3.7. Flash Linux

This section documents how to set up OSPI MTD partitions, generate UBI root filesystem(ospi.rootfs), flash files (u-boot/kernel/dtb/rootfs) to the OSPI flash, and perform full linux boot including a UBI root filesystem mounted on OSPI MTD partition. For simplicity we assume the files are being loaded from an SD card, but these files can just as easily be loaded via other media that exists.

3.7.1. OSPI MTD re-partition

OSPI MTD re-partitions

In this example, the kernel/dtb will be first flashed to, and later booted directly from fixed MTD partions. By doing so it is possible to write/read kernel/dtb to/from OSPI flash via primitive u-boot “spi…” cmd. In this example, bootargs “mtdparts” is used to re-partition the default OSPI MTD partitions with addition of two new MTD partitions for kernel and dtb.

```
=> setenv mtdparts mtdparts=fc40000.spi.0:512k(ospi.tiboot3),2m(ospi.tispl),4m(ospi.u-
boot),256k(ospi.env),256k(ospi.env.backup),1m(ospi.dtb),20m@8m(ospi.kernel),36608k(ospi.rootfs),256k(ospi.ph

=> mtd list
List of MTD devices:
* nor0
- type: NOR flash
- block size: 0x40000 bytes
- min I/O: 0x1 bytes
- 0x000000000000-0x000004000000 : "nor0"
        - 0x000000000000-0x000000080000 : "ospi.tiboot3"
        - 0x000000080000-0x000000280000 : "ospi.tispl"
        - 0x000000280000-0x000000680000 : "ospi.u-boot"
        - 0x000000680000-0x0000006c0000 : "ospi.env"
        - 0x0000006c0000-0x000000700000 : "ospi.env.backup"
        - 0x000000700000-0x000000800000 : "ospi.dtb"
        - 0x000000800000-0x000001c00000 : "ospi.kernel"
        - 0x000001c00000-0x000003fc0000 : "ospi.rootfs"
        - 0x000003fc0000-0x000004000000 : "ospi.phypattern
```

OSPI MTD partition for ospi.rootfs

In preparation to generate UBI rootfs, let’s continue to boot the kernel from SD card. In this example, “mtdinfo -a” after kernel boot-up is used to list MTD partition information, where mtd7 partition (ospi.rootfs) parameters will be used next to create UBI rootfs for OSPI flash on AM64x GP EVM.

```
root@am64xx-evm:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes
...
mtd7
Name:                           ospi.rootfs
Type:                           nor
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          143 (37486592 bytes, 35.7 MiB)
Minimum input/output unit size: 16 bytes
Sub-page size:                  16 bytes
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true
...
```

3.7.2. UBIFS/UBI rootfs files generation

The following two commands running on Linux host machine are used to generate UBIFS/UBI rootfs files for AM64x EVM on-board OSPI flash, where the tiny file system “sdk-tiny-fs” is first untared from the file “tisdk-tiny-image-am64xx-evm.tar.xz” that is included in the AM64x Processor SDK Linux package.

```
1. "mkfs.ubifs -r sdk-tiny-fs -o am6-sdk-tiny-fs.ubifs -m 16 -e 262016 -c 143 -v"
2. "ubinize -o am6-sdk-tiny-fs.ubi -m 16 -s 16 -p 256KiB ubinize.cfg -v"
ubinize.cfg:
[ubifs]
mode=ubi
image=am6-sdk-tiny-fs.ubifs
vol_id=0
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
```

The parameters used in the above two commands (mkfs.ubifs/ubinize) are based on OSPI mtd7 partition (ospi.rootfs) information from the previous step:

```
1. mkfs.ubifs -r sdk-tiny-fs -o am6-sdk-tiny-fs.ubifs -m 16 -e 262016 -c 143 -v"
- "Minimum input/output unit size: 16 bytes"               => "-m 16"
- 262144-128 (see the note below on ubi headers) = 262016   => "-e 262016"
- "Amount of eraseblocks:   143 (37486592 bytes, 35.7 MiB)" => "-c 143"
2. "ubinize -o am6-sdk-tiny-fs.ubi -m 16 -s 16 -p 256KiB ubinize.cfg -v"
- "Minimum input/output unit size: 16 bytes"               => "-m 16"
- "Sub-page size:                  16 bytes"               => "-s 16"
- "Eraseblock size:                262144 bytes, 256.0 KiB" => "-p 256KiB"
```

Note on “ubi header” from http://www.linux-mtd.infradead.org/doc/ubi.html#L_ubi_headers

UBI stores 2 small 64-byte headers at the beginning of each non-bad physical eraseblock:

1. Erase counter header (or EC header) which contains the erase counter of the physical eraseblock (PEB) plus other information;
2. volume identifier header (or VID header) which stores the volume ID and the logical eraseblock (LEB) number to which this PEB belongs.

