PTR - Super-Lab

Execution Framework (XF)

Introduction

In this laboratory, you will get the opportunity to implement your own E**x**ecution **F**ramework (XF). The goal is to implement the XF on two platforms (Qt and the PTR Embedded System).

The idea is to use Qt as a simulation environment so that a first version of the XF can be implemented quickly. Afterwards it will be adapted to the Embedded System.

A *Test-Bench* containing some tests is provided to you, with which you can check if your XF implementation functions correctly. Because the Test-Bench's code assumes a specific interface to the XF, the XF's header files with the class definitions will be provided to you as well.

Additional information can be found in the *XF* and *Test-Bench* documentation. It is in HTML format and was generated out of the code comments using the doxygen tool.

Figure 1: Another Kind of XF

All the information needed is at the following location:

https://cyberlearn.hes-so.ch/mod/folder/view.php?id=939611

- In the folder *Files/work* you will find a file named **work.zip**. It provides the following information:

Folder / File	Comment
test-bench	Folder with different tests
src/xf	Header and source files of the XF
src/mdw	Middleware package (with *trace* package)
src/platform	Platform specific code (Qt and Embedded System)
documentation.html	HTML documentation (Test-Bench, XF and trace)

- Also in the folder *Files/work* you will find a file named **trace-log.zip**. It contains a program named *TraceLog* to log and timestamp debug messages received via a serial communication channel. With this tool, you can check and prove that the timings are correct on the Embedded System.

Development Tools

Tool	Comment
Qt Creator	To develop the XF port for the Qt platform
System Workbench (SW4STM32)	To develop the XF port for the Embedded System
TraceLog	To log text output of the Embedded System

Additional Information

XF Components

The XF is made out of two components: The *XF core* and the *XF port*. The XF core component contains classes, which are the same for all platforms. The XF port contains classes, which must be adjusted to suit the needs for a specific platform. Every platform may have its specific port classes. Nevertheless, most of the time port classes do not need any adjustment. That is why a default implementation is present for every port class.

You are going to implement following ports:

Port name	Comment
idf-qt	Qt port with only one active dispatcher
idf-stmcube	Embedded system port based on STM32 ARM microcontrollers
default	Default port classes

The *idf-stmcube* port uses only classes from the default port. The *idf-qt* port implements some port classes and some are taken from the default port. The selection of the right classes for each XF port is already done for you using defines in the *xf-config.h* header file. Your job is to implement the code for the classes used.

Test-Bench

You are free to extend the interface of the XF – but you **must not change** the content of the test projects in the test bench. In case a test does not output the needed result it is not the fault of the test, it shows that your XF is not running as expected!

You are free to add additional tests to check the behaviour of the XF!

Project Import into Qt Creator

To implement the code for the XF it is best to take one of the tests in the *Test-Bench* and start coding there. To open a project in Qt Creator double-click the *.pro* file for the corresponding test.

Project Import into System Workbench

Import the test projects into System Workbench using the *Import...* functionality. Use "General/Existing Projects into Workspace" to import them. In the import projects dialog please verify that the option "Copy projects into workspace" is unchecked.

Embedded System

The display of the Embedded System is not used. The text output generated by the test classes is sent over the debug cable to the host PC. On the PC, you need to see if the virtual COM port is open to receive the text output from the Embedded System. You may use a serial terminal like *putty* to receive the output, but to get the timestamps added to the text received you must definitely use the *TraceLog* software.

In the XF code, you may also print text messages using the Trace class. Simply `#include "trace/trace.h"` and call `Trace::out("my message")`.

Goal of the Super-Lab

- Design an algorithm which lets the `XFTimeoutManagerDefault` behave in the same timely manner independent of whether it handles one or n timeouts
- Implement the core classes of the XF
- Implement the port classes for following ports: *default*, *idf-qt*, *idf-stm32cube*
- Test and comment the code of the XF
- The tests provided need to run using Qt and on the Embedded System platform
- Demonstrate the right functioning of the XF by comparing and documenting the output created by the tests running on the Embedded System

Super-Lab Delivery

Deposit on the moodle server a zip file containing the following information:

- The code of your XF
- A small report describing your XF (diagrams, key features, test results, status quo, possible improvements, etc.)

Link to XF Delivery page: https://cyberlearn.hes-so.ch/mod/assign/view.php?id=939612