
EXAMPLE 2

Description:
Using SWI instruction.

Notes:
GCC compiler requires to use inline assembler. Parameter transfer is up to the application programmer who HAS to know what he is doing

IAR uses pragma directive and __SWI extended keyword. This solution is simple and elegant.

Danger warning for IAR code: NONE

GCC code example:

```
inline void swi_watchdog (void)
{
  asm volatile(" swi 0x07 ");
}

inline U32 swi_fiq_disable(void)
{
  unsigned long retval;
  asm volatile(" swi 0x02 ");
  asm volatile (" mov  %0, r0" : "=r" (retval) : /* no inputs */  );
  return retval;
}

void Test_SWI_Code(void)
{
  unsigned long swistate,fiqstate;

  swi_watchdog();
  fiqstate =  swi_fiq_disable();
}
```

GCC assembler output:

```
            .align     2
            .global    swi_watchdog
            .type      swi_watchdog,function
swi_watchdog:
            @ args = 0, pretend = 0, frame = 0
            @ frame_needed = 0, current_function_anonymous_args = 0
            @ link register save eliminated.
             swi 0x07
            mov        pc, lr
.Lfe2:
            .size      swi_watchdog,.Lfe2-swi_watchdog
            .align     2
            .global    swi_fiq_disable
            .type      swi_fiq_disable,function
swi_fiq_disable:
            @ args = 0, pretend = 0, frame = 0
            @ frame_needed = 0, current_function_anonymous_args = 0
            @ link register save eliminated.
             swi 0x02
             mov  r0, r0
            mov        pc, lr
```

Acknowledgment: Ivan Paulik @ PASAT, Bratislava

```
.Lfe3:
           .size      swi_fiq_disable,.Lfe3-swi_fiq_disable
           .align     2
           .global    Test_SWI_Code
           .type      Test_SWI_Code,function
Test_SWI_Code:
           @ args = 0, pretend = 0, frame = 0
           @ frame_needed = 0, current_function_anonymous_args = 0
           @ link register save eliminated.
            swi 0x07
            swi 0x02
            mov  r3, r0
           mov        pc, lr
.Lfe4:
           .size      Test_SWI_Code,.Lfe4-Test_SWI_Code
```

Equivalent IAR C code:

```
#pragma swi_number=0x07
__swi void swi_watchdog(void);

#pragma swi_number=0x02
__swi unsigned long swi_fiq_disable(void);

void Test_SWI_Code(void)
{
  unsigned long swistate,fiqstate;

  swi_watchdog();
  fiqstate =  swi_fiq_disable();
}
```

IAR assembler output:

```
     41          #pragma swi_number=0x07
     42          __swi void swi_watchdog(void);
     43
     44          #pragma swi_number=0x02
     45          __swi unsigned long swi_fiq_disable(void);
     46

   \                                In section .text, align 2, keep-with-next
     47          void Test_SWI_Code(void)
     48          {
   \                        Test_SWI_Code:
   \   00000000   80B5               PUSH     {R7,LR}
     49            unsigned long swistate,fiqstate;
     50
     51            swi_watchdog();
   \   00000002   07DF               SVC      0x7
     52            fiqstate =  swi_fiq_disable();
   \   00000004   02DF               SVC      0x2
     53          }
   \   00000006   01BD               POP      {R0,PC}          ;; return
```

Acknowledgment: Ivan Paulik @ PASAT, Bratislava