NT96680 SDK UBoot Programmig Guide

Table of Content

1 Features

- To be responsible for multi-core boot flow
- Boot from Nand/Nor/SD/Ethernet
- Update all-in-one image
  - We support FW96680T.bin/FW96680A.bin, both images are produced by NvtPack tool. Amoung them, the FW96680T.bin can boot system without Nand or Nor flash, FW96680A.bin is used to update firmware to flash.
  - Provide basic image checksum

2 Environment Setup and Compiling

2.1 Environment setup

Please refer to the document of *Linux Development Environment User Guide.pdf*.

2.2 Compilation

In the root directory of NA51000_BSP, to type “make uboot” to build U-Boot and “make uboot_clean” to clean U-Boot.

Our uboot image can support LZ compression mode image, it will produce two images are u-boot.bin and u-boot.lz.bin. We also implement checksum mechanism to check whether uboot binary is workable or not.

```
$ make uboot
$ make uboot_clean
```

3 System configuration

3.1 Functionality setting

Uboot functionalities enable or disable is controlled through "include/configs/nvt-na51000.h", this file bind to uitron codebase to do features configuration (uitron/Project/DemoKit/ModelConfig_{YOUR_MODEL}.txt). The important setting are introduced as below.

- To select whether support nor or nand flash:

```
#define CONFIG_NVT_SPI_NAND
/*#define CONFIG_NVT_SPI_NOR*/
```

- Memory mapping setting:
  Using “NT96680 Memory Map.xls” to calculate memory mapping address directly, and fill them into ModelConfig_{YOUR_MODEL}.txt as the following.

  ModelConfig_{YOUR_MODEL}.txt:

```
BOARD_DRAM_ADDR =   0x00000000
BOARD_DRAM_SIZE =   0x20000000
BOARD_REV_ADDR =    0x00000000
BOARD_REV_SIZE =    0x00002000
BOARD_IPC_ADDR =    0x00002000
BOARD_IPC_SIZE =    0x001FE000
BOARD_LINUX_ADDR =  0x00200000
BOARD_LINUX_SIZE =  0x01000000
BOARD_UBOOT_ADDR =  0x01200000
BOARD_UBOOT_SIZE =  0x00600000
BOARD_UITRON_ADDR = 0x01800000
BOARD_UITRON_SIZE = 0x19800000
BOARD_UITRON_RESV_SIZE = 0x00000000
BOARD_RAMDISK_ADDR = 0x1B000000
BOARD_RAMDISK_SIZE = 0x03000000
BOARD_DSP2_ADDR =   0x1F000000
BOARD_DSP2_SIZE =   0x00800000
BOARD_DSP1_ADDR =   0x1F800000
BOARD_DSP1_SIZE =   0x00800000
BOARD_LOADER_ADDR = 0x1FF00000
BOARD_LOADER_SIZE = 0x00100000
BOARD_EXTDRAM_ADDR = 0x40000000
BOARD_EXTDRAM_SIZE = 0x20000000
```

include/configs/nvt-na51000.h:

```
#ifdef CONFIG_NT96680
	#include "novatek/na51000_ca53_evb.h"
	#define CONFIG_NVT_FW_NAME                    _BIN_NAME_".bin"                /*
Update firmware name */
	#define CONFIG_NVT_RUNFW_NAME                 _BIN_NAME_T_".bin"              /*
iTron firmware name for sd boot test */
	#define CONFIG_NVT_MODELEXT_NAME              _BIN_NAME_".ext.bin"            /*
Modelext name */
#endif /* CONFIG_NT96680 */
```

- Uboot environment setting:
  In the uboot shell, the environment can be stored in Nand/Nor/RAM is based on your config. The partition table is binding to "uitron/DrvExt/DrvExt_src/ModelExt/EVB/independant/emb_partition_info.c" when you enable Nand or Nor flash, uboot environment offset and size needs to be the same with it.

  emb_partition_info.c:

```
EMB_PARTITION ind_emb_partition_info_data[EMB_PARTITION_INFO_COUNT] __attribute__ ((section
("modelext_data.emb_partition_info"))) = {
	[0] = {EMBTYPE_LOADER,  0,  BLK_UNIT(0x00000000), BLK_UNIT(0x00040000),
BLK_UNIT(0x00000000)},
	[1] = {EMBTYPE_MODELEXT,0,  BLK_UNIT(0x00040000), BLK_UNIT(0x00040000),
BLK_UNIT(0x00000000)},
	[2] = {EMBTYPE_UITRON,  0,  BLK_UNIT(0x00080000), BLK_UNIT(0x00880000),
BLK_UNIT(0x00000000)},
	[3] = {EMBTYPE_UBOOT,   0,  BLK_UNIT(0x00900000), BLK_UNIT(0x00200000),
BLK_UNIT(0x00000000)},
	[4] = {EMBTYPE_UENV,    0,  BLK_UNIT(0x00B00000), BLK_UNIT(0x00040000),
BLK_UNIT(0x00000000)},
	[5] = {EMBTYPE_LINUX,   0,  BLK_UNIT(0x00B40000), BLK_UNIT(0x00400000),
BLK_UNIT(0x00000000)},
	[6] = {EMBTYPE_DSP,     0,  BLK_UNIT(0x00F40000), BLK_UNIT(0x00500000),
BLK_UNIT(0x00000000)},
	[7] = {EMBTYPE_DSP,     1,  BLK_UNIT(0x01440000), BLK_UNIT(0x00800000),
BLK_UNIT(0x00000000)},
	[8] = {EMBTYPE_PSTORE,  0,  BLK_UNIT(0x01C40000), BLK_UNIT(0x00400000),
BLK_UNIT(0x00000000)},
	[9] = {EMBTYPE_ROOTFS,  0,  BLK_UNIT(0x02040000), BLK_UNIT(0x057C0000),
BLK_UNIT(0x00000000)},
	[10]= {EMBTYPE_FAT,     0,  BLK_UNIT(0x07800000), BLK_UNIT(0x00800000),
BLK_UNIT(0x00000000)},
};
```

include/configs/nvt-na51000.h:

```
#if defined(_NVT_UBOOT_ENV_IN_STORG_SUPPORT_NAND_)
#define CONFIG_CMD_SAVEENV
#define CONFIG_ENV_IS_IN_NAND
#define CONFIG_ENV_OFFSET                       0x00B00000              /*
Defined by iTron emb_partition_info.c */
#define CONFIG_ENV_SIZE                         (128 << 10)             /* Unit:
Block size: 128 KiB */
#define CONFIG_ENV_RANGE                        2 * CONFIG_ENV_SIZE      /*
Defined by iTron emb_partition_info.c */
#elif defined(_NVT_UBOOT_ENV_IN_STORG_SUPPORT_NOR_)
#define CONFIG_CMD_SAVEENV
#define CONFIG_ENV_IS_IN_SPI_FLASH
#define CONFIG_ENV_OFFSET                       0x003b0000              /* It
must be aligned to an erase secrote boundary */
#define CONFIG_ENV_SIZE                         0x00040000              /* Sync
to emb_partition_info.c */
#define CONFIG_ENV_SECT_SIZE                    (64 << 10)              /*
Define the SPI flash's sector size */
#elif defined(_NVT_UBOOT_ENV_IN_STORG_SUPPORT_MMC_)
#define CONFIG_CMD_SAVEENV
#define CONFIG_ENV_IS_IN_MMC
#define CONFIG_ENV_OFFSET                       (6 * 64 * 1024)
#define CONFIG_ENV_SIZE                         0x00040000
#else
#define CONFIG_ENV_IS_NOWHERE
#define CONFIG_ENV_SIZE                         (8 << 10)
#endif
```

