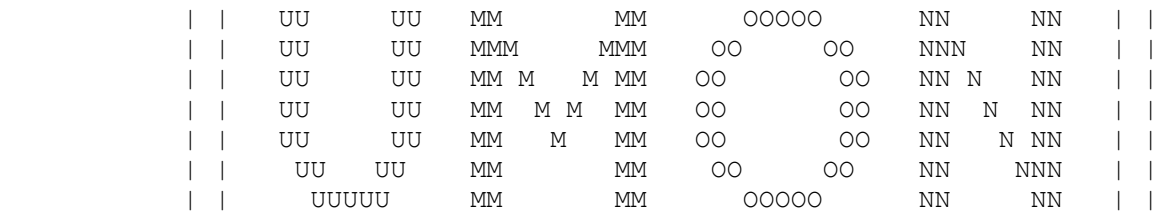
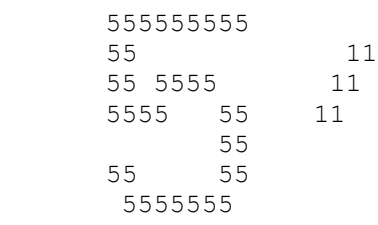




UŽIVATELSKÁ PŘÍRUČKA

Verze 2.7 a 3.1

Promis, Aloisina výšina 631, 46005 Liberec

P Ř E D M L U V A

Tento manuál obsahuje návod na použití monitoru UMON-51 jako vývojového prostředku pro vývoj a ladění uživatelského programového vybavení pro řídící systémy na bázi MCS-8051/52.

Je zaměřen na popis činnosti při aplikaci typu MCS-51. Kde je to nutné, je věnován popisu aplikace člena MCS-52 zvláštní odstavec. Termín 8051 znamená odkaz do rodiny MCS-51, termín 8052 odkaz do rodiny MCS-8052.

Manuál obsahuje popis příkazů a funkcí monitoru UMON-51 a dále krátké příklady použití jednotlivých příkazů, případně i jednoduché příklady aplikačních programů.

V textu je přihlédnuto ke zvláštnostem plynoucím z implementace na řídící systém ELIS-51, jehož vývoj a výrobu zabezpečuje státní podnik ELITRON Liberec.

Manuál neobsahuje popis instrukcí 8051/8052 ani popis funkce hardware 8051/8052. Nezabývá se ani popisem činnosti ELIS-51.

Tento manuál je rozdělen to následujících částí:

Část 1 Úvod do činnosti a funkcí monitoru UMON-51 a specifikace hardware pro umožnění jeho správné funkce.

Část 2 Obsahuje popis příkazů monitoru, příklady použití jednotlivých příkazů a popis reakcí monitoru zobrazovaných na operátorskou konzolu (display), případně potřebné zásahy operátora z klávesnice. Operace jsou rozděleny do čtyř základních skupin, a sice zapnutí zdroje (power up), reset a vstup příkazu; příkazy pro vykonávání uživatelského programu; příkazy pro přístup do paměti; a příkazy pro komunikaci s uživatelem.

Část 3 Popisuje některá omezení plynoucí ze způsobu činnosti UMON-51 nebo ELIS-51, mapování dostupné paměti na systému ELIS-51.

Část 4 Dává informace o programování 8051 a návod na využití některých služeb monitoru přístupných uživateli.

Příloha Seznam chybových hlášení produkovaných monitorem UMON-51 s krátkým popisem jejich příčin.

N Ě K O L I K V Ě T N A Ú V O D

Monitor UMON-51 je Vám nabízen jako prostředek pro usnadnění práce se systémy postavenými na bázi MCS-51. Protože autoři předpokládají, že Vás bude provázet i při prvním seznámení s mikrokontroléry 8051/8052, přečtěte si prosím následující řádky.

Hlavním kritériem při psaní monitoru UMON-51 bylo pro nás hledisko uživatele, jednoduchost a srozumitelnost jednotlivých příkazů, přehlednost a pochopitelnost chybových hlášení a schopnost monitoru nabídnout uživateli co nejvíce funkcí, aby pokud možno v co největší míře nahradil složitější vývojové prostředky.

Z těchto důvodů je UMON-51 systémem příkazů kompatibilní se systémem příkazů ICE-51 firmy Intel, neboť se autoři domnívají, že právě způsob komunikace s tímto emulátorem nejvíce vyhovuje výše uvedeným uživatelským hlediskům, nehledě k té skutečnosti, že emulátory Intel jsou v naší zemi dobře známy a dosti dlouho využívány. Firma Intel přitom dbá na dědičnost svých zařízení, takže nevznikají problémy s nástupem nové techniky, neboť způsoby komunikce generačně navazujících systémů jeví vzájemnou značnou podobnost.

U P O Z O R N Ě N Í

V tomto manuálu se vyskytují následující upozornění.

! POZOR ! 2-19

Pokud uživatel instrukcí XBYTE nebo CBYTE přepíše lokace využívané systémem (viz kap.3.2), může dojít k chybnému chování systému.

! POZOR ! 2-23

Pokud uživatel instrukcí ASM přepíše lokace využívané systémem (viz kap.3.2), může dojít k chybnému chování systému.

O B S A H

ČÁST 1. VŠEOBECNÝ POPIS

Monitor UMON-51 vznikl jako pomůcka pro první seznámení s mikrokontrolérem 8051 a pro ladění a vyvíjení aplikačního programového vybavení všude tam, kde není k dispozici vývojový systém vybavený speciálním okolím pro práci s tímto typem mikrokontrolérů. Na druhé straně ale monitor UMON-51 poskytuje služby na úrovni dokonalejších prostředků pro ladění a vývoj programového vybavení řídících systémů s mikrokontroléry 8051, takže mnohdy je schopen tyto prostředky nahradit či úspěšně zastoupit.

Monitor UMON-51 představuje kompaktní programové vybavení napsané modulárním způsobem, což usnadňuje implementaci pro různé konfigurace hardware, takže UMON-51 je jen velmi málo závislý na konkrétní konfigurace hardware.

1.1. Stručný popis UMON-51

Příkazy UMON-51 jsou shodné po syntaktické stránce s příkazy emulátoru ICE-51 firmy Intel a i po stránce komunikací je zde velká podobnost.

Monitor UMON-51 poskytuje část systémových služeb i pro uživatelské programy. Tyto služby jsou přístupné prostřednic- tvím vstupního vektoru (kap. 4.2.).

Samotný monitor (program na desce zamýšlené aplikace v paměti EPROM) poskytuje většinu důležitých funkcí, které poskytuje emulátor ICE-51. Aby však bylo možno využít téměř celé škály příkazu uvedeného emulátoru, je možno k monitoru UMON-51 připojit hostitelský systém, na němž je implementováno hostitelské programové vybavení DEMON-51 pro komunikaci s UMON-51. Toto hostitelské programové vybavení podstatně rozšiřuje možnosti využití UMON-51, zejména o schopnosti symbolického ladění. Uvedené programové vybavení je k dispozici pod operačními systémy ISIS, PC-DOS a MS-DOS.

1.2. Požadavky na hardware

Již bylo řečeno, UMON-51 je možno implementovat na různé aplikace. To je dáno tím, že má minimální požadavky na konfiguraci hardware. Pro komunikaci s uživatelem využíva sériového interfejsu na čipu. Většina systémových proměnných využívá paměť dat na čipu, čímž jsou na minimum omezeny nároky na přístup do externí paměti.

Program monitor zaujímá oblast paměti 8kB. Platí jediné omezení pro paměť programu, neboť monitor pro přechodné ukládání proměnných potřebuje cca 512 byte paměti programu

v uživatelské oblasti (typu RAM, blíže viz kap 3.2.). Tato specifikace platí při implementaci na systém ELIS-51. Jiné implementace mohou vyuzívat jako pracovní oblast pro monitor též externí paměť dat.

Monitor UMON-51 vyžaduje jednoduchou úpravu hardware, která umožní zápis do paměti uživatelského programu jako do externí paměti dat (blíže viz kap 3.2.).

Monitor při implementaci na systém ELIS-51 nevyužívá externí paměť dat, takže uživatelský hardware jí nemusí být vůbec osazen.

ČÁST 2. POPIS ČINNOSTI

2.1. Popis činnosti UMON-51

Program UMON-51 poskytuje uživateli všechny důležité funkce, které umožňují ladění a vývoj aplikačního programového vybavení. Při jeho vývoji bylo přihlíženo k tomu, aby byl co možná nejméně závislý na konfiguraci hardware a aby jeho implementace na jiné systémy postavené na bázi mikrokontroléru 8051/8052 byla co nejjednodušší.

Množina příkazů monitoru UMON-51 je podmnožinou příkazů emulátoru ICE-51. To znamená, že komunikace s monitorem UMON-51 je stejná a zachovává si všechny rysy způsobu komunikace s emulátorem ICE-51.

Struktura programu UMON-51 je důsledně členěna do jednotlivých sekcí tak, aby bylo možné se v něm snadno orientovat a provádět případné změny. První blok obsahuje inicializační moduly, které zabezpečují nastavení komunikačních kanálů a uvedení uživatelských oblastí do takového stavu, který odpovídá statusu 8051 po resetu. Dále následuje modul dekodéru příkazů a chybových hlášení. Další část obsahuje vstupo-výstupní rutiny pro komunikaci s uživatelem. Za ní následuje modul obsahující rutiny pro zobrazení a změnu obsahu všech typů paměti obsluhovaných mikrokontrolérem 8051 (programová paměť, externí datová paměť, paměť dat na čipu a oblast paměti speciálních funkčních registrů). Za touto částí se nachází rutiny pro řízení běhu uživatelského programu (nastavování bodu přerušení, uvolňování přerušovací podmínky, krokování a spouštění uživatelského programu), pak následují rutiny pro zobrazení a změnu obsahu registrů a speciálních funkčních registrů, programového čítače, datového ukazatele aj., dále různé příkazy (příčina přerušení běhu programu, adresa poslední vykonané instrukce aj.), rutiny pro zavedení programu do paměti nebo uložení části programové paměti v hexadecimální formě (formát Intel) a konečně blok obsahující přímý asembler a disassembler v mnemokódu 8051.

Z výše uvedeného stručného přehledu je vidět, že monitor UMON-51 představuje poměrně mohutný nástroj pro ladění a vývoj uživatelských programů, a že si v mnohém nezadá s nepoměrně dražšími speciálními vývojovými zařízeními některých zahraničních firem.

2.2. Uživatelská konzola

Monitor UMON-51 komunikuje s uživatelem prostřednictvím sériového interfejsu na čipu. Proto je systém, na němž je UMON-51 implementovaný, připojitelný v zásadě k libovolnému sériovému terminálu nebo hostitelskému systému vybavenému sériovým rozhraním. Toto zařízení pak funguje jako systémová

konzola. Jednotlivé příkazy jsou monitoru zadávány jako posloupnosti alfanumerických znaků, zakončených návratem vozu. Tento symbol budeme v dalším textu označovat jako <cr> (cr=carriage return).

Stisk jakékoliv alfanumerické klávesy způsobí odvysílání příslušného znaku po sériovém interfejsu na displej. Monitor tyto znaky přijme, uloží do vstupní fronty příkazu a odvysílá zpět na konzolu přijatý znak (echo), takže uživatel má okamžitou kontrolu správnosti komunikace. Některé klávesy mají zvláštní význam:

<cr> je klávesa RETURN, ukončuje zadávaní příkazu.

<ESC> je klávesa ESCAPE, způsobí přerušení činnosti monitoru a návrat do režimu očekávání příkazu.

<shift> je klávesa SHIFT, která má za následek přechod k velkým znakům. Monitor ovšem nerozlišuje, zda jsou mu příkazy zadávány jako znaky velké abecedy nebo malé abecedy a automaticky je konvertuje na velké znaky, takže jako echo jsou vraceny velká písmena, i když operátor komunikuje s vypnutým shiftem.

<tab> je klávesa tabelace. Monitor nahrazuje tabelátor příslušným počtem mezer tak, aby platil ofset 8.

<rub> je klávesa RUBOUT (mazání). Způsobí výmaz posledního zadaného znaku a návrat kurzoru o jednu pozici vlevo.

<CTL-S> znamená stisk kláves <CONTROL> a <S> najednou a způsobí zastavení výpisu na konzolu až do té doby, než bude stisknuto <CTL-Q>, čímž je výstup znaků na konzolu opět povolen.

<CTL-Q> znamená stisk kláaves <CONTROL> a <Q> najednou a způsobí opětovné povolení výpisu na konzolu po předchozím zastavení výpisu na konzolu pomocí <CTL-S>. Samotný stisk <CTL-Q> bez předchozího stisku <CTL-S> nemá žádný význam.

Co se týče zobrazovače, monitor nemá žádné zvláštní nároky. Nepoužívá žádné speciální řídící znaky. Využívá zobrazení formátu 64 znaků na řádek, což vyhoví většině terminálů nebo mikropočítačů u nás běžně používaných.

Délka vstupního řádku je omezena na 128 znaků. Pokud uživatel chce zadat delší příkaz, musí jej rozdělit do více částí. Monitor zadání delší posloupnosti znaků nepřipouští a pokud délka příkazu dosáhne maximální hranice, monitor další znaky ignoruje dokud nepřijde znak <cr>.

2.3. Režimy činností

Monitor UMON-51 má pět následujících režimů činnosti:
- režim očekávání příkazu (po zapnutí, signálu reset nebo vykonání příkazu)
- režim zobrazování (dlouhé výpisy)
- režim asembleru
- režim běhu uživatelského programu
- režim krokování uživatelského programu

Monitor přechází automaticky po připojení napájení nebo signálu reset do režimu očekávání příkazu. Zobrazí symbol '-' (prompt) a očekává příkazy uživatele. Veškeré příkazy jsou zadávány v tomto režimu.

Režim zobrazování je takový režim, kdy monitor vypisuje dlouhé zprávy (zobrazování dlouhých úseků paměti, disassembler).

Režim asembleru umožňuje uživateli zadávat instrukce přímo v mnemonice 8051. Tento režim se zavede příkazem ASM, zakončeným <cr>. Monitor očekává nyní vstup instrukce v mnemokódu 8051.

Režim běhu programu umožnuje spouštění napsaného programu (režimem asembleru nebo zavedeného příkazem LOAD) a jeho ladění.

Režim krokování umožňuje krokovat program po jednotlivých instrukcích a nabízí uživateli efektivní ladění programu. Monitor zobrazuje vykonávanou instrukci v mnemonice 8051 a obsah registrů po každém kroku.

2.4. Klíčová slova a čísla v příkazech

Příkaz zadávaný monitoru UMON-51 začíná klíčovým slovem příkazu, které indikuje operaci, jež má být prováděna. Toto klíčové slovo může být následováno jedním nebo více operandy, které vymezují nebo definují parametry operace. Následující příklad způsobí nastavení stacku na adresu 20H:

sekvence	komentář
SP=20<cr>	Nastaví ukazatel stacku uživatele na adresu 20H. SP je klíčové slovo, = je přiřazovací operátor a 20 je parametr nastavující stack pointer.

Termín klíčové slovo označuje tedy množinu slov nebo jejich zkratek, které jsou monitorem interpretovány tak, že na ně definovaným způsobem reaguje. Zkratky klíčových slov můžete výhodně použít tam, kde nechcete psát celou dlouhou posloupnost znaků příkazu a tak můžete urychlit průběh komunikace s monitorem UMON-51.

Ve většině případů se nejmenší zkratky klíčových slov neliší od zkratek použitých u emulátoru ICE-51. Tabulka 2-1 shrnuje klíčová slova a jejich zkratky.

Tabulka 2-1 Klíčová slova a jejich zkratky

Klíčové slovo	Minimální zkratka	Klíčové slovo	Minimální zkratka
ACC	ACC	PPC	PP
ASM	AS	PSW	PS
B	B	R0	R0
BR	BR	R1	R1
BR0	BR0	R2	R2
BR1	BR1	R3	R3
CAUSE	CA	R4	R4
CBYTE	CB	R5	R5
CHIP	CH	R6	R6
DASM	DAS	R7	R7
DBYTE	DB	RBIT	RBI
DPTR	DP	RBS	RBS
FOREVER	FO	RBYTE	RB
FROM	F	REGISTERS	R
GO	G	RESET	RES
GR	GR	SAVE	SA
LOAD	LO	SP	SP
OFF	OF	STACK	STA
ON	ON	STEP	S
OR	OR	TILL	T
ORG	ORG	TM0	TM0
PBYTE	PB	TM1	TM1
PC	PC	TO	TO
		XBYTE	XB

Parametry mohou být další klíčová slova nebo číselné výrazy. Číselné výrazy se dělí na:

číslo	hexadecimální číslice v rozsahu 0..F
byte	hexadecimální číslo složené ze dvou čísel (ze tří čísel, požaduje-li se vedoucí nula)
word, adresa	hexadecimální číslo složené ze čtyř čísel (z pěti čísel, požaduje-li se vedoucí nula)
part	adresa nebo výraz 'adresa TO adresa'. Part obsahuje jednu nebo dvě adresy. Obsahuje-li dvě adresy, musí být druhá adresa od prvé oddělena klíčovým slovem TO a musí platit nerovnost druhá adresa je větší než první.
string	řetězec znaků ASCII uzavřených mezi apostrofy, např.: 'TOTO JE STRING'

Monitor chápe všechna čísla jako hexadecimální. V tomto textu je hexadecimální číslo indentifikováno příznakem H za číslem, např. 0123H. Monitor příznak H nevyžaduje, ale je možno jej používat.

POZNÁMKA

Monitor potřebuje, aby všechna čísla začínala dekadickým číslem. Proto hexadecimální čísla, která této podmínce nevyhoví, musí začínat vedoucí nulou. Například číslo 7F je akceptováno, ne však číslo F7; to musí být zadáno jako 0F7.

2.5. Poznámky k syntaxi příkazů

Syntaxe všech dále popsaných příkazů je zapsána jednotným způsobem. Zápis ukazuje, jaká klíčová slova musí uživatel zadat, jaká může vynechat nebo volitelně vnořit a naznačuje použití parametrů v příkaze. Pro zápis syntaxe příkazů v tomto manuálu platí následující pravidla:

- kličová slova jsou zapsána velkými písmeny
- číselné parametry jsou zapsány malými písmeny
- pokud příkaz vyžaduje parametr, není uzavřen do hranatých závorek, např.:

  SAVE part

  V tomto případě jsou jak klíčové slovo SAVE, tak parametr part povinné.

- pokud je zadání parametru volitelné, jsou parametry uzavřeny v hranatých závorkách, např.:

  ASM[ORG adresa]
  V tomto případě je klíčové slovo ASM povinné, výraz ORG adresa volitelný.

- pokud některý parametr je volitelný a může být vícenásobně zadán, je uzavřen do hranatých závorek a následován pokračovacím příznakem (...), např.:

  XBYTE 100=byte,[byte]...
  Zde se po klíčovém slově XBYTE a přiřazovacím operátoru = vyžaduje zadání minimálně jednoho parametru, který může být následován libovolným počtem dalších parametrů, oddělených čárkami, až do maximálního počtu 128 znaků.

- pokud příkaz umožňuje výběr jednoho nebo žádného ze seznamu parametrů, je seznam těchto parametrů uzavřen v hranatých závorkách a seřazen nad sebou, např.:

```
GO[ FROM adresa][ARM adresa][ FOREVER]
                            [ TILL BR0]
                            [ TILL BR1]
                            [ TILL BR]
```

  V tomto případě je povinné pouze slovo GO, vše ostatní je volitelné uživatelem.

- pokud příkaz vyžaduje právě jeden ze seznamu parametrů, je tento seznam uzavřen v lomených závorkách a seřazen nad sebou, např.:

```
RESET {CHIP}
      {BR}
      {BR0}
      {BR1}
```

Zde klíčové slovo RESET musí být následováno právě jedním ze seznamu uvedených klíčových slov.

2.6. Příkazy pro zobrazení a změnu obsahu

Přiřazovací operátor (znak =) se používá pro změnu obsahu. Pokud je příkaz vložen bez operátoru přiřazení, znamená to, že objekt referovaný klíčovým slovem (obsah paměti, registry, adresy přerušení běhu programu atd.) mají být ponechány beze změny a pouze zobrazeny. Pokud je ale klíčové slovo nebo výraz následován operátorem přiřazení, znamená to, že objekt referovaný příkazem bude změněn tak, jak to odpovídá parametru, který musí za přiřazovacím operátorem následovat; např.:

```
sekvence        komentář
RBIT 7<cr>      zobrazí hodnotu bitu na adrese 7
RBIT 7=0<cr>    přiřadí bitu na adrese 7 hodnotu 0
```

2.7. Přehled příkazů a činností

Příkazy monitoru UMON-51 a jim odpovídající činnosti se dělí do čtyř skupin:

- připojení zdroje, signál reset a vstup příkazu
- příkazy pro vykonávání a ladění uživatelského programu
  - příkazy nastavení programové zarážky (breakpoint)
  - příkazy řízení běhu GO a GR
  - příkaz STEP
  - příkaz zobrazení příčiny zastavení běhu programu CAUSE
  - příkazy PC a PPC
- příkazy pro přístup do paměti
  - příkazy CBYTE, DBYTE, RBYTE, XBYTE, PBYTE a RBIT
  - příkazy R0..R7, ACC, B, PSW, SP, DPTR, TM0, TM1
  - příkazy ASM a DASM
  - příkaz STACK
- příkazy pro komunikaci s hostitelským systémem
  - příkazy LOAD a SAVE

2.7.1. Připojení napájení

Po připojení napájení (zdroj +5V,+/-12V) se monitor na připojeném komunikačním zařízení (sériový terminál, počítač se sériovým interfejsem) přihlásí hlavičkou ve tvaru:

```
UMON-51 MONITOR VERS. x.y
CPU=zz
-
```

a monitor přejde do stavu očekávání příkazu, což signalizuje prompt (znak '-' zobrazený na novém řádku pod hlavičkou). Kód x.y je číslo vaší verze programu UMON-51, kód zz je typ použitého čipu (může být buď 51, 52 nebo 44). Tato vlastnost je využívána pro činnost programu DEMON-51.

2.7.2. Reset

Signál reset má stejný účinek jako připojení zdrojů.

POZNÁMKA

Je třeba důsledně odlišovat signál reset, který je vždy zapsán malými písmeny, od příkazu RESET, který je vždy zapsán velkými písmeny.

2.7.3. Vstup příkazu

Po připojení zdrojů nebo signálu reset přechází monitor do režimu očekávání příkazu. Délka jednoho příkazu může být až 128 znaků. Pokud zadá uživatel neznámý příkaz nebo použije nesprávné parametry, zobrazí se hlášení 'ERR=XX' a příslušný text. Chybová hlášení jsou popsána v příloze.

Pokud uživatel zadá správný příkaz, příkaz se vykoná a monitor po jeho provedení zobrazí prompt, čímž oznamuje připravenost pro příjem dalšího příkazu.

2.8. Příkazy pro vykonávání uživatelského programu

Následující příkazy mohou být použity pro řízení vykonávání uživatelského programu. Příkaz BREAK nastavuje adresu přerušení běhu uživatelského programu (breakpoint, programová zarážka). Příkazy GO a STEP spouštějí uživatelský program. Příkaz GR řídí podmínky zastavení běhu uživatelského programu. Příkaz CAUSE umožní uživateli zjistit, proč došlo k přerušení běhu uživatelského programu. Příkaz PC umožní nastavit novou adresu vykonávání uživatelského programu. Příkaz PPC slouží ke zjištění adresy poslední vykonané instrukce. Příkaz RESET umožní zrušit body zastavení běhu uživatelského programu nebo inicializovat stav uživatelského programu do stavu, který se nastaví po signálu reset.

2.8.1. Příkazy BREAK

Tato skupina příkazů umožňuje uživateli nastavit programové zarážky, tj. body přerušení běhu uživatelského programu. Pro řízení vykonávání uživatelského programu je možno použít dva, jeden nebo žádný bod přerušení uživatelského programu. Programové zarážky jsou uvolněny příkazy GO nebo GR. Pokud se při vykonávání programu uživatele narazí na programovou zarážku, dojde k přerušení běhu uživatelského programu, vypíše se zpráva o přerušení běhu uživatelského programu a monitor přejde do režimu očekávání příkazu.

