XASMAVR V2.1
Macro Assember for the Atmel AVR® Microcontroller family

COPYRIGHT NOTICE

Licence information

GNU General Public License

DISCLAIMER
The information in this document is subject to change without notice. While the information contained herein is assumed to be accurate, no responsibility is taken for any errors or omissions. In no event the author of this document be liable for special, direct, indirect, or consequential damage, losses, costs, charges, claims, demands, claim for lost profits, fees, or expenses of any nature or kind.
This program is provided for educational purpose and personal use WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

TRADEMARKS
AVR®, 'AVR® Studio', Atmel and Microchip are registered trademarks of Atmel/Microchip Corporation. All other product names are trademarks or registered trademarks of their respective owners. Fujitsu, SIEMENS, IBM, Atmel, Intel, Motorola, Microsoft are registered enterprises, brands and registered trademarks. The mentioned companies have no relation to the variants of XASM and are therefore not responslible for any problems that occur when using any of the XASM variants. Thank you for your products, support and efforts.

RELEASE NOTICE
First release: October 2024

1. Abstract

The XASMAVR Macro Assembler V2.1 runs under Windows® and generates fix code for the Atmel/Microchip AVR® microcontroller family. XASMAVR.exe is the newest variant of haXASM.

XASMAVR 2.1 implements most of the features described in the following document:
'Microchip AVR® Assembler'
'©2017 Microchip Technology Inc. User Guide DS40001917A'

haXASM project sources `*.cpp` are written in explicit 'C', however, observing the syntax of the MS C++ Compiler. In order to gain an educational effect, extensive "pointerism" as well as any complex structure constructs have been intentionally avoided. The sources are decently commented to show the "where and how" (columns are tabulated 3 5 7 ..). Some redundancies were implemented to enhance understanding of certain functionality.

To build the 'XASM*.EXE' executables as a 32bit-Version for Windows XP, Vista, Windows 10, 11, . .
all ***.cpp *h *nmk** files reside in the folder **c:\temp600\\__\\*.*** .
They are compiled within the *"Visual Studio 2010 Developer Command Prompt":*

```
'Setting environment for using Microsoft Visual Studio 2010 x86 tools.'
'C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC>NMAKE c:\temp600\__\haXASM.nmk'
```

The build process generates the following 4 different Cross-Assemblers:

- XASM6800.exe – Cross-Assembler for Motorala CPU MC6800/6802
- XASM8042.exe – Cross-Assembler for Intel's UPI-C42
- XASM6805.exe – Macro-Assembler for Motorola CPU MC68HC05 (all types)
- XASMAVR.exe – Macro-Assembler for Atmel/Microchip AVR® devices instruction set

... see the Nmake-script **'haXASM.NMK'** for details.

Note: When compiled with MS Visual Studio 2019, XASMAVR will no longer run under Windows XP.
To build and run the haXASM project under another OS, minor adaptations may be required.

The build process automatically generates 64bit versions of XASM when invoked under:

```
**************************************************************************
** Visual Studio 2019 Developer Command Prompt v16.8.4
** Copyright (c) 2020 Microsoft Corporation
**************************************************************************
[vcvarsall.bat] Environment initialized for: 'x64'
'C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>NMAKE c:\temp600\__\haXASM.nmk'
```

... see the Nmake-script **'haXASM.NMK'** for details.

1.1 The long history of XASM

```
//******************************************************************************
//*        -----------------------------------------------                     *
//*       |   C R O S S - A S S E M B L E R  Version 2.1  |                    *
//*        -----------------------------------------------                     *
//* PURPOSE: Cross-Assembler for ISIS-II & MS-DOS Systems                      *
//*           re-written from 8085 asm-code, running under MS-DOS > 3.20       *
//*           re-written in C running under Windows XP 32bit or greater        *
//*          .LST-file output                                                  *
//*          .HEX-file output (Intel)                                          *
//*          .S19-file output (Motorola)                                       *
//*          .BIN-File output (PROM-programming)                               *
//*                                                                            *
//* AUTHOR:  h. altmann,       (c)Copyright 1980,1990 by ha.                   *
//*                            (c)Copyright 2024 by ha.                        *
//*                                                                            *
//* ABSTRACT: The X-Assembler was originally written for Intel ISIS-II systems *
//*           running on the 8085. To make the XASM run on an IBM-XT/AT PC     *
//* under MS-DOS operating system the following steps were performed:          *
//* The 8085 source has been converted into 8086 compatible source using the   *
//* Intel tool "CONV86" on an AT-PC DOS 3.20 system under "XIRUN" environment. *
//* Then the ISIS-II system-macros were re-written according to the MS-DOS     *
//* V3.20 conventions. After the correct functionality had been verified,      *
//* further optimizations were applied to the 8086 asm-source including the    *
//* implementation of the UPI-41/42 instruction set and some helpful functions *
//* for expression evaluation. Previously this XASM had been designed to       *
//* assemble Motorola 6800/6802 code on an ISIS-II system.                     *
//* ..those were the days!                                                      *
//* Finally the old x86 assembler source code was transscribed into 32bit C    *
//* (compiled with Microsoft C++ compiler) as a Windows console application.   *
//* Next step should port the C-source to real C++ object modules, focused on  *
//* speed and performance.                                                     *
//*                                                                            *
//* Currently XASM derivations for Windows console are tested and available:   *
//* Version 2.0                                                                *
//*  XASM8042 - Cross-Assembler for Intel MCS48, 8042, UPI-C42                 *
//*  XASM6802 - Cross-Assembler for Motorola MC6800/6802                       *
//*  XASM6805 - Cross-Assembler for Motorola MC68HC05 (all types)              *
//*                                                                            *
//* Version 2.1 (XASMAVR, XASM68* include conditional assembly and macros)     *
//*  XASM8042 - Cross-Assembler for Intel MCS48, 8042, UPI-C42                 *
//*  XASM6802 - Macro-Assembler for Motorola MC6800/6802                       *
//*  XASM6805 - Macro-Assembler for Motorola MC68HC05 (all types)              *
//*  XASMAVR  - Macro-Assembler for Atmel/Microchip AVR(R) uC family           *
//*                                                                            *
//* Instruction sets of other microcontrollers can be easily added.            *
//*                                                                            *
//******************************************************************************
```

To assemble an AVR® source file, run **`'XASMAVR myavrproj.asm'`**.
You will end up with a listing at **`'myavrproj.LST'`** of the source file,
and with **`'myavrproj.hex' / 'myavrproj.eep.hex'`** in Intel HEX format.

2. XASMAVR special Features

With respect to the original Atmel assembler AVRASM2
extra preprocessor directives have been added in XASMAVR:

.MODEL BYTE	*BYTE addresses in listing*
.MODEL WORD	*WORD addresses in listing (default)*
.MODEL NOINFO	*suppress additional info on screen*
.MODEL SYNTAX	*extended syntax check*

2.1 CSEG Byte or Word addresses in listing

*Controling the rendition of the *.LST file*

1) show BYTE addresses in code segment with instructions splitted into byte entities
Commandline: **'XASMAVR /Mb'**. In source using directive: '.MODEL BYTE'

2) show WORD addresses in code segment where instructions are word entities
Commandline: **'XASMAVR /Mw'**. In source using directive: '.MODEL WORD'

Note: Listing byte addresses instead of word addresses sometimes can be very useful, because of the direct correspondence with the layout in *.HEX files.

Invoke **'XASMAVR /?'** for more commandline options.

2.2 List File Formatting Directives

.MODEL BYTE	*CSEG Byte addresses in listing (special XASMAVR feature)*
.MODEL WORD	*CSEG Word addresses in listing (default)*
.MODEL NOINFO	*Suppress some info displayed on console*
.TITLE text	*Title in page header*
.SUBTTL text	*Subtitle in page header*
.PAGELENGTH(number)	*Lines per page*
.PAGEWIDTH(number)	*Colums per page*
.EJECT	*New listing page ejected ('FormFeed')*
.SYMBOLS	*Symbol map appended in listing*
.NOSYMBOLS	*Listing without symbol map (default)*
.NOLISTMACRO	*Disable listing of macro expansion (default)*
.NOLISTMAC	*Disable listing of macro expansion (default)*
.LISTMACRO	*Enable listing of macro expansion*
.LISTMAC	*Enable listing of macro expansion*
.NOLIST	*Suppress source lines in listing*
.LIST	*Show source lines in listing (default)*

3. Syntax Overview

3.1 Comments

```
00000000  0000          :C      1   nop
                                2   ; standard assembler comment
                                3   // same as ';'
                                5   /* Multiline text block.
                                6      The enclosed text is a comment
                                7   */ Note: Block marker must be 1st in text line.
00000001  0000          :C      8   nop
```

3.2 Line continuation

Source lines with .DB .DW .DD .DQ directives can be continued by backslash '\' as the last character.

Example (.MODEL WORD)

```
                                   10000   ;-----------------------------------------------
00000000                           10001   verstr: .DB      __DATE__, __TIME__,        \
                                                            "XASMAVR Macro Assembler", \
                                                            'V', REV, '.', REV0, REV00,
00000000  3232 312f 2f30  :C
00000003  3032 3432 3031  :C
00000006  333a 3a38 3231  :C
00000009  4158 4d53 5641  :C
0000000c  2052 614d 7263  :C
0000000f  206f 7341 6573  :C
00000012  626d 656c 2072  :C
00000015  3256 312e 30    :C
00000017  00              :C   10002           .EVEN
= 0000002f                         10003   .SET VERSTR_LENB=STRLEN()  ; Bytes in .DB statement
= 00000017                         10004   .SET SIGNON_LENB=STRLEN("XASMAVR Macro Assembler")
= 00000018                         10005   .SET VERSTR_LENW=PC-verstr ; Words in .DB statement
00000018                  :C   10006   next:
                                   10007   ;-----------------------------------------------
```

Example (.MODEL BYTE)

```
                                   10000   ;-----------------------------------------------
00000000                           10001   verstr: .DB      __DATE__, __TIME__,        \
                                                            "XASMAVR Macro Assembler", \
                                                            'V', REV, '.', REV0, REV00,
00000000  32 32 2F 31 30  :C
00000005  2F 32 30 32 34  :C
0000000A  31 30 3A 34 30  :C
0000000F  3A 33 37 58 41  :C
00000014  53 4D 41 56 52  :C
00000019  20 4D 61 63 72  :C
0000001E  6F 20 41 73 73  :C
00000023  65 6D 62 6C 65  :C
00000028  72 20 56 32 2E  :C
0000002D  31 30           :C
0000002F  00              :C   10002           .EVEN
= 0000002f                         10003   .SET VERSTR_LENB=STRLEN()  ; Bytes in previous .DB statement
= 00000017                         10004   .SET SIGNON_LENB=STRLEN("XASMAVR Macro Assembler")
= 00000018                         10005   .SET VERSTR_LENW=PC-verstr ; Words in .DB statement
00000030                  :C   10006   next:
                                   10007   ;-----------------------------------------------

                                   10008   ;-----------------------------------------------
00000030                           10009          _db3:    .DB       _11,_12, _13, 'Z'+1,    \
                                                                     0b10101, 0b1, 0b101,    \
                                                                     "1234567890abcd ",      \
                                                                     'A',_db3 AND 0xFF,      \
                                                                     "1234567890abcde"       \
                                                                     $FF,$FF
00000030  11 12 13 5B 15  :C
00000035  01 05 31 32 33  :C
0000003A  34 35 36 37 38  :C
0000003F  39 30 61 62 63  :C
00000044  64 20 41 18 31  :C
00000049  32 33 34 35 36  :C
0000004E  37 38 39 30 61  :C
00000053  62 63 64 65 FF  :C
00000058  FF              :C
00000059  00              :C                   .EVEN
                                   10010   ;-----------------------------------------------
```

3.3 Assembler Directives

.CSEG - Code Segment

.DSEG - Data Segment

.ESEG - EEPROM Segment

Instructions are accepted in .ESEG, allowing special applications to store code chunks in EEPROM.

*.CSEGSIZE Flash Program Memory Size in Kword (K*16 = 2K*8)*

Example

```
.CSEGSIZE 12 ; Flash ROM size = 12K x 16 (or 24Kbyte).
```

.BYTE - Reserve bytes in RAM as space for a variable

Example

```
                                  61           .DSEG                    ; RAM area
00000200                  :D      62     cmdparm        .BYTE    1       ; keyboard command parameter
00000201                  :D      63     set3Modtb      .BYTE   32       ; set3 Typematic/Make/Break
= 00000020                        64     .SET SET3MODTB_L = $-set3Modtb  ; XASMAVR: $=PC
= 00000020                        65     .SET SET3MODTB_L = PC-set3Modtb
00000221                  :D      66                    .BYTE   SET3MODTB_L
00000241                  :D      67     next:
                                  68           .CSEG                    ; Flash ROM area
00000001  e0c0            :C      69           ldi R28,LOW(cmdparm)     ; Load Y reg low
00000002  e0d2            :C      70           ldi R29,HIGH(cmdparm)    ; Load Y reg high
00000003  e0c0            :C      71           ldi YL,LOW(cmdparm)      ; (YL/YH=*def.inc)
00000004  e0d2            :C      72           ldi YH,HIGH(cmdparm)     ; Load Y reg high
00000005  8018            :C      73           ld  R1,Y                 ; ld cmdparm into R1
                                  74     ;-------------------------------------------------
```

.DB - Define constant byte(s) in CSEG and ESEG

Examples (.MODEL BYTE)

```
00002468                  :C      76           .ORG    0x1234
00002468                  :C      77     Exdb  .DB     LOW(PC),0b11,%110,111b,0x56,$ab, \
                                                       0CDh, 123, 'G'+1,"ABC”,HIGH($)
00002468  34 03 06 07 56  :C
0000246D  AB CD 7B 48 41  :C
00002472  42 43 12        :C
00002475  00              :C                   .EVEN
                                  79
00002476  01 0D 30 31 32  :C      80           .DB     1,13,"012'3':; ,'",'A','"',''','',"",22,25
0000247B  27 33 27 3A 3B  :C
00002480  20 2C 27 41 22  :C
00002485  27 00 00 16 19  :C
0000248A  0014            :C      81           .DW     STRLEN()
0000248C  01 0D 30 31 32  :C      82           .DB     1,13,'012"3"45 ,"','A','"',''','',"",22,25
00002491  22 33 22 34 35  :C
00002496  20 2C 22 41 22  :C
0000249B  27 00 00 16 19  :C
000024A0  0014            :C      83           .DW  STRLEN()
                                  54     ;-------------------------------------------------
```

.DW - Define constant word(s) in CSEG and ESEG

Example (.MODEL WORD)

```
00001234                  :C      81           .ORG    0x1234
00001234                          82     Exdw  .DW     $,0b11,%110,111b,0x56, \
                                                       $456789ab,0CDEFh,65535
00001234  1234            :C
00001235  0003            :C
00001236  0006            :C
00001237  0007            :C
00001238  0056            :C
00001239  89ab            :C
<<<XASMAVR_test.ASM: |WARNING| Out of range, value is masked: 0x456789AB
0000123a  cdef            :C
0000123b  ffff            :C
0000123c                  :C      83     next:
                                  84     //-----------------------------------------------
```

.DD - Define constant double word(s) in CSEG and ESEG

Example (.MODEL WORD)

```
 00001234                 :C      86          .ORG    0x1234
 00001234                         87   Exdd    .DD     PC, 0b11, %110, 111b, 0x56, \
                                                       $3456789ab, 89ABCDEFh,4294967296-2
 00001234  00001234       :C
 00001236  00000003       :C
 00001238  00000006       :C
 0000123A  00000007       :C
 0000123C  00000056       :C
 0000123E  456789ab       :C
<<<XASMAVR_test.ASM: |WARNING| Out of range, value is masked: 0x3456789AB
 00001240  89abcdef       :C
 00001242  fffffffe       :C

                                  89          .ESEG
 00001004                 :C      90          .ORG    0x1234
 00001004  00001004       :E      91          .DD     PC, 0b11, %110, 111b, 0x56
 00001008  00000003       :E
 0000100C  00000006       :E
 00001010  00000007       :E
 00001014  00000056       :C
                                  92          .CSEG
 00001244                 :C      93   next:
                                  94   ;---------------------------------------------
```

.DQ - Define constant quad-word(s) in CSEG and ESEG

Example (.MODEL WORD)

```
 00001234                 :C      91    .ORG   0x1234
 00001234                         92   exdq:
                                        .DQ    0b1110001111111111111111111111111111111111111000111010101001010101, \
                                               18446744073709551615-1, 0x1234567890ABCDEF, exdq, PC
 00001234  e3ffffffffe3aa55:C
 00001238  fffffffffffffffe:C
 0000123C  1234567890abcdef:C
 00001240  0000000000001234:C
 00001244  0000000000001244:C
 00001248                 :C      93   next:
                                  94   ;------------------------------------------------------------
```

.DEF - Assign a symbolic name to a register

Example

```
 = 00000011                      101   .DEF TMPR=R17
 = 00000000                      102   .DEF IOR =R0
                                 103           .CSEG
 00000001  ea1a           :C     104           ldi     TMPR, 0xAA ; Load 0xAA into TMPR register
 00000002  b60f           :C     105           in      IOR, 0x3F  ; Read SREG into IOR register
 00000003  2510           :C     106           eor     TMPR, IOR  ; XOR TMPR and IOR
                                 107   ;---------------------------------------------------------
```

.EQU - Assign a constant symbol equal to an expression.

Example

```
 = 00000050                      201   .EQU RAMSTART   = 0x0050        ; User scratch pad
 = 000000B0                      202   .EQU RAM_SIZE   = 176
 = 000000C0                      203   .EQU STACK      = RAMSTART+(RAM_SIZE-STACK_SIZE) ; Stack area
 = 00000040                      204   .EQU STACK_SIZE = 64
                                 205           .DSEG                   ; User RAM
 00000050                 :D     206           .ORG RAMSTART
 00000050                 :D     207   typem   .BYTE   16
                                 208   ;       . . . . . . . . . . . . . . . . . . . . . . . .
 000000C0                 :D     209           .ORG STACK              ; Program STACK ($00C0 .. 00FF)
 000000C0                 :D     210           .BYTE   STACK_SIZE      ; 64 bytes user stack area
                                 211   ;       . . . . . . . . . . . . . . . . . . . . . . . .
 = 00000100                      212   .EQU RAMEND = PC                ; 0x00FF = last RAM location
                                 213           .CSEG
                                 214   ;---------------------------------------------------------
```

.SET - Assign a redefinable symbol equal to an expression.

Example

```
 = 00000268                   216   .SET _VAR = 1234/2-1
 = 00002c5d                   217   .SET _VAR = 5678*2+1
                              225   ;---------------------------------------
```

.EVEN

Code alignment on a word address boundary. Automatically appended to .DB with odd number of bytes.

