

OS3



Licensing is required when using any Micrium 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.

ZC702 Example Project Read-Me

The provided example project for which this Read-Me was made utilizes the Xilinx ZC702 (XC7Z020 CLG484-1 AP SoC) evaluation board from the ZYNQ Family. The MCU found on this development board conforms with the ARM_Cortex_A9 architecture.

- [Project Download](#)
- [Toolchain IDE Versions](#)
- [Micrium Product Versions](#)
- [Hardware Setup](#)
- [Loading & Running The Project on the Board](#)
 - [ARM DS-5](#)
- [µC/OS-III](#)

Project Download

Download Link	Micrium_ZC702_OS3.zip
---------------	---------------------------------------

Toolchain IDE Versions

IDE/Toolchain	Version
ARM DS-5	5.20.0

Micrium Product Versions

Product	Version
µC/CPU	1.30.01.01
µC/LIB	1.38.01
µC/OS-III	3.04.04
µC/CAN	2.41.00

Hardware Setup

1. Have the board connected via the **D-Stream** into the board debugging input (**JTAG**).
2. Connect the Power Supply to the Power Connector (J60)..

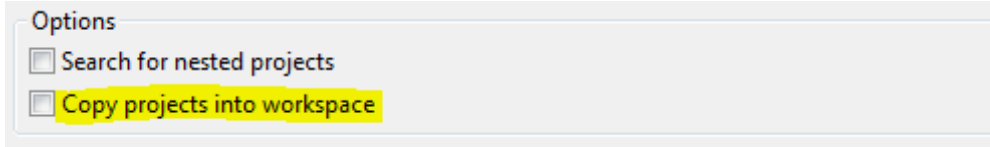
Loading & Running The Project on the Board



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

ARM DS-5

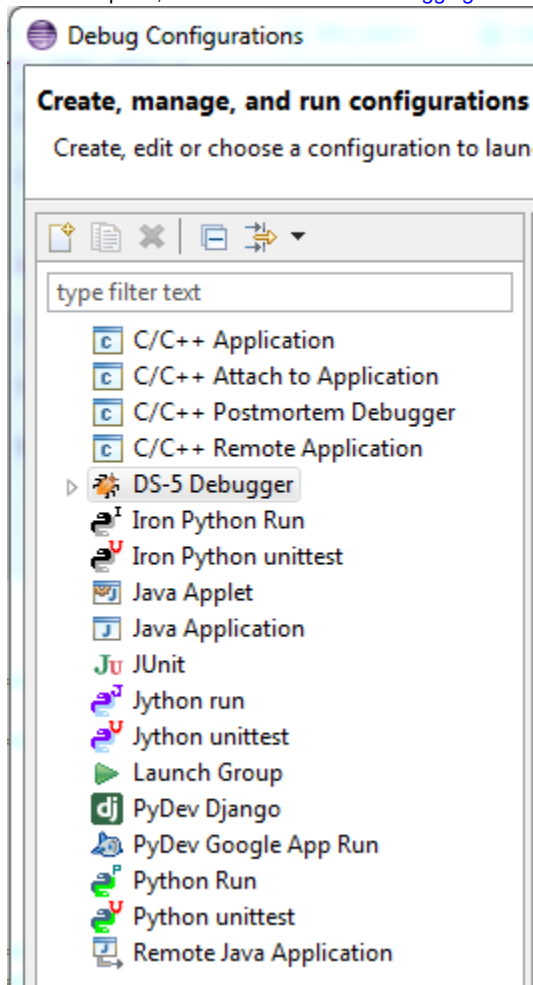
1. Click on [File->Import...](#)
2. Select [Existing Projects into Workspace](#).
3. Navigate to the directory where the workspace is located: `$Micrium\Examples\Xilinx\ZC702\OS3\ARM_DS5`
4. Click [OK](#).
5. Make sure the "Copy projects into workspace" check-box is unchecked.