Příkaz BR slouží pro zapnutí nebo vypnutí možnosti přerušení běhu uživatelského programu klávesou <ESC> a pro zobrazení programových zarážek. Pro jejich nastavení slouží příkazy BR0 a BR1, ty se však dají používat i pro jejich zobrazení.

Syntaxe

```
BR [=ON]<cr>
   [=OFF]<cr>
BR0[=adresa]<cr>
BR1[=adresa]<cr>
```

2.8.1.1. Zobrazení programových zarážek

Pro zobrazení obou programových zarážek použijte příkaz:

sekvence	komentář
BR<cr>	Monitor zobrazí např.: BR=ON BR0=RESE BR1=2500 což znamená že programová zarážka 0 byla zrušena nebo nebyla nastavena a programová zarážka 1 ukazuje na adresu 2500, přerušení běhu programu uživatelem je zapnuto
BR1<cr>	monitor zobrazí např.: BR1=2500 což znamená, že programová zarážka 1 je nastavena na adresu 2500H

2.8.1.2. Nastavení programových zarážek

Pokud chcete nastavit nebo změnit nastavení programové zarážky, zadejte např.:

sekvence	komentář
BR0=2400<cr>	nastavení programové zarážky 0 na adresu 2400H
BR=OFF	vypnutí možnosti přerušení běhu uživatelského programu klávesou <ESC>

POZNÁMKA

Mějte na paměti, že pouhé nastavení programových zarážek ještě neznamená, že se uživatelský program na odpovídajících bodech přeruší. Toho se dosáhne povolením programových zarážek příkazy GO či GR.

! POZOR !

Pokud příkazem BR=ON zapnete možnost přerušení běhu programu klávesou <ESC>, systém při krokování programu nebo jeho spuštění příkzem GO přezkoumá obsah adresy obsluhy přerušení od sériového kanálu. Pokud obsahuje instrukci LJMP směrovanou na obsluhu tohoto přerušení ve Vašem programu, ponechá tuto lokaci beze změny. V opačném případě sem zapíše posloupnost instrukcí

```
CLR  RI
RETI
```

Pokud Váš program používá přerušení od sériové linky, musí z tohoto tyto lokace obsahovat instrukci LJMP. Při použití kompilátoru PL/M-51 nebo C-51 je tato podmínka splněna automaticky. Dále nesmí Váš program trvale zamaskovat přerušení od sériové linky nebo předefinovat její parametry.

! POZOR !

Pokud se vyskytne programová zarážka (breakpoint), systém zapíše 4 byte do oblasti stacku. Stack se sice obnoví, ale uživatel musí při umisťování stacku pamatovat na tuto skutečnost a umístit stack tak, aby systém měl tyto 4 lokace k dispozici. V opačném případě dojde k chybnému chování systému.

! POZOR !

Programová zarážka se vykoná jako volání (dlouhé) podprogramu na adresu 0006H (LCALL 0006H). Z tohoto důvodu nesmí uživatel použít programovou zarážku na těch lokacích programu, které jsou vzdáleny o méně než 3 byte od adres, které jsou cílové pro instrukce větvení. Ze stejného důvodu musí uživatel při použití obou programových zarážek specifikovat jejich adresy tak, aby minimální rozdíl umístění programových zarážek byl 3 lokace paměti programu.

2.8.2. Příkaz RESET

Příkazem RESET můžete rušit nastavení programových zarážek nebo simulovat signál reset pro Váš aplikační program.

Syntaxe

```
RESET{ BR}<cr>
     { BR0}<cr>
     { BR1}<cr>
     { CHIP}<cr>
```

2.8.2.1. Zrušení programových zarážek

Pro zrušení programových zarážek použijte příkaz RESET např. takto:

sekvence	komentář
RESET BR<cr>	zruší nastavení obou programových zarážek
RESET BR0<cr>	zruší nastavení programové zarážky 0

2.8.2.2. Simulace signálu reset

Pokud potřebujete v aplikačním programu provést nastavení počátečních hodnot všech registrů jako po signálu reset (inicializační hodnoty), použijte tento tvar příkazu RESET:

sekvence	komentář
RES CHIP<cr>	nastaví stav procesoru do stejného stavu, jaký nastaví signál reset a vypíše hlavičku monitoru s verzí

2.8.3. Příkaz GO

Příkaz GO spouští běh uživatelského programu. Zahrnuje nastavení adresy spuštění běhu programu nebo spuštění programu od nastavené (běžné) pozice programového čítače, povolení či zákaz přerušení běhu programu od jedné či obou programových zarážek a konečně podmíněné povolení programových zarážek.

Syntaxe

```
GO[ FROM adresa][ ARM adresa][ FOREVER]<cr>
                             [ TILL BR]<cr>
                             [ TILL BR0]<cr>
                             [ TILL BR1]<cr>
```

2.8.3.1. Spuštění běhu uživatelského programu

sekvence	komentář
GO<cr>	Spustí program uživatele od nastavené pozice programového čítače beze změny povolení programových zarážek. Na konzolu vypíše hlášení 'EXECUTION BEGUN'. Běh programu skončí, když se vyskytne programová zarážka (povolená). Po jejím výskytu se objeví hlášení 'EXECUTION HALTED, PC=nnnn' a monitor přejde do režimu očekávání příkazu

2.8.3.2. Spuštění běhu od zadané adresy

Pro spuštění běhu od zvolené adresy zadejte příkaz:

```
GO FROM adresa
```

Například:

sekvence	komentář
GO FROM 80<cr>	spustí vykonávání programu od adresy 80H beze změn nastavení programových zarážek

2.8.3.3. Spuštění běhu s povolením zarážky

Následující tvar příkazu použijte pro spuštění běhu uživatelského programu, pokud chcete povolit přerušení běhu uživatelského programu.

```
GO[ FROM adresa][ ARM adresa]{ TILL BR}
                             { TILL BR0}
                             { TILL BR1}
```

Například:

sekvence	komentář
GO TILL BR<cr>	spustí program uživatele od nastavené hodnoty programového čítače a povolí přerušení od obou programových zarážek
G F 100 ARM 177 T BR0<cr>	spustí uživatelský programu od adresy 100H, zakáže programovou zarážku 1 a povolí programovou zarážku 0, ale až poté, co program projde lokací 177H

2.8.3.4. Spuštění běhu se zákazem zarážky

Následující tvar příkazu použijte pro spuštění běhu uživatelského programu, pokud chcete zakázat předešlým příkazem povolené přerušení běhu uživatelského programu.

```
GO[ FROM adresa] FOREVER
```

Například:

sekvence	komentář
GO FOREVER<cr>	spustí běh od nastavené hodnoty programového čítače a zakáže přerušení běhu programu uživatele
G F 80 FOR<cr>	spustí běh od adresy 80H, zakáže přerušení běhu programu uživatele

POZNÁMKA

Pokud příkazem GO spustíte program a povolíte přerušení běhu programu, aniž byste předem příkazem BR zadali adresu bodu přerušení, nemůže se samozřejmě žádné přerušení provést.

POZNÁMKA

Pokud uživatelský program využívá služeb monitoru, např. některou z rutin, přístupných vstupním vektorem, pak není možné do těchto rutin umísťovat programovou zarážku. Pokus o umístění programové zarážky v oblasti pevné paměti vede k chybě (viz přílohu).

2.8.4. Příkaz GR

Příkaz GR umožňuje zobrazit nebo nastavit podmínky přerušení běhu uživatelského programu. Je analogický příkazu GO, avšak pouze povoluje nebo zakazuje přerušení běhu uživatelského programu, uživatelský program nespouští.

Syntaxe

```
GR[=FOREVER]<cr>
  [=[ARM adresa ]TILL BR]<cr>
  [=[ARM adresa ]TILL BR0]<cr>
  [=[ARM adresa ]TILL BR1]<cr>
```

2.8.4.1. Zobrazení přerušovacích podmínek

Pro zobrazení nastavených přerušovacích podmínek použijte následujícího příkazu:

sekvence	komentář
GR<cr>	Zobrazí nastavení podmínek přerušení např: GR=ARM 13 TILL BR0 BR0=80 BR1=RESET čímž oznamuje, že po průchodu adresou 13 bude povoleno přerušení běhu uživatelského programu na adrese 80H, na níž je nastaven bod 0, bod 1 není nastaven

2.8.4.2. Povolení a zákaz programových zarážek

Následující tvar příkazu použijte, pokud chcete povolit nebo zakázat přerušení běhu uživatelského programu od jedné či obou programových zarážek.

```
GR{=[ARM adresa ]TILL BR}
  {=[ARM adresa ]TILL BR0}
  {=[ARM adresa ]TILL BR1}
  {=FOREVER}
```

Například:

sekvence	komentář
GR=TILL BR<cr>	povolí přerušení od obou programových zarážek
GR=ARM 7 T BR0<cr>	po průchodu adresou 7 povolí přerušení od zarážky 0, zakáže přerušení od zarážky 1
GR=FOREVER<cr>	zakáže obě zarážky

2.8.5. Příkaz STEP

Příkazem STEP můžete provést vykonání jedné nebo více instrukcí, přičemž po vykonání každé instrukce dojde k přerušení vykonávání programu, zobrazení stavu všech registrů a některých dalších speciálních funkčních registrů (ACC, B, SP, PSW, DPTR), a programového čítače. Kromě toho můžete zvolit ještě zobrazení jednoho byte (nebo bitu) z Vámi udané oblasti paměti.

Syntaxe

```
STEP[ FROM adresa][,memory-typ adresa]<cr>
```

Kde memory-typ značí jedno z následujících klíčových slov:

DBYTE	paměť dat na čipu
RBYTE	oblast speciálních funkčních registrů
CBYTE	paměť programu
XBYTE	externí paměť dat s kontrolou zápisu
PBYTE	externí paměť dat bez kontroly zápisu (pro bezchybný zápis do V/V obvodů - memory mapped I/O)
RBIT	bitově adresovatelná oblast paměti

2.8.5.1. Provedení jedné instrukce

Pro provedení jedné instrukce zadejte příkaz:

```
STEP[ FROM adresa]
```

Například:

sekvence	komentář
STEP<cr>	vykoná jednu instrukci na nastavené (běžné)

	pozici programového čítače, vypíše níže popsaným způsobem stav procesoru (obsah registrů) a přejde do režimu krokování. V režimu krokování znamená stisk libovolné klávesy vyjma <ESC> vykonání dalšího kroku; klávesa <ESC> má za následek přechod do režimu očekávání příkazu
STEP FROM 100<cr>	vykoná instrukci na adrese 100H, zobrazí stav procesoru a přejde do režimu krokování. <ESC> v režimu krokování znamená povel k zastavení krokování a přechodu do režimu očekávání příkazu

Jak bylo řečeno, po každém kroku se vypíše stav procesoru. Tento výpis má následující tvar:

```
 PC  SP ACC B R0 R1 R2 R3 R4 R5 R6 R7 DPTR   PSW
xxxx xx xx xx xx xx xx xx xx xx xx xx xxxx iiiiiiii
```

(výrazy xx nebo xxxx odpovídají zobrazení v hexadecimální formě a výraz iiiiiiii zobrazení v binární formě)

Záhlaví (první řádek výpisu) označuje registry, jejichž hodnoty (obsahy) jsou vypisovány a jejich názvy odpovídají mnemonice 8051; druhý řádek zobrazuje obsahy těchto registrů formou hexadecimálních výrazů (byte nebo adresa) s vyjímkou stavového slova programu (PSW), které je zobrazeno binárně. Binární zobrazení v tomto případě bylo voleno z důvodu lepší přehlednosti, neboť pro ladění programu jsou důležité jednotlivé bity PSW. Zobrazované obsahy registrů ve všech případech odpovídají obsahům po provedení instrukce.