.MESSAGE, .WARNING, .ERROR - Display information on console during assembly

Example

```
.MESSAGE "Info text"
.WARNING "Warning info text"
.ERROR    "Error: Assembly aborted text"

#message "Info text"
#warning "Warning info text"
#error    "Error: Assembly aborted text"
;------------------------------------
```

3.4 Conditional assembly

.DEFINE (not specified by Atmel)
.UNDEF
.IFDEF
.IF DEFINED
.IFNDEF
.IF !DEFINED
.IF
.ELIF
.ELSEIF
.ELIF DEFINED
.ELIF !DEFINED
.ELSE
.ENDIF

#define (Commandline: `'XASMAVR /D<symbol>' Define text symbol`)
#undef
#ifdef
#if defined
#ifndef
#if !defined
#if
#elif
#elseif
#elif defined
#elseifdef
#elif !defined
#elseifndef
#else
#endif

3.5 Macros

.MACRO
.ENDM
.ENDMACRO
.NOLISTMAC (XASMAVR Feature)
.NOLISTMACRO (XASMAVR Feature)
.LISTMAC
.LISTMACRO

Example Source (.MODEL WORD):

```
.LISTMAC
;;--------------------------------------------------------------
; Macro Test (Macro(s) within Macro and local Macro labels)
.MACRO wrInit   ;; Macro #1
mtst:   jmp     mtst    ;; local label mtst
        call    init    ;; local
        call    cmd     ;; local
        call    _delay  ;; local
        jmp     delay   :: global
        .DW     LOW(cmd+init), HIGH(init)
        .DW     cmd, cmd+Init+_delay
        WRCMD (1<<_F)|(0<<_F_8B)
cmd:    WRCMD (1<<_F)|(0<<_F_8B)|(1<<_F_2L)
        WRCMD (1<<_CLR)
_delay: WRCMD (1<<_ENTRY_MODE)|(1<<_ENTRY_INC)
        WRCMD (1<<_ON)|(1<<_ON_DISPLAY)|(0<<_ON_CURSOR)|(0<<_ON_BLINK)
init:   WRCMD (1<<_HOME)
.ENDM

.MACRO wrCmd    ;;Macro #2
        ldi     R17, @0
 wrc:   call    cmd
        ROUT    10, R17, 9
.ENDM

.MACRO rOut     ;;Macro #3
        ldi     @1, @2
 rou:   out     @0, @1
.ENDMACRO
;;--------------------------------------------------------------

.EQU    _CLR            = 0b00000000 ;
.EQU    _HOME           = 0b00000001 ;
.EQU    _ENTRY_INC      = 0b00000001 ;
.EQU    _ENTRY_MODE     = 0b00000010 ;

.EQU    _ON_BLINK       = 0b00000000 ;
.EQU    _ON_CURSOR      = 0b00000001 ;
.EQU    _ON_DISPLAY     = 0b00000010 ;
.EQU    _ON             = 0b00000011 ;

.EQU    _F_2L           = 0b00000011 ;
.EQU    _F_8B           = 0b00000100 ;
.EQU    _F              = 0b00000101 ;

call    init
call    delay
call    cmd

mtst:   WRINIT ;; Macro expansion #1
;       ------

        WRINIT ;; Macro expansion #2
;       ------
.NOLISTMAC
        WRINIT ;; Macro expansion #3
;       ------
init:   ret
delay:  ret
cmd:    ret
```

Listing:

```
= 00000000                        76   .EQU    _CLR             = 0b00000000 ;
= 00000001                        77   .EQU    _HOME            = 0b00000001 ;
= 00000001                        78   .EQU    _ENTRY_INC       = 0b00000001 ;
= 00000002                        79   .EQU    _ENTRY_MODE      = 0b00000010 ;
                                  80
= 00000000                        81   .EQU    _ON_BLINK        = 0b00000000 ;
= 00000001                        82   .EQU    _ON_CURSOR       = 0b00000001 ;
= 00000002                        83   .EQU    _ON_DISPLAY      = 0b00000010 ;
= 00000003                        84   .EQU    _ON              = 0b00000011 ;
                                  85
= 00000003                        86   .EQU    _F_2L            = 0b00000011 ;
= 00000004                        87   .EQU    _F_8B            = 0b00000100 ;
= 00000005                        88   .EQU    _F               = 0b00000101 ;
                                  89
00000100  940e 018a       :C      90   call    init
00000102  940e 018b       :C      91   call    delay
00000104  940e 018c       :C      92   call    cmd
                                  93
00000106                  :C +    94   mtst:   WRINIT ;; Macro expansion #1
00000106  940c 0106       :C +    95   mtst_00020001:  jmp      mtst_00020001
00000108  940e 012d       :C +    96           call    init_00020001
0000010a  940e 0119       :C +    97           call    cmd_00020001
0000010c  940e 0123       :C +    98           call    _delay_00020001
0000010e  940c 018b       :C +    99           jmp     delay
00000110  0046            :C +   100           .DW     LOW(cmd_00020001+init_00020001), HIGH(init_00020001)
00000111  0001            :C
00000112  0119            :C +   101           .DW     cmd_00020001, cmd_00020001+Init_00020001+_delay_00020001
00000113  0369            :C
00000114                  :C +   102           WRCMD (1<<_F)|(0<<_F_8B)
00000114  e210            :C +   103           ldi     R17, (1<<_F)|(0<<_F_8B)
00000115  940e 018c       :C +   104    wrc_00030001:  call    cmd
00000117                  :C +   105           ROUT 10, R17, 9
00000117  e019            :C +   106           ldi     R17, 9
00000118  b91a            :C +   107    rou_00040001:  out     10, R17
00000119                  :C +   108    cmd_00020001:  WRCMD (1<<_F)|(0<<_F_8B)|(1<<_F_2L)
00000119  e218            :C +   109           ldi     R17, (1<<_F)|(0<<_F_8B)|(1<<_F_2L)
0000011a  940e 018c       :C +   110    wrc_00030002:  call    cmd
0000011c                  :C +   111           ROUT 10, R17, 9
0000011c  e019            :C +   112           ldi     R17, 9
0000011d  b91a            :C +   113    rou_00040002:  out     10, R17
0000011e                  :C +   114           WRCMD (1<<_CLR)
0000011e  e011            :C +   115           ldi     R17, (1<<_CLR)
0000011f  940e 018c       :C +   116    wrc_00030003:  call    cmd
00000121                  :C +   117           ROUT 10, R17, 9
00000121  e019            :C +   118           ldi     R17, 9
00000122  b91a            :C +   119    rou_00040003:  out     10, R17
00000123                  :C +   120   _delay_00020001:        WRCMD (1<<_ENTRY_MODE)|(1<<_ENTRY_INC)
00000123  e016            :C +   121           ldi     R17, (1<<_ENTRY_MODE)|(1<<_ENTRY_INC)
00000124  940e 018c       :C +   122    wrc_00030004:  call    cmd
00000126                  :C +   123           ROUT 10, R17, 9
00000126  e019            :C +   124           ldi     R17, 9
00000127  b91a            :C +   125    rou_00040004:  out     10, R17
00000128                  :C +   126           WRCMD (1<<_ON)|(1<<_ON_DISPLAY)|(0<<_ON_CURSOR)|(0<<_ON_BLINK)
00000128  e01c            :C +   127           ldi     R17, (1<<_ON)|(1<<_ON_DISPLAY)|(0<<_ON_CURSOR)|(0<<_ON_BLINK)
00000129  940e 018c       :C +   128    wrc_00030005:  call    cmd
0000012b                  :C +   129           ROUT 10, R17, 9
0000012b  e019            :C +   130           ldi     R17, 9
0000012c  b91a            :C +   131    rou_00040005:  out     10, R17
0000012d                  :C +   132    init_00020001: WRCMD (1<<_HOME)
0000012d  e012            :C +   133           ldi     R17, (1<<_HOME)
0000012e  940e 018c       :C +   134    wrc_00030006:  call    cmd
00000130                  :C +   135           ROUT 10, R17, 9
00000130  e019            :C +   136           ldi     R17, 9
00000131  b91a            :C +   137    rou_00040006:  out     10, R17
                                 138   ;       ------
                                 139
00000132                  :C +   140           WRINIT ;; Macro expansion #2
00000132  940c 0132       :C +   141   mtst_00020002:  jmp      mtst_00020002
00000134  940e 0159       :C +   142           call    init_00020002
00000136  940e 0145       :C +   143           call    cmd_00020002
00000138  940e 014f       :C +   144           call    _delay_00020002
0000013a  940c 018b       :C +   145           jmp     delay
0000013c  009e            :C +   146           .DW     LOW(cmd_00020002+init_00020002), HIGH(init_00020002)
0000013d  0001            :C
0000013e  0145            :C +   147           .DW     cmd_00020002, cmd_00020002+Init_00020002+_delay_00020002
0000013f  03ed            :C
```

```
00000140                :C +    148         WRCMD (1<<_F)|(0<<_F_8B)
00000140  e210          :C +    149         ldi     R17, (1<<_F)|(0<<_F_8B)
00000141  940e 018c     :C +    150   wrc_00030007:  call    cmd
00000143                :C +    151         ROUT 10, R17, 9
00000143  e019          :C +    152         ldi     R17, 9
00000144  b91a          :C +    153   rou_00040007:  out     10, R17
00000145                :C +    154   cmd_00020002:  WRCMD (1<<_F)|(0<<_F_8B)|(1<<_F_2L)
00000145  e218          :C +    155         ldi     R17, (1<<_F)|(0<<_F_8B)|(1<<_F_2L)
00000146  940e 018c     :C +    156   wrc_00030008:  call    cmd
00000148                :C +    157         ROUT 10, R17, 9
00000148  e019          :C +    158         ldi     R17, 9
00000149  b91a          :C +    159   rou_00040008:  out     10, R17
0000014a                :C +    160         WRCMD (1<<_CLR)
0000014a  e011          :C +    161         ldi     R17, (1<<_CLR)
0000014b  940e 018c     :C +    162   wrc_00030009:  call    cmd
0000014d                :C +    163         ROUT 10, R17, 9
0000014d  e019          :C +    164         ldi     R17, 9
0000014e  b91a          :C +    165   rou_00040009:  out     10, R17
0000014f                :C +    166  _delay_00020002:        WRCMD (1<<_ENTRY_MODE)|(1<<_ENTRY_INC)
0000014f  e016          :C +    167         ldi     R17, (1<<_ENTRY_MODE)|(1<<_ENTRY_INC)
00000150  940e 018c     :C +    168   wrc_00030010:  call    cmd
00000152                :C +    169         ROUT 10, R17, 9
00000152  e019          :C +    170         ldi     R17, 9
00000153  b91a          :C +    171   rou_00040010:  out     10, R17
00000154                :C +    172         WRCMD (1<<_ON)|(1<<_ON_DISPLAY)|(0<<_ON_CURSOR)|(0<<_ON_BLINK)
00000154  e01c          :C +    173         ldi     R17, (1<<_ON)|(1<<_ON_DISPLAY)|(0<<_ON_CURSOR)|(0<<_ON_BLINK)
00000155  940e 018c     :C +    174   wrc_00030011:  call    cmd
00000157                :C +    175         ROUT 10, R17, 9
00000157  e019          :C +    176         ldi     R17, 9
00000158  b91a          :C +    177   rou_00040011:  out     10, R17
00000159                :C +    178   init_00020002: WRCMD (1<<_HOME)
00000159  e012          :C +    179         ldi     R17, (1<<_HOME)
0000015a  940e 018c     :C +    180   wrc_00030012:  call    cmd
0000015c                :C +    181         ROUT 10, R17, 9
0000015c  e019          :C +    182         ldi     R17, 9
0000015d  b91a          :C +    183   rou_00040012:  out     10, R17
                                184  ;      ------
                                185  .NOLISTMAC
0000015e                :C +    186         WRINIT ;; Macro expansion #3
0000016c                :C +                WRCMD (1<<_F)|(0<<_F_8B)
0000016f                :C +                ROUT 10, R17, 9
00000171                :C +          cmd_00020003:   WRCMD (1<<_F)|(0<<_F_8B)|(1<<_F_2L)
00000174                :C +                ROUT 10, R17, 9
00000176                :C +                WRCMD (1<<_CLR)
00000179                :C +                ROUT 10, R17, 9
0000017b                :C +         _delay_00020003: WRCMD (1<<_ENTRY_MODE)|(1<<_ENTRY_INC)
0000017e                :C +                ROUT 10, R17, 9
00000180                :C +                WRCMD (1<<_ON)|(1<<_ON_DISPLAY)|(0<<_ON_CURSOR)|(0<<_ON_BLINK)
00000183                :C +                ROUT 10, R17, 9
00000185                :C +          init_00020003:  WRCMD (1<<_HOME)
00000188                :C +                ROUT 10, R17, 9
                                187  ;      ------
0000018a  9508          :C      188  init:  ret
0000018b  9508          :C      189  delay: ret
0000018c  9508          :C      190  cmd:   ret
```

