TMDSDOCK28335_OS2

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.

TMDSDOCK28335 Example Project Read-Me

The provided example project for which this Read-Me was made utilizes the TI TMDSDOCK28335 (TMS320F28335) evaluation board from the TMS320x2833x Family. The MCU found on this development board conforms with the C28x architecture.

Project Download

Download Link	Micrium_TMDSDOCK28335_OS2.zip

Toolchain IDE Versions

IDE/Toolchain	Version
Code Composer Studio	6.1.0

Micriμm Product Versions

Product	Version
µC/CPU	1.30.02
µC/LIB	1.38.01
µC/OS-II	2.92.11

Hardware Setup

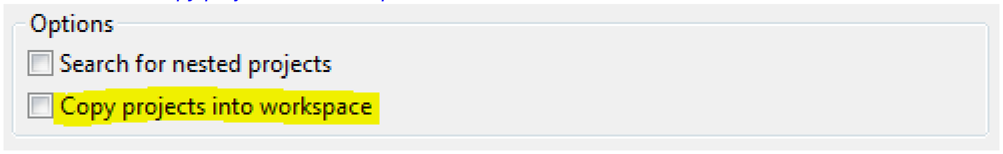
1. Have the board connected via the **TI XDS100v2** into the board debugging input (**JP2**).
2. Power will be provided by the USB port (JP2).

Loading & Running The Project on the Board

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

Code Composer Studio 6™

1. Click on *File–>Import...*
2. Select *CCS Projects*.
3. Navigate to the directory where the workspace is located: *$\Micrium\Examples\TI\TMDSDOCK28335\OS2\CCS*
4. Click *OK*.
5. Make sure the "*Copy projects into workspace*" check-box is **UNCHECKED**.



Options
☐ Search for nested projects
☐ Copy projects into workspace

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–>Code Composer Debug Session*.
    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-II

```
void  main (void)
{
    ...

    OSInit();                                           /* Initialize uC/OS-II
*/        (1)

    ...

    OSTaskCreateExt( AppTaskStart,                      /* Create the start task
*/        (2)
                     0u,
                    &AppTaskStartStk[APP_CFG_TASK_START_STK_SIZE - 1u],
                     APP_CFG_TASK_START_PRIO,
                     APP_CFG_TASK_START_PRIO,
                    &AppTaskStartStk[0u],
                     APP_CFG_TASK_START_STK_SIZE,
                     0u,
                    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
    OSStart();                                          /* 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)
    }
}
```

Listing - app.c

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

(2)
OSTaskCreateExt() creates a task to be managed by uC/OS-II. 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-II. 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-II is running, it should get hit periodically every 500 milliseconds.

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