DPIC / EPIC

Uživatelská příručka

(Verze dokumentu: 1/98)

ASIX®

ASIX s.r.o.
Grafická 37
150 00 Praha 5 - Smíchov

E-mail: asix@asix.cz
WWW: http://www.asix.cz/
Tel./fax: (02) 573 123 78

Obsah

1. Úvodní informace 7

2. Popis hardware emulátoru EPIC16A 9

2.1 Přední panel 9

2.1.1 Popis emulačního konektoru: 9
2.1.2 Popis konektoru sond 10
2.1.3 Barevné označení sond 10

2.2 Zadní panel 10

2.2.1 Připojení emulátoru k PC 10
2.2.2 Popis napájecího konektoru: 11

3. Integrované vývojové prostředí DPIC. 12

3.1 Pracovní plocha 12

3.1.1 Programové objekty integrovaného prostředí 13
3.1.2 Menu 14
3.1.3 Okna výpisů 14
3.1.3.1. Pohyb oknem po pracovní ploše 15
3.1.3.2 Pohyb textem okna 16
3.1.4 Dialogová okna 16
3.1.5 Rolovací lišta 17
3.1.6 Tlačítka 18
3.1.7 Řádkové editory 19
3.1.8 Seznamy 20
3.1.9 Podpůrné programové objekty 20

3.2 Hlavní menu 21

3.2.1 Nabídka ≡ 21
3.2.2 Nabídka File 21
3.2.3 Nabídka Window 23
3.2.4 Nabídka Edit 23
3.2.5 Nabídka Views 24
3.2.6 Nabídka Run 25
3.2.7 Nabídka Debug 26
3.2.8 Nabídka Option 26

3.3 Zdrojový text, projekt a objekt 27
3.3.1 Organizace adresářů 27
3.3.2 Zdrojový text a projekt 28
3.3.3 Soubor objektu 28
3.3.4 Projekt 28
3.4 Editor zdrojového textu 29
3.4.1 Editace textu 29
3.4.2 Další pomocné funkce editoru 30
3.5. Editor datové a programové paměti 31
3.6 Zpětný překladač (Disassembler) 32
3.7 Výpis trasovací paměti (Trace Buffer) 33
3.7.1 Uspořádání trasovací paměti 33
3.7.2 Nastavení a volby 34
3.8 Uživatelské výpisy proměnných (Watch) 35
3.8.1 Formátování a editace hodnoty 36
3.9 Výpis a editace zásobníku 37
3.9.1 Editor položky zásobníku 37
3.9.2 Editor ukazatele zásobníku 37
3.10 Výpisy a editace událostí 38
3.10.1 Výpisy událostí 38
3.10.2 Editace událostí 39
3.11 Globální události 40
3.12 Nastavení emulátoru 41
3.13 Chip options 43
3.14 Nastavení parametrů překladače 44
3.15 Nastavení parametrů prostředí 45
3.16 Nastavení parametrů výpisu pamětí 46
3.16.1 Data Selector 46
3.16.2 Editor nastavení výpisu 47
3.17 Nastavení výpisu trasovací paměti 47

3.18 Nastavení barev prostředí 48

3.19 Pomocné objekty na pracovní ploše 49

3.19.1 Status 49
3.19.2 Programový manažer 50
3.19.3 Indikátor dostupné paměti 50

3.20 Okno chybových hlášení překladače 50

3.21 ASCII tabulka 51

3.22 Calculator 51

3.23 Videostop 52

3.24 Information 52

3.25 Chybová hlášení integrovaného prostředí 52

3.26 Informační hlášení prostředí 55

4. Překladač jazyka ASSEMBLER 57

4.1 Zdrojový text programu 58

4.2 Konstanty 58

4.2.1 Číselné konstanty 58
4.2.2 Textové konstanty 59

4.3 Výrazy 59

4.4 Symboly 60

4.4.1 Symbolická jména konstant 61
4.4.2 Datové typy 61
4.4.3 Návěští 63

4.5 Instrukční soubor procesorů PIC 64

4.5.1 Bajtově orientované instrukce 64
4.5.2 Bitově orientované instrukce 65
4.5.3 Instrukce pracující s konstantou 65
4.5.4 Řídící instrukce 65
4.5.5 Přehled instrukcí rodiny PIC 16C5X 66
4.5.6 Instrukční soubor rodiny PIC 16CXX 71

4.6 Direktivy překladače 72
4.6.1 Přehled direktiv 73
4.6.2 Direktiva BANK 73
4.6.3 Direktiva BIT 73
4.6.4 Direktiva BYTE 74
4.6.5 Direktiva CONST BYTE 74
4.6.6 Direktiva DB 75
4.6.7 Direktiva DW 75
4.6.8 Direktiva END 75
4.6.9 Direktiva EQU 75
4.6.10 Direktivy IF - ELSE - ENDIF 76
4.6.11 Direktiva INCLUDE 76
4.6.12 Direktivy MACRO - LOCAL -ENDM 76
4.6.13 Direktiva ORG 78
4.6.14 Direktiva PRAGMA 78
4.6.15 Direktiva SET 78
4.6.16 Direktiva TABLE 78
4.7 Chybová hlášení překladače a varování 79

Poznámky: 83

1. Úvodní informace

Blahopřejeme Vám, že jste si vybrali náš emulátor EPIC16A. Tento přístroj jsme koncipovali tak, aby Vám přinesl maximální efektivitu práce při vývoji programů pro nejužívanější mikrokontroléry Microchip PIC.

Unikátní konstrukce emulátoru nepoužívá běžný emulační čip, ale moderní programovatelné logické obvody (ASIC). Umožnila dosáhnout všech standardních funkcí, a navíc řady vlastností, z nichž mnohé nejsou obvyklé nebo principiálně dostupné ani u emulátorů podstatně vyšší cenové kategorie:

- i za plného běhu emulovaného procesoru lze zobrazovat nebo modifikovat veškeré registry (tj. File registry i speciální interní registry jako např. PC, STACK, PORT, TRIS, LATCH, OPTION, PRESCALER, WREG, ...). Výjimkou je pouze zakázaná modifikace STACKu v režimu Run.
- rozsáhlé možnosti zastavení procesoru (Break) na základě mnoha různých podmínek, nejen obvyklé libovolně nastavitelné breaky v paměti kódu, ale i breaky v datové paměti s volbou pro přístup k registru, zápis do registru, zápis hodnoty omezené maskou a speciální breaky: přetečení časovače, přetečení trasovacího bufferu, přetečení WDT, přetečení/podtečení stacku, break z externí sondy emulátoru (s volbou náběžné nebo sestupné hrany)
- mimořádně flexibilní nastavení hodinové frekvence emulovaného procesoru v rozsahu 20 kHz až 20 MHz s krokem 40 Hz pomocí interního frekvenčního syntezátoru
- oddělené napájení vnitřních obvodů emulátoru a I/O pinů emulovaného procesoru umožňuje pracovat s aplikačním napájením v plném rozsahu napájecích napětí mikrokontrolérů PIC (3 až 6V), tj. uživatel není nucen používat pouze napětí 5 V
- na emulátoru je k dispozici i stabilizovaný zdroj napětí 5 V s proudovým omezením 100 mA pro napájení periferií emulovaného procesoru
- režim off-line: po opuštění řídícího programu může emulátor pokračovat v emulaci nezávisle na řídícím počítači standardu IBM PC
- přehledná indikace režimu pomocí různobarevných LED pro sledování činnosti emulátoru (i v off-line režimu): oranžová LED - power on/emulační obvod nakonfigurován, červená LED - Halted, žlutá LED - Break, zelená LED - Run/Step, modrá LED - Sleep
- je-li emulátor v činnosti při spuštění ovládacího software, nastaví se prostředí do stavu, ve kterém bylo před jeho opuštěním. Informace o tomto stavu se ukládají do speciální paměti přímo v hardware emulátoru, ne pouze do konfiguračního souboru v PC.

- možnost povolení nebo potlačení resetu z pinu -MCLR a od poklesu napájecího napětí (rozhodovací úroveň je v tomto případě 2V), detekce i krátkého resetu
- flexibilní trasování - vypnuto, zapnuto, nebo trasování pouze vybraných částí programu (lze např. potlačit trasování čekacích smyček, a trasovat pouze průběh hlavního programu). Kapacita trasovacího bufferu je 32 kB, tj. 8192 instrukcí.
- osm sond pro volitelné trasování externích signálů synchronně se čtením I/O pinů emulovaného procesoru
- výstup CLKOUT má správnou frekvenci i fázi odpovídající nastavenému režimu
- korektní emulace přetečení čítačů Watchdog, TMR0 (RTCC) a zápisu do interní EEPROM (PIC16C8x) i v režimu Halted/Step, kdy se příslušné čítače pozastavují mimo platný instrukční cyklus
- flexibilita interních obvodů emulátoru umožňuje softwarové upgrady hardware emulátoru (např. prostřednictvím Internetu), event. individuální modifikace podle přání uživatele

Integrované vývojové prostředí DPIC je určeno pro vývoj aplikací s jednočipovými mikropočítači řady 16C5X a 16CXX firmy Microchip. Obsahuje všechny potřebné nástroje: editor zdrojového textu, překladač a prostředky pro ovládání emulátoru firmy ASIX Praha. Prostředí standardně obsahuje demonstrační verzi simulátoru všech podporovaných klonů rodin 16C5X a 16CXX. Plná verze simulátoru (bez omezení) je dostupná jako samostatný produkt nezávisle na emulátoru.

Integrované prostředí obsahuje překladač jazyka assembler, jehož základní charakteristiky jsou:

- jedná se o řádkový překladač, tj. na jednom řádku může být pouze jeden příkaz
- pracuje se symboly na vyšší úrovni abstrakce, než bývá zvykem u obdobných produktů, umožňuje definovat a používat složitější datové typy (bajty, bity, konstanty, pole, tabulky)
- podmíněné řízení překladu
- makroinstrukce
- možnost volby přednastavené číselné soustavy na dekadickou a hexadecimální (viz. konstanty)
- výstupní binární kód ve formátu INHEX-16 nebo INHEX-8M
- kompatibilní s překladačem firmy Microchip

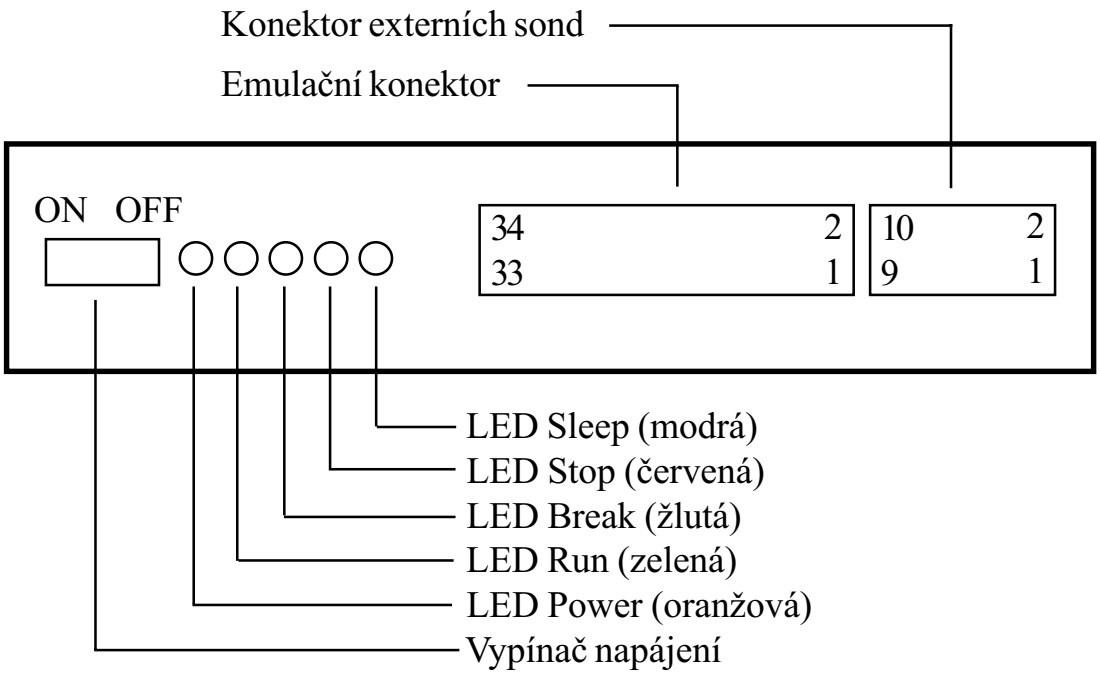
Integrovaném prostředí doporučujeme provozovat na osobním počítač standardu IBM PC s procesorem 486 a 16MB paměti RAM a s přibližně 4MB volného místa na pevném disku. Minimální konfigurace počítače, na níž je možné integrované prostředí spustit, představuje počítač s procesorem 386SX a alespoň 2MB paměti RAM.

2. Popis hardware emulátoru EPIC16A

Rozměry: 168 (š) x 140 (h) x 42 (v) mm

Teplota okolí při provozu: 15 - 30 °C

2.1 Přední panel



LED Power: blikání resp. trvalé svícení označuje nenakonfigurovaný resp. nakonfigurovaný emulační obvod.

2.1.1 Popis emulačního konektoru:

1 - +5 V / 100 mA	10 - OSC1/CLKIN	19 - PORTA1	28 - PORTC0
2 - +5 V / 100 mA	11 - nezapojeno	20 - PORTC4	29 - PORTB2
3 - GND	12 - OSC2/CLKOUT	21 - PORTA2	30 - PORTB7
4 - GND	13 - GND	22 - PORTC3	31 - PORTB3
5 - nezapojeno	14 - PORTC7	23 - PORTA3	32 - PORTB6
6 - nezapojeno	15 - nezapojeno	24 - PORTC2	33 - PORTB4
7 - TOCKI (RTCC)	16 - PORTC6	25 - PORTB0	34 - PORTB5
8 - -MCLR	17 - PORTA0	26 - PORTC1	
9 - VDD	18 - PORTC5	27 - PORTB1	

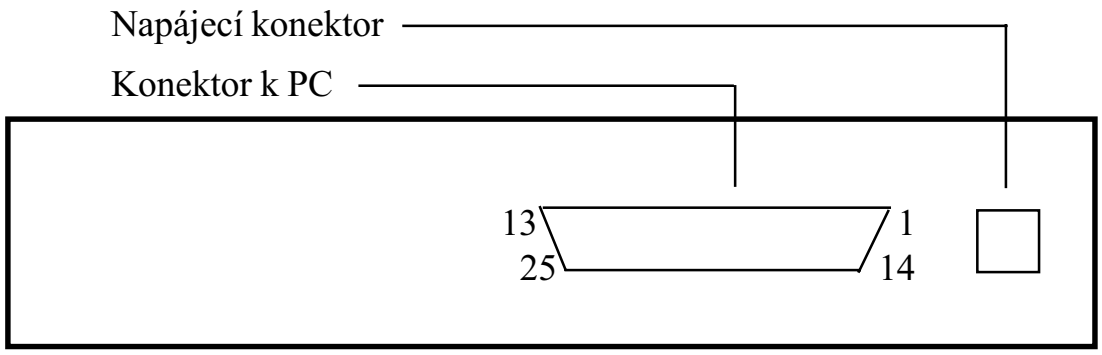
2.1.2 Popis konektoru sond

1 - PROBE 6	6 - PROBE 3
2 - PROBE 7	7 - PROBE 0
3 - PROBE 4	8 - PROBE 1
4 - PROBE 5	9 - GND
5 - PROBE 2	10 - GND

2.1.3 Barevné označení sond

PROBE 0 - bílá	PROBE 4 - zelená
PROBE 1 - šedá	PROBE 5 - žlutá
PROBE 2 - fialová	PROBE 6 - oranžová
PROBE 3 - modrá	PROBE 7 - červená
GND - černá	

2.2 Zadní panel



2.2.1 Připojení emulátoru k PC

Konektor emulátoru je Cannon 25 female, kabel Cannon 25M-25M, propojení 1:1, maximální délka kabelu 2 m.

Upozornění: Pozor na záměnu předepsaného kabelu s jinak zapojenými kabely, např. křížově propojeným kabelem Laplink pro LPT!

1 - STROBE	6 - DATA 4	11 - BUSY	16 - INIT	21 - GND
2 - DATA 0	7 - DATA 5	12 - PE	17 - SLCTIN	22 - GND
3 - DATA 1	8 - DATA 6	13 - SLCT	18 - GND	23 - GND
4 - DATA 2	9 - DATA 7	14 - AUTOLF	19 - GND	24 - GND
5 - DATA 3	10 - ACK	15 - ERR	20 - GND	25 - GND

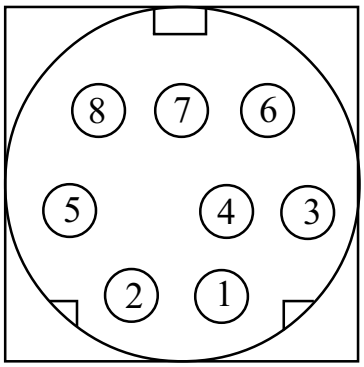
2.2.2 Popis napájecího konektoru:

Konektor AMP Shielded Miniature Circular DIN Plug 8 Pin P/N 749179-1 nebo ekvivalent. Napájecí napětí 5 V ±5% / 1 A, 12 V ±5% / 500 mA.

Doporučený typ napájecího zdroje: Computer Products SCL25-7618.

Důležité upozornění: Při nesprávném připojení napájecích napětí může dojít k vážnému poškození emulátoru!

1 - +5 V	5 - GND
2 - +5 V	6 - GND
3 - +5 V	7 - GND
4 - +12 V	8 - GND

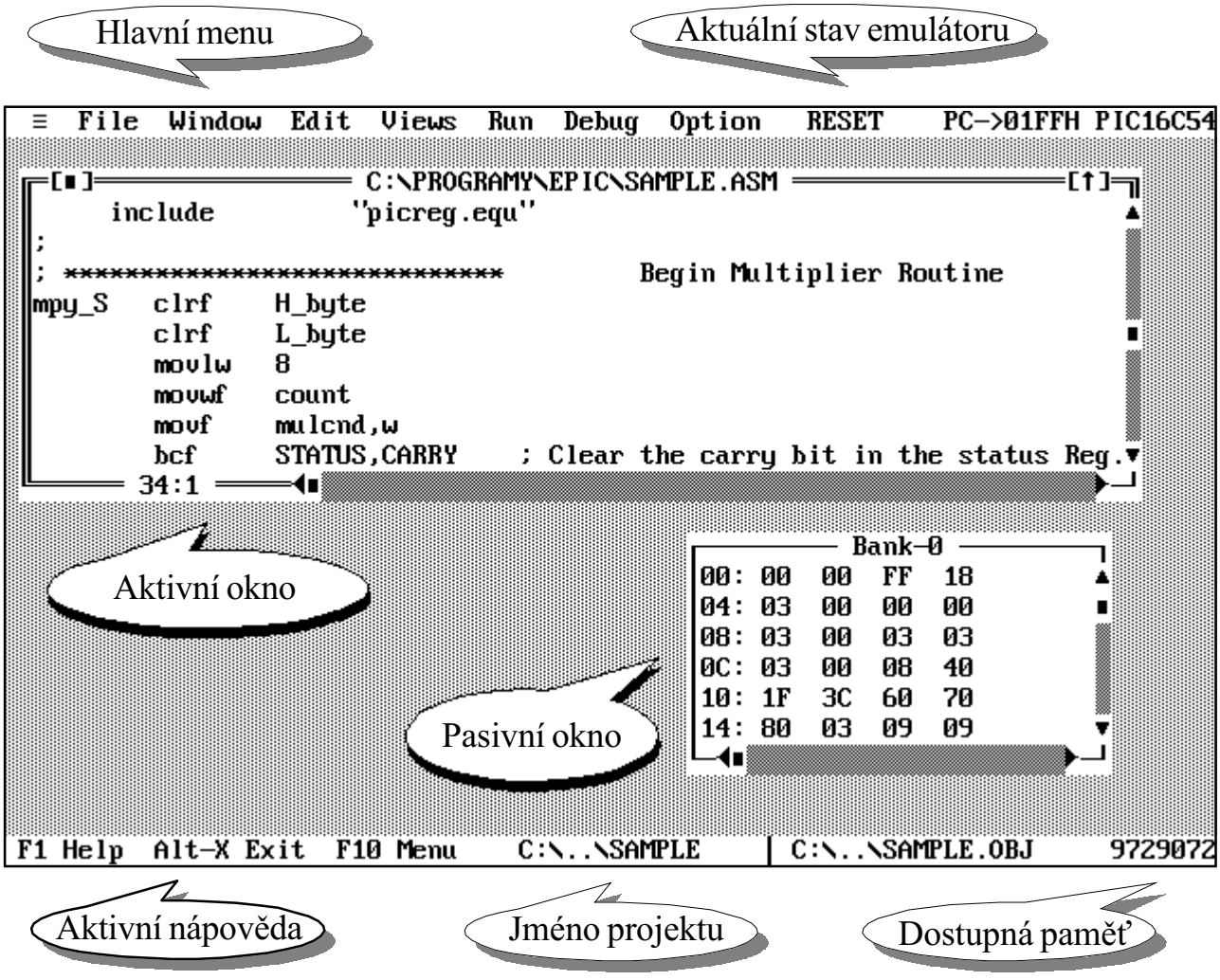


3. Integrované vývojové prostředí DPIC.

Integrované vývojové prostředí je tvořeno několika základními moduly. Do těchto modulů počítáme textový editor, nástroje pro editaci paměti emulovaného čipu, systém pro zadávání a editaci událostí, manažer projektů, překladač a ovladače pro jednotlivé typy emulovaných mikropočítačů. Všechny funkce prostředí jsou dostupné přes základní menu, přičemž nejužívanější funkce mají přiřazeny horké klávesy analogicky prostředí překladačů firmy Borland. Plně je podporováno ovládání pomocí myši.

3.1 Pracovní plocha

Pracovní plocha je základní obrazovka obsahující informace užitečné při práci v integrovaném prostředí. Tato základní obrazovka obsahuje v horní části lištu hlavního



menu spolu se zobrazením aktuálního stavu emulátoru. Dolní lišta obrazovky je tvořena aktivní nápovědou tří nejvýznamnějších horkých kláves, zobrazením jména programu přítomného v emulační paměti a informací o velikosti dostupné paměti v počítači. Zbytek pracovní plochy je určen na umístění oken pro jednotlivé programové nástroje a objekty.

3.1.1 Programové objekty integrovaného prostředí

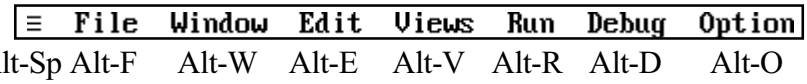
Pojmem programové objekty integrovaného prostředí se označují základní typy zobrazení údajů o stavu systému na obrazovku počítače:

- ***Menu*** - objekt sloužící k vyvolání zvolených příkazů akcí apod.
- ***Okna výpisů*** - objekt ohraničuje jednotlivé výpisy na obrazovce počítače.
- ***Dialogy*** - okna, pomocí nichž je realizováno zadávání hodnot, nastavení systému apod. Od oken výpisů se liší tím, že v daném okamžiku je aktivní (schopné přijímat příkazy) pouze jediné, tj. do okamžiku jeho uzavření jsou ostatní objekty na pracovní ploše neaktivní.
- ***Rolovací lišta*** - jedná se o objekt, kterým jsou obvykle vybavena okna výpisů. Objekt reaguje na myš a umožňuje pohyb výpisu v okně. Používá se především v okamžiku, kdy je počet sloupců či řádků výpisu větší než prostor vymezený oknem výpisu.
- ***Tlačítka*** - tlačítka nebo jejich skupiny se používají na nastavení a volby typu ano/ne, a na potvrzení nebo zrušení vyvolané akce. Tlačítka mohou být bez aretace, tzn., že akce je vyvolána prostým stiskem. Reagují na myš nebo na stisk klávesy zvýrazněného písmena v názvu. Tlačítka s aretací mohou být slučovací nebo vylučovací. Slučovací tlačítka je možné zapnout nebo vypnout nezávisle na sobě. U vylučovacích tlačítek, která se vyskytují vždy v sadě, je možné zapnout pouze jediné v dané sadě.
- ***Řádkové editory*** - jedná se o objekt, pomocí něhož jsou zadávány řetězce znaků nebo číselné konstanty. Editory jsou obvykle vybaveny historií, tj. seznamem, který obsahuje soupis řetězců z předchozích provedených zadání. V řadě případů jsou editory vybaveny i soupisem dostupných jmen, tj. např. seznamem jmen známých proměnných. Tento soupis proměnných se vždy naplní po úspěšném překladu zdrojového textu.
- ***Seznamy*** - seznamy jsou objekty, které sdružují soubor povelů nebo možností volby. Volba se provádí pomocí kurzoru a stisku klávesy *Enter*.

