NT96680 AN Boot from EMMC

Table of Content

1 Features

- Support Loader and system image in EMMC storage.
- Support movie recording in both SD and emmc storage.
- Provide a tool to do multiple partitions layout format.
- Support system recovery if boot crash

2 System configuration

2.1 configs

We are already added EMMC boot model as below.
sdk/configs/cfg_EMMC_EVB. nvt-na51000-peri.dtsi, nvt-na51000-storage-partition.dtsi and nvt-na51000-nvtpack.dtsi are different from normal config of nand boot.

2.2 Compilation

According Linux user guide to build overall system image as below:

```
$ make all
```

2.3 nvt-na51000-peri.dtsi

EMMC must always use the SDIO3 that need to be pluged in nvt-na51000-peri.dtsi.

```
aliases {
        mmc0 = &mmc0;   /* Fixed to mmcblk0 for sdio1 */
        mmc1 = &mmc1;   /* Fixed to mmcblk1 for sdio2 */
        mmc2 = &emmc; /* Fixed to mmcblk2 for sdio3 */
};

emmc: mmc@f0510000 {
        #address-cells = <2>;
        #size-cells = <2>;
        compatible = "nvt,nvt_mmc3";
        reg = <0xf0510000 0x1000>;
        interrupts = <GIC_SPI 32 IRQ_TYPE_LEVEL_HIGH>;
        max-frequency = <48000000>;
        voltage-switch = <1>;
        max-voltage = <3300>;
        bus-width = <8>;
        driving = <20 15 15 20 15 15 30 25 25 40 30 30>;
        cd_gpio = <0 GPIO_FALLING_EDGE GPIO_INTERRUPT>;
        /*ro_gpio = <D_GPIO(3) GPIO_RISING_EDGE GPIO_POLLING>;*/
        /*power_en = <D_GPIO(4) GPIO_RISING_EDGE>;*/
};
```

2.4 nvt-na51000-storage-partition.dtsi

```
&emmc {
     partition_mbr {       label = "mbr";              reg = <0x0 0x00000000 0x0 0x00000200>; };
     partition_fdt {       label = "fdt";              reg = <0x0 0x00040000 0x0 0x00040000>; };
     partition_fdt.restore { label = "fdt.restore";      reg = <0x0 0x00080000 0x0 0x00040000>; };
     partition_uboot {     label = "uboot";            reg = <0x0 0x000C0000 0x0 0x00200000>; };
     partition_uenv {      label = "uenv";             reg = <0x0 0x002C0000 0x0 0x00040000>; };
```

```
        partition_kernel {     label = "kernel";              reg = <0x0 0x00300000 0x0 0x00400000>; };

        partition_rootfs {     label = "rootfs";  active="true"; reg = <0x0 0x00700000 0x0 0x04000000>; };

        partition_rootfs1 {    label = "rootfs1";             reg = <0x0 0x04700000 0x0 0x05000000>; };

        partition_rootfs2 {    label = "rootfs2";             reg = <0x0 0x09700000 0x0 0x0CC00000>; };

        partition_mbr1 {       label = "mbr1";                reg = <0x0 0x16300000 0x0 0x00400000>; };

        partition_rootfsl1{    label = "rootfsl1";            reg = <0x0 0x16700000 0x0 0x00000000>; };

};
```

The difference with nand boot is MBR image, this image can be used to generate DOS partition table.

The important thing of the EMMC boot is partitioning EMMC, this step will generate several MBR images which is according to nvt-na51000-storage-partition.dtsi.

The partition_mbr is emmc boot special type, please don't modify it to avoid system fault, In our design, we will use three primary partitions and multiple logical partitions model. You can see rootfs0, rootfs1 and rootfs2 three primary partitions, and rootfsl1 is a logical partition. Besides, The last element partition_rootfsl1size is 0, it means that last partition will extend to the end block of emmc device.

For partition naming rule, primary partition must be partition_rootfs***n*** except partition 0. The logical partition muse be partition_rootfsl***n*** from start 1. rootfs, rootfs1 and rootfs2, mbr1, rootfsl1, mbr2, rootfsl2, ... all must be inascending order and continuous in the bottom of emmc. rootfs, rootfs1 and rootfs2 must be existing; mbr1, rootfsl1 are optional.

In other words, if you needs to add more logical partition, it can be the following sample.

```
&emmc {

        partition_mbr {       label = "mbr";                 reg = <0x0 0x00000000 0x0 0x00000200>; };

        partition_fdt {       label = "fdt";                 reg = <0x0 0x00040000 0x0 0x00040000>; };

        partition_fdt.restore { label = "fdt.restore";          reg = <0x0 0x00080000 0x0 0x00040000>; };

        partition_uboot {      label = "uboot";              reg = <0x0 0x000C0000 0x0 0x00200000>; };

        partition_uenv {       label = "uenv";               reg = <0x0 0x002C0000 0x0 0x00040000>; };

        partition_kernel {     label = "kernel";              reg = <0x0 0x00300000 0x0 0x00400000>; };
        partition_rootfs {     label = "rootfs";  active="true"; reg = <0x0 0x00700000 0x0 0x04000000>; };

        partition_rootfs1 {    label = "rootfs1";             reg = <0x0 0x04700000 0x0 0x05000000>; };

        partition_rootfs2 {    label = "rootfs2";             reg = <0x0 0x09700000 0x0 0x0CC00000>; };

        partition_mbr1 {       label = "mbr1";                reg = <0x0 0x16300000 0x0 0x00400000>; };

        partition_rootfsl1{    label = "rootfsl1";            reg = <0x0 0x16700000 0x0 0x05000000>; };

        partition_mbr2 {       label = "mbr2";                reg = <0x0 0x1B700000 0x0 0x00400000>; };

        partition_rootfsl2{    label = "rootfsl2";            reg = <0x0 0x1BB00000 0x0 0x00000000>; };

};
```

