

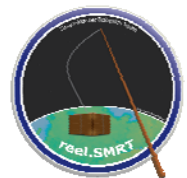
Appendix 4.

Software

4.1 Protocol Specification

4.2 FreeRTOS and microcontroller test report

reel.SMRT
Critical Design Review
24 May 2009



Appendix 4.1 Protocol specification v0.1

The reel.SMRT protocol is designed to maintain communication and data transmission of reel.SMRT payload, FISH and ground station in reel.SMRT project. The protocol consists of 2 links: ground station – reel.SMRT payload and reel.SMRT payload and FISH. Both of the links have to initiate the connection, detect the connection loss and recovery the connection in the case of connection loss. The data have to be transmitted from FISH to reel.SMRT payload then to ground station with lossless schema.

Abbreviations

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver/Transmitter

RF – Radio Module

Protocol Definition

The connection and data transmission are different depend on operation status which can be divided in 2 modes in reel.SMRT payload and FISH. The ground station has only one mode which is waiting for data or heart beat. The operations in each mode are described here.

Mode

Reel.SMRT payload and FISH can be operated in 2 modes:

- Normal mode
 - o Wait for command to change mode
 - o Heart beat packet has been sent to maintain the connection checking
- Data mode
 - o Data is sent and data packets are also checking the connection status

Note – Data mode is changed to normal mode automatically when there is no data packet left to send.

Link 1: Ground Station (GS) – (E-link) – reel.SMRT payload

- Using TCP internet protocol via E-link of the balloon in client-server architecture
- Reel.SMRT payload is client
- Ground station is server
- reel.SMRT payload tries to init the connection every 5 second if no connection is established
- When connection is established, reel.SMRT payload sent heart beat packet to ground station. Ground station do **not** sent acknowledgement to reel.SMRT payload because the TCP protocol has already handle the acknowledgement process
- Ground station can sent command to reel.SMRT payload, reel.SMRT payload has to reply back with the result status of the command

- When the ground station sends the command to start the experiment, reel.SMRT payload sends the command to the FISH and determines the status. Then reel.SMRT payload replies the result to ground station. If the start experiment is success, reel.SMRT payload changes its mode to data mode. The example can be seen in example section.
- reel.SMRT payload sends data packet that it got from FISH to ground station.

Connection loss detection

Ground station side – wait for certain time, if there is no packet from reel.SMRT payload, consider connection loss

reel.SMRT payload side – if there is no acknowledgement in TCP protocol process, consider connection loss

Link 2: reel.SMRT payload – (RF module) – FISH

- The RF module is connected to the UART of microcontroller, so no connection state is required. The data can be sent directly to the UART. The RF module has no packet control feature, so the protocol has to handle the data loss itself.
- FISH sends heart beat packet every 1 second
- reel.SMRT payload sends acknowledgement packet to the heart beat packet
- reel.SMRT payload can sent command to FISH, FISH has to reply back with the result status of the command
- When the reel.SMRT payload receives the command to start the experiment from ground station, reel.SMRT payload sends the command to the FISH to change it to data mode. FISH reply with the status, reel.SMRT payload sends result back to ground station. Then reel.SMRT payload starts the experiment. The example can be seen in example section.
- FISH sent data packet in data mode

Connection loss detection

reel.SMRT payload side – wait for certain time, if there is no packet from FISH, consider connection loss

FISH side – if there is no acknowledgement in from reel.SMRT payload for certain time, consider connection loss

Example operation

1. Drop command from ground station to reel.SMRT payload.
2. reel.SMRT payload checks that all factors for drop are ready(reel.SMRT payload gets heart beat from FISH). If fail , go to number 5.
3. reel.SMRT payload sends change to data mode command to FISH, if fail go to number 5.
4. FISH send status to reel.SMRT payload and FISH change to drop mode. if fail go to number 5
5. reel.SMRT payload replies to drop command with status to ground station

- If reel.SMRT payload cannot send the status packet to ground station, try to send again until successful.
 - If it cannot send for certain time, the connection loss is considered. Then reel.SMRT payload send command to fish to back to normal mode
6. If everything success, reel.SMRT payload changes to drop mode and starts drop

Action when link failure

Ground station

- Wait for client to connect again

reel.SMRT payload(Ground station side)

- First try reconnect the connection
- If the connection cannot be established, restart Ethernet module and reconnect again
- When connection establish, send packet that has not been seen again. If no packet left, start send heart beat

reel.SMRT payload (FISH side)

- Restart RF module and reconfigure it
- Then wait for package, try to go back to which mode depend on packet

FISH

- try to restart Zigbee and reconfigure it, send the packet that has not been sent

Packet structure

Ground station → reel.SMRT payload

Command

1	1 byte	1 byte	2 bytes	1 byte
C(67)	Packet Number	Command Number	Parameter	LF (0x10)

*no checksum because TCP already do it.

Packet Number (unsigned char) – running number of packet

Command Number (unsigned char) – number represent command

48 – (0 in ascii) Start slow reel ,

Parameter

- Time - How long to reel down and up e.g. 100 m

49 – (1 in ascii) Start drop ,

Parameter

- how much time to free fall e.g. 1000 ms

65 – 70 (A-F in ascii) Subcommand 1- 6,

- Command to move motor-one-by-one manually
 - To be defined later

97 – (Z in ascii) Check all device status
No parameter (just ignore the parameter field)

99 – (c small) Emergency Stop ,
No parameter (just ignore the parameter field)

Acknowledgement

No acknowledgement to Heart beat because TCP do it.

reel.SMRT payload → Ground station

Heart beat

1 byte

O(72)

If everything is ok (get connected to fish and all device is OK).

1 byte

H(72)

If something is wrong (Ground station will send check command to check status)

Reply to command

1 1 n (exact) 1 byte

C	Packet Number	result	LF (0x10)
---	---------------	--------	-----------

*no checksum because TCP already do it.

Packet Number (unsigned char) – packet number of command packet

Result (n-byte) – result status to command

- To be defined later

Data packet of reel.SMRT payload

1 1 n (exact) 1 byte

M(77)	Packet Number	Data	LF (0x10)
-------	---------------	------	-----------

Packet Number (unsigned char) – running number of packet

Data (n-byte) – data of the reel.SMRT payload e.g. accelerometer data, gyro data, motor temperature etc.

Data packet of FISH

1 1 n (exact) 1 byte

F(70)	Packet Number	Data	LF (0x10)
-------	---------------	------	-----------

* when get the packet from FISH, the reel.SMRT payload puts data into the memory for send to ground station and reply acknowledgement to FISH when the data is sent

Packet Number (unsigned char) – running number of packet

Data (n-byte) – data of the FISH e.g. accelerometer data, gyro data, pressure etc.

reel.SMRT payload → FISH

Acknowledgement to Heart beat

1 byte

A(65)

Acknowledgement to Data

1 1 byte

F(70)	Packet Number
-------	---------------

Packet Number (unsigned char) – running number of packet

Command

1 1 byte 1 byte 1 byte 1 byte

C(67)	Packet Number	Command Number	checksum	LF (0x10)
-------	---------------	----------------	----------	-----------

Packet Number (unsigned char) – running number of packet

Command Number (unsigned char) – number represent command

68 – (D in ascii) change to data mode

Checksum(1 byte) – checksum, calculated by all character before checksum XOR each byte

FISH → reel.SMRT payload

Heart beat

1 byte

H(72)

Data of FISH

1	1	n (exact)	1 byte	1 byte
F	Packet Number	Data	checksum	LF (0x10)

Packet Number (unsigned char) – running number of packet

Data (n-byte) – data of the FISH e.g. accelerometer data, gyro data, pressure etc.

Checksum(1 byte) – checksum, calculated by all character before checksum XOR each byte

Reply to command

1	1	1 byte	1 byte	1 byte
C	Packet Number	result	checksum	LF (0x10)

Packet Number (unsigned char) – packet number of command packet

Result (signed char) – result status to command

- 0, if command complete
- Other, to be defined later

Checksum(1 byte) – checksum, calculated by all character before checksum XOR each byte

Appendix 4.2 Microcontroller and FreeRTOS Test Report

Objective

1. Setup the development environment for microcontroller
2. Test basic functionality of microcontroller that available on evaluation board that related to project
3. Test functionality of FreeRTOS
4. Test functionality of FreeRTOS simulator

Tool

- Eclipse C++
- Yagarto arm compiler
- Flash Magic
- Visual studio
- NXP LPC2368 evaluation board kit
- USB-to-serial cable
- FreeRTOS source packet

FreeRTOS and Microcontroller Procedure

1. Download FreeRTOS source packet into your computer from <http://www.freertos.org>.
2. Follow the instruction at <http://www.freertos.org/Eclipse.html> and <http://www.yagarto.de/howto/yagarto2/index.html#download> to install development tool need to compile the FreeRTOS source code with Eclipse.
 - o Now we have Eclipse C++ with Yagarto
3. Open the demo project for LCP2368 for Eclipse in the demo directory in source packet and follow <http://www.freertos.org/Eclipse.html> in section"Opening and using a FreeRTOS.org demo project".
 - o Now we have the demo project in Eclipse with correct reference directory
4. Try to compile the project and we got the problem of compilation error.
 - 4.1 Standard IO problem.

Because of new version of Yagarto was built to support newlib stub. We have to add the stub by adding "syscalls.c" from <http://www.yagarto.de/download/yagarto/syscalls.c> to the project.
Then include syscalls.c to the makefile.
 - 4.2 Uncomment this line in FreeRTOSConfig.h

```
/* Value to use on rev 'A' and newer devices. */  
#define configPINSEL2_VALUE 0x50150105
```
5. Finish compilation and get FreeRTOSDemo.hex
6. Download FlashMagic from www.flashmagictool.com
7. Connect USB-to-serial cable to the computer and install the driver.
8. Configure jumper according to this instruction http://www.keil.com/support/man/docs/mcb2300/mcb2300_fp_flash_alone.htm
9. Burn the "FreeRTOSDemo.hex" into evaluation board using FlashMagic.

FreeRTOS Simulation Procedure

1. Open “Unsupported demo” in the demo folder of FreeRTOS and unzip “x86_VisualStudio8_DJ.zip”.
2. Open the project file in the demo/WIN32 folder.
3. Try to compile and run.

Result

The development environment has been setup in Eclipse as seen in Figure 1.

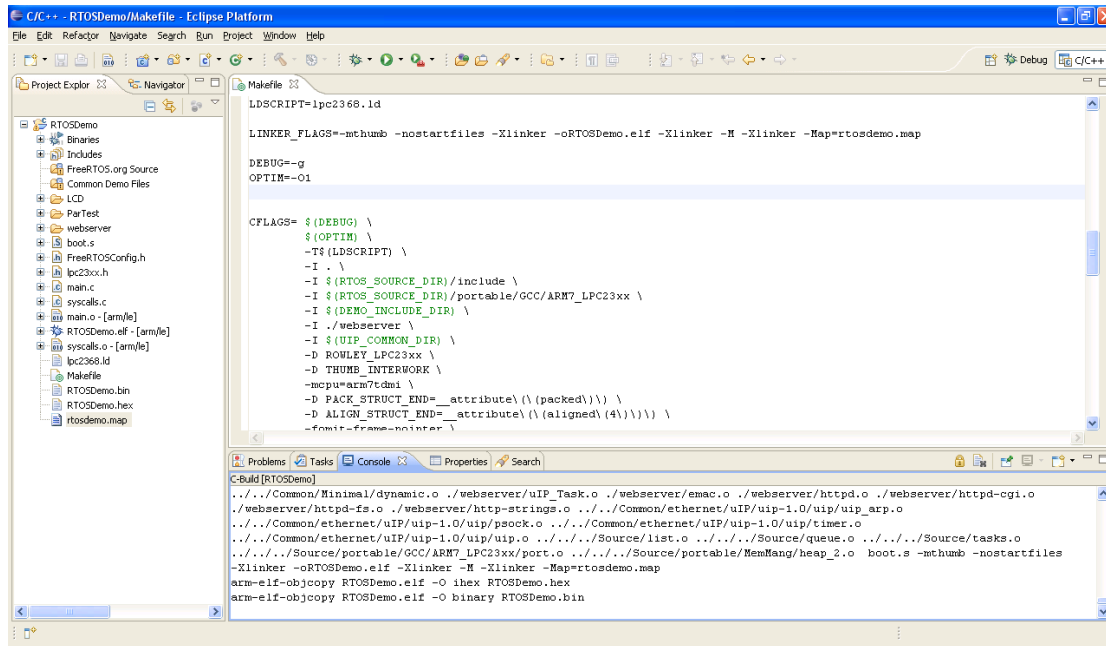


Figure 1 Eclipse development environment with demo project.

The evaluation board is running with FreeRTOS and successfully perform the LCD, LED and web server. The mutual excursion and real-time response also has been test inside the demo. The hardware connection with computer can be seen in Figure 2.

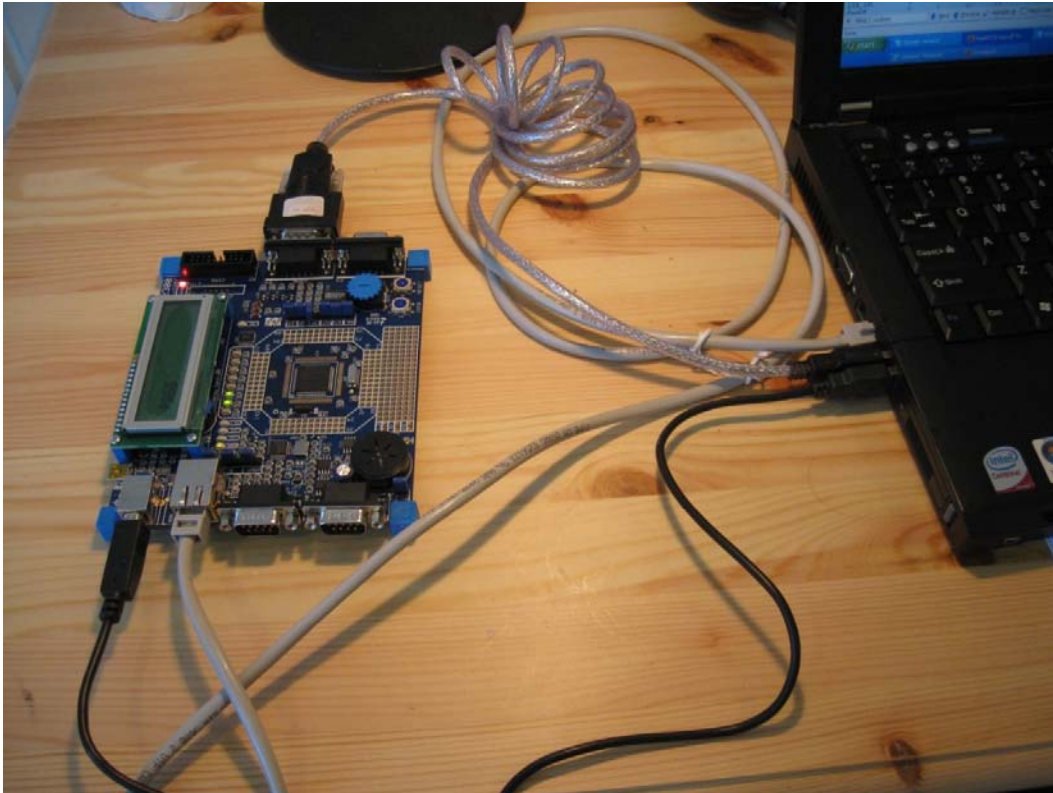


Figure 2 Evaluation board with FreeRTOS demo.

The web server has been test in computer. The response is quick and correct. The result page can be seen in Figure 3.

