EOOS RT Automotive Demo Project for Versatile Application Baseboard

EOOS RT is an embedded object-oriented real-time operating system (RTOS) complied with MISRA C++ rules. It has been written in C++ language (ISO/IEC 14882:1998) and aimed to be used into microprocessor-based systems.

This project aims to cover microcontrollers which usually have limited hardware resources by AUTOSAR Adaptive Platform and to have possibility to execute AUTOSAR Adaptive Application on such kind of microcontrollers.

System requirements

The project is being developed using Ubuntu 18.04 and all stuff is tested on this version of operating system.

How to obtain environment

All necessary programs which are needed for building, demonstrating, developing and debugging can be obtained for the naked OS by executing ‘get-environment.sh’ script. If OS already has some programs installed on it, it is better to execute commands from the script manually in a console.

```
REPOSITORY/scripts$ source get-environment.sh
```

How to build the project

The project can be built by executing ‘build.sh’ script.

```
REPOSITORY/scripts$ source build.sh
```

The script has the next key parameters:

```
$1    --rebuild - rebuilds the project by removing the 'build' directory.
$2    --nomake  - prohibits to call 'make'.
$2/$3 --memdump - dumps information about memory sections.
```

Be informed that the parameters can be only put on the positions $P where P is a position.

How to run the project

Having built the project, it can be run on QEMU emulator by executing ‘run-qemu.sh’ script.

```
REPOSITORY/scripts$ source run-qemu.sh
```

Please note: The program does not terminate QEMU when it finishes. It simply stops on an idle instruction, or continues executing an infinite loop. Therefore, to terminate QEMU ARM emulator, first press Ctrl+A, then release the keys, and after press X.

How do debug the project

The built project program can be debugged in a variety of ways a developer knows. Here we will introduce only two ways how we do this. The first way is to use terminal GDB debugging, and the second one is to use Eclipse IDE, which is of course more convenient way.

For both of the ways, the next command has to be executed in a separated terminal.

```
REPOSITORY/scripts$ source run-qemu.sh --gdb
```

This command starts QUEM ARM emulator, loads the program, and freezes it for waiting a connection of GDB client.

Terminal GDB debugging

To connect to GDB server and the frozen program, the next command has to be executed in other terminal.

```
REPOSITORY/scripts$ source run-gdb.sh
```

This command starts GDB client and allows to execute any GDB commands in the terminal. For instance, the next command list demonstrates how to connect to the GDB server running on QUEMU and go through the program.

```
1. (gdb) target remote localhost:1234
2. (gdb) b main
3. (gdb) continue
4. (gdb) b _terminate
5. (gdb) continue
6. (gdb) quit
```

Executed the third line, GDB server will be forced to run the program to the main() function, and to display the operating system message:

```
[0.000000000] EOOS RT execution started
```

After the fifth line execution, GDB server will release the program until its termination, and the operating system message will be displayed on success when the termination is occurred:

```
[X.CCCCCCCCC] EOOS RT execution completed successfully
```

If the termination is occurred with an error, the error operating system message will be displayed:

```
[X.EEEEEEEEE] EOOS RT execution terminated with an error
```

The sixth line terminates GDB client. And to terminate the QUEM ARM emulator, you can press “CTRL+A X” keys.

Eclipse IDE debugging

Creating a project in Eclipse

For debugging the built project first of all an Eclipse project has to be created. The creation can be done by creating a Make project in Eclipse for existing CMake project.

1. Go to File > New > Makefile Project with Existing Code.

2. In the Project Name field enter a project name, for instance ‘eoos-exe-arm-arm926pxp-versatile’.

3. In the Existing Code Location field enter a path to the root directory of the Git repository, which is also a path to the CMake project.

4. Pressing the Finish button the Eclipse project will be created. The .cproject and .project files will appear in the root directory. All the root directory structure will be seen in C/C++ Projects dialog of Eclipse.

5. If the Cmake project has not built, build it following the “How to build the project” instruction. This will generate ‘build’ directory in the root directory and build a Make project.

Changing Make location in Eclipse

By default Eclipse looks for the Make project in the root directory. The next steps change this directory to the build directory.

1. Go to Project > Properties > C/C++ Build

2. In the Builder > Build command field, print ‘make install’. For this unset the Use default build command checkbox. This step will install an .elf file to the ‘REPOSITORY/build/CMakeInstallDir/bin/’ directory regarding the root directory of the repository.

3. In the Builder Settings > Build location > Build directory field, add the ‘build’ directory name to the existing path for having ‘${workspace_loc:/eoos-exe-arm-arm926pxp-versatile}/build’.

4. Press the Apply and Close buttons.

Configuring GDB debugger in Eclipse

1. Go to Run > Debug Configurations > GDB Hardware Debugging, and press the New Configuration button.

2. On Main tab for Project field browse ‘eoos-exe-arm-arm926pxp-versatile’ project.

3. For C/C++ Application browse ‘REPOSITORY/build/CMakeInstallDir/bin/eoos-app-arm926pxp-versatile.elf’ file.

4. Go to Debugger tab, and in GDB Setup > GDB Command field browse ‘/opt/gcc-linaro-7.3.1-2018.05-x86_64_arm-eabi/bin/arm-eabi-gdb’ which is path to GDB of the toolchain.

5. In the Remote Target > Port Number set 1234 the TCP port of the GDB server.

6. Go to Startup tab, and for Load Image and Symbols uncheck Load Image.

7. Press the Apply and Debug buttons.

Having done the steps above, Eclipse opens Debug window. After that you can open ‘system.Main.cpp’ file and set a Breakpoints to the main() function, and press Resume button (F8). You will see the operating system message in GDB server console.

```
[0.000000000] EOOS RT execution started
```

Program will stop on main() function so that everything can be debugged.