```

AVRASM ver. 2.2.7  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr\bcd_to_hex_mux_intr\main.asm Tue Nov 10 18:42:51 2020

E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr\bcd_to_hex_mux_intr\main.asm(37): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr\bcd_to_hex_mux_intr\main.asm(417): warning: Register r14 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr\bcd_to_hex_mux_intr\main.asm(418): warning: Register r15 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr\bcd_to_hex_mux_intr\main.asm(37): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'


                                 ;*
                                 ;* Title: bcd_to_hex_mux_intr
                                 ;* Author: Judah Ben-Eliezer
                                 ;* Version:    1.0
                                 ;* Last updated:   11/10/2020
                                 ;* Target:         ;ATmega4809 @3.3MHz
                                 ;*
                                 ;* DESCRIPTION
                                 ;* This program polls the flag associated with pushbutton 1. This flag is
                                 ;* connected to PE0. If the flag is set, the contents of the array bcd_entries
                                 ;* is shifted left and the BCD digit set on the least significant 4 bits of
                                 ;* PORTA_IN are stored in the least significant byte of the bcd_entries array.
                                 ;* Then the corresponding segment values for each digit in the bcd_entries
                                 ;* display are written into the led_display. Note: entry of a non-BCD value
                                 ;* is ignored.
                                 ;*
                                 ;* This program also continually multiplexes the display so that the digits
                                 ;* entered are constantly seen on the display. Before any digits are entered
                                 ;* the display displays 0000.
                                 ;*
                                 ;* This program also polls the flag associated with pushbutton 2. This flag
                                 ;* is connected to PE2. If the flag is set,
                    the digits in the bcd_entries
                                 ;* array are read and passed to the prewritten subroutine BCD2bin16. This
                     ;* subroutine performs a BCD to binary conversion. The binary result is
                                 ;* partitioned into hexadecimal and placed into the array hex_results. The
                                 ;* contents of the hex_results array is converted to seven segment values
                                 ;* and placed into the led_display array. The multiplexing then causes
                                 ;* the hexadecimal equivalent of the BCD value entered to be displayed in
                                 ;* hexadecimal.
                                 ;*
                                 ;* VERSION HISTORY
                                 ;* 1.0 Original version
                                 ;******************************************************************************

                                 .list

                                 .equ PERIOD_EXAMPLE_VALUE = 325

                                 .dseg
002800           bcd_entries: .byte 4
002804           led_display: .byte 4
002808           digit_num: .byte 1
002809           hex_results: .byte 4

                                 .cseg

                                 .org TCA0_OVF_vect
00000e 940c 008a    jmp mux_display_ISR     ;vector for all TCA0_OVF pin change IRQs

                                 .org PORTE_PORT_vect
000046 940c 007a    jmp porte_ISR       ;vector for all PORTE pin change IRQs

                                 start:
000048 9880         cbi VPORTE_DIR, 0
000049 9882         cbi VPORTE_DIR, 2
00004a e000         ldi r16, $00
00004b b900         out VPORTA_DIR, r16
00004c 9500         com r16
00004d b90c         out VPORTD_DIR, r16
00004e b908         out VPORTC_DIR, r16
00004f e2b8         ldi XH, HIGH(bcd_entries)
```

```
000050 e0a0                         ldi XL, LOW(bcd_entries)
000051 e2d8                         ldi YH, HIGH(led_display)
000052 e0c4                         ldi YL, LOW(led_display)
000053 9500                         com r16
000054 930d                         st X+, r16
