabiky
8208   D305            OUT  5         ;výstup slabiky do pře-
                                      ;vodníku D/A
820A   C30482          JMP  CYKL      ;opakování cyklu
```

Výstupní napětí kontrolujeme měřícím přístrojem zapojeným mezi svorky GND a ANALOG OUT.

Příklad 4.13: Sestavte program pro převodník A/D, řízený programem metodou přírůstků.

```
                        ORG 8000H
8000  C5       AD1:     PUSH B          ;uschování reg. B, C
8001  97                SUB  A          ;nulování střadače
8002  D305     CYKL:    OUT  5          ;výstup na převodn.D/A
8004  47                MOV  B,A        ;uschování vysl.kódu
8005  3E20              MVI  A,20H      ;kód zdržení 240 /us
                                        ;pro ustálení dat
8007  CD3802            CALL DELYA      ;začátek progr.zpoždění
                                        ;podle obsahu střadače
800A  DB0D              IN   0DH        ;vsup z P2V (pro P2B3)
800C  E608              ANI  00001000B  ;maska pro P2B3
800E  C21780            JNZ  HVT        ;je-li P2B3 = 1 skok na
                                        ;ukončení převodu
8011  78                MOV  A,B        ;obnova kódu čísla
8012  C601              ADI  1          ;inkrementace testova-
                                        ;ného kodu
8014  D20280            JNC  CYKL       ;a opakování jestliže
                                        ;nedošlo k přetečení
8017  78        HVT:    MOV  A,B        ;převzetí uchovaného
                                        ;výsledného kodu
8018  C1                POP  B          ;obnovení stavu reg.B,C
8019  C9                RET             ;návrat z podprogramu
```

Podprogram převodu A/D používá registr B, a proto jej vždy chráníme uložením do zásobníku. Činnost uvedeného programu AD1 může sledovat v režimu STEP (zde je vhodné vypustit podprogram DELYA).

Příklad 4.14: Sestavte program, který zobrazí naměřenou hodnotu analogového napětí z příkladu 4.13 v pravé části displeje.

```
                        ORG 8200H
8200  3E90      CP30:   MVI  A,90H      ;řídicí slovo pro
                                        ;8255 č. 1 brána B
                                         = výstup, brána C
                                         = výstup
```

pokračování

pokračování

```
8202  D307          OUT  7          ;výstup řídicího slo-
                                    ;va do 8255
8204  3E92          MVI  A,92H      ;řídicí slovo pro na-
                                    ;stavení 8255 č.1
                                    ;brána B = vstup
8206  D30F          OUT  0FH        ;výstup řídicího slova
8208  CD0080   AD:  CALL ADl        ;volání převodu A/D
820B  CD9502        CALL DBYTE      ;zobrazení naměřené hodn.
820E  C30882        JMP  AD         ;opakování měření
```

Příklad 4.15: Sestavte program převodu A/D využívající metody postupné aproximace. Pro jeho ověření použijte program CP30 z příkladu 4.14. Záporná napětí budou zobrazena jako 00H, přepětí kódem FFH.

```
                    ORG 8000H
800   C5      AD2:  PUSH B          ;uchování reg.B,C
8001  97            SUB  A          ;nulování střadače
8002  0680          MVI  B,10000000B ;nastavení nejvyšší-
                                    ;ho bitu kódu
8004  B0      CYKL: ORA  B          ;přepsání nastaveného
                                    ;bitu do střadače
8005  D305          OUT  5          ;vyslání kódu do D/A
8007  4F            MOV  C,A        ;úschova střadače do
                                    ;registru C
8008  3E30          MVI  A,30H      ;zpoždění pomocí podprog.
800A  CD3802        CALL DELYA      ;DELYA asi 300 /us
800D  DB0D          IN   0DH        ;vstup brány P2B
                                    ;(kvůli P2B3)
800F  E608          ANI  08H        ;určení stavu P2B3
8011  79            MOV  A,C        ;obnovení kódu ve střad.
8012  CA1780        JZ   OBSK       ;přeskočení změny bitu
8015  A8            XRA  B          ;změna naposled nastave-
                                    ;ného bitu
8016  4F            MOV  C,A        ;úschova nového kódu
                                    ;do registru C
8017  78      OBSK: MOV  A,B        ;příprava obsahu B
                                    ;na posun
8018  0F            RRC             ;posun o bit doprava
```

pokračování

pokračování

```
8019  DA2180        JC   HTV      ;je-li převod ukon-
                                  ; čen skok
801C  47            MOV  B,A     ;jinak návrat posunu
                                  ;do registru B
801D  79            MOV  A,C     ;obnovení měřené hodnoty
801E  C30480        JMP  CYKL    ;opakování cyklu
8021  79      HTV:  MOV  A,C     ;přesun výsledku do stř.
8022  C1            POP  B       ;obnova registrů B, C
8023  C9            RET          ;návrat z podprogramu
```

Příklad 4.16: Navrhněte program pro generování sinusového obdélníkového, pilového a trojúhelníkového průběhu pomocí převodníku D/A. Tvar signálu lze volit z klávesnice následovně:

1 ..... sinusový průběh
2 ..... trojúhelníkový průběh
3 ..... pilový průběh
4 ..... obdélníkový průběh

Stisknutí jakékoliv jiné klávesy způsobí chybové hlášení (Err) a zastavení programu.

Kontrolu činnosti programu proveďte:

a) osciloskopem
b) posouzením změny tónu reproduktorem.

```
                        ORG 8200H
8200  3E90      CP21: MVI  A,90H    ;řídicí slovo pro
                                    ;8255 č. 1
                                    ;brány B,C = výstup
8202  D307            OUT  7        ;vyslání řídicího slova
8204  210080          LXI  H,TABL   ;začátek tab.obdélníku
8207  220081          SHLD 8100H    ;úschova aktuální adre-
                                    ;sy tabulky
820A  0620      N1:   MVI  B,32     ;počet vzorků v tab.
820C  2A0081          LHLD 8100H    ;převzetí adresy tab.
820F  7E        NO:   MOV  A,M      ;výběr položky z tab.
8210  D305            OUT  5        ;a její vyslání do D/A
```

pokračování

pokračování

```
8212  23              INX  H        ;další položka tab.
8213  05              DCR  B        ;jsou všechny pol.tab.
                                    ; přečteny ?
8214  C20F82          JNZ  NO       ;jestliže ne, opaková-
                                    ;ní výběru z tabulky
8217  CD5702          CALL SCAN     ;jinak testování klávesn.
821A  D20A82          JNC  N1       ;nebyla stisknuta žádná
                                    ;klávesa a proto opak.
821D  0604            MVI  B,4      ;štyři možnosti vstupu
                                    ;z klávesnice
821F  210080          LXI  H,TABL   ;adresa první tab-obdélníky
822   112000          LXI  D,32     ;vzdálenost tabulek
8225  220081    N2:   SHLD 8100H    ;odložení adresy aktuál-
                                    ;ní tabulky
8228  B8              CMP  B        ;je v registru B kód
                                    ;stisknuté klávesy ?
8229  CA0A82          JZ   N1       ;jestliže ano, opaková-
                                    ;ní výběru z tabulky
822C  19              DAD  D        ;jestliže ne, adresa
                                    ;další tabulky
822D  05              DCR  B        ;snížení parametru cyklu
822E  C22582          JNZ  N2       ;a jeho opakování
8231  CDBC00          CALL ERROR    ;když program došel až
                                    ;sem, znamená to, že byla
                                    ;stisknuta chybná klávesa
8234  76              HLT           ;a proto zastavení progr.
```

;následují tabulky průběhu signálu

```
                        ORG 8000H
8000  FFFFFFFF  TABL:  DB  OFFH,OFFH,OFFH,OFFH; tab.obdélníků
8004  FFFFFFFF        DB  OFFH,OFFH,OFFH,OFFH
8006  FFFFFFFF        DB  OFFH,OFFH,OFFH,OFFH
800C  FFFFFFFF        DB  OFFH,OFFH,OFFH,OFFH
8010  00000000        DB  0,0,0,0
8014  00000000        DB  0,0,0,0
8018  00000000        DB  0,0,0,0
801C  00000000        DB  0,0,0,0
8020  00081018  TAB2:  DB  0,8,10H,18H          ;tab.pily
8024  20283038        DB  20H,28H,30H,38H
8028  40485058        DB  40H,48H,50H,58H
```

pokračování

pokračování

```
802C  60687078        DB  60H,68H,70H,78H
8030  80889098        DB  80H,88H,90H,98H
8034  A0A8B0B8        DB  0A0H,0A8H,0B0H,0B8H
8038  C0C8D0D8        DB  0C0H,0C8H,0D0H,0D8H
803C  E0E8F0F8        DB  0E0H,0E8H,0F0H,0F8H
8040  00102030 TAB3:  DB  0,10H,20H,30H
8044  40506070        DB  40H,50H,60H,70H
8048  8090A0B0        DB  80H,90H,0A0H,0B0H
804C  C0D0E0F0        DB  0C0H,0D0H,0E0H,0F0H
8050  FFF0E0D0        DB  0FFH,0F0H,0E0H,0D0H
8054  C0B0A090        DB  0C0H,0B0H,0A0H,90H
8058  80706050        DB  80H,70H,60H,50H
805C  40302010        DB  40H,30H,20H,10H
8060  8099B1C7  TAB:  DB  80H,99H,0B1H,0C7H     ;sinus
8064  DBEBF6FD        DB  0DBH,0EBH,0F6H,0FDH
8068  FFFDF6EB        DB  0FFH,0FDH,0F6H,0EBH
806C  DBC7B199        DB  0DBH,0C7H,0B1H,99H
8070  80675439        DB  80H,67H,54H,39H
8074  25160A05        DB  25H,16H,0AH,5H
8078  00050A16        DB  0,5H,0AH,16H
807C  25395467        DB  25H,39H,54H,67H
```

4.6 Programy s elektromotorem

V základní sestavě ŠMS - VÚVT je k dispozici elektromotor na jehož hřídeli je optický kotouč pro přerušování světla diody LED a tím umožňující zpětné měření počtu otáček. Ovládání elektromotoru je možné z výkonového stupně ŠMS - VÚVT. Řízení rychlosti otáčení motorku je možné buď stejnosměrným napětím z D/A převodníku, nebo impulsně programem, případně s použitím časovače 8253.

Příklad 4.17: Navrhněte program pro řízení rychlosti otáčení elektromotoru v závislosti na kódu stisknuté klávesy.

```
                        ORG 8200H
8200  3E92   CP25:  MVI  A,92H       ;inicializace bran:
8202  D307         OUT  7           ;P1C = výstup
8204  3E02   CYKL:  MVI A,00000010B;nastavení P1C1 = 0
8206  D307         OUT  7           ;
8208  97           SUB  A           ;nulování střadače
8209  FE00   CP:    CPI  0           ;test parametru cyklu
820B  F5           PUSH  PSW        ;úschova paramtru cyklu
820C  C21382       JNZ  DELA        ;přeskok nastavení P1C1
                                    ;= 1
820F  3E03         MVI A,00000011B;nastavení P1C1 = 1
8211  D307         OUT  7           ;
8213  3E10   DELA:  MVI  A,10H      ;nastavení zpoždění
                                    ;cca 110 /us
8215  CD3802       CALL DELYA       ;podprogram zpoždění
8218  F1           POP  PSW         ;obnova paramtru cyklu
8219  3C           INR  A           ;inkrementace par.cyklu
821A  C20982       JNZ  CP          ;není-li konec cyklu,
                                    ;opakování
821D  CD5702       CALL SCAN        ;vstup klávesy
8220  D20482       JNC  CYKL        ;nebyla-li stisknuta
                                    ;nová klávesa, pokračo-
                                    ;vání novým cyklem se
                                    ;starou hodnotou
8223  07           RLC              ;posuv bitů o 4
8224  07           RLC              ;
8225  07           RLC              ;
8226  07           RLC              ;
8227  320A82       STA  CP + 1      ;uložení nové rozhodo-
                                    ;vací hodnoty
822A  C30482       JMP  CYKL        ;pokračování novým cyklem
```

Program nastaví P1C na výstupní režim pro řízení diody LED optočlenu bitem P1C1. (MOT CTL + připojen na + 5 V). Řízení začíná nastavením plného napětí motoru (P1C1 = 0) a poté běží cyklus 256 krát a dle stavu střadače se buzení motoru odpojí v okamžiku, kdy se zjistí shoda paramtru cyklu a hodnoty zadané z klávesnice (instrukce CPI na adrese 8209H). Trvání cyklu je upraveno programovým zdržením DELYA.

Po ukončení cyklu se testuje klávesnice. Není-li stisknuta žádná klávesa, začíná nový cyklus se shodnou hodnotou rozhodování pro vypnutí buzení, v opačném případě se kód klávesy posune o 4 místa doleva (dosažení rozsahu hodnot 00--F0H s krokem 10H) a takto upravená hodnota se uloží přímo do instrukce rozhodování na adrese 8209H.

Podobným způsobem je možno sestavit řadu dalších programů pro řízení rychlosti otáčení včetně smyčky zpětné vazby přes snímání stavu otáček hřídele.

4.7 Standardní programy

Jak již bylo uvedeno, podává vyčerpávající přehled základních programů lit. /26/, případně snáze dostupná lit. /23/, /27/, /28/.

V dalším uvedeme některé z nich (lit. /26/), upravené pro paměťovou lokaci ŠMS - VÚVT. V přehledu jsou uvedeny nejčastěji používané programy, bližší viz citovaná literatura.

Příklad 4.18: Nulování paměťového místa.
Vynulujte paměťové místo s adresou 8000H.

