SMARC T335X

Build and Install Linux System for SMARC T335X

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 T335X 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 T335X 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: http://git.ti.com/git/ti-linux-kernel/ti-linux-kernel.git
    - TI's overall Processor SDK build and test process: http://arago-project.org/git/projects/tisdk-build-scripts.git

    - Arago Project TI Staging tree: http://arago-project.org/git/projects/?p=linux-am33x.git;a=shortlog;h=refs/heads/v3.2-staging
    - Embedian smarc-t335x kernel source tree for linux 3.2: http://git.embedian.com/developer/linux-smarc-t335x-v3.2.git
    - Embedian smarc-t335x kernel source tree for linux 3.12 and after: http://git.embedian.com/developer/smarc-ti-linux-kernel
- ARM based rootfs
    - Debian Squeeze: http://www.debian.org/

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

To build Embedian's smarct335x u-boot and linux kernel, you will need to install the Linaro arm compiler that TI used for their release:

For **u-boot v2017.01** and **Linux kernel v4.9.41**, use Linaro arm compilier that TI used in their Processor SDK 04.01.00.06

```
$ wget -c http://releases.linaro.org/components/toolchain/binaries/6.2-2016.11/arm-linux-gnueabihf/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz

$ sudo tar -C /opt -xJf gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz

$ export CC=/opt/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

For **u-boot v2016.05** and **Linux kernel v4.4.12**, use Linaro arm compilier that TI used in their Processor SDK 03.00.00.04

```
$ wget
http://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-
```

For **u-boot 2015.07** and **Linux 4.1.10**, you need to use the following newer Linaro arm compilier that TI used in their Processor SDK 02.00.01.07.

```
$ wget -c http://releases.linaro.org/archive/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 2014.04**, **u-boot 2014.07**, **Linux 3.2**, **Linux 3.12** and **Linux 3.14**, use the following newer Linaro arm compilier that TI used in theirAMSDK 6 and AMSDK7..

```
$ wget http://releases.linaro.org/archive/13.04/components/toolchain/binaries/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux.tar.xz

$ sudo tar -C /opt -xJf gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux.tar.xz

$ export CC=/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/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.

Download:

For u-boot v2017.01 (Processor-SDK-04.01.00.06):

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git
$ cd smarc-t335x-uboot
$ git checkout v2017.01-smarct3x
```

If Boot up from eMMC, change #define *CONFIG_SYS_MMC_ENV_DEV* from 0 to **1** in include/configs/smarct335x_evm.h file and compile again.

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

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git
$ cd smarc-t335x-uboot
$ git checkout v2016.05-smarct3x
```

For u-boot v2015.07 (Processor-SDK-02.00.00.00):

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git
$ cd smarc-t335x-uboot
$ git checkout v2015.07-smarct33
```

```
If Boot up from eMMC, change mmcdev=0 to mmcdev=1 in include/configs/smarct335x_evm.h file and compile
again.
```

Configure and Build:

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

$ make ARCH=arm CROSS_COMPILE=${CC} smarct335x_evm_uart3_defconfig

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

For u-boot v2014.07 (Processor-SDK-01.00.00.03):

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git
$ cd smarc-t335x-uboot
$ git checkout v2014.07-smarct33
```

For u-boot v2014.04 (AMSDK6 and AMSDK7):

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git
$ cd smarc-t335x-uboot
$ git checkout v2014.04-smarct33
```

```
For SBC-SMART-MEN

$ git checkout v2014.04-smartmen
```

Configure and Build:

```
$ make ARCH=arm CROSS_COMPILE=${CC} distclean
$ make ARCH=arm CROSS_COMPILE=${CC} smarct335x_evm_uart3_config
$ make ARCH=arm CROSS_COMPILE=${CC}
```

The uart**x** in u-boot config (defconfig) file is to specify the debug console outputs.