2.8.5.2. Krokování se zobrazením paměti

V případě potřeby zobrazení jiného byte nebo bitu paměti, než Vám nabízí výše posaný formát výpisu při krokování, můžete použít následující tvar příkazu STEP:

```
STEP[ FROM adresa],{CBYTE adresa}
                   {DBYTE adresa}
                   {RBYTE adresa}
                   {XBYTE adresa}
                   {PBYTE adresa}
                   {RBIT adresa}
```

Výpis stavu procesoru má v tomto případě tvar:

```
 PC  SP ACC B R0 R1 R2 R3 R4 R5 R6 R7 DPTR   PSW
xxxx xx xx xx xx xx xx xx xx xx xx xx xxxx iiiiiiii (xx)
```

kde hodnota v závorkách je obsah zadané adresy specifikované oblasti paměti. Jinak forma výpisu souhlasí s výše popsaným formátem výpisu stavu procesoru.

Příkaz STEP se zobrazením paměti lze použit např. takto:

sekvence	komentář
STEP,DBYTE 55<cr>	vykoná jednu instrukci na nastavené (běžné)

	pozici programového čítače, vypíše výše popsaným způsobem stav procesoru (obsah registrů) a obsah paměti dat na čipu na adrese 55H a přejde do režimu krokování. V režimu krokování znamená stisk libovolné klávesy vyjma <ESC> vykonání dalšího kroku; klávesa <ESC> má za následek přechod do režimu očekávání příkazu
S F 8,XB 90<cr>	vykoná instrukci na adrese 8H, zobrazí stav procesoru, obsah paměti dat na adrese 90H a přejde do režimu krokování
STEP,RBIT 50<cr>	vykoná instrukci na adrese určené okamžitým stavem programového čítače, zobrazí stav procesoru, hodnotu bitu na adrese 50H v bitově adresovatelném prostoru paměti a přejde do režimu krokování

POZNÁMKA

Pokud program využívá služeb monitoru, např. některou z rutin, přístupných vstupním vektorem, pak tyto rutiny není možné krokovat. Pokus o krokování v oblasti pevné paměti vede k chybě (viz přílohu).

2.8.6. Příkaz CAUSE

Příkaz CAUSE slouží k zobrazení příčiny zastavení běhu uživatelského programu.

Syntaxe

CAUSE<cr>

Příkaz CAUSE může produkovat jedno z následujících tří hlášení:

Hlášení	Význam
Program break	za běhu uživatelského programu se narazilo na jednu z programových zarážek.
Single step	poslední instrukce byla vykonána v režimu krokování.
No break	uživatelský program nebyl vůbec spuštěn, popř. přišel signál reset nebo byl zadán příkaz RESET CHIP.
User exit	Činnost uživatelského programu byla ukončena předáním řízení na monitorovskou službu EXIT
User abort	Běh uživatelského programu byl přerušen stiskem klávesy <ESC>

2.8.7. Příkaz PC

Tento příkaz slouží k zobrazení nebo nastavení programového čítače uživatelského programu.

Syntaxe

PC[=adresa]<cr>

Následující příklad ilustruje použití příkazu PC:

sekvence	komentář
PC<cr>	zobrazí nastavenou (běžnou) pozici programového čítače uživatelského programu.
PC=0FEEEH<cr>	nastaví programový čítač programu uživatele na adresu FEEEH, takže pokud příští příkaz STEP nebo GO nebudou obsahovat část FROM adresa, vykoná se instrukce na adrese FEEEH

2.8.8. Příkaz PPC

Příkaz PPC slouží k zobrazení adresy poslední instrukce, která se vykonala před přerušením běhu programu. To znamená, že bude zobrazena adresa té instrukce, která způsobila nastavení programového čítače do stávajícího stavu. Chcete-li tuto adresu zjistit, zadejte:

sekvence	komentář
PPC<cr>	zobrazí adresu poslední vykonané instrukce za běhu uživatelského programu či v režimu krokování

2.9. Příkazy pro přístup do paměti

Pro přístup do paměti obsluhované mikrokontrolérem 8051 slouží tři skupiny příkazů:

1. přístup zadáním typu paměti (memory-typ)
2. přístup zadáním klíčové-slovo
3. ostatní přístupy

2.9.1. Specifikace udáním typu paměti

Tímto způsobem můžete adresovat veškerou paměť, se kterou mikrokontrolér 8051 pracuje. Pro typ paměti, který je zde označen výrazem memory-typ, platí údaje podle tabulky 2-2.

Tabulka 2-2 Paměťové oblasti mikrokontroléru 8051

memory-typ	rozsah	druh paměti
DBYTE	0000-007F	paměť dat na čipu
RBYTE	0080-00FF	speciální funkční registry
CBYTE	0000-FFFF	paměť programu
XBYTE	0000-FFFF	externí paměť dat
PBYTE	0000-FFFF	vstupně-výstupní obvody
RBIT	0000-00FF	bitově adresovatelná paměť

Syntaxe

```
memory-typ adresa[=byte[,byte[,<cr>byte,'string',..]<cr>
memory-typ part[=byte]<cr>
memory-typ part=memory-typ part<cr>
```

Z popsaných syntaxí je patrné, že je možno prohlížet jeden byte nebo úsek vybraného typu paměti, zapisovat do vybraného typu paměti jednotlivé byte nebo textové řetězce s možností pokračování na pokračovacím řádku, plnit oblast paměti konstantou nebo přenášet jednotlivé úseky paměti mezi sebou. Pro lepší srozumitelnost a názornost jsou dále uvedeny některé typické příklady:

sekvence	komentář
CBYTE 0<cr>	zobrazí kód instrukce (byte) na adrese 0H
CBYTE 80 TO 9F<cr>	zobrazí kódy instrukcí (byte) na adresách 80H až 9FH.
XB 0 TO 6F=55<cr>	vyplní oblast externí paměti na adresách 00H až 6FH konstantou 55H
DB 8=3,4,5<cr>	do paměti dat na čipu zapíše na adresy 8H, 9H, 0AH posloupnost hodnot 3H,4H,5H
XB 5='INIT',0D<cr>	do externí paměti dat zapíše od adresy 5H string 'INIT' a ukončí jej znakem 0DH
CB 2000=0A,'0123456789',0,4,'AHOJ',<cr>	zapíše do paměti programu od adresy 2000H konstantu 0AH, dále zapíše string '0123456789', konstanty 0H a 4H, string 'AHOJ' a pokračuje na pokračovacím řádku (monitor vypíše na nový řádek 'CBYTE 2011=' a očekává další vstupy, tj. byte nebo string, pro plnění paměti)
XB 0 TO 80=CB 100<cr>	překopíruje oblast paměti programu na adresách 100H až 180H do externí paměti dat na adresy 0H až 80H
DB 8=RBIT 20 TO 30<cr>	překopíruje bity z adres 20H až 30H do paměti dat na čipu od adresy 8H a konvertuje přitom každý bit do délky byte (tj. jednotlivé byte budou nabývat hodnot 0H nebo 1H)

Formát výpisu při zobrazení je následující:

```
XBYT 1000=41,42,43,00,01,02,03,20,20,4F,00,00,00,00,00,00
          A  B  C  .  .  .  .        O  .  .  .  .  .  .
```

Číselné hodnoty na první řádku jsou hexadecimální čísla, pod nimi na druhém řádku se zobrazují ASCII reprezentace těchto hodnot, pokud to jsou zobrazitelné znaky, v opačném případě se zobrazí tečka.

Pro bitově adresovatelnou paměť postrádají ASCII reprezentace smysl, proto se nevypisují. Formát zobrazení je např:

```
RBIT 0009=00,01,01,00,00,00,01,00,01,01,01,00,00,00,00,01
```

POZNÁMKA

Pokud uživatel jako parametr instrukce RBIT použije parametru typu byte nebo string, bude pro plněni bitové oblasti paměti využito nejnižšího bitu z každého byte, resp. každého znaku stringu.

2.9.1.1. Paměť dat na čipu

Paměť dat na čipu je možno zobrazovat a měnit příkazem DBYTE. V této oblasti paměti se nachází zásobník (stack), dále 4 banky registrů mnemonicky označených R0..R7 a bitově adresovatelná oblast paměti na adresách 0..7FH. Délka této paměti je 128 byte, takže zaujímá prostor adres 0..7FH.

Stack musí být pro mikrokontroléry 8051 umístěn v této oblasti a nikdy nesmí přesáhnout hodnotu 7FH. Tabulka 2-4 znázorňuje oblast paměti dat na čipu.

Pro mikrokontroléry 8052 je možné používat pro stack nebo nepřímé adresování paměti dat na čipu i oblast adres 80..FFH. V této oblasti se může nacházet i stack.

2.9.1.2. Speciální funkční registry

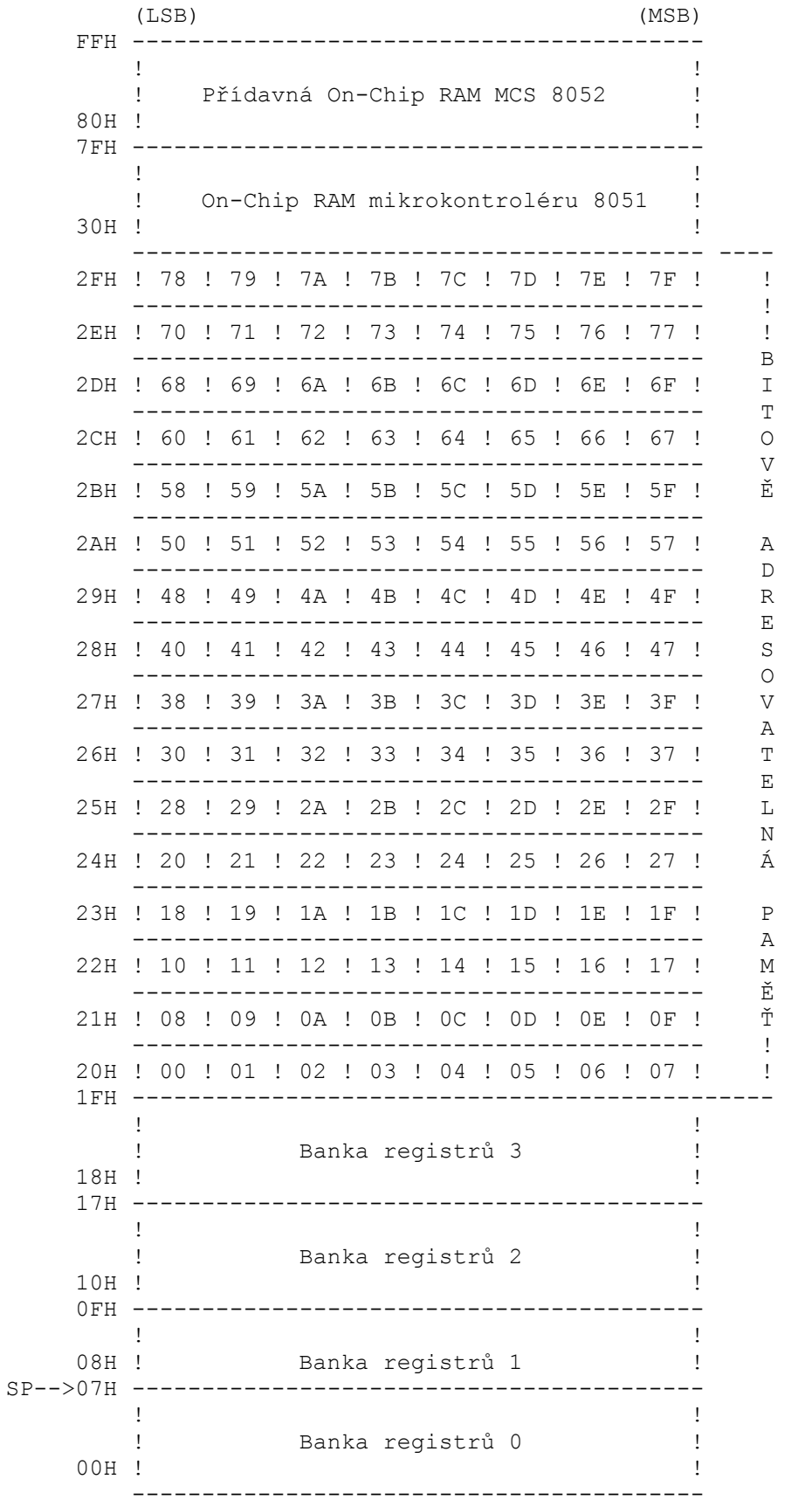
Adresní prostor 80H..FFH paměti na čipu není u 8051 spojitý. Jsou obsazeny pouze některé adresy. Tyto adresy reprezentují registry, které mají zvláštní funkce, proto se tato oblast paměti nazývá oblast speciálních funkčních registrů (SFR). Tabulka 2-3 obsahuje seznam SFR. Pro přístup do této oblasti paměti slouží příkaz RBYTE.

