EPOS @ EPOSMote III

Table of contents

EPOSMote III is at the heart of most projects at LISHA's IoT Platform. It is also one of the main target platforms for EPOS 2.

1. Setup

Before reading this page, you are recommended to read the EPOSMoteIII Quick-Start page to learn how to download, compile, and run a simple application on EPOS on EPOSMote III. Also, refer to EPOS User Guide for any question not covered here.

2. Running EPOS on EPOSMote III

For the basic cases, follow the guidelines in EPOSMoteIII Quick-Start to get a program running on EPOSMote III. Remember that bootable ROM images of EPOS for EPOSMote III must be in Intex HEX format. The conversion is transparently managed by `make flash`. The resulting bootable ROM image is saved as `img/<application>.hex`.
There are two ways to upload an image to a mote: via USB or via radio. The next steps depend on which method you are using.

- If your EPOSMote is loaded with the **USB Bootloader** (default), simply plug it into the USB port (the device is assumed to appear on /dev/ttyACM0) and the image just compiled will be flashed. When it is done, execution starts immediately on the device.
- If you are using EPOS **Radio Bootloader**, you need two devices: a programmer and a target. Simply plug the programmer into the USB port (the device is assumed to appear on /dev/ttyACM0) and wait until you see the following messages:

  ```
  /dev/ttyACM0 found, trying to open it
  /dev/ttyACM0 Opened. Sending handshake messages until a response is detected
  ===== This is what the mote said =====
  ```

  Then turn on (or reset) your target device and it shall be programmed by the programmer via radio. When it is done, execution starts immediately on the target device.

If your device is appearing at a port other than ACM0, say ACM2, you can inform that with the `ACM` argument when running `make`, for example:

```
$ make APPLICATION=hello ACM=2 flash
```

Your application should now be running!
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 in flash memory, it will start executing that image.

2.1. 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, you can upload your image using the bootloader. You need to use JTag to accomplish one of the following:

- Load a stand-alone image, with no bootloader;
- Load EPOS' USB bootloader;
- Load EPOS' Network bootloader.

You will need `JLinkExe` to perform these actions. You can install it from the "J-Link Software and Documentation Pack" available for download at 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. This tutorial was last verified with the following JLinkExe version:

```
SEGGER J-Link Commander V6.14h (Compiled May 10 2017 18:39:45)
DLL version V6.14h, compiled May 10 2017 18:39:37
```

2.1.1. Load an image directly

To load an image directly with JTag, you can issue the following JLinkExe commands:

```
$ JLinkExe
J-Link>Device = cc2538sf53
J-Link>connect
TIF>J
JTAGConf><Enter>
Speed><Enter>
J-Link>h
J-Link>erase
J-Link>loadbin <application>.hex, 0
J-Link>exit
```

If everything was successful, you should see a message like this after the loadbin command:

```
Downloading file [<application>.hex]...
Comparing flash   [100%] Done.
Erasing flash     [100%] Done.
Programming flash [100%] Done.
Verifying flash   [100%] Done.
J-Link: Flash download: Flash programming performed for 2 ranges (10240 bytes)
J-Link: Flash download: Total time needed: 0.122s (Prepare: 0.022s, Compare: 0.004s, Erase:
0.000s, Program: 0.081s, Verify: 0.001s, Restore: 0.012s)
O.K.
```

***Note**: EPOS memory map for EPOSMote III assumes there will be a bootloader in place. If you are loading an image that will stand alone, without a bootloader, then the memory map must be adjusted accordingly. Edit `include/machine/cortex/emote3/emote3_traits.h` to make the following modifications and 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;
```

2.1.2. Load EPOS' USB bootloader

Get the latest version of EPOS' USB bootloader at img directory on EPOS Project EmoteIII Bootloader branch at LISHA's GitLab, and then upload it using JLinkExe.

2.1.3. Load EPOS' Network bootloader

Get the latest version of EPOS' Network bootloader at img directory on EPOS Project EmoteIII Bootloader branch at LISHA's GitLab, and then upload it using JLinkExe.

To use the Network bootloader, you will also need a separate mote to act as a Network programmer. Get the latest version of EPOS' Network programmer at img directory on EPOS Project EmoteIII Bootloader branch at LISHA's GitLab, and then upload it using JLinkExe.