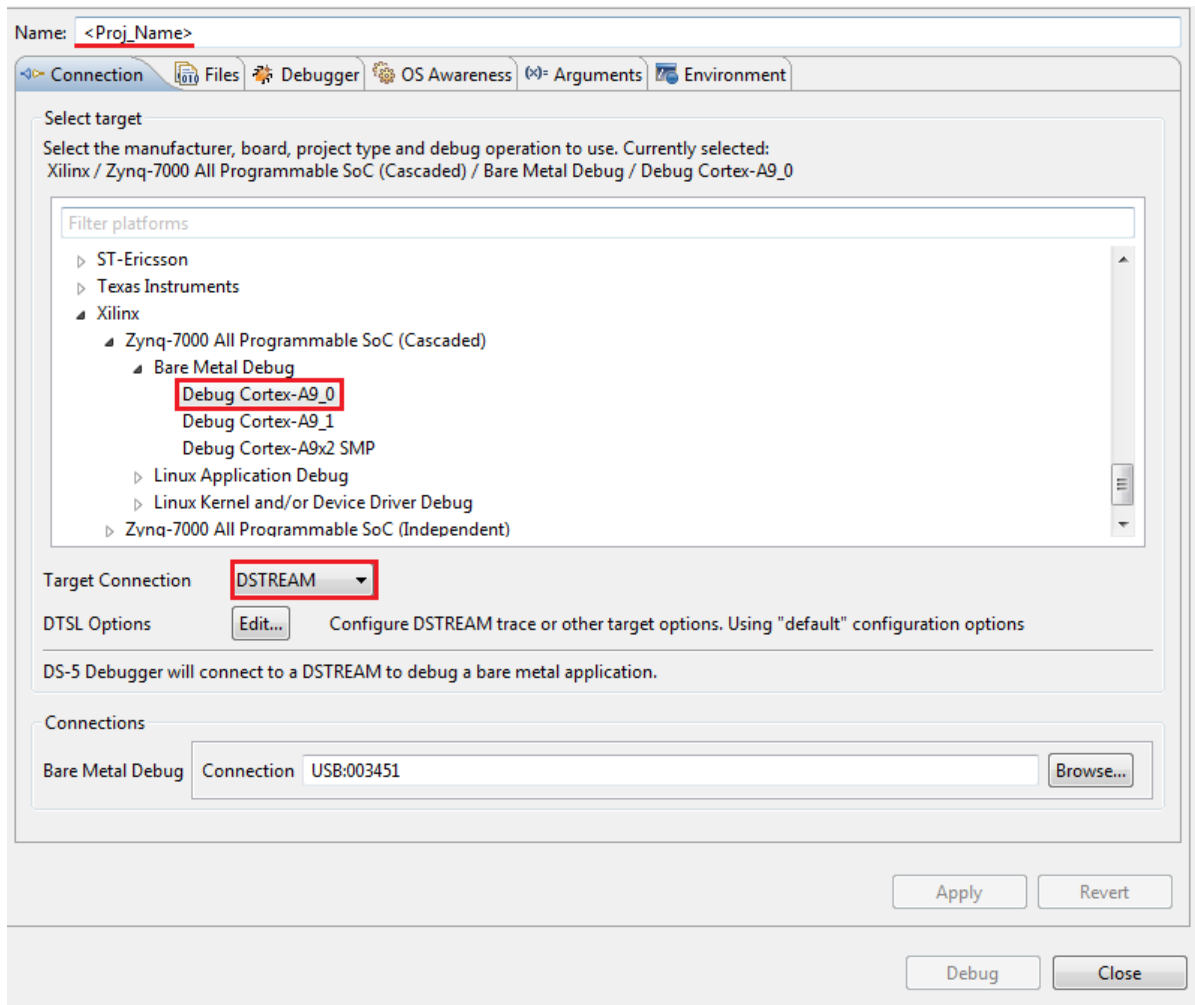
6. Make sure that the project has been selected under the [Projects](#) check-box.
7. Click [Finish](#).
8. For safety, clean the project by clicking on [Project->Clean](#) (if available).
9. Compile the project by clicking on [Project->Build All](#). The project should build successfully.
10. Make sure your hardware setup (as previously described) is correct.
11. A debug configuration is required to run the project, to create one, click on [Run->Debug Configurations...](#)

The following sequence of images show the steps required to create a debug configuration in order to run the project:

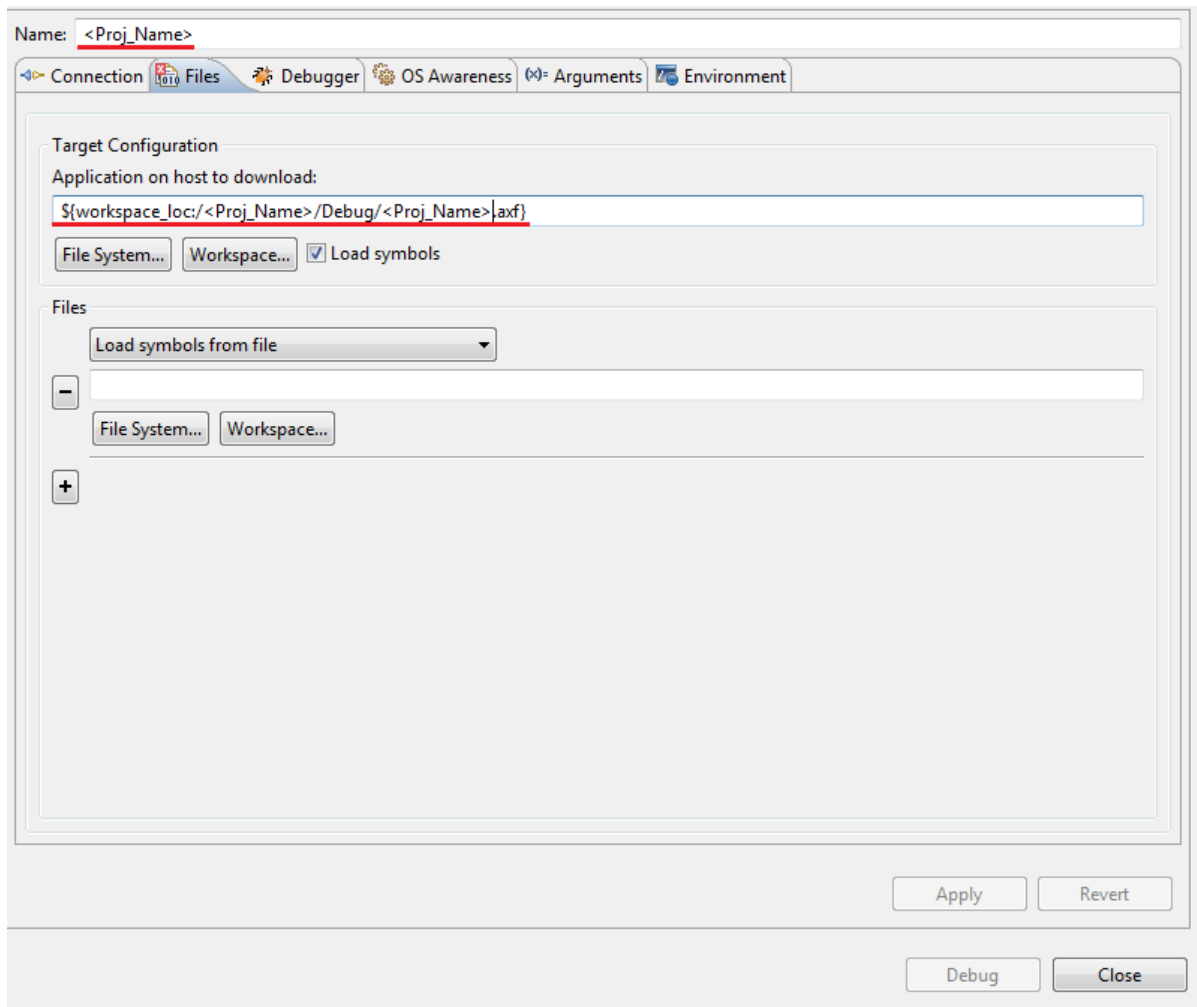
- a. On the left pane, double-click on [DS-5 Debugging](#).



