SMARC T437X

Build and Install Linux System for SMARC T437X

This document provides instructions for advanced users how Embedian offers patches and builds a customized version of u-boot and linux kernel for Embedian's SMARC T437X product platform and how to install the images to bring the evaluation board up and running.

Our aim is to fully support our hardware through device drivers. We also provide unit tests so that testing a board is easy and custom development can start precisely.

Availability

SMARC-T437X at Embedian

Carrier Board

SBC-SMART-BEE (module and carrier board) at Embedian

SBC-SMART-MEN (module and carrier board) at Embedian

Basic Resources

- ARM Cross Compiler
    - Linaro: https://launchpad.net/linaro-toolchain-binaries
- Bootloader
    - Das U-Boot – the Universal Boot Loader http://www.denx.de/wiki/U-Boot
    - Source – http://git.denx.de/?p=u-boot.git;a=summary
- Linux Kernel
    - Linus's Mainline tree: http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=summary
    - Linux omap tree: http://git.kernel.org/?p=linux/kernel/git/tmlind/linux-omap.git
    - TI Linux source tree: git://git.ti.com/processor-sdk/processor-sdk-linux.git
    - Embedian smarc-437x kernel source tree for linux 4.1.13: http://git.embedian.com/developer/linux-smarc-ti-linux-kernel.git

- ARM based rootfs
  - Debian Squeeze: http://www.debian.org/
  - Uubuntu 14.04: https://rcn-ee.com/rootfs/eewiki/minfs/

ARM Cross Compiler: GCC

This is a pre-built (32bit) version of Linaro GCC that runs on generic linux, so 64bit users need to make sure they have installed the 32bit libraries for their distribution.

debian based	**extra**	**pkgs: (sudo apt-get update ; sudo apt-get install xyz)**
Ubuntu 12.04		ia32-libs
Debian 7 (Wheezy)	sudo dpkg --add-architecture i386	libc6:i386 libstdc++6:i386 libncurses5:i386 zlib1g:i386
Ubuntu 12.10 -> 14.04		libc6:i386 libstdc++6:i386 libncurses5:i386 zlib1g:i386
Red Hat/Centos/Fedora		libstdc++.i686 ncurses-devel.i686 zlib.i686
Red Hat based (rpm)	**extra**	**pkgs: (yum install xyz)**
Red Hat/Centos/Fedora		libstdc++.i686 ncurses-devel.i686 zlib.i686
Ubuntu 12.04		ia32-libs
Ubuntu 12.10 -> 14.04		libc6:i386 libstdc++6:i386 libncurses5:i386 zlib1g:i386

For **u-boot v2015.07** and **Linux kernel v4.1**, use Linaro arm compilier that TI used in their Processor SDK 02.00.01.07

```
$ wget -c https://releases.linaro.org/15.05/components/toolchain/binaries/arm-linux-gnueabihf/gcc-linaro-4.9-2015.05-x86_64_arm-linux-gnueabihf.tar.xz

$ sudo tar -C /opt -xJf gcc-linaro-4.9-2015.05-x86_64_arm-linux-gnueabihf.tar.xz

$ export CC=/opt/gcc-linaro-4.9-2015.05-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

For **u-boot v2016.05** and **Linux kernel v4.4**, use Linaro arm compilier that TI used in their Processor SDK 03.00.00.04

```
$ wget https://releases.linaro.org/components/toolchain/binaries/5.3-2016.02/arm-linux-gnueabihf/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf.tar.xz

$ sudo tar -C /opt -xJf gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf.tar.xz

$ export CC=/opt/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Test:

If this test fails, verify that you have the 32bit libraries installed on your development system.

```
$ ${CC}gcc --version
arm-linux-gnueabihf-gcc (crosstool-NG linaro-1.13.1-4.7-2013.04-20130415 - Linaro GCC 2013.04) 4.7.3 20130328 (prerelease)
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Generating SSH Keys

We recommend you use SSH keys to establish a secure connection between your computer and Embedian Gitlab server. The steps below will walk you through generating an SSH key and then adding the public key to our Gitlab account.

Step 1. Check for SSH keys

First, we need to check for existing ssh keys on your computer. Open up Git Bash and run:

```
$ cd ~/.ssh
$ ls
# Lists the files in your .ssh directory
```

Check the directory listing to see if you have a file named either `id_rsa.pub` or `id_dsa.pub`. If you don't have either of those files go to **step 2**. Otherwise, you already have an existing keypair, and you can skip to **step 3**.

Step 2. Generate a new SSH key

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

```
$ ssh-keygen -t rsa -C "your_email@example.com"
# Creates a new ssh key, using the provided email as a label
# Generating public/private rsa key pair.
# Enter file in which to save the key (/c/Users/you/.ssh/id_rsa): [Press enter]
$ ssh-add id_rsa
```

Now you need to enter a passphrase.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

Which should give you something like this:

```
Your identification has been saved in /c/Users/you/.ssh/id_rsa.
Your public key has been saved in /c/Users/you/.ssh/id_rsa.pub.
The key fingerprint is:
01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your_email@example.com
```

Step 3. Add your SSH key to Embedian Gitlab Server

Copy the key to your clipboard.

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAABDAQABAAABAQDQUEnh8uGpfxaZVU6+uE4bsDrs/tEE5/BPW7jMAxak
6qgOh6nUrQGBWS+VxMM2un3KzwvLRJSj8G4TnTK2CSmlBvR+X8ZeXNTyAdaDxULs/StVhH+QRtFEGy4o
iMIzvIlTyORY89jzhIsgZzwr01nqoSeWWASd+59JWtFjVy0nwVNVtbek7NfuIGGAPaijO5Wnshr2uChB
Pk8ScGjQ3z4VqNXP6CWhCXTqIk7EQl7yX2GKd6FgEFrzae+5Jf63Xm8g6abbE3ytCrMT/jYy5OOj2XSg
6jlxSFnKcONAcfMTWkTXeG/OgeGeG5kZdtqryRtOlGmOeuQe1dd3I+Zz3JyT your_email@example.c
om
```

Go to Embedian Git Server. At Profile Setting --> SSH Keys --> Add SSH Key

Paste your public key and press "Add Key" and your are done.

Bootloader: U-Boot

Clone the U-Boot source code from Embedian Git Server.

For u-boot v2015.07 (Processor-SDK-02.00.01.07):

Download

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git smarc-t437x-uboot
$ cd smarc-t437x-uboot
$ git checkout v2015.07-smarct4x
```

For u-boot v2016.05 (Processor-SDK-03.00.00.04):

Download

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git smarc-t437x-uboot
$ cd smarc-t437x-uboot
$ git checkout v2016.05-smarct4x
```

Configure and Build:

```
$ make ARCH=arm CROSS_COMPILE=${CC} distclean
$ make ARCH=arm CROSS_COMPILE=${CC} smarct437x_evm_spi_uart3_defconfig
$ make ARCH=arm CROSS_COMPILE=${CC}
```


1. If users use other uart ports as their debug port, simply change *uart3* to other index.
2. The *SMARC-T4378* module always boot up from the onboard *SPI NOR* flash. The factory default will be *MLO.byteswap* and *u-boot.img* pre-installed. In some cases when the *SPI NOR* flash is empty or needs to be upgraded. Users can shunt crossed the *TEST#* to ground. In this way, the *SMARC-T4378* module will boot up to carrier SD card. If *TEST#* pin is shunt crossed, the config will be:

$ make ARCH=arm CROSS_COMPILE=${CC} smarct437x_evm_uart3_defconfig

Linux Kernel

For 4.1.13 (Processor-SDK-02.00.01.07, Stable, LTS):

Download:

```
$ git clone git@git.embedian.com:developer/smarc-ti-linux-kernel.git

$ cd smarc-ti-linux-kernel

$ git checkout smarct4x-800-processor-sdk-linux-02.00.01
```


Branch "*smarct4x-800-processor-sdk-linux-02.00.01*" is for *SMARC-T4378-800* variant. If users use *SMARC-T4378-01G,* use "*smarct4x-processor-sdk-linux-02.00.01*" branch instead.

$ git checkout smarct4x-processor-sdk-linux-02.00.01

For 4.4.12 (Processor-SDK-03.00.00.04, Stable, LTS):

Download:

```
$ git clone git@git.embedian.com:developer/smarc-ti-linux-kernel.git

$ cd smarc-ti-linux-kernel

$ git checkout smarct4x-processor-sdk-linux-03.00.00.04
```


SMARC-T4378-800 variant and *SMARC-T4378-01G* both share the same kernel branch "*smarct4x-processor-sdk-linux-03.00.00.04*" at Linux kernel v4.4.12.

Configure and Build

```
$ make ARCH=arm CROSS_COMPILE=${CC} distclean

$ make ARCH=arm CROSS_COMPILE=${CC} smarct437x_defconfig

$ make ARCH=arm CROSS_COMPILE=${CC} zImage modules am437x-smarct437x.dtb
```

Note:

- The kernel sources packaged in this release do not have the required PM firmware binary already copied in the firmware/ folder of the kernel sources. Due to this building the kernel using the default kernel configuration will fail with this error:

```
MK_FW   firmware/am335x-pm-firmware.elf.gen.S
make[2]: *** No rule to make target `firmware/am335x-pm-firmware.elf', needed by
`firmware/am335x-pm-firmware.elf.gen.o'.  Stop.
make[1]: *** [firmware] Error 2
make: *** [uImage] Error 2
```

To resolve this issue, after you clone the kernel sources, copy the am335x-pm-firmware.elf and am43x-evm-scale-data.bin into the firmware/ folder of kernel sources

```
$ cd smarc-ti-linux-kernel/firmware

$ wget http://developer.embedian.com/download/attachments/24608772/am335x-pm-firmware.elf

$ wget http://developer.embedian.com/download/attachments/24608772/am43x-evm-scale-data.bin
```

- If you see the error message like this:

```
"mkimage" command not found - U-Boot images will not be built
```

You can simply install the mkimage by:

```
$ sudo apt-get install uboot-mkimage
```

and make the kernel again.

Root File System

Arago:

User	Password
root	N/A

Processor-SDK-02.00.01.07 Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/arago/smarct437x-rootfs-image-smarct437x.tar.gz
```

Verify:

```
$ md5sum smarct437x-rootfs-image-smarct437x.tar.gz
6f109f7d2d36866e54c4b15fcdc6603c  smarct437x-rootfs-image-smarct437x.tar.gz
```

Processor-SDK-03.00.00.04 Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/arago/smarct437x-rootfs-image-smarct437x-sdk3.tar.gz
```

Verify:

```
$ md5sum smarct437x-rootfs-image-smarct437x-sdk3.tar.gz
5b9cd1fccbcd92b064223def790bccda smarct437x-rootfs-image-smarct437x-sdk3.tar.gz
```

Ubuntu 14.04:

User	Password
root	root
ubuntu	temppwd

Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/trusty/smarct4x-ubuntu-14.04.tar.gz
```

Verify:

```
$ md5sum smarc4x-ubuntu-14.04.tar.gz
2ef6477f7e506e651c2c8a7b4290fefc  smarct4x-ubuntu-14.04.tar.gz
```

Setup SD Card

For these instruction, we are assuming: DISK=/dev/mmcblk0, "lsblk" is very useful for determining the device id.

```
$ export DISK=/dev/mmcblk0
```

Erase SD card:

```
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=16
```

Create Partition Layout:

With util-linux v2.26, sfdisk was rewritten and is now based on libfdisk.

sfdisk
```
$ sudo sfdisk --version
sfdisk from util-linux 2.27.1
```

Create Partitions:

sfdisk >=2.26.x
```
$ sudo sfdisk ${DISK} <<-__EOF__
1M,48M,0xE,*
,,,-
__EOF__
```

sfdisk <=2.25
```
$ sudo sfdisk --in-order --Linux --unit M ${DISK} <<-__EOF__
1,48,0xE,*
,,,-
__EOF__
```

Format Partitions:

```
for: DISK=/dev/mmcblk0
$ sudo mkfs.vfat -F 16 ${DISK}p1 -n boot
$ sudo mkfs.ext4 ${DISK}p2 -L rootfs

for: DISK=/dev/sdX
$ sudo mkfs.vfat -F 16 ${DISK}1 -n boot
$ sudo mkfs.ext4 ${DISK}2 -L rootfs
```

Mount Partitions:

On some systems, these partitions may be auto-mounted...

```
$ sudo mkdir -p /media/boot/
$ sudo mkdir -p /media/rootfs/

for: DISK=/dev/mmcblk0
$ sudo mount ${DISK}p1 /media/boot/
$ sudo mount ${DISK}p2 /media/rootfs/

for: DISK=/dev/sdX
$ sudo mount ${DISK}1 /media/boot/
$ sudo mount ${DISK}2 /media/rootfs/
```

Install Bootloader

If SPI NOR Flash is not empty

The MLO.byteswap and u-boot.img is pre-installed in SPI NOR flash at factory default. SMARC-T4378 is designed to always boot up from SPI NOR flash and to load zImage, device tree blob and root file systems based on the setting of *BOOT_SEL*. If users need to fuse their own u-boot or perform u-boot upgrade. This section will instruct you how to do that.

Copy MLO.byteswap/u-boot.img to the boot partition.

~/smarc-t437x-uboot

```
$ sudo cp -v MLO.byteswap /media/boot/
$ sudo cp -v u-boot.img /media/boot/spi-u-boot.img
```

Fuse MLO.byteswap/u-boot.img to the SPI NOR flash.

Stop at U-Boot command prompt (Press any key when booting up).

u-boot command prompt

```
U-Boot# sf probe
U-Boot# sf erase 0 400000
U-Boot# mmc rescan
U-Boot# fatload mmc 0 ${loadaddr} MLO.byteswap
U-Boot# sf write ${loadaddr} 0 ${filesize}
U-Boot# fatload mmc 0 ${loadaddr} spi-u-boot.img
U-Boot# sf write ${loadaddr} 0x20000 ${filesize}
```

⚠ MLO.byteswap and u-boot.img are from smarct437x_evm_spi_uart3_defconfig

If SPI NOR Flash is empty

In some cases, when SPI NOR flash is erased or the u-boot is under development, we need a way to boot from SD card first. Users need to shunt cross the ***TEST#*** pin to ground. In this way, SMARC-T437X will always boot up from SD card.

Copy MLO/u-boot.img to the boot partition

~/smarc-t437x-uboot

```
$ sudo cp -v MLO /media/boot/
$ sudo cp -v u-boot.img /media/boot/
```

⚠ MLO and u-boot.img will be from smarct437x_evm_uart3_defconfig.

If your u-boot hasn't been finalized and still under development, it is recommended to shunt cross the test pin and boot directly from SD card first. Once your u-boot is fully tested and finalized, you can make smarct437x_evm_spi_uart3_defconfig again fuse your u-boot to SPI NOR flash.

uEnv.txt based bootscript

Create "uEnv.txt" boot script: (vim uEnv.txt)

~/uEnv.txt

```
optargs="consoleblank=0 mem=512M"
#u-boot eMMC specific overrides; Angstrom Distribution (SMARC-T437X) 2014-05-20
kernel_file=zImage
initrd_file=initrd.img

loadaddr=0x82000000
```

```
initrd_addr=0x88080000
fdtaddr=0x88000000
fdtfile=am437x-smarct437x.dtb

initrd_high=0xffffffff
fdt_high=0xffffffff

loadimage=load mmc ${mmcdev}:${mmcpart} ${loadaddr} ${kernel_file}
loadinitrd=load mmc ${mmcdev}:${mmcpart} ${initrd_addr} ${initrd_file}; setenv initrd_size ${filesize}
loadfdt=load mmc ${mmcdev}:${mmcpart} ${fdtaddr} /dtbs/${fdtfile}
#

##Un-comment to enable systemd in Debian Wheezy
#optargs=quiet init=/lib/systemd/systemd

console=ttyS4,115200n8
mmcroot=/dev/mmcblk1p2 ro
mmcrootfstype=ext4 rootwait fixrtc

mmcargs=setenv bootargs console=${console} root=${mmcroot} rootfstype=${mmcrootfstype} ${optargs}

#zImage:
uenvcmd=run loadimage; run loadfdt; run mmcargs; bootz ${loadaddr} - ${fdtaddr}

#zImage + uInitrd: where uInitrd has to be generated on the running system.
#boot_fdt=run loadimage; run loadinitrd; run loadfdt
#uenvcmd=run boot_fdt; run mmcargs; bootz ${loadaddr} ${initrd_addr}:${initrd_size} ${fdtaddr}

###Begin Rootfs from NFS
#serverip=192.168.1.51
#rootpath=/srv/nfs/smarct335x/ubuntu1204/
#nfsopts=nolock,acdirmin=60
#netargs=setenv bootargs console=${console} ${optargs} root=/dev/nfs nfsroot=${serverip}:${rootpath},${nfsopts} rw ip=dhcp
##netboot=echo Loading kernel from SDCARD and booting from NFS ...; run loadimage; run netargs; bootz ${loadaddr} - ${fdtaddr}
##uenvcmd=run netboot
###End Rootfs from NFS

###Begin Load kernel from TFTP
#netmask=255.255.255.0
#ipaddr=192.168.1.65
#serverip=192.168.1.51
#netboot=echo Loading kernel and device tree from TFTP and booting from NFS ...; setenv autoload no; tftp ${loadaddr} ${kernel_file}; tftp
${fdtaddr} ${fdtfile}; run netargs; bootz ${loadaddr} - ${fdtaddr}
#uenvcmd=run netboot
###End Load kernel from TFTP
```


1. If you use SMARC-T4378-800, mem=512M in optargs. Otherwise, mem has to change to 1024M in optargs.
2. mmcroot=/dev/mmcblk1p2 when SDMMC eMMC is not present on carrier board. If there is an eMMC present on carrier board. SD card will be emulated as /dev/mmcblk2 and mmcroot=/dev/mmcblk2p2.

Install Kernel zImage

Copy zImage to the boot partition:

~/smarc-ti-linux-kernel

```
$ sudo cp -v arch/arm/boot/zImage /media/boot
```

Install Kernel Device Tree Binary

```
$ sudo mkdir -p /media/boot/dtbs

$ sudo cp -v arch/arm/boot/dts/am437x-smarct437x.dtb /media/boot/dtbs
```

Install Root File System and Kernel Modules

Copy Root File System:

Processor-SDK-02.00.01.07:

directory where your root file system is

```
$ sudo tar xvfz smarct437x-rootfs-image-smarct437x.tar.gz -C /media/rootfs
```

Ubuntu 14.04:

directory where your root file system is

```
$ sudo tar xvfz smarct4x-ubuntu-14.04.tar.gz -C /media/rootfs
```

Copy Kernel Modules:

~/smarc-ti-linux-kernel

```
$ sudo make ARCH=arm INSTALL_MOD_PATH=/media/rootfs modules_install
```

Networking:

Edit: /etc/network/interfaces

```
$ sudo vim /media/rootfs/etc/network/interfaces
```

Add:

/media/rootfs/etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

Remove SD card:

```
$ sync
$ sudo umount /media/boot
$ sudo umount /media/rootfs
```

Setup eMMC

Setting up eMMC usually is the last step at development stage after the development work is done at your SD card or NFS environments. eMMC on module will be always emulated as /dev/mmcblk0. Setting up eMMC now is nothing but changing the device descriptor.

This section gives a step-by-step procedure to setup eMMC flash. Users can write a shell script your own at production to simplify the steps.

First, we need to backup the final firmware from your SD card or NFS.

Prepare for eMMC binaries from SD card (or NFS):

Insert SD card into your Linux PC. For these instructions, we are assuming: DISK=/dev/mmcblk0, "lsblk" is very useful for determining the device id.

For these instruction, we are assuming: DISK=/dev/mmcblk0, "lsblk" is very useful for determining the device id.

```
$ export DISK=/dev/mmcblk0
```

Mount Partitions:

On some systems, these partitions may be auto-mounted...

```
$ sudo mkdir -p /media/boot/
$ sudo mkdir -p /media/rootfs/

for: DISK=/dev/mmcblk0
$ sudo mount ${DISK}p1 /media/boot/
$ sudo mount ${DISK}p2 /media/rootfs/

for: DISK=/dev/sdX
$ sudo mount ${DISK}1 /media/boot/
$ sudo mount ${DISK}2 /media/rootfs/
```

Copy zImage to rootfs partition:

```
$ sudo cp -v /media/boot/zImage /media/rootfs/home/root
```


Note

1. If your rootfs is Ubuntu 14.04, copy to */media/rootfs/home/ubuntu* instead of /media/rootfs/home/root

Copy uEnv.txt to rootfs partition:

Copy and paste the following contents to /media/rootfs/home/root ($ sudo vim /media/rootfs/home/root/uEnv.txt)

~/uEnv.txt

```
optargs="consoleblank=0 mem=512M"
#u-boot eMMC specific overrides; Angstrom Distribution (SMARC-T437X) 2014-05-20
kernel_file=zImage
initrd_file=initrd.img

loadaddr=0x82000000
initrd_addr=0x88080000
fdtaddr=0x88000000
fdtfile=am437x-smarct437x.dtb

initrd_high=0xffffffff
fdt_high=0xffffffff

loadimage=load mmc ${mmcdev}:${mmcpart} ${loadaddr} ${kernel_file}
loadinitrd=load mmc ${mmcdev}:${mmcpart} ${initrd_addr} ${initrd_file}; setenv initrd_size ${filesize}
loadfdt=load mmc ${mmcdev}:${mmcpart} ${fdtaddr} /dtbs/${fdtfile}
#

##Un-comment to enable systemd in Debian Wheezy
#optargs=quiet init=/lib/systemd/systemd

console=ttyS4,115200n8
mmcroot=/dev/mmcblk0p2 ro
mmcrootfstype=ext4 rootwait fixrtc

mmcargs=setenv bootargs console=${console} root=${mmcroot} rootfstype=${mmcrootfstype} ${optargs}

#zImage:
```

```
uenvcmd=run loadimage; run loadfdt; run mmcargs; bootz ${loadaddr} - ${fdtaddr}

#zImage + uInitrd: where uInitrd has to be generated on the running system.
#boot_fdt=run loadimage; run loadinitrd; run loadfdt
#uenvcmd=run boot_fdt; run mmcargs; bootz ${loadaddr} ${initrd_addr}:${initrd_size} ${fdtaddr}

###Begin Rootfs from NFS
#serverip=192.168.1.51
#rootpath=/srv/nfs/smarct335x/ubuntu1204/
#nfsopts=nolock,acdirmin=60
#netargs=setenv bootargs console=${console} ${optargs} root=/dev/nfs nfsroot=${serverip}:${rootpath},${nfsopts} rw ip=dhcp
##netboot=echo Loading kernel from SDCARD and booting from NFS ...; run loadimage; run netargs; bootz ${loadaddr} - ${fdtaddr}
##uenvcmd=run netboot
###End Rootfs from NFS

###Begin Load kernel from TFTP
#netmask=255.255.255.0
#ipaddr=192.168.1.65
#serverip=192.168.1.51
#netboot=echo Loading kernel and device tree from TFTP and booting from NFS ...; setenv autoload no; tftp ${loadaddr} ${kernel_file}; tftp
${fdtaddr} ${fdtfile}; run netargs; bootz ${loadaddr} - ${fdtaddr}
#uenvcmd=run netboot
###End Load kernel from TFTP
```


1. If you use SMARC-T4378-800, mem=512M in optargs. Otherwise, mem has to change to 1024M in optargs.
2. The uEnv.txt is exactly the same as that is created in SD card except the eMMC device descriptor now is mmcroot=/dev/mmcblk0p2.

Copy device tree blob to rootfs partition:

```
$ sudo cp -v /media/boot/dtbs/am437x-smarct437x.dtb /media/rootfs/home/root/am437x-smarct437x.dtb
```

Copy real rootfs to rootfs partition:

Yocto Built Root File Systems

```
$ pushd /media/rootfs

$ sudo tar cvfz ~/smarct437x-emmc-rootfs.tar.gz .

$ sudo mv ~/smarct437x-emmc-rootfs.tar.gz /media/rootfs/home/root

$ popd
```

Ubuntu 14.04 Root File Systems

```
$ sudo vim /media/rootfs/etc/udev/rules.d/70-persistent-net.rules

Delete all contents starting with "SUBSYSTEM=="

$ pushd /media/rootfs

$ sudo tar cvfz ~/smarct437x-emmc-rootfs.tar.gz .

$ sudo mv ~/smarct437x-emmc-rootfs.tar.gz /media/rootfs/home/root

$ popd
```

Remove SD card:

```
$ sync
$ sudo umount /media/boot
$ sudo umount /media/rootfs
```

Copy Binaries to eMMC from SD card:

Insert this SD card into your SMARC-T437X device (carrier board).

Now it will be almost the same as you did when setup your SD card, but the eMMC device descriptor is */dev/mmcblk0* now. (SD card will be /dev/mmcblk1)

```
$ export DISK=/dev/mmcblk0
```

Erase SD card:

```
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=16
```

Create Partition Layout:

```
$ sudo sfdisk --in-order --Linux --unit M ${DISK} <<-__EOF__
1,48,0x83,*
,,,-
__EOF__
```

Format Partitions:

```
$ sudo mkfs.vfat -F 16 ${DISK}p1 -n boot
$ sudo mkfs.ext4 ${DISK}p2 -L rootfs
```

Mount Partitions:

```
$ sudo mkdir -p /media/boot/
$ sudo mkdir -p /media/rootfs/
$ sudo mount ${DISK}p1 /media/boot/
$ sudo mount ${DISK}p2 /media/rootfs/
```

Install binaries for partition 1

Copy uEnv.txt/zImage/*.dtb to the boot partition

```
$ sudo cp -v zImage uEnv.txt /media/boot/
```

Install Kernel Device Tree Binary

```
$ sudo mkdir -p /media/boot/dtbs

$ sudo cp -v am437x-smarct437x.dtb /media/boot/dtbs
```

Install Root File System

```
$ sudo tar -zxvf smarct437x-emmc-rootfs.tar.gz -C /media/rootfs
```

Unmount eMMC:

```
$ sync
$ sudo umount /media/boot
$ sudo umount /media/rootfs
```

Switch your Boot Select to eMMC and you will be able to boot up from SPI NOR flash and import u-boot environmental parameters and load kernel zImage and device tree blob from eMMC now.

==End of Document==

Last updated 2016-08-24