000055 930d                         st X+, r16
000056 930d                         st X+, r16
000057 930c                         st X, r16
000058 e0a0                         ldi XL, LOW(bcd_entries)

                                    ;Configure interrupts
000059 9100 0490                    lds r16, PORTE_PIN0CTRL ;set ISC for PE0 to pos. edge
00005b 6002                         ori r16, 0x02       ;set ISC for rising edge
00005c 9300 0490                    sts PORTE_PIN0CTRL, r16

00005e 9100 0492                    lds r16, PORTE_PIN2CTRL ;set ISC for PE2 to pos. edge
000060 6002                         ori r16, 0x02       ;set ISC for rising edge
000061 9300 0492                    sts PORTE_PIN2CTRL, r16

                                    ;configure TCA0
000063 e000                         ldi r16, TCA_SINGLE_WGMODE_NORMAL_gc ;WGMODE normal
000064 9300 0a01                    sts TCA0_SINGLE_CTRLB, r16

                                    ;enable overflow interrupt
000066 e001                         ldi r16, TCA_SINGLE_OVF_bm
000067 9300 0a0a                    sts TCA0_SINGLE_INTCTRL, r16

                                    ;load period low byte then high byte
000069 e405                         ldi r16, LOW(PERIOD_EXAMPLE_VALUE)
00006a 9300 0a26                    sts TCA0_SINGLE_PER, r16
00006c e001                         ldi r16, HIGH(PERIOD_EXAMPLE_VALUE)
00006d 9300 0a27                    sts TCA0_SINGLE_PER + 1, r16

                                    ;set clock and start timer
00006f e00d                         ldi r16, TCA_SINGLE_CLKSEL_DIV256_gc | TCA_SINGLE_ENABLE_bm
000070 9300 0a00                    sts TCA0_SINGLE_CTRLA, r16

000072 9100 0a0a                    lds r16, TCA0_SPLIT_INTCTRL
000074 6001                         ori r16, 0x01
000075 9300 0a0a                    sts TCA0_SPLIT_INTCTRL, r16

000077 9478                         sei         ;enable global interrupts

```

```
                                   main_loop:
000078 0000                           nop
000079 cffe                           rjmp main_loop

                                   ;Interrupt service routine for any PORTE pin change IRQ
                                   porte_ISR:
00007a 930f                           push r16         ;save r16 then SREG
00007b b70f                           in r16, CPU_SREG
00007c 930f                           push r16
00007d 94f8                           cli              ;clear global interrupt enable

                                      ;Determine which pins of PORTE have IRQs
00007e 9100 0489                      lds r16, PORTE_INTFLAGS ;check for PE0 IRQ flag set
000080 fd00                           sbrc r16, 0
000081 d013                           rcall PB1_sub           ;execute subroutine for PE0

000082 9100 0489                      lds r16, PORTE_INTFLAGS ;check for PE2 IRQ flag set
000084 fd02                           sbrc r16, 2
000085 d019                           rcall PB2_sub           ;execute subroutine for PE2


000086 910f                           pop r16          ;restore SREG then r16
000087 bf0f                           out CPU_SREG, r16
000088 910f                           pop r16
000089 9518                           reti             ;return from PORTE pin change ISR

                                   ;Interrupt service routine for any overflow of counter
                                   mux_display_ISR:
00008a 930f                           push r16
00008b b70f                           in r16, CPU_SREG
00008c 930f                           push r16

00008d d06d                           rcall multiplex_display     ;multiplexes display

00008e e001                           ldi r16, TCA_SINGLE_OVF_bm   ;clear OVF flag
00008f 9300 0a0b                      sts TCA0_SINGLE_INTFLAGS, r16

000091 910f                           pop r16          ;restore SREG then r16
000092 bf0f                           out CPU_SREG, r16
000093 910f                           pop r16
```

```
000094 9518                          reti

                                  pb1_sub:
000095 d045                          rcall reverse_bits
000096 301a                          cpi r17, $0A
000097 f430                          brsh non_bcd
000098 d048                          rcall shift_bcd_entries
000099 d050                          rcall bcd_to_led
00009a e001                          ldi r16, PORT_INT0_bm   ;clear IRQ flag for 
   PE0