3.1.2 Menu

Všechny položky menu je možné ovládat buď z klávesnice nebo pomocí myši. Pro vstup do menu slouží klávesa *F10*, pohyb v menu se provádí pomocí kurzorových kláves po jednotlivých položkách nebo stiskem zvýrazněného znaku v názvu položky. Položky hlavního menu a některých příkazů mají přiřazeny ještě horké klávesy pro usnadnění výběru položky. Horké klávesy položek lišty hlavního menu jsou konstruovány jednotně v systému zvýrazněné písmeno a klávesa *Alt* (např. File *Alt-F*, Debug *Alt-D*). Volba příkazu menu pomocí myši je složena z vyhledání položky menu kurzorem myši a ze stisku levého tlačítka.

F10 = Menu



Alt-Sp Alt-F Alt-W Alt-E Alt-V Alt-R Alt-D Alt-O

3.1.3 Okna výpisů

Okna výpisů, editorů, atd. se při práci nacházejí buď v aktivním (vybraném) stavu nebo ve stavu neaktivním. Tyto stavy oken jsou rozlišeny typem rámečku (jednoduchý rámeček = neaktivní okno, dvojitý rámeček = aktivní okno).

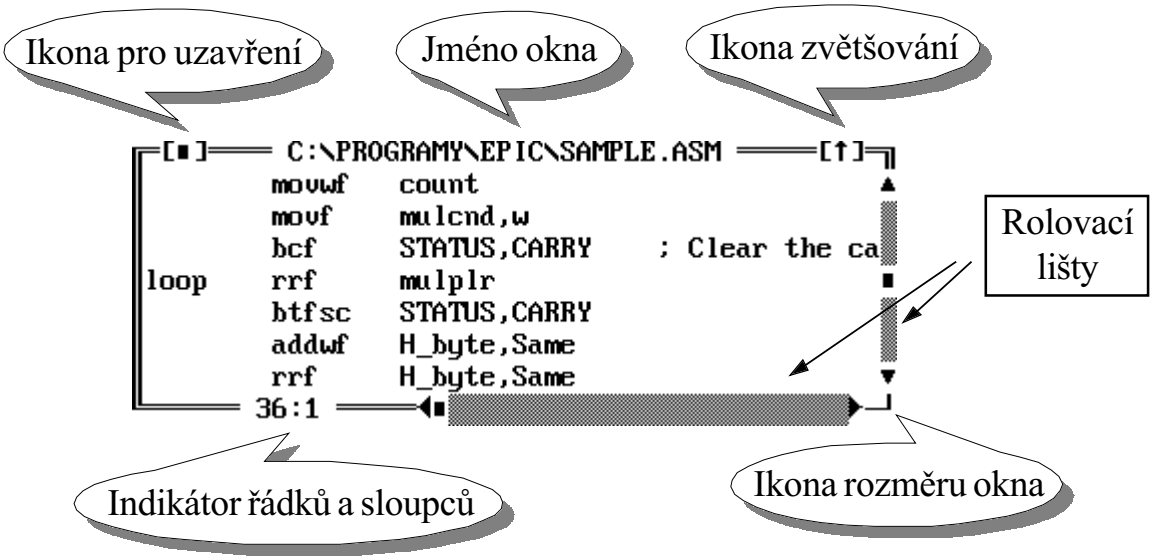
Okno v aktivním stavu přijímá povely z klávesnice a příkazy menu. V neaktivním stavu na tyto akce okno nereaguje. Na pracovní ploše může být aktivní vždy jen jediné okno.

Aktivitu okna je možné volit buď z klávesnice pomocí příkazů "*Window→Next" (F6)* a *"Window→Previous" (Shift-F6)* nebo kursorem myši se současným potvrzením volby aktivity pomocí stisku levého tlačítka.

Mezi další atributy okna patří:

- ***Jméno okna*** - jméno je umístěné uprostřed horní hrany rámečku. Informuje v případě textového editoru o jménu editovaného souboru a v ostatních případech o typu výpisu v okně.
- ***Rolovací lišty*** - umožňují posun textu v okně vodorovným nebo svislým směrem.
- ***Ikona pro uzavření okna*** - reaguje na myš a vyvolá uzavření okna a jeho odstranění z pracovní plochy. Je umístěna na rámečku v levém horním rohu okna. Uzavření okna je možné provést též příkazem *"Window→Close" (Alt-F3)* z hlavního menu.
- ***Ikona rozměru okna*** - je umístěna na pravém dolním rohu rámečku okna. Umožňuje měnit plynule velikost okna a reaguje na myš nebo na příkaz hlavního menu *"Window→Resize" (Ctrl-F5)*.

- ***Ikona zvětšování*** - je umístěna v pravém horním rohu a reaguje na myš nebo na příkaz *"Window→Zoom" (F5)*. Ikona přepíná maximální a nastavený rozměr okna.
- ***Indikátor řádků a sloupců*** - je umístěn v levém dolním rohu okna a ukazuje u textového editoru polohu kurzoru ve formátu [řádek:sloupec].



3.1.3.1. Pohyb oknem po pracovní ploše

Okna po pracovní ploše je možné přesouvat, zvětšovat a nebo zmenšovat.

a) ***Přesunutí okna myší*** - kurzor myši umístíme na rámeček aktivního okna mimo ikonu rozměru okna, uzavření okna a zvětšování okna. Stiskneme levé tlačítko a při jeho držení pohybujeme myší. Okno sleduje pohyb myši. V okamžiku, kdy jsme s jeho polohou spokojeni, uvolníme tlačítko myši a tím okno umístíme.

b) ***Přesunutí okna klávesnicí*** - příkazem *"Window→Resize"* přepneme okno do režimu editace rozměru. Pomocí kursorových kláves vyhledáme požadovanou polohu okna a umístíme ho stiskem klávesy *Enter*.

c) ***Úprava rozměru pomocí myši*** - kurzor myši umístíme na ikonu rozměru okna a stiskneme levé tlačítko. Pohybujeme kursorem myši za současného držení tlačítka. Levý dolní roh okna sleduje pohyb kurzoru, zatímco zbytek okna polohu nemění. Tímto způsobem nastavíme požadovanou velikost okna a okno zafixujeme uvolněním tlačítka myši.

d) ***Úprava rozměrů pomocí klávesnice*** - příkazem hlavního menu *"Window→Resize"* přepneme okno do režimu editace rozměru. Pomocí kombinace klávesy *Shift* s kursorovými klávesami nastavíme požadovanou velikost okna a zafixujeme ji stiskem klávesy *Enter*.

e) ***Zvětšení okna pomocí myši*** - zvětšení okna na maximální rozměr nebo zpět na rozměr původní provedeme umístěním kurzoru myši na ikonu zvětšování a stiskem levého tlačítka.

f) ***Zvětšení okna pomocí klávesnice*** - zvětšení okna na maximální rozměr nebo zpět provedeme stiskem horké klávesy *F5* nebo příkazem *"Window→Zoom"* hlavního menu.

3.1.3.2 Pohyb textem okna

a) ***Pohyb textem okna pomocí myši*** - myší je možné pohybovat textem v okně za pomoci rolovacích lišt. Posunutí textu o jeden znak nebo řádek provedeme umístěním kurzoru myši na ikonu jednotkového posunu a stiskneme levé tlačítko. Text se posune v závislosti na zvolené ikoně a rolovací liště o jeden znak vlevo nebo vpravo (horizontální lišta) resp. o jeden řádek nahoru nebo dolů (vertikální lišta). Podobně je možné posunout text o stránku umístěním kurzoru myši na lištu mezi ikonu jednotkového posunu a kurzor lišty s následným stisknutím levého tlačítka myši. Posun do obecné polohy provedeme umístěním kurzoru myši na kurzor rolovací lišty. Poté současným stiskem levého tlačítka myši a posunem kurzoru myši přemístíme kurzor rolovací lišty na požadovanou polohu. Uvolnění tlačítka myši vyvolá posun textu v okně.

b) ***Pohyb textem okna pomocí klávesnice*** - nejprve přemístíme kurzor ke kraji okna v požadovaném směru pro pohyb textu. Poté stisknutím odpovídající kurzorové klávesy posunujeme text v požadovaném směru.

3.1.4 Dialogová okna

Dialogová okna slouží obvykle k nastavení různých předvoleb integrovaného prostředí. Na pracovní ploše je možné umístit pouze jediné okno tohoto typu, přičemž aktivita ostatních objektů je až do uzavření dialogu potlačena. Uvnitř dialogového okna jsou nastavovací objekty (nejčastěji tlačítka, řádkové editory). Z těchto objektů je vždy aktivní pouze jeden, a tudíž pouze s jedním je možné pracovat. Chceme-li přejít na nastavování parametrů v jiném objektu, stiskneme opakovaně klávesu *Tab,* až je aktivován požadovaný objekt. Druhou možností je umístit kurzor myši na požadovaný objekt a stisknout levé tlačítko myši. Poslední možností je použít horké klávesy pro vybraný objekt. Horká klávesa je konstruována z kombinace zvýrazněného znaku v názvu objektu s klávesou *Alt*. Editace a nastavování parametrů v jednotlivých objektech je popsáno v kapitolách Tlačítka s aretací, Řádkový editor, Seznamy, Objekt historie, Objekt nabídky.

Každé dialogové okno je vybaveno ikonou uzavření, která reaguje na myš, a má význam ignorování změn v nastavení. Dále pak jsou dialogová okna vybavena tlačítky

bez aretace, která slouží v převážné míře k akceptování či zrušení změn v nastavení s následným uzavřením okna po jejich stisku. Jedno z tlačítek dialogového okna je vždy vybráno, a je možné vyvolat jeho funkci z jakéhokoli objektu dialogového okna stiskem klávesy *Enter*.

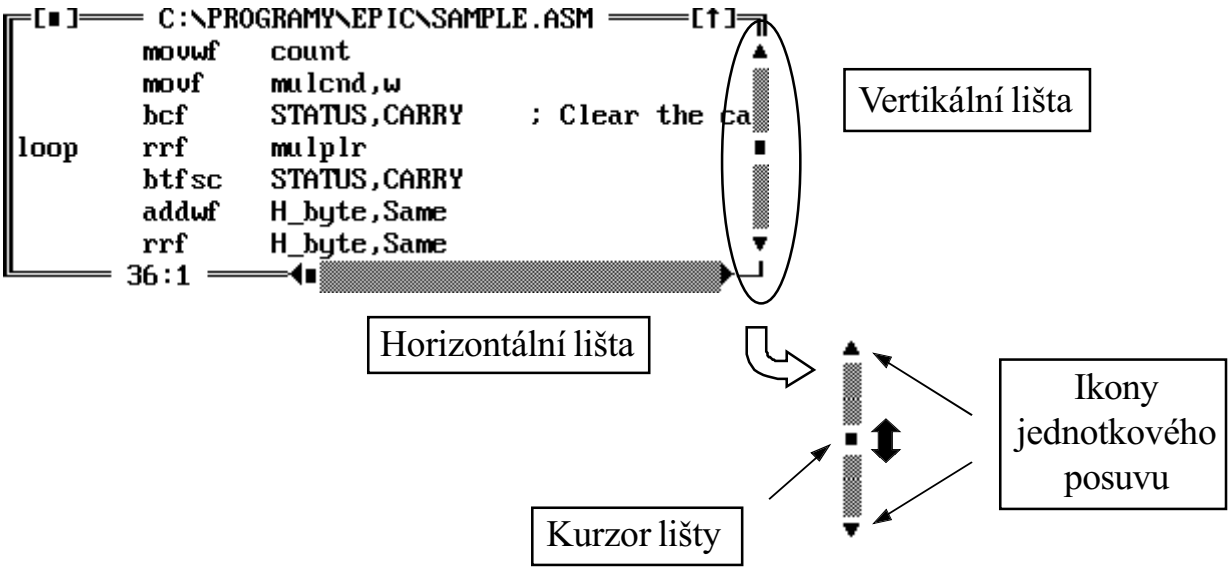
3.1.5 Rolovací lišta

Rolovací lišta je pomocný ovládací objekt převážně pro okna výpisů. Objekt má tři oblasti, které reagují na kurzor myši za současného stisku levého tlačítka. Jedná se o oblasti:

a) ***Ikony jednotkového posunu*** - tyto ikony jsou umístěny na okrajích rolovací lišty a při jejich "stisknutí" se předává povel odpovídajícímu oknu s údajem o směru pohybu s jednotkovým krokem.

b) ***Kurzor lišty*** - kurzor rolovací lišty se může pohybovat v rozmezí ikon jednotkového posunu. Kurzorem myši najedeme na kurzor rolovací lišty. Poté stikneme levé tlačítko myši a současně pohybujeme kurzorem myši v požadovaném směru. Kurzor rolovací lišty následuje kurzor myši. Po uvolnění tlačítka se předá povel s údajem o pozici kurzoru rolovací lišty a podle hodnoty se provede posun textu na požadovaný řádek nebo sloupec.

c) ***Prostor na liště*** - prostor na liště mezi ikonami a kurzorem rolovací lišty reaguje na myš vysláním povelu na odstránkování textu příslušného okna o stránku vpřed či vzad.



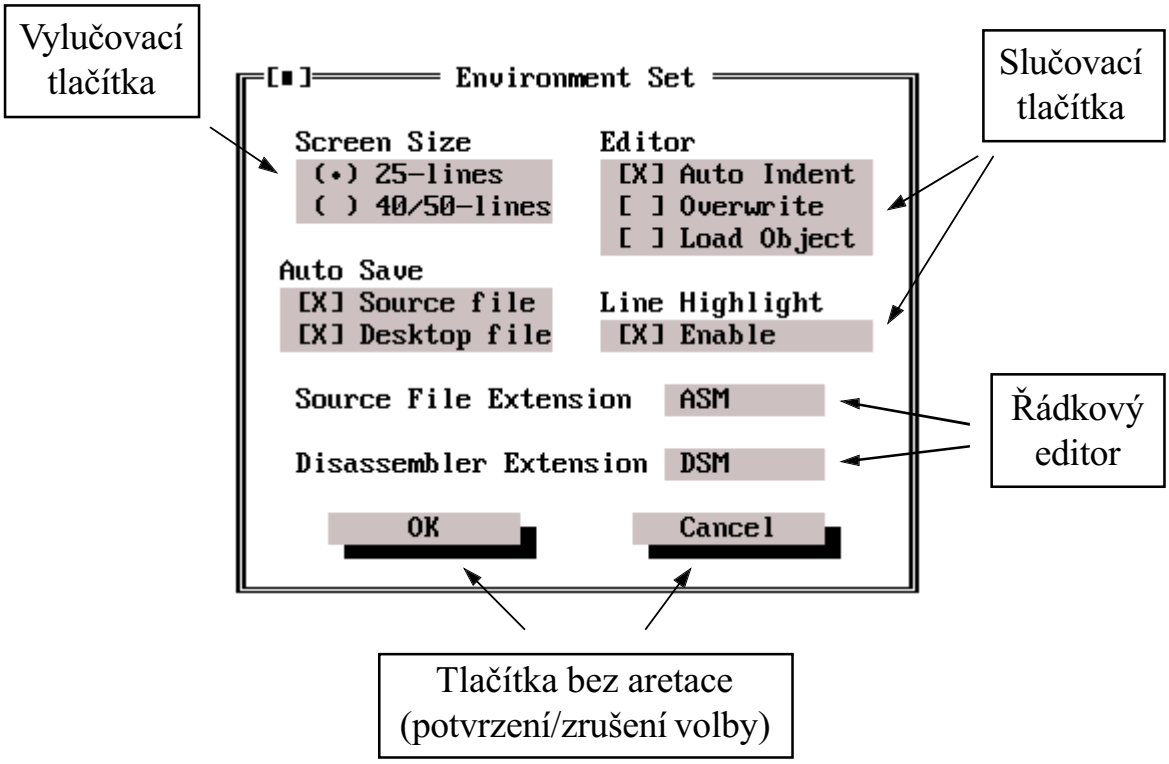
3.1.6 Tlačítka

Tlačítka mohou být třech typů:

a) ***Tlačítko bez aretace*** - tlačítko bez aretace je objekt, který slouží k potvrzování či rušení volby. Reaguje na ovládání myší, tj. na kurzor myši a stisk levého tlačítka myši. Z klávesnice se tlačítka ovládají pomocí stisku klávesy odpovídající zvýrazněnému znaku ve jménu tlačítka.

b) ***Slučovací tlačítka*** - slučovací tlačítka jsou obvykle sdružena po skupinách do jediného objektu. Jsou označena jménem se zvýrazněným znakem, který odpovídá horké klávese konstruované z klávesy *Alt* a zvýrazněného znaku. Pro pohyb kurzoru ve skupině slučovacích tlačítek jsou určeny kurzorové klávesy. Zapnutí resp. vypnutí tlačítka provedeme stiskem mezerníku na vybraném tlačítku. Myší se tlačítka zapínají resp. vypínají tak, že umístíme kurzor myši na ikonu tlačítka ([ ] resp. [X]) a stiskneme levé tlačítko myši.

c) ***Vylučovací tlačítka (radio tlačítka)*** - vylučovací tlačítka jsou uspořádána do skupin, a od slučovacích tlačítek se liší tím, že může být zapnuto vždy pouze jediné tlačítko ve skupině. Ovládání zapínání tlačítka pomocí klávesnice a myši je stejné jako ovládání tlačítek slučovacích.



3.1.7 Řádkové editory

Pomocí řádkových editorů zadáváme číselný nebo textový údaj. Řádkové editory jsou ve většině případů vybaveny objektem historie a objektem nabídky dostupných symbolů.

Editaci textu provádíme pomocí standardních kláves editorů:

- ***kurzorové klávesy*** pro pohyb kurzoru
- ***Ins*** pro přepínání režimu vkládání a přepisování textu
- ***Del*** pro mazání znaku
- ***Home*** pro přesunutí kurzoru na začátek textu
- ***End*** pro přesunutí kurzoru na konec textu

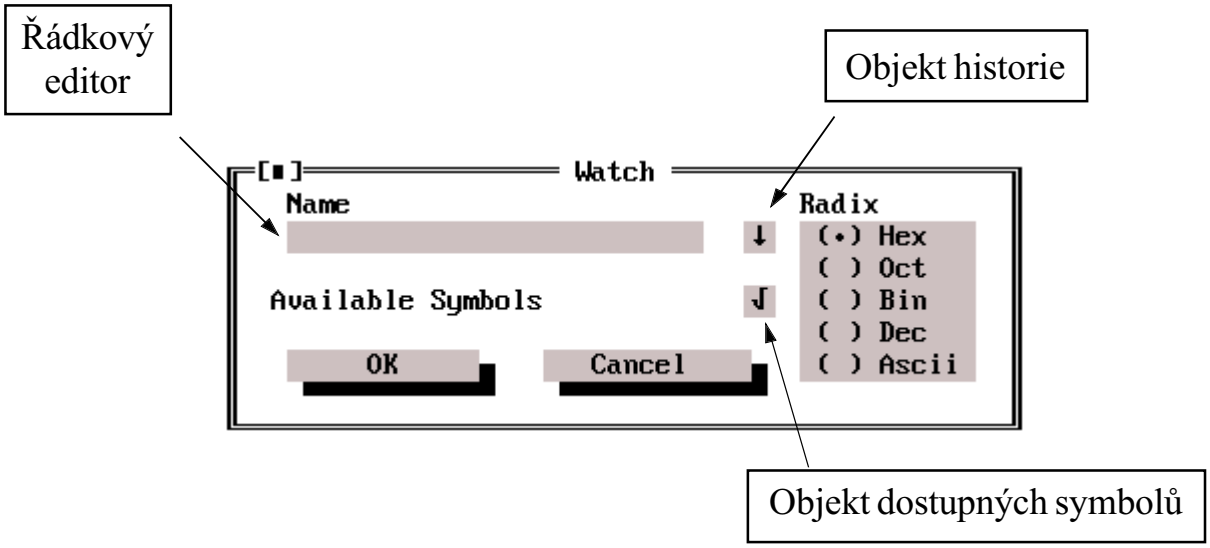
Pro vyvolání podpůrných programových objektů používáme:

- ***šipka dolů*** pro vyvolání objektu historie
- ***Ctrl-Enter*** pro vyvolání objektu dostupných symbolů

Podpůrné programové objekty jsou tvořeny seznamy, jejichž ovládání je popsáno v kapitole 3.1.8 a 3.1.9.

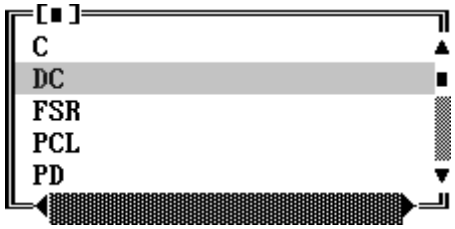
Pomocí objektu historie, ve kterém jsou uloženy všechny předcházející editované řetězce znaků, můžeme snáze a rychleji zadávat již jednou zadané řetězce.

V objektu dostupných symbolů jsou pro usnadnění zadávání k dispozici všechna aktuálně známá jména a symboly generované při úspěšném překladu zdrojového textu.



3.1.8 Seznamy

Seznamy jsou objekty obsahující soupisy jmen a symbolů, přičemž s těmito soupisy pracují jako s databází. Kurzorem je možné vybrat zvolenou položku a pomocí klávesy *Enter* ji potvrdíme. Položku můžeme taktéž vybrat pomocí myši. Umístíme na ni kurzor myši a stiskneme levé tlačítko. Je-li v seznamu více položek než je možné v odpovídajícím prostoru zobrazit, pak je seznam doplněn rolovací lištou. Pomocí rolovací lišty pak můžeme listovat seznamem přes všechny jeho položky.



3.1.9 Podpůrné programové objekty

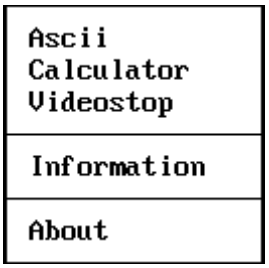
Podpůrné programové objekty jsou určeny pro usnadnění zadávání jmen a symbolů popřípadě matematických výrazů. Integrované prostředí nabízí tyto podpůrné programové objekty.

- ***Historie*** [↓] - objekt historie souvisí vždy s řádkovým editorem. Na pracovní ploše dialogového okna má vyhrazenu ikonu, která reaguje na klávesnici nebo myš pouze v okamžiku, je-li aktivován jí příslušející editor. Stiskneme-li klávesu šipka dolů (kurzorová klávesa), pak se ikona historie rozvine v seznam, který obsahuje všechny předešlé textové řetězce zadávané editorem do prostředí. Po stisku klávesy *Enter* je vložen vybraný řetězec do editoru a seznam je zpět nahrazen ikonou. Uzavření seznamu bez vložení textu provedeme klávesou *Esc*.
- ***Dostupné symboly*** [√] - objekt dostupných symbolů je seznam obdobný seznamu historie. Má stejné ovládání a účinky. Od seznamu historie se liší tím, že je plněn po úspěšném překladu překladačem. Pod pojmem dostupné symboly rozumíme jména, která lze v zadaném výrazu vyhodnotit.

3.2 Hlavní menu

Hlavní menu integrovaného prostředí tvoří ovládací lišta v horní části pracovní plochy. Základní nabídku tvoří položky ≡, File, Window, Edit, Views, Run, Debug a Options.

3.2.1 Nabídka ≡



Nabídka označená znakem ≡ obsahuje pomocné nástroje a informace, které pomáhají při vývoji aplikací. Nabídka je tvořena položkami:

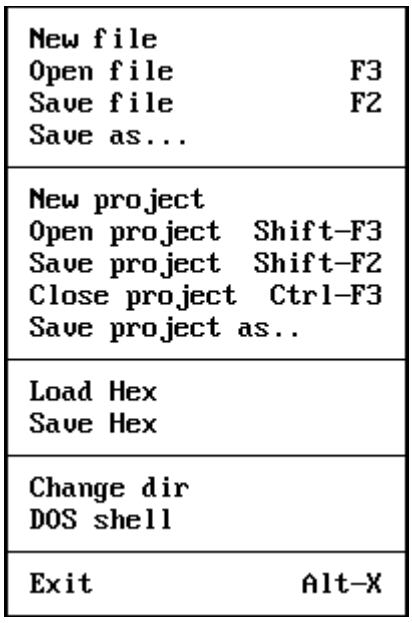
- ***Ascii*** - zobrazení rozšířené tabulky ASCII znaků, včetně převodů na číselné kódy.
- ***Calculator*** - vyvolání programového modulu realizujícího kalkulátor v pevné řádové čárce, základními operacemi včetně převodů čísel mezi jednotlivými číselnými soustavami.
- ***Videostop*** - jednoduchá hra pro chvíle úplného vyčerpání
- ***Information*** - zobrazení základních informací o verzi integrovaného prostředí, překladače a emulátoru.
- ***About*** - zobrazení jmen a informací o autorech programu

3.2.2 Nabídka File

Nabídka File je souhrnem příkazů pro práci se soubory, projekty, adresáři atp. Obsahuje následující položky rozdělené do několika skupin.