2.5 nvt-na51000-nvtpack.dtsi

Accroding to the sample of nvt-na51000-storage-partition.dtsi, the corresponding content shown as following,

```
&emmc {

        /**

         * partition_name is $1 as in partition_$1 is referred

         * to nvt-na51000-storage-partition. dtsi

         */

        nvtpack {

                ver = "NVTPACK_FW_INI_16072017"; /* Fixed */

                method = <1>; /* Fixed */

                index {

                        id0 { partition_name = "mbr";       source_file = "MBR_0.bin"; }; /* Fixed */

                        id1 { partition_name = "fdt";       source_file = "nvt-na51000-smp-evb.dtb"; }; /* Fixed */

                        id2 { partition_name = "fdt.restore"; source_file = ""; }; /* Fixed */

                        id3 { partition_name = "uboot";     source_file = "u-boot.bin"; };

                        id4 { partition_name = "uenv";      source_file = ""; };

                        id5 { partition_name = "kernel";    source_file = "uImage.bin"; };

                        id6 { partition_name = "rootfs";    source_file = "rootfs.ext4.bin"; };

                        id7 { partition_name = "rootfs1";   source_file = "rootfs_1.cache.vfat.bin"; };

                        id8 { partition_name = "rootfs2";   source_file = ""; };

                        id9 { partition_name = "mbr1";      source_file = "MBR_1.bin"; };

                        id10 { partition_name = "rootfsl1";  source_file = ""; };

                };
        };

};
```

The generated rootfs named rootfs.ext4.bin means use ext4 file system. The generated rootfs1 named rootfs_1.cache.vfat.bin means use pre-formated vfat file system.

3 Update and recovery

3.1 Update from SD card

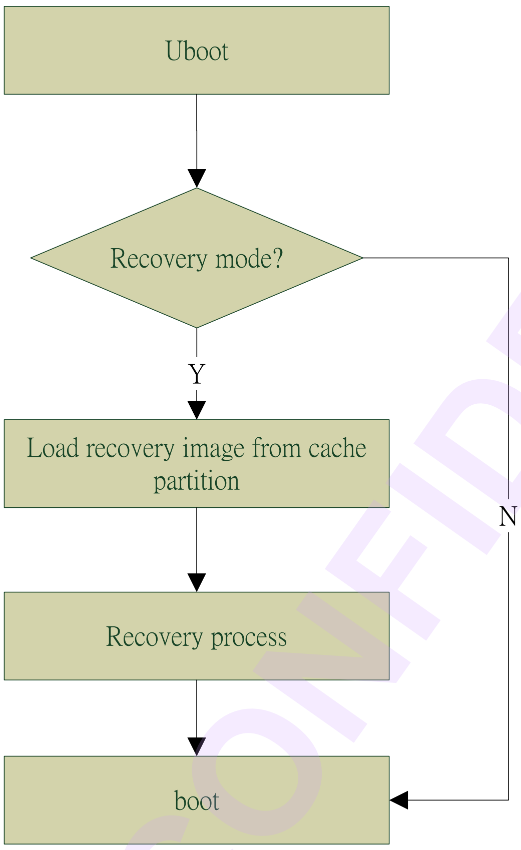
Loader and all-in-one image update procedures are the same with previous nand boot. You just need to put them(LD{SOC}A.bin and FW{SOC}A.bin) to SD card, the update process can start automatically.

3.2 OTA Update

TBD

3.3 Recovery

SDK provide a FAT file system cache partition can do recovery procedure, the details will be listed as below. The recovery image must have contained iTron, Linux and rootfs or DSP…etc images. Of course, if your uboot is broken, you will need to recovery overall system from loader. Loader related handling will involve some customization, you can modify it based on your requirement.



Using nvt pack tool to generate FW{SOC}R.bin firstly and perform below procedure, we have two methods to generate recovery partition image.

1. Put prebuild all-in-one image firstly.

   Copy your recovery image FW{SOC}R.bin into “root-fs/tools/rootfs_recovery”, this image should not include rootfs_1.cache.vfat.bin, otherwise cache partition data will be rewrited if recovery mode is triggered.

   ```
   $ make rootfs
   ```

2. Put prebuild all-in-one image from NT96680 EVB as below.

   ```
   root@NVTEVM:~$ mount /dev/mmcblk2p3 /mnt/recovery
   root@NVTEVM:~$ cp /mnt/sd/FW{SOC}R.bin /mnt/recovery/
   root@NVTEVM:~$ sync
   ```

4 Limitation

We have several limitations currently as below:

1. All-in-one image size should be smaller than itron dram partition.
2. Update from USB or Ethernet are not fully verified, please contact us if you want to use this feature.
3. EMMC transmission speed can’t support ultra high speed in Uboot.
4. Not support update loader in all-in-one image.