Tabulka 2-3 Speciální funkční registry

Adresa	Význam
*80H	Port 0 (P0)
81H	Stack pointer (SP)
82H	Data pointer, nižší byte (DPL)
83H	Data pointer, vyšší byte (DPH)
87H	Power control (PCON)
*88H	Timer - řízení (TCON)
89H	Timer - mód činnosti (TMOD)
8AH	Timer 0,nižší byte (TL0)
8BH	Timer 1,nižší byte (TL1)
8CH	Timer 0,vyšší byte (TH0)
8DH	Timer 1,vyšší byte (TH1)
*90H	Port 1 (P1)
*98H	Sériový interface - řízení (SCON)
99H	Sériový interface - data (SBUF)
*A0H	Port 2 (P2)
*A8H	Interrupt enable (IE)
*B0H	Port 3 (P3)
*B8H	Interrupt priority control (IP)
#*C8H	Timer 2 - řízení (T2CON)
# CAH	Timer 2 záchytný register - nižší byte (RCAP2L)
# CBH	Timer 2 záchytný register - vyšší byte (RCAP2H)
# CCH	Timer 2,nižší byte (TL2)
# CDH	Timer 2,vyšší byte (TH2)
*D0H	Stavové slovo programu (PSW)
*E0H	Akumulátor (ACC)
*F0H	Multiplikační registr (B)

* - bitově adresovatelné
# - pouze pro 8052

Tabulka 2-4 Rozložení paměti dat na čipu.



2.9.1.3. Externí paměť dat a paměť programu

Pro přístup do externí paměti dat je určen příkaz XBYTE. Příkaz CBYTE slouží pro přístup do paměti programu, kromě tohoto příkazu je však paměť programu přístupná i pomocí příkazů ASM (kap. 2.9.3.) a DASM (kap. 2.9.4.). Jejich použití není omezeno adresním rozsahem, takže je možno oslovovat celou paměť v rozsahu 0000H..FFFFH.

```
---------
! POZOR !
---------
```

Pokud uživatel instrukcí XBYTE nebo CBYTE přepíše lokace využívané systémem (viz kap.3.2), může dojít k chybnému chování systému.

2.9.1.4. Bitově adresovatelná paměť

Část paměti dat na čipu a části paměti speciálních funkčních registrů (SFR) jsou bitově adresovatelné.

Bitově adresovatelná oblast paměti dat na čipu je znázorněna v tabulce 2-4. Tyto bity jsou všeobecně použitelné a nemají žádnou zvláštní funkci.

Bitově adresovatelné registry v oblasti SFR jsou znázorněny v tab. 2-3. Většina bitů těchto registrů má nebo nebo může mít speciální funkce, jak je popsáno v tabulce 2-5.

Pro přístup do oblasti bitově adresovatelné paměti slouží příkaz RBIT.

Tabulka 2-5 Adresy bitů v oblasti SFR

Adresa	Význam
80H	Port 0, bit 0 (P0.0)
81H	Port 0, bit 1 (P0.0)
82H	Port 0, bit 2 (P0.0)
83H	Port 0, bit 3 (P0.0)
84H	Port 0, bit 4 (P0.0)
85H	Port 0, bit 5 (P0.0)
86H	Port 0, bit 6 (P0.0)
87H	Port 0, bit 7 (P0.0)
88H	Timer 0 interrupt, řízení typu (IT0)
89H	Timer 0 interrupt, flag hrany (IE0)
8AH	Timer 1 interrupt, řízení typu (IT1)
8BH	Timer 1 interrupt, flag hrany (IE1)
8CH	Timer 0 řízení běhu (TR0)
8DH	Timer 0 flag přeběhu (TF0)
8EH	Timer 1 řízení běhu (TR1)
8FH	Timer 1 flag přeběhu (TF1)

Tabulka 2-5 Adresy bitů v oblasti SFR (pokračování)

Adresa	Význam
90H	Port 1, bit 0 (P1.0)
91H	Port 1, bit 1 (P1.1)
92H	Port 1, bit 2 (P1.2)
93H	Port 1, bit 3 (P1.3)
94H	Port 1, bit 4 (P1.4)
95H	Port 1, bit 5 (P1.5)
96H	Port 1, bit 6 (P1.6)
97H	Port 1, bit 7 (P1.7)
98H	Receive interrupt flag (RI)
99H	Transmit interrupt flag (TI)
9AH	Receive bit 8 (RB8)
9BH	Transmit bit 8 (TB8)
9CH	Receiv enable (REN)
9DH	Řízení módu sériového přenosu, bit 2 (SM2)
9EH	Řízení módu sériového přenosu, bit 1 (SM1)
9FH	Řízení módu sériového přenosu, bit 0 (SM0)
A0H	Port 2, bit 0 (P2.0)
A1H	Port 2, bit 1 (P2.1)
A2H	Port 2, bit 2 (P2.2)
A3H	Port 2, bit 3 (P2.3)
A4H	Port 2, bit 4 (P2.4)
A5H	Port 2, bit 5 (P2.5)
A6H	Port 2, bit 6 (P2.6)
A7H	Port 2, bit 7 (P2.7)
A8H	Enable external interrupt 0 (EX0)
A9H	Enable Timer 0 interrupt (ET0)
AAH	Enable external interrupt 1 (EX1)
ABH	Enable Timer 1 interrupt (ET1)
ACH	Enable serial port interrupt (ES)
AFH	Enable all interrupts (EA)
B0H	Sériový interfejs, přijímaná data (RXD)
B1H	Sériový interfejs, vysílaná data (TXD)
B2H	Interrupt 0, vstup (/INT0)
B3H	Interrupt 1, vstup (/INT1)
B4H	Timer/counter 0 externí vstup (T0)
B5H	Timer/counter 1 externí vstup (T1)
B6H	Write signál (externí paměť dat) (/WR)
B7H	Read signál (externí paměť dat) (/RD)
B8H	Priorita externího přerušení 0 (PX0)
B9H	Priorita přerušení pro timer 0 (PT0)
BAH	Priorita externího přerušení 1 (PX1)
BBH	Priorita přerušení pro timer 1 (PT0)
BCH	Priorita přerušení sériového interfejsu (PS)
D0H	Parity flag (P)
D1H	Flag pro všeobecné použití (PSW).1
D2H	Overflow flag (OV)
D3H	Register bank select bit 0 (RBS0)
D4H	Register bank select bit 1 (RBS1)
D5H	Flag 0 (F0)
D6H	Auxiliary carry flag (AC)
D7H	Carry flag (CY)
E0H	Akumulátor, bit 0 (ACC.0)
E1H	Akumulátor, bit 1 (ACC.1)
E2H	Akumulátor, bit 2 (ACC.2)

Tabulka 2-5 Adresy bitů v oblasti SFR (dokončení)

Adresa	Význam
E3H	Akumulátor, bit 3 (ACC.3)
E4H	Akumulátor, bit 4 (ACC.4)
E5H	Akumulátor, bit 5 (ACC.5)
E6H	Akumulátor, bit 6 (ACC.6)
E7H	Akumulátor, bit 7 (ACC.7)
F0H	Multiplikační registr, bit 0 (B.0)
F1H	Multiplikační registr, bit 1 (B.1)
F2H	Multiplikační registr, bit 2 (B.2)
F3H	Multiplikační registr, bit 3 (B.3)
F4H	Multiplikační registr, bit 4 (B.4)
F5H	Multiplikační registr, bit 5 (B.5)
F6H	Multiplikační registr, bit 6 (B.6)
F7H	Multiplikační registr, bit 7 (B.7)

2.9.1.5. Přístup na vstupně-výstupní obvody

Pro přístup na vstupně-výstupní obvody byl zaveden příkaz PBYTE. Jeho funkce je stejná jako u příkazu XBYTE, avšak na rozdíl od příkazu XBYTE se neprovádí zpětná kontrola zápisu. To umožňuje uživateli zapisovat do vstupně-výstupních obvodů, které bývají u MCS 8051 připojeny jako externí datová paměť. Zápis na vstupně-výstupní obvody je možný i příkazem XBYTE, to ale způsobí chybu číslo 04 (viz přílohu).

2.9.2 Specifikace zadáním klíčového slova

Tato skupina příkazů umožňuje zobrazovat nebo měnit obsah registrů a speciálních funkčních registrů. Tabulka 2-6 uvádí přehled jednotlivých klíčových slov s krátkým popisem jejich významů. Vzhledem k tomu, že registry i SFR leží v oblasti paměti na čipu, je zřejmé, že i příkazy popsané v předchozí kapitole mohou měnit obsah registrů a SFR, avšak pro větší přehlednost a názornost komunikace jsou v monitoru UMON-51 implementovány i následující příkazy.

Syntaxe

klíčové_slovo[=byte]<cr>

Příklady použití:

sekvence	komentář
DPTR<cr>	zobrazí hodnotu datového pointru, např.: DPTR=5600
R0=0EF<cr>	registr R0 ve vybrané sadě registrů bude po provedení příkazu obsahovat hodnotu EFH
TM1=1020<cr>	dosadí do vyššího byte timeru 1 (TH1) hodnotu 10H, do nižšího byte timeru 1 (TL1) hodnotu 20H

PSW<cr>	binárně zobrazí stavové slovo programu (PSW se jako jediný registr zobrazí vždy binárně)

Tabulka 2-6 Klíčová slova pro přístup do paměti

klíčové_slovo	význam
ACC	akumulátor
B	multiplikační registr B
R0	registr 0
R1	registr 1
R2	registr 2
R3	registr 3
R4	registr 4
R5	registr 5
R6	registr 6
R7	registr 7
SP	ukazatel zásobníku
PSW	stavové slovo programu
DPTR	16-ti bitový datový ukazatel
TM0	16-ti bitový timer 0
TM1	16-ti bitový timer 1

2.9.3. Příkaz ASM

Příkaz ASM umožňuje zapisovat program v mnemokódu asemsembleru 8051.

Syntaxe

ASM[ORG adresa]<cr>

Monitor zobrazí na novém řádku adresu, na kterou bude instrukce zapsána a očekává vstup textového řetězce, který odpovídá instrukci v mnemonice 8051 (přejde do režimu asembleru). Po zadání instrukce přechází na nový řádek se zobrazením nové adresy, na níž bude nová instrukce zapsána. Ukončení režimu asembleru je možné kdykoliv stiskem <ESC> nebo na začátku nového řádku stiskem <cr> (zadáním prázdného řádku). Pokud chybí část 'ORG adresa', začíná se od poslední lokace použité při předchozím příkazu ASM. Tato adresa je po signálu reset nastavena na hodnotu 0000H.

Následující text ukazuje příklad zapsání části programu uživatelem:

sekvence	komentář
ASM ORG 2200<cr>	monitor přejde do režimu asembleru. Na nový řádek zobrazí zadanou adresu a očekává vstup instrukce v mnemonice 8051, tedy: 2200-

MOV A,8<cr>	monitor zapíše kód odpovídající této instrukci do paměti programu od adresy 2200H, odpovídajícím způsobem zvětší hodnotu ukazatele pro režim asembleru (délka instrukce je dva byte) a na nový řádek zobrazí nový stav ukazatele a očekává další vstupy: 2212-
<cr>	tento vstup ukončuje režim asembleru a vrací monitor do režimu očekávání příkazu

```
---------
! POZOR !
---------
```

Pokud uživatel instrukcí ASM přepíše lokace využívané systémem (viz kap.3.2), může dojít k chybnému chování systému.

