Real Time Group

RT Embedded Linux Solutions

Final Project for SE Embedded Systems

Introduction

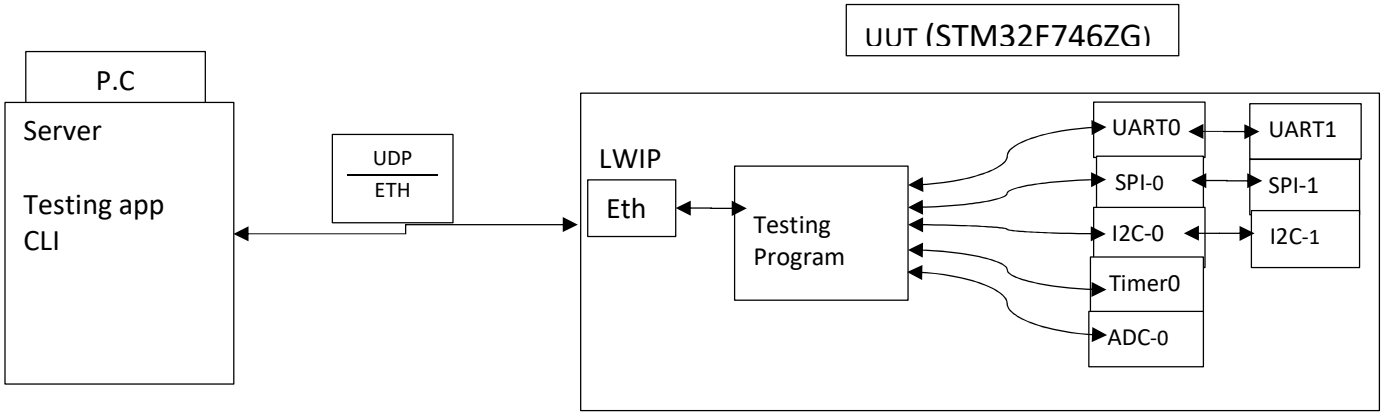
A manufacturing company is interested in testing the peripherals on the STM32F746ZG card in order to insure hardware correctness.

The company is asking you to take part in card's hardware verification for the following peripherals:

- UART
- I2C
- ADC

The verification resembles a client server program, the testing program (server) running on the P.C, sends commands using a propriety protocol over UDP (IP/Ethernet) to the UUT.

Once the UUT Testing program receives the proprietary protocol, finds out the peripheral to be tested and starts running the test.



Testing Block diagram

Real Time Group

RT Embedded Linux Solutions

Testing Procedure

1. The Testing program running on the P.C. (the Server) communicates with UUT (the client) through the UDP/IP communication protocol.
2. The server will send the commands (Perform the unit testing) on the UUT (the Client), and wait for the response from the UUT.
3. The UUT Ethernet device will receive the command and using the LWIP stack pass it to the "UUT Testing Program"
4. The test at the UUT is done by acquiring the needed parameters from the incoming command, such as:
    - Which peripheral is about to be tested,
    - what is the string of characters sent to the peripheral.
    - Test Iteration (number of times test has to be run at the UUT).
5. Once the UUT Testing has all the data needed for the test it will run the test for the number of defined iterations, each test will result in success or failure, each peripheral will have its own set of tests (elaborated later on).
6. The UUT test result per peripheral will be sent to the P.C. Testing program.

Proprietary Protocols

1. The command sent from the P.C. Testing Program to the UUT will contain the following:
    - **Test-ID** – 4 Bytes (a number given to the test so it will be easy to map it to the later on test result).
    - **Peripheral to be tested** – 1 Byte (a bitfield for the peripheral being tested: 1 –Timer, 2 – UART, I2C – 8, ADC – 16).
    - **Iterations** – 1 Byte (the number of iterations the test should run at the UUT (Unit Under Test)).
    - **Bit pattern length** – 1 Byte (the size of the Bit pattern string sent to UUT).
    - **Bit pattern** – Bit pattern length (the actual string of characters sent to the UUT).

2. The result protocol sent from the UUT back to the P.C. Testing Program will contain the following:

- **Test-ID** – 4 Bytes (a number given to the test so it'll be easy to map it to the later on test result).
- **Test Result** – 1 Byte (bitfield: 1 – test succeeded, 0xff –test failed).

P.C. Testing Program (Server Side)

1. Implementing the Testing Program using CLI (Command Line Interface) is good enough.
2. Implementation should be done using C \C++ on a Linux machine.
3. P.C. Testing Program should have persistent (saved on file system) testing records per Test-ID, and be ready to print on demand. including:
   - TEST-ID.
   - Date and time test has been sent to the UUT.
   - Test length measured in seconds (the amount of time it took to send the test command until receiving the result).
   - Result – Success or Failure.

UUT Testing Program (Client Side)

1. Once the UUT Testing Program receives the test command, it will acquire the needed parameters and initiate the test on the required peripheral.
2. Tests will vary between peripherals; the following includes Unit Testing for every Peripheral:
   - **UART, I2C**:
     - The below procedure is described for UART testing but it stands to I2C.
     - Peripheral testing is required to be done using DMA mode if possible.
     - Each Peripheral testing will require peripheral parameters, please choose (assume) parameters needed at your convenience (i.e. for UART you can assume BAUD Rate 115200, 8bit Data, 1 Stop Bit, No parity).
     - For the amount of needed iteration, the Testing Program will send the received Bit Pattern to the UART0, which in turn will pass the data to the UART1 port on the UUT.

- UART1 will send back the received string to UART0 (predefined program waiting for incoming data).
- For every iteration, the UUT Testing program will receive the incoming data from UART0 and compare it to sent data.
- It the testing has been successful for all iterations; a success result should be sent to the Testing P.C. Program.
- If at any time during the iterations a test has failed, testing should be stopped and a Failure result should be sent to the P.C. Testing Program.
- **(BONUS)**
  Calculate how long does it take to preform each I2C transmit and receive with 0.1 second and in 0.5 second resolution (separately).

- **ADC**:
  - Use ADC required parameters at your own convenience (i.e. 12 bit ADC).
  - Running the test beforehand, we should already have the bitstream for the analog to digital conversion at the current voltage.
  - For each iteration Run the conversion and compare the ADC result with already known result.
  - Send the P.C. Testing Program the final test result (after all the iterations).

Real Time Group

RT Embedded Linux Solutions

NOTE:
Please implement the project according to the following guidelines

The following requirements are mandatory:

1. **Functionality:**
   Code must be compliable, executable and achieves its goals.

2. **Deep understanding of the design:**
   The full understanding of the project requirements and the created solution. Explaining any technical decisions.

3. **Clean functions and classes**
   Code must be compact. Each element is achieved a specific purpose. Try to avoid many arguments, meaningless names and use of the flag arguments.

4. **Comments**
   Explain yourself in code, don't be redundant, and don't comment out code - just remove it.

5. **No "Code smell"**
   Avoid needless complexity or repetitions. The code must be simple and understandable.

6. **Tests**
   The tests that were done already (also manual ones are acceptable for this project) must be built as independent, fast, and repeatable units. Pay attention to the unexpected behavior and edge cases coverage.

7. **For more reading**
   a. https://missing.csail.mit.edu
   b. https://gist.github.com/wojteklu/73c6914cc446146b8b533c0988cf8d29

GOOD LUCK
(Although luck has noting to do with it, just hard work 😊)