3.6 Include files

.INCLUDE "AVRdef.inc"
#include <AVRdef.inc>

Example (test.inc, test_0.inc, test_1.inc)

File = .test.inc

```
.MESSAGE "test.inc - START"
.INCLUDE "test_0.inc"     ;; >><test.inc>: .include .include <test_0.inc>
.MESSAGE "test.inc - Finish test_0.inc"
.INCLUDE "test_1.inc"     ;; >><test.inc>: .include .include <test_1.inc>
.MESSAGE "test.inc - Finish test_1.inc"
.EXIT                     ;; >>test: 'Warning-Needed': Forced abort test.inc
;;-------------------
        nop               ;; >>test
;;-------------------
.MESSAGE "test.inc - END" ;; >> test.inc - normal EOF
```

```
                                             ;-------------------------------------------------------------------
                                           19   #include "test.inc"
                                           20 C
                                           21 C .MESSAGE "test.inc - START"
                                           22 C .INCLUDE <test_0.inc>          ;; >>test.inc: .include <test_0.inc>
                                           23 C #message "test_0.inc - START"
                                           24 C #include <test_1.inc>          ;; >>test_0: .include <test_1.inc>
                                           25 C #message "test_1.inc - START"
                                           26 C ;;--------------------
 00000000  00 00           :C              27 C nop                            ;; >>test_1
                                           28 C ;;--------------------
                                           29 C #message "test_1.inc - END"     ;; >>test_1.inc - Normal EOF
                                           30 C .MESSAGE "test_0.inc - Finish test_1.inc"
                                           31 C ;;---------------------
 00000002  00 00           :C              32 C nop                            ;; >> test_0
                                           33 C ;;---------------------
                                           34 C #message "test_0.inc - END"     ;; >> test_0.inc - Normal EOF
                                           35 C .MESSAGE "test.inc - Finish test_0.inc"
                                           36 C .INCLUDE <test_1.inc>          ;; >>test.inc: .include <test_1.inc>
                                           37 C #message "test_1.inc - START"
                                           38 C ;;--------------------
 00000004  00 00           :C              39 C nop                 ;; >>test_1
                                           40 C ;;--------------------
                                           41 C #message "test_1.inc - END"     ;; >>test_1.inc - Normal EOF
                                           42 C .MESSAGE "test.inc - Finish test_1.inc"
<<<test.inc: |WARNING| Check END-OF-FILE directive: .EXIT
                                             ;-------------------------------------------------------------------
```

3.7 Other directives

.LIST
.NOLIST
.EXIT - Force to exit current file (abort assembly)

3.8 Directives for program memory layout

.ORG
.OVERLAP
.NOOVERLAP
.DEVICE
#pragma

3.8.1 .DEVICE - AVR Part Related

*Refers to the device pool of XASMAVR to define memory layout without any *def.inc files*

Example

```
.DEVICE ATmega2560
```

Information displayed on console during assembly:

```
AVR Macro-Assembler, Version 2.1
        DEVICE PART_NAME AVRPART ATmega2560
        MEMORY PROG_FLASH START_ADDR 0x0000
        MEMORY PROG_FLASH SIZE 262144 (256K)
        MEMORY INT_SRAM START_ADDR 0x0200
        MEMORY INT_SRAM SIZE 8192 (8K)
        MEMORY EEPROM START_ADDR 0x0000
        MEMORY EEPROM SIZE 4096 (4K)
Creating Symbol Xref Table...

ASSEMBLY COMPLETE,   NO ERRORS
```

3.8.2 Printing the list of assembler supported AVR devices

Invoke ***'XASMAVR /d'*** *to list all assembler supported AVR microcontrollers.*
Invoke ***'XASMAVR /d >_devLst.txt'*** *to save the list of supported AVR microcontrollers into a txt-file.*

Example

```
------------------------------------------------------------------------
DeviceName     FLASH Start  Size   SRAM Start  Size   EEPROM Start  Size
------------------------------------------------------------------------
ATmega16       FLASH=0x0000 (16K)  SRAM=0x0060 (1K)   EEPROM=0x0000 (512)
ATmega2560     FLASH=0x0000 (256K) SRAM=0x0200 (8K)   EEPROM=0x0000 (4K)
...
ATtiny102      FLASH=0x0000 (1K)   SRAM=0x0040 (032)  EEPROM=0x0000 (000)
ATtiny85       FLASH=0x0000 (8K)   SRAM=0x0060 (512)  EEPROM=0x0000 (512)
...
AVR128DA28     FLASH=0x0000 (128K) SRAM=0x4000 (1632) EEPROM=0x1400 (512)
AVR128DA32     FLASH=0x0000 (128K) SRAM=0x4000 (1632) EEPROM=0x1400 (512)
...
ATxmega32E5    FLASH=0x0000 (32K)  SRAM=0x2000 (4K)   EEPROM=0x1000 (1K)
ATxmega384C3   FLASH=0x0000 (384K) SRAM=0x2000 (32K)  EEPROM=0x1000 (4K)
...
ATA6286        FLASH=0x0000 (8K)   SRAM=0x0000 (512)  EEPROM=0x0000 (320)
ATA6612C       FLASH=0x0000 (8K)   SRAM=0x0000 (000)  EEPROM=0x0000 (000)
------------------------------------------------------------------------
DeviceName     INSTRUCTIONS_NOT_SUPPORTED
------------------------------------------------------------------------
ATmega2560     INSTRUCTIONS_NOT_SUPPORTED   :DES:LAC:LAS:LAT:XCH
ATtiny102      INSTRUCTIONS_NOT_SUPPORTED   :DES:LAC:LAS:LAT:XCH
                                            :FMUL:FMULS:FMULSU:MUL:MULSU
                                            :ADIW:MOVW:SBIW:LD:LDD:STD
                                            :EICALL:EIJMP:CALL:JMP:IJMP:ICALL
                                            :LPM:ELPM:SPM:BREAK
AVR128DA28     INSTRUCTIONS_NOT_SUPPORTED   :DES:LAC:LAS:LAT:XCH
                                            :EICALL:EIJMP
ATxmega32E5    Full instruction set support
ATA6286        INSTRUCTIONS_NOT_SUPPORTED   :DES:LAC:LAS:LAT:XCH
                                            :FMUL:FMULS:FMULSU:MUL:MULSU
                                            :ADIW:MOVW:SBIW:LD:LDD:STD
                                            :EICALL:EIJMP:CALL:JMP:IJMP:ICALL
                                            :LPM:ELPM:SPM:BREAK
```