2.9.4. Příkaz DASM

Příkaz DASM umožňuje zobrazavat hodnoty v paměti programu formou instrukcí 8051 (disasemblovat paměť).

Syntaxe

DASM part<cr>

Příkaz DASM zobrazí obsah paměti programu formou instrukcí v mnemonice 8051. Monitor automaticky dokompletuje instrukci, i v případě, že výraz part zahrnuje pouze její začátek. Po ukončení disasemblování se monitor vrací do režimu očekávání příkazu.

sekvence	komentář
DASM 2217<cr>	zobrazí instrukci, jejíž operační kód začíná na adrese 2217H, tedy např.: 2217=020137 LJMP 0137 zobrazení začíná adresou začátku instrukce, pak následuje znak přiřazení, za nímž je zobrazen operační kód instrukce a za ním následuje mnemonická reprezentace tohoto kódu
DA 0 TO 7F<cr>	zobrazí instrukce na úseku paměti od adresy 0 až po adresu 7FH

2.9.5. Příkaz STACK

Příkaz STACK umožňuje zobrazovat obsah zásobníkové paměti. Příkaz má následující syntaxi:

Syntaxe

STACK [byte]<cr>

Příkaz STACK zobrazí obsah zásobníkové paměti počínaje vrcholem zásobníku. Počet zobrazených lokací udává výraz byte, který zde vyjadřuje počet lokací (hexadecimálně), které mají být zobrazeny. Pokud chybí nebo vyjadřuje větší počet lokací, než je od vrcholu zásobníku po lokaci 0, končí výpis zobrazením lokace 0.

sekvence	komentář
STACK<cr>	zobrazí obsah zásobníkové paměti od vrcholu zásobníku až po lokaci 0; tedy např.: SP STACK 07 89 06 78 05 00 04 36 03 45 02 11 01 43 00 07
STACK 3<cr>	zobrazí 3 lokace zásobníkové paměti počínaje vrcholem, např.: SP STACK 07 89 06 78 05 00

2.9.6. Příkaz REGISTERS

Příkaz REGISTERS zobrazuje obsah nejdůležitějších registrů.

Syntaxe

REGISTERS<cr>

Činnost příkazu nejlépe osvětlí následující příklad:

sekvence	komentář
REG<cr>	zobrazí nejdůležitější registry níže popsaným způsobem: PC SP ACC B R0 R1 R2 R3 R4 R5 R6 R7 DPTR PSW xxxx xx xx xx xx xx xx xx xx xx xx xx xxxx iiiiiiii (výrazy xx nebo xxxx odpovídají zobrazení v hexadecimální formě a výraz iiiiiiii zobrazení v binární formě)

2.9.7. Příkaz RBS

Příkaz RBS zobrazí nebo změní vybranou sadu registrů.

Syntaxe

RBS[={0,1,2,3}]<cr>

Následující příklad ukazuje použití příkazu RBS

sekvence	komentář
RBS<cr>	zobrazí číslo vybrané sady registrů, tedy např.: RBS=00
RBS=1<cr>	vybere sadu registrů 1, tj. následující operace s registry budou probíhat se sadou registrů 1

2.10. Příkazy komunikace s hostitelským systémem

Monitor UMON-51 je vybaven skupinou příkazů pro komunikaci s hostitelským systémem. Tímto systémem může být libovolný mikropočitačavý systém vybavený sériovým rozhraním podle standardu RS-232C.

Spojení s hostitelským systémem přináší řadu výhod. První znatelnou výhodou je možnost využití schopností hostitelského systému ve spojení s monitorem pro vývoj uživatelského programového vybavení. Pokud je hostitelský systém vybaven assemblerem ASM51, případně i vyššími kompilátory (PL/M51, C-51, PASCAL-51), je možno na hostitelském systému tvořit aplikační programové vybavení a ladit je ve spojení s monitorem.

Monitor je schopen přijímat cílový kód z hostitelského systému (ve formátu Intel HEX) a ukládat tento kód do programové paměti.

Další možností využití spojení monitoru s hostitelským systémem je využití paměťových médií hostitelského systému pro uložení odladěného cílového kódu, případně i zdrojového kódu.

V úvodu bylo již řečeno, že pro monitor UMON-51 je vyvinuto pomocné programové vybavení DEMON-51, které dále rozšiřuje schopnosti monitoru UMON-51 a staví ho na úroveň mnohem dokonalejších vývojových zařízení.

Hlavní předností tohoto pomocného programového vybavení je schopnost symbolického ladění programu, tj. možnost buď nadefinovat symboly uživatelem nebo spolu s cílovým kódem přijmout tabulku symbolů. Pak je možno se na lokace programu nebo proměnné dovolávat jejich symbolickými jmény tak, jak byly nadefinovány uživatelem při tvorbě zdrojového textu.

Programové vybavení DEMON-51 dále rozšiřuje množinu příkazů monitoru UMON-51 o některé další příkazy, jejichž popis se však vymyká této příručce. Autoři Vám doporučují používat program UMON-51 ve spojení s programem DEMON-51, což podstatně zvýší vývojovou mohutnost Vašeho pracoviště i komfort Vaší práce.

Samotný monitor má implementovány pro komunikaci s hostitelským systémem pouze dva příkazy, LOAD a SAVE. Tyto příkazy slouží k zavedení cílového kódu z paměťových médií hostitelského systému do paměti programu monitoru, případně naopak uložení odladěného cílového kódu z paměti programu monitoru do paměti hostitelského systému.

2.10.1. Formát souborů

Jak bylo uvedeno, pro přenos cílového kódu je použit formát Intel HEX. Jedná se tedy o standardní datový formát, který není nutno zde popisovat.

Monitor přijímá jednotlivé záznamy a počítá z nich kontrolní sumu, kterou porovnává s kontrolní sumou vysílanou na konci každého záznamu. Pokud nesouhlasí, vytiskne chybová hlášení a ukončí činnost zavádění cílového kódu do paměti programu.

Při činnosti vysílání cílového kódu do paměti programu monitor převádí obsah paměti programu do formátu Intel HEX a tvoří záznamy o délce 16byte (10H).

2.10.2. Interfejs

Stavebnice ELIS-51 je opatřena na desce procesoru standardním konektorem CANON, který se běžně používá pro sériovou komunikaci. Obsazení jednotlivých špiček konektoru je uvedeno v následující tabulce.

Tabulka 2-7 Obsazení špiček konektoru CANON na desce ELIS-51

Špička	Signál
1	ochranná zem
2	RS-232C vysílaná data (TxD)
3	RS-232C přijímaná data (RxD)
4	neobsazeno
5	neobsazeno
6	neobsazeno
7	signálová zem
8-25	neobsazeno

Baudová rychlost komunikace je po signálu RESET nastavena automaticky na hodnotu 2400Bd při použití krystalu 6MHz, pro krystal 12MHz je to 4800d.

2.10.3 Příkaz SAVE

Syntaxe

SAVE part<cr>

Monitor po zadání tohoto příkazu začne po sériovém kanálu vysílat na připojené zařízení obsah paměti programu, počínaje zadanou adresou. Provádí konverzi dat do formátu Intel HEX. Délka záznamu je nastavena na 16 byte dekadicky (10H). Na konci každého záznamu je kontrolní suma. Poslední záznam obsahuje startovací adresu, která odpovídá první adrese, od níž bylo ukládání zahájeno. Nakonec monitor odvysílá znak ukončení souboru, což je znak 1AH (CTL-Z), což je běžná konvence používaná operačními systémy CP/M nebo ISIS.

2.10.4.Příkaz LOAD

Syntaxe

LOAD<cr>

Po zadání tohoto příkazu monitor očekává na sériovém vstupu data ve formátu Intel HEX, která mají být uložena do paměti programu. Na konzolu monitor vypíše hlášení 'HEX LOAD MODE'. Pak očekává vstup jednotlivých záznamů přijímaného hexadecimálního souboru. Délka těchto záznamů není prakticky ohraničena. Monitor z každého záznamu počítá kontrolní sumu, kterou po příjmu celého rekordu porovnává s kontrolní sumou, která je součástí přijímaného záznamu. Pokud tyto kontrolní sumy nesouhlasí, vypíše monitor chybové hlášení, ukončí příjem cílového kódu a přejde do režimu očekávání příkazu. Jednotlivé záznamy mohou, ale nemusí být ukončeny znaky konec řádku (cr-lf, 0DH,0AH) a na konci souboru může, ale nemusí být znak konec souboru (CTL-Z, 1AH).

2.10.5. Komunikace s hostitelským systémem

S programem UMON-51 jsou stadardně dodávány komunikační programy TER24, TER48, SEND24 a SEND48. Programy TER slouží k tomu, aby se Váš PC choval jako sériový terminál. Programy SEND slouží k odvysílání přeloženého cílového kódu ve formátu Intel HEX do paměti RAM ve Vaší aplikaci. Číslo 24 nebo 48 určuje baudovou rychlost komunikace, tedy buď 2400 Bd nebo 4800 Bd.

Pokud potřebujete v režimu komunikace zavést do paměti RAM Váš aplikační program, použijte dříve popsaný příkaz LOAD. Po zobrazení hlášení 'HEX LOAD MODE' stiskněte CTL-C, čímž ukončíte činnost programu TER. Nyní použijte program SEND k přenosu Vašeho aplikačního programu do paměti RAM aplikace. Program SEND má vlastní jednoduché menu, které nepotřebuje komentář. Po odvysílání aplikačního programu použijte znovu program TER a můžete pokračovat v odlaďování.

2.11. Seznam příkazů v abecedním pořádku

```
ACC[=byte]
ASM[ ORG adresa]
B[=byte]
BR[=ON]
  [=OFF]
BR0[=adresa]
BR1[=adresa]
CAUSE
CBYTE adresa[=byte[,<cr>byte[,'string',...]
CBYTE part[=byte]
CBYTE part=memory_type part
DASM part
DBYTE adresa[=byte[,<cr>byte[,'string',...]
DBYTE part[=byte]
DBYTE part=memory_type part
DPTR[=adresa]
GO[ FROM adresa][ ARM adresa][ FOREVER]
                      [ TILL BR]
                      [ TILL BR0]
                      [ TILL BR1]
GR[=FOREVER]
  [=TILL BR]
  [=TILL BR0]
  [=TILL BR1]
LOAD
PC[=adresa]
PPC
PSW[=byte]
R0[=byte]
R1[=byte]
R2[=byte]
R3[=byte]
R4[=byte]
R5[=byte]
R6[=byte]
R7[=byte]
RBIT adresa[=bit[,<cr>bit,...]
RBIT part[=bit]
RBIT part=memory_type part
RBS[={0,1,2,3}]
RBYTE adresa[=byte[,<cr>byte[,'string',...]
RBYTE part[=byte]
RBYTE part=memory_type part
REGISTERS
RESET{ BR,BR0,BR1,CHIP}
SAVE part
SP[=byte]
STACK[ byte]
STEP[ FROM adresa][,memory_type adresa]
TM0[=adresa]
TM1[=adresa]
XBYTE adresa[=byte[,<cr>byte[,'string',...]
XBYTE part[=byte]
XBYTE part=memory_type part
```

ČÁST 3. DALŠÍ VŠEOBECNÉ INFORMACE

3.1. Funkční omezení

Vzhledem k tomu, že monitor UMON-51 nevyužívá žádný speciální hardware, plynou z principů jeho činnosti některá funkční omezení.

Nejdůležitějším omezením jsou pravidla pro používání programových zarážek. Je vždy nutné umístit zarážku na adresu operačního kódu instrukce, tj. na její první byte. V opačném případě se neprovede přerušení programu a navíc dojde k přepsání části uživatelského programu (3byte) kódem instrukce LCALL 0006H.

Vzhledem k tomu, že mechanismus provádění přerušení běhu uživatelského programu způsobí dočasné přepsání programu uživatele, je možné umísťovat programové zarážky nebo krokovat program pouze v paměti RAM.

Konečně je nutno přísně dodržovat pravidla umísťování programových zarážek tak, jak byla uvedena v kap. 2.8.1.2.