```
                        ORG 8200H
8200   97      START:   SUB  A        ;nulování střadače
8201   320080           STA  8000H    ;nulování RWM
8204   E7      STOP:    RST4          ;návrat do monitoru
                        END
```

Příklad 4.19: Jedničkový doplněk.
Vytvořte doplněk k číslu na adrese 8000H a výsledek uložte na adresu 8001H.

výchozí data (8000) = 6A
výsledek (8001) = 95

```
                              ORG 8200H
8200   3A0080   START:LDA   8000H      ;čtení dat RWM
8203   2F              CMA             ;jedničkový doplněk
8204   320180          STA   8001H     ;uložení dat
8207   E7       STOP:  RST4            ;návrat do monitoru
                       END
```

Příklad 4.20: Osmibitové sčítání

Sečtěte čísla na adrese 8000H a 8001H a výsledek uložte na adresu 8002H.

výchozí data (8000) = 38
(8001) = 2B
výsledek (8002) = 63

```
8200   210080   START:LXI   H,8000H   ;adresa prvního operandu
                                      ;do HL
8203   7E              MOV   A,M       ;první operand
8204   23              INX   H         ;adresa druhého operandu
8205   86              ADD   M         ;součet
8206   23              INX   H         ;adresa pro výsledek
8207   77              MOV   M,A       ;uložení součtu
8208   E7       STOP:  RST4            ;návrat do monitoru
                       END
```

Příklad 4.21: Posuv o jeden bit vlevo

Posuňte obsah paměti s adresou 8000H o jeden bit vlevo a prázdnou pozici bitu b0 vynulujte. Výsledek uložte na adresu 8001H.

výchozí data (8000) = 6F
výsledek (8001) = DE

```
                              ORG 8200H
8200   3A0080   START:LDA   8000H      ;čti data
8203   87              ADD   A         ;posun dat vlevo
```

pokračování

pokračování

```
8204  320180          STA  8001     ;uložení dat do RWM
8207  E7        STOP: RST4          ;návrat do monitoru
                    RND
```

Příklad 4.22: Maskování vyšších 4 bitů slova

Přesuňte nižší čtyři bity slova z adresy 8000H na adresu 8001H. Vyšší čtyři bity slova na adrese 8001H vynulujte.

výchozí data (8000) = B8
výsledek (8001) = 08

```
                        ORG 8200H
8200  3A0080  START: LDA  8000H     ;čti data
8203  E60F           ANI  00001111B ;maskování 4 vyšších
                                    ;bitů
8205  320180         STA  8001H     ;uložení dat do RWM
8208  E7      STOP: RST4            ;návrat do monitoru
                    END
```

Příklad 4.23: Určení většího ze dvou čísel

Porovnejte čísla na adresách 8000H a 8001H a větší z nich uložte na adresu 8002H. Uložená čísla jsou binární čísla bez znaménka.

výchozí data (8000) = 6F
(8001) = 1A
výsledek (8002) = 6F

```
                        ORG 8200H
8200  210082          LXI  H,8200H   ;nastavené adresy
8203  7E              MOV  A,M       ;a přečtení prvního
                                     ;operandu
8204  23              INX  H         ;druhý operand
8205  BE              CMP  M         ;je druhý operand větší?
8206  D20A82          JNC  DALE      ;není
8209  7E              MOV  A,M       ;je, a proto druhý do
                                     ;střadače
820A  23        DALE: INX  H         ;nastavení adresy paměti
                                              pokračování
```

pokračování

```
820B   77          MOV  M,A      ;uložení výsledku
820C   E7          RST4         ;návrat do monitoru
                   END
```

Poznámka: Instrukce CMP M porovnává obsah střadače a paměti a nastaví:

Z = 1 pro A = M
Z = 0 pro A ≠ M
CY = 1 pro A < M
CY = 0 pro A ≥ M

Příklad 4.24: Sestavte program pro násobení dvou osmibitových čísel. Výchozí data jsou uložena na adresu 8000H a 8001H, výsledek uložte na adresy 8002H a 8003H.

výchozí data (8000) = 02
(8001) = 07
výsledek (8002) = 0EH
(8003) = 00

```
                        ORG 8200H
8200   210080    ZAC:  LHI  H,8000H   ;adresa paměti
8203   5E              MOV  E,M      ;čtení násobence
8204   1600            MVI  D,0H     ;rozšíření na 16 bitů
8206   23              INX  H        ;nová adresa
8207   7E              MOV  A,M      ;násobitel
8208   2100            LXI  H,0H     ;součin 0
820B   0608            MVI  B,8H     ;nastavení počitadla
                                     ;cyklu
820D   29        MULT:  DAD  H        ;posun o 1 bit vlevo
820E   87               ADD  A       ;posun násobitele do CY
820F   D21382           JNC  TEST    ;je testovaný bit = 1?
8212   19               DAD  D       ;ano
8213   05        TEST:  DCR  B       ;
8214   C20D82          JNZ  MULT     ;konec násobení?
```

pokračování

pokračování

```
8217   220280          SHLD 8002H      ;uložení výsledku
821A   E7        STOP: RST4            ;návrat do monitoru
                      END
```

Příklad 4.25: Dělte šestnáctibitové číslo (bez znaménka) uložené na adrese 8000H (LSB) a 8001H (MSB) osmibitovým číslem uloženým na adrese 8002H. Čísla jsou upravena tak, že MSB dělence i dělitele jsou nulové a číslo na adrese 8002H je větší než číslo na adrese 8001H, podíl tedy může být vyjádřen osmibitovým číslem. Výsledek uložte na adresu 8003H, zbytek na adresu 8004H.

1) výchozí data (8000) = 40H
(8001) = 00
(8002) = 08
výsledek (8003) = 08
(8004) = 00
(64 : 8) = 8

2) výchozí data (8000) = 6DH
(8001) = 32H
(8002) = 47H
výsledek (8003) = B5H
(8004) = 3AH
(12909 : 71 = 181 + 58/71)

```
                      ORG 8200H
8200   2A0080  ZAC:  LHLD  8000H      ;čtení dělence
8203   3A0280        LDA   8002H      ;čtení dělitele
8206   4F            MOV   C,A        ;
8207   0608          MVI   B,8        ;nastavení počítadla
                                      ;cyklu
8209   29      DEL:  DAD   H          ;posuv dělence a podí-
                                      ;lu o 1 bit vlevo
                                      ;(b = 0)
820A   7C            MOV   A,H        ;
820B   91            SUB   C          ;8 zbylých MSB dělen-
                                      ;ce  dělitel
820C   DA1182        JC    DALE       ;není, pokračuj dal-
                                      ;ším cyklem
820F   67            MOV   H,A        ;ano, odečtení dělitele
8210   2C            INR   L          ;přičtení 1 k podílu
8211   05      DALE: DCR   B          ;
```

pokračování

pokračování

```
8212  C20982        JNZ  DEL        ;konec dělení?
                                     ;počítadlo = 0)
8215  220380        SHLD 8003H      ;uložení výsledku
8218  E7      STOP: RST4            ;návrat do monitoru
                    END
```

Příklad 4.26: Převeďte obsah paměťové buňky s adresou 8000H do kódu ASCII. Výsledek uložte na adresu 8001H.

výchozí data (8000) = 0CH
výsledek (8001) = 43H = 'C'

```
                    ORG 8200H
8200  3A0080  ZAC:  LDA  8000H      ;přečtení dat
8203  FE0A         CPI  10D        ;jsou data 10?
8205  DA0A82       JC   ASCZ       ;
8208  C607         ADI  'A'-'9'-1  ;ano, přičti posuv
                                     ;pro písmena
820A  C630   ASCZ: ADI  '0'        ne, přičti '0' ASCII
820C  320180       STA  8001H      ;uložení výsledku
820F  E7     STOP: RST4            ;návrat do monitoru
                    END
```

Poznámka: Připočtením kódu '0' ASCII převedeme správně pouze číslice 0 až 9. Zbývajících 7 znaků při převodu hexadecimálních čísel A až F musíme překlenout.

Příklad 4.27: Převeďte kód ASCII znaku z adresy 8000H na dekadické číslo a uložte je na adresu 8001H.

výchozí data (8000) = 36H = '6'
výsledek (8001) = 06

```
                    ORG 8200H
8200  06FF   ZAC:  MVI  B,0FFH     ;uložení značky chy-
8202  3A0080       LDA  8000H      ;by (FFH)
8205  D630         SUI  '0"        ;data do střadače
8207  DA1082       JC   'STORE     ;odečtení '0' ASCII
                                     ;jsou-li data než
                                     ;ASCII '0' pak CY = 1
```

pokračování

pokračování

```
820A   FE0A          CPI  10D        ;je výsledek  10D?
820C   D21082        JNC  STORE      ;jestliže ano, CY = 0
820F   47            MOV  B,A        ;
8210   78      STORE: MOV  A,B        ;
8211   320180        STA  8001H      ;uložení výsledku
8214   E7      STOP: RST4            ;návrat do monitoru
                     END
```

Poznámka: První rozhodnutí o převodu se uskuteční odečtením kódu '0' ASCII (data ASCII '0'), druhé rozhodování instrukcí CPI.

Příklad 4.28: Sestavte podprogram pro převod hexadecimálního kódu na kód ASII. Výchozí data jsou uložena na adrese 8000H, hlavní program na adrese 8200H tato data vyzvedne, vyvolá podprogram z adresy 8100H a výsledek uloží na adresu 8001H. Princip převodu použijte shodný s příkladem 4.26.

1) výchozí data (8000) = 0EH
   výsledek (8001) = 45H = 'E'

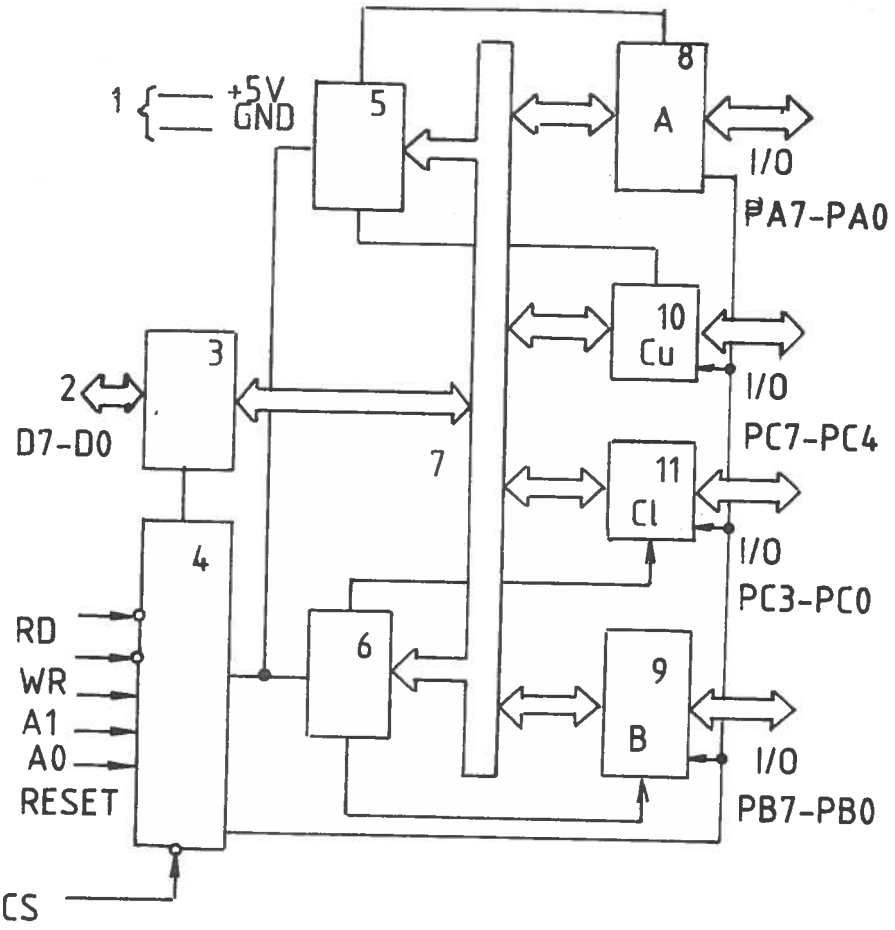
2) výchozí data (8000) = 04
   výsledek (8001) = 34H = '4'

```
                     ORG 8200H
                                   ;zde je uložen hlav-
                                   ;ní program
8200   319082   ZAC: LXI  SP,8290H ;vytvoření zásobníku
                                   ;na adrese 8290H
8203   3A0080       LDA  8000H     ;data do střadače
8206   CD0081       CALL HEXASC    ;zavolání podprogramu
                                   ;převodu kodu
8209   320180       STA  8001H     ;uložení výsledku
820C   E7     STOP: RST4           ;návrat do monitoru
```

pokračování

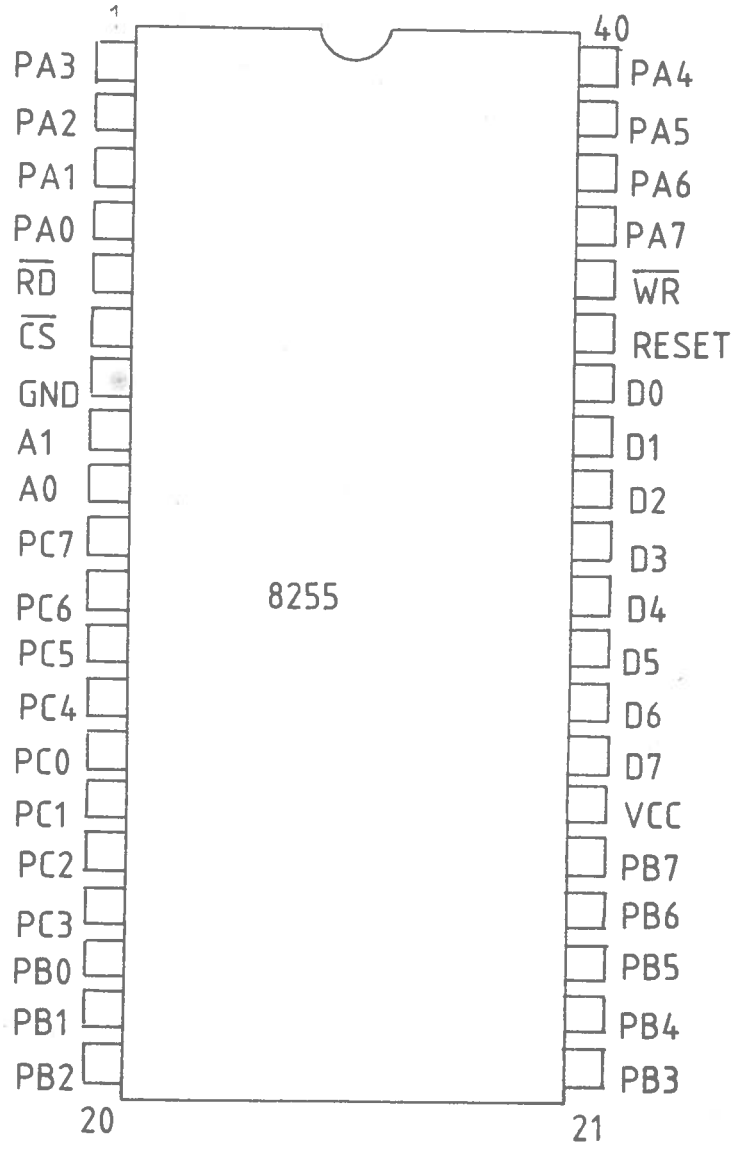
pokračování