3.7.3. OSPI flashing

u-boot/kernel/dtb OSPI flashing at u-boot prompt

u-boot/kernel/dtb can be flashed to OSPI at u-boot prompt, where u-boot booting from SD card is used:

1. Flash u-boot (tiboot3.bin/tispl.bin/u-boot.img) from “mmc 1:1” to OSPI

```
=> mmc rescan
=> sf probe
=> load mmc 1:1 ${loadaddr} tiboot3.bin
=> sf update $loadaddr 0x0 $filesize
=> load mmc 1:1 ${loadaddr} tispl.bin
=> sf update $loadaddr 0x80000 $filesize
=> load mmc 1:1 ${loadaddr} u-boot.img
=> sf update $loadaddr 0x280000 $filesize
```

2. Flash kernel/dtb from “mmc 1:2 boot” to OSPI

```
=> load mmc 1:2 ${loadaddr} boot/Image
19137024 bytes read in 800 ms (22.8 MiB/s)
=> sf update $loadaddr 0x800000 $filesize
device 0 offset 0x800000, size 0x1240200
=> load mmc 1:2 ${loadaddr} boot/k3-am642-evm.dtb
53223 bytes read in 5 ms (10.2 MiB/s)
=> sf update $loadaddr 0x700000 $filesize
device 0 offset 0x700000, size 0xcfe7
```

Note the size of the kernel (Image) and the size of the dtb (k3-am642-evm.dtb). These size parameters will be used later when booting kernel/dtb from OSPI.

UBI rootfs OSPI flashing at kernel

One option to flash UBI rootfs “am6-sdk-tiny-fs.ubi” to OSPI uses the kernel mtd utility after the kernel starts running.

```
root@am64xx-evm:/boot# ubiformat /dev/mtd7 -f am6-sdk-tiny-fs.ubi
```

The following commands show attaching mtd7 (ospi.rootfs) for testing purpose

```
root@am64xx-evm:/boot# ubiattach -m 7
root@am64xx-evm:/boot# ubinfo -a
root@am64xx-evm:/boot# mkdir /mnt/fs_test
root@am64xx-evm:/boot# mount -t ubifs ubi0 /mnt/fs_test
root@am64xx-evm:/boot# cd /mnt/fs_test
root@am64xx-evm:/boot# ls -lrt
```

3.7.4. Boot from OSPI

After all files are flashed to OSPI, configure BOOTMODE pin to boot u-boot from OSPI. The default u-boot environment currently does not contain an OSPI kernel boot command. In this example, for simplicity, run the following u-boot commands to boot kernel/dtb from OSPI. Alternatively these environment variable and bootargs can be built into u-boot itself.

1. Run the following u-boot commands to repartition OSPI MTD partitions:

```
setenv mtdparts mtdparts=fc40000.spi.0:512k(ospi.tiboot3),2m(ospi.tispl),4m(ospi.u-
boot),256k(ospi.env),256k(ospi.env.backup),1m(ospi.dtb),20m@8m(ospi.kernel),36608k(ospi.rootfs),256k(
```

2. Run the following u-boot commands to set up bootargs/optargs for mounting rootfs from the OSPI MTD partition:

```
run args_all; setenv bootargs console=${console} ${optargs} root=ubi0:rootfs rw
ubi.mtd=ospi.rootfs rootfstype=ubifs rootwait=1
```

3. Run the following u-boot commands to load/run kernel and dtb from OSPI to DDR, and then boot the kernel. The size parameters used in this example are from the kernel/dtb file size in the previous OSPI flashing section:

```
sf probe; sf read ${loadaddr} 0x800000 0x1240200; sf read ${fdtaddr} 0x700000 0xcfe7; run run_kern
```