3.8.3 #pragma - AVR Part Related

Processed in XASMAVR to define memory layout with ***def.inc** *files*
see **"©2017 Microchip Technology Inc. User Guide DS40001917A-page 29"**

Example

```
.INCLUDE "m2560def.inc"
```

*Information displayed on console during assembly (derived from #pragma in *def.inc):*

```
AVR Macro-Assembler, Version 2.1
        partinc 0 "m2560def.inc"
        ADMIN PART_NAME ATmega2560
        CORE CORE_VERSION V3
        MEMORY PROG_FLASH 262144 (256K)
        MEMORY EEPROM 4096 (4K)
        MEMORY INT_SRAM SIZE 8192 (8K)
        MEMORY INT_SRAM START_ADDR 0x200
 Creating Symbol Xref Table...
                                    .
ASSEMBLY COMPLETE,   NO ERRORS
```

4. Pre-defined Macros

Note: %DATE% %TIME%.. etc, - this format is not supported. Using __DATE__ __TIME__ instead

Examples (.MODEL BYTE)

```
----------------------------------------------------------------------------------
 If source with line continuation and terminating zero
                                             .DB      __DATE__, __TIME__, '__CENTURY__', \
                                                      0x00
 then listing shows expanded text string
 000001F8                         100   _db10   .DB      '18/12/2024', '12:12:46', '21', \
                                                      0x00
 000001F8  31 38 2F 31 32  :C
 000001FD  2F 32 30 32 34  :C
 00000202  31 32 3A 31 32  :C
 00000207  3A 34 36 32 31  :C
 0000020C  00              :C
 0000020D  00              :C           .EVEN
----------------------------------------------------------------------------------

 else default: Listing without showing expanded text
 00000000  58 41 53 4D 41  :C     111           .DB __FILE__
 00000005  56 52 5F 74 65  :C
 0000000A  73 74 2E 41 53  :C
 0000000F  4D              :C
                                  112
 00000010  0015            :C     113           .DW __CENTURY__
 00000012  32 31           :C     114           .DB "__CENTURY__"
                                  115
 00000014  32 32 2F 31 30  :C     116           .DB __DATE__
 00000019  2F 32 30 32 34  :C
                                  117
 0000001E  07E8            :C     118           .DW __YEAR__
 00000020  32 30 32 34     :C     119           .DB "__YEAR__"
                                  120
 00000024  0A              :C     121           .DB __MONTH__
 00000025  00              :C                   .EVEN
 00000026  31 30           :C     122           .DB "__MONTH__"
                                  123
 00000028  16              :C     124           .DB __DAY__
 00000029  00              :C                   .EVEN
 0000002A  32 32           :C     125           .DB "__DAY__"
                                  126
 0000002C  32 33 3A 33 33  :C     127           .DB __TIME__
 00000031  3A 33 33        :C
                                  128
 00000034  15              :C     129           .DB __HOUR__
 00000035  00              :C                   .EVEN
 00000036  32 31           :C     130           .DB "__HOUR__"
                                  131
 00000038  21              :C     132           .DB __MINUTE__
 00000039  00              :C                   .EVEN
 0000003A  33 33           :C     133           .DB "__MINUTE__"
                                  134
 0000003C  23              :C     135           .DB __SECOND__
 0000003D  00              :C                   .EVEN
 0000003E  33 35           :C     136           .DB "__SECOND__"
```

5. Functions – Provided by the assembler.

LOW(expression) returns the low byte of an expression
HIGH(expression) returns the second byte of an expression

```
00000000  AB 56          :C      138          .DB LOW(0x56AB), HIGH(0x56AB)
```

BYTE2(expression) is the same function as HIGH
BYTE3(expression) returns the third byte of an expression
BYTE4(expression) returns the fourth byte of an expression

```
00000002  CD AB 89       :C      139      .DB BYTE2(0x89ABCDEF), BYTE3(0x89ABCDEF), BYTE4(0x89ABCDEF)
00000005  00             :C               .EVEN
```

LWRD(expression) returns bits 0-15 of an expression
HWRD(expression) returns bits 16-31 of an expression

```
00000006  CDEF           :C      140     .DW LWRD(0x89ABCDEF), (0x89ABCDEF AND 0x0000FFFF)      ; Bits [15:0]
00000008  CDEF           :C
0000000A  89AB           :C      141     .DW HWRD(0x89ABCDEF), (0x89ABCDEF SHR 16)              ; Bits [31:16]
0000000C  89AB           :C
```

"PAGE(expression) returns bits 16-21 of an expression"
see ***Microchip AVR Assembler Manual 2017 Chapt 7.1:***

```
0000000E  002B           :C      142     .DW PAGE(0x89ABCDEF), (0x89ABCDEF & 0x003F0000) >> 16  ; Bits [21:16]
00000010  002B           :C
00000012  0034           :C      143     .DW PAGE(0x12345678), (0x12345678 & 0x003F0000) >> 16  ; Bits [21:16]
00000014  0034           :C
```

EXP2(expression) returns 2 to the power of expression

```
00000018  0010           :C      144          .DW EXP2(4),  16
0000001A  0010           :C
0000001C  8000           :C      145          .DW EXP2(15), 32768, 1<<15
0000001E  8000           :C
00000020  8000           :C
00000022  0000           :C      146          .DW EXP2(16)
<<<XASMAVR_test.ASM: |WARNING| Out of range, value is masked: 0x10000
00000024  00010000       :C      147          .DD EXP2(16), 65536              ; =$000010000
00000028  00010000       :C
0000002C  00800000       :C      148          .DD EXP2(23), 8388608            ; =$008000000
00000030  00800000       :C
00000034  80000000       :C      149          .DD EXP2(31), 2147483648         ; =$080000000 Bits [31:0]
00000038  80000000       :C
0000003C  00000001       :C      150          .DD EXP2(32), 4294967296         ; =$100000000 Out of range
<<<XASMAVR_test.ASM: |WARNING| Out of range, value is masked: 0x100000000
00000040  00000000       :C
<<<XASMAVR_test.ASM: |WARNING| Out of range, value is masked: 0x100000000
                                 151
00000044  0000000100000000:C     152   .DQ EXP2(32)                           ; =$0000000100000000
0000004C  8000000000000000:C     153   .DQ EXP2(63), 1<<63                    ; Max range = Bits [63:0]
00000054  8000000000000000:C
```

LOG2(expression) returns the integer part of log2(expression)

```
0000005C  02 02          :C      154          .DB LOG2(4),                    2
0000005E  04 04          :C      155          .DB LOG2(16),                   4
00000060  17 17          :C      156          .DB LOG2(8388608),             23
00000062  1F 1F          :C      157          .DB LOG2(2147483648),          31
00000064  20 20          :C      158          .DB LOG2(4294967296),          32
00000066  3F 3F          :C      159          .DB LOG2(18446744073709551615), 63
```

Floating point 1.7 assembler functions (Microchip AVR)

INT(expression) Truncates a floating point expression to integer (i.e. discards fractional part)

```
00000068  0003             :C      160          .DW     INT(3.1415926)
0000006A  0001             :C      162          .DW     INT(1.780029)
0000006C  FFFF             :C      163          .DW     INT(-1.780029)
```

FRAC(expression) Extracts fractional part of a floating point expression (i.e. discards integer part).

```
0000006E  000BE6E0         :C      164          .DD     FRAC(1.780000)          ;; =780000
00000072  E6E0             :C      165          .DW     FRAC(1.780000) & 0xFFFF ;; = 59104(!)
```

Q7() and Q15() Convert a fractional floating point expression to a form suitable for the *FMULU/FMULSU* instructions *(atmel-0856-avr-instruction-set-manual.pdf)*

Q7(expression). (Sign + 7-bit fraction.)

Q15(expression) (Sign +15-bit fraction.)

Examples

```
00000074  0049             :C      166          .DW     Q7(0.575)       ;; = $49
00000076  00B2             :C      167          .DW     Q7(1.390625)    ;; = $B2
00000078  004E             :C      168          .DW     Q7(-1.390625)   ;; = $4E
0000007A  004E             :C      169          .DW     Q7(0.609375)    ;; = $4E
0000007C  00EC             :C      170          .DW     Q7(1.85)        ;; = $EC (->1.6C = 1.110 1100)
                                   171
0000007E  4999             :C      172          .DW     Q15(0.575)      ;; = $4999
00000080  B200             :C      173          .DW     Q15(1.390625)   ;; = $B200
00000082  4E00             :C      174          .DW     Q15(-1.390625) ;; = $4E00
00000084  4E00             :C      175          .DW     Q15(0.609375)   ;; = $4E00
00000086  ECCC             :C      176          .DW     Q15(1.85)       ;; = $ECCC (->1.6C = 1.110 1100)
                                   177
00000088  0050             :C      178          .DW     Q7(0.625)       ;; = $50
0000008A  00B0             :C      179          .DW     Q7(-0.625)      ;; = $B0 = (NOT $50) + 1
0000008C  00B0             :C      180          .DW     Q7(1.375)       ;; = $B0 = (NOT $50) + 1
                                   181
0000008E  00E8             :C      182          .DW     Q7(1.8125)      ;; = $E8
00000090  0080             :C      183          .DW     Q7(-1)          ;; = $80
00000092  7F FF            :C      184          .DB     Q7(0.09921875), Q7(0.09921875) ;; = $7F, $FF
                                   185
00000094  7FFF             :C      186          .DW     Q15(0.999969482421875)
00000096  63D7             :C      187          .DW     Q15(0.780029)  ;; = nearest of 1.78 in memory
00000098  E3D7             :C      188          .DW     Q15(1.78)
0000009A  291A             :C      189          .DW     Q15(0.321117799673)     ;; = 10522
```

