Application Note

Using embOS for Cortex-M in a bootloader

Document: AN01005
Revision: 0
Date: April 4, 2016

A product of SEGGER Microcontroller GmbH & Co. KG

www.segger.com

Disclaimer

Copyright notice

Trademarks

Manual versions

This manual describes the current software version. If any error occurs, inform us and we will try to assist you as soon as possible.
Contact us for further information on topics or routines not yet specified.

Print date: April 4, 2016

Revision	Date	By	Description
0	160404	RH	First version.

Introduction

This application note describes the usage of embOS within bootloader applications for Cortex-M devices.

A bootloader is a program which allows the stand-alone programming of application code to the device it is running on. The bootloader has to ensure that the CPU is in the same state as after reset if the application is started.

There are no restrictions for building a bootloader using embOS.

Jump from bootloader to application

Some steps have to be performed before the main application can be safely accessed from a task. For the following example, we assume that the bootloader is located at address 0x00000000 and the application at 0x00100000.

```
#define THUMB_BIT         1
#define APP_START_ADDR    0x00100000
#define APP_STACK_PTR     (*(volatile OS_U32*)(APP_START_ADDR + 0x00))
#define APP_RESET_PTR     (*(volatile OS_U32*)(APP_START_ADDR + 0x04))
void (*AppPtr)(void);

static void HPTask(void) {
  while (1) {
    //
    // Start the application
    //
    OS_IncDI();
    AppPtr     = (void (*)(void))(APP_RESET_PTR | THUMB_BIT);
    SCB->VTOR = APP_START_ADDR;
    __set_MSP(APP_STACK_PTR);  // Set main stack pointer to application
                               // initial stack value
    __set_CONTROL(0);          // Use MSP and Privileged in thread mode
    AppPtr();                  // Start the application, we will not return
                               // from this function
  }
}
```

The following description analyzes the task step by step:

```
#define THUMB_BIT         1
#define APP_START_ADDR    0x00100000
#define APP_STACK_PTR     (*(volatile OS_U32*)(APP_START_ADDR + 0x00))
#define APP_RESET_PTR     (*(volatile OS_U32*)(APP_START_ADDR + 0x04))
void (*AppPtr)(void);
```

After reset two words, the start value of the stack pointer and the reset vector, will be fetched from the beginning of the vector table, which is initially located at address 0x00000000. The reset vector is used to start the program execution from the reset vector address.

Both programs, the bootloader and the application, have their own vector table, so it is neccessary to change the stack pointer value and the reset vector manually before the application can be started.

```
static void HPTask(void) {
  while (1) {
    //
    // Start the application
    //
    OS_IncDI();
```

At first interrupts must be disabled, no interrupt initialized within the bootloader should affect the start of the application.

```
AppPtr    = (void (*)(void))(APP_RESET_PTR | THUMB_BIT);
SCB->VTOR = APP_START_ADDR;
```

A function pointer will be used to jump to the new reset vector address. To indicate that the accessed reset handler is thumb code, the LSB of the vector must be set to 1.

The CPU stores the address of the vector table in the vector table offset register (VTOR), it has to be changed accordingly from reset value to the start address of the application.

```
__set_MSP(APP_STACK_PTR);  // Set main stack pointer to application
                           // initial stack value
__set_CONTROL(0);          // Use MSP and Privileged in thread mode
```

The main stack pointer must be set to the initial value used for the application. Because embOS tasks use the process stack pointer, the control register must be set to switch to the main stack pointer. Furthermore, the thread mode should be set to privileged.

```
    AppPtr();                   // Start the application, we will not return
                                // from this function
  }
}
```

Calling the function finally starts the application.

Demo project

A simple demo project for the emPower board, which demonstrates these steps, can be downloaded from *http://segger.com*. The demo project located at the same website where you can download this application note.