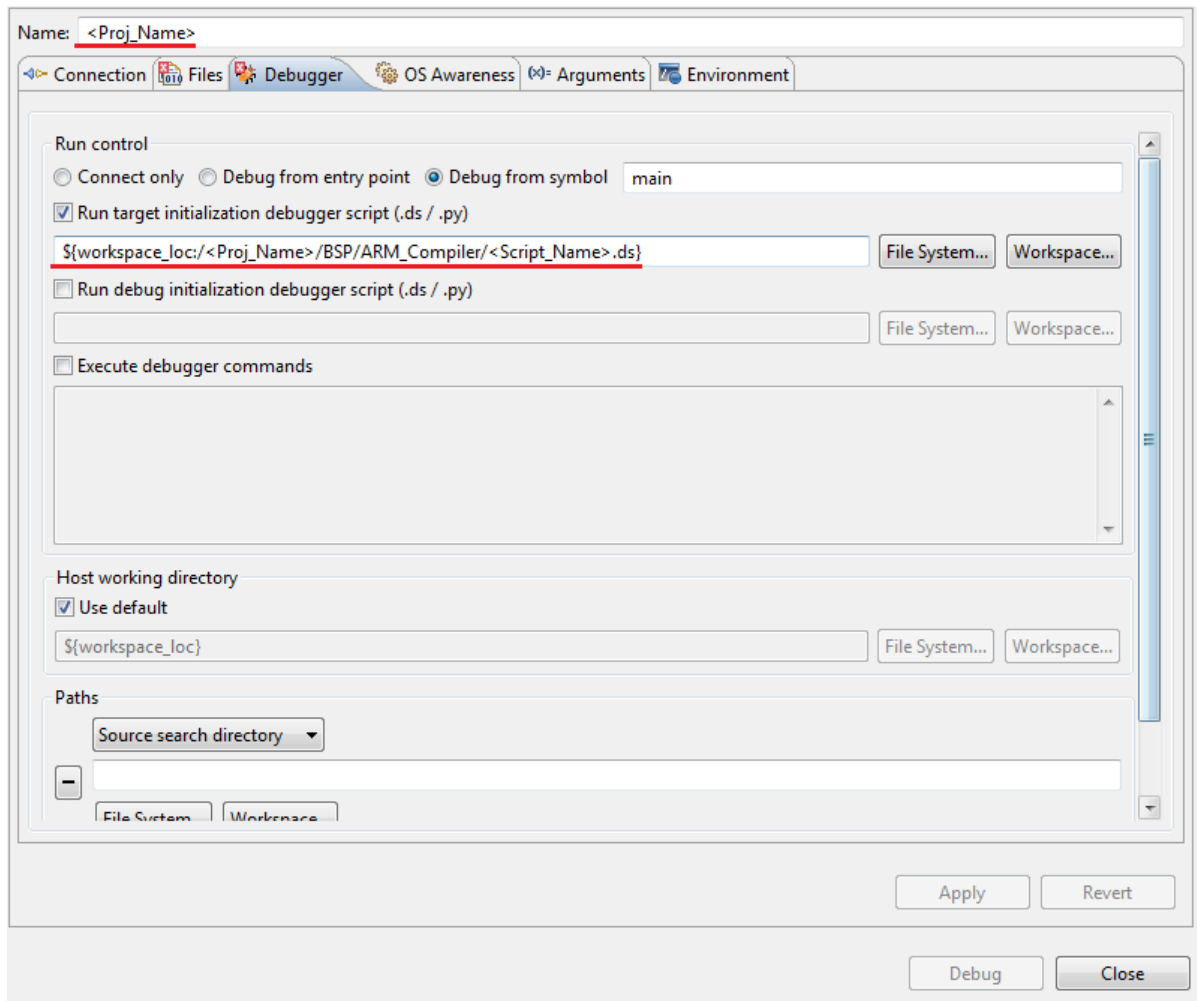
- b. The name of the project should automatically populate. The Target Debug and Target Connection (D-Stream) highlighted in **RED** must be configured to suit the project's requirements.






- c. Click on the [Files](#) tab, the information underlined in **RED** should automatically populate with the appropriate target configuration and project name. If not, for the target configuration, click on [Workspace...](#) and select the **.axf** file in the [Debug](#) Folder.

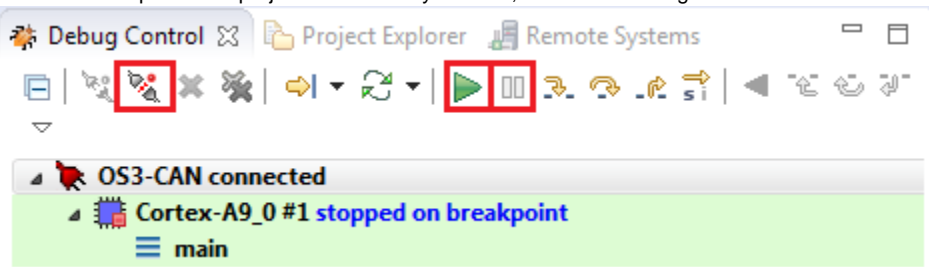


- d. Click on the *Debugger* tab, configure the 'Run Control' settings. Click on *Workspace...* and select the *.ds* file in the *BSP* Folder.



e. Click on [Apply](#) and then [Debug](#).

12. Run the project by clicking [Continue](#) button () under the [Debug Control](#) Tab. To pause the debugging session, click on the [Interrupt](#) button (). To terminate the debugging session, click on the [Disconnect from Target](#) button ().
 - a. ARM DS-5 will pause the project automatically in **main**, unless the setting has been disabled.



Please make sure the project is running on the proper **Build Configuration**. The project runs properly on the **Debug** Build Configuration. This can be found/changed in the following location: [Project->Build Configurations->SetActive->Debug](#).

```

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](#).