```
                ;podprogram HEXASC slouží pro převod hexa-
                ;decimálního čísla, uloženého ve střadači,
                ;na znak ASCII výsledek ponechá ve střada-
                ;či
                ;
                ;vstupy: A - hexadecimální číslo
                ;
                ;výstupy: A - znak ASCII
                ;
                ;ovlivňují: A, F
                ;
                ;volá podprogramy: žádné
                ;
                        ORG 8100H
8100  FE0A HEXASC: CPI   10D          ;jsou data   10D?
8102  DA0781       JC    POSUN        ;
8105  C607         ADI   'A'-'9'-1    ;ano, přičtení posu-
                                       ;nu pro písmena
8107  C630  POSUN: ADI   '0'          ;přičtení posunu
                                       ;pro ASCII
8109  C9           RET                 ;návrat z podprogramu
                        END
```

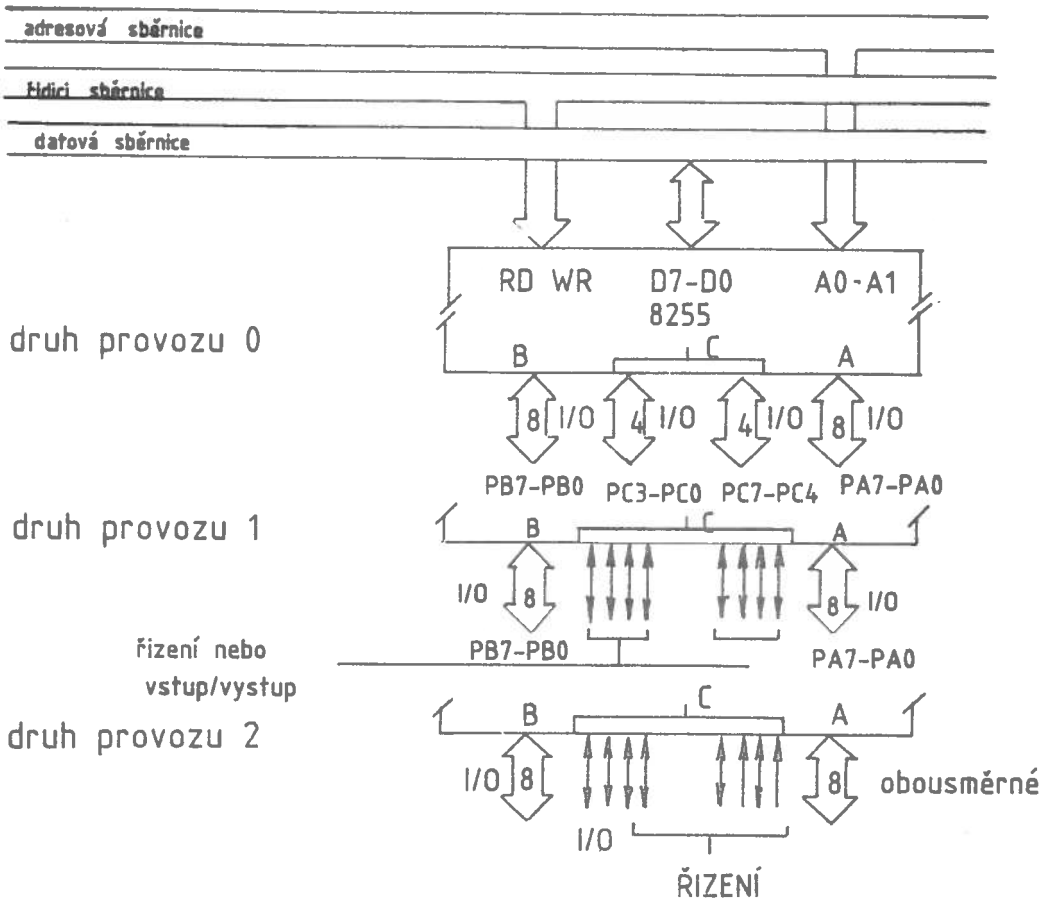


Obr. 2.9 Blokové schéma zapojení 8255

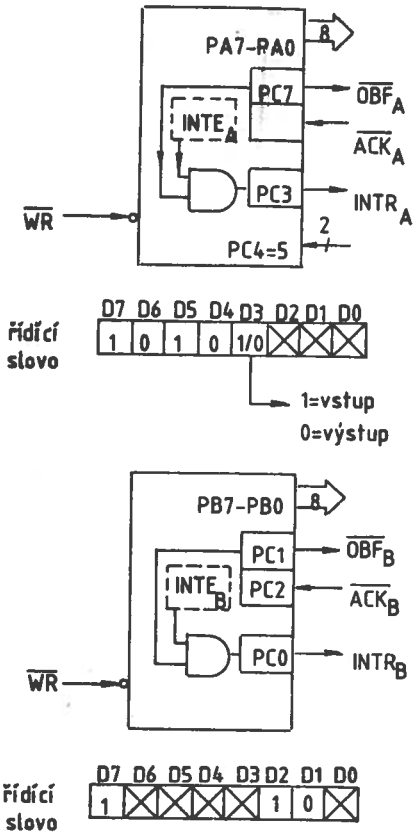
1-napájecí napětí, 2-datová sběrnice, 3-buffer datové sběrnice, 4-řídicí logika pro čtení a zápis, 5-řízení skupiny A, 6-řízení skupiny B, 7-vnitřní datová sběrnice, 8-skupina A, obvod A, 9-skupina B, obvod B, 10-skupina A, obvod C (vyššího řádu), 11-skupina B, obvod C (nižšího řádu).



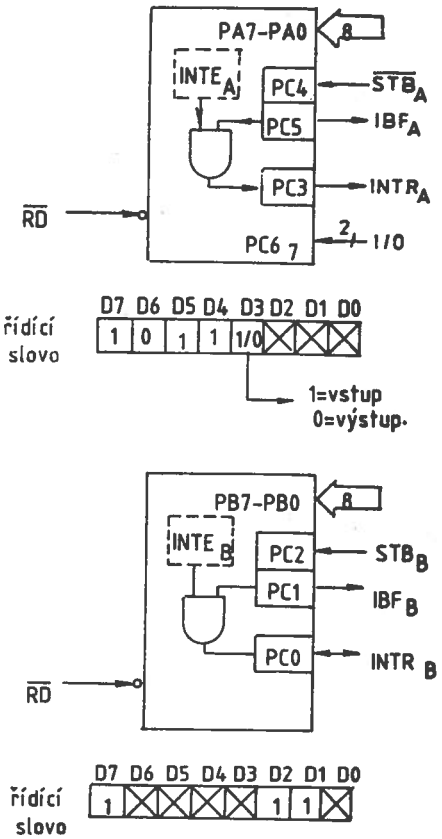
Obr. 2.10 Rozmístění vývodů obvodu 8255



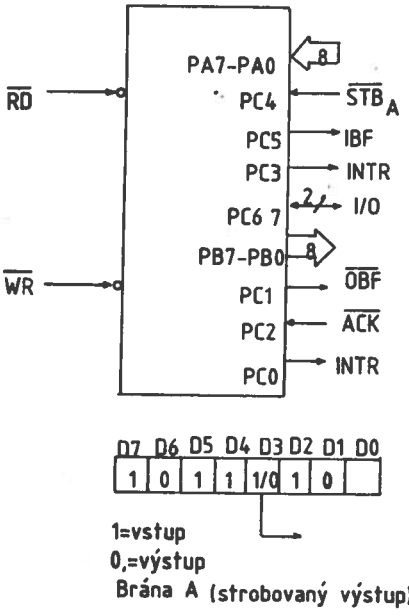
Obr. 2.12 Definice druhů provozu a interface pro sběrnici

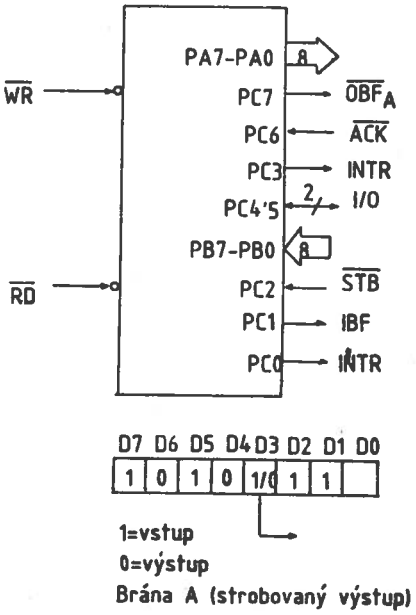


Obr. 2.14 Druh provozu 1 - výstupy

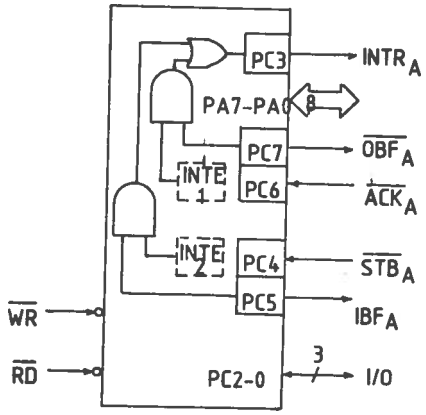


Obr. 2.15 Druh provozu 1 - vstupy

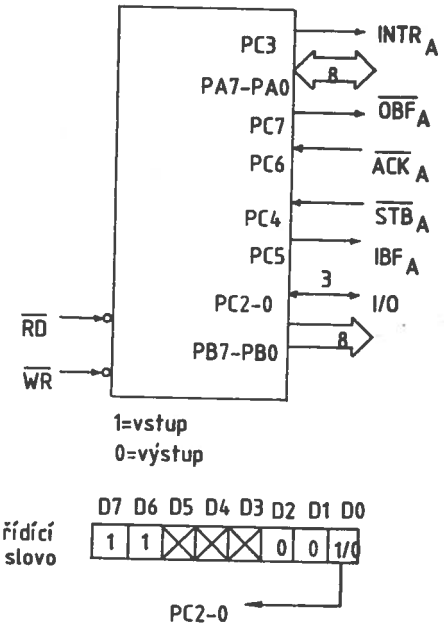




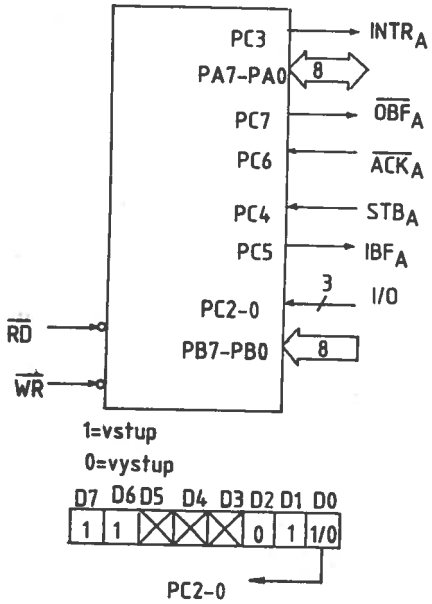
Obr. 2.16 Varianty při druhu provozu 1



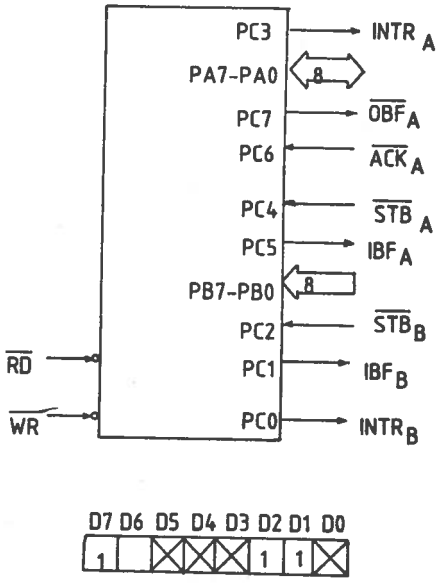
Obr. 2.18 Druh provozu 2



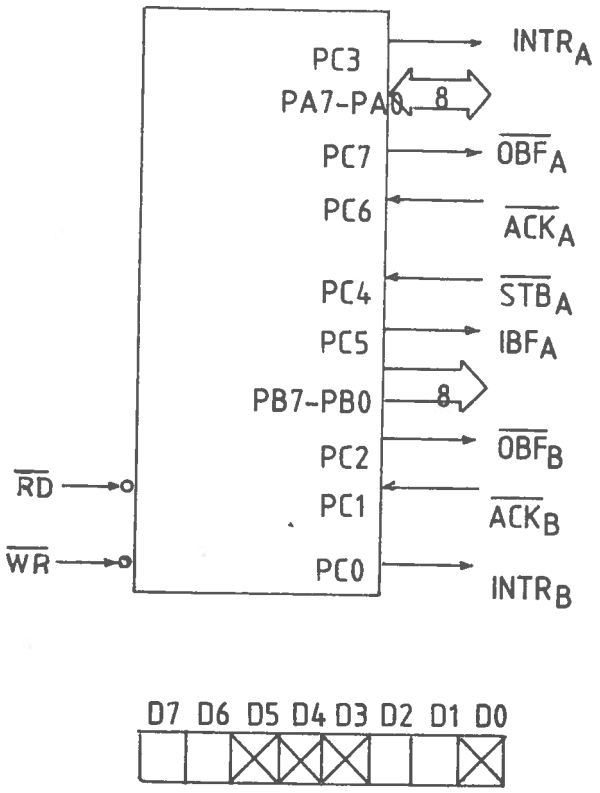
Obr. 2.19 Druh provozu 2 a 0 (výstupní operace)



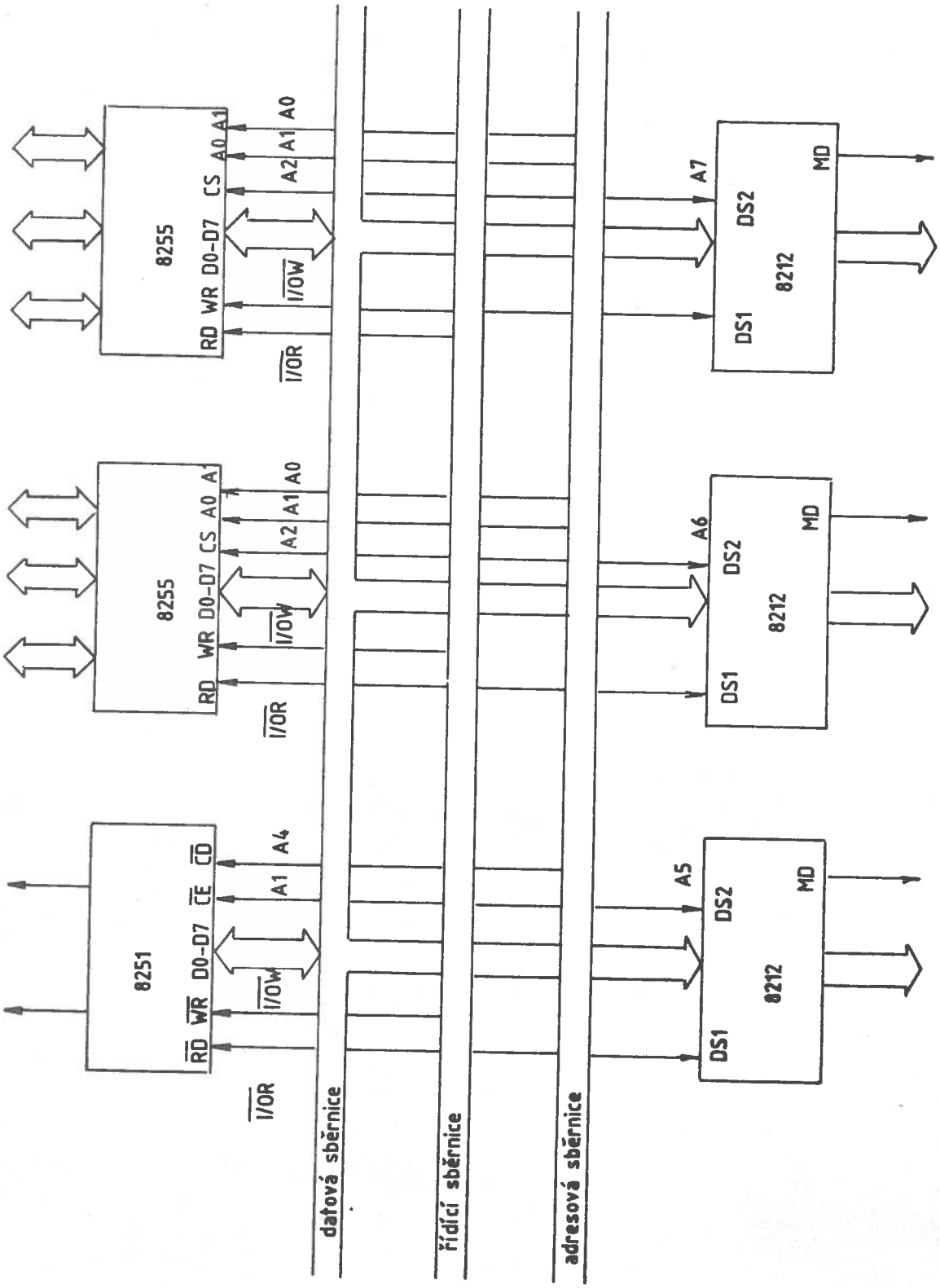
Obr. 2.20 Druh provozu 2 a 0 (vstupní operace)



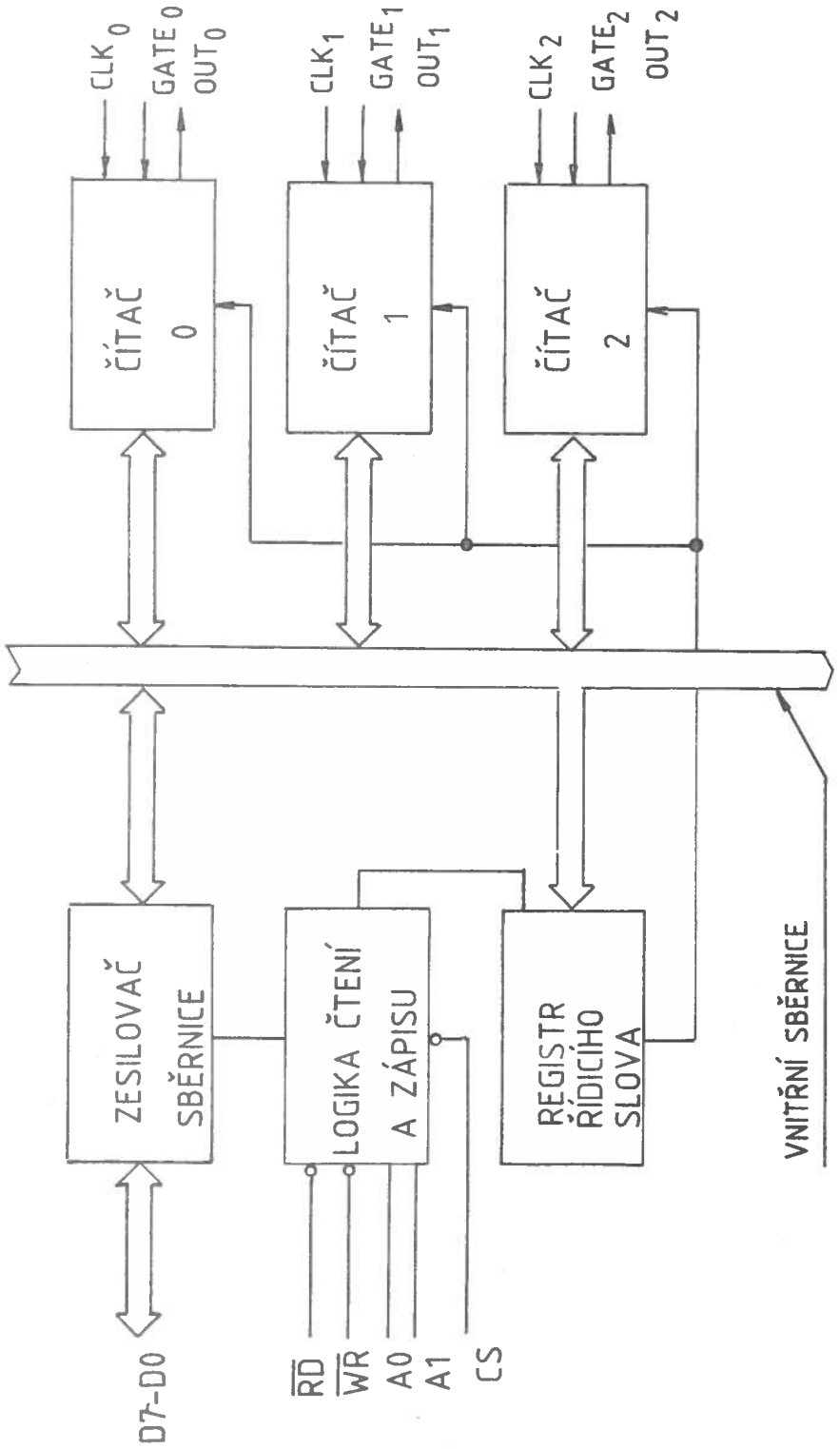
Obr. 2.21 Druh provozu 2 a 1 (vstupní operace)



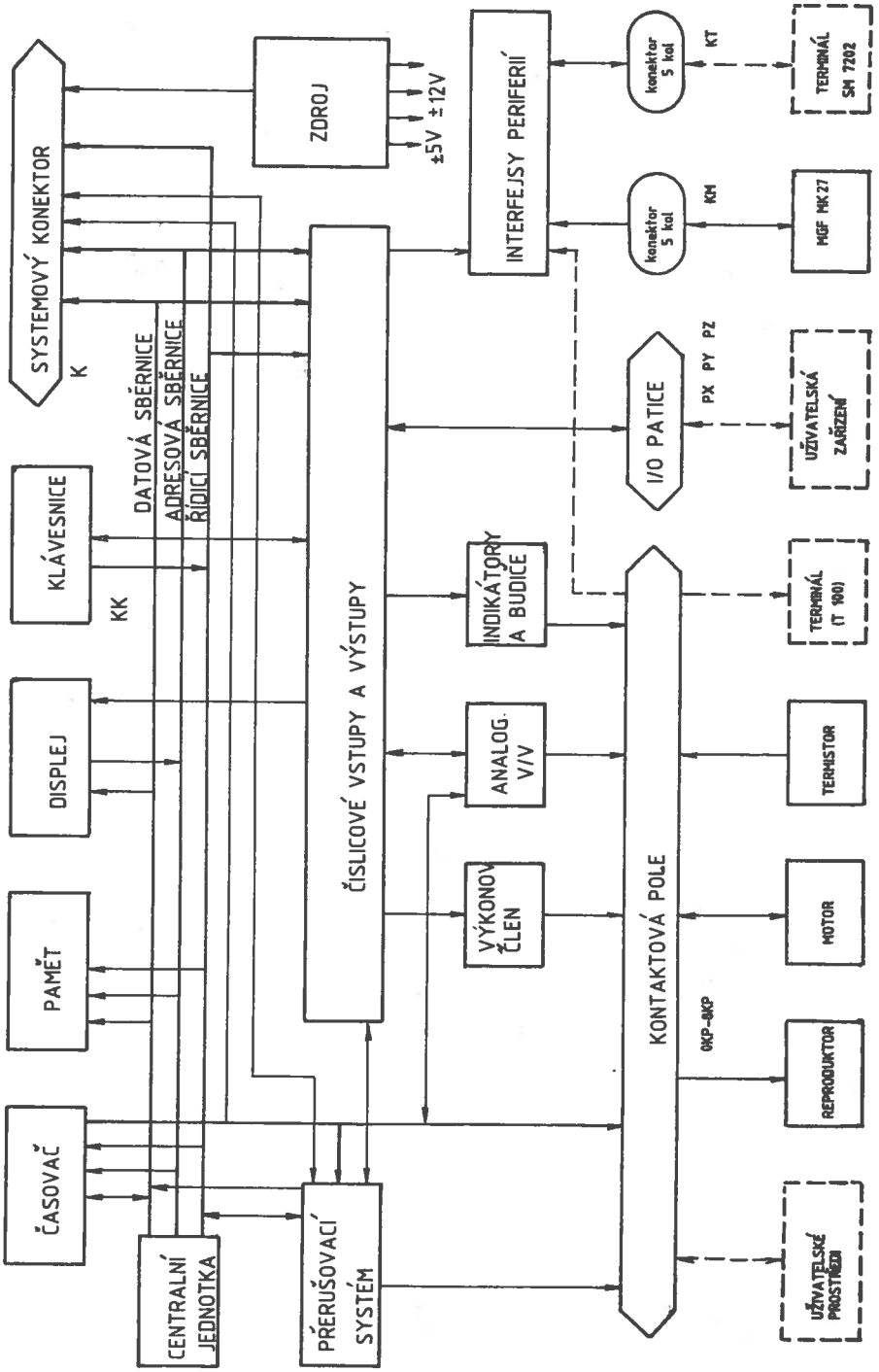
Obr. 2.22 Druh provozu 2 a 1 (výstupní operace)



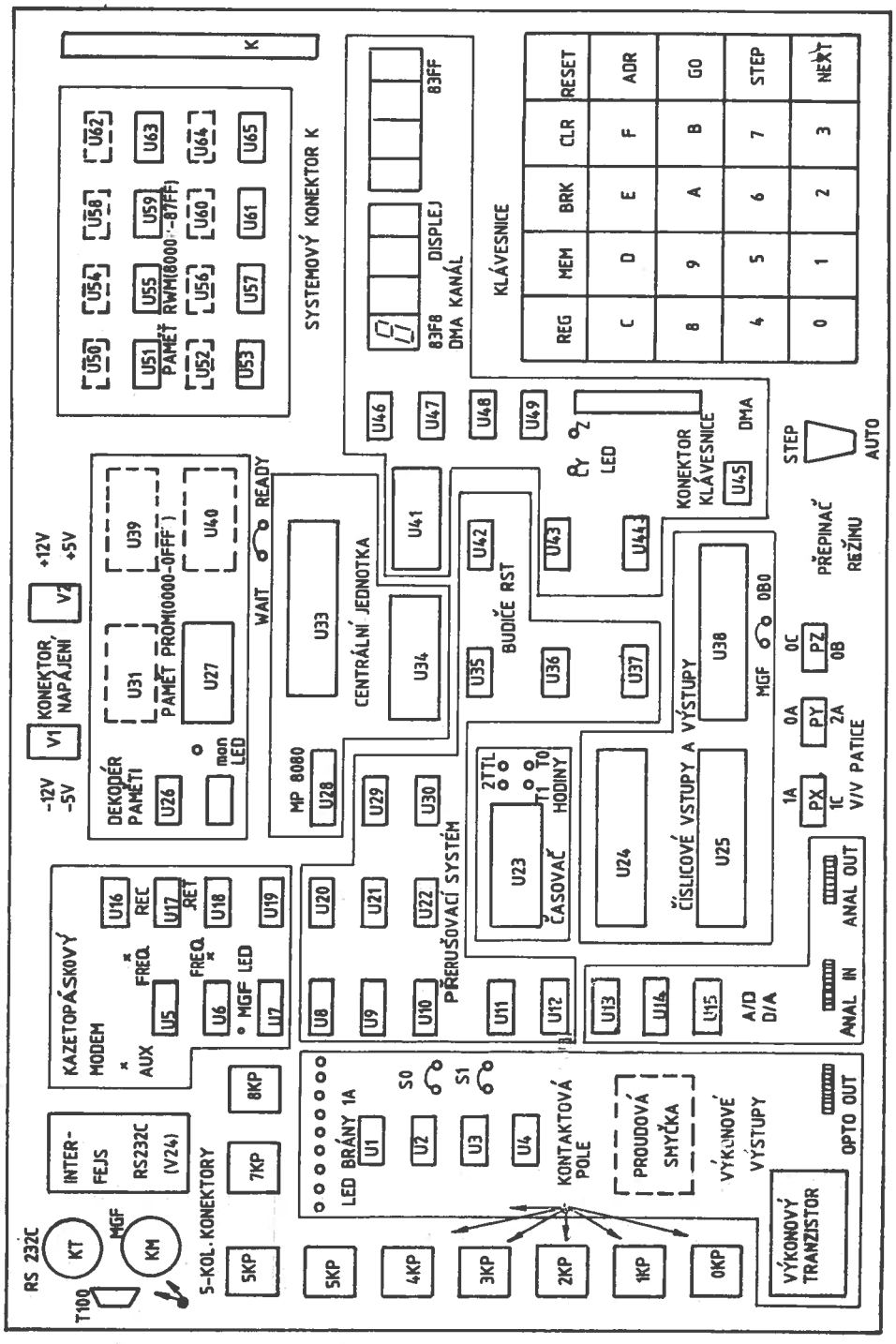
Obr. 2.24 Připojení obvodů V/V



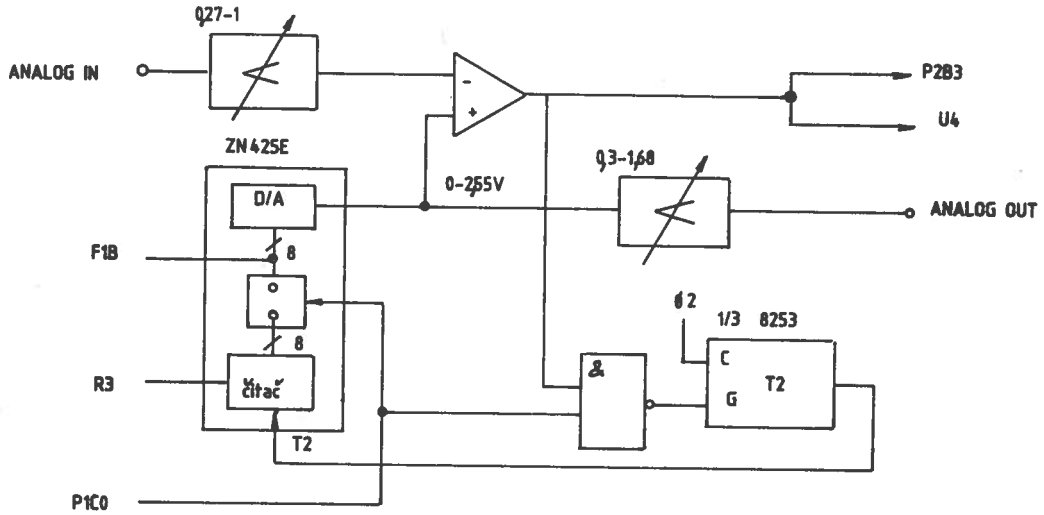
Obr. 2.25 Blokové schéma obvodu 8253



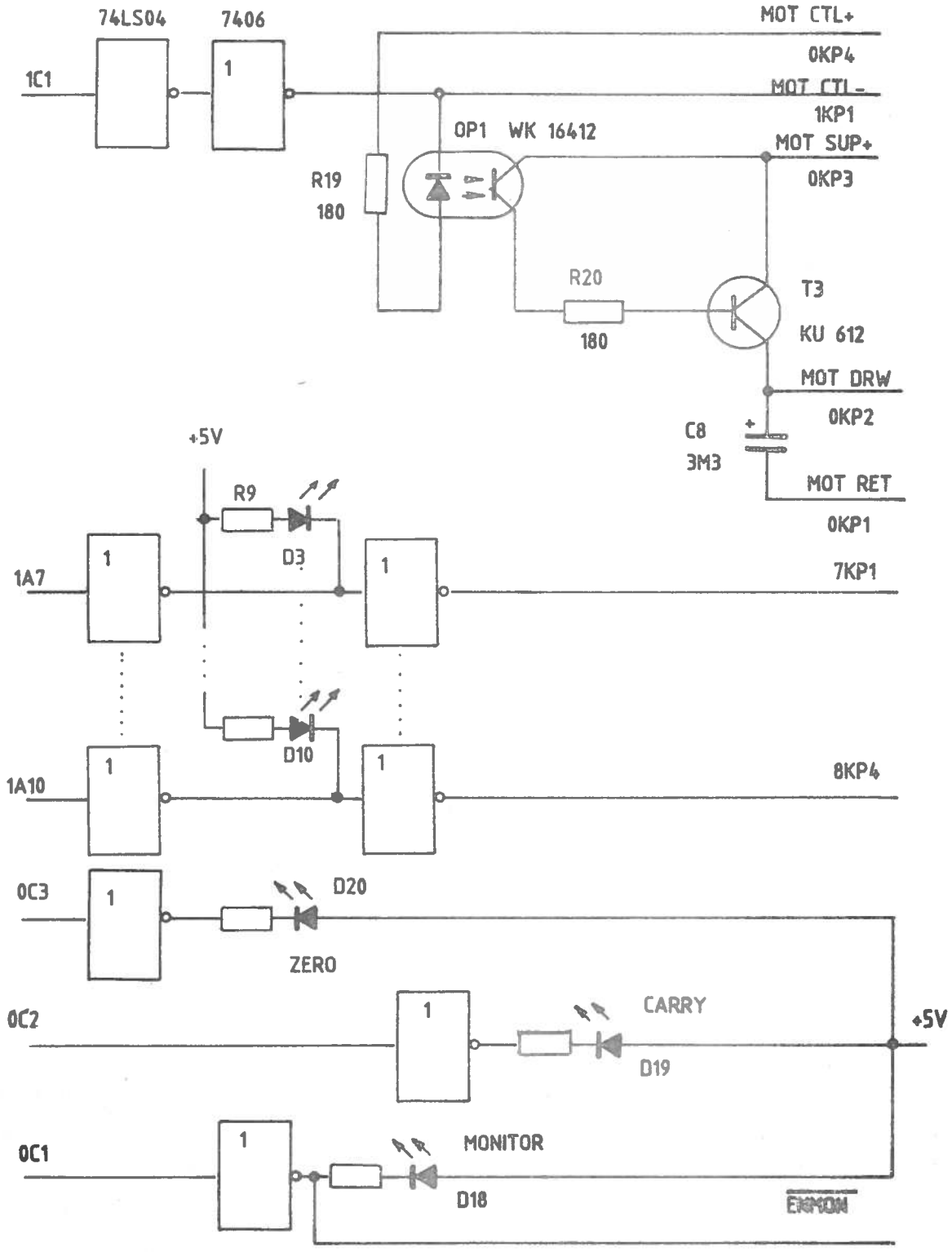
Obr. 3.1 Blokové schéma SMS - VÚVT



Obr. 3.2 Rozložení funkčních bloků na desce SMS - VÚVT



Obr. 3.5 Blokové schéma převodníku A/D a D/A



Obr. 3.6 Zapojení výkonových výstupů ŠMS - VÚVT

5. Příloha - Instrukční soubor 8080A

V následující kapitole je podrobněji rozveden instrukční soubor mikroprocesoru 8080A obsažený v kapitole 2. dílu I. Součástí kapitoly je i tabulka kódu ASCII.

5.1 Skupina přenosových instrukcí (Data Transfer Group)

Neovlivňují žádné příznakové bity.

5.1.1 Osmibitové přenosové instrukce

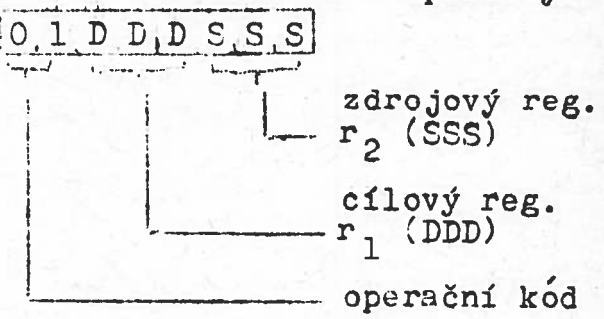
MOV — MOVE REGISTER (Přesuň registr)

Operace: $(r_1) \leftarrow (r_2)$

Formát: MOV r_1, r_2

$(r_1, _2$ = A,B,C,D,E,H,L)

tab.2.11 - operandy



Taktů: 1
Dob: 5
Ovlivňuje: ------

Instrukce MOV r_1, r_2 přesune obsah zdrojového registru r_2 do cílového registru r_1. Obsah zdrojového registru r_2 zůstává nezměněn.

Lze použít i shodné operandy (např. MOV B,B), pak instrukce pracuje shodně s NOP, ale je o jednu dobu delší.

MOV

MOVE FROM MEMORY (Přesuň paměť do registru)

Operace: (r) ← (M,H,L) ——— tab. 2.11

Formát: MOV r,M

0	1	D	D	D	1	1	0

(r = A,B,C,D,E,H,L)

Taktů: 2
Dob: 7
Ovlivňuje: -----

Instrukce MOV r,M přesune obsah paměťového místa M, adresovaného registrovým párem HL, do cílového registru . Obsah paměťového místa M zůstává nezměněn. Není povolen formát MOV, M,M.

MOV

MOVE TO MEMORY (Přesuň registr do paměti)

Operace: (M,H,L) ← (r) ——— tab. 2.11

Formát: MOV M,r

0	1	1	1	0	S	S	S

(r = A,B,C,D,E,H,L)

Taktů: 2
Dob: 7
Ovlivňuje: -----

Instrukce MOV M,r přesune obsah zdrojového registru r do paměťového místa M adresovaného registrovým párem HL. Obsah zdrojového registru r zůstává nezměněn. Není povolen formát MOV M,M'

MVI

MOVE IMMEDIATE (Naplň registr přímým operandem)

Operace: (r) ← (B2) tab. 2.11

Formát: MVI r,data

0	0	D	D	D	1	1	0	B2 data

(r = A,B,C,D,E,H,L)

Taktů: 2
Dob: 7
Ovlivňuje: -----

Instrukce MVI r,data naplní cílový registr r daty danými druhou slabikou instrukce.

MVI

Operace: (M,H⁄L) ← (B2)

Formát: MVI M,data

	B2
0 0 1 1 0 1 1 0	data

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce MVI M,data naplní paměťové místo M, adresované registrovým párem HL, daty uloženými v druhé slabice instrukce.

LDA

LOAD ACCUMULATOR DIRECT (Naplň střadač /přímá adr./)

Operace: (A) ← (M,B3B2)

Formát: LDA,adr

	adr	
0 0 1 1 1 0 1 0	adr LSB	adr MSB

Taktů: 4
Dob: 13
Ovlivňuje: -----

Instrukce LDA adr přesune obsah paměťového místa M, adresovaného druhou a třetí slabikou instrukce, do střadače. Obsah paměťového místa M zůstává nezměněn.

STA

STORE ACCUMULATOR DIRECT (Ulož střadač /přímá adr./)

Operace: (M,B3B2) ← (A)

Formát: STA adr

	adr	
	B2	B3
0 0 1 1 0 0 1 0	adr LSB	adr MSB

Taktů: 4
Dob: 13
Ovlivňuje: -----

Instrukce STA adr uloží obsah střadače do paměťového místa adresovaného druhou a třetí slabikou instrukce. Obsah střadače zůstává nezměněn.

LDAX — LOAD ACCUMULATOR INDIRECT (Naplň střadač /nepřímá adresa/)

Operace: (A) ← (M,rp)

Formát: STAX rp — 0,0,rp,0,0,1,0

(rp = B&C, D&E)

Taktů: 2

Dob: 7

Ovlivňuje: -----

Instrukce LDAX rp přesune obsah paměťového místa M, adresovaného registrovým párem rp, do střadače. Obsah paměťového místa M, zůstává nezměněn.

STAX — STORE ACCUMULATOR INDIRECT (Ulož střadač /nepřímá adresa/)

Operace: (M,rp) ← (A)

Formát: STAX rp — 0,0,rp,0,0,1,0

(rp = B&C,D&E)

Taktů: 2

Dob: 7

Ovlivňuje: -----

Instrukce STAX rp uloží obsah střadače do paměťového místa adresovaného rp. Obsah střadače zůstává nezměněn.

5.1.2 Šestnáctibitové přenosové instrukce

XCHG — EXCHANGE H AND L WITH D AND E (Zaměň HL a DE)

Operace: (H) ⟷ (D), (L) ⟷ (E)