ABS(expression) Returns the absolute value of a constant expression

Example

```
0000009C  007B             :C      190          .DW     ABS(123)        ;; = $7B
0000009E  007B             :C      191          .DW     ABS(-123)       ;; = $7B
```

STRLEN(string) Returns the length of a string constant, in bytes

Example (.MODEL BYTE)

```
000000A0  1D               :C      193          .DB     STRLEN("XASMAVR Macro Assembler V2.1 ")
000000A1  00               :C                   .EVEN
```

XASMAVR special Feature

STRLEN() Returns the number of bytes in previous .DB statement

Example

```
00000000  58 41 53 4D 41  :C    10   signonMsg: .DB     "XASMAVR Macro Assembler V2.1 "
00000005  56 52 20 4D 61  :C
0000000A  63 72 6F 20 41  :C
0000000F  73 73 65 6D 62  :C
00000014  6C 65 72 20 56  :C
00000019  32 2E 31 20     :C
0000001D  00              :C                    .EVEN
0000001E  000F            :C    11              .DW     PC-signonMsg, STRLEN()
00000020  001D
= 0000001D                      12              .EQU    SIGNONMSG_LENGTH = STRLEN()
```

DEFINED(symbol). Returns true if symbol is previously defined using .EQU/.SET/.DEF directives. Normally used in conjunction with .IF directives (.IF DEFINED(foo)), but may be used in any context. It only makes sense to use a single symbol as argument.

Example

```
                                202   ;-----------------------------------------
 = 00000001                     203   #define _flag1
 = 00000001                     204   .DEFINE _flag2
0000009E  01 00          :C     205   .DB DEFINED(_flag1), !DEFINED(_flag1)
000000A0  01 00          :C     206   .DB DEFINED(_flag2), !DEFINED(_flag2)
                                207
                                208   #if DEFINED(_flag1)
000000A2  00 00          :C     209   nop            ; flag1
                                210   #elif DEFINED(_flag2)
                                211   #endif
                                212
                         .      213   #if !DEFINED(_flag1)
                                214   #elif DEFINED(_flag2)
000000A4  00 00          :C     215   nop            ; flag2
                                216   #endif
                                217   ;-----------------------------------------
```

6. Operands - The following operands can be used:

- *User defined labels, which are given the value of the location counter at the place they appear*
- *User defined variables with the .SET directive, user defined constants with the .EQU directive*
- *Constants can be given in several formats:*

```
- Decimal (default):                10, 255, 65536
- Hexadecimal (three notations):    0x10, $10, 10h, 0xff, $ff, 0FFh
- Binary (three notations):         0b00001010, %11111111, 11111111b
- Octal: not supported.             -
- Current Program memory location counter (three notations): PC, *, $
- Floating point constants: Sign + 7-bit fraction, Sign + 15-bit fraction.
```

7. Operators - The Assembler supports a number of operators

(see also "©2017 Microchip Technology Inc. User Guide DS40001917A-page 34..")

!	Logical not	
~ NOT	Bitwise Not	('NOT' is the same as '~' XASMAVR Feature)
-	Unary Minus	
*	Multiplication	
/	Division	
% MOD	Modulo	('MOD' is the same as '%' XASMAVR Feature)
+	Addition	
-	Subtraction	
<< SHL	Shift left	('SHL' is the same as '<<' XASMAVR Feature)
>> SHR	Shift right	('SHR' is the same as '>>' XASMAVR Feature)
<	Less than (LT)	
<=	Less than or equal (LE)	
>	Greater than (GT)	
>=	Greater than or equal (GE)	
==	Equal (EQ)	
!=	Not equal (NE)	
& AND	Bitwise And	('AND' is the same as '&' XASMAVR Feature)
^ XOR	Bitwise Xor	('XOR' is the same as '^' XASMAVR Feature)
| OR	Bitwise Or	('OR' is the same as '|' XASMAVR Feature)
&&	Logical And	
||	Logical Or	

7.1 Operator precedence – Parenthesis in complex expressions are recommended!

To yield the expected result of a complex expression, especially when arithmetic operators are mixed with logical operators, ***the usage of braces it is strongly recommended****.*

Assemblers/Compilers may sometimes slightly differ about the operator precedence rules in such expressions, generating a result that may not always be expected.

Example

```
                                       //----------------------------------------------
                                  218  // Warning: Use parenthesis in complex expressions!
= 00A60075                        219  .SET _VAR = 0xa600*256+0x75a2>>8
00000000  00A60075       :C       220  .DD _VAR                                  ;; =00A60075 NOT EXPECTED ?!
= 00A60075                        221  .SET _VAR = (0xa600*256)+(0x75a2>>8)
00000004  00A60075       :C       222  .DD _VAR                                  ;; =00a60075 expected
= 0000A675                        223  .SET _VAR = (0xa600*256+0x75a2)>>8
00000008  0000A675       :C       224  .DD _VAR                                  ;; =0000a675 expected
                                       //----------------------------------------------
```