Druhé důležité omezení plyne ze způsobu komunikace s terminálem nebo hostitelským systémem. Pro tuto komunikaci je využito sériového interfejsu na čipu. Je tedy zřejmé, že uživatelský program nemůže tento sériový interfejs využívat k jiným účelům. Je sice možné, aby program uživatele využíval sériový interfejs na čipu (např. vysílání zpráv atp.), pak je ale vhodné využít služeb poskytovaných monitorem (viz kap. 4.2.). Pokud uživatel nechce využít služby monitoru a napíše vlastní komunikační rutiny, může dojít ke kolizím komunikace při ladění uživatelského programového vybavení. Z tohoto důvodu nelze psaní vlastních komunikačních rutin doporučit, naopak je lépe využít služeb monitoru, což šetří čas i práci.

3.2. Mapování paměti

Monitor UMON-51 principiálně neklade uživateli žádná omezení co se týče přístupu do paměti. Proto je možno oslovovat celý rozsah jednotlivých druhů paměti mikrokontroléru 8051 tak, jak tento rozsah uvádí tab. 2-2.

Z implementace na konkrétní hardware řídícího systému, na kterém je monitor UMON-51 použit, však plynou některá omezení. Protože tato příručka přihlíží ke zvláštnostem plynoucím z implementace monitoru na řídící systém ELIS-51, uvádí následující tabulka mapu paměti na tomto systému.

Tabulka 3-1 Mapování paměti UMON-51 na systému ELIS-51

rozsah adres	paměť programu	externí paměť dat
0000H - 1FFFH	program UMON-51	blok uživatele
2000H - 3DFFH	paměť RAM pro programy uživatele	neobsazeno
3E00H - 3FFFH	pracovní oblast monitoru UMON-51	neobsazeno
4000H - FFFFH	neobsazeno	neobsazeno

Takto konfigurovaná paměť odpovídá použití paměťových čipů s kapaciou 8kB.

3.3 Omezení uživatele

Z popsaného mapování paměti plyne i omezení uživatele při využívání paměťového prostoru. V podstatě existuje jediné omezení, kterým je zákaz používání paměti, která slouží jako pracovní oblast monitoru. V této oblasti jsou uloženy vnitřní proměnné monitoru při vykonávání programu uživatele a je zde též uložen vnitřní stav paměti na čipu tak, jak byl změněn či nastaven programem uživatele.

Zápis do této části paměti programu může vést k ne správnému chování programu uživatele nebo k porušení činnosti programu UMON-51.

ČÁST 4. PROGRAMOVÁNÍ S MONITOREM UMON-51

4.1. Programování s monitorem UMON-51

Pro tvorbu uživatelských programů je monitor UMON-51 vyba- ven přímým řádkovým asemblerem. Je rovněž možné zapisovat do paměti programu uživatelské programy přímo zápisem operačních kódů instrukcí. Konečně je také možné programy vyvíjet na hostitelském systému a do paměti řídícího systému, na němž je monitor implementován tyto programy přenést pomocí příkazu LOAD.

4.2. Přístup k podprogramům monitoru

Aby bylo psaní uživatelských programů ve spolupráci s monitorem UMON-51 usnadněno, poskytuje monitor UMON-51 uživateli celou řadu služeb. Tyto služby (vlastně podprogramy monitoru) jsou uživateli dostupné formou vstupního vektoru monitoru. Tento vstupní vektor se nachází na začátku programu monitor, tj. v paměti programu od adresy 0000H.

Tabulka 4-1 obsahuje popis vstupního vektoru monitoru UMON-51.

4.2.1. Přerušovací vektor

Body vstupního vektoru EINT0, EINT1, TCNT0, TCNT1, TCNT2 a SERINT neslouží pro volání z program uživatele, ale jsou použity pro obslužné rutiny přerušení. Jsou umístěny v paměti ROM monitoru a směřují na odpovídající místa v paměti RAM pro program uživatele tak, aby jejich fyzické umístění v paměti programu uživatele odpovídalo adresám, které jsou definovány jako vstupní body přerušení pro 8051.

Této skutečnosti lze výhodně využít, neboť program, který byl odladěn pod monitorem a který využívá přerušení, lze pak bez úprav prostou relokací nahrát do paměti ROM, která bude ve finálním zařízení umístěna místo programu UMON-51.

Tabulka 4-1 Vstupní vektor monitoru UMON-51

adresa	název	činnost
0000H	RESETMON	skok do inicializační procedury, nastavení UMON-51 jako po resetu
0003H	EINT0	skok na obsluhu externího přeruše- ní 0
0009H	URESET	nastavení stavu procesoru uživate- le do stavu stejném jako po resetu
000BH	TCNT0	skok na obsluhu přerušení od čítače/časovače 0
000EH	ASCII_HEX	převod ASCII na hexa
0011H	INIT_IO	nastavení sériového interfejsu pro komunikace s konzolou
0013H	EINT1	skok na obsluhu externího přeruše- ní 0
0019H	NEWLINE	nový řádek
001BH	TCNT1	skok na obsluhu přerušení od čítače/časovače 1
0021H	EXIT	skok do monitoru bez provádění inicializace (warm start)
0023H	SERINT	skok na obsluhu přerušení od sériového interfejsu na čipu
0026H	GET_STRING	vstup textového řetezce pro zpra- cování užvatelským programem
0029H	CONIN	vstup znaku ze sériového interfejsu (konzole)
002BH	TCNT2	skok na obsluhu přerušení od čítače/časovače 2 (cpu=8052)
002EH	CSTS	status konzole
0030H	CONOUT	výstup znaku na sériový interfejs (konzolu)
0032H	LSTBYTE	zobrazí předaný parametr typu BYTE
0034H	LSTWORD	zobrazí předaný parametr typu WORD
004BH	PRINT_STRING	vypíše string na konzolu, předáva- ný parametr je adresa stringu

4.2.2. Volání RESETMON

Službu RESETMON může aplikační program využít pouze pro ukončení své činnosti a předání řízení monitoru, přičemž se provede inicilaizace systému i stavu procesoru uživatele stejně jako při externím signálu reset.

Služba nevyžaduje žádné parametry. Tuto službu je možné volat přímo z konzole, pokud je třeba nastavit stav ekvivalentně jako při signálu reset a přitom signál reset není k dispozici nebo je nežádoucí.

Příklady:

```
LJMP    0000      ;ukončení běhu programu uživatele a inicia-
                  ;lizace monitoru i stavu procesoru uživate-
                  ;le jako při signálu reset
GO FROM 0<cr>     ;přímé volání z konzoly
```

4.2.3. Volání URESET

Použití tohoto bodu vstupního vektoru je obdobné volání RESETMON, které bylo posáno v kap. 4.2.2 s tím rozdílem, že se neprovede inicializace monitoru, ale pouze stavu procesoru uživatele. To znamená, že toto volání má za následek nastavení stavu procesoru uživatele ekvivalentnímu s nastavením, které se provede po signálu reset. To umožňuje uživateli simulovat signál reset tak, jak bude vykonán při nahrání aplikačního programu do paměti ROM a jeho provozování bez přítomnosti monitoru UMON-51.

Příklady:

```
LJMP    0009           ;ukončení běhu programu uživatele a
                       ;inicializace stavu procesoru uživa-
                       ;tele jako při signálu reset
GO FROM 9<cr>          ;přímé volání z konzoly
```

POZNÁMKA

Přímé volání služby URESET z konzole má stejný následek, jako použití příkazu RESET CHIP, viz kap. 2.8.2.2.

4.2.4. Nastavení komunikačních kanálů

Pokud uživatel chce použít sériového interfejsu na čipu ve svém programovém vybavení, je vhodné využít služeb monitoru. Pro inicializaci sériového interfejsu je k dispozici služba INIT-IO, která zabezpečí nastavení sériového interfejsu do režimu duplexního provozu se zvolenou rychlostí, formát je 8 datových bitů, 1 start bit, dva stop bity, bez kontroly parity. Služba vyžaduje jeden parametr, a to koeficient pro nastaveni baudové rychlosti komunikace.

Parametr se předává v akumulátoru. Tabulka 4-2 obsahuje seznam hodnot pro nastavení zvolené baudové rychlosti.

Příklady:
```
MOV     A,#3            ;Inicializace sériového interfejsu na
LCALL   11              ;rychlost 2400Bd (krystal 6MHz)
```

Tabulka 4-2 Hodnoty pro nastaveni baudové rychlosti

(ACC)	rychlost Bd (@6MHz)	rychlost Bd (@12MHz)
0	300	600
1	600	1200
2	1200	2400
3	2400	4800
4	4800	9600
5	9600	19200

4.2.5. Služba EXIT

Tato služba zajišťuje normální návrat uživatelského programu do monitoru (warm start). Neprovádí inicializace monitoru ani stavu procesoru uživatele. Stav procesoru uživatele se zachová.

Příklad:

```
LCALL   0021    ;ukončení programu, návrat do monitoru
```

4.2.6. Výstup znaku na konzolu (služba CONOUT)

Tato služba zajistí výstup znaku na konzolu. Znak se předává jako parametr v registru R2. Před prvním použitím této služby je třeba voláním INIT-IO zajistit nastavení sériového interfejsu.

Příklady:
```
MOV     R2,#41
LCALL   0030            ;vypíše na konzolu znak 'A'
```

4.2.7. Vstup znaku z konzole (služba CONIN)

Tato služba zajistí vstup znaku z konzole do akumulátoru. Před prvním použitím této služby je třeba voláním INIT-IO zajistit nastavení sériového interfejsu.

Příklady:
```
LCALL   0029            ;čeká na vstup znaku z konzole
```

4.2.8. Stav konzole (služba CSTS)

Tato služba vrací nastavený příznak CY, pokud je ve vyrovnávacím bufferu sériového interfejsu připraven znak z konzoly pro převzetí. V opačném případě nuluje CY. Před prvním použitím této služby je třeba voláním INIT-IO zajistit nastavení sériového interfejsu.

Příklady:

```
INKEY:  LCALL   002E            ;čti status konzole
        JNC     INKEY           ;čekej na příchod znaku
        CLR     98              ;nuluj příznak znak připraven
        MOV     A,99            ;přečti znak z bufferu
```

4.2.9. Služba NEWLINE

Tato služba zajistí přechod kurzoru na nový řádek. Před jejím prvním použitím je třeba službou INIT-IO nastavit komunikační kanály. Služba nepožaduje žádné parametry.

Následující příklad způsobuje výmaz obrazovky (25-ti násobné vyslání znaků cr-lf na konzolu).

Příklady:

```
        MOV     R7,#19          ;19H=25
CLS:    LCALL   0019            ;vyšli cr-lf na konzolu
        DJNZ    R7,CLS          ;opakuj pokud R7<>0
```

4.2.10. Zobrazení čísla typu byte (služba LSTBYTE)

Pomocí této služby je zajištěna konverze hexadecimální osmibitové konstanty do dvou znaků ASCII a jejich zobrazení na konzolu. Před prvním použitím této služby je třeba službou INIT-IO nastavit komunikační kanály. Služba požaduje jeden parametr, a to byte, který má být zobrazen, v registru R2.

Příklady:

```
PUSH    E0              ;Uschová hodnotu v akumulátoru
MOV     R2,#41
LCALL   0030            ;vypíše na konzolu znak 'A'
MOV     R2,#3D
LCALL   0030            ;vypíše na konzolu znak '='
POP     E0              ;obnoví akumulátor
MOV     R2,A
LCALL   0032            ;zobrazí obsah akumulátoru
MOV     R2,#48
LCALL   0030            ;vypíše na konzolu znak 'H'
LCALL   0019            ;vyšli cr-lf na konzolu
```

4.2.11. Zobrazení čísla typu word (služba LSTWORD)

Pomocí této služby je zajištěna konverze hexadecimální šestnáctibitové konstanty do čtyř znaků ASCII a jejich zobrazení na konzolu. Před prvním použitím této služby je třeba službou INIT-IO nastavit komunikační kanály. Služba požaduje jeden parametr, a to adresu (word), která má být zobrazena, ve dvojici registrů R2,R3.