Skupina pro práci se soubory:

- ***New File*** - umožní otevřít a pomocí editoru editovat nový soubor
- ***Open File*** (*F3*) - umožní otevřít již existující, popř. založit nový soubor zdrojového textu.
- ***Save File*** (*F2*) - příkaz k uložení obsahu libovolného objektu na pracovní ploše. Objektem je míněn



textový soubor nebo obsah libovolného programového nástroje pro editaci paměti emulačního obvodu.

- ***Save as*** - příkaz slouží k uložení textu zdrojového souboru pod novým zadaným názvem.

Skupina pro práci s projektem:

Tato skupina je určena k ovládání manažeru projektů. Manažer projektů umožňuje efektivně pracovat současně na více projektech tím, že zabezpečuje uložení nastavení rozpracovaného problému při přerušení práce. Při opětovném otevření projektu je prostředí z aktuálního stavu přenastaveno podle definic uložených v projektu.

- ***New Project*** - příkaz umožňuje otevřít zcela nový projekt s tím, že prostředí je nastaveno na implicitní hodnoty.
- ***Open Project*** (*Shift-F3*) - otevření již exitujícího projektu
- ***Save Project*** (*Shift-F2*) - uložení nastavení projektu do definičního souboru projektu
- ***Close Project*** (*Ctrl-F3*) - ukončení práce v aktuálně otevřeném projektu
- ***Save Project as*** - uložení nastavení projektu podle zadaného jména

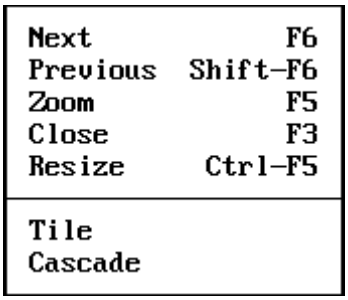
Skupina vstupů a výstupů:

- ***Load Hex*** - inicializace emulační paměti obsahem souboru ve formátu Intel HEX-16.
- ***Save Hex*** - uložení obsahu emulační paměti ve formátu Intel Hex-16.

Ostatní příkazy:

- ***Change Dir*** - změna pracovního adresáře
- ***DOS Shell*** - spuštění programové řádky DOSu. Vzhledem k poměrně značnému využití konvenční paměti není obvykle možné spustit z příkazové řádky další program.
- ***Exit*** *(Alt-X)* - ukončení práce a opuštění programu.

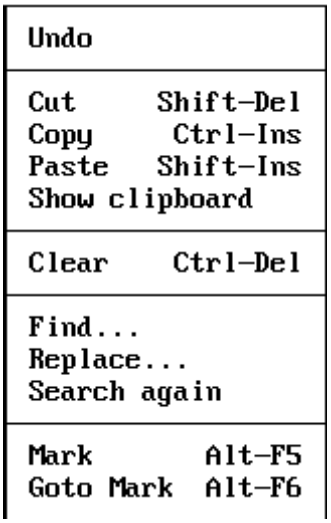
3.2.3 Nabídka Window



Nabídku Window tvoří příkazy pro práci s okny jednotlivých programových nástrojů pro editaci emulační paměti a pro textový editor zdrojového textu. Nabídka obsahuje následující příkazy:

- ***Next*** (*F6*) - přepnutí z aktivního okna na následující. Aktivní okno je zvýrazněno dvojitou čarou rámečku
- ***Previous*** (*Shift-F6*) - přepnutí z aktivního okna na předcházející okno v řadě.
- ***Zoom*** (*F5*) - přepnutí aktivního okna z aktuální velikosti na velikost maximální a zpět.
- ***Close*** (*Alt-F3*) - uzavření aktivního okna.
- ***Resize*** (*Ctrl-F5*) - přepnutí okna do režimu zvětšování a zmenšování, případně pohybu oknem po pracovní ploše.
- ***Tile*** - uspořádání oken po ploše obrazovky tak, aby byla všechna otevřená okna viditelná. Tento režim je vhodný při poměrně malém počtu otevřených oken.
- ***Cascade*** - uspořádání oken za sebe. Uspořádání připomíná jednotlivé lístky kartotéky.

3.2.4 Nabídka Edit

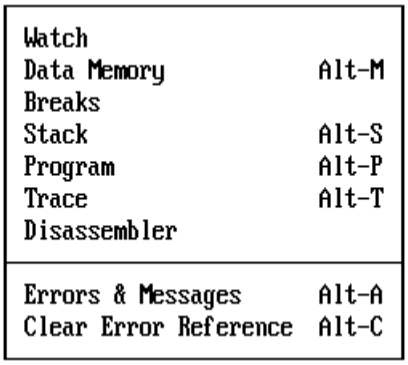


Nabídka Edit zahrnuje příkazy a nástroje pro textový editor. Nabídka se skládá z těchto povelů:

- ***Undo*** - vrácení poslední úpravy textu např. vymazání slova či řádku.
- ***Cut*** (*Shift-Del*) - vystřižení bloku označeného textu (před vystřižením je označený text zkopírován do pomocného editoru)
- ***Copy*** (*Ctrl-Ins*) - zkopírování označeného textu do pomocného editoru (Clipboard)
- ***Paste*** (*Shift-Ins*) - přenesení textu z pomocného editoru do editovaného textu.

- ***Show Clipboard*** - zobrazení obsahu pomocného editoru.
- ***Clear*** (*Ctrl-Del*) - vymazání označeného textu bez náhrady
- ***Find*** - vyhledání zadaného řetězce znaků (slova)
- ***Replace*** - vyhledání řetězce znaků a jeho nahrazení řetězcem zadaným.
- ***Search again*** - opětovné vyhledání zadaného řetězce v textu.
- ***Mark*** - umístění značky na řádek textu, který odpovídá pozici kurzoru. Funkce slouží k označení místa textu, na které se chceme vrátit pomocí příkazu "Goto Mark"
- ***Goto Mark*** - návrat na řádek označený značkou. Řádek označíme pomocí příkazu "Mark"

3.2.5 Nabídka Views

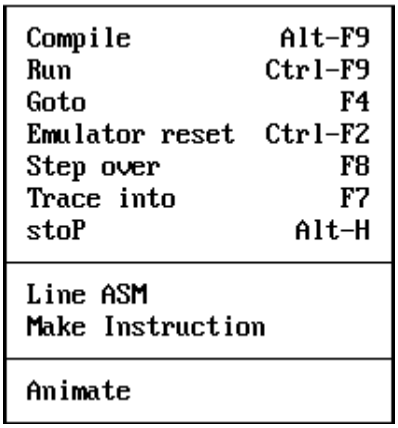


Nabídka Views sdružuje příkazy na spouštění jednotlivých nástrojů pro editaci emulační paměti. K dispozici jsou příkazy:

- ***Watch*** - spuštění zobrazování a editace programových proměnných podle uživatelské volby.
- ***Data Memory*** (*Alt-M*) - zobrazení celého obsahu datové paměti (registrové) banky emulovaného mikropočítače.
- ***Breaks*** - zobrazení databáze událostí. Mezi události jsou zahrnuty všechny typy breaků (bod zastavení běhu programu) a události podmíněného trasování.
- ***Stack*** (*Alt-S*) - zobrazení obsahu zásobníku se zvýrazněním ukazatele zásobníku (Stack Pointer).
- ***Program*** (*Alt-P*) - zobrazení obsahu programové paměti.
- ***Trace*** (*Alt-T*) - zobrazení obsahu trasovací paměti
- ***Disassembler*** - zobrazení obsahu programové paměti včetně disasemblování do mnemoniky instrukčního souboru.

- ***Errors & Messages*** (*Alt-A*) - zobrazení okna chyb a varování vzniklých při překladu.
- ***Clear Error Reference*** (*Alt-C*) - potlačení barevného podsvícení chybné řádky ve zdrojovém textu.

3.2.6 Nabídka Run



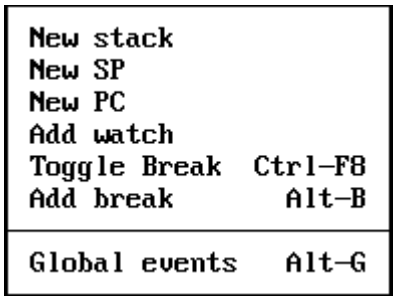
Nabídka Run je určena pro příkazy související se zpracováním zdrojového textu překladačem a s různými druhy spouštění a běhu programu aplikace.

Nabídka zahrnuje tyto povely:

- ***Compile*** (*Alt-F9*) - překlad zdrojového textu programu aplikace
- ***Run*** (*Ctrl-F9*) - spuštění programu v emulační paměti. Spuštění programu není blokováno ani při neúspěšném překladu z důvodu chyb. V takovém případě bude prováděn buď program podle posledního úspěšného překladu a nebo program podle náhodného obsahu programové paměti.
- ***Goto*** (*F4*) - příkaz provede spuštění programu k pozici kurzoru ve zdrojovém textu
- ***Emulator reset*** (*Ctrl-F2*) - provede se reset emulátoru tj. emulovaný procesor se nastaví shodně s katalogovou specifikací stavu reset.
- ***Step Over*** (*F8*) - program se spustí podle aktuálního obsahu programové paměti na jeden krok, přičemž jeden krok může odpovídat buď jedné instrukci, podprogramu nebo makru. V tomto případě se tedy jedná o krokování programem s tím, že makra a podprogramy jsou prováděny plnou rychlostí.
- ***Trace Into*** (*F7*) - krokování programu po instrukcích
- ***Line Asm*** - vyvolání řádkového překladače. Nejčastěji se tento povel uplatní při drobných korekturách programu přímo v programové paměti. Pozor - příkaz sice změní obsah programové paměti, ale tato změna není promítnuta do zdrojového textu. Dochází tak k nekonzistenci zdrojového textu a programu.

- ***Make instruction*** - příkaz umožňuje vykonat instrukci tzv. mimo pořadí tj. nezávisle na obsahu programové paměti.
- ***Animate*** - příkazem se uvádí emulátor do režimu animace, tj. režimu, kdy je aplikační program automaticky krokován, a po každém kroku je kompletně vyčten obsah datové paměti a stav ostatních vnitřních registrů emulovaného procesoru.

3.2.7 Nabídka Debug

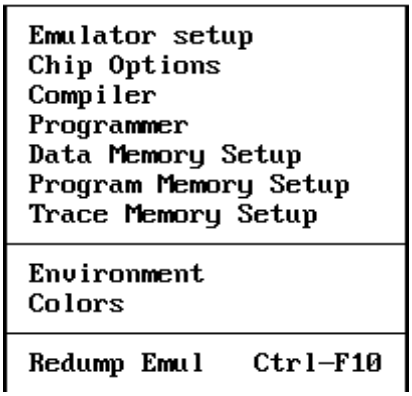


V nabídce Debug jsou zahrnuty všechny dostupné příkazy pro modul debuggeru. Jedná se o modul umožňující zadávat a modifikovat interní registry emulovaného mikroprocesoru, které jsou programově nepřístupné. Jsou k dispozici tyto příkazy:

- ***New Stack*** - příkazem je možné editovat obsah zásobníku.
- ***New SP*** - nastavení nové hodnoty ukazatele zásobníku.
- ***New PC*** - nastavení nové hodnoty programového čítače.
- ***Add Watch*** - příkazem se přidá položka do uživatelského formátu zobrazení (Watch).
- ***Toggle Break*** (*Ctrl-F8*) - zadání prostého adresového breaku (bod zastavení běhu programu na zadané adrese).
- ***Add Break*** (*Alt-B*) - zadání nového breaku nebo události pomocí editoru událostí.
- ***Global Events*** (*Alt-G*) - nastavení a konfigurace pro všeobecné události, při nichž má dojít k zastavení běhu programu.

3.2.8 Nabídka Option

Nabídka Option sdružuje všechna uživatelská nastavení, a to jak prostředí, tak emulátoru. K dispozici jsou příkazy:



- ***Emulator Setup*** - nastavení parametrů emulátoru, např. hodinový kmitočet, typ procesoru apod.
- ***Compiler*** - nastavení parametrů překladače, formátu výpisu a formátu výstupního souboru překladače.
- ***Chip Options*** - nastavení specifických parametrů emulovaného procesoru. Jedná se o možnost nastavení např. módu oscilátoru, povolení "start up timeru", povolení "wakeup timeru", povolení obvodu "watchdog" apod. Toto nastavení se odlišuje pro každý typ procesoru (popř. rodinu procesorů). Ve značné míře odpovídají položky nastavení konfiguračnímu bajtu procesoru (nastavení pojistek).
- ***Data Memory Setup*** - nastavení parametrů výpisu datové paměti emulovaného procesoru
- ***Program Memory Setup*** - nastavení parametrů výpisu programové paměti emulovaného procesoru
- ***Trace Memory Setup*** - nastavení parametrů výpisu trasovací paměti emulovaného procesoru
- ***Environment*** - uživatelské nastavení prostředí
- ***Colors*** - nastavení barev jednotlivých objektů a výpisů na obrazovku.
- ***Redump Emul*** - příkazem je znovu vyčten úplný obsah emulační paměti a stavu emulátoru.

3.3 Zdrojový text, projekt a objekt

Vývoj aplikace s pomocí integrovaného prostředí lze realizovat několika cestami. Pro jednoduché pokusy je možné využít přímo překladu zdrojového textu programu. Pro složitější aplikace nebo při vývoji programů, u nichž je nezbytné udržet konzistenci s dalšími prvky vývoje, např. s plošným spojem a přiřazením vývodů procesoru apod., doporučujeme použít možnosti konstrukce projektu.

3.3.1 Organizace adresářů

Při vývoji aplikací je nutné mít na paměti systém uložení informací o zpracovávaném programu na pevném disku počítače. Integrované prostředí rozpoznává v zásadě vždy dva adresáře, které můžeme označit jako adresář instalace a adresář

pracovní. Pojmem adresář instalace máme na mysli adresář obsahující nezbytně nutné soubory pro instalaci a spuštění prostředí. Tento adresář je sice možné používat jako pracovní, ale z důvodu bezpečnosti uložených dat to nelze doporučit. Navíc při instalaci na počítačové síti je obvyklé omezení přístupových práv pro běžné uživatele. Instalační adresář může mít nastaven atribut read-only (integrované prostředí využívá adresář pouze ke čtení dat). Pojmem pracovní adresář označujeme libovolný adresář, do něhož je možné zapisovat a z něhož je možné číst data. V tomto adresáři vytváří systém vždy standardní konfigurační projekt, tj. soubor označený DPIC.PRJ a soubor palety barev DPIC.CFG. Soubor standardního projektu obsahuje informace o nastavení prostředí při posledním spuštění, v němž nebyl otevřen žádný uživatelský projekt. Soubor palety barev obsahuje uživatelské kódy barev pro všechny projekty v daném pracovním adresáři.

3.3.2 Zdrojový text a projekt

Zdrojový text programu vytváříme textovým editorem. Soubor je možné otevřít a uložit pomocí nabídky File v hlavním menu. V případě, že píšeme zdrojový text bez otevření projektu, může být soubor nazván libovolným platným názvem pro operační systém MS DOS. Je-li však zdrojový text psán v otevřeném projektu, pak hlavní soubor zdrojového textu tohoto projektu musí mít s projektem shodný název. Po uzavření projektu je vytvořen v pracovním adresáři konfigurační soubor projektu "jméno.PRJ". Podle jména projektu je pak hledán na pevném disku hlavní soubor projektu se zdrojovým textem a současně, pokud existuje, i příslušný soubor objektu.

3.3.3 Soubor objektu

Soubor objektu je vytvořen překladačem na základě úspěšného ukončení překladu. Soubor obsahuje kromě operačních kódů instrukcí další důležité informace, např. o umístění proměnných v datové paměti nebo o umístění programových návěští v paměti programu. V případě, že je otevřen projekt, má soubor objektu přiřazeno shodné jmého se jménem hlavního zdrojového textu, a tudíž i se jménem projektu.

3.3.4 Projekt

Vzhledem k tomu, že prostředí umožňuje mít na pracovní ploše otevřeno více textových souborů se zdrojovým ale i obecným textem, je nezbytné řešit prioritu překladu zdrojového textu. V prostředí platí: je-li otevřen projekt, volá se vždy, a to nezávisle na aktivitě toho či onoho okna se zdrojovým textem, překlad hlavního souboru "Projektu". V případě, že projekt otevřen není, je přeložen zdrojový text právě aktivního editoru. Pro manipulaci s projekty slouží příkazy z nabídky File hlavního menu.

3.4 Editor zdrojového textu

Editor zdrojového textu umožňuje zpracovávat zdrojový text do velikosti 64 KByte. V některých případech, a to převážně při bohatém komentáři zdrojového textu, může dojít k vyčerpání tohoto rozsahu. Aby uživatel nebyl omezován, může uvedenou situaci řešit pomocí direktivy vložení souboru. Tato direktiva umožní zahrnout do překladu více souborů zdrojového textu a tím umožní i rozšíření paměťových možností pro editaci. U rozsáhlejších aplikací doporučujeme rozdělení zdrojového souboru do více částí nejen z důvodu omezení paměťového prostoru editoru, ale i vzhledem k přehlednosti textu.

3.4.1 Editace textu

- ***přesunutí kurzoru*** - pohyb kurzoru po aktivní ploše editoru je prováděn pomocí kurzorových šipek po jednotlivých znacích, pomocí kombinace kláves *Ctrl→* a *Ctrl←* po jednotlivých slovech. Kurzor je možné přemístit i pomocí myši umístěním kurzoru myši na požadovaný znak, a stisknutím levého tlačítka
- ***označování bloku textu*** - označení bloku textu je možné buď pomocí kombinace kláves *Ctrl-KB* pro začátek a *Ctrl-KK* pro konec bloku nebo pohybem myši za současného držení levého tlačítka.
- ***zrušení označení bloku*** - označení bloku zrušíme buď pohybem kurzoru po ukončení označení bloku a nebo pomocí kombinace kláves *Ctrl-H*.
- ***přesunutí a kopie textu*** - pro přesun a kopii textu je využit skrytý editor (clipboard). Zvolený text označíme a pomocí menu nebo horkých kláves zkopírujeme do skrytého editoru. K přesunu textu volíme vystřižení, tj. příkaz *"Edit→Cut"* (*Shift-Del*), ke zkopírování volíme příkaz *"Edit→Copy" (Ctrl-Ins)*. Poté přesuneme kurzor na pozici nového vložení textu a volíme příkaz *"Edit→Paste"* (*Shift-Ins*).
- ***vymazání textu, bloku*** - pro vymazání textu je k dispozici klávesa *Del* (vymaže písmeno na pozici kurzoru), klávesa *Back-Space* (vymaže písmeno před pozicí kurzoru) a příkaz *"Edit→Clear"* (*Ctrl-Del*), kterým je možné vymazat celý označený text.
- ***zrušení vymazání*** - pro zrušení posledního vymazání textu a jeho navrácení do původní podoby použijeme příkaz *"Edit→Undo"*.
- ***zobrazení skrytého editoru*** - zobrazení obsahu skrytého editoru vyvoláme příkazem *"Edit→Show Clipboard"*.

- ***funkce hledání řetězce v textu (příkaz "Edit→Find")*** - editor umožňuje vyhledávat sekvence znaků bez nebo s rozlišením malých a velkých písmen, s možností zvolit hledání pouze celých slov.
- ***funkce nahrazení textu*** - příkazem *"Edit→Replace"* vyvoláme funkci vyhledávání textu a nahrazování textem jiným. Příkazem je možné vyhledávat bez nebo s rozlišením malých a velkých písmen, vyhledávání omezit pouze celými slovy a vyhledat první nebo všechny výskyty zadaného textového řetězce. Současně je možné zvolit automatické nahrazení nebo nahrazení s potvrzením od uživatele.

3.4.2 Další pomocné funkce editoru

- ***funkce automatické volby začátku řádku (Autoindent)*** - volbou této funkce v nastavení prostředí uvedeme editor do stavu, ve kterém při přechodu na další řádek nastavuje pozici kurzoru podle počátku řádku předchozího. Není-li funkce nastavena, pak je kurzor po přechodu na další řádek umístěn vždy na první sloupec textové obrazovky.
- ***funkce zvýraznění syntaxe (Highlight)*** - funkce se volí globálně pro všechny objekty integrovaného prostředí v uživatelském nastavení, a tudíž na ni reaguje i editor. Tato funkce barevně odlišuje klíčová slova od ostatního zdrojového textu.
- ***funkce zobrazení čítače instrukcí*** - při emulaci provádí mikropočítač aplikační program podle zadaných instrukcí, přičemž se mění hodnota programového čítače. Editor průběžně zobrazuje hodnotu programového čítače pomocí barevného podsvícení řádku se zápisem instrukce, jejíž umístění v programové paměti odpovídá aktuální hodnotě programového čítače.
- ***funkce pro zobrazení bodů zastavení a bodů trasování*** - tyto funkce umožňují zadat a zobrazit adresový bod zastavení (Address Break) a zobrazit bod trasování. Bod trasování je zobrazen zvýrazněním prvního znaku na řádce, bod zastavení pak zvýrazněnou barvou řádky. Pokud je prvním znakem řádky mezera, je bod trasování označen znakem ±
- ***funkce pevných odkazů na chyby a body zastavení programu*** - jedná se o pomocnou funkci editoru. Funkce umožnuje editaci zdrojového textu, v němž jsou vyznačeny chyby překladačem tak, že při případném vypuštění řádku, či jeho vložení, sleduje editor nové relativní polohy tohoto označení chyby. Taktéž je tato funkce využita pro posuvy poloh značek pro body zastavení a body podmíněného trasování. Detailně jsou tyto funkce popsány v kapitole 3.19.

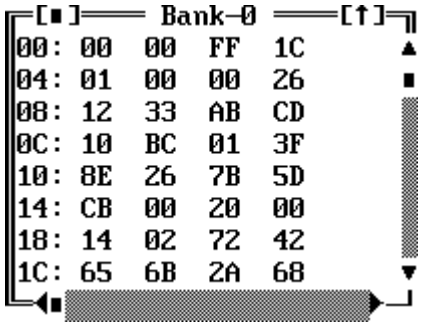
3.5. Editor datové a programové paměti

Editory datové a programové paměti jsou nástroje na přímou editaci paměťového prostoru emulovaného mikroprocesoru. Jedná se o maticově orientované výpisy. První sloupec výpisu obsahuje vždy adresu prvního bajtu či slova na příslušném řádku zobrazení. Dále jsou uvedeny hodnoty následujících paměťových buněk. Např.

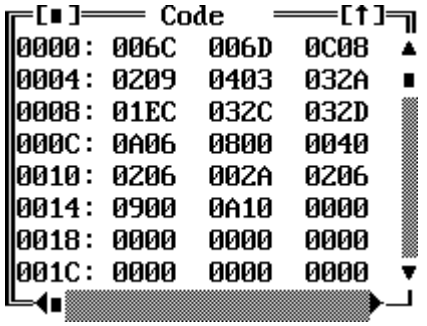
08: 12 33 AB CD
0C: 15 1F E0 A1

V uvedeném příkladu je tedy na adrese 8 hodnota 12, na adrese 9 hodnota 33 atd.

Editor datové paměti



Editor programové paměti



Adresy jsou vypisovány vždy v hexadecimální soustavě, zatímco pro výpis obsahu paměti je možné volit jeden z pěti formátů výpisu a to v soustavě hexadecimální, dekadické, binární, oktalové a pomocí ASCII znaků. Dále je možné zvolit počet sloupců výpisu od 1 do 6 a nastavit pro ně počet vypisovaných znaků. Všechny tyto volby je možné provést pomocí příkazů *"Option→Data Memory Setup"* resp. *"Option→Program Memory Setup"* v nastavení formátu výpisu paměťového prostoru na obrazovku.