- NVT boot and update flow setting:
  The default setting will use NVT all-in-one image update mechanism, you also can customize your boot flow after undefine “CONFIG_NVT_LINUX_AUTOLOAD”.

```
/* NVT boot related setting */
#define CONFIG_NVT_LINUX_AUTOLOAD                                   /* Support for uboot
autoboot (loading and boot FW96680T.bin[for RAMDISK] or FW96680A.bin[for mtd device]) */
#ifdef CONFIG_NVT_LINUX_AUTOLOAD
      #define CONFIG_NVT_LINUX_AUTODETECT                           /* Support for detect
FW96680A.bin/FW96680T.bin automatically. (Only working on mtd device boot method) */
      #define CONFIG_NVT_BIN_CHKSUM_SUPPORT                         /* This option will
check rootfs/uboot checksum info. during update image flow */
      #if defined(_NVT_ROOTFS_TYPE_RAMDISK_)
            /* the ramdisk dram base/size will be defined in itron modelext info. */
            #define CONFIG_NVT_LINUX_RAMDISK_BOOT                   /* Loading ramdisk
image rootfs.bin from SD card */
            #define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON "root=/dev/ram0
rootfstype=ramfs rdinit=/linuxrc "
      #elif defined(_NVT_ROOTFS_TYPE_NAND_UBI_)                     /* UBIFS
rootfs boot */
            #define CONFIG_NVT_LINUX_SPINAND_BOOT
            #define CONFIG_NVT_UBIFS_SUPPORT
            #define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON "root=ubi0:rootfs
rootfstype=ubifs rw ubi.fm_autoconvert=1 init=/linuxrc "
            #define CONFIG_CMD_UBI                                  /* UBI-formated MTD
partition support */
            #define CONFIG_CMD_UBIFS                                /* Read-only UBI volume
operations */
      #elif defined(_NVT_ROOTFS_TYPE_NAND_SQUASH_)                  /* SquashFs rootfs
boot */
            #define CONFIG_NVT_LINUX_SPINAND_BOOT
            #define CONFIG_NVT_SQUASH_SUPPORT
            #define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON
"rootfstype=squashfs ro "
      #elif defined(_NVT_ROOTFS_TYPE_NAND_JFFS2_)                   /* JFFS2 rootfs boot
*/
```

```
		#define CONFIG_NVT_LINUX_SPINAND_BOOT                       /* Boot from spinand
or spinor (Support FW96680A.bin update all-in-one) */
		#define CONFIG_NVT_JFFS2_SUPPORT
		#define CONFIG_CMD_NAND_TRIMFFS
		#define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON "rootfstype=jffs2 rw
rootwait "
	#elif defined(_NVT_ROOTFS_TYPE_NOR_SQUASH_)                          /* Squashfs rootfs
boot */
		#define CONFIG_NVT_LINUX_SPINOR_BOOT                        /* Boot from spinand
or spinor (Support FW96680A.bin update all-in-one) */
		#define CONFIG_NVT_SQUASH_SUPPORT
		#define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON
"rootfstype=squashfs ro "
	#elif defined(_NVT_ROOTFS_TYPE_NOR_JFFS2_)                           /* JFFS2 rootfs boot
*/
		#define CONFIG_NVT_LINUX_SPINOR_BOOT                        /* Boot from spinand
or spinor (Support FW96680A.bin update all-in-one) */
		#define CONFIG_NVT_JFFS2_SUPPORT
		#define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON "rootfstype=jffs2 rw
rootwait "
	#else
		#define CONFIG_NVT_LINUX_SD_BOOT                           /* To handle RAW SD
boot (e.g. itron.bin, uImage.bin, uboot.bin...) itron.bin u-boot.bin dsp.bin dsp2.bin must be
not compressed.*/
		#define CONFIG_BOOTARGS             CONFIG_BOOTARGS_COMMON
"root=/dev/mmcblk0p2 noinitrd rootfstype=ext3 init=/linuxrc "
	#endif /* _NVT_ROOTFS_TYPE_ */
#endif /* CONFIG_NVT_LINUX_AUTOLOAD */
```

- DSP boot:

Control the DSP boot from itron ModelConfig_{YOUR_MODEL}.txt if your SOC can support DSP.

ModelConfig_{YOUR_MODEL}.txt:

```
# [DSP1_TYPE]
# DSP1_NONE
# DSP1_FREERTOS
DSP1_TYPE = DSP1_FREERTOS


# [DSP2_TYPE]
# DSP2_NONE
# DSP2_FREERTOS
DSP2_TYPE = DSP2_FREERTOS
```

include/configs/nvt-na51000.h:

```
#if defined(_DSP1_FREERTOS_)
#define CONFIG_DSP1_FREERTOS
#endif


#if defined(_DSP2_FREERTOS_)
#define CONFIG_DSP2_FREERTOS
#endif
```

- Boot command:

The “nvt_boot” is an instruction to handle nvt boot flow. System boot will call “nvt_boot” to finish some procedures as below:

1. Detect all-in-one image is FW96680T.bin or FW96680A.bin
2. Launch itron firmware
3. Launch Linux/DSP1/DSP2 firmwares
4. Provide necessary cmdline to do partition table setting.

You can customize “nvt_boot” procedure at “u-boot/board/novatek/nvt-na51000/na51000_utils.c”

```
#define CONFIG_BOOTCOMMAND                    "nvt_boot"
```

3.2 User modification

The uboot features can be divided into two parts, one is binding to itron codebase, the other is normal uboot setting. If your option is binding to itron codebase, please follow above chapter suggestion to configure it, because some features will affect multiple codebases (e.g. rootfs/uboot/application).

