Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

Při použití makra napíšeme pouze jeho název a případné parametry. Při překladu pak dochází k jeho rozvinutí – expanzi (překladač dosadí řádky podle definice makra, parametry jsou nahrazeny skutečnými registry).

Nevýhodou makra je skutečnost, že k rozvinutí dochází při jeho každém použití. Takže se zvyšuje délka cílového souboru.

.MACRO – definice makra

`.MACRO` definuje počátek makra. Parametrem je jméno makra.

Je-li makro použito později v programu, dochází k jeho rozvinutí (expanzi).

Makro může mít až 10 parametrů. Tyto parametry jsou odkázány symboly `@0` až `@9`. Při volání makra se parametry oddělují čárkami.

Formát:

```
.MACRO jméno_makra
```

.ENDMACRO – konec definice makra

`.ENDMACRO` definuje konec makra. Tato direktiva nemá parametry.

Formát:

```
.ENDMACRO
```

Příklad použití

Dále je definováno makro `SUBI16`, které od slova tvořeného obsahem dvou registrů odečte konstantu v rozměru slova. Prvním parametrem je slovo, které se odečítá. Další dva parametry jsou vstupní a zároveň výstupní registry.

Makro je použito pro odečtení konstanty `$1234` od obsahu `R17:R16`. Výsledek je uložen do `R17:R16`.

```
.MACRO SUBI16              ;Začátek definice makra
SUBI @1,LOW(@0)            ;dolní bajt
SBCI @2,HIGH(@0)           ;horní bajt + výpůjčka
.ENDMACRO                  ;konec definice makra

.CSEG
SUBI16 $1234,r16,r17       ;použití makra
```

6.2.3 Řízení výpisu překladu

Další tři direktivy slouží k řízení výpisu překladu.

.LIST – zapnutí generace výpisu

`.LIST` sděluje překladači, že má začít generaci výpisu. Výpisem se rozumí speciální soubor, který je kombinací překládaného zdrojového kódu, adres a přeložených operačních kódů (jméno stejné jako zdrojový soubor, přípona LST).

Výpis je zapnut automaticky na začátku překladu. Lze jej vypnout direktivou `.NOLIST`. Tak lze řídit, která část se objeví ve výpisu.

Formát:

```
.LIST
```

.NOLIST – vypnutí generace výpisu

`.NOLIST` vypne generaci výpisu. Formát:

```
.NOLIST
```

.LISTMAC – zapnutí expanze maker ve výpisu

`.LISTMAC` sděluje překladači, aby prováděl rozvoje maker při výpisu (zobrazí se obsah makra). Výchozí je expanze pouze těch maker, která mají parametry.

Formát:

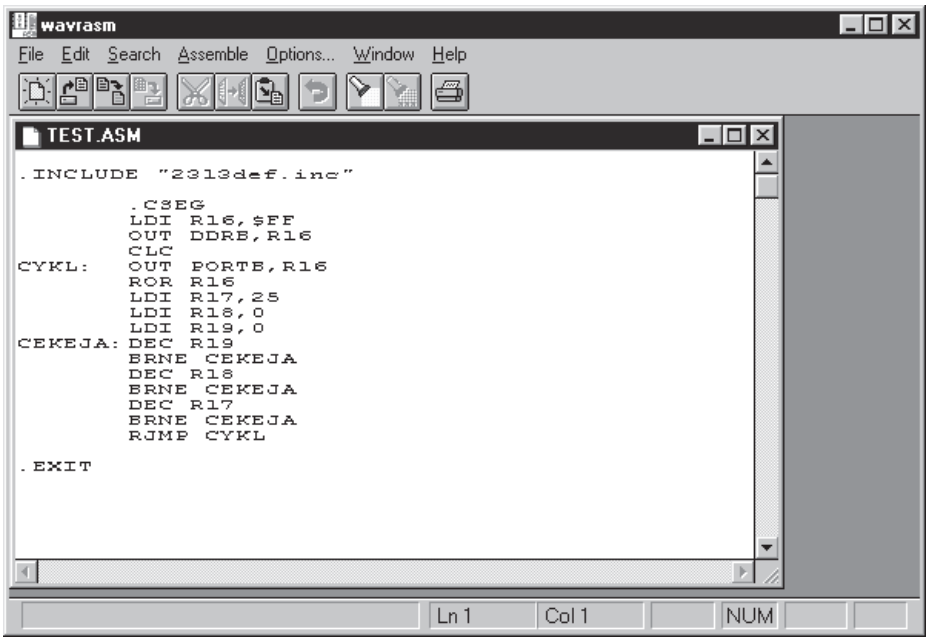
```
.LISTMAC
```

6.3 OVLÁDÁNÍ AVR 3.1

Ovládání překladače AVR 3.1 je vskutku jednoduché.

Hlavní okno této aplikace ukazuje *obr. 6.1*. Aplikace obsahuje editor zdrojového textu, případně další okna (například s výpisem překladu).

V hlavním menu jsou klasické položky pro práci se soubory a editaci. Překlad se aktivuje položkou **Assemble**.

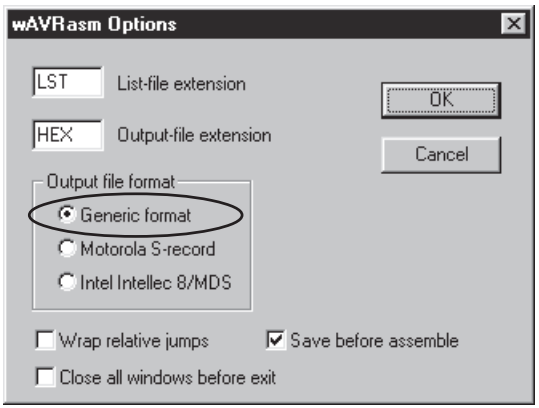


Obr. 6.1 Assembler AVR 3.1 pro Windows

Nejdůležitější je dialog voleb **wAVRasm|Options**. Zobrazí se po aktivaci položky **Options** v menu. Viz *obr. 6.2*.

Význam voleb

- **List-file Extension** – přípona souboru s protokolem o překladu (výpisu),
- **Output-file Extension** – přípona cílového souboru překladu,
- **Output file Format** – formát cílového souboru, volby:
  - **Generic format** – adresa následovaná operačním kódem (používá **SDKAVR**),
  - **Motorola S-record** – formát definovaný pro mikrokontroléry Motorola,
  - **Intel Intellec 8/MDS** – klasický IntelHex formát,
- **Wrap relative jumps** – použitelné pouze pro mikrokontroléry s velikostí FLASH 8K: cílové adresy relativních skoků a volání podprogramů jsou omezeny na adresní prostor 4 Kslova (8 KB),
- **Save before assemble** – uloží zdrojový soubor před překladem (důležitá volba: překládá se totiž vždy soubor uložený na disku, ne aktuální obsah okna editoru),
- **Close all windows before exit** – zajistí, že po novém spuštění se *neotevře* dříve editovaný soubor (pracovní plocha bude prázdná).



Obr. 6.2 Dialog Options

6.4 CO NAJDETE V SOUBORECH INC

Při zápisu programu musí překladač znát registry, kterými mikrokontrolér disponuje. Proto se výše popsanou direktivou **.INCLUDE** vkládají soubory definic, které tyto údaje obsahují.