8. AVR® Instruction Set

For information about the AVR® instruction set, refer to the 8-bit AVR® Instruction Set Manual.
*(see also "****©2021 Microchip Technology Inc. Manual DS40002198B****")*

```
AVR Macro-Assembler, Version 2.1                    26/10/2024  PAGE  1
'XASMAVR_test.asm': Test source for XASMAVR.EXE
The AVR instruction set

 LOC       OBJ                   LINE   SOURCE

 00000100                :C      2456           .ORG 0x100
 00000100                :C      2457   _instruction_set:
 00000100  1f00          :C      2458           adc     R16,R16
 00000101  0f00          :C      2459           add     R16,R16
 00000102  96cf          :C      2460           adiw    R25:R24,63
 00000103  96cf          :C      2461           adiw    R24,63
 00000104  2300          :C      2462           and     R16,R16
 00000105  7f0f          :C      2463           andi    R16,$FF
 00000106  9525          :C      2464           asr     R18
 00000107  9488          :C      2465           bclr    0
 00000108  f920          :C      2466           bld     R18,0
 00000109  f408          :C      2467           brbc    0,PC+2
 0000010a  f008          :C      2468           brbs    0,PC+2
 0000010b  f408          :C      2469           brcc    PC+2
 0000010c  f008          :C      2470           brcs    PC+2
 0000010d  f7e8          :C      2471           brbc    0,PC-2
 0000010e  f3e8          :C      2472           brbs    0,PC-2
 0000010f  f7e8          :C      2473           brcc    PC-2
 00000110  f3e8          :C      2474           brcs    PC-2
 00000111  9598          :C      2475           break
 00000112  f3e9          :C      2476           breq    PC-2
 00000113  f7ed          :C      2477           brhc    PC-2
 00000114  f3ed          :C      2478           brhs    PC-2
 00000115  f7ef          :C      2479           brid    PC-2
 00000116  f3ef          :C      2480           brie    PC-2
 00000117  f3e8          :C      2481           brLo    PC-2
 00000118  f3ec          :C      2482           brlt    PC-2
 00000119  f3ea          :C      2483           brmi    PC-2
 0000011a  f7e9          :C      2484           brne    PC-2
 0000011b  f7ea          :C      2485           brpl    PC-2
 0000011c  f7e8          :C      2486           brsh    PC-2
 0000011d  f7ee          :C      2487           brtc    PC-2
 0000011e  f3ee          :C      2488           brts    PC-2
 0000011f  f7eb          :C      2489           brvc    PC-2
 00000120  f3eb          :C      2490           brvs    PC-2
 00000121  9408          :C      2491           bset    0
 00000122  fa00          :C      2492           bst     R0,0
```

```
AVR Macro-Assembler, Version 2.1                  26/10/2024  PAGE  2
'XASMAVR_test.asm': Test source for XASMAVR.EXE
The AVR instruction set

 LOC       OBJ                 LINE   SOURCE

 00000123  940e 000a      :C    2493           call    10
 00000125  940e 0100      :C    2494           call    _instruction_set
 00000127  98f8           :C    2495           cbi     31,0
 00000128  7000           :C    2496           cbr     R16,255
 00000129  9488           :C    2497           clc
 0000012a  94d8           :C    2498           clh
 0000012b  94f8           :C    2499           cli
 0000012c  94a8           :C    2500           cln
 0000012d  2722           :C    2501           clr     R18
 0000012e  94c8           :C    2502           cls
 0000012f  94e8           :C    2503           clt
 00000130  94b8           :C    2504           clv
 00000131  9498           :C    2505           clz
 00000132  9520           :C    2506           com     R18
 00000133  1700           :C    2507           cp      R16,R16
 00000134  0700           :C    2508           cpc     R16,R16
 00000135  3f0f           :C    2509           cpi     R16,255
 00000136  1300           :C    2510           cpse    R16,R16
 00000137  952a           :C    2511           dec     R18
 00000138  95d8           :C    2512           elpm
 00000139  9126           :C    2513           elpm    R18,Z
 0000013a  9127           :C    2514           elpm    R18,Z+
 0000013b  2700           :C    2515           eor     R16,R16
 0000013c  0308           :C    2516           fmul    R16,R16
 0000013d  0380           :C    2517           fmuls   R16,R16
 0000013e  0388           :C    2518           fmulsu  R16,R16
 0000013f  9509           :C    2519           icall
 00000140  9409           :C    2520           ijmp
 00000141  b72f           :C    2521           in      R18,63
 00000142  9523           :C    2522           inc     R18
 00000143  940c 000a      :C    2523           jmp     10
 00000145  940c 0100      :C    2524           jmp     _instruction_set
 00000147  912c           :C    2525           ld      R18,  X
 00000148  912d           :C    2526           ld      R18,  X+
 00000149  912e           :C    2527           ld      R18, -X
 0000014a  8128           :C    2528           ld      R18,  Y
 0000014b  9129           :C    2529           ld      R18,  Y+
 0000014c  912a           :C    2530           ld      R18, -Y
 0000014d  8120           :C    2531           ld      R18,  Z
 0000014e  9121           :C    2532           ld      R18,  Z+
 0000014f  9122           :C    2533           ld      R18, -Z
 00000150  ad2f           :C    2534           ldd     R18,  Y+63
 00000151  ad27           :C    2535           ldd     R18,  Z+63
 00000152  ef0f           :C    2536           ldi     R16,255
 00000153  9120 ffff      :C    2537           lds     R18,65535
 00000155  95c8           :C    2538           lpm
 00000156  9124           :C    2539           lpm     R18, Z
 00000157  9125           :C    2540           lpm     R18, Z+
 00000158  0f22           :C    2541           lsl     R18
 00000159  9526           :C    2542           lsr     R18
 0000015a  2f00           :C    2543           mov     R16,R16
 0000015b  01de           :C    2544           movw    XH:XL,YH:YL
 0000015c  01de           :C    2545           movw    X,Y
 0000015d  9f00           :C    2546           mul     R16,R16
 0000015e  0200           :C    2547           muls    R16,R16
 0000015f  9521           :C    2548           neg     R18
 00000160  0000           :C    2549           nop
 00000161  2b00           :C    2550           or      R16,R16
 00000162  6f0f           :C    2551           ori     R16,$FF
 00000163  bf2f           :C    2552           out     63,R18
```

```
AVR Macro-Assembler, Version 2.1                         26/10/2024  PAGE  3
'XASMAVR_test.asm': Test source for XASMAVR.EXE
The AVR instruction set

 LOC       OBJ                    LINE   SOURCE

 00000164  912f            :C     2553           pop     R18
 00000165  932f            :C     2554           push    R18
 00000166  de8f            :C     2555           rcall   -10      ;; -10-(PCw+1) won't make any sense
 00000167  dff5            :C     2556           rcall   PC-10
 00000168  9508            :C     2557           ret
 00000169  9518            :C     2558           reti
 0000016a  ce9f            :C     2559           rjmp    +10      ;;  10-(PCw+1) won't make any sense
 0000016b  c009            :C     2560           rjmp    PC+10
 0000016c  1f22            :C     2561           rol     R18
 0000016d  9527            :C     2562           ror     R18
 0000016e  0b00            :C     2563           sbc     R16,R16
 0000016f  4f0f            :C     2564           sbci    R16,255
 00000170  9af8            :C     2565           sbi     31,0
 00000171  99f8            :C     2566           sbic    31,0
 00000172  9bf8            :C     2567           sbis    31,0
 00000173  97cf            :C     2568           sbiw    R25:R24,63
 00000174  97cf            :C     2569           sbiw    R24,63
 00000175  6f0f            :C     2570           sbr     R16,255
 00000176  fd20            :C     2571           sbrc    R18,0
 00000177  ff20            :C     2572           sbrs    R18,0
 00000178  9408            :C     2573           sec
 00000179  9458            :C     2574           seh
 0000017a  9478            :C     2575           sei
 0000017b  9428            :C     2576           sen
 0000017c  ef0f            :C     2577           ser     R16
 0000017d  9448            :C     2578           ses
 0000017e  9468            :C     2579           set
 0000017f  9438            :C     2580           sev
 00000180  9418            :C     2581           sez
 00000181  9588            :C     2582           sleep
 00000182  95e8            :C     2583           spm
 00000183  932c            :C     2584           st       X, R18
 00000184  932d            :C     2585           st       X+,R18
 00000185  932e            :C     2586           st      -X, R18
 00000186  8328            :C     2587           st       Y, R18
 00000187  9329            :C     2588           st       Y+,R18
 00000188  932a            :C     2589           st      -Y, R18
 00000189  8320            :C     2590           st       Z, R18
 0000018a  9321            :C     2591           st       Z+,R18
 0000018b  9322            :C     2592           st      -Z, R18
 0000018c  af2f            :C     2593           std     Y+63,R18
 0000018d  af27            :C     2594           std     Z+63,R18
 0000018e  9320 ffff       :C     2595           sts     65535,R18
 00000190  1b00            :C     2596           sub     R16,R16
 00000191  5f0f            :C     2597           subi    R16,255
 00000192  9522            :C     2598           swap    R18
 00000193  2322            :C     2599           tst     R18
 00000194  95a8            :C     2600           wdr
                                  2601
                                  2602   ; Advanced instructions implemented in some ATmega* µC
 00000195  945b            :C     2604           des     5
 00000196  9519            :C     2605           eicall
 00000197  9419            :C     2606           eijmp
 00000198  9326            :C     2607           lac     Z, R18
 00000199  9325            :C     2608           las     Z, R18
 0000019a  9327            :C     2609           lat     Z, R18
 0000019b  95e8            :C     2610           spm     Z+
 0000019c  9324            :C     2611           xch     Z, R18
 0000019d                  :C     2617   _End_of_instruction_set:
```

9. XASMAVR Listing with formatted Symbol Map and Info Block

Example

```
AVR Macro-Assembler, Version 2.1                          26/10/2024  PAGE  101

USER SYMBOLS

_1F . . . . . . . . . . . . . . . . . .  00000055 A  __FAR_away. . . . . . . . . . . . . . .  00010007 C
_db0. . . . . . . . . . . . . . . . . .  00001358 C  _db1. . . . . . . . . . . . . . . . . .  0000136B C
_db2. . . . . . . . . . . . . . . . . .  0000137F C  _db3. . . . . . . . . . . . . . . . . .  00000018 C
_DEBUG. . . . . . . . . . . . . . . . .  00000001 A  _End_of_instruction_set . . . . . . . .  0000019D C
_instruction_set. . . . . . . . . . . .  00000100 C  _M2560DEF_INC_. . . . . . . . . . . . .  00000001 A
....
WDP3. . . . . . . . . . . . . . . . . .  00000005 A  WDRF. . . . . . . . . . . . . . . . . .  00000003 A
WDTaddr . . . . . . . . . . . . . . . .  00000018 A  WDTCSR. . . . . . . . . . . . . . . . .  00000060 A
WDTON . . . . . . . . . . . . . . . . .  00000004 A  WGM00 . . . . . . . . . . . . . . . . .  00000000 A
WGM01 . . . . . . . . . . . . . . . . .  00000001 A  WGM02 . . . . . . . . . . . . . . . . .  00000003 A
WGM10 . . . . . . . . . . . . . . . . .  00000000 A  WGM11 . . . . . . . . . . . . . . . . .  00000001 A
WGM12 . . . . . . . . . . . . . . . . .  00000003 A  WGM13 . . . . . . . . . . . . . . . . .  00000004 A
WGM20 . . . . . . . . . . . . . . . . .  00000000 A  WGM21 . . . . . . . . . . . . . . . . .  00000001 A
WGM22 . . . . . . . . . . . . . . . . .  00000003 A  WGM30 . . . . . . . . . . . . . . . . .  00000000 A
WGM31 . . . . . . . . . . . . . . . . .  00000001 A  WGM32 . . . . . . . . . . . . . . . . .  00000003 A
WGM33 . . . . . . . . . . . . . . . . .  00000004 A  WGM40 . . . . . . . . . . . . . . . . .  00000000 A
WGM41 . . . . . . . . . . . . . . . . .  00000001 A  WGM42 . . . . . . . . . . . . . . . . .  00000003 A
WGM43 . . . . . . . . . . . . . . . . .  00000004 A  WGM50 . . . . . . . . . . . . . . . . .  00000000 A
WGM51 . . . . . . . . . . . . . . . . .  00000001 A  WGM52 . . . . . . . . . . . . . . . . .  00000003 A
WGM53 . . . . . . . . . . . . . . . . .  00000004 A  xa0 . . . . . . . . . . . . . . . . . .  00A60075 A
XH. . . . . . . . . . . . . . . . . . .  0000001B A  XL. . . . . . . . . . . . . . . . . . .  0000001A A
XMBK. . . . . . . . . . . . . . . . . .  00000007 A  XMCRA . . . . . . . . . . . . . . . . .  00000074 A
XMCRB . . . . . . . . . . . . . . . . .  00000075 A  XMM0. . . . . . . . . . . . . . . . . .  00000000 A
XMM1. . . . . . . . . . . . . . . . . .  00000001 A  XMM2. . . . . . . . . . . . . . . . . .  00000002 A
XRAMEND . . . . . . . . . . . . . . . .  0000FFFF A  Y12 . . . . . . . . . . . . . . . . . .  00000020 A
ya0 . . . . . . . . . . . . . . . . . .  0000A675 A  YH. . . . . . . . . . . . . . . . . . .  0000001D A
YL. . . . . . . . . . . . . . . . . . .  0000001C A  ZH. . . . . . . . . . . . . . . . . . .  0000001F A
ZL. . . . . . . . . . . . . . . . . . .  0000001E A

 Info - Memory segments organization (.OVERLAP)
        CSEG: Start = 0x00000018  End = 0x000000D2  Size = 186 word(s)
        CSEG: Start = 0x00000100  End = 0x00000200  Size = 256 word(s)
        CSEG: Start = 0x00000500  End = 0x00000518  Size = 24 word(s)
        CSEG: Start = 0x00000508  End = 0x00000510  Size = 8 word(s)
        CSEG: Start = 0x00001234  End = 0x00001392  Size = 350 word(s)
        CSEG: Start = 0x00004000  End = 0x00004008  Size = 8 word(s)
        CSEG: Start = 0x00006000  End = 0x00006008  Size = 8 word(s)
        CSEG: Start = 0x00007FFF  End = 0x00008007  Size = 8 word(s)
        CSEG: Start = 0x00009000  End = 0x00009009  Size = 9 word(s)
        CSEG: Start = 0x0000FB00  End = 0x0000FB08  Size = 8 word(s)
        CSEG: Start = 0x0000FFFF  End = 0x00010007  Size = 8 word(s)
        CSEG: Start = 0x00011000  End = 0x00011002  Size = 2 word(s)
        CSEG: Code size = 1750 bytes

        DSEG: Start = 0x00000200  End = 0x00000218  Size = 24 byte(s)
        DSEG: Start = 0x00000300  End = 0x00000308  Size = 8 byte(s)
        DSEG: Data size = 32 bytes

        ESEG: Start = 0x00000000  End = 0x00000034  Size = 52 byte(s)
        ESEG: Data size = 52 bytes

 Info - Used instructions below are missing in some AVR MicroChips.
        See the specific uC Data Sheet to confirm the instructions.
        ---------   ---------   ---------   ---------   ---------
        des         lac         las         lat         xch
        ---------   ---------   ---------   ---------   ---------

ASSEMBLY COMPLETE,   NO ERRORS
```