- ***pohyb kurzorem*** - editory obsahu paměti zobrazují kurzory dva. První z nich je shodný s kurzorem textového editoru a je možné s ním pohybovat po ploše okna editoru pomocí kurzorových šipek. Druhý kurzor ve tvaru bloku se objevuje pouze na pozicí, kde je možné editovat obsah paměti. Přesun kurzoru je možné provést též pomocí myši umístěním kurzoru myši na vybranou pozici a stiskem levého tlačítka.
- ***pohyb blokového kurzoru*** - blokový kurzor je možné přesouvat vždy na následující nebo předcházející položku paměti. Není-li kurzor zobrazen, pak se přesune na nejbližší následující či předcházející položku vzhledem ke kurzoru standardnímu. Přemístění blokového kurzoru na následující položku je možné pomocí klávesy *Tab* nebo stiskem kombinace kláves *Ctrl→*. Předcházející položku

vyhledáme pomocí kombinace kláves *Ctrl*←. Na první položku editoru paměti přemístíme kurzor pomocí kláves *Ctrl-PgUp*, na poslední pomocí kombinace *Ctrl-PgDn*. První položku na řádku zvolíme klávesou *Home*, poslední klávesou *End*.

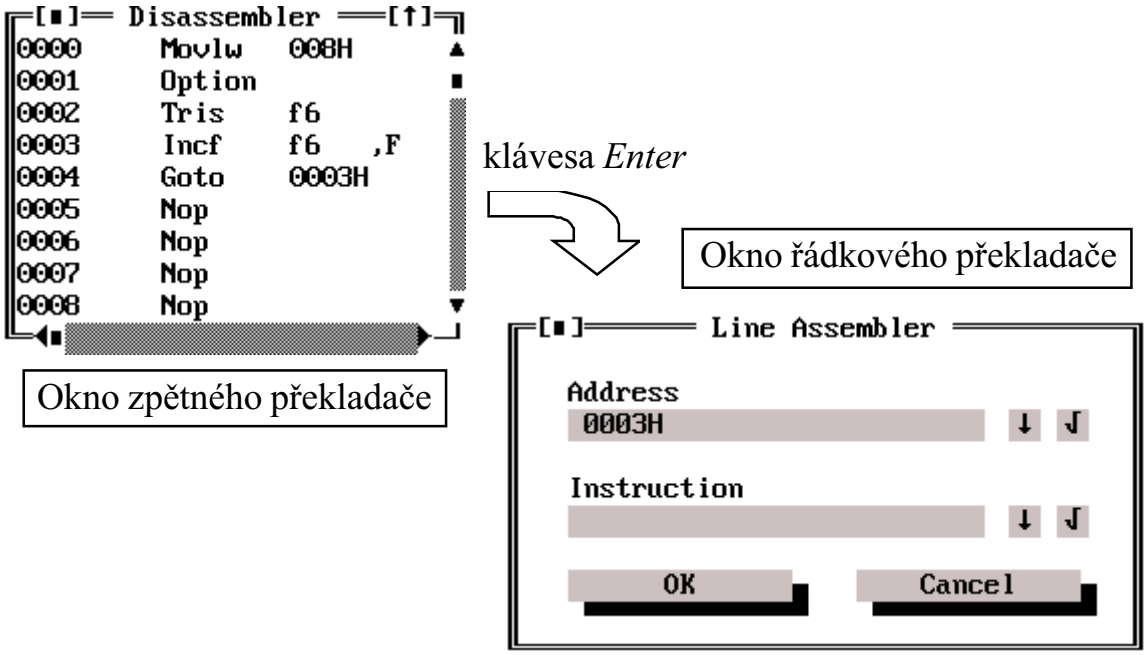
- ***funkce pro body zastavení a body trasování*** - tyto funkce umožňují zadat a zobrazit adresový bod zastavení (Address Break) a zobrazit bod trasování. Bod trasování je zobrazen zvýrazněním prvního znaku položky, bod zastavení zvýrazněnou barvou položky.
- ***editace paměťového místa a funkce automatického posunu*** - editor umožňuje editovat zvolenou položku pouze pomocí platných znaků ve zvolené číselné soustavě zobrazení, tj. v binární soustavě jsou povoleny znaky 1 a 0, v dekadické od 0 do 9 atd. Funkce automatického posunu kurzoru spočívá v tom, že po editaci posledního znaku položky přejde kurzor automaticky na položku následující.

3.6 Zpětný překladač (Disassembler)

Zpětný překladač je určen pro zobrazení obsahu programové paměti emulovaného mikropočítače včetně jeho zpětného převodu do mnemoniky instrukcí. Výpis je uspořádán do dvou sloupců, přičemž v prvním sloupci je hexadecimálně uvedena adresa paměti a v druhém sloupci je uvedena instrukce, kterou reprezentuje příslušný operační kód uložený v paměti.

Ovládání zpětného překladače je následující:

- ***pohyb kurzoru*** - pohyb kurzoru v okně zpětného překladače je umožněn pomocí kurzorových kláves nebo myši zcela analogicky k editoru zdrojového textu či editoru programové nebo datové paměti.
- ***editace a překlad*** - pomocí zpětného překladače je možné editovat programovou paměť pomocí mnemoniky instrukcí. Při editaci umístíme kurzor na požadovanou adresu a stiskneme klávesu *Enter*. Po stisku klávesy se otevře dialogové okno řádkového překladače. Pomocí dvou polí zadáme adresu (případně ponecháme adresu beze změny) a zapíšeme instrukci platnou mnemonikou. Stiskneme tlačítko OK v dialogovém okně a pokud bylo možné zápis vyhodnotit a přeložit, pak je obsah programové paměti modifikován. Případná chyba v zápisu je oznámena pomocí chybového hlášení do dialogového okna chyb a obsah paměti se nezmění.
- ***funkce zobrazení bodů zastavení a trasování*** - body zastavení jsou překladačem zobrazovány pomocí zvýrazněného řádku. Body trasování jsou označovány znakem ².
- ***funkce zobrazení čítače instrukcí*** - v průběhu emulace prochází mikrokontrolér aplikační program podle zadaných instrukcí, přičemž se mění hodnota programového čítače. Zpětný překladač zobrazuje hodnotu programového čítače



pomocí barevného podsvícení instrukce, jejíž umístění v programové paměti odpovídá aktuální hodnotě programového čítače.

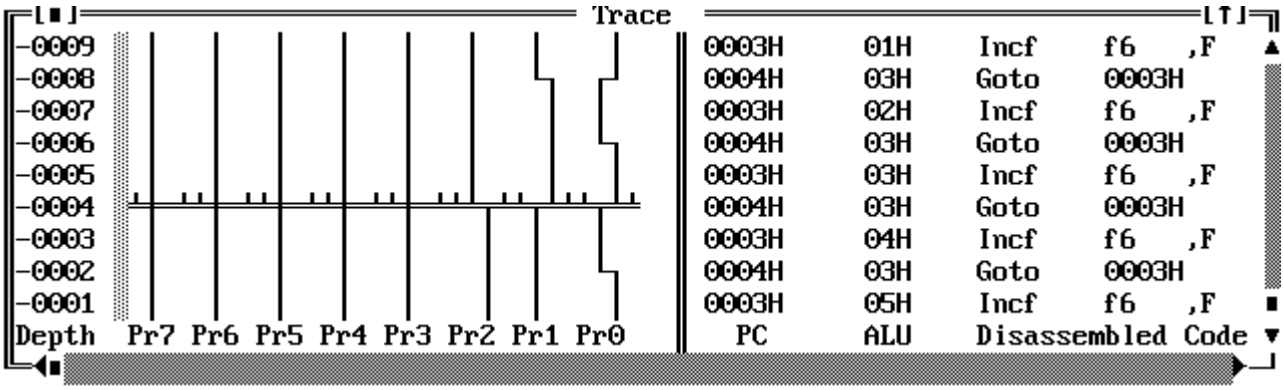
- ***funkce zvýraznění syntaxe (Highlight)*** - funkce se volí globálně pro všechny objekty integrovaného prostředí v uživatelském nastavení, a tudíž na ni reaguje i zpětný překladač. Tato funkce barevně odlišuje významy klíčových slov a znaků.

3.7 Výpis trasovací paměti (Trace Buffer)

Do trasovací paměti se v průběhu ladění programu ukládají informace o běhu programu. Ukládat je možné buď všechny kroky, kterými program procházel, nebo pouze vybrané pasáže programu (podmíněné trasování). Výpis trasovací paměti je uspořádán po jednotlivých adresách a je vypisován tak, že spodní řádek výpisu odpovídá nejbližší minulosti a vrchní řádek výpisu odpovídá nejstaršímu příkazu. Výpis je možné uživatelsky formátovat do nejvhodnějšího tvaru.

3.7.1 Uspořádání trasovací paměti

Trasovací paměť má šířku slova 32 bitů, které jsou z hlediska záznamu informací rozděleny na tři díly. Do prvních šestnácti bitů se v při běhu programu ukládá hodnota programového čítače spolu s podpůrnými informacemi o spouštění a zastavování trasování. Dalších osm bitů paměťového slova je určeno pro uložení výsledků z aritmeticko-logické jednotky. Obsah posledních osmi bitů je volitelný. Buď je možné do nich ukládat hodnoty přesunované po vnitřní datové sběrnici mikroprocesoru, nebo vzorky hodnot z osmi externích logických sond. Výpis trasovací paměti je konstruován



z těchto údajů. Každý řádek výpisu je rozdělen do následujících polí:

- ***Depth*** - ukazatel hloubky trasovací paměti. Jedná se o číslo se znaménkem mínus v rozsahu (-1 až -4096), vyjadřující historii uložených dat.
- ***Probes*** - pole pro osm sloupců pro zobrazení vzorků logických signálů z externích sond. V případě, že je zvoleno trasování interní sběrnice mikropočítače, jsou v těchto sloupcích zobrazována data přenášená po interní sběrnici. Jednotlivé sloupce je možné uživatelsky pojmenovat. V případě zobrazení interní sběrnice jsou zvoleny inplicitní názvy Bs7 .. Bs0. Jména se zadávají pomocí nastavení výpisu trasovací paměti, viz 3.17.
- ***PC*** - ve sloupci je zobrazena hodnota programového čítače pro aktuální historii řádku.
- ***ALU*** - sloupec zobrazuje výsledky aritmeticko-logické jednotky mikropočítače pro danou historii, tj. pro instrukci vykonanou v daném okamžiku
- ***Disassembler*** - sloupec zobrazuje vykonávanou instrukci v dané historii, která je rekonstruována ze zaznamenané hodnoty programového čítače a obsahu programové paměti.

Pohyb kurzoru ve výpisu trasovací paměti je možné ovládat kurzorovými klávesami nebo myší. Přesun kurzoru na počátek řádky lze vyvolat stiskem klávesy *Home*, přesun na konec řádky pomocí klávesy *End*. Klávesami *PgDn* a *PgUp* se kurzor přemístí na další stránku. Klávesy *Ctrl-PgDn* a *Ctrl-PgUp* přesunou kurzor na počátek trasovací paměti, tj. k nejmladšímu údaji, resp. na konec trasovací paměti k údaji nejstaršímu.

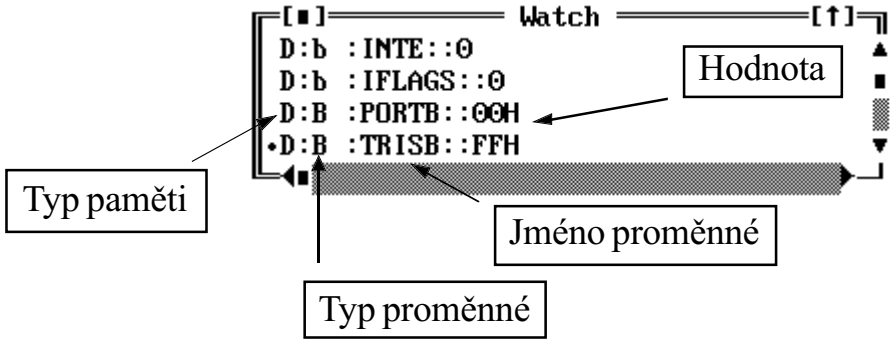
3.7.2 Nastavení a volby

Formátování výpisu trasovací paměti je možné uživatelsky nastavit příkazem *"Option→Trace Memory Setup"* hlavního menu. K dispozici jsou volby typu zobrazit ([X]), resp. nezobrazit ([ ]) pro jednotlivé sloupce pole Probes, sloupec ALU a sloupec Disassembler. U osmice sloupců "Probes" je možné volit pro každý zobrazovaný sloupec třípísmenné jméno, vybírat separátně zobrazení sloupce, a volit mezi číselným nebo semigrafickým zobrazením.

3.8 Uživatelské výpisy proměnných (Watch)

Editor uživatelského výpisu proměnných (Watch) je určen především k formátování výpisu obsahu jednotlivých registrů emulovaného mikropočítače na obrazovku. Editor umožňuje snadnou modifikaci obsahu registrů a zobrazení a modifikaci i registrů procesoru, které nejsou přístupné v jeho datové paměti. Editor uživatelských výstupů umožňuje zobrazit maximálně 256 údajů. Vyšší počet zobrazovaných údajů by v okně působil nepřehledně. Aby i přesto bylo možné sledovat větší počet proměnných, je umožněno současně otevřít více editorů. Jejich počet není programově nijak omezen. Jediné omezení je dáno dostatkem operační paměti osobního počítače.

- ***pohyb kurzoru*** - barevně odlišené podsvícení vybraného údaje a současně tečka v prvním sloupci představuje kurzor v editoru výpisů. K pohybu v okně je možné použít klávesy kurzorových šipek nebo myš. Dále je možné využít kláves *PgDn* a *PgUp* pro stránkování, případně kláves *Ctrl-PgUp* a *Ctrl-PgDn* pro přemístění kurzoru na první resp. poslední záznam editoru.
- ***formát výpisu*** - každá zobrazovaná proměnná je vypisována v předem definovaném formátu, který je částečně pevný a částečně volitelný. Implicitní výpis se týká prvních dvou údajů. První údaj oznamuje typ paměti či registru, a může nabývat hodnot **V**irtual, **D**ata a **A**ddress. Pojmem Virtual označujeme registry, které nejsou obecně dostupné v datové paměti, např. střadač WREG (označení střadače je zvoleno odlišně z důvodu kolize při zápisu cílového registru výstupu operace v instrukci - bit destination může být buď W a nebo F). Pojmem Data jsou označeny registry či proměnné umístěné v programově přístupné paměti dat. Další sloupec výpisu obsahuje údaj o typu proměnné. V tomto slouci se může objevit jedno ze čtyř označení:
  - ***B*** (byte) - označení osmibitové proměnné (bajtu), registru
  - ***b*** (bit) - označení pro jednotlivý bit
  - ***BA*** (byte array) - označení pro pole osmibitových registrů (bajtů)
  - ***bA*** (bit array) - označení pro pole bitů



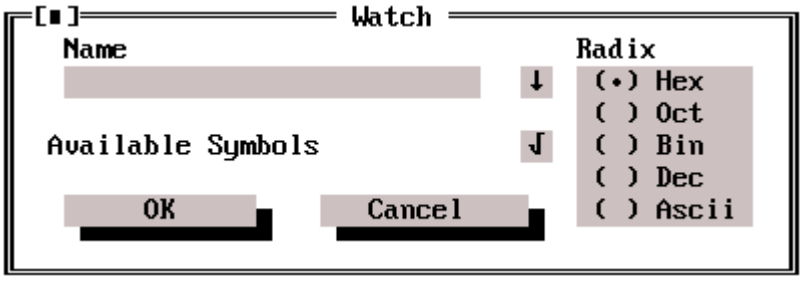
Bližší význam typů proměnných je možné nalézt v kapitole 4.4.2 věnované rozšířeným datovým typům.

Posledním identifikačním pojmem je **A**ddress tj. označení, že se jedná o zobrazení adresy, na které je proměnná v paměti alokována.

Dalším údajem na řádce je jméno proměnné či registru. V případě, že se na řádku zobrazuje adresa registru, předchází jeho jménu znak @. Posledním údajem je hodnota registru (adresy) v emulační paměti mikroprocesoru. Hodnota může být zobrazována celkem ve čtyřech číselných soustavách, tj. v hexadecimální, decimální, oktalové, binární, nebo jako znaky ASCII. Zobrazení adresy je vždy hexadecimální. U adresy bitových proměnných je za čárkou doplněno k adrese i pořadí bitu z rozsahu 0..7. Jedná-li se o pole bitů, pak jsou bity číslovány od počátečního směrem k vyšším řádům dvojkové soustavy a k vyšším adresám bajtů.

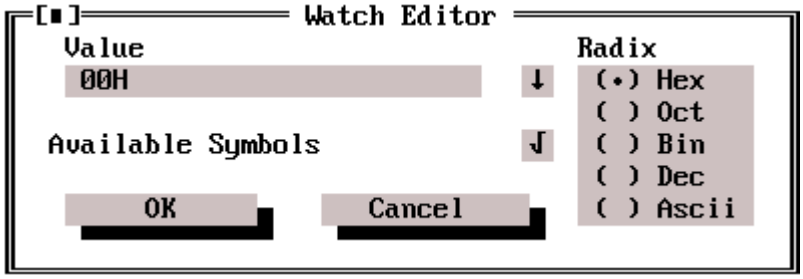
3.8.1 Formátování a editace hodnoty

Přidání proměnné do výpisu je možné provést buď příkazem *"Debug→Add Watch"* z hlavního menu nebo stiskem klávesy *Ins*. Reakce editoru pak závisí na poloze kurzoru. Pokud umístíme kurzor nakonec seznamu, je proměnná přidána. Umístěním kurzoru na položku, která již existuje, můžeme novou položku do seznamu vložit. Editace hodnoty je vyvolána na kurzorem vybrané položce a stiskem klávesy *Enter*.



Přidání proměnné

Editace hodnoty

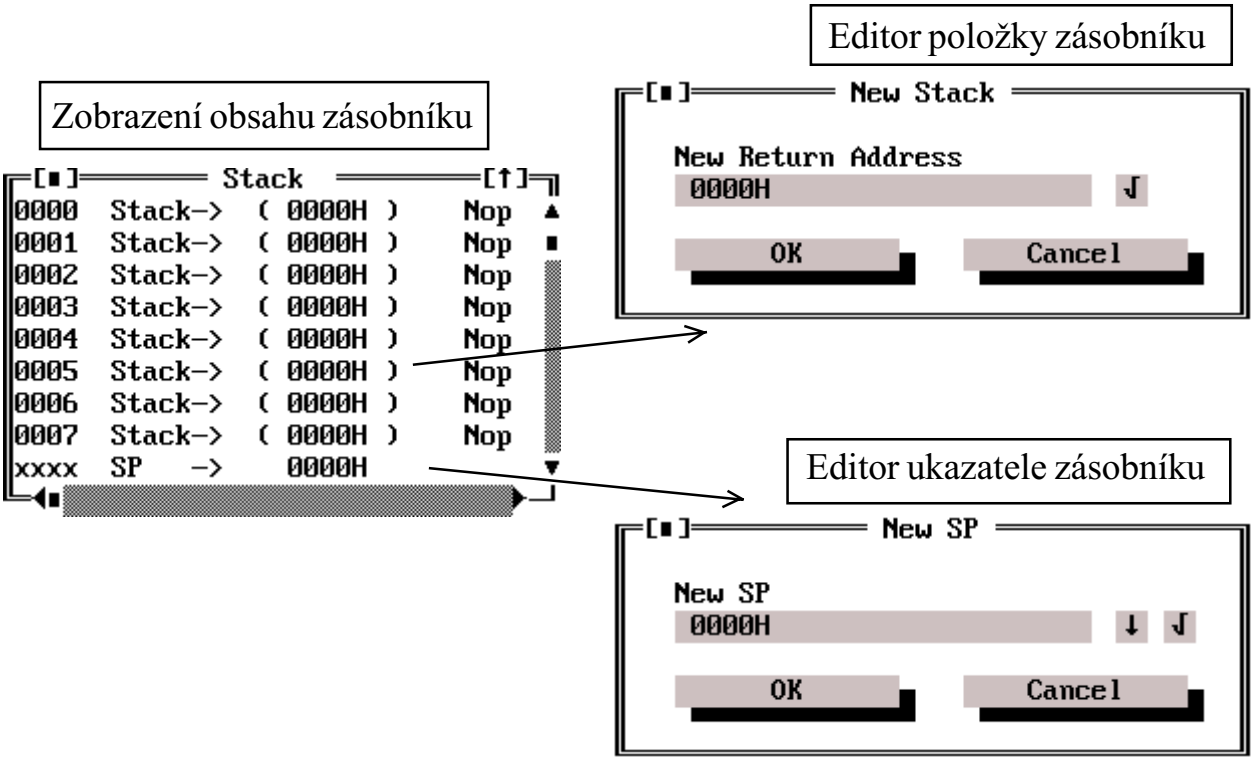


3.9 Výpis a editace zásobníku

Výpis obsahu zásobníku je kostruován podle uložených návratových adres a aktuální hodnoty ukazatele zásobníku. Návratové adresy a hodnotu ukazatele zásobníku je možné editovat pomocí příkazů:

- ***New Stack*** - spustí editor položky zásobníku
- ***New SP*** - spustí editor ukazatele zásobníku

Tyto příkazy vyvoláme pomocí menu a nebo stiskem klávesy *Enter* na příslušné položce zobrazení.



3.9.1 Editor položky zásobníku

Jedná je o prostý řádkový editor vybavený objektem zobrazení dostupných symbolů. V tomto zobrazení jsou uvedeny symboly návěští a konstant zjištěných v průběhu překladu. Návratovou adresu je možné zadat pomocí symbolu (návěští) nebo pomocí číselného zápisu, který je možné vyhodnotit.

3.9.2 Editor ukazatele zásobníku

Editor je určen k modifikaci aktuální hodnoty ukazatele zásobníku. Hodnota se zadá pomocí číselného zápisu a nebo pomocí výrazu, který je možné v aktuálním stavu vyhodnotit.

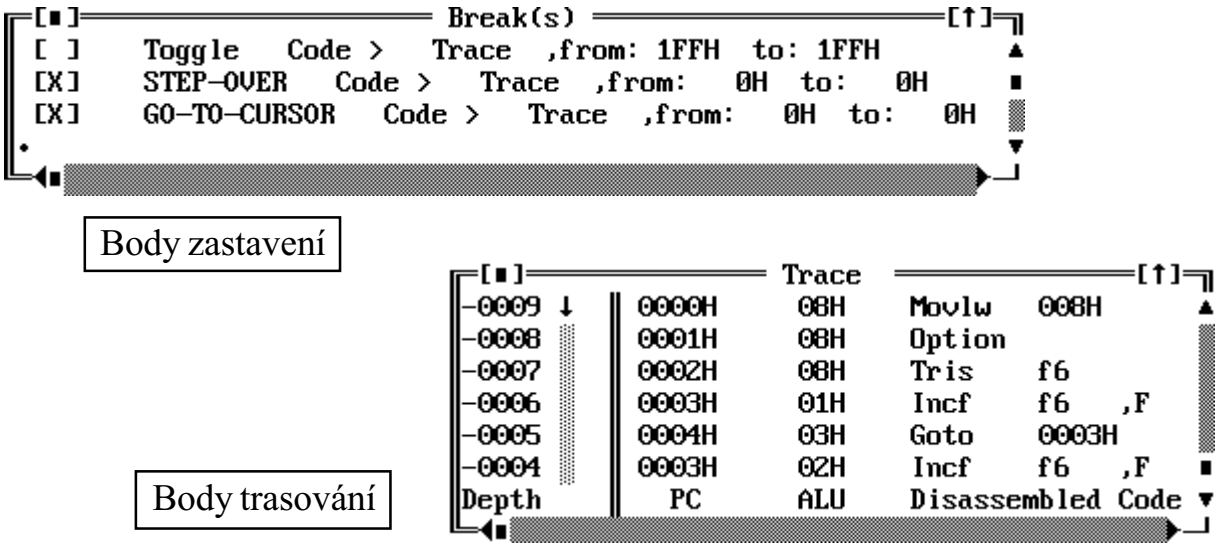
3.10 Výpisy a editace událostí

Mezi události počítáme obecně body zastavení běhu programu (Breakpoints) a body trasování programu (Trace Points). Pomocí editoru událostí je možné zadat až 256 odlišných záznamů, které určují podmínky zastavení či trasování programu aplikace. Pro zobrazení a editaci slouží editor bodů zastavení (Breaks).