Příklady:

```
MOV     R2,83           ;vyšší byte adresy (DPH) do R2
MOV     R3,82           ;nižší byte adresy (DPL) do R3
LCALL   0034            ;zobraz obsah DPTR
```

4.2.12. Služba ASCII-HEX

Tato služba zajišťuje konverzi čísla vyjádřeného ASCII znakem do hexadecimální formy. Služba vyžaduje jeden parametr, a to znak v registru R2. Služba testuje, zda tento znak reprezentuje číslo a vrací příznak chyby (nastavením CY=1), pokud ne. Pokud ano, vrací služba v dolních čtyřech bitech akumulátoru odpovídající hexadecimální číslo, horní čtyři bity nuluje. Zároveň nastaví CY=0.

Příklady:

```
LCALL   0029            ;znak ze sériového kanálu
LCALL   000E            ;konvertuj ASCII znak do hexa
JC      2000            ;chyba při neplatném znaku
SWAP    A               ;převeď na významnější půli
MOV     R7,A            ;přechodně ulož do R7
LCALL   0029            ;druhý znak
LCALL   000E            ;konvertuj do hexa
JC      2000            ;chyba při neplatném znaku
ADD     A,R7            ;přičtením významnější půle
                        ;vzniká celé osmibitové číslo
```

Služba, která zajistí výpis chybového hlášení, je popsána v následujícím příkladu.

4.2.13. Výpis stringu na konzolu

Výpis stringů na konzolu zajišťuje služba LSTSTRING. Služba vyžaduje jeden parametr, a sice adresu stringu, předanou ve dvojici registrů R2,R3. String musí začínat údajem o délce stringu, což je jeden byte. Z toho plyne, že maximální délka stringu je 255 znaků. Za tímto údajem musí bezprostředně následovat řetězec znaků ASCII, tvořící string.

Před prvním použitím služby je nutno zajistit nastavení komunikačních kanálů prostřednictvím služby INIT-IO.

Následující příklad ukazuje výpis chybového hlášení

použitého v příkladu z kap. 4.2.12.) s využitím služby LSTSTRING.

Příklady:

```
MOV     R2,#20          ;MSB adresy stringu
MOV     R3,#17          ;LSB adresy stringu
LCALL   004B            ;vypiš string
LCALL   0019            ;nový řádek
RET                     ;návrat z podprogramu
```

Chybové hlášení je možné zapsat na adresu 2017 např. takto:

```
CBYT 2017=0E,'INVALID FORMAT'<cr>
```

POZNÁMKA

Pokud uživatelský program používá více služeb, využívajících sériového interfejsu na čipu, pak je nutno před prvním použitím těchto služeb nastavit službou INIT-IO sériový interfejs na čipu. Je samozřejmé, že toto nastavení stačí provést pouze jednou před prvním použitím jedné ze služeb využívajících sériový interfejs.

4.2.14. Vstup textového řetězce (služba GETSTRING)

Tato služba umožňuje uživatelskému programu pracovat s řetězci znaku, např. pro předávání parametrů pro zpracování uživatelským programem atp. Vstupující řetězec je možné řádkově editovat podle stejných pravidel, jako se editují příkazy monitoru UMON-51. Před prvním použitím této služby je nutno voláním INIT-IO nastavit komunikační kanály.

Služba vyžaduje jeden parametr, a to adresu, na níž bude volajícímu programu string k dispozici. Tento parametr se předává v datovém ukazateli (DPTR). Je třeba počítat s tím, že voláním této služby se ovlivní 10 lokací v oblasti zásobníku.

Služba vrací na zadané adrese řetězec znaků, přičemž první byte na zadané adrese obsahuje počet znaků tvořících string, následující byte jsou vlastní znaky stringu. String je zakončen znakem <cr>; tento znak se započítává do celkové délky.

Příklad:

```
MOV     DPTR,#2800
LCALL   0026
```

Vstupující řetězec znaků se vrací na adrese 2801H a je ukončen znakem <cr> (kód 0DH). Na adrese 2800H bude délka zadaného stringu. Vstoupil-li např. string 'AHOJ, bude obsah paměti na adrese 2800H následující:

```
2800H: 05,41,48,4F,4A,0D
```

4.3. Příklad programu s využitím interruptu

Následující ukázkový program demonstruje využití přerušovací architektury 8051. Externí přerušení 0 je použito jako obsluha žádosti o zobrazení stavu 16-ti bitového čítače TM0.

Program je napsán jako zdrojový text pro překlad assemblerem ASM51.

```
NAME    DEMO_EINT0

INIT_IO CODE    0011H
LSTWORD CODE    0034H
NEWLINE CODE    0019H

CSEG    AT      2000H
        LJMP    START
        LJMP    EINT0

ORG     2020H

START:  MOV     SP,#1FH         ;Nastavení stacku
        MOV     A,#3            ;Koficient pro 2400Bd (6MHz)
        CALL    INIT_IO         ;Nastavení komunikace
        ORL     TMOD,#05H       ;Timer/counter 0 použit jako
                                ;16-ti bitový čítač
        SETB    IT0             ;Přerušení hranou
        CLR     IE0             ;Ošetření příznaku provedení
                                ;přerušení
        SETB    PX0             ;Vyšší priorita přerušení pro
                                ;externí přerušení 0
        SETB    EX0             ;Povol externího přerušení 0
        SETB    EA              ;Povol přerušení
        SETB    TR0             ;Spuštění čítače 0
        JMP     $               ;Čeká ve smyčce na přerušení


USING   1

EINT0:  PUSH    PSW
        PUSH    ACC
        SETB    PSW.3
        CLR     PSW.4           ;Výběr sady registrů 1
        CLR     TR0             ;Zastav čítač 0
        MOV     A,TL0           ;Nižší byte čítače
        MOV     R3,A            ;Přesune do R3
        MOV     A,TH0           ;Vyšší byte čítače
        MOV     R2,A            ;Přesune do R2
        SETB    TR0             ;Opět spustí čítač
        CALL    LSTWORD         ;Zobrazí obsah čítače
        CALL    NEWLINE         ;Přechod na nový řádek
        POP     ACC
        POP     PSW
        RETI                    ;Návrat z obsluhy přerušení

END
```

Tento program lze po překladu a konverzi na formát HEX Intel přenést do uživatelské paměti programu příkazem LOAD (překladač ASM51 produkuje OBJECT formát, který je nutno pro přenos do monitoru konvertovat pomocí programu OBJHEX, viz /2/).

Program je přeložen od adresy 2000H tak, aby vstupní vektor přerušení mikrokontroléru 8051 odpovídal fyzické adrese zvětšené o ofset 2000H.

PŘÍLOHA SEZNAM CHYBOVÝCH HLÁŠENÍ

Tato příloha obsahuje seznam chybových hlášení monitoru UMON-51 a stručné vysvětlení jejich příčin.

Chyba číslo 00 - FIRMWARE CHECKSUM ERR
Tato chyba se může vyskytnout pouze po signálu reset nebo volání RESET. Je to neodstranitelná chyba. Nejpravděpodobnější příčinou je porucha paměti EPROM nebo závada na desce. Je třeba zkusit vytáhnout paměť EPROM z patice a znovu pečlivě zasunout. Pokud i potom chyba trvá, je nejlépší obrátit se na autory monitoru UMON-51.

Chyba číslo 01 - SYNTAX ERROR
Tato chyba je způsobena zadáním neznámého příkazu (klíčového slova) nebo zkratky. Je třeba zkusit zapsat příkaz znovu podle popisu v tomto manuálu.

Chyba číslo 02 - INVALID TOKEN
Byl zadán správný příkaz, ale je následován nesprávnými parametry nebo větším počtem parametrů, než je přípustné. Přečtěte si popis příkazu a zapište příkaz znovu podle popisu v tomto manuálu.

Chyba číslo 03 - PGM MEMORY FAILURE
Při zpětné kontrole zápisu do programové paměti nesouhlasila přečtená data se zapisovanými. Možnou příčinou je pokus o zápis do paměti typu ROM nebo paměť na zadané adrese není vůbec nainstalována. Přečtěte si kapitolu 3.2. o mapování paměti.

Chyba číslo 04 - DATA MEMORY FAILURE
Při zpětné kontrolo zápisu do externí paměti dat nesouhlasila přečtená data se zapisovanými. Možnou příčinou je pokus o zápis do paměti, která není vůbec nainstalována. Přečtěte si kapitolu 3.2. o mapování paměti.

Chyba číslo 05 - CONTROL MEMORY FAILURE
Při operaci na paměti RAM využívané monitorem došlo k chybě. Zkuste paměť programu uživatele (CMOS-RAM) vytáhnout z patice a znovu zasunout. Pokud chyba trvá i nadále, vyměňte paměť programu uživatele RAM za novou a bezchybnou. Nepomůže-li ani to, hledejte chybu na desce nebo se obraťte na autory UMON-51.

Chyba číslo 06 - UNWRITEABLE MEMORY
Tato chyba je způsobena pokusem povolit vykonání programové zarážky nastavené do paměti ROM nebo do paměti, která není vůbec instalována. Přečtěte si kap. 3.2. o mapování paměti a použijte programovou zarážku pouze v paměti RAM. Stejná chyba se vyskytne při pokusu o krokování v ROM nebo v oblasti nenainstalované paměti programu.

Chyba číslo 07 - UNAPROPRIATE NUMBER
Číslo použité jako parametr příkazu leží mimo povolený rozsah (např. v příkaze DBYT 0E0 je adresa větší než je dovoleno pro typ 8051).

Chyba číslo 08 - INVALID REG BANK NUMBER
Použitý parametr (číslo) v příkaze RBS leží mimo platný rozsah (viz kap. 2.9.7.).

Chyba číslo 09 - PARTITION NOT ALLOWED
Použitý parametr typu part není v tomto příkaze povolen. Smí být použit poze parametr typu adresa.

Chyba číslo 0A - PARTITION BOUNDS ERROR
V parametru typu part je druhá adresa menší než prvá. Porovnejte s kap. 2.4. o parametrech příkazů.

Chyba číslo 0B - COMMAND TOO LONG
Zadaný příkaz obsahuje více než 128 znaků. Příkaz je třeba rozdělit do více částí.

Chyba číslo 0C - NON-CHANGEABLE ITEM
Příkaz který způsobil tuto chybu lze použít pouze k zobrazení, nikoliv ke změně obsahu.

Chyba číslo 0D - INVALID OBJECT FILE
Pokus o zavedení programu v jiném než HEX formátu Intel do paměti (příkazem LOAD). Používejte pouze soubory ve formátu HEX Intel.

Chyba číslo 0E - FILE CHECKSUM ERROR
Při zavádění programu do paměti (příkazem LOAD) nesouhlasí kontrolní suma. Zkuste znovu zavést a pokud se chyba opakuje, pokuste se o nové vytvoření souboru ve formátu Intel HEX.

Chyba číslo 0F - REL OFFSET TOO LARGE
Při zápisu do paměti programu v režimu asembleru došlo k pokusu větvit program instrukcí relativního větvení nai cílovou adresu, která je příliš vzdálena od instrukce větvení. Je třeba jinak uspořádat program nebo použít jinou instrukci.

Chyba číslo 10 - ABS OFFSET TOO LARGE
Při zápisu do paměti programu v režimu asembleru došlo k pokusu větvit program instrukcí absolutního větvení na cílovou adresu, která je příliš vzdálena od instrukce větvení. Je třeba jinak uspořádat program nebo použít jinou instrukci.

Chyba číslo 11 - UNDEFINED OPCODE
Při disasemblování paměti se narazilo na kód, kterému není přiřazena žádná instrukce.

Chyba číslo 12 - ASSEMBLY IMPOSSIBLE
Při zápisu do paměti programu v režimu asembleru došlo k pokusu zadat neexistující instrukci. Seznamte se s přesnou mnemonikou instrukcí 8051.

Chyba číslo 13 - EXCESSIVE ITERATED DATA
Při použití blokového přenosu oblasti paměti (kap 2.9.1.) bylo použito dvakrát parametru typu part, přičemž délka úseku paměti specifikovaná prvním použitím part je jiná, než specifikuje druhé použití parametru typu part (např. CB 0 TO 7=XB 0 TO 8). Použijte parametry typu part tak, aby ohraničovaly stejnou délku nebo použijte parametr typu part pouze jednou (CB 0 TO 7= XB 0).

SEZNAM TABULEK
==============