00009b 9300 0489                     sts PORTE_INTFLAGS, r16
00009d 9508                          ret

                                  non_bcd:
00009e 9518                          reti

                                  pb2_sub:
00009f e2b8                          ldi XH, HIGH(bcd_entries)
0000a0 e0a0                          ldi XL, LOW(bcd_entries)
0000a1 e020                          ldi r18, $00
0000a2 911d                          ld r17, X+
0000a3 9512                          swap r17
0000a4 913d                          ld r19, X+
0000a5 2b13                          or r17, r19
0000a6 910d                          ld r16, X+
0000a7 9502                          swap r16
0000a8 913c                          ld r19, X
0000a9 2b03                          or r16, r19
0000aa d07d                          rcall BCD2bin16

                                     ;load_to_hex_results
0000ab e2b8                          ldi XH, HIGH(hex_results)
0000ac e0a9                          ldi XL, LOW(hex_results)
0000ad e030                          ldi r19, $00
0000ae 293f                          or r19, tbinH
0000af 7f30                          andi r19, $F0
0000b0 9532                          swap r19
0000b1 933d                          st X+, r19
0000b2 e030                          ldi r19, $00
0000b3 293f                          or r19, tbinH
0000b4 703f                          andi r19, $0F
0000b5 933d                          st X+, r19
0000b6 e030                          ldi r19, $00
0000b7 293e                          or r19, tbinL
0000b8 7f30                          andi r19, $F0
0000b9 9532                          swap r19
0000ba 933d                          st X+, r19
0000bb e030                          ldi r19, $00
0000bc 293e                          or r19, tbinL
```

```
0000bd 703f                         andi r19, $0F
0000be 933d                         st X+, r19

                                    ;load to led_display
0000bf e2b8                         ldi XH, HIGH(led_display)
0000c0 e0a4                         ldi XL, LOW(led_display)
0000c1 e020                         ldi r18, $00
0000c2 292f                         or r18, tbinH
0000c3 7f20                         andi r18, $F0
0000c4 9522                         swap r18
0000c5 d06e                         rcall hex_to_7seg
0000c6 932d                         st X+, r18
0000c7 e020                         ldi r18, $00
0000c8 292f                         or r18, tbinH
0000c9 702f                         andi r18, $0F
0000ca d069                         rcall hex_to_7seg
0000cb 932d                         st X+, r18
0000cc e020                         ldi r18, $00
0000cd 292e                         or r18, tbinL
0000ce 7f20                         andi r18, $F0
0000cf 9522                         swap r18
0000d0 d063                         rcall hex_to_7seg
0000d1 932d                         st X+, r18
0000d2 e020                         ldi r18, $00
0000d3 292e                         or r18, tbinL
0000d4 702f                         andi r18, $0F
0000d5 d05e                         rcall hex_to_7seg
0000d6 932d                         st X+, r18
0000d7 e004                         ldi r16, PORT_INT2_bm   ;clear IRQ flag for PE2
0000d8 9300 0489                    sts PORTE_INTFLAGS, r16
0000da 9508                         ret

                                 ;***************************************************************************
                                 ;*
                                 ;* "reverse_bits" - Reverse Bits
                                 ;*
                                 ;* Description: Reverses the bit positions in a byte passed in. Bit 0
                                 ;* becomes bit 7, bit 6 becomes bit 1, and so on.
                                 ;*
                                 ;* Author:                 Judah Ben-Eliezer
                                 ;* Version:                    1.0
                                 ;* Last updated:           101120
                                 ;* Target:                 ATmega4809
                                 ;* Number of words:            8
                                 ;* Number of cycles:
```

```
                        ;* Low registers modified:  r16, r17, r18
                        ;* High registers modified:     none
                        ;*
                        ;* Parameters: r16: byte to be reversed.