Formát: XCHG — 1,1,1,0,1,0,1,1

Taktů: 1

Dob: 4

Ovlivňuje: -----

Instrukce XCHG vzájemně zamění obsahy registrových párů HL a DE.

LHLD

LOAD H AND L DIRECT (Naplň HL /přímá adresa/)

Operace: (L) ← (M,B3B2)
(H) ← (M,B3B2 + 1)

adr

Formát: LHLD adr

0 0 1 0 1 0 1 0	adr LSB	adr MSB

Taktů: 5
Dob: 16
Ovlivňuje: -----

Instrukce LHLD přesune obsah paměťového místa M, adresovaného druhou a třetí slabikou instrukce, do registru L. Do registru H přenese obsah následujícího paměťového místa (adr + 1). Obsahy paměťových míst se nemění.

Instrukci LHLD používáme k zadání nové adresy do HL pro instrukce pracující s pamětí (MOVr,M,DCRM atd.).

SHLD

STORE H AND L DIRECT (Ulož HL /přímá adresa/)

Operace: (M,B3B2) ← (L)
(M,B3B2 + 1) ← (H)

adr

Formát: SHLD adr

0 0 1 0 0 0 1 0	adr LSB	adr MSB

Taktů: 5
Dob: 16
Ovlivňuje: -----

Instrukce SHLD adr uloží obsah registru L na adresu paměťového místa daného druhou a třetí slabikou instrukce. Na následující paměťové místo (adr + 1) uloží obsah registru H. Obsah registrového páru HL zůstává nezměněn.

LXI — LOAD IMMEDIATE REGISTER PAIR (Naplň registrový pár /přímá adresa/)

Operace: (rh) ← B3, (r_1) ← (B2)

Formát: LXI rp	0 0 r p 0 0 0 1 (tab. 2.12)	data LSB	data MSB (data 16)

(rp = B,D,H,SP)

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce LXI rp naplní specifikovaný registrový pár daty obsaženými v druhé a třetí slabice instrukce.

5.2 Instrukce měnící obsah registru nebo paměti

(Increment and Decrement Group)

5.2.1 Operace s osmibitovými daty

Ovlivňují všechny příznakové bity s výjimkou CY.

INR — INCREMENT REGISTER (Inkrementuj registr)

Operace: (r) ← (r) + 1 (tab. 2.11)

Formát: INR r	0 0 D D D 1 0 0

(r = A,B,C,D,E,H,L)

Taktů: 1
Dob: 5
Ovlivňuje: Z, S, P, -, AC

Instrukce INR r přičte k obsahu specifikovaného registru r jedničku.

INR

INCREMENT MEMORY (Inkrementuj paměť)

Operace: (M,H&L) ← (M,H&L) + 1

Formát: INR M 0 0 1 1 0 1 0 0

Taktů:

Dob:

Ovlivňuje: Z, S, P, -, AC

Instrukce INR M přičte jedničku k obsahu paměťového místa adresovaného registrovým párem HL.

DCR

DECREMENT REGISTER (Dekrementuj registr)

Operace: (r) ← (r) - 1 ——— tab. 2.11

Formát: DCR r 0 0 D D D 1 0 1

(= A,B,C,D,E,H,L)

Taktů: 1

Dob: 5

Ovlivňuje: Z, S, P, -, AC

Instrukce DCR r odečte od specifikovaného registru jedničku.

DCR

DECREMENT MEMORY (Dekrementuj paměť)

Operace: (M,H&L) — (M,H&L) - 1

Formát: DCR M 0 0 1 1 0 1 0 1

Taktů: 3

Dob: 10

Ovlivňuje: Z, S, P, -, AC

Instrukce DCR M odečte jedničku od obsahu paměťového místa adresovaného registrovým párem HL.

5.2.2 Operace s šestnáctibitovými daty

Neovlivňují žádné příznakové bity.

INX

INCREMENT REGISTER PAIR (Inkrementuj registrový pár)

Operace: (rh)(rl) ← (rh)(rl) + 1 — tab. 2.12

Formát: INX rp — 0 0 r p 0 0 1 1

(rp = B,D,H,SP)

Taktů: 1

Dob: 5

Ovlivňuje: -----

Instrukce INX rp přičte jedničku k obsahu specifikovaného registrového páru.

DCX

DECREMENT REGISTER PAIR (Dekrementuj registrový pár)

Operace: (rh),(rl) ← (rh),(rl) - 1 — tab. 2.12

Formát: DCX rp — 0 0 r p 1 0 1 1

(rp = B,D,H,SP)

Taktů: 1

Dob: 5

Ovlivňuje: -----

Instrukce DCX rp odečte jedničku od obsahu specifikovaného registrového páru.

5.3 Skupina aritmetických operací (Arithmetic Group)

5.3.1 Přičtení osmibitového operandu k obsahu střadače

Ovlivňuje všechny příznaky.

ADD

ADD REGISTER TO A (Sečti registr se střadačem)

Operace: (A) ← (A) + (M,H×L) ——— tab. 2.11

Formát: ADD r 1,0,0,0,0,S,S,S

(r = A,B,C,D,E,H,L)

Taktů: 1

Dob: 4

Ovlivňuje: Z, S, P, CY, AC

Instrukce ADD r přičte obsah specifikovaného registru ke střadači a výsledek ponechá ve střadači.

ADD

ADD MEMORY TO A (Sečti paměť se střadačem)

Operace: (A) ← (A) + (M,H×L)

Formát: ADD M 1,0,0,0,0,1,1,0

Taktů: 2

Dob: 7

Ovlivňuje: Z, S, P, CY, AC

Instrukce ADD M sečte obsah paměťového místa, adresovaného registrovým párem HL se střadačem a výsledek ponechá ve střadači.

ADC

ADD REGISTER TO A WITH CARRY (Sečti registr se střadačem včetně CY)

Operace: (A) ← (A) + (r) + (CY) ——— tab. 2.11

Formát: ADC r 1,0,0,0,1,S,S,S

(r = A,B,C,D,E,H,L)

Taktů:

Dob:

Ovlivňuje: Z, S, P, CY, AC

Instrukce ADC r sečte obsah střadače a CY bitu s obsahem specifikovaného registru. Výsledek ponechá ve střadači.

ADC

ADD MEMORY TO A WITH CARRY (Sečti paměť se střadačem včetně CY)

Operace: (A) ← (A) + (M,H&L) + (CY)

Formát: ADC M | 1,0,0,0,1,1,1,0 |

Taktů: 2
Dob: 7
Ovlivňuje: Z, S, P, CY, AC

Instrukce ADC M sečte obsah střadače a bitu CY s obsahem paměťového místa adresovaného registrovým párem HL. Výsledek ponechá ve střadači.

ADI

ADD IMMEDIATE (Sečti přímý operand)

Operace: (A) ← (A) + (B2)

Formát: ADI data | 1,1,0,0,0,1,1,0 | data |

Taktů: 2
Dob: 7
Ovlivňuje: Z, S, P, CY, AC

Instrukce ADI data přičte data, obsažená ve druhé slabice instrukce, ke střadači a výsledek ponechá ve střadači.

ACI

ADD IMMEDIATE TO A WITH CARRY (Sečti přímý operand se střadačem včetně CY)

Operace: (A) ← (A) + (B2) + (CY)

Formát: ACI data | 1,1,0,0,1,1,1,0 | data |

Taktů: 2
Dob: 7
Ovlivňuje: Z, S, P, CY, AC

Instrukce ACI data přičte data, obsažená ve druhé slabice instrukce a hodnotu příznaku CY je střadači. Výsledek ponechá ve střadači.

5.3.2 Odečítání osmibitového operandu od obsahu střadače

Ovlivňuje všechny příznaky.

SUB — SUBTRACT REGISTER FROM A (Odečti registr od střadače)

Operace: (A) ← (A) - (r) —— tab. 2.11

Formát: SUB r — 1,0,0,1,0,S,S,S

(r = A,B,C,D,E,H,L)

Taktů: 1
Dob: 4
Ovlivňuje: Z, S, P, CY, AC

Instrukce SUB r odečte obsah specifikovaného registru od střadače a výsledek ponechá ve střadači.

Pro operaci se používá dvojkový doplněk a výsledná hodnota CY je negována.

SUB — SUBTRACT MEMORY FROM A (Odečti paměť od střadače)

Operace: (A) ← (A) - (M,H,L)

Formát: SUB M — 1,0,0,1,0,1,1,0

Taktů: 2
Dob: 7
Ovlivňuje: Z, S, P, CY, AC

Instrukce SUB M odečte obsah paměťového místa, adresovaného registrovým párem HL, od střadače a výsledek ponechá ve střadači.

Pro operaci se používá dvojkový doplněk a výsledná hodnota CY je negována.

SBB — SUBTRACT REGISTER FROM A WITH BORROW (Odečti registr od střadače včetně výpůjčky)

