```
/* File: main_C.c
// A simple C program that calls an subroutine defined external to this
 source module.
//
*/
#include <avr/io.h>
#include <avr/interrupt.h>


// These are global variables available to any function

// int: 2 bytes each, whether initialized or not
 int w;
 int x;
 int y=3;
 int z;
 char msg[]="Hello"; // 6 bytes including null terminator

int main()          /* every C program needs a "main" */
{

// These are local variables limited in scope to "main"
      int i1 = 1;
      int i2 = 2;
      int i3 = 3;
      int i4 = 4;
      int i5 = 5;

      i1 = w;
      i2 = x;
      i3 = y;
      i4 = z;
      i5 = msg[0];


      return 0xAB12;
}

//ISR Macro used to indicate to GCC that this is an ISR
ISR(INT1_vect) {
    PORTB = 0xFF;
}
```

```
  68:   f1 e0           ldi r31, 0x01   ; 1
  6a:   02 c0           rjmp    .+4         ; 0x70 <.do_copy_data_start>

0000006c <.do copy data loop>:
  6c:   05 90           lpm r0, Z+
  6e:   0d 92           st  X+, r0

00000070 <.do copy data start>:
  70:   a8 36           cpi r26, 0x68   ; 104
  72:   b1 07           cpc r27, r17
  74:   d9 f7           brne    .-10        ; 0x6c <.do_copy_data_loop>

00000076 <  do clear bss>:
  76:   10 e0           ldi r17, 0x00   ; 0
  78:   a8 e6           ldi r26, 0x68   ; 104
  7a:   b0 e0           ldi r27, 0x00   ; 0
  7c:   01 c0           rjmp    .+2         ; 0x80 <.do_clear_bss_start>

0000007e <.do clear bss loop>:
  7e:   1d 92           st  X+, r1

00000080 <.do clear bss start>:
  80:   ae 36           cpi r26, 0x6E   ; 110
  82:   b1 07           cpc r27, r17
  84:   e1 f7           brne    .-8         ; 0x7e <.do_clear_bss_loop>
  86:   0e 94 49 00     call    0x92    ; 0x92 <main>
  8a:   0c 94 aa 00     jmp 0x154   ; 0x154 <_exit>

0000008e <  bad interrupt>:
  8e:   0c 94 00 00     jmp 0   ; 0x0 <__vectors>

00000092 <main>:
  92:   cf 93           push    r28
  94:   df 93           push    r29
  96:   cd b7           in  r28, 0x3d   ; 61
  98:   de b7           in  r29, 0x3e   ; 62
  9a:   2a 97           sbiw    r28, 0x0a   ; 10
  9c:   0f b6           in  r0, 0x3f    ; 63
  9e:   f8 94           cli
  a0:   de bf           out 0x3e, r29   ; 62
  a2:   0f be           out 0x3f, r0    ; 63
  a4:   cd bf           out 0x3d, r28   ; 61
  a6:   81 e0           ldi r24, 0x01   ; 1
  a8:   90 e0           ldi r25, 0x00   ; 0
  aa:   9a 87           std Y+10, r25   ; 0x0a
  ac:   89 87           std Y+9, r24    ; 0x09
  ae:   82 e0           ldi r24, 0x02   ; 2
  b0:   90 e0           ldi r25, 0x00   ; 0
  b2:   98 87           std Y+8, r25    ; 0x08
  b4:   8f 83           std Y+7, r24    ; 0x07
  b6:   83 e0           ldi r24, 0x03   ; 3
  b8:   90 e0           ldi r25, 0x00   ; 0
  ba:   9e 83           std Y+6, r25    ; 0x06
  bc:   8d 83           std Y+5, r24    ; 0x05
  be:   84 e0           ldi r24, 0x04   ; 4
  c0:   90 e0           ldi r25, 0x00   ; 0
  c2:   9c 83           std Y+4, r25    ; 0x04
  c4:   8b 83           std Y+3, r24    ; 0x03
  c6:   85 e0           ldi r24, 0x05   ; 5
  c8:   90 e0           ldi r25, 0x00   ; 0
  ca:   9a 83           std Y+2, r25    ; 0x02
  cc:   89 83           std Y+1, r24    ; 0x01
  ce:   80 91 6a 00     lds r24, 0x006A
  d2:   90 91 6b 00     lds r25, 0x006B
  d6:   9a 87           std Y+10, r25   ; 0x0a
```

copy from Flash to SRAM

copy from 0x60 to 0x67

0x68 to 0x6D is cleared (set w, x, z to 0)

copy SP to Y

decr SP by 10 bytes

copy SREG to r0 & clear interrupts

reset SP & SREG (SP decremented)