                        ;* Returns: r16: reversed byte
                        ;*
                        ;* Notes:
                        ;*
                        ;*******************************************************************************

                        reverse_bits:
0000db e028                ldi r18, $08
                        loop_8:
0000dc 9507                ror r16
0000dd 1f11                rol r17
0000de 952a                dec r18
0000df f7e1                brne loop_8
0000e0 9508                ret

                        shift_bcd_entries:
0000e1 e023                ldi r18, $03
                        shift_loop:
0000e2 e2b8                ldi XH, HIGH(bcd_entries)
0000e3 e0a0                ldi XL, LOW(bcd_entries)
0000e4 952a                dec r18
0000e5 0fa2                add XL, r18
0000e6 913d                ld r19, X+
0000e7 933c                st X, r19
0000e8 f7c9                brne shift_loop
0000e9 9508                ret

                        ;*******************************************************************************
                        ;*
                        ;* "bcd_to_led" - title
                        ;*
                        ;* Description:    Converts bcd input to 7seg output, from bcd_entries array to led_display array
                        ;*
                        ;* Author:
                        ;* Version:
                        ;* Last updated:
                        ;* Target:
                        ;* Number of words:
                        ;* Number of cycles:
                        ;* Low registers modified:
                        ;* High registers modified:
                        ;*
```

```
                          ;* Parameters:
                          ;* Returns:
                          ;*
                          ;* Notes:
                          ;*
                          ;***************************************************************************

                          bcd_to_led:
0000ea e0a0                  ldi XL, LOW(bcd_entries)
0000eb e2b8                  ldi XH, HIGH(bcd_entries)
0000ec 931c                  st X, r17
0000ed e044                  ldi r20, $04
                          conversion_loop:
0000ee 954a                  dec r20
0000ef e2b8                  ldi XH, HIGH(bcd_entries)
0000f0 e0a0                  ldi XL, LOW(bcd_entries)
0000f1 e2d8                  ldi YH, HIGH(led_display)
0000f2 e0c4                  ldi YL, LOW(led_display)
0000f3 0fa4                  add XL, r20
0000f4 0fc4                  add YL, r20
0000f5 912c                  ld r18, X
0000f6 d03d                  rcall hex_to_7seg
0000f7 8328                  st Y, r18
0000f8 3040                  cpi r20, $00
0000f9 f7a1                  brne conversion_loop
0000fa 9508                  ret


                          ;***************************************************************************
                          ;*
                          ;* "multiplex_display" - title
                          ;*
                          ;* Description:    outputs values from led_display array to 7 segment display on PORTD driven by highest two bits of PORTC
                          ;*
                          ;* Author: Judah Ben-Eliezer
                          ;* Version:    1.0
                          ;* Last updated:    11/10/2020
                          ;* Target: ATmega4809
                          ;* Number of words:
                          ;* Number of cycles:
                          ;* Low registers modified:
                          ;* High registers modified:
                          ;*
                          ;* Parameters:
                          ;* Returns:
```

```
                          ;*
                          ;* Notes:
                          ;*
                          ;****************************************************************************

                          multiplex_display:
0000fb e064                  ldi r22, $04
0000fc e070                  ldi r23, $00
0000fd 9370 2808             sts digit_num, r23
                          loop_4:
0000ff e0c4                  ldi YL, LOW(led_display)
000100 9110 2808             lds r17, digit_num
000102 9140 2808             lds r20, digit_num
000104 7013                  andi r17, $03
000105 0fc1                  add YL, r17
000106 8128                  ld r18, Y
000107 e850                  ldi r21, $80
000108 9543                  inc r20
                          loop:
000109 9556                  lsr r21
00010a 954a                  dec r20
00010b f7e9                  brne loop
00010c 0f55                  lsl r21
00010d 9550                  com r21
00010e b959                  out VPORTC_OUT, r21
00010f b92d                  out VPORTD_OUT, r18
000110 9513                  inc r17
000111 9310 2808             sts digit_num, r17
000113 956a                  dec r22
000114 f751                  brne loop_4
000115 9508                  ret

                          ;****************************************************************************
                          ;*
                          ;* "BCD2bin16" - BCD to 16-Bit Binary Conversion
                          ;*
                          ;* This subroutine converts a 5-digit packed BCD number represented by
                          ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit number (tbinH:tbinL).