Operace: (A) ← (A) - (r) - (CY) ——— tab. 2.11

Formát: SBB r — 1 0 0 1 1 S S S

(r = A,B,C,D,E,H,L)

Taktů: 1

Dob: 4

Ovlivňuje: Z, S, P, CY, AC

Instrukce SBB r odečte obsah specifikovaného registru a CY od střadače. Výsledek ponechá ve střadači. Obsah registru r zůstává nezměněn.

Pro operaci používá dvojkový doplněk a výsledná hodnota CY je negována.

SBB — SUBTRACT MEMORY FROM A WITH BORROW (Odečti paměť od střadače včetně výpůjčky)

Operace: (A) ← (A) - (M,H&L) - (CY)

Formát: SBB M — 1 0 0 1 1 1 1 0

Taktů:

Dob:

Ovlivňuje: Z, S, P, CY, AC

Instrukce SBB M odečte obsah paměťového místa, adresovaného registrovým párem HL, a CY od střadače. Výsledek ponechá ve střadači. Obsah paměti zůstává nezměněn.

Pro operaci používá dvojkový doplněk a výsledná hodnota CY je negována.

SUI

SUBTRACT IMMEDIATE FROM A (Odečti /přímý operand/)

Operace: (A) ← (A) - (B2)

Formát: SUI data

1 1 0 1 0 1 1 0	data

Taktů: 2
Dob: 7
Ovlivňuje: Z, S, P, CY, AC

Instrukce SUI data odečte data, obsažená ve druhé slabice instrukce, od střadače a výsledek ponechá ve střadači.

Pro operaci používá dvojkový doplněk a výsledná hodnota CY je negována.

SBI

SUBTRACT IMMEDIATE FROM A WITH BORROW (Odečti včetně výpůjčky /přímý operand/)

Operace: (A) ← (A) - (B2) -(CY)

Formát: SBI data

1 1 0 1 1 1 1 0	data

Taktů:
Dob:
Ovlivňuje: Z, S, P, CY, AC

Instrukce SBI data odečti data, obsažená ve druhé slabice instrukce, od střadače a výsledek ponechá ve střadači.

Pro operaci používá dvojkový doplněk a výsledná hodnota CY je negována.

5.3.3 Přičítání šestnáctibitových operandů k registrovému páru HL

Ovlivňuje pouze bit CY.

DAD — DOUBLE ADD REGISTER PAIR TO HL (Dvojslabikové sčítání)

Operace: (H&L) ← (H&L) + (rh)(rl) — tab. 2.12

Formát: DAD rp — 0 0 r p 1 0 0 1

(rp = B,D,H,SP)

Taktů: 3

Dob: 10

Ovlivňuje: ----- CY

Instrukce DAD rp přičte obsah specifikovaného registrového páru k registrovému páru HL. Výsledek ponechá v HL.

5.3.4 Porovnání obsahu střadače s obsahem registrů a paměti

Ovlivňuje všechny příznakové bity.

CMP — COMPARE REGISTER WITH A (Porovnej registr se střadačem)

Operace: (A) - (r) — tab. 2.11

Formát: CMP r — 1 0 1 1 1 S S S

(r = A,B,C,D,E,H,L)

Taktů: 1

Dob: 4

Ovlivňuje: Z, S, P, CY, AC

Instrukce CMP r porovná obsah specifikovaného registru s obsahem střadače a podle výsledku nastaví příznakové bity Z a CY. Obsah střadače i registru zůstává nezměněn. Porovnání se provádí odečtením obsahu střadače a registru s použitím dvojkového doplňku. Pro binární čísla bez znaménka platí:

A = r => Z = 1 CY = 0

A > r => Z = 0 CY = 0

A < r => Z = 0 CY = 1

Tedy:

$Z = 1 \Rightarrow A = r$
$Z = 0 \Rightarrow A \neq r$
$CY = 1 \Rightarrow A < r$
$CY = 0 \Rightarrow A \geq r$

CMP — COMPARE MEMORY WITH A (Porovnej paměť se střadačem)

Operace: (A) - (M,HéL)

Formát: CMP M | 1 0 1 1 1 1 1 0 |

Taktů: 2
Dob: 7
Ovlivňuje: Z, S, P, CY, AC

Instrukce CMP M porovná obsah paměťového místa M, adresovaného registrovým párem HL, s obsahem střadače a podle výsledku nastaví příznakové bity. Význam Z a CY je shodný s instrukcí CMP r.

CPI — COMPARE IMMEDIATE WITH A (Porovnej přímý operand se střadačem)

Operace: (a) - (B2)

Formát: CPI data | 1 1 1 1 1 0 0 1 | data |

Taktů: 2
Dob: 7
Ovlivňuje: Z,S,P,CY,AC

Instrukce CPI data porovná obsah střadače s daty obsaženými v druhé slabice instrukce a podle výsledku nastaví příznakové bity Z a CY. Obsah střadače zůstává nezměněn. Význam bitů CY a Z je shodný jako u instrukce CMP r.

5.4 Skupina logických operací (Logical Group)

5.4.1 Podskupina logických součinů

Ovlivňuje všechny příznakové bity. (CY = 0, AC = 0)

ANA — LOGICAL AND WITH ACCUMULATOR (Operace AND mezi střadačem a registrem)

Operace: (A) ← (A)∧(r) — tab. 2.11

Formát: ANA r — 1 0 1 0 0 S S S

(r = A,B,C,D,E,H,L

Taktů: 1

Dob: 4

Ovlivňuje: Z,S,P,CY = 0,AC = 0

Instrukce ANA r provede operaci AND mezi střadačem a obsahem specifikovaného registru. Výsledek uloží ve střadači, nastaví CY = 0, AC = 0.

ANA — LOGICAL AND MEMORY WITH ACCUMULATOR (Operace AND střadače s pamětí)

Operace: (A) ← (A)∧(M,H&L)

Formát: ANA M — 1 0 1 0 0 1 1 0

Taktů: 2

Dob: 7

Ovlovňuje: Z,S,P,CY = 0, AC = 0

Instrukce ANA M provede operaci AND mezi střadačem a obsahem paměťového místa, adresovaného registrovým párem HL. Výsledek uloží ve střadači a nastaví CY = 0, AC = 0.

ANI — AND IMMEDIATE WITH ACCUMULATOR (Operace AND mezi střadačem a přímým operandem)

Operace: (A) ← (A)∧(B2)

Formát: ANI data

1 1 1 0 0 1 1 0	data

Taktů: 2

Dob: 7

Ovlivňuje: Z,S,P,CY = 0,AC = 0

Instrukce ANI data provede operaci AND mezi daty, obsaženými ve druhé slabice instrukce, a obsahem střadače. Výsledek uloží ve střadači a nastaví CY = 0, AC = 0.

5.4.2 Podskupina logických součtů

Ovlivňuje všechny příznakové bity, nastaví CY = 0, AC = 0.

ORA — INCLUSIVE OR WITH ACCUMULATOR (Operace OR mezi střadačem a registrem)

Operace: (A) ← (A)∨(r) —— tab. 2.11

Formát: ORA r

1 0 1 1 0 S S S

(r = A,B,C,D,E,H,L

Taktů: 1

Dob: 4

Ovlivňuje: Z,S,P,CY = 0, AC = 0

Instrukce ORA r provede operaci OR mezi obsahem specifikovaného registru a střadačem. Výsledek uloží do střadače. Obsah registru se nezmění. Nastaví CY =0, AC = 0.

Pozn.: ORA A nastaví CY = 0 a obsah střadače se nezmění.

ORA — INCLUSIVE OR MEMORY WITH ACCUMULATOR (Operace OR mezi střadačem a pamětí)

Operace: (A) ← (A) ∨ (M,H&L)

Formát: ORA M

1	0	1	1	0	1	1	0

Taktů: 2
Dob: 7
Ovlivňuje: Z,S,P,CY = 0,AC = 0

Instrukce ORA M provede operaci OR mezi střadačem a obsahem paměťového místa M adresovaného registrovým párem HL. Obsah paměťového místa M se nezmění. Výsledek uloží do střadače, nastaví CY = 0, AC = 0.

ORI — INCLUSIVE OR IMMEDIATE (Operace OR /přímý operand/)

Operace: (A) ← (A) ∨ (B2)

Formát: ORI data

1	1	1	1	0	1	1	0	data

Taktů: 2
Dob: 7
Ovlivňuje: Z,S,P,CY = 0,AC = 0

Instrukce ORI data provede operaci OR mezi obsahem střadače a daty obsaženými ve druhé slabice instrukce. Výsledek uloží ve střadači. Nastaví CY = 0, AC = 0.

5.4.3 Podskupina exklusivních logických součtů

Ovlivňuje všechny příznakové bity. (CY = 0, AC = 0)