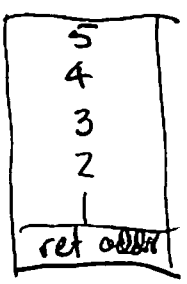
"Push" 1 to stack

"Push" 2 to stack

etc

addr of w

il = w



```
GNUGCC.elf:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000156  00000000  00000000  00000074  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  00800060  00000156  000001ca  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000006  00800068  0000015e  000001d2  2**0
                  ALLOC
  3 .stab         00000378  00000000  00000000  000001d4  2**2
                  CONTENTS, READONLY, DEBUGGING
  4 .stabstr      0000005f  00000000  00000000  0000054c  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 00000020  00000000  00000000  000005ab  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_pubnames 0000004a  00000000  00000000  000005cb  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_info   000001f1  00000000  00000000  00000615  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_abbrev 00000086  00000000  00000000  00000806  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_line   000000c0  00000000  00000000  0000088c  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_frame  00000030  00000000  00000000  0000094c  2**2
                  CONTENTS, READONLY, DEBUGGING
Disassembly of section .text:

00000000 <__vectors>:
   0:   0c 94 2a 00     jmp 0x54        ; 0x54 <__ctors_end>
   4:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
   8:   0c 94 90 00     jmp 0x120       ; 0x120 <__vector_2>
   c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  10:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  14:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  18:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  1c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  20:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  24:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  28:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  2c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  30:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  34:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  38:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  3c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  40:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  44:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  48:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  4c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  50:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>

00000054 <__ctors_end>:
  54:   11 24           eor r1, r1
  56:   1f be           out 0x3f, r1    ; 63
  58:   cf e3           ldi r28, 0x3F   ; 63
  5a:   d8 e0           ldi r29, 0x08   ; 8
  5c:   de bf           out 0x3e, r29   ; 62
  5e:   cd bf           out 0x3d, r28   ; 61

00000060 <__do_copy_data>:
  60:   10 e0           ldi r17, 0x00   ; 0
  62:   a0 e6           ldi r26, 0x60   ; 96
  64:   b0 e0           ldi r27, 0x00   ; 0
  66:   e6 e5           ldi r30, 0x56   ; 86
```

clear SREG

init SPL, SPH

X = start addr of SRAM

Z = start addr of Flash data

see next pg

```
 d8:   89 87           std Y+9, r24     ; 0x09
 da:   80 91 68 00     lds r24, 0x0068
 de:   90 91 69 00     lds r25, 0x0069
 e2:   98 87           std Y+8, r25     ; 0x08
 e4:   8f 83           std Y+7, r24     ; 0x07
 e6:   80 91 60 00     lds r24, 0x0060
 ea:   90 91 61 00     lds r25, 0x0061
 ee:   9e 83           std Y+6, r25     ; 0x06
 f0:   8d 83           std Y+5, r24     ; 0x05
 f2:   80 91 6c 00     lds r24, 0x006C
 f6:   90 91 6d 00     lds r25, 0x006D
 fa:   9c 83           std Y+4, r25     ; 0x04
 fc:   8b 83           std Y+3, r24     ; 0x03
 fe:   80 91 62 00     lds r24, 0x0062
102:   88 2f           mov r24, r24
104:   90 e0           ldi r25, 0x00    ; 0
106:   9a 83           std Y+2, r25     ; 0x02
108:   89 83           std Y+1, r24     ; 0x01
10a:   82 e1           ldi r24, 0x12    ; 18
10c:   9b ea           ldi r25, 0xAB    ; 171
10e:   2a 96           adiw    r28, 0x0a   ; 10
110:   0f b6           in  r0, 0x3f     ; 63
112:   f8 94           cli
114:   de bf           out 0x3e, r29    ; 62
116:   0f be           out 0x3f, r0     ; 63
118:   cd bf           out 0x3d, r28    ; 61
11a:   df 91           pop r29
11c:   cf 91           pop r28
11e:   08 95           ret

00000120 <__vector_2>:
120:   1f 92           push    r1
122:   0f 92           push    r0
124:   0f b6           in  r0, 0x3f     ; 63
126:   0f 92           push    r0
128:   11 24           eor r1, r1
12a:   8f 93           push    r24
12c:   ef 93           push    r30
12e:   ff 93           push    r31
130:   cf 93           push    r28
132:   df 93           push    r29
134:   cd b7           in  r28, 0x3d    ; 61
136:   de b7           in  r29, 0x3e    ; 62
138:   e8 e3           ldi r30, 0x38    ; 56
13a:   f0 e0           ldi r31, 0x00    ; 0
13c:   8f ef           ldi r24, 0xFF    ; 255
13e:   80 83           st  Z, r24
140:   df 91           pop r29
142:   cf 91           pop r28
144:   ff 91           pop r31
146:   ef 91           pop r30
148:   8f 91           pop r24
14a:   0f 90           pop r0
14c:   0f be           out 0x3f, r0     ; 63
14e:   0f 90           pop r0
150:   1f 90           pop r1
152:   18 95           reti

00000154 <__exit>:
154:   ff cf           rjmp    .-2          ; 0x154 <__exit>
```

addr of x

i2 = x

i3 = y

i4 = z

i5 = msg[0]

load 0xAB12 to r24 r25

add 10 to SP

restore SP ("pop" local vars off stack)