                          ;* MSD of the 5-digit number must be placed in the lowermost nibble of fBCD2.
                          ;*
                          ;* Let "abcde" denote the 5-digit number. The conversion is done by
                          ;* computing the formula: 10(10(10(10a+b)+c)
              +d)+e.
                          ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add operation
                          ;* which is repeated four times during the computation.
                          ;*
                          ;* Number of words  :30
                          ;* Number of cycles     :108
                          ;* Low registers used  :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
                          ;* High registers used  :4 (fBCD0,fBCD1,fBCD2,adder)
                          ;*
                          ;*************************************************************************************

                          ;***** "mul10a"/"mul10b" Subroutine Register Variables

                          .def    copyL   =r12        ;temporary register
                          .def    copyH   =r13        ;temporary register
                          .def    mp10L   =r14        ;Low byte of number to be multiplied by 10
                          .def    mp10H   =r15        ;High byte of number to be multiplied by 10
                          .def    adder   =r19        ;value to add after multiplication

                          ;***** Code

                          mul10a:    ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high nibble
000116 9532                      swap    adder
                          mul10b:    ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low nibble
000117 2cce                      mov copyL,mp10L ;make copy
000118 2cdf                      mov copyH,mp10H
000119 0cee                      lsl mp10L       ;multiply original by 2
00011a 1cff                      rol mp10H
00011b 0ccc                      lsl copyL       ;multiply copy by 2
00011c 1cdd                      rol copyH
00011d 0ccc                      lsl copyL       ;multiply copy by 2 (4)
00011e 1cdd                      rol copyH
00011f 0ccc                      lsl copyL       ;multiply copy by 2 (8)
000120 1cdd                      rol copyH
000121 0cec                      add mp10L,copyL ;add copy to original
000122 1cfd                      adc mp10H,copyH
000123 703f                      andi    adder,0x0f   ;mask away upper nibble of adder
```

```
000124 0ee3                         add mp10L,adder ;add lower nibble of adder
000125 f408                         brcc    m10_1       ;if carry not cleared
000126 94f3                         inc mp10H       ;   inc high byte
000127 9508                      m10_1: ret

                                 ;***** Main Routine Register Variables

                                 .def    tbinL   =r14        ;Low byte of binary result (same as mp10L)
                                 .def    tbinH   =r15        ;High byte of binary result (same as mp10H)
                                 .def    fBCD0   =r16        ;BCD value digits 1 and 0
                                 .def    fBCD1   =r17        ;BCD value digits 2 and 3
                                 .def    fBCD2   =r18        ;BCD value digit 5

                                 ;***** Code

                                 BCD2bin16:
000128 702f                         andi    fBCD2,0x0f  ;mask away upper nibble of fBCD2
000129 24ff                         clr mp10H
00012a 2ee2                         mov mp10L,fBCD2 ;mp10H:mp10L = a
00012b 2f31                         mov adder,fBCD1
00012c dfe9                         rcall   mul10a      ;mp10H:mp10L = 10a+b
00012d 2f31                         mov adder,fBCD1
00012e dfe8                         rcall   mul10b      ;mp10H:mp10L = 10(10a+b)+c
00012f 2f30                         mov adder,fBCD0
000130 dfe5                         rcall   mul10a      ;mp10H:mp10L = 10(10(10a+b)+c)+d
000131 2f30                         mov adder,fBCD0
000132 dfe4                         rcall   mul10b      ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
000133 9508                         ret



                                 ;***************************************************************************
                                 ;*
                                 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
                                 ;*
                                 ;* Description: Converts a right justified hexadecimal digit to the seven
                                 ;* segment pattern required to display it. Pattern is right justified a
```

```
                          ;* through g. Pattern uses 0s to turn segments on ON.