3.10.1 Výpisy událostí

- ***pohyb kurzoru*** - kurzor v uvedeném editoru výpisů představuje barevně odlišené podsvícení vybraného údaje a současně tečku v prvním sloupci výpisu. K pohybu v okně je možné použít klávesy kurzorových šipek nebo myš. Dále pak je možné využít kláves *PgDn* a *PgUp* pro stránkování, případně kláves *Ctrl-PgUp* a *Ctrl-PgDn* pro přemístění kurzoru na první resp. poslední záznam editoru.
- ***formát výpisu*** - každá zobrazovaná událost je vypisována v implicitním formátu. Výpis se skládá ze sloupce pro lokální povolení či zakázání aktivity události, sloupce s výpisem alokace události, sloupce s výpisem definice účinku události, jména události a adresového rozsahu události.
- ***povolení/zákaz události*** - povolení nebo zákaz aktivity události je naznačen jedním z následujících výpisů:
  - [ ] - aktivita události je potlačena
  - [x] - aktivita události je lokálně povolena, ale její aktivita je potlačena globálně, tj. událost není aktivní
  - [X] - aktivita události je povolena lokálně i globálně, a tudíž je událost aktivní
- ***alokace události*** - sloupec udává paměťový prostor, do něhož je událost svojí definicí alokována. Alokace události může být dvojího typu:
  - *Code* - bod zastavení na adrese či rozsahu adres programové paměti.
  - *Data* - bod zastavení na adrese či rozsahu adres datové paměti
- ***typ události*** - sloupec udává typ události, pro nějž je záznam definován. Události mohou nabývat následujících typů:

  a) pro alokaci typu Code
  - *Address* - bod zastavení programu na adrese či v rozsahu adres programové paměti (prostý adresový break s možností volit rozsah)
  - *Trace* - bod trasování (průběh programu bude ukládán v zadaném rozsahu adres)



b) pro alokaci typu Data

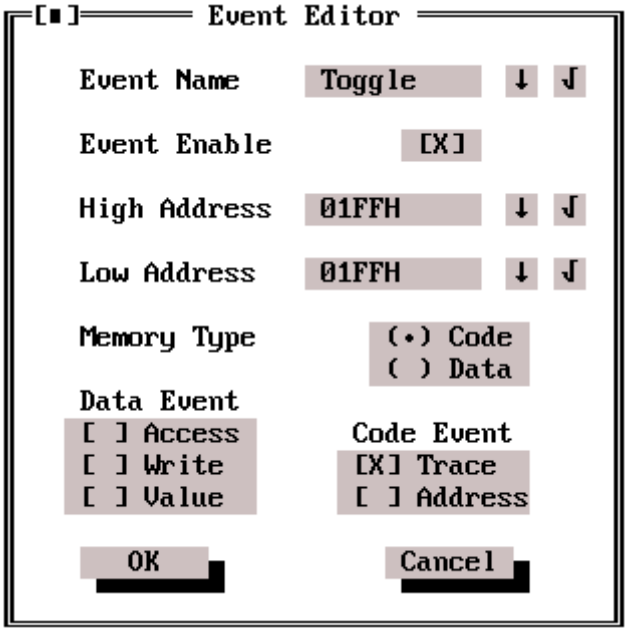
- *Access* - běh programu bude zastaven při přístupu na registr zadané adresy či rozsahu adres (přístupem je míněno čtení nebo zápis libovolné hodnoty)
- *Write* - běh programu bude zastaven při zápisu libovolné hodnoty do registru na dané adrese nebo v zadaném rozsahu adres
- *Value* - běh programu bude zastaven v případě, že v dojde k zápisu na zadanou adresu (rozsah adres) a současně zapisovaná data vyhovují zadané hodnotě včetně bitové masky. Emulátor umožňuje zadat jednu hodnotu a jednu masku společnou pro všechny události tohoto typu. Maska a hodnota jsou implementovány operací AND, tj. z hodnoty jsou platné pouze bity, jejichž pozice odpovídá jedničkovým bitům v masce. Ostatní bity zadané hodnoty jsou ignorovány, tj. na jejich hodnotě nezáleží. Zadáme-li např. hodnotu 0xFF a masku 0x1, a implementujeme-li na registr STATUS bod zastavení typu Value, pak dojde k zastavení vždy v okamžiku, kdy je carry bit (C) nastaven zápisem na hodnotu 1.

V rámci typu alokace je možné typy událostí sdružovat, což se ve výpisu projeví spojením událostí pomocí znaku &. Znak & je míněn ve významu aktivity události, tj. např. událost je na adrese aktivní ve funkci bodu zastavení a současně ve funkci bodu trasování (odpovídající výpis je Code > Address & Trace).

3.10.2 Editace událostí

Pro zadávání a editaci událostí je určen editor událostí. Je možné jej vyvolat příkazy *"Debug→Add Break"* (*Alt-B*) hlavního menu nebo stiskem klávesy *Ins* v okně výpisu událostí v případě, že chceme přidat definici nové události do seznamu. Editaci události

již zadané, vyvoláme stiskem klávesy *Enter* v okně výpisu událostí. V případě, že se kurzor nachází na prázdném řádku výpisu, bude nová událost zadána. Je-li kurzor na události již zadané, bude událost editorem modifikována.



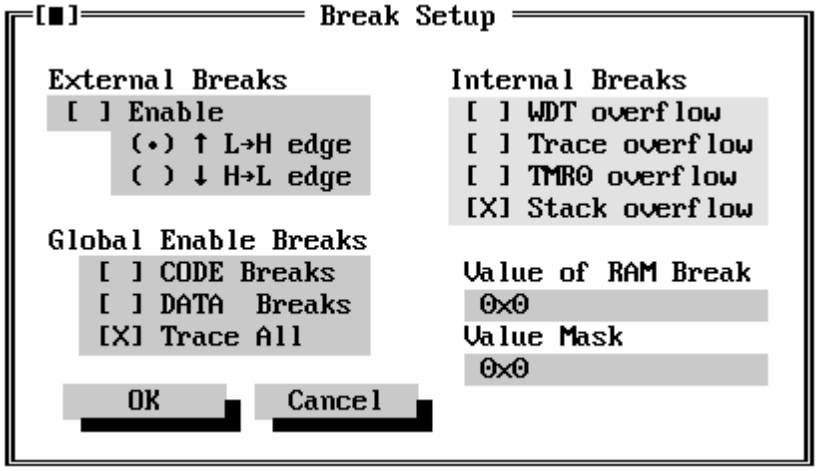
3.11 Globální události

Globální události jsou volby, kterými může uživatel ovlivňovat běh programu a nastavovat různé funkce emulátoru. V nastavení jsou k dispozici nabídky:

- ***External Breaks*** - slouží k povolení zastavení programu externím signálem
  - *Enable* - povolení-zákaz externího signálu zastavení
    - L-H edge
    - H-L edge - volba aktivní hrany signálu zastavení
- ***Internal Breaks*** - souhrn voleb pro vyvolání zastavení programu některou z interních událostí emulačního čipu
  - *Wdt overflow* - zastavení v okamžiku přetečení čítače "Watch-Dog"
  - *Trace overflow* - zastavení v okamžiku naplnění trasovací paměti
  - *TMR0 overflow* - zastavení v okamžiku přetečení časovače TMR0. (Původní název TMR0 je RTCC, pro kompatibilitu se současnou dokumentací firmy Microchip se budeme držet nového názvu.)
  - *Stack overflow/underflow* - zastavení v okamžiku přetečení nebo podtečení ukazatele zásobníku

- ***Global Enable Breaks*** - společná aktivace a deaktivace bodů zastavení a trasování
  - *Code Breaks* - povolení/zákaz všech bodů zastavení v programové paměti
  - *Data Breaks* - povolení/zákaz všech bodů zastavení v datové paměti
  - *Trace All* - povolení/zákaz podmíněného trasování (je-li podmíněné trasování zakázáno, trasují se všechny instrukce)
- ***Value of Ram Breaks, Value Mask*** - editor hodnoty a masky pro body zastavení v datové paměti podle hodnoty a masky (hodnotu a masku je možné zadat dekadicky nebo hexadecimálně v konvenci jazyka C, tj. 0x10 apod)

Tlačítko OK potvrzuje nastavení. Tlačítko Cancel způsobí ignorování změn v nastavení.



3.12 Nastavení emulátoru

Volby Nastavení Emulátoru slouží k základní konfiguraci emulačního čipu a tomu odpovídajícímu nastavení integrovaného prostředí.

- ***Processors*** - nastavení typu emulovaného procesoru
- ***Clock Source*** - volba zdroje hodinového signálu pro emulovaný procesor.
  - *Internal* - spouští interní zdroj hodin. V tomto případě je hodinová frekvence nastavena parametrem Clock.
  - *External* - zdroj hodinového signálu pro emulační čip je získáván z aplikace prostřednictvím hodinového vstupu procesoru

    Default: Internal

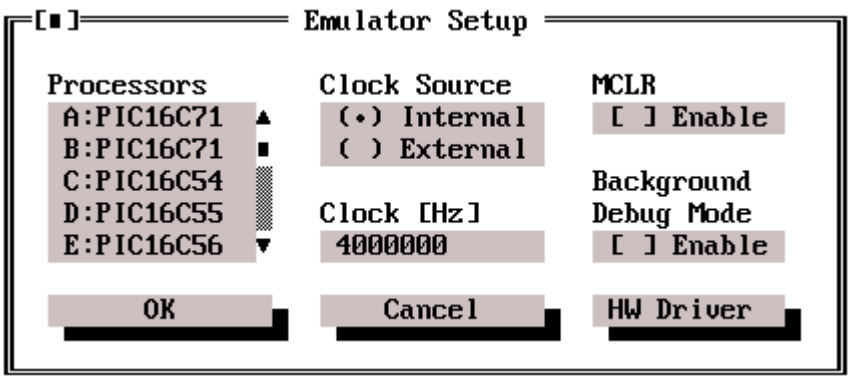
Poznámky, doporučení:

1. Externí zdroj hodin používejte výhradně v případech, kdy je z hlediska aplikace naprosto nezbytný. Z důvodu rychlosti odezvy systému používejte externí zdroj hodin s kmitočtem alespoň 25kHz. Naprostá většina interních operací emulačního

čipu je totiž synchronní s hodinami emulovaného procesoru, a k provedení potřebuje několik taktů hodin.

2. V případě volby Internal je vstup OSC1/CLKIN emulovaného procesoru ignorován, v případě volby External se na tento vstup přivádí aktivní hodiny z externího zdroje. V obou případech se tedy ignorují jakékoli připojené krystaly, keramické rezonátory nebo RC členy.

- ***MCLR Enable*** - povolení resetu emulačního čipu z pinu -MCLR



emulovaného procesoru nebo při poklesu napájecího napětí emulovaného procesoru pod hodnotu asi 2.5 V.

Default: Disabled

- ***Clock*** - nastavení hodinového kmitočtu pro interní zdroj hodin. Kmitočet lze nastavit v rozsahu od 25kHz do 20MHz, resp. do maximálního povoleného kmitočtu emulovaného procesoru. Zadaná hodnota se zaokrouhlí na nejbližší hodnotu, kterou je interní frekvenční syntezátor schopen vygenerovat (krok 40Hz). Maximální odchylka, způsobená tímto zaokrouhlením, je tedy pro minimální povolenou frekvenci 25 kHz menší než 0.1%, a pro vyšší kmitočty dále úměrně klesá a stává se zanedbatelnou proti vlastní přesnosti použitého krystalového oscilátoru. Zobrazovaná hodnota je vždy zadaná hodnota (nezaokrouhlená).

- ***Background Debug Mode***
  - *Enable* - vyčítání a editace datové paměti emulátoru v reálném čase, tj. za plného běhu programu v emulačním čipu.
  - *Disable* - vyčítání a editace datové paměti emulátoru pouze ve stavu Halted.

Pozn.: V režimu Background Debug Mode je umožněno editovat datovou paměť mikropočítače. Při editaci je však nutné si uvědomit, že emulační čip běží oproti editaci mnohonásobně rychleji, a tudíž editace může při nesprávném použití zhroutit aplikační program. V tomto režimu se též provádí automaticky cyklické vyčítání datové paměti.

- ***Hw Driver*** - stisknutím tlačítka se vyvolá okno s popisem hardwarového ovladače emulátoru. V tomto popisu jsou uvedeny vlastnosti ovladače a případná omezení v emulaci zvoleného typu procesoru.

Tlačítko OK potvrzuje nastavení. Tlačítko Cancel způsobí ignorování změn v nastavení.

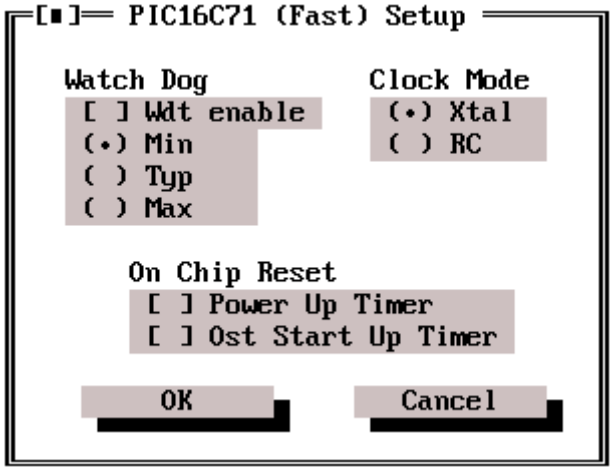
3.13 Chip options

je dialog, kterým se nastavují speciální funkce emulovaných čipů. Obsah dialogu se mění podle aktuálně nastaveného procesoru. Uveďme alespoň nejčastější položky dialogu.

- ***Watch Dog***
  - *Wdt Enable* - povolení činnosti čítače Watchdog timer (WDT). I v případě, že je WDT povolen, je možné program krokovat, aniž by to narušilo jeho funkci, protože WDT je v režimu krokování mezi jednotlivými kroky pozastavován. Default: Disabled
  - *Min, Typ, Max* - trojice voleb pro nastavení doby přetečení WDT (volby najdou uplatnění při toleranční analýze programu, v němž je WDT využit). Hodnoty jsou nastaveny na základě katalogových údajů výrobce pro daný typ procesoru, např. pro PIC16C5x na 9, 18 a 30 ms.

    Default: Typical
- ***Clock Mode*** - nastavení režimu výstupu OSC2/CLKOUT
  - *Xtal* - na výstupu CLKOUT je kmitočet zadaný volbou Clock.
  - *RC mode* - na výstupu CLKOUT je k dispozici čtvrtinový kmitočet tak, jak to odpovídá specifikaci emulovaných procesorů s oscilátorem RC.

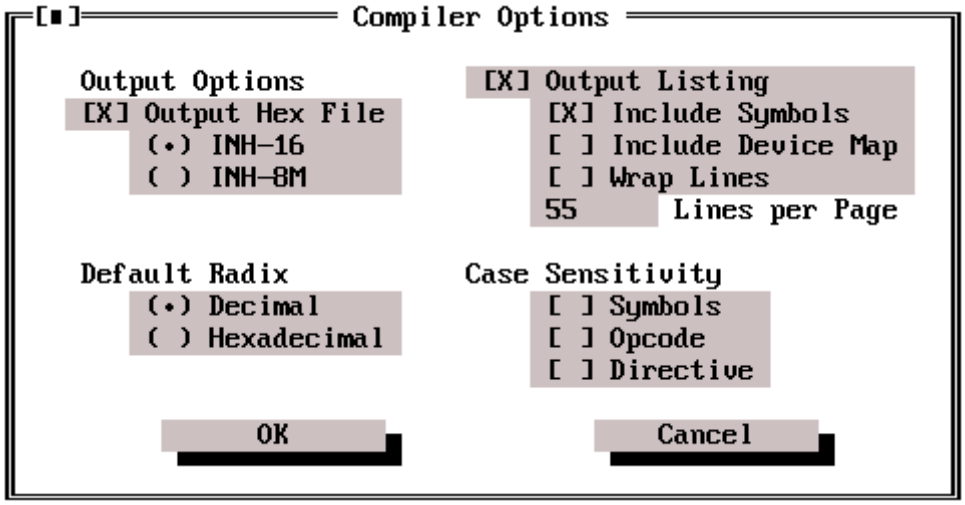
    Default: Xtal
- ***On Chip Reset***
  - *Power Up Timer* - zapnutí časovače pro Power Up Reset.
  - *Ost Start Up Timer* - zapnutí časovače “Start Up Timer”.



3.14 Nastavení parametrů překladače

Toto nastavení ovlivňuje běh integrovaného překladače. Jedná se zde především o uživatelské volby pro formátování výstupních souborů.

- ***Output Options***
  - *Output HEX File* - generování výstupního souboru ve formátu Intel Hex
    - INH-16 - výstupní soubor bude uložen ve formátu Intel Hex-16.
    - INH-8M - výstupní soubor bude uložen ve formátu Intel Hex-8M.
- ***Output Listing*** - povolení generování výstupního souboru s textovým protokolem o překladu
  - *Include Symbols* - ve výpisu bude/nebude uveden seznam symbolů.
  - *Include Device Map* - ve výpisu bude/nebude uvedeno pouzdro procesoru, pro který byl proveden překlad.
  - *Wrap Lines* - překladač bude zalamovat řádky v případě, že budou po překladu přesahovat 80 znaků.
  - *Lines Per Page* - nastavení počtu řádek na jednu stránku výpisu.
- ***Default Radix*** - nastavení základní číselné soustavy, v níž překladač interpretuje zadávané číslice bez uvedené specifikace soustavy
  - *Decimal* - všechna čísla, u nichž není spacifikována soustava, budou interpretována jako dekadická.
  - *Hexadecimal* - všechna čísla, u nichž není spacifikována soustava, budou interpretována jako hexadecimální.



- ***Case Sensitivity*** - překladač bude rozlišovat velká a malá písmena u:
  - *Symbols* - symbolů, tj. jmen proměnných, návěští, atd..
  - *Opcode* - u mnemoniky instrukcí.
  - *Directive* - u direktiv překladače.

Tlačítko OK potvrzuje nastavení. Tlačítko Cancel způsobí ignorování změn provedených v nastavení.

Poznámky:

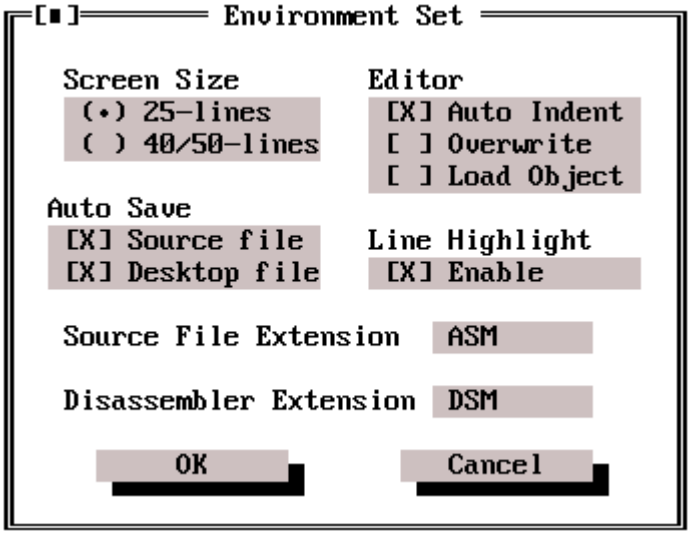
a) Mnemonika instrukcí pro rozlišování velkých a malých písmen se zapisuje vždy s prvním písmenem velkým. Ostatní písmena jsou malá. Totéž platí i pro direktivy překladače. U symbolů jsou písmena rozlišena podle prvního zápisu symbolu ve zdrojovém textu.

b) Parametrem překladače je též typ procesoru, pro který je překlad prováděn. Tento parametr je překladači předáván z nastavení emulátoru.

3.15 Nastavení parametrů prostředí

Nastavení parametrů prostředí slouží k uživatelské konfiguraci vzhledu obrazovky a chování systému v určitých specifických stavech. Nastavení se ukládá vždy s projektem (je-li otevřen) nebo do souboru DPIC.PRJ v pracovním adresáři.

- ***Screen Size***
  - *25* , *40/50 lines* - nastavení počtu řádků na obrazovce.
- ***Editor*** - nastavení parametrů editoru zdrojového textu, s nimiž bude editor spouštěn.
  - *Auto Indent* - automatické odsazování začátku textu podle předchozího řádku.
  - *OverWrite* - zapnutí režimu přepisování.
  - *Load Object* - v případě, že nebude otevřen žádný projekt, pokusí se prostředí nalézt a nainstalovat do emulační paměti přeložený “object soubor” podle jména otevíraného souboru.
- ***Auto Save***
  - *Source File* - všechny zdrojové soubory budou automaticky uloženy při ukončení práce v integrovaném prostředí.
  - *Desktop File* - automaticky bude uloženo nastavení prostředí včetně rozmístění oken na pracovní ploše.
- ***Line Highlight*** - všechny typy editorů budou zobrazovat jednotlivé skupiny symbolů a značek zdrojového textu přidělenými barvami (jedná se též



o jednoduchou syntaktickou kontrolu pro mnemoniku instrukcí, známých operandů apod.)

- ***Source File Extension*** - třípísmenná extenze pro soubory obsahující zdrojový text
- ***Disassembler Extension*** - třípísmenná extenze pro ukládání souborů s disasemblovanými soubory

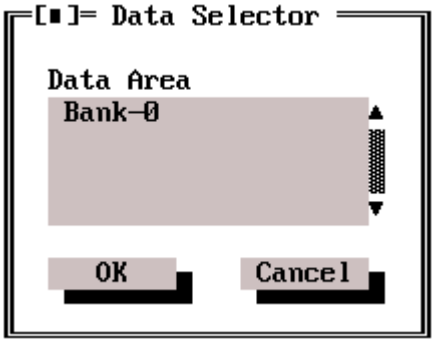
Tlačítko OK potvrzuje nastavení. Tlačítko Cancel způsobí ignorování provedených změn.

3.16 Nastavení parametrů výpisu pamětí

Příkazem *"Option→Data Memory Setup"* se vyvolá okno Data Selectoru, kde zvolíme registrovou banku a po stisku klávesy Enter se vyvolá okno editoru nastavení výpisu datové paměti. Nastavení výpisu programové paměti se vyvolá obdobně příkazem *"Option→Program Memory Setup"*.

3.16.1 Data Selector

Okno Data Selectoru je tvořeno seznamem Data Area, který je vybaven vertikální rolovací lištou a kurzorem. Pomocí kurzoru zvolíme banku (oblast) datové paměťi a stiskem klávesy *Enter* vyvoláme editor pro nastavení obsahu paměti. Tlačítkem Cancel můžeme volbu zrušit a uzavřít okno Data Selectoru.

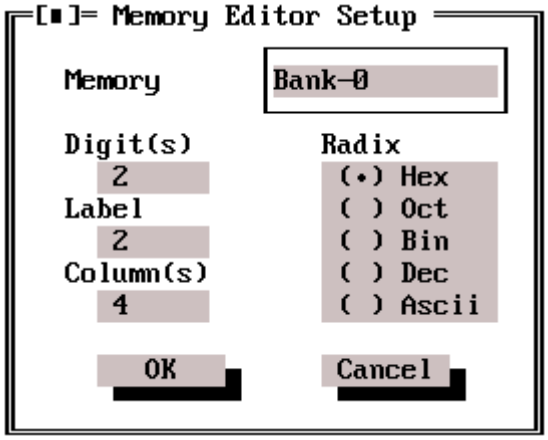


3.16.2 Editor nastavení výpisu

Editorem nastavení výpisu můžeme výpis formátovat následujícími parametry:

- ***Digit(s)*** - řádkový editor pro nastavení počtu číslic pro výpis jedné paměťové buňky. Povolený rozsah je od 1 do 16. Je-li zvolený počet míst menší než potřebný, pak bude číslo oříznuto zleva tj. od vyšších řádů. V případě, že zadaný rozsah bude větší, než je potřebný počet číslic, dojde ve výpisu k předsazení nul, tak aby byl požadovaný počet číslic dodržen.
- ***Label*** - nastavení počtu míst pro zobrazení adresy. Povolený rozsah je 1 až 8. Adresa je zobrazována v hexadecimální soustavě s případnými předřazenými nulami.
- ***Columns*** - nastavení počtu sloupců výpisu v rozsahu 1-5
- ***Radix*** - nastavení požadované soustavy pro výpis obsahu paměti (v takto zvolené soustavě bude pracovat i příslušný editor výpisu).

Tlačítko OK potvrzuje nastavení. Tlačítko Cancel způsobí ignorování provedených změn.



