
 EXAMPLE 6

Description:
Reference to segment start, end, etc.

Notes:
GCC allows to reference segment or region directly from inline assembler.

IAR relies on pragma directive to define a section name and the section operators `__section_begin`, `__section_end` and `__section_size`. (You need to know what you are doing !)

 Danger warning for IAR code: LOW

 GCC code example:

```
char * Test_Section(int incr)
{
  extern char end asm ("end"); /* Defined by the linker. */
  char *heap_end;

  heap_end = & end;

  return heap_end;
}
```

 GCC assembler output:

```

        .global      Test_Section
        .type        Test_Section,function
Test_Section:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 0, current_function_anonymous_args = 0
        @ link register save eliminated.
        ldr         r0, .L12
        mov         pc, lr
.L13:
        .align      2
.L12:
        .word       end
.Lfe5:
        .size       Test_Section,.Lfe5-Test_Section
```

 Equivalent IAR C code:

```
char * Test_Section(int incr)
{
  void * EndPtr;
  char *heap_end;
#pragma section="HEAP"
  EndPtr = __section_end("HEAP");

  heap_end = EndPtr;
  return heap_end;
}
```

IAR assembler output:

```
\                               In section .text, align 4, keep-with-next
112      char * Test_Section(int incr)
113      {
114          void * EndPtr;
115          char *heap_end;
116          #pragma section="HEAP"
117          EndPtr = __section_end("HEAP");
118
119          heap_end = EndPtr;
120          return heap_end;
\          Test_Section:
\      00000000      0048          LDR      R0,??Test_Section_0 ;; SFE(HEAP)
\      00000002      7047          BX      LR              ;; return
\          ??Test_Section_0:
\      00000004      .....          DC32      SFE(HEAP)
121      }
```