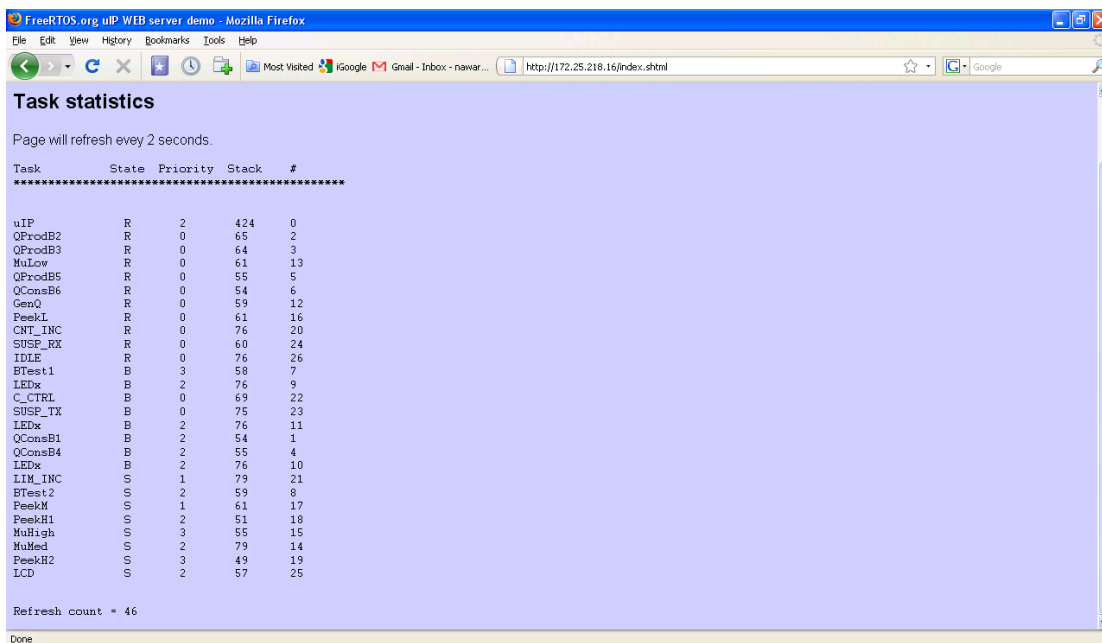
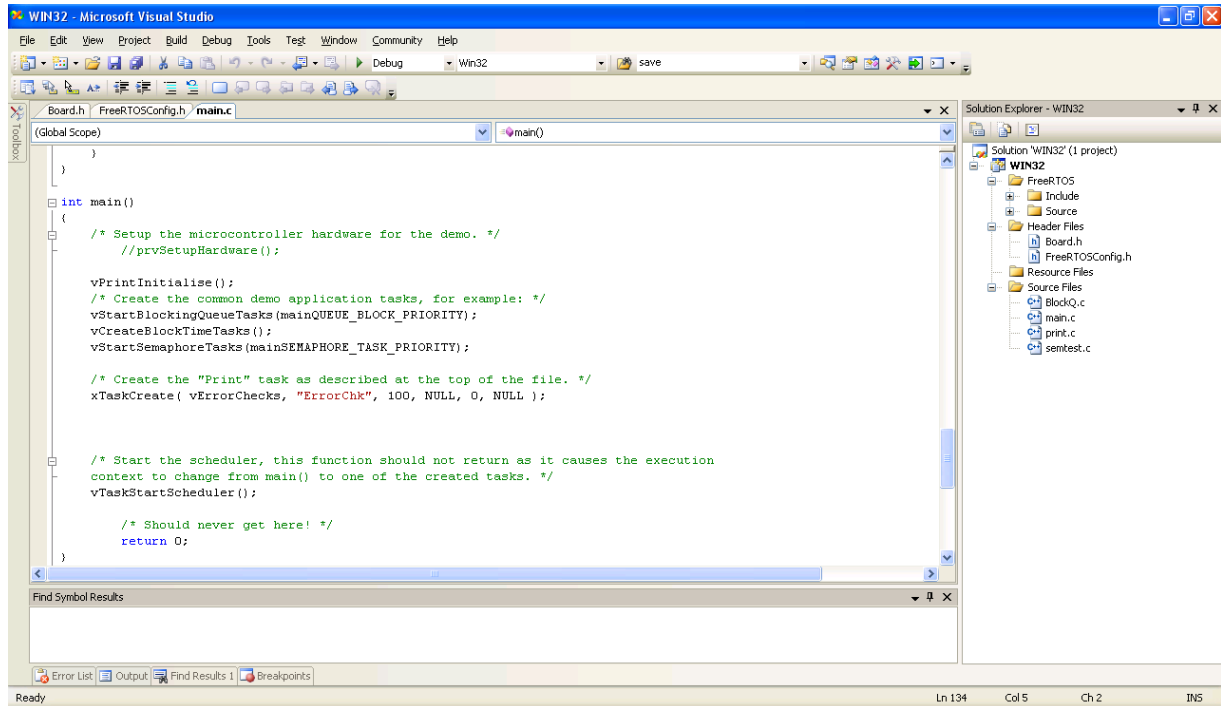


Figure 3 Web server demo.

Simulation environment is working fine, the task and FreeRTOS code is working in the simulation. The code in the Visual Studio 2005 development platform can be seen in Figure 4.



```
WIN32 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Community Help
Debug Win32 save
Board.h FreeRTOSConfig.h main.c
(Global Scope) main()
}
}
int main()
{
    /* Setup the microcontroller hardware for the demo. */
    //prvSetupHardware();

    vPrintInitialize();
    /* Create the common demo application tasks, for example: */
    vStartBlockingQueueTasks(mainQUEUE_BLOCK_PRIORITY);
    vCreateBlockTimeTasks();
    vStartSemaphoreTasks(mainSEMAPHORE_TASK_PRIORITY);

    /* Create the "Print" task as described at the top of the file. */
    xTaskCreate( vErrorChecks, "ErrorChk", 100, NULL, 0, NULL );

    /* Start the scheduler, this function should not return as it causes the execution
    context to change from main() to one of the created tasks. */
    vTaskStartScheduler();

    /* Should never get here! */
    return 0;
}
Find Symbol Results
Error List Output Find Results 1 Breakpoints
Ready Ln 134 Col 5 Ch 2 INS
```

Figure 4 FreeRTOS simulation code.

Conclusion

The development platform for FreeRTOS and simulation has been setup and tested. FreeRTOS has been tested which can provide good real-time deadline task. The Ethernet functionality has been tested.

To conclude, FreeRTOS and NXP LCP2368 have been proved in concept that can handle and control the experiment. The development phase can start.