9. XASMAVR Listing with Address Range Overlap

Example (.OVERLAP / .MODEL BYTE)

```
AVR Macro-Assembler, Version 2.1                         27/10/2024  PAGE   101
Memory range beyond 64K (>10000h)

LOC       OBJ                    LINE  SOURCE

                                  15  .OVERLAP
00000200                 :C     16  .org $100
00000200  54 45 53 54    :C     17  .DB "TEST"
0000E000                 :C     18  .org $7000
0000E000  77 77 77 77 77 :C     19  .DB $77,$77,$77,$77,$77,$77,$77,$77,$77,$77,$77,$77,$77,$77,$77,$77
0000E005  77 77 77 77 77 :C
0000E00A  77 77 77 77 77 :C
0000E00F  77             :C
00008000                 :C     20  .org $4000
00008000  44 44 44 44 44 :C     21  .DB $44,$44,$44,$44,$44,$44,$44,$44,$44,$44,$44,$44,$44,$44,$44,$44
00008005  44 44 44 44 44 :C
0000800A  44 44 44 44 44 :C
0000800F  44             :C
00000A10                 :C     22  .org $508                                       ;=Overlap test
00000A10  58 58 58 58 58 :C     23  .DB $58,$58,$58,$58,$58,$58,$58,$58,$58,$58,$58,$58,$58,$58,$58,$58
00000A15  58 58 58 58 58 :C
00000A1A  58 58 58 58 58 :C
00000A1F  58             :C
00000A00                 :C     24  .org $500                                       ;=Overlap test
00000A00  55 55 55 55 55 :C     25  .DB $55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55
00000A05  55 55 55 55 55 :C
00000A0A  55 55 55 55 55 :C
00000A0F  55             :C
00000A10  55 55 55 55 55 :C     26  .DB $55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55,$55
00000A15  55 55 55 55 55 :C
00000A1A  55 55 55 55 55 :C
00000A1F  55             :C
0000FFFE                 :C     28  .org $7FFF                                      ;=$0FFFE
0000FFFE  EEEE           :C     29  .DW $EEEE                                       ;=:02 FFFE 00 EEEE 25
00010000  EEEE           :C     30  .DW $EEEE,$EEEE,$EEEE,$EEEE,$EEEE,$EEEE,$EEEE   ;=:0E 0000 00
00010002  EEEE           :C
00010004  EEEE           :C
00010006  EEEE           :C
00010008  EEEE           :C
0001000A  EEEE           :C
0001000C  EEEE           :C
00012000                 :C     32  .org $9000                                      ;=$12000
00012000  99 99 99 99 99 :C     33  .DB $99,$99,$99,$99,$99,$99,$99,$99,$99,$99,$99,$99,$99,$99,$99,$99
00012005  99 99 99 99 99 :C
0001200A  99 99 99 99 99 :C
0001200F  99             :C
                                34
0001F600                 :C     35  .org $FB00                                      ;=$1F600
0001F600  BB BB BB BB BB :C     36  .DB $BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB,$BB
0001F605  BB BB BB BB BB :C
0001F60A  BB BB BB BB BB :C
0001F60F  BB             :C
0001FFFE                 :C     38  .org $FFFF                                      ;=$1FFFE
0001FFFE  FFFF           :C     39  .DW $FFFF                                       ;=:02 FFFE 00 FFFF 03
00020000  FFFF           :C     40  .DW $FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF   ;=:0E 0000 00
00020002  FFFF           :C
00020004  FFFF           :C
00020006  FFFF           :C
00020008  FFFF           :C
0002000A  FFFF           :C
0002000C  FFFF           :C
0002000E                 :C     41  __FAR_away:
                                42
00022000                 :C     43  .org $11000                                     ;=$22000
00022000  55AA           :C     44  .DW $55AA, $AA55
00022002  AA55           :C
                                45
0003FFF0                 :C     46  .org $1FFF8                                     ;=$3FFF0
0003FFF0  2121           :C     47  .DW $2121, $2121, $2121
0003FFF2  2121           :C
0003FFF4  2121           :C
0003FFF6                 :C     48  _ExitFar:
                                49  .EXIT
```

```
__FAR_away. . . . . . . . . . . . . . . .  0002000E C  _ExitFar. . . . . . . . . . . . . . . .  0003FFF6 C

 Info - Memory segments organization (.OVERLAP)
        CSEG: Start = 0x00000200  End = 0x00000204  Size = 4 byte(s)
        CSEG: Start = 0x00000A00  End = 0x00000A20  Size = 32 byte(s)
        CSEG: Start = 0x00000A10  End = 0x00000A20  Size = 16 byte(s)
        CSEG: Start = 0x00008000  End = 0x00008010  Size = 16 byte(s)
        CSEG: Start = 0x0000E000  End = 0x0000E010  Size = 16 byte(s)
        CSEG: Start = 0x0000FFFE  End = 0x0001000E  Size = 16 byte(s)
        CSEG: Start = 0x00012000  End = 0x00012010  Size = 16 byte(s)
        CSEG: Start = 0x0001F600  End = 0x0001F610  Size = 16 byte(s)
        CSEG: Start = 0x0001FFFE  End = 0x0002000E  Size = 16 byte(s)
        CSEG: Start = 0x00022000  End = 0x00022004  Size = 4 byte(s)
        CSEG: Start = 0x0003FFF0  End = 0x0003FFF6  Size = 6 byte(s)
        CSEG: Code size = 158 bytes

        DSEG: Start = 0x00002000  End = 0x00002000  Size = 0 byte(s)
        DSEG: Data size = 0 bytes

        ESEG: Start = 0x00001000  End = 0x00001000  Size = 0 byte(s)
        ESEG: Data size = 0 bytes

ASSEMBLY COMPLETE,   NO ERRORS
```

Hex-File (.OVERLAP)

```
:020000020000FC
:0402000054455354BA
:10E0000077777777777777777777777777777777A0
:108000004444444444444444444444444444444430
:100A1000585858585858585858585858585858585856
:100A0000555555555555555555555555555555555596
:100A1000555555555555555555555555555555555586
:02FFFE00EEEE25
:020000021000EC
:0E000000EEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
:102000009999999999999999999999999999999940
:10F60000BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB4A
:02FFFE00FFFF03
:020000022000DC
:0E000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF00
:04200000AA5555AADE
:020000023000CC
:06FFF000212121212121 45
:00000001FF
```

Example (.NOOVERLAP / .MODEL WORD)

```
_AVR Macro-Assembler, Version 2.1                       27/10/2024   PAGE   101
Memory range beyond 64K (>10000h)

 LOC       OBJ                  LINE   SOURCE

 00000100                 :C      16   .org $100
 00000100  4554 5453      :C      17   .DB "TEST"
 . . . .                       . . . . . . . . .
 0001fffb                 :C      48   _ExitFar:
                                  49   .EXIT

USER SYMBOLS

__FAR_away. . . . . . . . . . . . . . . .  00010007 C  _ExitFar. . . . . . . . . . . . . . . .  0001FFFB C

 Info - Memory segments organization
        CSEG: Start = 0x00000100  End = 0x00000102  Size = 2 word(s)
        CSEG: Start = 0x00000500  End = 0x00000510  Size = 16 word(s)
>>>_avrtest01.ASM: |ERROR| ORG directive misplaced (segment overlap)
        CSEG: Start = 0x00000508  End = 0x00000510  Size = 8 word(s)
        CSEG: Code size = 36 bytes

        DSEG: Start = 0x00002000  End = 0x00002000  Size = 0 byte(s)
        DSEG: Data size = 0 bytes

        ESEG: Start = 0x00001000  End = 0x00001000  Size = 0 byte(s)
        ESEG: Data size = 0 bytes

ASSEMBLY COMPLETE  ***    1 ERROR(S), (    49)
```