STM3240G-EVAL_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.

STM3240G-EVAL Example Project Read-Me

The provided example project for which this Read-Me was made utilizes the ST STM3240G-EVAL (STM32F407IG) evaluation board from the STM32F4x Family. The MCU found on this development board conforms with the ARM_Cortex_M4 architecture.

- Project Download
- Toolchain IDE Versions
- Micriμm Product Versions
- Hardware Setup
- Loading & Running The Project on the Board
  - IAR Embedded Workbench™
  - Keil uVision5™
  - Atollic TrueSTUDIO™
- µC/OS-II
- µC/Probe
  - Running with J-Link

Project Download

Download Link	Micrium_STM3240G-EVAL_OS2.zip

Toolchain IDE Versions

IDE/Toolchain	Version
IAR EW for ARM	7.40.2
Atollic TrueSTUDIO	5.3.0
Keil uVision	5.14
STM32CubeF4 Libraries	1.5.0

Micriμm Product Versions

Product	Version
µC/CPU	1.30.02
µC/LIB	1.38.01
µC/OS-II	2.92.11
µC/Serial	2.00.01
µC/Probe	3.5

Hardware Setup

1. Have the board connected via the **JTAG** into the board debugging input (**CN14**).
2. Power will be provided by an external power supply of 5V.
3. Make sure that JP18 is set to PSU.

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™

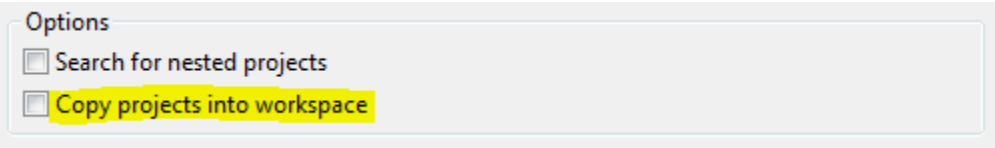
1. Click on *File–>Open–>Workspace...*
2. Navigate to the directory where the workspace is located: *$\Micrium\Examples\ST\STM3240G-EVAL\OS2\IAR\OS2.eww*
3. Click *Open*.
4. For safety, clean the project by clicking on *Project–>Clean* (if available).
5. Compile the project by clicking on *Project–>Make*.
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 uVision5™

1. Click on *Project–>Open Project...*
2. Navigate to the directory where the workspace is located: *$\Micrium\Examples\ST\STM3240G-EVAL\OS2\KeilMDK\OS2.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\ST\STM3240G-EVAL\OS2\TrueSTUDIO*
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. 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-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**.

µC/Probe

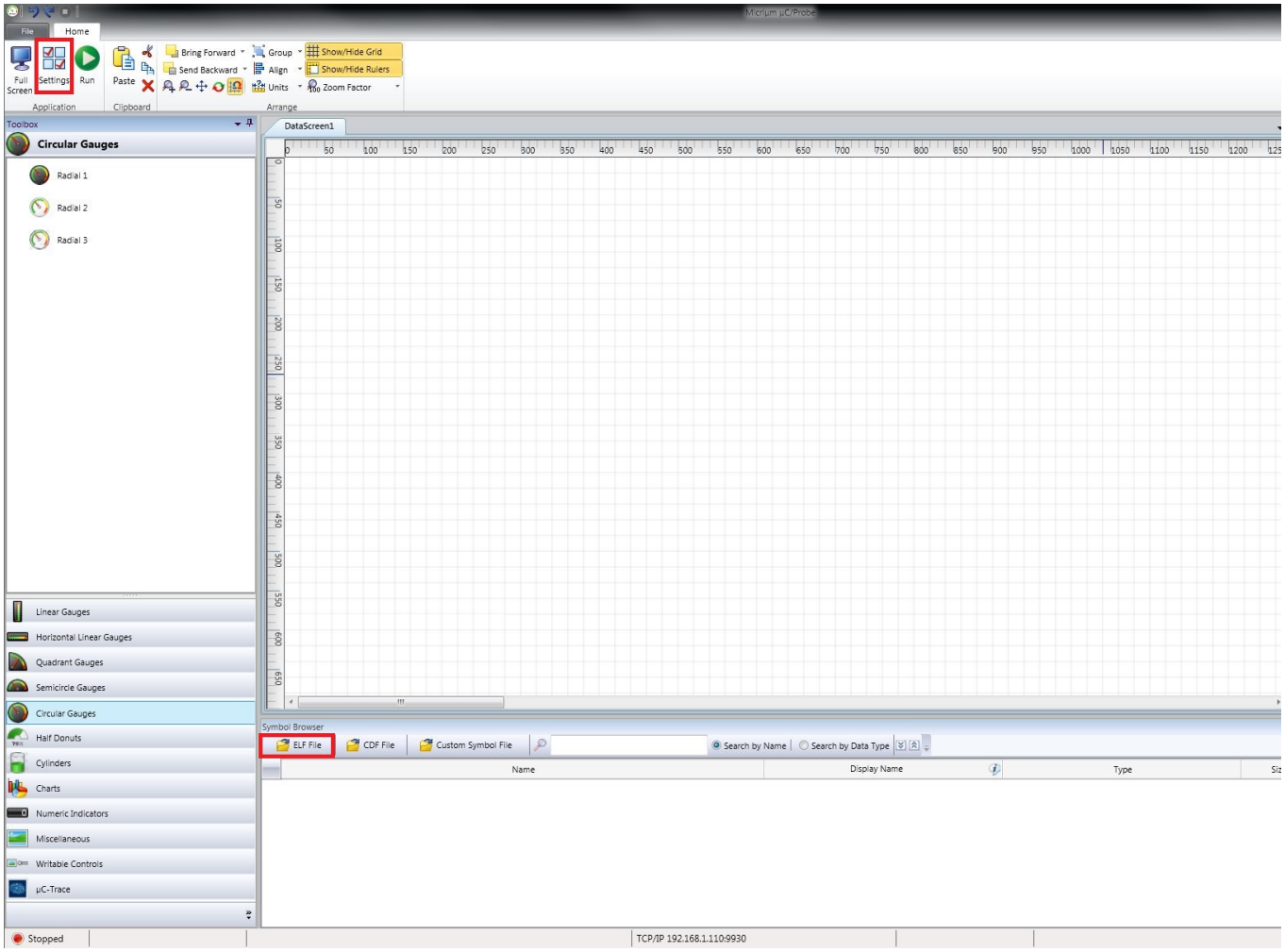
µC/Probe, is a Micriμm Windows™ application to graphically view the internals of any embedded system. This example project includes a pre-configured µC/Probe workspace that can be found at:

$\Micrium\Examples\ST\STM3240G-EVAL\OS2\<IDE>\OS2.wspx

> ⚠ Please compile the project (as described earlier in this document) prior to opening a pre-configured µC/Probe workspace.

In order for µC/Probe to display symbols, an **ELF file** that is generated by the compiler is required. After the example project has compiled, look for the ELF file that is usually found inside the compiler auto-generated binaries folder.

The following image shows where the **ELF file** (highlighted in **RED**) button is found to search for the project's ELF file.



If creating a new µC/Probe workspace, you must configure µC/Probe with the proper communication protocol used in your project. The following communication protocols are currently available for this example project:

Running with J-Link

When running a Micriμm example project that is using the J-Link debugger to interface with μC/Probe, there is no additional set-up necessary other than to configure μC/Probe's settings to "J-Link".

In μC/Probe's settings, under the *Communication* tab, select *J-Link* under the *Interfaces* section and configure the *Speed* and *Interface Mode* you desire that suits your project's needs. Along with the J-Link settings, μC/Probe also allows you to change the endianness of the device, how to receive statistics, and the rate at which the data collection is done.

The following image illustrates how the settings should look: