COMPILING YOUR FIRST LKM AND LOADING ON THE EVB

In order to compile Embedded Linux Applications \ Modules and download them to the Evaluation board, you first need to complete the " Installing Course SW" guide, please make sure you have done so.

**** You need to run as root just enter:**
Red-Hat \ Centos: *su*
Debian \ Ubunto: *sudo su*

By now you should have:

1) Installed the **arm-linux-gcc Cross-compiler.**
   a. in order to make the cross-compiler run from everywhere use the following command line in /root/.bashrc
      PATH=$PATH:/usr/local/arm/4.3.2/bin
      ** don't forget to open a new terminal so the script will be run
   b. To verify it works type from the command line:
      arm-linux-gcc –v
   c. In Case you are working on a 64 bit Linux Machine you might need to install support for 32bit libraries:
      sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0

2) Installed the **mini2440-kernel source-code**,
   if not, please do the following:
   a. Download the mini2440-kernel source code (link) (downloaded file is called **linux-2.6.32.2-mini2440.zip**)
   b. Create a new directory called kernels under /usr/src (so you"ll get **/usr/src/kernels**) , with the following command:
      mkdir /usr/src/kernels

c. Move **linux-2.6.32.2-mini2440.zip** to /usr/src/kernels with the command:
   mv linux-2.6.32.2-mini2440.zip /usr/src/kernels
d. Unzip the **linux-2.6.32.2-mini2440.zip** file with the command:
   unzip linux-2.6.32.2-mini2440.zip
e. Change the mini2440-kernel source-code directory name to "linux-2.6.32.2" to match the lessons Makefiles:
   KERNELDIR=/usr/src/kernels/linux-2.6.32.2
   use the following cmd:
   mv linux-2.6.32.2-mini2440   linux-2.6.32.2
   your mini2440-kernel source-code path should be:
   /usr/src/kernels/linux-2.6.32.2

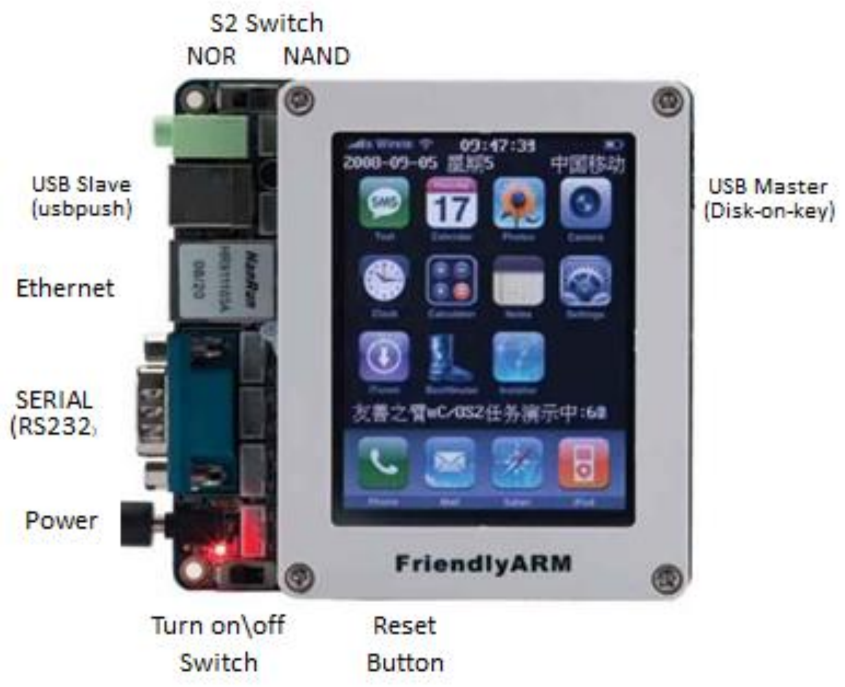
3) Installed the **Serial RS232 manager – minicom**
   Installing minicom:
   Centos: yum install minicom
   Ubunto: apt-get install minicom
   ** for minicom configuration please read **Installing minicom** section of the " Installing Course SW" guide

Now that you have installed all the needed Software on your Linux Machine (or Virtual Machine), we can go ahead and start.