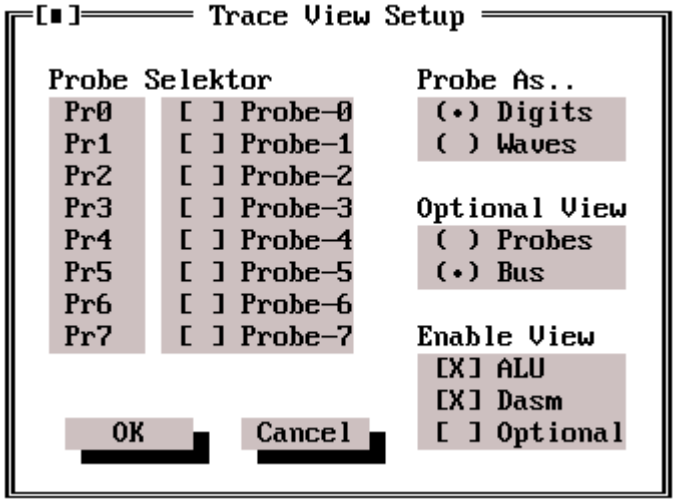
3.17 Nastavení výpisu trasovací paměti

- ***Probe Selector***
  - *Probe* - umožňuje zadat pro každý vypisovaný signál sond třípísmenný název.
  - *Selector* - umožňuje vybrat sondu, která bude ve výpisu zobrazována.
- ***Probe As..*** - přepínač typu zobrazení sond dává volbu mezi semigrafickým a číselným zobrazením logické hodnoty.
- ***Optional View*** - přepínač mezi interními a externími sondami. Interní sondy jsou vždy zapojeny na interní datovou sběrnici procesoru v cyklu Q1. Pro instrukce pracující s File Registry se zde objevuje vstupní hodnota registru, pro instrukce

s literály je zde kopie literálu. Externí sondy jsou vyvedeny na konektor, a je možné je zapojit libovolně do aplikace v rozsahu napájecího napětí emulovaného procesoru. Rozhodovací úroveň je přibližně 1.6V. Do trasovací paměti se zapisuje jejich stav na konci cyklu Q1, tedy ve stejném okamžiku, kdy se čte stav pinů procesoru.

- ***Enable View*** - povolení zobrazení jednotlivých bloků trasovací paměti v okně.
  - *ALU* - zobrazení výstupu aritmeticko-logické jednotky.
  - *Dasm* - zobrazení disasemblovaného programu.
  - *Optional* - zobrazení nastaveného typu sond, tj. buď externí nebo interní, podle nastavení v Option a View.

Tlačítko OK potvrzuje nastavení. Tlačítko Cancel způsobí ignorování provedených změn.



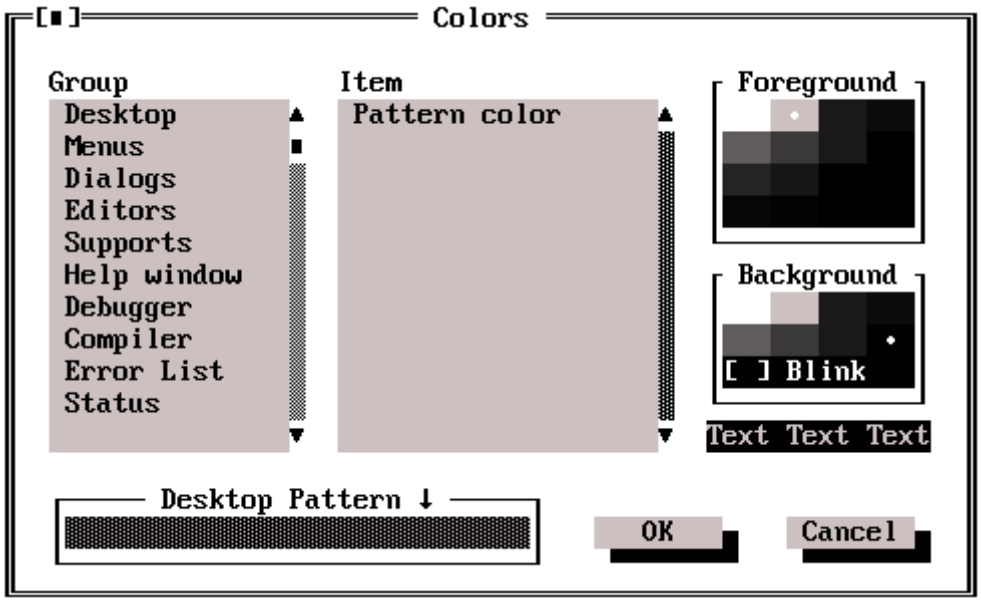
3.18 Nastavení barev prostředí

Tato volba umožňuje uživateli nastavit barvy jednotlivých objektů na obrazovce spolu s dalšími parametry zobrazení. Okno nastavení je rozděleno do bloků:

- ***Group*** - seznam umožňující volbu hlavní skupiny objektů
- ***Item*** - seznam pro volbu podskupiny objektů ze skupiny Group
- ***Foreground*** - nastavení barvy textu pro zvolený objekt
- ***Background*** - nastavení barvy podkladu pro text zvoleného objektu
  - *Blink* - nastavení atributu blikání pro text zvoleného objektu
- ***Desktop Pattern*** - nastavení vzorku pracovní plochy

Pozn.: Volba vzorku pracovní plochy (Desktop pattern) umožňuje volbu znaku,

který bude zobrazován na pracovní ploše. Vybereme-li v dialogovém okně toto nastavení, pak pomocí stisku šipky dolů vyvoláme okno se zobrazením rozšířených ASCII znaků. Kurzorem zvolíme požadovaný znak a stiskneme klávesu *Enter*. Znak je po uzavření dialogového okna pro nastavení barev zaveden na pracovní plochu a vytvoří tak podklad.



3.19 Pomocné objekty na pracovní ploše

Pracovní plocha je mimo menu tvořena ještě pomocnými objekty zobrazení, udávajícími informace o integrovaném prostředí a o emulačním systému.

3.19.1 Status

Status je umístěn na horní liště společně s hlavním menu. Slouží k zobrazení stavu emulátoru, hodnoty programového čítače (pokud je známa) a typu emulovaného procesoru. Jednotlivé stavy emulátoru jsou:

- ***Undefined*** - neznámý stav v průběhu spouštění prostředí, kdy ještě není kontakt s emulátorem, nebo poskytl-li emulátor nějaká neznámá data při komunikaci.
- ***Error*** - stav, který nastane v okamžiku zjištění nějaké chyby v komunikaci s emulátorem nebo při požadavku od uživatele, který se nepodařilo splnit. Např. verifikace obsahu paměti programu nebyla úspěšná.
- ***Running*** - označuje reálný běh emulátoru.
- ***Halted*** - emulátor je zastaven.
- ***Break*** - emulátor zastavil při splnění podmínky zastavení.

- ***Step*** - je vykonáván programový krok
- ***Download*** - probíhá konfigurace emulátoru
- ***Load*** - probíhá načítání přeloženého kódu do emulační paměti
- ***Verify*** - probíhá verifikace kódu z emulační paměti
- ***Busy*** - emulátor je zaneprázdněn a čeká se na dokončení zadané operace

3.19.2 Programový manažer

Zobrazení stavu programového manažeru je umístěno uprostřed spodní lišty pracovní plochy. Zobrazení má dvě části. Nalevo od dělící čáry se zobrazuje jméno projektu a napravo jméno objektového souboru, který je aktuálně přítomen v emulační paměti. Není-li otevřen projekt nebo objekt, zobrazuje se v příslušném poli stav "None".

3.19.3 Indikátor dostupné paměti

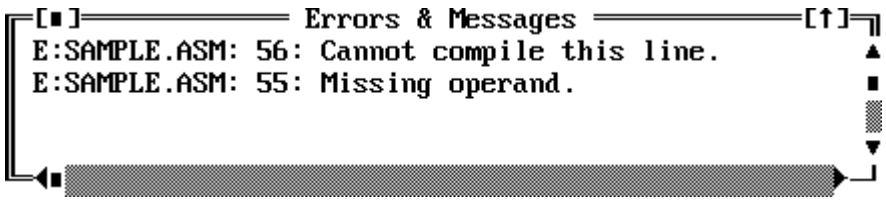
V pravém dolním rohu je zobrazována velikost dostupné paměti pro integrované prostředí. Hodnota je uvedena v bajtech.

3.20 Okno chybových hlášení překladače

Okno chybových hlášení překladače je objekt pro usnadnění hledání řádků s chybami vyhodnocenými překladačem. Okno je automaticky otevřeno při nalezení jakékoliv chyby ve zdrojovém textu v libovolné fázi překladu. Po otevření okna jsou chyby zobrazeny ve formátu:

číslo řádky, soubor, typ chyby

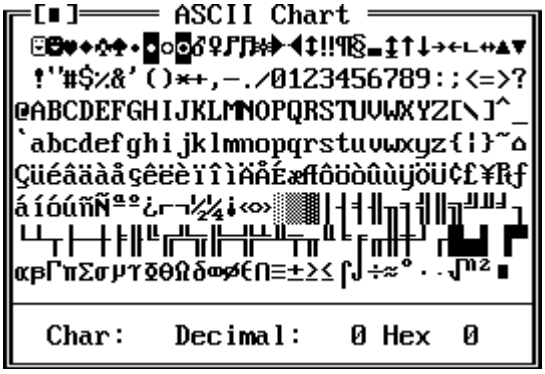
Stisknutím klávesy *Enter* je ve zdrojovém textu nalistován řádek, kde byla chyba nalezena, a současně je editor zdrojového textu aktivován. Hledání dalších chyb je možné pomocí kláves *Alt-F8* pro přechod k následující chybě a *Alt-F7* pro přechod k předcházející chybě. V případě, že se překlad skládá z více souborů zdrojového textu a chyby jsou alespoň ve dvou z nich, je možné stiskem klávesy mezerníku zvolenou chybu vybrat a označit. Listování chyb pak bude probíhat pouze v souboru, jehož chyby byly takto označeny.



Ve zdrojovém textu je možné nalezenou chybu ihned opravovat, aniž by se ztratily odkazy na chyby další. Pokud řádek s chybou vymažeme, dojde k vymazání příslušné chyby z výpisu chyb.

V průběhu překladu může dojít k nalezení nesrovnalostí či jiných problémů, které vyžadují vydat uživateli upozornění. V těchto případech není okno chyb otevřeno automaticky. Pokud ho chceme otevřít, stiskneme v editoru zdrojového textu kombinaci kláves *Alt-A*.

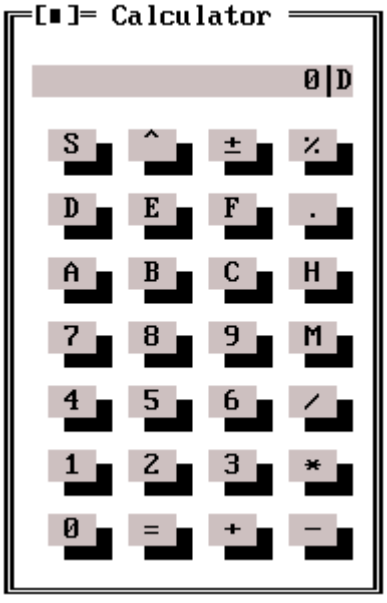
3.21 ASCII tabulka



Pomocí příkazu *"≡→ASCII"* je možné otevřít okno se zobrazením ASCII tabulky. Ve spodní části okna je umístěna lišta, na níž je zobrazen znak vybraný kurzorem společně s jeho číselnou reprezentací v dekadické a hexadecimální soustavě.

3.22 Calculator

Pomocí příkazu *"≡→Calculator"* vyvoláme okno s jednoduchým kalkulátorem zpracovávající celá čísla v rozsahu -32768 až 32767. Kalkulátor pracuje v dekadické, hexadecimální a binární číselné soustavě.

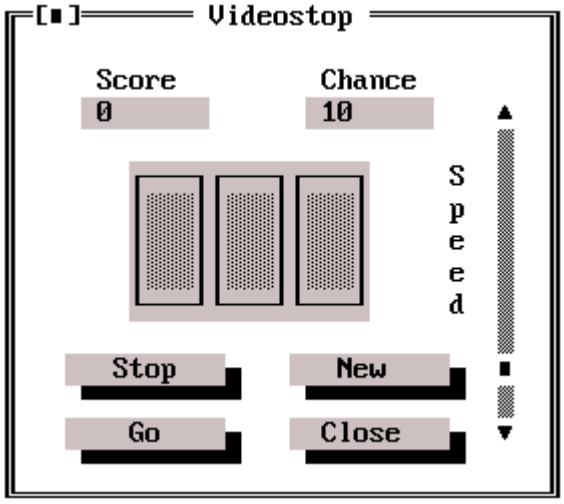


Umožňuje operace:

mazání chyby	-	klávesa S
změna znaménka	-	klávesa _
druhý doplněk	-	klávesa ^
sčítání	-	klávesa +
odčítání	-	klávesa -
celočíselné dělení	-	klávesa /
zbytek po dělení	-	klávesa M
násobení	-	klávesa *
provedení výpočtu	-	klávesa Enter
dekadická soustava	-	klávesa .
hexadecimální soustava	-	klávesa H
binární soustava	-	klávesa %

Klávesami odpovídajícími ostatním tlačítkům se zadávají číselné operandy výpočtů s tím, že pro binární soustavu mají význam pouze tlačítka 1 a 0, pro dekadickou tlačítka 0 až 9, a pro hexadecimální tlačítka 0 až 9 a A až F.

3.23 Videostop



Nefunguje-li emulovaná aplikace, videostop Vás spolehlivě dorazí. Videostop je jednoduchá hra pro chvíle oddechu. Cílem je po spuštění pomocí tlačítka Go zastavit překlápění kostek v okamžiku, kdy jsou číslice alespoň na dvou kostkách stejné. K dispozici je při jedné hře deset pokusů. Za každý úspěšný pokus je započítáno 10 bodů. Při neúspěšném pokusu je deset bodů odečteno. Podaří-li se zastavit kostky v okamžiku, kdy všechny tři zobrazují stejné číslo, je přidělena prémie 100 bodů. Pomocí rolovací lišty je možné nastavit rychlost překlápění kostek. Pomocí prvního stisku tlačítka Go jsou odkryty kostky. Druhým stiskem téhož tlačítka spustíme překlápění kostek. Tlačítkem Stop zadáváme okamžik pro zastavení kostek. Tlačítko New umožňuje nastavit hru do počátečního stavu. Tlačítkem Close se ukončí hra a okno Videostopu je uzavřeno.

3.24 Information

Okno Information podává nejzákladnější informace o verzi hardware a software.

3.25 Chybová hlášení integrovaného prostředí

Symbol not exist

Zadaný symbol je neznámý, a tudíž nepřístupný. Chyba může být způsobena překlepem a nebo tím, že zadání nepředcházela úspěšná kompilace zdrojového textu.

Syntax error

Ve výrazu byla nalezena chyba, a výraz nelze vyhodnotit. Chyba může být způsobena překlepem ve jménu symbolu, chybným zápisem závorek u prvku polí, neznámým operandem apod.

Out of symbol range

Byl překročen počet symbolů zobrazitelných oknem Watch. Chybu odstraníme otevřením dalšího okna a zadáním požadovaného symbolu v něm.

Not accessible symbol

Zadaný symbol není dostupný z důvodu neúplné definice. Chyba vzniká např. při editaci konstanty apod.

Insufficient memory

Nepodařilo se zaalokovat požadovaný blok paměti. Chyba vzniká při alokaci pracovních paměťových struktur.

Error of breakpoint type

Byl nalezen chybný typ bodu zastavení nebo trasování. Chyba vznikla v průběhu definice události.

Out of break memory

Byl překročen přípustný počet (256) definic událostí. Chybu můžeme částečně eliminovat použitím většího rozsahu adres v definicích událostí.

Not accessible break(s)

Bod zastavení není dosažitelný. Např. rozsah je mimo přípustnou oblast paměti emulovaného procesoru.

Break out of range

Definice události obsahuje položku, která činí událost hardwarově nedostupnou. Převážně se jedná o překročení paměťového rozsahu pro nastavený typ procesoru.

Unable to edit address

Pokus o editaci na adrese, kde to není v daném okamžiku a za daných nastavení systému možné.

Value has been unchanged

Při pokusu o zápis do emulační paměti se nepodařilo hodnotu paměťové buňky změnit.

Access denied

Požadovaná akce byla odmítnuta hardwarem nebo firmwarem emulátoru.

Disk Full

Kapacita disku byla vyčerpána, a uložení souboru nebylo korektně dokončeno.

Unable to open *.DEF file

Nebylo možné otevřít definiční soubor procesoru. Na chybu prostředí reaguje nastavením implicitních hodnot. Chybu odstraníme zpřístupněním definičního souboru pro čtení.

*.DEF file read error

Nastala chyba čtení definičního souboru. Soubor je poškozen. Chybu opravíme přeinstalováním souboru z instalačních disket.

*.DEF file not found

Definiční soubor procesoru nebyl nalezen v domovském adresáři integrovaného prostředí.

Could not open project file

Soubor s definicí projektu není možné otevřít. Soubor nebyl nalezen nebo vznikla chyba čtení.

Could not create project file

Nebylo možné vytvořit soubor projektu. Chyba vznikla buď nedostatkem místa na disku, nebo interní chybou v prostředí.

Project not opened

Pokus o uzavření projektu, aniž by byl nějaký otevřen.

Unable to load object file

Soubor objektu nebyl nalezen, nebo nastala chyba čtení.

Can not compile this line

Řádkový překladač není schopen přeložit zadaný řádek.

Download file not found

Nebyl nalezen inicializační soubor emulátoru. Jedná se o soubory s příponou BIN, které musí být umístěny v domovském adresáři integrovaného prostředí.

Download error

Chyba při konfiguraci emulátoru. Jedná se o poruchu v hardware, nebo nekompatibilitu konfiguračního souboru.

Unable to find help file

Nenalezen soubor nápovědy. Soubor musí být umístěn v domovském adresáři integrovaného prostředí. Má příponu HLP.

Uncompatible project format

Pokus o otevření projektu, který svojí strukturou neodpovídá formátu projektu integrovaného prostředí.

Recompile the program first

Byla nalezena nekonzistence mezi vznikem souboru objektu a zdrojového souboru. Chybu odstraníme novou kompilací zdrojového textu.

No Line to PC reference exists

Zpracovávaná řádka nemá referenci na adresu v programové paměti. Chyba může být způsobena pokusem o zadání přepínacího bodu zastavení po neúspěšném překladu nebo pokusem zadat bod zastavení na řádek, který nemá reprezentaci v operačním kódu, tj. např kometářový řádek.

Uncompatible object format

Formát souboru objektu neodpovídá strukturou objektu integrovaného prostředí.

Unable to Evaluate Expression

Vyhodnocení zadaného výrazu není možné z důvodu nesprávné syntaxe, neexistence symbolu apod.

Error in Emul Init Procedure

Chyba při instalaci ovladače emulovaného procesoru. Jedná se o interní chybu, která může být způsobena nekompatibilitou ovladače a hardware, vadným komunikačním portem tiskárny, špatným kabelem nebo poruchou emulátoru.

3.26 Informační hlášení prostředí

Save current desktop ? , Save current project ?

Není-li nastaveno automatické ukládání obsahu pracovní plochy po ukončení práce s projektem, je uživatel požádán o potvrzení pokusu uložit aktuální nastavení prostředí.

Source has been changed - Rebuild ?

Zdrojový text neodpovídá souboru, z něhož byl vytvořen překlad. Neexistují tedy reference na řádky, nebo tyto reference neodpovídají aktuálnímu stavu. V případě, že takový zdrojový text nebude zkompilován, pak v průběhu krokování programem nebude ve zdrojovém textu zvýrzněn řádek odpovídající hodnotě programového čítače.

Load object file ?

V případě, že není otevřen žádný projekt a uživatel zadá příkaz k otevření souboru s extenzí odpovídající extenzi zdrojového textu v nastavení prostředí, je požádán o potvrzení změny obsahu emulační paměti. Není-li změna potvrzena soubor bude otevřen s tím, že nebudou naistalovány reference na řádky zdrojového textu se všemi důsledky z toho plynoucími.

Emulator not found. Run Demo ?

V případě, že nebyla navázána komunikace s emulátorem, je možné situaci řešit buď potvrzením žádosti, tj. bude spuštěn demo simulátor, anebo odmítnutím žádosti tj. prostředí se bude pokoušet znovu navázat kontakt s emulátorem. V případě, že emulátor není k dispozici, je možné prostředí spustit přímo v demo módu pomocí parametru /DEMO.

Project Exists - Overwrite ?

Žádost o potvrzení akce příkazu Save Project as.. v okamžiku, kdy projekt zadaného jména již existuje, a byl by tudíž přepsán.

PC Overflow

Uživatel je informován o situaci, kdy došlo k nekorektnímu přetečení čítače instrukcí vlivem chyby v programu vyvíjené aplikace. K situaci dojde v případě, že v průběhu vykonávání programu je hodnota čítače instrukcí změněna z maxima na nulu, tj. uživatelský program "zabloudil". Hlášení je možné potlačit pomocí zákazu této akce v nastavení Global Events.

TMR0 Overflow

Došlo k přetečení čítače/časovače. Hlášení je možné potlačit pomocí zákazu v nastavení Global Events.

WatchDog OverFlow

Došlo k přetečení čítače Watch Dog s následným resetem procesoru. Hlášení je možné zakázat spolu se zákazem zastavení v Global Events.

Stack Overflow

Došlo k přetečení nebo podtečení zásobníku. K chybě dojde v okamžiku, kdy je v aplikačním programu voláno více podprogramů nebo více návratů z podprogramů než dovoluje velikost zásobníku emulovaného procesoru. V aplikačních programech však může být přetečení nebo podtečení zásobníku záměrně používáno. Pro takové aplikace je možnost zastavení a hlášení zakázat v nastavení Global Events. Pro standardní aplikace doporučujeme hlídání přetečení nebo podtečení zásobníku ponechat zapnutou z důvodu kontroly vyvíjeného programu na chyby tohoto typu.

Trace Overflow

Došlo k zastavení při přetečení trasovací paměti. Obsah trasovací paměti byl naplněn a vykonání každé následující instrukce způsobí ztrátu jedné položky historie zaznamenávané do trasovací paměti. Zastavení a hlášení je možné potlačit nastavením v Global Events.

External Pin Break

Došlo k zastavení emulátoru z důvodu detekce hrany na externím uživatelském pinu (sonda Probe0). Hlášení je možné potlačit v nastavení Global Events.

External Reset Occured

Při běhu programu došlo k resetu procesoru z externího zdroje. Externím zdrojem resetu máme na mysli jakýkoliv reset vyvolaný v emulovaném procesoru jinak než z klávesnice počítače.

4. Překladač jazyka ASSEMBLER

Tato část příručky popisuje programovací jazyk assembler pro procesory řad PIC16C5X a PIC16CXX.

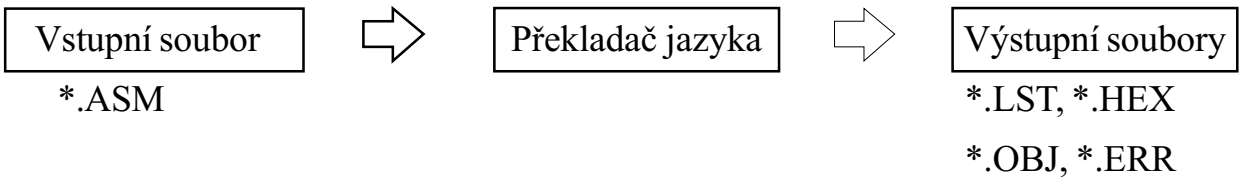
Překladač jazyka je program, který provádí překlad zdrojového textu programu (který je zapsán podle pravidel příslušného programovacího jazyka) do binárního kódu, který je možné zavést do procesoru a spustit.

Vstupem překladače je soubor se zdrojovým textem programu (obvykle soubor s příponou "ASM").

Překlad zdrojového textu programu probíhá ve dvou fázích. První z nich je přípravná (preprocessing), kdy dochází ke kontrole správnosti zápisu programu, je provedeno rozvinutí makroinstrukcí, definování symbolů a vložení souborů. Běh této fáze je možné řídit pomocí direktiv překladače, ať už přímo zapsaných ve zdrojovém textu programu nebo nastavených přes dialogové okno integrovaného prostředí. Po dokončení přípravné fáze probíhá vlastní překlad zdrojového textu programu do binárního kódu. Výstupem překladače jsou následující soubory:

- tzv. "listing", který obsahuje kompletní protokol o průběhu překladu, očíslované řádky zdrojového textu, označené chyby, tabulku použitých symbolů a jejich referencí. Tento soubor má příponou "LST".
- soubor s binárním kódem zvoleného procesoru ve formátu INHEX-8M nebo INHEX-16 (dáno nastavením překladače v dialogu integrovaného prostředí). Soubor má příponu "HEX".
- vyskytnou-li se při překladu chyby, je vytvořen chybový soubor, ve kterém je uvedeno, ve kterém souboru a na jakém řádku byla chyba nalezena a stručný popis chyby. Soubor má příponu "ERR".
- tzv. "object file", který obsahuje informace důležité pro integrované prostředí, jako např. jména a hodnoty použitých symbolů a odkazy na řádky zdrojového textu programu (nutné pro krokování). Soubor má příponu "OBJ".

Blokové schéma funkce překladače:



4.1 Zdrojový text programu

Zdrojový text programu je složen z programových řádků. Programový řádek je obecně složen ze čtyř částí, které jsou navzájem odděleny pomocí mezer nebo tabelátorů. Formát programového řádku je:

kde:

- ***návěští*** - je symbolický odkaz na příslušný řádek programu, který se užívá především k zadání cíle skoků při větvení programu. Návěští není povinné a zpravidla se užívá jen tehdy, je-li potřeba na příslušný řádek provést skok z jiného místa programu. Jméno návěští musí splňovat podmínky kladené na symboly (viz symboly)
- ***příkaz*** - mnemotechnická zkratka instrukce z instrukčního souboru daného procesoru (viz instrukční soubor) nebo direktiva překladače (viz direktivy překladače)
- ***operandy*** - operandy instrukce nebo direktivy překladače, jejich počet závisí na typu instrukce (viz instrukční soubor) nebo direktivy (viz direktivy překladače). Operandy se mezi sebou oddělují čárkou.
- ***komentář*** - jakýkoliv text, který začíná středníkem. Komentáře mohou obsahovat libovolné znaky a jsou překladačem zcela ignorovány.

4.2 Konstanty

Konstantami chápeme veškeré číselné nebo textové informace v programu, které jsou známé již v době překladu programu (tzn. ne obsahy registrů, kterým se přiřazují hodnoty až při běhu programu). Aritmetika překladače interně převádí číselné konstanty na 16 bitová kladná čísla (rozsah 0-65535) a textové konstanty na pole 16 bitových čísel.

4.2.1 Číselné konstanty

Číselné konstanty slouží např. k zadávání operandů instrukcí (čísla registrů, čísla bitů, konstanty). Mohou být zadány pomocí různých číselných soustav nebo pomocí kódů ASCII. K dispozici jsou číselné soustavy: binární (základ 2), osmičková (základ 8), dekadická (základ 10) a hexadecimální (základ 16). Překladači je možno nastavit, zda čísla s nespecifikovanou soustavou má chápat jako čísla dekadická nabo hexadecimální (přednastavená soustava). Pozor na zápisy, které mohou způsobit různou interpretaci čísel v závislosti na přednastavené soustavě (např. 1D je v přednastavené dekadické soustavě 1, zatím co v soustavě hexadecimální 1D = 29 dekadicky). V zápisech soustavy se nerozlišují malá a velká písmena, tj. např. 22h=22H.

Soustava	Zápis	Příklad	Interpretace v nast. soustavě	
			dekadické	hexadecimální
binární	B'<binární číslo>'	B'10'	2	2
	<binární číslo>B	1B	**1**	**1B (27)**
osmičková	O'<osmičkové číslo>'	O'100'	64	40 (64)
	Q'<osmičkové číslo>'	Q'100'	64	40 (64)
	<osmičkové číslo>O	100O	64	40 (64)
	<osmičkové číslo>Q	100Q	64	40 (64)
	\<osmičkové číslo>	\100	64	40 (64)
dekadická	<dekadické číslo>	50	**50**	**50 (80)**
	D'<dekadické číslo>'	D'100'	100	64 (100)
	<dekadické číslo>D	1D	**1**	**1D (29)**
	.<dekadické číslo>	.100	100	64 (100)
hexadecimální	H'<hexadecimální číslo>'	H'10'	16	10 (16)
	<hexadecimální číslo>H	10H	16	10 (16)
	0x<hexadecimální číslo>	0x10	16	10 (16)
ASCII	A'<ASCII znak>'	A'd'	100	64 (100)
	'<ASCII znak>'	'd'	100	64 (100)

4.2.2 Textové konstanty

Textové konstanty představují speciální typ konstanty, kterou si je možné představit jako pole o několika prvcích, kde každý prvek pole je číselná konstanta. Textové konstanty nelze použít všude, neboť jejich použití je povoleno pouze v následujících případech:

- *definice tabulky* (direktiva TABLE)
- *definice obsahu programové paměti* (direktivy DB, DW)
- *zadání konstantního pole bajtů* (direktiva CONST BYTE[])

Textové konstanty se skládají z ASCII znaků uzavřených v uvozovkách. (Př. “Toto je text”). Potřebujeme-li vložit do řetězce netisknutelné znaky nebo jiné číselné hodnoty, je tak možno učinit pomocí hexadecimálních kódů znaků ve formátu \x<hex. číslo> (Př. “text1\x20text2” vloží ASCII znak s hexadecimálním kódem 20 (mezera) mezi text1 a text2).

4.3 Výrazy

Výrazem rozumíme matematické nebo logické operace prováděné mezi konstantami. Např. “3+4” je výraz, který představuje matematický součet konstant 3 a 4, jehož výsledkem je konstanta 7. Pro vytváření aritmetických a logických výrazů je možno využít operace, které shrnuje tabulka. Je-li výraz složen z více matematických

Priorita	Operátor	Popis operace	Příklad
1.	()	Závorky	(7+8)/3
2.	! ~ + -	Logická negace Jedničkový doplněk čísla Unární plus (kladné číslo) Unární mínus (záporné číslo)	!(p1<10) ~8 +3 -4
3.	* / %	Násobení Celočíselné dělení Zbytek po celočíselném dělení	2*5 5/3 5%3
4.	+ -	Binární plus (sčítání) Binární mínus (odčítání)	2+2 6-2
5.	<< >>	Rotace vlevo Rotace vpravo	5<<2 4>>1
6. (L)	< <= > >=	Menší než Menší než nebo rovno Větší než Větší než nebo rovno	x<y x<=y x>y x>=y
7. (L)	== !=	Je rovno Není rovno	x==y x!=y
8.	&	Binární logický součin (AND)	x&y
9.	^	Binární nonekvivalence (XOR)	x^y
10.	|	Binární logický součet (OR)	x|y
11. (L)	&&	Logický součin (AND)	x&&y
12. (L)	||	Logický součet (OR)	x&&y

nebo logických operací, pořadí prováděných operací se určí podle priorit. Nejdříve se vyhodnocují závorky, pak operace s prioritou 2, následují operace s prioritou 3, atd.. Má-li více operací stejnou prioritu, pak se výraz vyhodnocuje zleva doprava.V případě logických operací (v tabulce označeny písmenem "L" v kolonce priorita) je výsledkem vyhodnocení výrazu buď hodnota 0 (výraz je nepravdivý, např. 1>2), nebo hodnota 1 (výraz je pravdivý, např 2!=3).

4.4 Symboly

Pod pojmem symboly v programech rozumíme:

- ***symbolická jména konstant*** (definovaná direktivami EQU a SET),
- ***jména datových typů*** (definovaná direktivami BIT, BYTE, CONST BYTE),
- ***návěští*** (viz programový řádek),
- ***jména makroinstrukcí*** (viz makroinstrukce).

Každý symbol má své jméno a vlastní definici symbolu, jejíž zápis závisí na tom, o jaký typ symbolu se jedná.

Jméno symbolu musí být v celém programu jedinečné a musí splňovat následující podmínky, aby bylo správně interpretováno:

1. musí začínat písmenem ‘A’-‘Z’, ‘a’-‘z’,
2. smí být složeno pouze z písmen ‘A’-‘Z’, ‘a’-‘z’, čísel ‘0’-‘9’ a podtržítka ‘_’, tedy nesmí obsahovat oddělovače (mezera, tabulátor).

4.4.1 Symbolická jména konstant

Chceme-li v programu nadefinovat symbolické jméno pro konstantu, můžeme použít direktivy překladače EQU nebo SET. Obě direktivy mají stejný zápis a symbolickému jménu přiřazují hodnotu (konstantu). Místo hodnoty lze uvést i výraz, a pak se symbolickému jménu přiřadí hodnota vzniklá vyhodnocením výrazu.

```
<jméno symbolu>  EQU  <hodnota>
<jméno symbolu>  SET  <hodnota>
```

```
Př: Data1   SET     13h
    Data2   EQU     10h
    Data3   EQU     (1+Data1)*Data2
```

Rozdíl mezi direktivou EQU a SET spočívá pouze v možnosti změnit hodnotu symbolu. Je-li hodnota symbolu definovaná direktivou SET, je možné jí libovolně přiřadit novou hodnotu buď opět direktivou SET nebo direktivou EQU. Je-li hodnota symbolu definovaná pomocí direktivy EQU, při pokusu u předefinování hodnoty jak direktivou EQU, tak i direktivou SET překladač ohlásí chybu.

4.4.2 Datové typy

Datové typy byly do programovacího jazyka zavedeny především pro ulehčení práce s registry procesoru. Jejich základním rysem je, že umožňují zcela jednoznačně adresovat bajty nebo bity. Definice datového typu je složena ze jména symbolu (splňujícího podmínky kladané na jméno symbolu), klíčového slova označujícího datový typ a buď informací o umístění v paměti (adresa, bit, banka), nebo hodnoty.

Kromě základních typů je možné definovat i rozšířený typ - pole. Pole je tvořeno prvky stejného typu a jeho velikost je nutné u pole bitů a pole bajtů zadat. Jednotlivé prvky pole se číslují od 0 a jsou přístupné pomocí indexů, které se zapisují bezprostředně za jméno symbolu do hranatých závorek. Jméno symbolu bez indexu označuje první prvek pole stejně tak jako jméno symbolu s indexem 0.

Typ bajt je předurčen pro použití v instrukcích, které pracují s registry, typ bit je určen pro bitově orientované instrukce a typ konstanta je vhodný pro instrukce vyžadující konstanty.

Překladač jazyka rozeznává a pracuje s následujícími datovými typy:

- ***Byte*** - jeden registr (bajt) v datové paměti. Definice se skládá z určení adresy a banky registru (viz příklad).

```
<jméno symbolu> BYTE @<adresa>,<banka>
```

- ***Bit*** - jeden bit registru v datové paměti. Definice se skládá z určení adresy a banky registru a čísla bitu daného registru (0=LSB, 7=MSB). Není-li číslo bitu uvedeno, předpokládá se 0 (viz příklad)

```
<jméno symbolu> BIT @<adresa>,<bit>,<banka>
```

- ***Byte Array*** - pole registrů (bajtů) v datové paměti. Definice se skládá z rozměru pole (udávajícího kolika bajty je pole tvořeno) a adresy a banky, které udávají adresu prvního bajtu pole. Následující prvky pole (bajty) jsou alokovány směrem k vyšším adresám. (viz příklad).

```
<jméno symbolu> BYTE[rozměr] @<adresa>,<banka>
```

- ***Bit Array*** - pole bitů v datové paměti. Definice se skládá z rozměru udávajícího kolika prvky (bity) je pole tvořeno, adresy a banky registru a bitu daného registru. Všechny tyto údaje přesně adresují první prvek pole (bit). Další prvky pole jsou alokovány směrem k významnějším bitům a směrem k vyšším adresám. Není-li číslo bitu uvedeno, předpoládá se 0 (viz příklad).

```
<jméno symbolu> BIT[rozměr] @<adresa>,<bit>,<banka>
```

- ***Const Byte*** - 8 bitová konstanta (bajt), ekvivalent symbolů definovaných pomocí direktivy EQU. Definice se skládá z hodnoty, která se přiřadí symbolu. Místo hodnoty lze uvést výraz, pak se symbolu přiřadí hodnota odpovídající výsledku vyhodnocení výrazu (viz příklad).

```
<jméno symbolu> CONST BYTE = <hodnota>
```

- ***Const Byte Array*** - pole 8 bitových konstant (bajtů). Definice se v tomto případě skládá z rozměru pole, a hodnot oddělených čárkami. Pokud není rozměr pole číselně uveden, automaticky se vypočítá podle počtu uvedených hodnot. Hodnoty lze zadat jako číselnou konstantu, výraz, textovou konstantu nebo i již definované pole konstantních bajtů.

```
<jméno symbolu> CONST BYTE[rozměr] = <hodnota> {,<hodnota>…}
```

Následující příklad demonstruje zápisy a umístění různých datových typů. Nejsou udány banky, proto se předpoládá přednastavená banka 0.

```
pocet   BYTE     @8
stav    BIT      @9
pole    BYTE[3]  @.11
bity    BIT[10]  @9,2
cislo   CONST BYTE = 1+2*3
data    CONST BYTE[] = "text", cislo, @stav, 22h, 3*5
```

Adresa	7	6	5	4	3	2	1	0
08h	pocet							
09h	bity[5]	bity[4]	bity[3]	bity[2]	bity[1]	bity[0]		stav
0Ah					bity[9]	bity[8]	bity[7]	bity[6]
0Bh	pole[0]							
0Ch	pole[1]							
0Dh	pole[2]							

Konstantní symboly nejsou přímo v datové paměti uloženy, jsou známy pouze překladači, který je v případě použití v programu nahradí jejich numerickou hodnotou. Hodnoty konstanty a pole konstant jsou: cislo=7, data=data[0]=116 (A"t"), data[1]=101 (A"e"), data[2]=120 (A"x"), data[3]=116 (A"t"), data[4]=7, data[5]=9 (adresa symbolu "stav"), data[6]=34 (22h), data[7]=15.

Upozornění: Typy bajt resp. bit a jejich pole představují obsah registru, resp. hodnotu bitu, které nejsou v době překladu ještě známy (např. instrukce "CLRF pocet" provede vynulování obsahu registru 8, nikoliv vynulování symbolu "pocet"). Proto tyto datové typy je možné použít pouze v kombinaci s vhodnými instrukcemi, pro které jsou určené. Pokud chceme získat adresu typu bajt bit nebo jejich pole, která je pak chápána jako obecná konstanta, je nutné napsat před jméno symbolu znak adresy "@".

4.4.3 Návěští

Návěští se zadává jako první položka v příkazovém řádku (viz příkazový řádek). Že se jedná o definici návěští může být zvýrazněno tím, že se za jméno návěští zapíše dvojtečka, která se však do jména návěští nikterak nepromítne, protože je ignorována.

```
Př: ...
    cti1        MOVLW   10h ;definuje návěští cti1
    ...
```

Pokud je například potřeba definovat pro jednu adresu více návěští, je možné použít ještě jeden způsob definice - tzv. samostatné návěští. Definujeme-li samostatné návěští, nesmí být na řádku uveden příkaz ani direktiva. Komentář na řádku být může. Navíc musí samostatné návěští začínat v prvním sloupci zdrojového textu, nebo musí být zakončeno dvojtečkou.

Př: ...

```
    labe1:
    label2  CLRF        8h  ;definuje dvě návěští
```

4.5 Instrukční soubor procesorů PIC

Instrukční soubor je dán výrobcem procesoru, tj. v našem případě firmou Microchip. Každá rodina procesorů má svůj instrukční soubor, který je zpravidla vytvořen pomocí zkratek operací, které daný procesor umožňuje provádět. Zkratky se označují jako mnemotechnické (symbolické) názvy instrukcí.

Překladač umožňuje překládat programy pro rodiny procesorů 16C5X a 16CXX. Obě rodiny procesorů jsou si velice podobné a mají až na malé odchylky u některých typů stejnou instrukční sadu (mnemotechnické názvy instrukcí). Binární kódy instrukcí obou rodin jsou však rozdílné, neboť rodina 16C5X používá 12 bitů široké instrukce, a 16CXX používá 14 bitů široké instrukce.

Obecně lze instrukční soubor obou rodin rozdělit na:

- ***bajtově orientované instrukce***
- ***bitově orientované instrukce***
- ***instrukce pracující s konstantou***
- ***řídící instrukce***

4.5.1 Bajtově orientované instrukce

Bajtově orientované instrukce pracují s datovým registrem. Umožňují provádět přesuny dat mezi datovým registrem a pomocným registrem W, modifikaci hodnoty registru (přičtení 1, odečtení 1, rotace, ...) a aritmetické a logické operace s hodnotou v registru a ve W.

Operandem instrukce je číslo registru, které může být zadáno přímo konstantou nebo pomocí datového typu bajt.

Většina instrukcí této kategorie vyžaduje ještě druhý operand, udávající kam má být uložen výsledek operace. Je-li zadána hodnota 0, je výsledek uložen do pomocného registru W, je-li hodnota 1, je výsledek uložen do registru, který je zadán prvním operandem. Pro tyto účely má překladač předdefinovány symboly W a F, kde W=0 a

F=1. Pokud není uvedeno, kam má být výsledek uložen, bude uložen do registru, který je zadán prvním operandem.

```
Př: CLRF    8h      ; vynulování obsahu registru 8h
    ; zvětší hodnotu reg. 10 o 1
    INCF    10h,1
    INCF    10h,F
    INCF    10h
    ; hodnotu reg. 10 zvětšenou o 1 ulož do pom. reg. W
    INCF    10h,0
    INCF    10h,W
    ;použijeme-li definice z příkladu datových typů, pak
    MOVWF pole[1]   ;zkopíruje obsah W do reg. 12
```

4.5.2 Bitově orientované instrukce

Bitově orinetované instrukce pracují s bity v datových registrech procesoru. Umožňují nastavit hodnotu bitu do logické jedničky nebo do logické nuly, nebo testovat logickou hodnotu daného bitu a podle výsledku větvit program.

Operandem instrukcí tohoto typu může být datový typ bit, příp. jeden prvek pole bitů. Tyto typy obsahují kompletní určení bitu.

Pokud nepoužijeme zadání operandu pomocí typu bit, musíme použít operandy dva. První operand udává číslo registru a druhý operand udává číslo bitu v registru (0-7).

```
Př: BCF 8,2     ; vynuluje bit 2 registru 8
    ;použijeme-li definice z příkladu datových typů, pak
    BSF bity[2] ; nastaví bit 4 registru 9
```

4.5.3 Instrukce pracující s konstantou

Instrukce pracující s konstantou slouží k modifikaci obsahu pomocného registru W a k provádění větvení programu a volání podprogramů. Patří sem i instrukce skoku (GOTO), volání (CALL) a návratu z podprogramu (RETLW).

Operandem tohoto typu instrukcí je konstanta definovaná libovolným způsobem.

```
Př: MOVLW   10h ; zápis hodnoty 10h do pom. registru W
```

4.5.4 Řídící instrukce

Řídící instrukce slouží především k řízení činnosti procesoru (přechod do režimu "sleep", vynulování čítače "Watchdog"). Řídící instrukce kromě instrukce TRIS nemají operand, neboť nepotřebují žádné dodatečné informace. Instrukce TRIS slouží k nastavení jednotlivých pinů portů do vstupního nebo výstupního módu a parametrem je číslo registru portu.

4.5.5 Přehled instrukcí rodiny PIC 16C5X

ADDWF - ADD W and F

Zápis: ADDWF f,d
Operace: (W + f) → d
Popis: Sečte obsah registrů f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: C, DC, Z

ANDLW - AND Literal and W

Zápis: ANDLW k
Operace: (k & W) → W
Popis: Provede logický součin obsahu registru W s konstantou k, výsledek uloží do registru W.
Cyklů: 1 Ovlivňuje: Z

ANDWF - AND W with F

Zápis: ANDWF f,d
Operace: (W & f) → d
Popis: Provede logický součin obsahu registru f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

BCF - Bit Clear F

Zápis: BCF f,b
Operace: 0 → f(b)
Popis: Vynuluje bit b v registru f.
Cyklů: 1 Ovlivňuje: -

BSF - Bit Set F

Zápis: BSF f,b
Operace: 1 → f(b)
Popis: Nastaví do log. 1 bit b v registru f.
Cyklů: 1 Ovlivňuje: -

BTFSC - Bit Test F, Skip if Clear

Zápis: BTFSC f,b
Operace: skok, je-li f(b)=0
Popis: Je-li bit b v registru f v log. 0, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Cyklů: 1(2- při skoku) Ovlivňuje: -

BTFSS - Bit Test F, Skip if Set

Zápis: BTFSS f,b
Operace: skok, je-li f(b)=1
Popis: Je-li bit b v registru f nastaven do log. 1, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Cyklů: 1(2- při skoku) Ovlivňuje: -

CALL - subroutine CALL

Zápis: CALL k
Operace: PC+1 → TOS; k → PC<7:0>; 0 → PC<8>; PA2,PA1,PA0 → PC<11:9>
Popis: Návratovou adresu (PC+1) uloží do zásobníku, konstanta k se uloží na spodních 8 bitů PC, 9. bit PC se vynuluje, zbývající 3 bity PC se doplní z bitů PA2,PA1 a PA0. Program pokračuje podprogramem na adrese PC.
Cyklů: 2 Ovlivňuje: -

CLRF - CLeaR F register

Zápis: CLRF f,d
Operace: 00h → f, 00h → d
Popis: Vynuluje obsah registru f.
Cyklů: 1 Ovlivňuje: Z

CLRW - CLeaR W register

Zápis: CLRW
Operace: 00h → W
Popis: Vynuluje obsah registru W.
Cyklů: 1 Ovlivňuje: Z

CLRWDT - CLeaR WatchDog Timer

Zápis: CLRWDT
Operace: 00h → WDT, 0 → WDT předdělič
Popis: Nuluje čítač 'Watchdog' a jeho předděličku.
Cyklů: 1 Ovlivňuje: 1 → TO, 1 → PD

COMF - COMplement F

Zápis: COMF f,d
Operace: /f → d
Popis: Zamění jedničky a nuly v obsahu registru f (jedničkový doplněk čísla) a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

DECF - DECrement F

Zápis: DECF f,d
Operace: (f - 1) → d
Popis: Odečte jedničku od obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

DECFSZ - DECrement F Skip if Zero

Zápis: DECFSZ f,d
Operace: (f - 1) → d; skok, je-li výsledek 0
Popis: Odečte jedničku od obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Cyklů: 1(2 - je-li skok) Ovlivňuje: Z

GOTO - GO TO address (nepodmíněný skok)

Zápis: GOTO k
Operace: k → PC<8:0>; PA2,PA1,PA0 → PC<11:9>
Popis: Konstanta k (bere se z ní 9 bitů !!!) se uloží na spodních 9 bitů PC, zbývající 3 bity PC se doplní z bitů PA2,PA1 a PA0 v registru STATUS procesoru. Program pokračuje kódem na adrese PC.
Cyklů: 2 Ovlivňuje: -

INCF - INCrement F

Zápis: INCF f,d
Operace: (f + 1) → d
Popis: Přičte jedničku k obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

INCFSZ - INCrement F Skip if Zero

Zápis: INCFSZ f,d
Operace: (f + 1) → d; skok, je-li výsledek 0
Popis: Přičte jedničku k obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Cyklů: 1(2 - je-li skok) Ovlivňuje: Z

IORLW - Inclusive OR Literal with F

Zápis: IORLW k
Operace: (W | k) → W
Popis: Provede logický součet (OR) obsahu registru W s konstantou k, výsledek uloží do registru W.
Cyklů: 1 Ovlivňuje: Z

IORWF - Inclusive OR W with F

Zápis: IORWF f,d
Operace: (W | f) → d
Popis: Provede logický součet (OR) obsahu registrů f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

MOVF - MOVe F

Zápis: MOVF f,d
Operace: (f) → d
Popis: Obsah registru f přesune do registru W (je-li d=0) nebo zpět do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

MOVWF - MOVe W to F

Zápis: MOVWF f,d
Operace: W → d
Popis: Obsah registru W přesune do registru f
Cyklů: 1 Ovlivňuje: -

NOP - No OPeration

Zápis: NOP
Operace: neprovede nic
Popis: neprovede nic
Cyklů: 1 Ovlivňuje: -

OPTION - load OPTION register

Zápis: OPTION
Operace: W → OPTION
Popis: Obsah registru W přesune do registru OPTION
Cyklů: 1 Ovlivňuje: -

RETLW - RETurn Literal to W

Zápis: RETLW k
Operace: k → W; TOS → PC;
Popis: Návrat z podprogramu. Naplní PC ze zásobníku a registr W naplní konstantou k.
Cyklů: 1 Ovlivňuje: -

RLF - Rotate Left F through carry

Zápis: RLF f,d
Operace: f<n> → d<n+1>; f<7> → C; C → d<0>
Popis: Rotuje obsah registru f doleva přes bit C (carry), výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: C

RRF - Rotate Right F through carry

Zápis: RRF f,d
Operace: f<n> → d<n-1>; f<0> → C; C → d<1>
Popis: Rotuje obsah registru f doprava přes bit C (carry), výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: C

SLEEP - SLEEP

Zápis: SLEEP
Operace: 0 → PD, 1 → TO, 00h → WDT, 0 → WDT předdělič
Popis: Vynuluje 'power-down' bit PD, nastaví 'time-out' bit TO, vynuluje čítač 'Watchdog' a jeho předděličku. Procesor přejde do stavu SLEEP, oscilátor je vypnut.
Cyklů: 1 Ovlivňuje: TO, PD

SUBWF - SUBtract W from F

Zápis: SUBWF f,d
Operace: (f - W) → d
Popis: Odečte obsah registru W od obsahu registru f, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: C, DC, Z

SWAPF - SWAP F

Zápis: SWAPF f,d
Operace: f<0:3> → d<4:7>; f<4:7> → d<0:3>
Popis: Zamění spodní a horní 4 bity (nibble) obsahu registru f, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: -

TRIS - load TRIS register

Zápis: TRIS f
Operace: W → TRIS registr f
Popis: Uloží do řídícího registru portů TRIS (f=5, 6, nebo 7) obsah registru W,
Cyklů: 1 Ovlivňuje: -

XORLW - Exclusive OR Literal with W

Zápis: XORLW k
Operace: (W xor k) → W
Popis: Provede nonekvivalneci (XOR) obsahu registru W s konstantou k, výsledek uloží do registru W.
Cyklů: 1 Ovlivňuje: Z

XORWF - Exclusive OR W with f

Zápis: XORWF f,d
Operace: (W xor f) → d
Popis: Provede nonekvivalenci (XOR) obsahu registrů f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Cyklů: 1 Ovlivňuje: Z

4.5.6 Instrukční soubor rodiny PIC 16CXX

Jelikož instrukční soubor rodiny PIC 16CXX je nadmnožinou instrukčního souboru rodiny PIC 16C5X, zaměříme se v popisu pouze na ty instrukce, jejichž způsob provádění je jiný než bylo popsáno nebo které se v instrukčním souboru PIC 16C5X nevyskytují.

ADDLW - ADD Literal to W

Zápis: ADDLW k
Operace: (W + k) → W
Popis: Provede aritmetický součet obsahu registru W s konstantou k, výsledek uloží do registru W.
Cyklů: 1 Ovlivňuje: C, DC, Z

CALL - subroutine CALL

Zápis: CALL k
Operace: PC+1 → TOS; k → PC<10:0>; PCLATH<4:3> → PC<12:11>
Popis: Návratovou adresu (PC+1) uloží do zásobníku, konstanta k (vezme se z ní 11 bitů !!!) se uloží na spodních 11 bitů PC, zbývající bity PC se doplní z registru PCLATH (f3). Program pokračuje podprogramem na adrese PC.
Cyklů: 2 Ovlivňuje: -

GOTO - GO TO address (nepodmíněný skok)

Zápis: GOTO k
Operace: k → PC<10:0>; PCLATH<4:3> → PC<12:11>
Popis: Konstanta k (bere se z ní 11 bitů !!!) se uloží na spodních 11 bitů PC, zbývající bity PC se doplní z registru PCLATH (f3). Program pokračuje kódem na adrese PC.
Cyklů: 2 Ovlivňuje: -

RETFIE - RETurn From IntErrupt

Zápis: RETFIE
Operace: TOS → PC; 1 → GIE
Popis: Návrat z přerušení. Naplní hodnotu PC ze zásobníku a povolí přerušení nastavením bitu GIE (Global Interrupt Enable) do log. 1.
Cyklů: 2 Ovlivňuje: -

RETURN - RETURN from subroutine

Zápis: RETURN
Operace: TOS → PC;
Popis: Návrat z podprogramu. Naplní hodnotu PC ze zásobníku.
Cyklů: 2 Ovlivňuje: -

SUBLW - SUBtract Literal and W

Zápis: SUBLW k
Operace: (k - W) → W
Popis: Odečte obsah registru W od konstanty k, výsledek uloží do registru W.
Cyklů: 1 Ovlivňuje: C, DC, Z

4.6 Direktivy překladače

Direktivy překladače slouží k nastavení parametrů překladu a nejsou překládány do binárního kódu procesoru. Výjimku tvoří pouze ditektivy DB, DW a TABLE, které ukládají data do programové paměti procesoru. Část direktiv (např. zapnutí rozlišování velkých a malých písmen, nastavení délky stránky v listingu, potlačení vytváření některých souborů, …) se nastavuje v dialogovém okně integrovaného prostředí. Zbývající direktivy (např. vložení souboru, nastavení adresy pro překlad, podmíněný překlad, definování makroinstrukcí, …) se zapisují přímo do zdrojového textu programu. Těmito direktivami se zabývá tato část příručky. U většiny direktiv je možné provést zápis dvěma způsoby (viz zápisy jednotlivých direktiv). Při použití varianty používající znak '#' (ekvivalent zápisu direktiv překladače používaný ve vyšších programovacích jazycích - např. C), nelze na daném řádku definovat návěští.

4.6.1 Přehled direktiv

BANK	přednastavení banky registrů v datové paměti
BIT	definice datového typu bit nebo pole bitů
BYTE	definice datového typu bajt nebo pole bajtů
CONST BYTE	definice konstanty nebo pole konstant
DB	uložení dat do programové paměti
DW	uložení dat do programové paměti
ELSE	přepínač podmíněného překladu
END	konec programu
ENDIF	konec podmíněného překladu
ENDM	konec definice makroinstrukce
EQU	přiřazení hodnoty symbolu
IF	začátek podmíněného překladu
INCLUDE	vložení souboru
LOCAL	definice lokálního návěští v makroinstrukci
MACRO	začátek definice makroinstrukce
ORG	nastavení adresy překladu
PRAGMA	nastavení parametrů překladače (potlačení generování varovných chybových hlášení - "warnings")
SET	přiřazení hodnoty symbolu
TABLE	definice tabulky hodnot

4.6.2 Direktiva BANK

Direktiva BANK slouží k přednastavení čísla datové banky. Pokud definujeme datové typy (např. bajt nebo bit) a nezadáme číslo banky, bude použita přednastavená banka. Nepoužije-li se direktiva BANK, je přednastavená datová banka 0.

Zápis: `#BANK <číslo banky>`

Př:
```
#BANK 0  ; přednastaví banku 0
```

4.6.3 Direktiva BIT

Direktiva BIT slouží k definici datového typu bit nebo pole bitů. Definice se skládá ze jména symbolu, klíčového slova BIT (v případě pole bitů BIT[]) a jednoznačného určení bitu (adresa registru, číslo bitu, číslo banky), které v případě definice pole bitů určuje polohu prvního prvku pole. Pokud není uvedena banka, předpokládá se banka přednastavená direktivou BANK. Pokud není uvedeno číslo bitu, předpokládá se bit 0 (LSB). Pokud uvádíme číslo banky, musíme zadat i číslo bitu. Definujeme-li pole bitů, musíme zadat rozměr pole, který říká, kolik má pole

prvků (bitů). Jednotlivé prvky pole (bity) se rozlišují indexem zapsaným v hranatých závorkách za jménem pole. Prvky pole se číslují od nuly. Pokud není index uveden, předpokládá se prvek s indexem 0. Jednotlivé prvky pole (bity) se alokují od zadané pozice směrem k MSB a vyšším adresám (viz datové typy).

```
Zápis:  <symbol> BIT @<adresa>, {<bit>, {<banka>}}
        <symbol> BIT[<rozměr>] @<adresa>,{<bit>,{<banka>}}

Př:     bitik   BIT @8,1        ; bit 1 v registru 8
        pole1   BIT[3] @8,2 ; pole 3 bitů
```

4.6.4 Direktiva BYTE

Direktiva BYTE slouží k definici datového typu bajt nebo pole bajtů. Definice se skládá ze jména symbolu, klíčového slova BYTE (v případě pole bajtů BYTE[]) a jednoznačného určení bajtu (adresa registru, číslo banky), které v případě definice pole bitů určuje polohu prvního prvku pole (bajtu). Pokud není uvedena banka, předpokládá se banka přednastavená direktivou BANK. Definujeme-li pole bajtů, musíme zadat rozměr pole, který říká, kolik má pole prvků (bajtů). Jednotlivé prvky pole (bajty) se rozlišují indexem zapsaným v hranatých závorkách za jménem pole. Prvky pole se číslují od nuly. Pokud není index uveden, předpokládá se prvek s indexem 0. Jednotlivé prvky pole (bajty) se alokují od zadané adresy směrem k vyšším adresám (viz datové typy).

```
Zápis:  <symbol> BYTE @<adresa>, {<banka>}
        <symbol> BYTE[<rozměr>] {@<adresa>, {<banka>}}

Př:     bajt1   BYTE    @9
        pole2   BYTE[2] @.10    ; pole 2 bajtů
```

4.6.5 Direktiva CONST BYTE

Direktiva CONST BYTE slouží k definici konstanty (podobně jako direktivy EQU a SET) a pole konstant (CONST BYTE[]). Definice se skládá ze jména symbolu, klíčového slova CONST BYTE (v případě pole bajtů CONST BYTE[]) a hodnoty. Hodnota může být zadána jako číselná konstanta, textová konstanta nebo výraz. Definujeme-li pole konstant, nemusíme zadat rozměr pole, neboť ten se automaticky zjistí podle počtu zadaných hodnot. Jednotlivé prvky pole (bajty) se rozlišují indexem zapsaným v hranatých závorkách za jménem pole. Prvky pole se číslují od nuly. Pokud není index uveden, předpokládá se prvek s indexem 0.

```
Zápis:  <symbol> CONST BYTE = <hodnota>
        <symbol> CONST BYTE[{<rozměr>}] = <hodnoty>
```

```
Př:     konst1  CONST BYTE = 10h
        konst2  CONST BYTE[] = "abcd", 0x20, konst1
```

4.6.6 Direktiva DB

Direktiva DB slouží k uložení konstanty o rozsahu 8 bitů (bajt) do programové paměti na aktuální adresu. Hodnota může být zadána číselnou konstantou, textovou konstantou, polem konstant i výrazem.

```
Zápis:  {<návěští>} DB <hodnota>, {<hodnota>, ...}

Př:     ; uložení textu do programové paměti na adr. 100h
        ORG 100h
        DB  "verze programu 1.0"
```

4.6.7 Direktiva DW

Direktiva DW slouží k uložení konstanty o rozsahu maximálně šířky instrukce zvolené rodiny procesorů do programové paměti na aktuální adresu. Hodnota může být zadána číselnou konstantou, textovou konstantou, polem konstant i výrazem.

```
Zápis:  {<návěští>} DW <hodnota>, {<hodnota>, ...}

Př:     ; zápis konfiguračních pojistek 16C84
        ORG 2007h
        DW  0xXXX
```

4.6.8 Direktiva END

Direktiva END označuje konec programu. Veškeré řádky za direktivou END jsou překladačem ignorovány.

```
Zápis1: {<návěští>} END
Zápis2: #END

Př:     END
```

4.6.9 Direktiva EQU

Direktiva EQU slouží k definici hodnoty symbolu. Hodnota může být zadána jako číselná konstanta nebo výraz. Pokud již symbol stejného jména existuje, překladač ohlásí chybu.

```
Zápis:  <symbol> EQU <hodnota>

Př:     konst1   EQU  20+6*2
        konst2   EQU  konst1*3
```

4.6.10 Direktivy IF - ELSE - ENDIF

Direktivy IF - ELSE - ENDIF slouží k definování bloku podmíněného překladu. Definice bloku začíná direktivou IF, za kterou následuje výraz, který je možné vyhodnotit při překladu (musí tedy být složen pouze z konstant nebo symbolů, kterým byla přiřazena hodnota). Vyhodnocením výrazu mohou nastat dvě možnosti:

1. Výraz byl vyhodnocen jako pravdivý (výsledkem vyhodnocení bylo číslo různé od nuly), pak se překládá pouze <kód TRUE>, zatím co <kód FALSE> je překladačem ignorován.
2. Výraz byl vyhodnocen jako nepravdivý (výsledkem vyhodnocení byla nula), pak se překládá pouze <kód FALSE>, zatím co <kód TRUE> je překladačem ignorován.

Direktiva ELSE odděluje <kód TRUE> a <kód FALSE>. Direktiva ENDIF ukončuje definici bloku podmíněného překladu.

```
Zápis:  #IF <výraz>        Nebo:   IF <výraz>
          <kód TRUE>                   <kód TRUE>
        #ELSE                      ELSE
          <kód FALSE>                  <kód FALSE>
        #ENDIF                     ENDIF
```

4.6.11 Direktiva INCLUDE

Direktiva INCLUDE slouží k vložení obsahu souboru do zdrojového textu programu. Používá se pro vložení definic symbolů, makroinstrukcí, knihoven apod.. Operandem direktivy je jméno souboru, který má být vložen. Jméno souboru musí být uzavřeno v uvozovkách. Je povoleno maximálně devítinásobné vnoření.

```
Zápis1: {<návěští>} INCLUDE "<jméno souboru>"
Zápis2: #INCLUDE "<jméno souboru>"

Př:     INCLUDE "PICREG.EQU"
```

4.6.12 Direktivy MACRO - LOCAL -ENDM

Direktivy MACRO - LOCAL - ENDM slouží k definování makroinstrukcí. Makroinstrukce je blok instrukcí, které zpravidla tvoří nějaký funkční celek (např.

generátor zpoždění) nebo často se opakující úsek kódu. Makroinstrukce začíná definicí, ve které je uvedeno jméno makroinstrukce a její argumenty, které slouží pro předávání hodnot instrukcím tvořící tělo makroinstrukce. Argumentů může být maximálně 10. Uvnitř makra může být voláno jiné makro, ovšem počet vzájemných vnoření maker je omezeno na 5.

Zápis1:
```
#MACRO <jméno makra> {<argumenty>}
{#LOCAL <lok.n.>}
<instrukce>
#ENDM
```

Zápis2:
```
<jméno makra> MACRO {<argumenty>}
{LOCAL <lok.n.>}
<instrukce>
ENDM
```

Příkladem definice makroinstrukce může být makroinstrukce moje1, která má dva argumenty pojmenované par_a, par_b a která provádí uložení konstanty (reprezentovaná parametrem par_a) do datového registru (reprezentovaného par_b).

Př:
```
#MACRO moje1 par_a, par_b
  Movlw par_a
  Movwf par_b,F
#ENDM
```

Volání makroinstrukce má tvar:

Zápis: `{<návěští>} <jméno makra> {<argumenty>}`

Pak kód:

```
moje1 11h, 10h
```

Bude rozvinut do:

```
Movlw 11h
Movwf 10h
```

Překladač umožňuje automatické použití definované knihovny maker při překladu zdrojového textu. V adresáři, kde se nachází soubory vývojového prostředí je soubor STDPIC.MCR, který obsahuje nadefinované základní makroinstrukce pro zajištění kompatibility překladače s jinými produkty. Tento soubor je vždy před provedením kompilace programu překladačem načten, a tedy všechny v něm definované makroinstrukce jsou v době překladu známé a použitelné ve Vašich programech. Uživatelé si mohou do souboru STDPIC.MCR dopsat své vlastní makroinstrukce, ale raději doporučujeme vytvořit zvláštní soubor s makroinstrukcemi a ten vložit do překladače pomocí direktivy INCLUDE.

4.6.13 Direktiva ORG

Direktiva ORG slouží k nastavení adresy programové paměti, od které budou překládány následné příkazy. Pomocí této direktivy je možné nastavit překlad na libovolnou adresu programové paměti. U procesoru 16C84 je tak možné např. přímo zapisovat data do paměťového prostoru datové EEPROM, která je mapována do adresového prostoru od adresy 2100h

Zápis: {<návěští>} ORG <hodnota>

```
Př:     ORG 1FFh ; resetovací vektor 16C54
        Goto start

        ORG 0
start   ....
```

4.6.14 Direktiva PRAGMA

Direktiva PRAGMA slouží k nastavení různých parametrů překladače jazyka assembler. V současné době je implementována pouze možnost potlačení generování varování překladače (warnings) při kompilaci zdrojového textu programu. Tato direktiva nemá vliv na hlášení varování překladače při nenalezení knihovny maker, toto varování je generováno vždy.

Zápis: #pragma warn-

4.6.15 Direktiva SET

Direktiva SET slouží (podobně jako direktiva EQU) k definici hodnoty symbolu. Hodnota může být zadána jako konstanta nebo výraz. Hodnotu symbolu definovanou direktivou SET lze změnit další direktivou SET nebo EQU.

Zápis: <symbol> SET <hodnota>

```
Př:     konst1  SET  20+6*2
        konst1  SET  konst1*3 ; předefinování hodnoty
```

4.6.16 Direktiva TABLE

Direktiva TABLE slouží k definici tabulky hodnot v programové paměti. Při překladu se direktiva TABLE rozvine pomocí instrukcí RETLW, jejichž návratové kódy vytvoří požadovanou tabulku. Hodnoty mohou být zadány jako číselné konstanty,

textové konstanty nebo výrazy.

```
Zápis1: {<návěští>} TABLE <hodnoty>
Zápis2: #TABLE <hodnoty>

Př:     ADDWF 2,1
        TABLE 0xB7, 0x83, 0x11, 0x50
```

kód bude rozvinut do posloupnosti instrukcí:

```
        ADDWF 2,1
        RETLW 0xB7
        RETLW 0x83
        RETLW 0x11
        RETLW 0x50
```

4.7 Chybová hlášení překladače a varování

Can not open input file "<jméno souboru>"

Soubor <jméno souboru> nebyl nalezen. Přesvědčte se, zda takový soubor existuje.

Can not open include file "<jméno souboru>"

Direktivou INCLUDE je vložen soubor <jméno souboru>, který nelze otevřít. Přesvědčte se, zda při zadání jména souboru nedošlo k překlepu, zda je uvedena správná cesta k souboru (relativně vzhledem k pracovnímu adresáři) a zda soubor skutečně existuje.

Can not open output file "<jméno souboru>"

Nelze vytvořit a zapsat výstupní soubor <jméno souboru>. Disk je buď plný nebo je chráněn proti zápisu. Pokud pracujete v počítačové síti, možná nemáte dostatek přístupových práv do aktuálního adresáře.

Can not close file "<jméno souboru>"

Soubor byl otevřen, ale nemůže být uzavřen. Patrně došlo k závažné chybě záznamového média nebo není dostatek místa pro uložení souboru.

Can not compile this line

Překladač nerozumí řádce zdrojového programu. Zkontrolujte, zda je direktiva nebo mnemotechnický název instrukce syntakticky správně.

Maximum number of included files or macros exceeded

Byl překročen povolený počet vkládání souborů. Mohlo se např. stát, že soubor opakovaně vkládá sám sebe (direktivou INCLUDE) a došlo k zacyklení. Prověřte v každém vkládaném souboru, jaké soubory jsou do něj vkládány.

Unmatched parentheses.

Počet otevřených závorek ve výrazu není shodný s počtem uzavřených závorek. Zkontrolujte zápis výrazu.

Can not evaluate expression ‘<výraz>’

<výraz> nemůže být vyhodnocen. Zkontrolujte zápis výrazu, zda jsou správně zapsány matematické výrazy a konstanty.

Can not add new label ‘<návěští>’ to symbol table

<návěští> je v programu definováno více než jednou. Zkuste použít jiné jméno. Pokud je <návěští> součástí makroinstrukce, nadefinujte <návěští> jako lokální (direktiva LOCAL) nebo přepište definici makroinstrukce.

Can not add new symbol ‘<symbol>’ to symbol table

<symbol> již byl jednou definován. Zkuste použít pro symbol jiné jméno.

Unknown data type

Neznámý datový typ.

Unknown data type or expression

Neznámý datový typ nebo výraz.

Code placed on address ‘<adresa>’ replaced by another one

Došlo k přepisu programu na adrese <adresa>.

Keyword ‘const’ must be followed by keyword ‘byte’

Za klíčovým slovem CONST musí následovat klíčové slovo BYTE.

Insufficient memory to add symbol ‘<symbol>’ to symbol table

Došlo k vyčerpání dostupné paměti pro uchování symbolů. <symbol> není tedy možné přidat do tabulky symbolů. Zredukujte počet nepoužívaných symbolů.

Duplicate definition of symbol ‘<symbol>’

Pokus o předefinování hodnoty symbolu <symbol>.

Illegal bank number

Špatné číslo banky.

Address out of data memory

Adresa registru je mimo datovou paměť zvoleného procesoru.

Command allowed for 16CXX family only

Tato instrukce je definována pouze v rodině procesorů 16CXX.

Address out of program memory

Adresa programu je mimo programovou paměť.

Bit number should be in range 0-7. Number truncated!

Číslo bitu může být pouze v rozsahu 0-7. Číslo bylo oříznuto na 3 spodní bity.

File Register number should be 0-#. Number truncated!

Číslo registru může být pouze v rozsahu 0-X. Číslo bylo oříznuto.

File Register number should be 0-47. Number truncated!”

Číslo registru může být pouze v rozsahu 0-47. Číslo bylo oříznuto.

Call is possible to low 256 bytes of page only

V rodině procesorů 16C5X je možné volat podprogramy, jejichž počátek je ve spodní polovině stránky programové paměti.

Literal value truncated to 8 bits

Konstanta větší než 256 byla oříznuta na 8 bitů.

Unknown data type

Neznámý datový typ.

Can not evaluate address

U datového typu nemůže překladač vyhodnotit adresu. Zkontrolujte, zda je adresa uvozena znakem '@'.

Missing address

U datového typu není zadána adresa.

Can not evaluate bit number

U datového typu bit nebo pole bitů je špatně zadáno číslo bitu. Pravděpodobně došlo k překlepu, zkontrolujte zápis.

Can not evaluate bank number

U datového typu je špatně zadáno číslo banky registrů. Pravděpodobně došlo k překlepu, zkontrolujte zápis.

Bad constant syntax, '=' expected

Špatný zápis definice typu CONST BYTE. Je očekáván znak "=", za kterým musí následovat hodnota.

Illegal hexadecimal number

Špatný zápis hexadecimálního čísla. Číslo musí vždy začínat číslicí, ne písmenem. Např. FFh je špatně, správně má být 0FFh.

']' expected

Chybí uzavírací závorka indexu pole.

Syntax error

Špatný zápis příkazu nebo direktivy

Bad number of real macro parameters. Check macro definition

Počet parametrů předávaných do makroinstrukce není shodný s počtem parametrů uvedeným při definici makroinstrukce.

Directive '<direktiva>' is not supported

<direktiva> není překladačem podporována.

Unimplemented register

Číslo registru je mimo rozsah datové RAM.

Label is not allowed here

Na tomto řádku nelze definovat návěští.

Missing symbol name
: Chybí jméno symbolu.

Can not open macro file.
: Soubor s definicemi makroinstrukcí nelze otevřít. Existuje?

Can not close macro file
: Soubor s definicemi makroinstrukcí nelze uzavřít.

Duplicite macro definition '<makro>'
: <makro> již bylo definováno. Změňte jméno makroinstrukce.

Number of macro parameters exceeds limit
: Makroinstrukce může mít maximálně 10 parametrů. Zadáno jich je více.

Missing #endm directive
: Neukončená definice makroinstrukce.

Missing macro name
: V definici makroinstrukce není zadáno jméno makroinstrukce.

Number of macro local labels exceeds limit
: Překročen počet možných lokálních návěští v makrech (max. 10).

Number of macros exceeds limit
: Počet makroinstrukcí překročil zadaný limit.

Missing operand
: V instrukci nebo direktivě chybí operand.

Invalid TRIS argument
: Špatný operand instrukce TRIS

Jump to different page of program memory
: Skok programu na adresu mimo aktuální stránku programové paměti.

This is a *DEMO* version
: Je nainstalována demoverze překladače, která je omezena počtem překládaných řádků.

Too many errors & warnings
: Zdrojový text programu obsahuje příliš mnoho chyb.

Unknown error
: Neznámá chyba překladače.

Poznámky: