STM32F429II-SK_OS3

Licensing is required when using any Micriµm software, regardless of the state of the software (Library or Full Source). This project is only meant for example purposes. For projects using Library versions of the software, please contact our Sales office to obtain the Full Source version at +1 (954) 217-2036.

STM32F429II-SK Example Project Read-Me

The provided example project for which this Read-Me was made utilizes the Freescale STM32F429II-SK (KSK-STM32F429II-JL) evaluation board from the ARM-Cortex-M4 Family. The MCU found on this development board conforms with the ARM_Cortex_M4 architecture.

Project Download

Download Link	Micrium_STM32F429II-SK_OS3.zip

Toolchain IDE Versions

IDE/Toolchain	Version
IAR EW for ARM	7.30
Atollic TrueSTUDIO	5.1
Keil uVision	5.11

Micriµm Product Versions

Product	Version
µC/CPU	1.30.01.01
µC/LIB	1.38.01
µC/OS-III	3.04.04

Hardware Setup

1. Have the board connected via the **JTAG** into the board debugging input (**JTAG**).
2. Provide 5V DC to power the board..

Loading & Running The Project on the Board

Make sure to open the example project workspace using the mentioned IDE(s) version or newer.

IAR Embedded Workbench™

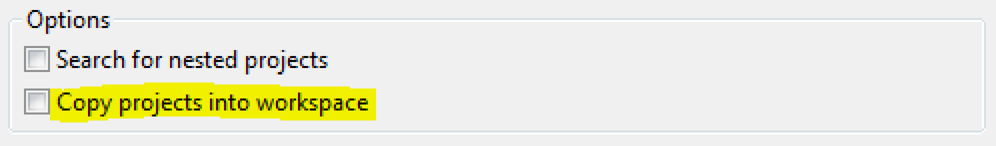
1. Click on *File–>Open–>Workspace...*
2. Navigate to the directory where the workspace is located: *$\Micrium\Examples\Freescale\STM32F429II-SK\OS3\IAR\OS3.eww*
3. Click *Open*.
4. For safety, clean the project by clicking on *Project–>Clean* (if available).
5. Compile the project by clicking on *Project–>Make*.
6. Make sure your hardware setup (as previously described) is correct.
7. Download the code to the board by clicking on *Project–>Download and Debug*.
8. Run the project by clicking on *Debug–>Go*. To stop the project from running, click on *Debug–>Stop Debugging*.

Keil uVision4™

1. Click on *Project–>Open Project...*
2. Navigate to the directory where the workspace is located: *$\Micrium\Examples\Freescale\STM32F429II-SK\OS3\KeilMDK\OS3.uvproj*
3. Click *Open*.
4. For safety, clean the project by clicking on *Project–>Clean Target* (if available).
5. Compile the project by clicking on *Project–>Build Target.*
6. *Make sure your hardware setup (as previously described) is correct*
7. Download the code to the board by clicking on *Debug–>Start/Stop Debug Session*.
8. Run the project by clicking on *Debug–>Run*. To stop the project from running, click on *Debug–>Start/Stop Debug Session* again.

Atollic TrueSTUDIO™

1. Click on *File–>Import...*
2. Select *Existing Projects into Workspace*.
3. Navigate to the directory where the workspace is located: *$\Micrium\Examples\Freescale\STM32F429II-SK\OS3\TrueSTUDIO*
4. Make sure the "*Copy projects into workspace*" check-box is unchecked.



Options

□ Search for nested projects

□ Copy projects into workspace

5. Click *OK*.
6. Make sure that the project has been selected under the Projects check-box.
7. Click *Finish*.
8. For safety, clean the project by clicking on *Project–>Clean* (if available).
9. Compile the project by clicking on *Project–>Build All*. The project should build successfully.
10. Make sure your hardware setup (as previously described) is correct.
11. Download the code to the board by right-clicking inside the project directory and selecting *Debug As–>Embedded C/C++ Application*.
    a. Select the appropriate interface inside the Debugger Tab (if needed).
12. Run the project by clicking on *Run-->Resume*. To stop the project from running click on *Run-->Terminate*.

µC/OS-III

```
void  main (void)
{
    ...
    OSInit(&os_err);                                   /* Initialize uC/OS-III
*/        (1)

 ...
    OSTaskCreate(&AppTaskStartTCB,                     /* Create the start task
*/        (2)
                 "App Task Start",
                  AppTaskStart,
                  0,
                  APP_CFG_TASK_START_PRIO,
                 &AppTaskStartStk[0],
                  APP_CFG_TASK_START_STK_SIZE / 10u,
                  APP_CFG_TASK_START_STK_SIZE,
                  0u,
                  0u,
                  0,
                 (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
                 &os_err);
    OSStart(&os_err);                                  /* Start multitasking
*/        (3)
}

static  void  AppTaskStart (void *p_arg)
(4)
{
    ....


    while (DEF_TRUE) {                                 /* Task body, always as an
infinite loop.      */        (5)
     ...
(6)

        OSTimeDlyHMSM( 0u, 0u, 0u, 500u,
(7)
                      OS_OPT_TIME_HMSM_STRICT,
                     &os_err);
    }
}
```

Listing - app.c

(1)
OSInit() initializes uC/OS-III and must be called prior to calling OSStart(), which actually starts multitasking.

(2)
OSTaskCreate() creates a task to be managed by uC/OS-III. Tasks can be created either prior to the start of multitasking or by a running task. In this case, the task "AppStartTask" gets created.

(3)
OSStart() starts multitasking under uC/OS-III. This function is typically called from the startup code but after calling OSInit().

(4)
AppTaskStart is the startup task created in
(2)
.

(5)
A task must be written as an infinite loop and must not return.

(6)
In most examples, there is hardware dependent code such as LED blink, etc.

(7)
OSTimeDlyHMSM() allows AppTaskStart to delay itself for a user-specified amount of time (500ms in this case). Rescheduling always occurs when at least one of the parameters is nonzero. Placing a break-point here can ensure that uC/OS-III is running, it should get hit periodically every 500 milliseconds.

For more information please refer to **uC/OS-III Users' Guide**.