In order to compile and load your first Linux Kernel Module, we need to get the system ready, so some additional preparations are in order (in this precise order please):

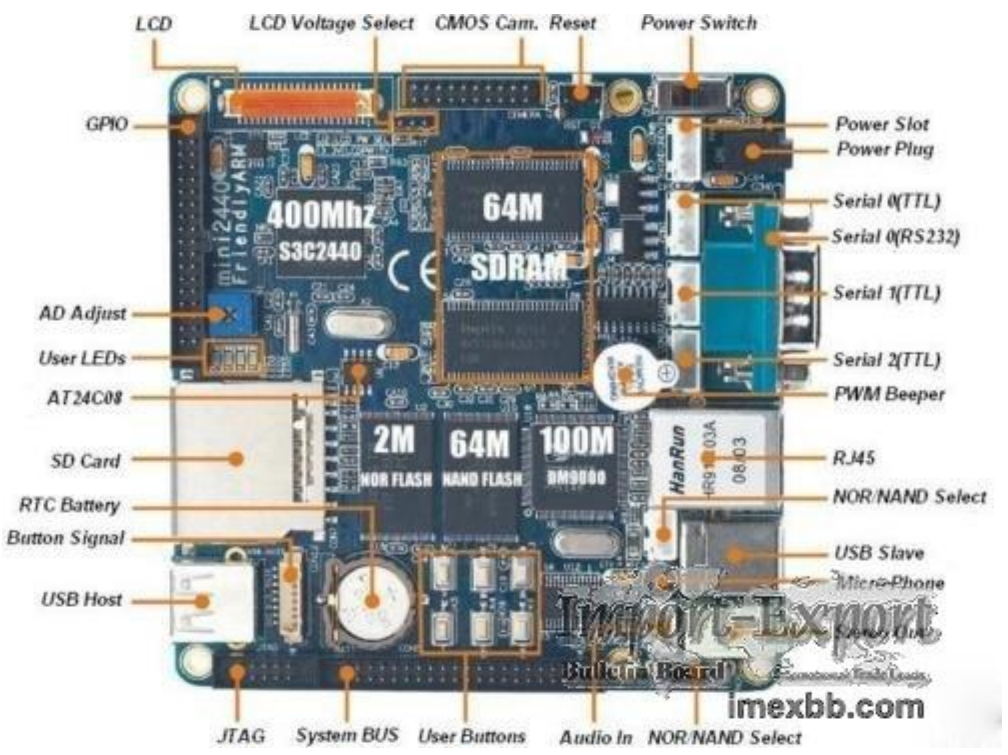
1) Get to know the mini2440 EVB
2) Connect the EVB to the Laptop \ P.C.
3) Turn on the EVB and make sure Linux is loading.
4) Compile the "hello-word" module on the Linux Machine
5) Copy the hello.ko (kernel-image-module) to the EVB file-system
6) Load the hello.ko module into the EVB kernel

1) Get to know the mini2440 EVB



Real Time Group

Hardware + Software = Embedded Solutions



Real-Time Group

rt-ed.co.il rt-dev.com rt-hr.co.il

Professor Shore Street, , Holon 972-77-7067057

2) Connecting the EVB to the Laptop \ P.C.