                          ;*
                          ;* Author:                       Ken Short
                          ;* Version:                          1.0
                          ;* Last updated:                 101620
                          ;* Target:                       ATmega4809
                          ;* Number of words:                  8
                          ;* Number of cycles:             13
                          ;* Low registers modified:       none
                          ;* High registers modified:          r16, r18, ZL, ZH
                          ;*
                          ;* Parameters: r18: right justified hex digit, high nibble 0
                          ;* Returns: r18: segment values a through g    right justified
                          ;*
                          ;* Notes:
                          ;*
                          ;***************************************************************************

                          hex_to_7seg:
000134 702f                   andi r18, 0x0F              ;clear ms       nibble
000135 e0f2                    ldi ZH, HIGH(hextable * 2)  ;set Z to       point to start of table
000136 e7e8                    ldi ZL, LOW(hextable * 2)
000137 e000                    ldi r16, $00                ;add offset to  Z pointer
000138 0fe2                    add ZL, r18
000139 1ff0                    adc ZH, r16
00013a 9124                    lpm r18, Z                  ;load byte      from table pointed to by Z
00013b 9508                   ret

                               ;Table of segment values to display digits 0 - F
                               ;!!! seven values must be added - verify    all values
00013c 4f01
00013d 0612
00013e 244c
00013f 0f20
000140 0400
000141 6008
000142 4231
```

```


RESOURCE USE INFORMATION
------------------------

Notice:
The register and instruction counts are symbol table hit counts,
and hence implicitly used resources are not counted, eg, the
'lpm' instruction without operands implicitly uses r0 and z,
none of which are counted.

x,y,z are separate entities in the symbol table and are
counted separately from r26..r31 here.

.dseg memory usage only counts static data declared with .byte

"ATmega4809" register use summary:
x  :  20 y  :   2 z  :   1 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   5
r13:   5 r14:   9 r15:   9 r16:  59 r17:  13 r18:  33 r19:  31 r20:   8
r21:   5 r22:   2 r23:   2 r24:   0 r25:   0 r26:  10 r27:   7 r28:   5
r29:   2 r30:   2 r31:   2
Registers used: 21 out of 35 (60.0%)

"ATmega4809" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   2 add   :   7 adiw  :   0 and   :   0
andi  :  12 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   1 brcs  :   0 break :   0 breq  :   0 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   5 brpl  :   0 brsh  :   1 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   2 cbr   :   0
clc   :   0 clh   :   0 cli   :   1 cln   :   0 clr   :   1 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   3 cp    :   0 cpc   :   0
cpi   :   2 cpse  :   0 dec   :   5 des   :   0 eor   :   0 fmul  :   0
fmuls :   0 fmulsu:   0 icall :   0 ijmp  :   0 in    :   2 inc   :   3
jmp   :   2 ld    :   7 ldd   :   0 ldi   :  47 lds   :   7 lpm   :   2
lsl   :   5 lsr   :   1 mov   :   7 movw  :   0 mul   :   0 muls  :   0
mulsu :   0 neg   :   0 nop   :   1 or    :  10 ori   :   3 out   :   7
pop   :   4 push  :   4 rcall :  16 ret   :   9 reti  :   3 rjmp  :   1
rol   :   5 ror   :   1 sbc   :   0 sbci  :   0 sbi   :   0 sbic  :   0
sbis  :   0 sbiw  :   0 sbr   :   0 sbrc  :   2 sbrs  :   0 sec   :   0
seh   :   0 sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   0
sev   :   0 sez   :   0 sleep :   0 spm   :   0 st    :  15 std   :   0
sts   :  13 sub   :   0 subi  :   0 swap  :   7 tst   :   0 wdr   :   0

Instructions used: 39 out of 114 (34.2%)

"ATmega4809" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
```

```
---------------------------------------------------------------
[.cseg] 0x00001c 0x000288    496     16     512   49152   1.0%
[.dseg] 0x002800 0x00280d      0     13      13    6144   0.2%
[.eseg] 0x000000 0x000000      0      0       0     256   0.0%

Assembly complete, 0 errors, 2 warnings

```