3.2.1 Disable nvt checksum

Top Makefile:

Remove this line which is contained “encrypt_bin” string to remove nvt checksum change.

```
uboot: checkenv
	@echo "##### Build u-boot loader #####"
	@$(call log_remove)
	@$(call log_stdout, make -C $(UBOOT_DIR) O="" distclean)
	@$(call log_stdout, make -C $(UBOOT_DIR) O="" nvt-na51000_config)
	@$(call log_stdout, make -C $(UBOOT_DIR) O="")
	@$(call log_stdout, make -C $(UBOOT_DIR) O="" env)
	@$(BUILD_DIR)/nvt-tools/encrypt_bin SUM $(UBOOT_DIR)/u-boot.bin 0x350 ub51000
	@$(BUILD_DIR)/nvt-tools/bfc c lz $(UBOOT_DIR)/u-boot.bin $(UBOOT_DIR)/u-boot.lz.bin 0 0
	@cp -af $(UBOOT_DIR)/u-boot.bin $(UBOOT_DIR)/u-boot.lz.bin $(OUTPUT_DIR)/
```

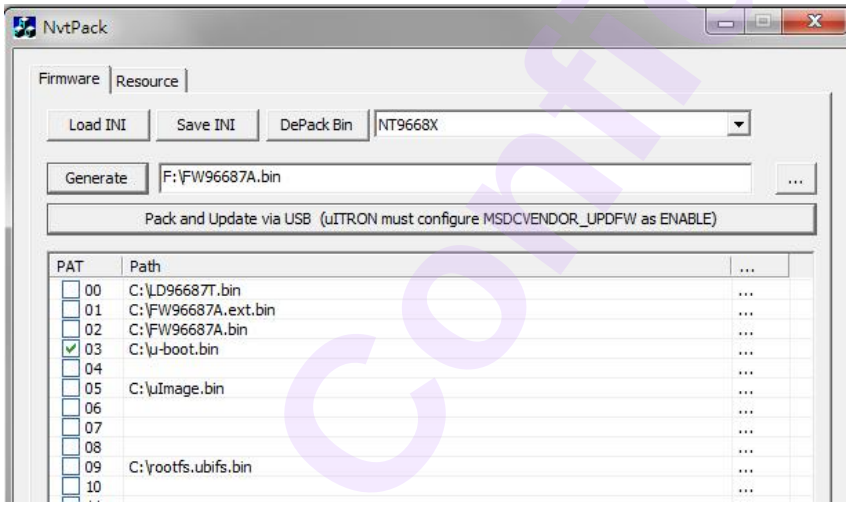
Loader:

You should modify loader source code to disable checksum checking. Please refer to Loader application note document.

4 Update flow

4.1 Update uboot from SD (all-in-one pack)

According to *Linux_Development_Environment_User_Guide.pdf* introduction, the NvtPack tool can update uboot image as below.



4.2 Update from Ethernet

TBD

4.2.1 TBD

TBD

4.2.2 TBD

TBD

4.3 Updated by uboot shell

We support basic sd/memory/nand/nor r/w instruction, you could use the basic command to update images.

4.3.1 Useful commands

$ mtdparts
List partition tables (Sync with iTron partition table setting)

$ fatload mmc 0:1 0x2000 uImage.bin
Loading uImage.bin from SD card to the memory address 0x2000

$ nand erase offset size
To do nand flash erase

$ nand write mem_addr mtd_offset mtd_size
Write memory address mem_addr data to nand flash mtd_offset with mtd_size bytes.

$ sf write mem_addr mtd_offset mtd_size
Write memory address mem_addr data to nor flash mtd_offset with mtd_size bytes.

Other uboot commands can refer to https://www.denx.de/wiki/DULG/Manual

4.3.2 Customization

Our update flow used uboot commands to implement, you also can do the customization by yourself.

5 Add new Flash support

As new NAND flash would be introduced, a specified NAND ID table is performed to maintain the parameters corresponding NAND Flashes.

5.1 SPI Nand

Add new flash ID to this file “u-boot/drivers/mtd/nand/nand_ids.c”.

```
const struct nand_flash_dev nvt_nand_ids[] = {
	// name                      id/pagesize(unit byte)/chip size(unit MB)
	{"NAND 256MiB 3,3V 8-bit",   0xDA, 2048, 256, 0x20000,      NAND_NO_SUBPAGE_WRITE},
	{"SPI-NAND 1GiB 3,3V 8-bit", 0xD1, 2048, 128, 0x20000,      NAND_NO_SUBPAGE_WRITE},
	{"SPI-NAND 128MiB 3V",       0xAA, 2048, 128, 0x20000,      NAND_NO_SUBPAGE_WRITE},
	{"SPI-NAND 128MiB 3V",       0x12, 2048, 128, 0x20000,      NAND_NO_SUBPAGE_WRITE},
};
```

5.2 SPI Nor

TBD

6 NVT defined command

We provide several instructions to support NVT boot related procedure. All the commands are defined at “u-boot/board/novatek/nvt-na51000/na51000_utils.c”.