Linux Kernel

Download:

For 4.9.41 (Processor-SDK-04.01.00.06, Stable, LTS):

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

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

```
$ git checkout smarct3x-processor-sdk-04.01.00.06
```

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

```
$ git clone git@git.embedian.com:developer/smarc-ti-linux-kernel.git
$ cd smarc-ti-linux-kernel
$ git checkout smarct3x-processor-sdk-linux-03.00.00.04
```

For 4.1.y (Processor-SDK-02.00.01.07, Stable, LTS):

```
$ git clone git@git.embedian.com:developer/smarc-ti-linux-kernel.git
$ cd smarc-ti-linux-kernel
$ git checkout smarct3x-processor-sdk-linux-02.00.01
```

For 3.14.y (Processor-SDK-01.00.00.03, Stable, LTS):

```
$ git clone git@git.embedian.com:developer/smarc-ti-linux-kernel.git
$ cd smarc-ti-linux-kernel
$ git checkout smarc-ti-linux-3.14.y
```

For 3.12.y (AMSDK7, Stable, LTS):

```
$ git clone git@git.embedian.com:developer/smarc-ti-linux-kernel.git
$ cd smarc-ti-linux-kernel
$ git checkout smarc-ti-linux-3.12.y
```

Configure and Build (v3.12 and later):

```
$ make ARCH=arm CROSS_COMPILE=${CC} distclean
$ make ARCH=arm CROSS_COMPILE=${CC} smarc_t335x_defconfig
$ make ARCH=arm CROSS_COMPILE=${CC} zImage modules am335x-smarct335x.dtb
```

For 3.2 (AMSDK6, Stable, LTS release):

```
$ git clone git@git.embedian.com:developer/linux-smarc-t335x-v3.2.git
$ cd linux-smarc-t335x-v3.2
$ git checkout v3.2_SMARCT335xPSP_04.06.00.11
```

For board SBC-SMART-MEN:

```
$ git checkout v3.2_SBC_SMARTMEN
```

Configure and Build:

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

$ make ARCH=arm CROSS_COMPILE=${CC} smarc_t335x_defconfig

$ make ARCH=arm CROSS_COMPILE=${CC} uImage modules
```

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

To resolve this for kernel 3.2, after you clone the kernel sources, copy the firmware binary into the firmware/ folder of kernel sources

```
$ cd linux-smarc-t335x-v3.2/firmware

$ wget http://developer.embedian.com/download/attachments/2883656/am335x-pm-firmware.bin
```

To resolve this for kernel 3.14, kernel 4.1, kernel 4.4 and kernel 4.9, after you clone the kernel sources, copy the firmware elf binary into the firmware/ folder of kernel sources

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

$ wget http://developer.embedian.com/download/attachments/2883656/am335x-pm-firmware.elf

$ wget http://developer.embedian.com/download/attachments/2883656/am335x-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-04.01.00.06 Download:

```
$ wget -c
ftp://ftp.embedian.com/public/dev/minfs/arago/processor_sdk_04.01.00.06/smarct335x-rootfs-image-smarct33
5x.tar.xz
```

Verify:

```
$ md5sum smarct335x-rootfs-image-smarct335x.tar.xz
e698ddf06918e5eb864d27d717b21f7c  smarct335x-rootfs-image-smarct335x.tar.xz
```

Processor-SDK-03.00.00.04 Download:

```
$ wget -c
ftp://ftp.embedian.com/public/dev/minfs/arago/processor_sdk_03.00.00.04/smarct335x-rootfs-image-smarct33
5x.tar.gz
```

Verify:

```
$ md5sum smarct335x-rootfs-image-smarct335x.tar.gz
788011e063b79ee8a18c1508f39aa9f1  smarct335x-rootfs-image-smarct335x.tar.gz
```

Processor-SDK-02.00.01.07 Download:

```
$ wget -c
ftp://ftp.embedian.com/public/dev/minfs/arago/processor_sdk_02.00.01.07/smarct335x-rootfs-image-smarct33
5x.tar.gz
```

Verify:

```
$ md5sum smarct335x-rootfs-image-smarct335x.tar.gz
ea0c7490047314d125c20231ad9eaa78  smarct335x-rootfs-image-smarct335x.tar.gz
```

SDK7 Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/arago/sdk7/smarct335x-rootfs-image-smarct335x.tar.gz
```

Verify:

```
$ md5sum smarct335x-rootfs-image-smarct335x.tar.gz
ee272266a6bbeb718c129f51f71c52f5  smarct335x-rootfs-image-smarct335x.tar.gz
```

Ubuntu 16.04:

User	Password
root	root

ubuntu	temppwd

Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/ubuntu/xenial/smarct3x-ubuntu-16.04.tar.gz
```

Verify:

```
$ md5sum smarct3x-ubuntu-16.04.tar.gz
957625f56f786a22d44b60480155cfd9  smarct3x-ubuntu-16.04.tar.gz
```

Ubuntu 14.04:

User	Password
root	root
ubuntu	temppwd

Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/ubuntu/trusty/smarct3x-ubuntu-14.04.tar.gz
```

Verify:

```
$ md5sum smarct3x-ubuntu-14.04.tar.gz
05db5b85224e84e9898a1c5925703b8b  smarct3x-ubuntu-14.04.tar.gz
```

Debian 9.8:

User	Password
root	root
debian	temppwd

Download:

```
$ wget -c
ftp://ftp.embedian.com/public/dev/minfs/debian/stretch/smarct3x-debian-9.8-armhf-2019-02-16.tar.gz
```

Verify:

```
$ md5sum smarct3x-debian-9.8-armhf-2019-02-16.tar.gz
8969f249307d9b2c2ef5c76f7ca1c6b8 smarct3x-debian-9.8-armhf-2019-02-16.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

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

~/smarc-t335x-uboot

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

uEnv.txt based bootscript

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

For v3.2:

~/uEnv.txt

```
optargs="consoleblank=0 mem=512M"
bootfile=zImage
loadaddr=0x80200000
#fdtaddr=0x80F80000
#fdtfile=am335x-smarct335x.dtb
console=ttyO3,115200n8
mmcroot=/dev/mmcblk0p2 rw
mmcrootfstype=ext4 rootwait fixrtc

#To boot old v3.2.x based kernel enable: (SMARC T335X and BeagleBone)
uenvcmd=run loadimage; run mmc_classic_boot

#For u-boot 13.10
#uenvcmd=run loadzimage; run mmc_classic_boot

###Begin Rootfs from NFS
#serverip=192.168.1.51
#rootpath=/srv/nfs/smarct335x/arago6/
#nfsopts=nolock
#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 loaduimage; run netargs; bootz ${loadaddr}
##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 from TFTP and booting from NFS ...; setenv autoload no; tftp ${loadaddr} ${bootfile}; run netargs; bootz
${loadaddr}
#uenvcmd=run netboot
###End Load kernel from TFTP
```

For 3.12.y or after :

~/uEnv.txt

```
optargs="consoleblank=0 mem=512M"
#u-boot eMMC specific overrides; Angstrom Distribution (SMARC T335X) 2014-05-20
kernel_file=zImage
initrd_file=initrd.img

loadaddr=0x82000000
initrd_addr=0x88080000
fdtaddr=0x88000000
fdtfile=am335x-smarct335x.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=ttyS3,115200n8
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
```

⚠ For kernel **3.12.y,** the serial port device descriptor changes from ***ttyS*** to ***ttyO***

The above uEnv.txt file needs to change accordingly. (ttyS3 --> ttyO3)

SD card will always be emulated as /dev/mmcblk1 at kernel 3.12 and newer version

Copy uEnv.txt to the boot partition:

~/

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

Install Kernel zImage

Copy zImage to the boot partition:

~/linux-smarc-t335x-v3.x (v3.2) or ~/smarc-ti-linux-kernel (v3.12 or after)

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

Only for v3.12.y or after:

Install Kernel Device Tree Binary

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

$ sudo cp -v arch/arm/boot/dts/am335x-smarct335x.dtb /media/boot/dtbs
```

Install Root File System and Kernel Modules

Copy Root File System:

Arago SDK7:

directory where your root file system is

```
$ sudo tar xvfz smarct335x-rootfs-image-smarct335x-sdk7.tar.gz -C /media/rootfs
```

Arago SDK6:

directory where your root file system is

```
$ sudo tar xvfz smarct335x-rootfs-image-smarct335x-sdk6.tar.gz -C /media/rootfs
```

Ubuntu 14.04:

directory where your root file system is

```
$ sudo tar xvfz smarc-ubuntu14.04.tar.gz -C /media/rootfs
```

Copy Kernel Modules:

~/linux-smarc-t335x-v3.2 (v3.2) or ~/smarc-ti-linux-kernel (v3.12 or after)

```
$ 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
```

If your rootfs is ubuntu, before un-mounting it, check

/media/rootfs/etc/fstab to see if the mounting point is /dev/mmcblk1p2 and

/media/rootfs/etc/init/serial.conf to see if your console port device descriptor correct or not.

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 zImage to rootfs partition:

```
$ sudo cp -v <kernel source dorectory>/arch/arm/boot/zImage /media/rootfs/home/root
```

For kernel v3.12.y or later:

```
$ sudo cp -v /media/boot/dtbs/am335x-smarct335x.dtb /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)

For kernel v3.2:

```
optargs="consoleblank=0 mem=512M"
bootfile=zImage
loadaddr=0x80200000
#fdtaddr=0x80F80000
#fdtfile=am335x-smarct335x.dtb
console=ttyO3,115200n8
mmcroot=/dev/mmcblk0p2 rw
mmcrootfstype=ext4 rootwait fixrtc

#To boot old v3.2.x based kernel enable: (SMARC T335X and BeagleBone)
uenvcmd=run loadimage; run mmc_classic_boot

#For u-boot 13.10
#uenvcmd=run loadzimage; run mmc_classic_boot

###Begin Rootfs from NFS
#serverip=192.168.1.51
#rootpath=/srv/nfs/smarct335x/arago6/
#nfsopts=nolock
#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 loaduimage; run netargs; bootz ${loadaddr}
##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 from TFTP and booting from NFS ...; setenv autoload no; tftp ${loadaddr} ${bootfile}; run netargs; bootz
${loadaddr}
#uenvcmd=run netboot
###End Load kernel from TFTP
```

For kernel v3.12.y+:

```
optargs="consoleblank=0 mem=512M"
#u-boot eMMC specific overrides; Angstrom Distribution (SMARC T335X) 2014-05-20
kernel_file=zImage
initrd_file=initrd.img

loadaddr=0x82000000
initrd_addr=0x88080000
fdtaddr=0x88000000
fdtfile=am335x-smarct335x.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=ttyS3,115200n8
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
```


For kernel **3.12.y** and after, the serial port device descriptor changes from ***ttyS*** to ***ttyO***

The above uEnv.txt file needs to change accordingly. (ttyS3 --> ttyO3)

The uEnv.txt file only changes mmcroot from /dev/mmcblk1 (SD) to /dev/mmcblk0 (eMMC)

Copy real rootfs to rootfs partition:

```
$ pushd /media/rootfs

$ sudo tar cvfz ~/smarct335x-emmc-rootfs.tar.gz .

$ sudo mv ~/smarct335x-emmc-rootfs.tar.gz /media/rootfs/home/root

$ popd
```

Remove SD card:

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

Insert this SD card into your SMARC T335X device and boot up the devices from SD card.

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

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

Erase eMMC Flash:

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

Create Partition Layout:

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

In SDK6, Arago rootfs will mount partition2 as /media/mmcblk0p2 automatically after executing the above command. Umount the partition first.

```
$ sudo umount /media/mmcblk0p2
```

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 MLO/u-boot.img/uEnv.txt/zImage to the boot partition

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

The U-Boot for SD Boot up and eMMC Boot up has one line difference in include/configs/smarct335x_evm.h. The mmcdev is 0 for SD boot up and is 1 for eMMC boot up.

Only for v3.12.y or after:

Install Kernel Device Tree Binary

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

$ sudo cp -v am335x-smarct335x.dtb /media/boot/dtbs
```

Install Root File System

```
$ sudo tar -zxvf smarct335x-emmc-rootfs.tar.gz -C /media/rootfs
```

Unmount eMMC:

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

⚠ If your rootfs id Ubuntu, you need to modify /media/rootfs/etc/fstab and change the mount point from /dev/mmcblk1p2 to /dev/mmcblk0p2 before un-mounting it.

Switch your Boot Select to eMMC and you will be able to boot up from eMMC now.

Last updated 2018-10-04