EPOSMote III Programming

Table of contents

- EPOSMote III Programming
- Creating a .hex image
- 1. Using JTag
    - 1.1. Load an image directly
    - 1.2. Load EPOS' USB bootloader
    - 1.3. Load EPOS' Network bootloader
    - 1.4. Enable EPOSMote III's ROM Serial bootloader
- 2. Using EPOS' USB bootloader
- 3. Using EPOS' Network bootloader
- 4. Using the ROM Serial bootloader
- 5. Help!

Creating a .hex image

There are currently four ways you can load an image to a local EPOSMote III device. For all of them, you need a .hex file, which you can create with EPOS' default makefile using the "flash" command, like so:

```
$ make APPLICATION=<your_application> flash
```

The resulting image shall be img/.hex

Keep in mind that for simple tests, you can use our remotely-accessible EPOSMote III boards.

1. Using JTag

You only need this method if you have a device that has never been programmed before. In case your device already has a bootloader loaded, you can skip to the bootloader's respective section.You need JTag to accomplish one of the following:

- Load an image directly
- Load EPOS' USB bootloader
- Load EPOS' Network bootloader
- Enable EPOSMote III's ROM Serial bootloader

You are going to need JLinkExe, which you can download from SEGGER's website. It is convenient to follow the steps in the .tgz's README to enable user-level access to the JTag device on Linux.

1.1. Load an image directly

To load an image directly with JTag, you can either use JLinkExe manually (1), or use EPOS' default makefile (2). If you wish to compile an application to load with JTag and run without EPOS' bootloader, you need to make the following adjustments to the memory map in include/machine/cortex_m/emote3_traits.h (remember to run make veryclean after):

```
MEM_BASE  = 0x20000000;

APP_LOW   = 0x20000000;
APP_CODE  = 0x00200000;
APP_DATA  = 0x20000000;
```

```
PHY_MEM   = 0x20000000;

SYS       = 0x00200000;
SYS_CODE  = 0x00200000;
SYS_DATA  = 0x20000000;
```

1. With JLinkExe (replace with the name of the image you want to load):
   ```
   $ JLinkExe
   J-Link>device = cc2538sf53
   J-Link>h
   J-Link>erase
   J-Link>loadbin <your_application>.hex,0
   J-Link>exit
   ```

   It is normal to see messages such as "Failed to identify target. Trying again with slow (4 kHz) speed." If everything was successful, you should see a message like this after the loadbin command:
   ```
   Downloading file [<your_application>.hex]...Info: J-Link: Flash download: Flash
   programming performed for 2 ranges (14336 bytes)
   Info: J-Link: Flash download: Total time needed: 0.461s (Prepare: 0.091s, Compare:
   0.007s, Erase: 0.153s, Program: 0.194s, Verify: 0.003s, Restore: 0.010s)
   O.K.
   ```

2. If you are compiling a new EPOS image, you can instead upload it to the device directly with EPOS' default makefile:
   ```
   $ make APPLICATION=<your_application> deploy
   J-Link>exit
   ```

1.2. Load EPOS' USB bootloader

1. Get the latest version of EPOS' USB bootloader:
   ```
   $ svn export
   https://svn.lisha.ufsc.br/openepos/epos2/branches/emote3_bootloader/img/emote3_usb
   _bootloader.hex .
   ```

2. Then load it into the device using JLinkExe (see 1.1.1).

1.3. Load EPOS' Network bootloader

1. Get the latest version of EPOS' Network bootloader:
   ```
   $ svn export
   https://svn.lisha.ufsc.br/openepos/epos2/branches/emote3_bootloader/img/emote3_nic
   _bootloader.hex .
   ```

2. Then load it into the device using JLinkExe (see 1.1.1).

1.4. Enable EPOSMote III's ROM Serial bootloader

TODO

2. Using EPOS' USB bootloader

If your device is loaded with EPOS' USB bootloader (as in 1.1.2), you can program your EPOSMote III via USB. You need a copy of EPOS' source and python3 with the pyserial module.

Note: when using EPOSMote III with USB, the "modemmanager" Linux service might get in the way. It is

recommended to stop this service. In Ubuntu, you can do this with the command:

```
$ sudo stop modemmanager
```

You can also disable this service permanently with:

```
# echo "manual" > /etc/init/modemmanager.override
```

1. Conect your EPOSMote III with a USB cable and check which device it shows up as. For example:
   ```
   $ dmesg | tail
   [104871.262738] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
   ```

2. Type in the following command, but do not press Enter:
   ```
   $ sudo python3 tools/emote3_programmer/emote3_programmer.py -d /dev/ttyACM0 -f
   img/<your_application>.hex
   ```

   Then reset the mote and issue the command (press Enter).

After these steps, your application should be running. Everytime the mote is reset, the bootloader will run and wait for a handshake for one second. If none is received and there is an image already loaded, it will start execution of that image.

Note: if you receive a message from python "ImportError: No module named 'serial'", you should install the python pip package. On Ubuntu:

```
sudo apt-get install python3-pip
sudo apt-get install python3-serial
```

Note: Kernel linux-image-3.13.0-65-generic 3.13.0-65 breaks Python based Serial communication. You might need to downgrade your kernel to use the programmer script.

3. Using EPOS' Network bootloader

If your device is loaded with EPOS' Network bootloader (as in 1.1.3), you can program your EPOSMote III via radio. You need a copy of EPOS' source, python3 with the pyserial module and a second mote connected to the PC to act as a programmer.

1. Get the latest version of the code to run on the programmer mote:
   ```
   $ svn export
   https://svn.lisha.ufsc.br/openepos/epos2/branches/emote3_bootloader/img/emote3_nic_prog
   rammer_via_jtag.hex .
   ```

   or: (Warning: the following image is currently unavailable, because programming the NIC programmer using the USB bootloader is not working correctly)
   ```
   $ svn export
   https://svn.lisha.ufsc.br/openepos/epos2/branches/emote3_bootloader/img/emote3_nic_prog
   rammer_via_bootloader.hex .
   ```

2. Load it into the programmer mote using an appropriate method as explained above (JTag, USB ...). If you are programming the programmer mote via JTag, you need the first image. Otherwise, if you are using the USB bootloader, you need the second one. Warning: programming the NIC programmer using the USB bootloader is currently not working correctly
3. Reset the programmer mote, connect it to the PC via USB and wait until it blinks its led a few times and leaves it on (it should take a few seconds).

4. Check which device it shows up as. For example:
   ```
   $ dmesg | tail
   [104871.262738] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
   ```

5. Type in the following command, but do not press Enter:
   ```
   $ sudo python3 tools/emote3_programmer/emote3_programmer.py -d /dev/ttyACM0 -f
   img/<your_application>.hex
   ```

6. Turn on the mote that has the Network bootloader loaded, and issue the command in the previous step (press Enter). It is normal to see a few "Wrong ACK" messages in this step.

After these steps, your application should be running on the mote that has the Network bootloader loaded. Every time the mote is reset, the bootloader will run and wait for a handshake for one second. If none is received and there is an image already loaded, it will start execution of that image.

Note: Kernel linux-image-3.13.0-65-generic 3.13.0-65 breaks Python based Serial communication. You might need to downgrade your kernel to use the programmer script.

4. Using the ROM Serial bootloader

TODO

5. Help!

If you have any questions or problems with this process, please contact me: davir@lisha.ufsc.br