The "USB to Serial" cable will be used **:**

a. **The USB end should be connected to the P.C.**

   1) before plugging in the USB,
   clean the kernel-print-buffer using: *dmesg –c*

   2) Once you have plugged in the USB plug into the Laptop\P.C.
   the kernel should detect it.
   Using the kernel *dmesg* command you can see the special-device-file that the "USB-to-Serial" device is mapped to, it should be of the following format: **ttyUSBx** [x= 0,1,2…]

   3) Make sure you use this Device file in your minicom configuration as well as 115200 8n1 (*minicom –s*)
   ** for Virtual-Machine users make sure you have enabled the USB device for the virtual machine

b. **The Serial end should be connected to the EVB**

3) Turning on the EVB and making sure Linux is loading.

a. You need to connect the EVB to the Power cord (make sure it's not the green sound interface!).
b. Before turning on the EVB, switch the S2 switch towards the Nand Flash side, The system will boot from Nand Flash (where the Embedded Linux Kernel is installed)
c. Turn on the EVB, using the minicom utility you should see the Embedded Linux kernel loading on the EVB.
   wait until the loading is finished, hit the keyboards enter-key to make sure its responsive.

4) Compiling the "hello-word" module on the Linux Machine

a. Download the lesson from the site: Lesson-4: Linux Kernel Modules
b. Unzip the lesson, we'll be focusing on lesson-4.1,
   go to that directory
c. Open the Makefile with an editor (gedit for example)
   Make sure that the following are correct:

i. The " build=INTEL" is remarked so the kernel will use the else <section> and build the module for ARM architecture.
ii. The Cross-Compiler: *arm-linux-gcc* is working properly and is in configured within the PATH environment variable.
iii. The KERNELDIR variable is mapped to the correct mini2440 kernel source code location.
KERNELDIR=/usr/src/kernels/linux-2.6.32.2
iv. In case you are using 64bit Linux Machine, you have installed the support for 32bit libraries:
sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0

d. Trying to compile your module now will result in error with the following
message:

```
ERROR: Kernel configuration is invalid…..
Run make oldconfig….
```

The LKM building process is integrated into the kernel and is based on the linux-kernel-source code.
In order to compile LKMs you need to:

i. Create a build configuration file - .config
(basically its made by one of the following utilities: make oldconfig \ make menuconfig \ make xconfig …
in order to make things easier will be using an already manufacturer config file- config_mini2440_n35 as follows:

```
cp   config_mini2440_n35 .config
```

ii. Compile the mini2440 kernel source code: run make at the root of the kernel tree

```
cd  /usr/src/kernels/linux-2.6.32.2
make
```

e. Go ahead and compile the lesson-4.1 (hello world) kernel module using the make command:
   > *make*
f. If there were no errors, you should get a ***hello.ko*** file
g.

5) Copying the hello.ko (kernel-image-module) to the EVB file-system can be done in two ways:
   a. using a disk-on-key
   b. using ftp client \ server

a. **using a disk-on-key to copy hello.ko image to EVB**
   when plugging in the disk-on-key into the Linux Machine, the kernel notices it and opens a directory mapped to that disk-on-key.
   copy the hello.ko file into that directory
b. be sure to umount (unmounts) \ eject thedisk-on-key before detaching it.
c. Inert the disk-on-key into the EVB's USB master (see figure-2 above).
d. In case the EVB doesn't recognize the disk-on-key, you need to mount it yourself:
   i. Use the dmesg command to know which device-file is used for the disk-on-key by the kernel, should be ***sda, sdb, sdc …***
   ii. Create a mounting directory on the EVB's file-system
       mkdir /mnt/usb
   iii. Using the device file found, mount the disk-on-key:
       mount *\/dev/sda* /mnt/usb
   iv. access the disk-on-key through the created directory
       cd *\/mnt/usb*

6) **Loading the hello.ko module into the EVB kernel**

   a. Before loading the hello.ko module' clean the kernel message buffer:
      dmesg -c
   b. load the hello.ko module:
      insmod hello.ko
   c. make dure the module has loaded:
      lsmod (look for the module name)
   d. have a look at the modules messages:
      dmesg -c

7) **Unloading the hello.ko module from the EVB's kernel**

   a. Use the following command:
      rmmod hello.ko
   b. have a look at the modules messages:
      dmesg -c