XRA — EXCLUSIVE OR WITH ACCUMULATOR (Operace EX-OR /neekvivalence/ se střadačem)

Operace: (A) ← (A) ⊻ (r) — tab. 2.11

Formát: XRA r

1	0	1	0	1	S	S	S

(r = A,B,C,D,E,H,L)

Taktů: 1
Dob: 4
Ovlivňuje: Z,S,P,CY = 0, AC = 0

Instrukce XRA r provede operaci EX-OR mezi střadačem a obsahem specifikovaného registru. Výsledek uloží do střadače. Obsah registru r zůstává nezměněn. Nastaví CY = 0, AC = 0.

XRA — EXCLUSIVE OR MEMORY WITH ACCUMULATOR (Operace EX-OR paměti se střadačem)

Operace: (A) ← (A)⊻(M, H&L)

Formát: XRA M

1	0	1	0	1	1	1	0

Taktů: 2
Dob: 7
Ovlivňuje: Z,S,P,CY = 0,AC = 0

Instrukce XRA M provede operaci EX-OR mezi střadačem a obsahem paměťového místa M, adresovaného registrovým párem HL. Výsledek uloží do střadače. Obsah paměťového místa M se nezmění. Nastaví CY = 0, AC = 0.

XRI — EXCLUSIVE OR IMMEDIATE WITH ACCUMULATOR (Operace EX-OR mezi střadačem a přímým operandem)

Operace: (A) ← (A)⊻(B2)

Formát: XRI data

1	1	1	0	1	1	1	0	data

Taktů: 2
Dob: 7
Ovlivňuje: Z,S,P,CY = 0,AC = 0

Instrukce XRI data provede operaci EX-OR mezi střadačem a daty obsaženými v druhé slabice instrukce. Výsledek uloží do střadače. Nastaví CY = 0, AC = 0.

5.4.4 Podskupina speciálních logických instrukcí

DAA — DECIMAL ADJUST ACCUMULATOR (Dekadická korekce střadače)

Operace: Dekadická korekce střadače

Formát: DAA | 0 0 1 0 0 1 1 1 |

Taktů: 1
Dob: 4
Ovlivňuje: Z,S,P,CY,AC

Instrukce koriguje obsah střadače tak, aby tvořil dvě čtyřbitová BCD čísla (binárně kódovaná dakadická čísla). DAA jako jediná instrukce testuje hodnotu příznakového bitu pomocného přenosu (AC). Používá se při sčítání dakadických čísel a při mnohoslabikových aritmetických operacích bývá zařazena bezprostředně po operaci sčítání dokud je příznakový bit AC správně nastaven. Činnost instrukce DAA je následující:

1) Je-li hodnota nižších čtyř bitů střadače (A) větší než 9 nebo je AC = 1, DAA přičte ke střadači 6.

2) Je-li hodnota vyšších čtyř bitů střadače (A) větší než 9 nebo je CY = 1, DAA přičte 6 k těmto vyšším čtyřem bitům střadače.

CMA — COMPLEMENT ACCUMULATOR (Negace střadače)

Operace: $(A) \leftarrow (\overline{A})$

Formát: CMA | 0 0 1 0 1 1 1 1 |

Taktů: 1
Dob: 4
Ovlivňuje: -----

Instrukce CMA neguje každý bit střadače. Po provedení instrukce CMA získáme jednotkový doplněk, přičtením jedničky pak doplněk dvojkový.

STC SET CARRY (Nastav CY)

Operace: (CY) ← 1

Formát: STC 0 0 1 1 0 1 1 1

Taktů: 1

Dob: 4

Ovlivňuje: ----- CY -

Instrukce STC nastaví CY = 1.

CMC COMPLEMENT CARRY (Negace CY)

Operace: (CY) ← $(\overline{CY})$

Formát: CMC 0 0 1 1 1 1 1 1

Taktů: 1

Dob: 4

Ovlivňuje: ----- CY -

Instrukce CMC provede negaci bitu CY.

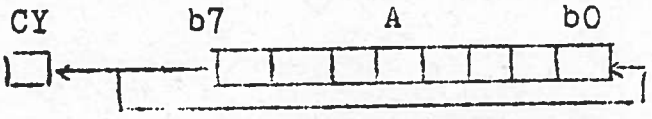
5.5 Skupina rotací a posunů (Rotate and Schift Group)

Ovlivňuje pouze bit CY

5.5.1 Rotace bez přenosového bitu

RLC ROTATE ACCUMULATOR LEFT (Rotace vlevo a do CY)

Operace:



Formát: RLC 0 0 0 0 0 1 1 1

Taktů: 1

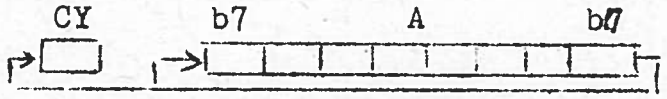
Dob: 4

Ovlivňuje: ----- CY -

Instrukce RLC posune obsah střadače o jeden bit vlevo. Bit b7 přesune do b0 a do CY.

RRC

ROTATE ACCUMULATOR RIGHT (Rotace vpravo a do CY)



Operace:

Formát: RRC 0,0,0,0,1,1,1,1

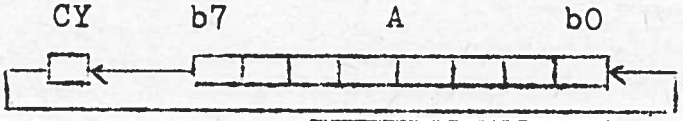
Taktů: 1
Dob: 4
Ovlivňuje: ---- CY -

Instrukce RRC posune obsah střadače (A) o jeden bit vpravo. Bit b0 přesune do b7 a do CY.

5.5.2 Rotace včetně přenosového bitu

RAL

ROTATE LEFT THROUGH CARRY (Rotace vlevo přes CY)



Operace:

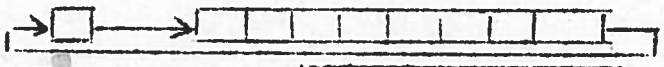
Formát: RAL 0,0,0,1,0,1,1,1

Taktů: 1
Dob: 4
Ovlivňuje: --- CY -

Instrukce RAL posune bit b7 střadače (A) do CY, CY do b0 a celý obsah střadače o jeden bit vlevo.

RAR

ROTATE RIGHT THROUGH CARRY (Rotace vpravo přes CY)



Operace:

Formát: RAR 0,0,0,1,1,1,1,1

Taktů: 1
Dob: 4
Ovlivňuje: --- CY -

Instrukce RAR posune bit b0 střadače (A) do CY, CY do b7 a celý obsah střadače o jeden bit vpravo.

5.6 Operace se zásobníkem a s ukazatelem zásobníku

Neovlivňuje žádné příznakové bity.

PUSH — PUSH REGISTER PAIR (Ulož do zásobníku registrový pár)

Operace: (M,SP-1) ← (rh)
(M,SP-2) ← (rl)

tab. 2.12

Formát: PUSH rp

1	1	r	p	0	1	0	1

(rp = B,D,H)

Taktů: 3
Dob: 11
Ovlivňuje: -----

Instrukce PUSH uloží obsah registrového páru do zásobníku. Instrukce PUSH dekrementuje ukazatel zásobníku (SP) a uloží vyšší slabiku zvoleného registrového páru do zásobníku (na adresu danou SP). Poté znovu dekrementuje SP a uloží do zásobníku nižší slabiku registrového páru. Obsah ukládaného registrového páru zůstává nezměněn.

Instrukce PUSH a POP slouží nejčastěji k ukládání a obnově těch registrů, jejichž obsah se nesmí během provádění podprogramů znehodnotit. Na začátku podprogramu uložíme instrukcí PUSH obsah potřebných registrů do zásobníku a před návratem do hlavního programu jejich obsah instrukcí POP obnovíme.

PUSH — PUSH PROCESOR STATUS WORD (Ulož do zásobníku PSW)

Operace: (M,SP-1) ← (A)
(M, SP-2) ← (F)

Formát: PUSH PSW | 1 1 1 1 0 1 0 1 |

Taktů: 3
Dob: 11
Ovlivňuje: -----

Instrukce PUSH PSW uloží do zásobníku (obdobně jako PUSH rp) obsah střadače a registru příznaků. Registr příznaků uloží ve formě | S | Z | 0 | AC | 0 | P | 1 | CY |

5.6.2 Obnovení (vyzvednutí) obsahu registrových párů

POP

POP REGISTER PAIR (Obnova obsahu registrů)

Operace: (rl) ← (M,SP)
(rh) ← (M,SP + 1) ——— tab. 2.12

Formát: POP rp | 1 1 r p 0 0 0 1 |

(rp = B,D,H

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce POP rp vyzvedne dvě položky ze zásobníku a uloží je do specifikovaného registrového páru.

POP

POP PROCESOR STATUS WORD (Obnova PSW)

Operace: (F) ← (M,SP)
(A) ← (M,SP + 1)

Formát: POP PSW | 1 1 1 1 0 0 0 1 |

Taktů: 3
Dob: 10
Ovlivňuje: Z,S,P,CY,AC

Instrukce POP PSW obnoví obsah střadače a registru příznaků daty uchovanými v zásobníku.

5.6.3 Nastavení obsahu ukazatele zásobníku (SP) a zásobníku

Pro nastavení ukazatele zásobníku lze použít, kromě instrukcí následujících, též instrukci LXI rp - viz odst. 5.1.2.

SPHL MOVE HL TO SP (Přesuň HL do SP)

Operace: (SP) ← (H&L)

Formát: SPHL | 1,1,1,1,1,0,0,1 |

Taktů: 1
Dob: 5
Ovlivňuje: -----

Instrukce SPHL přesune obsah registrového páru HL do ukazatele zásobníku SP. Obsah HL se nezmění.

XTHL EXCHANGE TOP OF STACK WITH H AND L (Zaměň vrchol zásobníku a HL)

Operace: (L) ← (M,SP)
(H) ← (M,SP + 1)

Formát: XTHL | 1,1,1,0,0,0,1,1 |

Taktů: 5
Dob: 18
Ovlivňuje: -----

Instrukce XTHL vzájemně zamění vrchol zásobníku s obsahem registrového páru HL.

XTHL zamění obsah registru L s paměťovým místem adresovaným ukazatelem zásobníku SP a obsah registru H s paměťovým místem s adresou SP + 1.

Ukazatel zásobníku SP zůstává nezměněn.

5.7 Skupina instrukcí větvících program (Branch Control Group)

Neovlivňuje žádné příznakové bity.

5.7.1 Instrukce skoků

JMP — JUMP UNCONDITIONAL (Nepodmíněný skok)

Operace: (PC) ← (B3,B2)

Formát: JMP adr | 1 1 0 0 0 0 1 1 | adr LSB | adr MSB |

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce JMP adr nastaví čítač instrukcí (PC) a adresu danou druhou a třetí slabikou instrukce. Program pokračuje na této adrese.

J"CCC" — JUMP ON CONDITION (Podmíněný skok na adresu je-li splněna podmínka)

Operace: (PC) ← (B3,B2) if ___ — viz tabulka

Formát: J "CCC" adr | 1 1 C C C 0 1 0 | adr LSB | adr MSB |

Taktů: 3
Dob: 10
Ovlivňuje: -----

Program pokračuje na adrese, dané druhou a třetí slabikou instrukce, jestliže je splněna podmínka. Dle typu podmínky je potom i specifikován název instrukce. Pokud není podmínka splněna, pokračuje program následující instrukcí.

Podmínka	CCC	Název instrukce	Poznámka
Z = 0	000	JNZ	Výsledek ≠ 0
Z = 1	001	JZ	Výsledek = 0
CY = 0	010	JNC	Žádný přenos
CY = 1	011	JC	Přenos
P = 0	100	JPO	Lichá parita
P = 1	101	JPE	Sudá parita
S = 0	110	JP	Výsledek kladný
S = 1	111	JM	Výsledek záporný

Poznámka: Instrukce JPE, JPO jsou užitečné při kontrole parity vstupních dat. Instrukce IN ale neovlivňuje žádný z příznakových bitů. Tuto obtíž odstraníme např. přičtením 00H ke střadači. Tato operace nastaví příznakové bity a neovlivní obsah střadače.

PCHL — MOVE HL TO PROGRAM COUNTER (Přesuň HL do čítače instrukcí)

Operace: (PC) ← (M,HL)

Formát: PCHL

1	1	1	0	1	0	0	1

Taktů: 1
Dob: 5
Ovlivňuje: -----

Instrukce PCHL přesune obsah registrového páru HL do čítače instrukcí (PC). Program pokračuje na adrese určené obsahem HL (nepřímý skok). (H přesune do vyšší poloviny PC, L do nižší).

5.7.2 Instrukce volání podprogramu

CALL — CALL UNCONDITIONAL (Nepodmíněné volání podprogramu)

Operace: (M,SP) ← (PC)
(PC) ← (B3,B2)

Formát: CALL adr

1	1	0	0	1	1	0	1	adr LSB	adr MSB

Taktů: 5
Dob: 17
Ovlivňuje: -----

Instrukce CALL je kombinací instrukcí PUSH a JMP. CALL uloží obsah čítače instrukcí (PC) (t.j. adresu instrukce následující v paměti po CALL) do zásobníku a čítač instrukcí nastaví na adresu danou druhou a třetí slabikou instrukce.

C"CCC" CALL ON CONDITION (Volání podprogramu, je-li splněna podmínka)

Operace: (M,SP) ← (PC)
(PC) ← (B3,B2)if ——— viz tabulka

Formát: C "CCC" adr | 1 1 C C C 1 0 0 | adr LSB | adr MSB |

Taktů: 3 nebo 5
Dob: 11 nebo 17
Ovlivňuje: -----

Instrukce C "CCC" adr testují, zda je splněna specifikovaná podmínka, pokud ano, uloží obsah čítače instrukcí (PC) do zásobníku a čítač instrukcí nastaví na adresu danou druhou a třetí slabikou instrukce. Pokud podmínka splněna není, pokračuje program instrukcí následující po C "CCC" adr.

Podmínka	CCC	Název instrukce	Poznámka
Z = 0	000	CNZ	Výsledek ≠ 0
Z = 1	001	CZ	Výsledek = 0
CY = 0	010	CNC	Žádný přenos
CY = 1	011	CC	Přenos
P = 0	100	CPO	Lichá parita
P = 1	101	CPE	Sudá parita
S = 0	110	CP	Výsledek kladný
S = 1	111	CM	Výsledek záporný

Pozn.: Instrukce CPE, CPO jsou užitečné při kontrole parity vstupních dat. Instrukce IN ale neovlivňuje žádný z příznakových bitů. Tuto obtíž odstraníme např. přičtením 00H ke střadači. Tato operace nastaví příznakové bity a neovlivní obsah střadače.

5.7.3 Instrukce návratu z podprogramu

RET — RETURN UNCONDITIONAL FROM SUBROUTINE (Nepodmíněný návrat z podprogramu)

Operace: (PC) ← (M,SP)

Formát: RET

1,1	0,0,1	0,0,1

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce RET vyzvedne dvě slabiky ze zásobníku a uloží je do čítače instrukcí (PC). Program pokračuje na této nové adrese.

Instrukce RET (a instrukce R "CCC") se používají spolu s instrukcemi CALL (C "CCC").

R"CCC" — RETURN ON CONDITION (Návrat k podprogramu je-li splněna podmínka)

Operace: (PC) ← (M,SP) if ——— viz tabulka

Formát: R "CCC" adr

1,1	C,C,C	0,0,0

Taktů: 1 nebo 3
Dob: 5 nebo 11
Ovlivňuje: -----

Instrukce R "CCC" adr testují, zda je splněna specifikovaná podmínka, pokud ano, vyzvednou dvě slabiky ze zásobníku a uloží je do čítače instrukcí (PC). Program pokračuje na nové adrese. V opačném případě program pokračuje instrukcí následující po R "CCC" adr.

Podmínka	CCC	Název instrukce	Poznámka
Z = 0	000	RNZ	Výsledek ≠ 0
Z = 1	001	RZ	Výsledek = 0
CY = 0	010	RNC	Žádný přenos
CY = 1	011	RC	Přenos
P = 0	100	RPO	Lichá parita

pokračování

pokračování

Podmínka	CCC	Název instrukce	Poznámka
P = 1	101	RPE	Sudá parita
S = 0	110	RP	Výsledek kladný
S = 1	111	RM	Výsledek záporný

5.7.4 Instrukce přerušení

RST RESTART (Volání podprogramu na pevně definované adrese)

Operace: (M,SP) ← (PC)
(PC) ← (NNN)*8

Formát: RST n | 1 1 N N N 1 1 1 |

(n = 0,1,2,3,4,5,6,7)

Taktů: 3
Dob: 11
Ovlivňuje: -----

0 0 0 0 0 0 0 0 0 0 N N N 0 0 0 PC - čítač instrukcí

Instrukce RST n je určena převážně pro volání podprogramu při přerušení. RST n uloží obsah čítače instrukcí do zásobníku a nastaví čítač instrukcí na jednu k osmi adres dle n. Program pokračuje dále na této adrese. Přiřazení adresy dle RST n udává následující tabulka.

RST n	NNN	Čítač instrukcí (PC)						
		BIN.					HEX.	DEK.
0	000	0000	0000	00	000	000	00	0
1	001	↑	↑	↑	001	↑	08	8
2	010	|	|	|	010	|	10	16
3	011	|	|	|	011	|	18	24
4	100	|	|	|	100	|	20	32
5	101	|	|	|	101	|	28	40
6	110	↓	↓	↓	110	↓	30	48
7	111	0000	0000	00	111	000	38	56

5.8 Instrukce vstupu a výstupu (I/O Control Group)

Neovlivňují žádné příznakové bity

IN — INPUT FROM PORT (Čtení vstupní brány)

Operace: (A) ← (I/O,B2) — adresa brány

Formát: IN port

1	1	0	1	1	0	1	1	port