Takové soubory najdete v adresáři **APPNOTES** vaší instalace **AVR 3.1**. Níže je uveden obsah souboru **2313DEF.INC**, který odpovídá mikrokontroléru **AT90S2313** přeložený do češtiny (a drobátko více okomentovaný):

```
;***** Určení typu mikrokontroléru:
.device AT90S2313
```

```
;***** Definice vstupně/výstupních registrů:
.equ     SREG=$3f
.equ     SPL=$3d
...
...
.equ     UBRR=$09
.equ     ACSR=$08
;*****  Definice bitů některých význačných
        vstupně/výstupních registrů:
;ukazatel vrcholu zásobníku:
.equ     SP7=7
.equ     SP6=6
.equ     SP5=5
.equ     SP4=4
.equ     SP3=3
.equ     SP2=2
.equ     SP1=1
.equ     SP0=0

;vstupy vnějšího přerušení:
.equ     INT1=7
.equ     INT0=6
.equ     INTF1=7
.equ     INTF0=6

;čítače/časovače 0 a 1:
.equ     TOIE1=7
.equ     OCIE1A=6
.equ     TICIE=3
.equ     TOIE0=1
.equ     TOV1=7
.equ     OCF1A=6
.equ     ICF1=3
.equ     TOV0=1
.equ     CS02=2
.equ     CS01=1
.equ     CS00=0
.equ     COM1A1=7
.equ     COM1A0=6
.equ     PWM11=1
.equ     PWM10=0
.equ     ICNC1=7
```

```
.equ     ICES1=6
.equ     CTC1=3
.equ     CS12=2
.equ     CS11=1
.equ     CS10=0

;bity registru MCUCR:
.equ     SE=5
.equ     SM=4
.equ     ISC11=3
.equ     ISC10=2
.equ     ISC01=1
.equ     ISC00=0

;bity registru WDTCR:
.equ     WDTOE=4
.equ     WDE=3
.equ     WDP2=2
.equ     WDP1=1
.equ     WDP0=0

;bity registru EECR:
.equ     EEMWE=2
.equ     EEWE=1
.equ     EERE=0

;bity PORTB:
.equ     PB7=7
.equ     PB6=6
.equ     PB5=5
.equ     PB4=4
.equ     PB3=3
.equ     PB2=2
.equ     PB1=1
.equ     PB0=0

;bity DDRB:
.equ     DDB7=7
.equ     DDB6=6
.equ     DDB5=5
.equ     DDB4=4
.equ     DDB3=3
.equ     DDB2=2
.equ     DDB1=1
```

```
.equ     DDB0=0

;bity PINB:
.equ     PINB7=7
.equ     PINB6=6
.equ     PINB5=5
.equ     PINB4=4
.equ     PINB3=3
.equ     PINB2=2
.equ     PINB1=1
.equ     PINB0=0

;podobně pro PORTD, DDRD, PIND. . .

;jednotka UART:

.equ     RXC=7
.equ     TXC=6
.equ     UDRE=5
.equ     FE=4
.equ     OR=3
.equ     RXCIE=7
.equ     TXCIE=6
.equ     UDRIE=5
.equ     RXEN=4
.equ     TXEN=3
.equ     CHR9=2
.equ     RXB8=1
.equ     TXB8=0

;analogový komparátor:
.equ     ACD=7
.equ     ACO=5
.equ     ACI=4
.equ     ACIE=3
.equ     ACIC=2
.equ     ACIS1=1
.equ     ACIS0=0

;ukazatelové registry:
.def     XL=r26
.def     XH=r27
.def     YL=r28
.def     YH=r29
```

```
.def     ZL=r30
.def     ZH=r31

; ***** Konce paměťových prostorů:
.equ     RAMEND=$DF      ;poslední adresa RAM
.equ     XRAMEND=$DF     ;poslední adresa datové paměti
.equ     E2END=$7F       ;poslední adresa E²PROM
.equ     FLASHEND=$3FF   ;poslední adresa FLASH

;***** Definice vektorů přerušení:
.equ     INT0addr=$001   ;vektor vnějšího přerušení 0
.equ     INT1addr=$002   ;vektor vnějšího přerušení 1
.equ     ICP1addr=$003   ;vektor přerušení od Input Capture č/č 1
.equ     OC1addr =$004   ;vektor přerušení od Output Compare č/č 1
.equ     OVF1addr=$005   ;vektor přetečení č/č 1
.equ     OVF0addr=$006   ;vektor přetečení č/č 0
.equ     URXCaddr=$007   ;vektor přerušení při dokončení
                         ;příjmu znaku přes UART
.equ     UDREaddr=$008   ;vektor přerušení při vyprázdnění
                         ;datového registru UART
.equ     UTXCaddr=$009   ;vektor přerušení při dokončení
                         ;vysílání znaku přes UART
.equ     ACIaddr =$00a   ;vektor přerušení od analogového
                         ;komparátoru
```

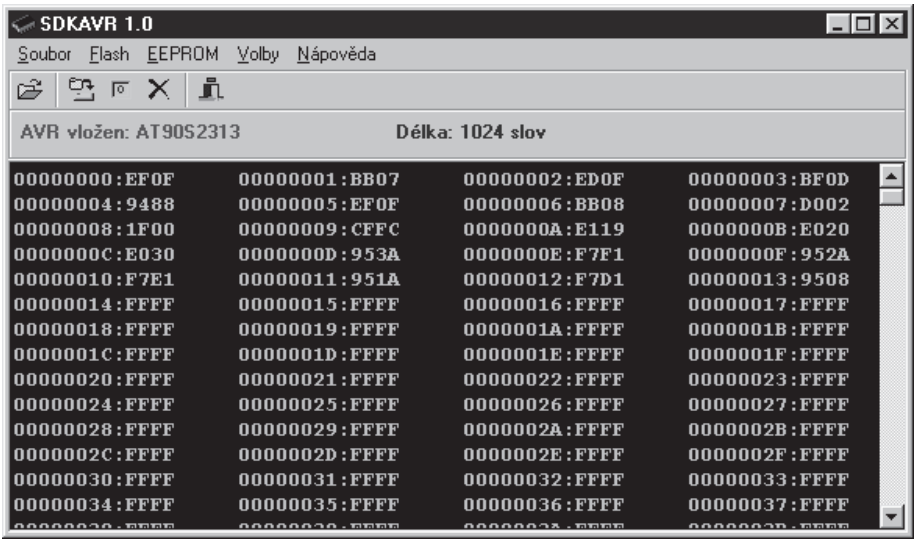
6.5 OVLÁDÁNÍ VÝVOJOVÉHO PROSTŘEDÍ SDKAVR 1.0

Vývojové prostředí **SDKAVR** umožňuje ovládat programovou i datovou paměť mikrokontrolérů ATMEL AVR, hlavní znaky:

- podporuje mikrokontroléry ATMEL AVR typů: **AT90S1200, AT90S2313, AT90S2343, AT90S4433, AT90S8515** a **AT90S8535**,
- možnost nastavení zámků chránících obsah mikrokontroléru proti zpětnému čtení,
- možnost smazat FLASH a E²PROM a provést reset mikrokontroléru.