6.1 nvt_boot

This is the normal boot command, to handle the boot flow which includes itron(CPU1)/Linux(CPU2)/FreeRTOS1(DSP1)/FreeRTOS2(DSP2).

6.2 nvt_booti

Boot CPU1 iTron OS only.

6.3 nvt_bootd

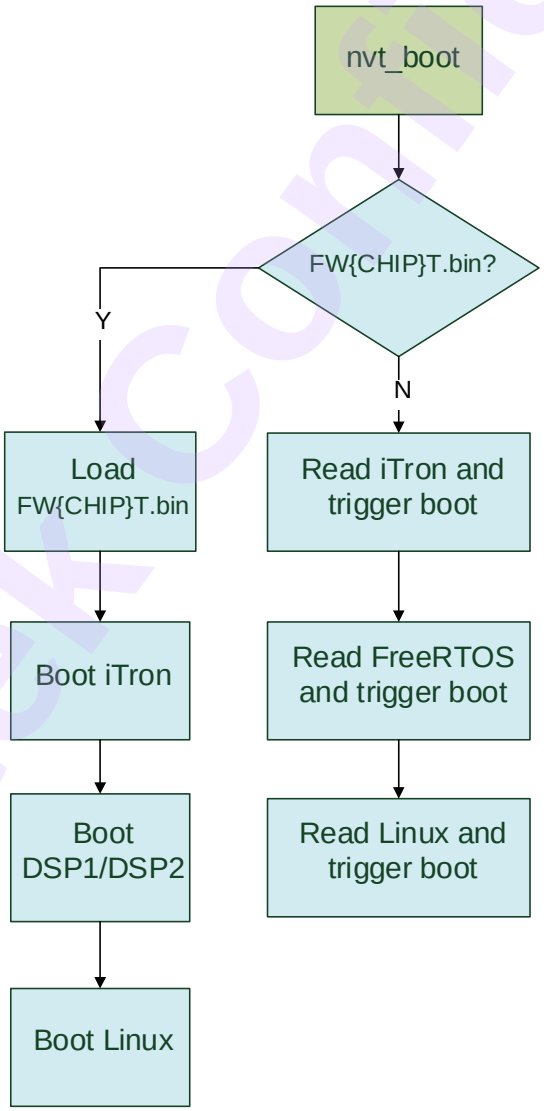
Boot DSP1/DSP2 FreeRTOS OS only.

7 Boot flow

Uboot will handle system boot up flow, the related details will be listed as below section.

7.1 Nand/Nor/SD boot

The “nvt_boot” flowchart is as below. FW{CHIP}T.bin can service booting from SD card, and the other handles booting from flash.



7.2 Ethernet boot

If you don't want to burn the Linux Kernel every time in the development stage, you could perform the following steps to download uImage into DRAM and start it.

1. The CONFIG_BOOTDELAY should be enabled in advance.

   - **Example**

```
#define    CONFIG_BOOTDELAY          5
```

2. Press any key when the auto boot is counting down in order to enter U-Boot command line.

3. Initiate ethernet driver
   uboot(NT96687) $ eth_init

4. Ping command to check network status
   uboot(NT96687) $ ping 192.168.1.1

```
uboot(NT96687) $ eth_init
eth_na51000
uboot(NT96687) $ ping 192.168.1.1
eqos_start(eth_dev=014a3370):
eth_na51000 Waiting for PHY auto negotiation to complete.... done
Using eth_na51000 device
host 192.168.1.1 is alive
```

5. Tftp download Linux uImage and modelext images
   Please follow the document NT96680 Memory Map.xls to get related address.

   e.g.
   uboot(NT96687) $ setenv bootargs *"YOUR_EVB_BOOTARGS"*
   uboot(NT96687) $ setenv serverip *server ip* (set TFTP server IP address)
   uboot(NT96687) $ setenv ipaddr evb ip (set EVB IP address)
   uboot(NT96687) $ tftp 0x2000 FW96687A.ext.bin (Loading modelext.bin to 0x2000)
   uboot(NT96687) $ tftp 0x200000 uImage.bin (Loading Linux kernel image)
   uboot(NT96687) $ tftp 0x01800000 FW96687A.bin (Loading itron.bin without compression)

```
uboot(NT96687) $ nvt_booti_tftp          (Boot itron without loading)
uboot(NT96687) $ bootm 0x01800000        (Boot Linux kernel)
```