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce IN port přesune do střadače obsah vstupní brány adresované druhou slabikou instrukce. Adresu brány vysílá procesor na adresových vodičích A0 ÷ A7 a opakovaně na A8 ÷ A15.

OUT — OUTPUT TO PORT (Zápis do výstupní brány)

Operace: (I/O,B2) ← (A) — adresa brány

Formát: OUT port

1	1	0	1	0	0	1	1	port

Taktů: 3
Dob: 10
Ovlivňuje: -----

Instrukce OUT port přesune obsah střadače do výstupní brány adresované druhou slabikou instrukce. Procesor vysílá adresu brány na adresových vodičích A0 ÷ A7 a opakovaně na A8 ÷ A15.

5.9 Řídicí instrukce (Machine Control Group)

Neovlivňují žádné příznakové bity.

NOP

NO OPERATION (Prázdná operace)

Operace: PRÁZDNÁ OPERACE

Formát: NOP

0	0	0	0	0	0	0	0

Taktů: 1
Dob: 4
Ovlivňuje: -----

Instrukce NOP vykoná prázdnou operaci a neovlivňuje žádné příznakové bity. Používá se jako časová výplň v časovacích smyčkách programu.

HLT

HALT (Stop)

Operace: Stop (Zastavení činnosti procesoru)

Formát: HLT

0	1	1	1	0	1	1	0

Taktů: 1
Dob: 7
Ovlivňuje: -----

Instrukce HLT zastaví činnost procesoru. V čítači instrukcí (PC) je adresa následující instrukce, ostatní registry i příznakové klopné obvody zůstanou nezměněny.

Stav HALT může procesor opustit:

1) po aktivaci signálu RESET (nulování)
2) po aktivaci signálu HOLD (DMA přenos). Po skončení signálu HOLD se ale procesor vrací zpět do stavu HALT
3) přijetím žádosti o přerušení (INT).

Před použitím instrukce HLT je tedy nutné povolit přerušení (instrukcí EI).

Instrukce HLT se používá především tehdy, má-li procesor očekávat žádost o přerušení z periferního zařízení a nic dalšího nemá v daném okamžiku k řešení.

DI

DISABLE INTERRUPT (Přerušení zakázáno)

Operace: INTE ← 0

Formát: DI

1	1	1	1	0	0	1	1

Taktů: 1
Dob: 4
Ovlivňuje: -----

Přerušovací systém mikroprocesoru je zablokován (INTE = 0) bezprostředně po provedení instrukce DI (je rovněž blokován automaticky po přijetí žádosti o přerušení). Přerušení je možné znovu povolit instrukcí EI.

EI

ENABLE INTERRUPT (Přerušení povoleno)

Operace: INTE ← 1 (až po první následující instrukci!)

Formát: EI

1	1	1	1	1	0	1	1

Taktů: 1
Dob: 4
Ovlivňuje: -----

Instrukce EI odblokuje přerušovací systém mikroprocesoru (INTE = 1) po první instrukci následující po EI. Povolení přerušení je o jednu instrukci zpožděno proto, aby byl možný správný návrat z podprogramů přerušení do hlavního programu.

Bit → / ↓	5	0	1	0	1	0	1	0	1
	6	0	0	1	1	0	0	1	1
	7	0	0	0	0	1	1	1	1
4321	HEX. → / ↓	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DCI	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^ (↑)	n	~
1111	F	SIm	US	/	?	O	- (←)	o	DEL

Tab. 5.1 Kód ASCII.

Mikroprocesorová technika I a II

Otázky ke zkoušce PGS

1) Charakteristika mikropočítače, blokové schéma, struktura mikropočítačem řízeného systému
2) Centrální jednotka a registry mikropočítače
3) Paměti v mikropočítačových systémech
4) Programové vybavení mikropočítače
5) Jazyky mikropočítačových systémů
6) Struktura mikropočítače s mikroporocesorem 8080A
7) Struktura mikroprocesoru 8080A
8) Časování mikroprocesoru 8080A, průběhy HOLD, HALT, VAIT, uvedení mikroprocesoru do provozu
9) Rozdělení instrukcí v instrukčním kódu 8080A
10) Struktura a rozdělení pamětí a obvodů V/V
11) Blokové schéma obvodu 8255
12) Blokové schéma časovače 8253
13) Programování 8255
14) Programování 8253
15) Mikropočítačová stavebnice MCS80
16) Příklady použití prvků mikropočítačové stavebnice MCS80
17) Vytváření programových smyček
18) Charakteristika podprogramu a příklady jeho použití

L I T E R A T U R A :

/1/ Lojík,V.-Sýkora,J.:Číslicová a impulsová technika I. Skriptum ČVUT-FEL, Praha 1981

/2/ Lojík,V.: Číslicová a impulsová technika II. Skriptum ČVUT-FEL, Praha 1984

/3/ CHod,J.: Číslicová a impulsová technika I. - cvičení, Skriptum ČVUT-FEL, Praha 1983

/4/ Integrované obvody, tranzistory, diody, triaky, optoelektronické součástky. Katalog fy. TESLA Rožnov, Rožnov 1983

/5/ Bernard,M.J.-Hugon,J.: Od logických obvodů k mikroprocesorům I. SNTL, Praha 1982

/6/ Bernard,M.J.-Hugon,J.: Od logických obvodů k mikroprocesorům II. SNTL, Praha 1983

/7/ Bernard,M.J.-Hugon,J.: Od logických obvodů k mikroprocesorům III. SNTL, Praha 1983

/8/ Bernard,M.J.-Hugon,J.: Od logických obvodů k mikroprocesorům IV. SNTL, Praha 1984

/9/ Pütz,J. a kolektiv: Úvod do číslicové techniky SNTL, Praha 1983

/10/ Katalog elektronických součástek, konstrukčních dílů, bloků a přístrojů 1983 - 1984, 1. díl

/11/ Mikropočítače a mikroprocesory. Příloha ARA 1/82 až ARA 10/82. Naše vojsko, Praha 1982

/12/ Čech,M.-Jandoš,J.: Úvod do mikropočítačů. Výběr informací z organizační a výpočetní techniky. Kancelářské stroje, Praha 1983

/13/ Jiřička,K.: Analogově-číslicové převodníky pro systémy s mikroprocesory. Sdělovací technika 2/1983. SNTL, Praha 1983

/14/ Mirtes,B.-Sýkora,Z.: Číslicové měření a zpracování analogových veličin. SNTL 1971

/15/ Sobotka,Z.: Přehled číslicových systémů. SNTL, Praha 1981

/16/ Budínský,J.: Polovodičové paměti a jejich použití. SNTL, Praha 1977

/17/ Mikroprocesor 8080 a základní obvody. Skriptum ČVUT-ČSVTS, Praha 1982

/18/ Základní instrukce mikroprocesoru 8080. Skriptum ČVUT-ČSVTS, Praha 1982

/19/ Dlabola,F.-Starý,J.: Systém s mikroprocesory a přenos dat. NADAS, Praha 1984

/20/ Dolejší,A.: Návod na použitie ŠMS - VÚVT Manuál mikropočítače ŠMS - VÚVT. Žilina 1983

/21/ Dolejší,A.: Školský mikropočítačový systém VÚVT. Datasystém, Bratislava 1982

/22/ Burger,J.: Cvičení programovanie na ŠMS - VÚVT. DT ČSVTS Bratislava. Bratislava 1983

/23/ Zděnek,J.: Číslicová a impulsová technika - cvičení. Skriptum ČVUT-FEL, Praha 1983

/24/ Intel: The 8080/8085 Microprocesor Bo k. John Wiley, New York 1980

/25/ Dědina,B.-Valášek,P.: Mikroprocesory a mikropočítače. SNTL, Praha 1983

/26/ Leventhal,L.A.: 8080A/8085 Assembly Language Programming. Osborne/McGraw-Hill, 1978

/27/ Sobotka,Z.-Starý,J.: Mikropočítače. ÚTERS TESLA VÚST, Praha 1980

/28/ Jeníček,Č.-Nohel,J.-Santus,A.: Programování v asembleru. ČSVTS, Praha 1982

/29/ Kraus,V.: Instrukční soubor mikroprocesoru 8080. ČSVTS, ČVUT-FEL, Praha 1983