1 Day 1

1.1 Warming up

Setup the debug environment and use magic config to setup the board to RAM debugging.
Browse the software/ directory and get familiar with the setup. Build some examples (software/examples/atmosfire) and try to run them.

1.2 Get UART up and running

Use the BSP UART driver to make UART port 1 echo text. Look at the example `atmosfire/uart/polling` for inspiration.

1.3 Use Graphics Library to create full screen effects and scrolling text

First play around the gfxlib API with the Lua interactive prompt in mtshell.

Look at the example `gfxlib_effect` for inspiration on how to use the API from C.

1.4 Use BSP and Graphics Library to set pixels/use effects according to UART data

Combine 1.1 and 1.2.

1.5 Compile and try out one ore more USB examples

Since these examples are interrupt driven, it's probably a good idea to remap internal flash to address 0x0 and run from there.

1.6 Setup a UART with 38.4kbps and echo characters

Do not use the BSP driver, but you can use it for inspiration! ☺ The BSP function is called bsp/atmosfire/src/uart.c

1.7 Add a timer interrupt and broadcast hello message on UART each sec

The example sets up the vector controller with jumps to ISRs in C. These ISR are located in the 71x_it.c file. So if you set up the timer correctly you should automagically end up in the corresponding ISR in that file.

1.8 Use an FreeRTOS to make X bouncing 4x4 balls

See example in `rtos/FreeRTOS/Demo/ARM7_STR71X_GCC`.

2 Day 2

The team with the most thrilling Lua application will win a fabulous price at the end of the day!

2.1 Write a Lua program that bounces X balls on the display

There are some Lua tutorials on the CD, read them! You should also look at the Lua bitfire API in atmosfire software developer's guide. Finally there are some Lua examples under software/examples/lua

2.2 Launch a luac compiled program on the target

This includes building lua with correct flags on the host. Cygwin is installed with the 'gcc' package. This gives you a native gcc compiler. Go to the CD's source directory and unpack the file called **lua-5.0.2.tar.gz**. This is the same version as is running in the mthsell example. Modify the **config** file and uncomment the line beginning with "NUMBER = -DLUA_USER_H…". Set –DUSE_INT here (the target uses INTs as Lua numbers). No just do make to build host versions of lua. Luac.exe will end up in bin/ directory.

2.3 Compile mtshell eCos application with some modifications

Try first to load it directly into external ram and run it from there. You can also put in redboots flash, but if you run it from ram you can launch it from eclipse and debug the application.

2.4 Create your own eCos application

Look at the mtshell makefile for inspiration. Make a threaded application that draws something on the display.

2.5 Recreate the atmosfire eCos libraries with some additions

Follow the instruction in the atmosfire software developer's guide (not for the faint of heart). Add some additional packages and write an app that uses them. In configtool, check out Build->Packages

2.6 Use an FreeRTOS to make X bouncing 4x4 balls that collides with each other (one task = one ball, use messages between tasks)

Build on the lab from day 1. Introduce messages between tasks.