Požadavky kladené na počítač:

- operační systém Windows 98 a vyšší,
- volný sériový port pro komunikaci s vývojovým kitem (viz kapitolu 3),
- postačí 8 MB RAM,
- instalace zabere méně než 1 MB prostoru na pevném disku.



Obr. 6.3 Vývojové prostředí SDKAVR

Inicializační soubor **SDKAVR.INI**:

- obsahuje číslo portu, který používáme pro komunikaci s vývojovým kitem,
- ukládají se do něj volby zámku a cesty k poslednímu souboru.

```
SDKAVR.INI:
[PORT]

Port=1          ;zvolený port (1 značí COM1)

[VOLBY]         ;poslední cesta a režim zámku:
Cesta=C:\Matousek\AVR\PROGRAMY\PROG_1
Zamek=6

[OKNO]          ;velikost a umístění okna
X=151
Y=137
V=342
```

Práce se soubory:

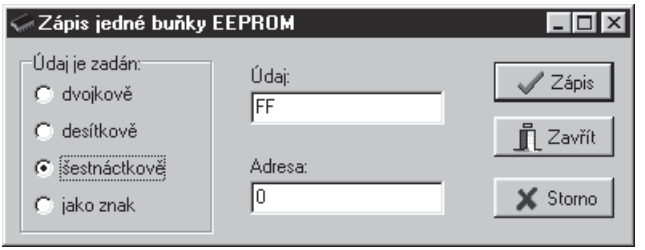
- **Soubor|Otevřít HEX** – otevře HEX soubor obsahující překlad programu (pozor, vývojové prostředí SDKAVR vyžaduje generický formát HEX souboru dle *obr. 6.2*),
- **Soubor|Uložit HEX** – uloží obsah okna do podoby HEX souboru,
- **Soubor|Čti Signaturu** – čte signaturu mikrokontroléru a tím zjišťuje jeho typ,
- **Soubor|Konec** – ukončí vývojové prostředí.

Ovládání programové paměti (FLASH):

- **FLASH|Programovat FLASH** – naprogramuje FLASH obsahem dříve načteného HEX souboru,
- **FLASH|Číst FLASH** – čte obsah FLASH a zobrazí jej v okně (lze uložit položkou menu **Soubor|Uložit HEX** do souboru),
- **FLASH|Reset** – resetuje mikrokontrolér,
- **FLASH|Smazat** – smaže obsah FLASH i E²PROM (nutné například před novým programováním po aplikaci zámků vyšší úrovně).

Ovládání datové paměti (E²PROM):

- **EEPROM|Editace** – umožní editaci zvolené buňky E²PROM jako dvojkové, desítkové nebo šestnáctkové číslo nebo znak (viz *obr. 6.4*),



Obr. 6.4 *Dialog pro editaci zvolené buňky E²PROM*

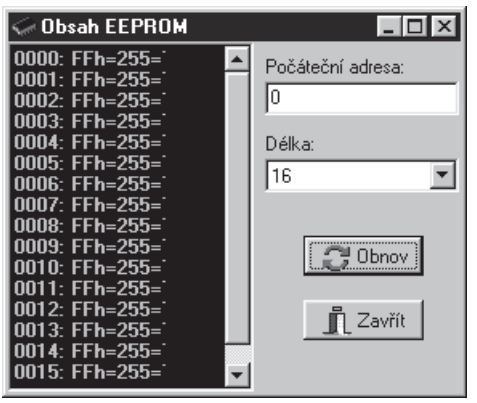
- **EEPROM|Dump** – zobrazí výpis zvolené části E²PROM (viz *obr. 6.5*),
- **EEPROM|Programovat EEP souborem** – naprogramuje E²PROM souborem s příponou EEP, který je výsledkem překladu (podobně jako HEX).

Zámky:

- **Volby|Zámek – volí zámek**, který použijeme pro ochranu návrhu. Možnosti: **Nepoužit**, **Odstavení zápisu, Odstavení zápisu a verifikace**.

Nápověda:

- **Nápověda|O aplikaci** – podá nápovědu o programu.



Obr. 6.5 *Dialog pro zobrazení obsahu E²PROM*