NT96680 SDK uITRON Programmig Guide

Table of Content

1 Introduction

1.1 SDK Overview

This Novatek NT96680 SDK provides a platform solution to shorten the development cycle of image-processing products. NT96680 has four cores, 2 ARM-CPUs and 2 DSPs. Core-1 always runs uITRON OS (codename is GS). Core-2 always runs on Linux. Core-3 and Core-4 run the freeRTOS OS. And this document will foucs on the uITRON programming.

1.2 GS (uITRON) Module Layers, Groups and Its Relationship

On the Core-1, conceptually, all modules of GS are divided into several layers with internal and external categories.

All modules are divided into the following 5 layers: *common, driver (Drv), library (Lib), application (App) and project*.

- The top layer “Project” implements the overall function of the entire image-processing system.
- The second layer “App” contains software flow integration and task control for different purpose applications.
- The third layer “Lib” contains basic function modules of the platform.
- The bottom layer “Drv” contains drivers that connect and communicate with hardware.
- In addition, a special layer “Common” contains common functions, include OS kernel functions, and it could be used by the all layers.

The three layers: driver, library and application, are also divided into 2 categories: *internal and external*.

- A module in an internal category means that it is built by Novatek. And, its source code will not disclose to users. SDK contains header files (.h) and binary files.
- The external category represents this module could be customized and even replaced.

SDK will contain most of header files and source code in the module.

- The “common” layer is categorized to an internal layer, and the “Project” layer is of course an external layer. And each of other three layers “App”, “Lib” and “Drv” is divided into an internal category and a customizable external category, respectively.

According to the principle, SDK has these groups: *Project, App, AppExt, Lib, LibExt, Drv, DrvExt, and Common*.

The GS software layers and groups are illustrated as Figure 1-5.

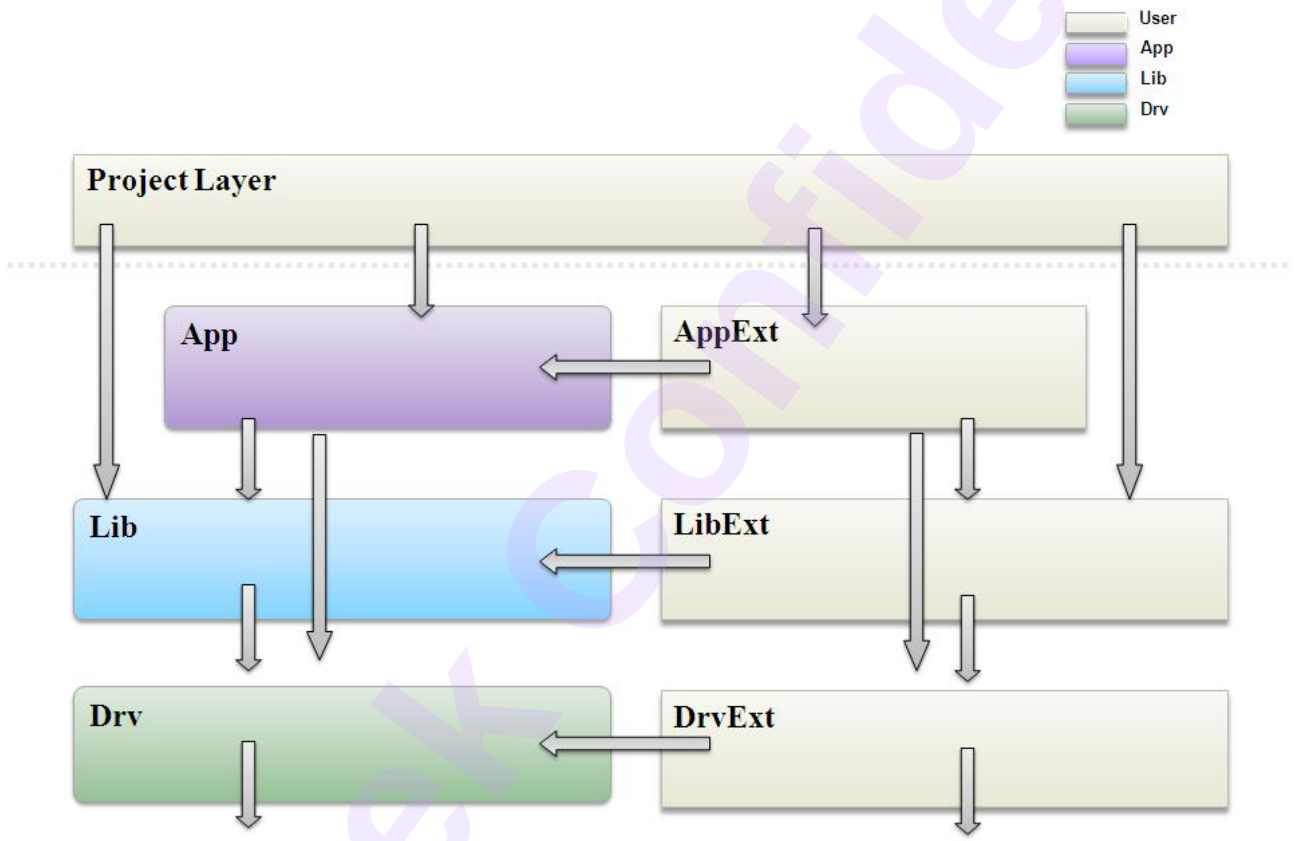

Figure 1-5 GS software layers and groups

Open interface rules of module of these groups. (Open: it represents that allows the calls directly.)

- The lower layer interface is open to upper layer, and not vice versa. That means the code of upper layer can call the function of lower layer, but the inverse direction is not allowed.
- At the same layer, interfaces of internal modules are open to external modules, and not vice versa. That means that code of external modules can call the function of

internal modules, but the inverse direction is not allowed

2 uITRON Basic Programming

2.1 How to create a new project?

The project layer is placed in this folder:
\Project

User can create its own project as 2nd sub-folder, ex: naming to MYPROJ
\Project\[MYPROJ]

After create this sub-folder, please copy these files from original SDK folder:
\Project\EVB\
\Project\[MYPROJ]\Debug_D.bat
\Project\[MYPROJ]\Debug_R.bat
\Project\[MYPROJ]\IceMode.ttl
\Project\[MYPROJ]\IceMode_D.bat
\Project\[MYPROJ]\IceMode_R.bat
\Project\[MYPROJ]\init.gdb
\Project\[MYPROJ]\init_IceMode.gdb
\Project\[MYPROJ]\LDS_EVB.lds
\Project\[MYPROJ]\LoadCode.ttl
\Project\[MYPROJ]\MakeConfig.txt
\Project\[MYPROJ]\Makefile
\Project\[MYPROJ]\ModelConfig.txt
\Project\[MYPROJ]\ModelConfig_EVB.txt
\Project\[MYPROJ]\Run_D.bat
\Project\[MYPROJ]\Run_R.bat

And, then modify the folder setting of the active project in ProjectConfig.txt file:
\ProjectConfig.txt

```
#-----------------------------------------------------------------
# Set project here
```

```
#----------------------------------------------------------------------
#ACTIVE_PROJECT = DemoKit
#ACTIVE_PROJECT = TestKit
ACTIVE_PROJECT = [MYPROJ]
```

Finally, modify the setting of the project name in a Makefile file:

\Project\[MYPROJ]\Makefile

```
#----------------------------------------------------------------------
# set the project name here
#----------------------------------------------------------------------
PRJ_NAME = [MYPROJ]
```

2.2 How to add a c file into makefile?

A new c file is inserted into the project, and place into the *SrcCode* folder or its sub-folders:

\Project\[MYPROJ]\SrcCode\[MYSOURCE.c]

or

\Project\[MYPROJ]\SrcCode\[SubFolder]\[MYSOURCE.c]

And, add this file into the source code list of Makefile file:

\Project\[MYPROJ]\Makefile

```
SRC = \
    SrcCode/Kernel/_main.c \
    SrcCode/Kernel/BinInfo.c \
    SrcCode/Kernel/SysCfg.c \
    SrcCode/System/main.c \
    SrcCode/[MYSOURCE.c]
```

2.3 How to make an executable BIN file?

At the directory name of the active project right click and select the item of "NVT Make" context menu to enter to NVT command line window.

You could enter command in this command window: Such as
\Project\[MYPROJ]\make release
It will compile and link into a release version F/W and generate these output files as follows:
\Project\[MYPROJ]\[MYPROJ]_Data\Release\[MYPROJ].axf
\Project\[MYPROJ]\[MYPROJ]_Data\Release\[MYPROJ].sym
\Project\[MYPROJ]\[MYPROJ]_Data\Release\[MYPROJ].txt
\Project\[MYPROJ]\[MYPROJ]_Data\Release\FW96680A.bin

You could enter command as follows:
\Project\[MYPROJ]\make debug
It will compile and link into a debug version F/W and generate these output files:
\Project\[MYPROJ]\[MYPROJ]_Data\Debug\[MYPROJ]_D.axf
\Project\[MYPROJ]\[MYPROJ]_Data\Debug\[MYPROJ]_D.sym
\Project\[MYPROJ]\[MYPROJ]_Data\Debug\[MYPROJ]_D.t
\Project\[MYPROJ]\[MYPROJ]_Data\Debug\FW96680A.bin

If any compiling/linking time errors or warnings occurred, it will be collected into this log file of **\Project\[MYPROJ]\log_Prj.txt**.

If you want to clean the last build result, and enter command
\Project\[MYPROJ]\make clean
That will clean all temporal object files and the bin file is deleted.

2.4 How to create a new model?

The project layer of the GS code is adapted rapidly to different kind of H/W design. The default model configure is Novatek “Evaluate Board”. Simply, it calls “EVB”:
\Project\[MYPROJ]\LDS_EVB.lds

\Project\[MYPROJ]\ModelConfig_EVB.txt

User could copy and add its own model to configure these files as follows:

\Project\[MYPROJ]\LDS_[MYMODEL].lds

\Project\[MYPROJ]\ModelConfig_[MYMODEL].txt

And, then modify the current model setting in ModelConfig.txt file:

\Project\[MYPROJ]\ModelConfig.txt

```
#----------------------------------------------------------------------
# Set model here
#----------------------------------------------------------------------
#MODEL = EVB
MODEL = [MYMODEL]
```

2.5 How to select the related device of a model and options?

The “ModelExt” code is in “DrvExt” groups, not only related H/W control, but also different peripheral devices. To select related device and options of the current model, you need to edit this ModelConfig_[MYMODEL].txt file:

\Project\[MYPROJ]\ModelConfig_[MYMODEL].txt

```
BIN_NAME = FW96$(CHIP)A
SCATTER = LDS_[MYMODEL].lds
#======= DrvExt =========
MODELEXT = MODELEXT_DEMO
LCD = Disp_IF8B_LCD1_AUCN01
LCD2 = OFF
TV = ON
HDMI = ON
SENSOR1 = CMOS_IMX317CQC
AUDIOEXT = AudCodecExt_WM8978
```

For example:

Change MODELEXT option can select different “ModelEx” driver of current H/W:

\ARC\Drv\Release\MODELEXT_[MYMODEL].a

To change the LCD, LCD2, TV and HDMI option could select the different display device:
\ARC\Drv\Release\Disp_PANELCOMMON.a
\ARC\Drv\Release\Disp_[LCD].a
\ARC\Drv\Release\Disp_TV.a
\ARC\Drv\Release\Disp_HDMI.a

To change AUDIOEXT option can select different audio device:
\ARC\Drv\Release\Audio_Common.a
\ARC\Drv\Release\Audio_[AUDIOEXT].a

To change SENSOR option can select different sensor device:
\ARC\Drv\Release\[SENSOR].a

2.6 How to use the debugging tool?

It is a free and open source debugging tool (gdb). It can be started with three operation modes: “load and break”, “load and run” and “immediate break”.

Start gdb with “load and break” mode:
\Project\[MYPROJ]\Debug_D.bat (for debug version FW bin)
\Project\[MYPROJ]\Debug_R.bat (for release version FW bin)

Start gdb with “load and run” mode:
\Project\[MYPROJ]\Run_D.bat (for debug version FW bin)
\Project\[MYPROJ]\Run_R.bat (for release version FW bin)

Start gdb with “immediate break” mode:
\Project\[MYPROJ]\IceMode_D.bat (for debug version FW bin)
\Project\[MYPROJ]\IceMode_R.bat (for release version FW bin)

For understanding the user interface, functions and debugging skills of gdb, please try to search related document in internet.

3 Setup ModelExt

Some global information located in uitron/DrvExt/DrvExt_src/ModelExt/{MODEL_NAME} /independent. FW*.ext.bin is made from those c files. Those information are divided into eight parts listed below:

1. modelext_info.c: DO NOT EDIT IT. It reversed for a tag to indicate modelext.
2. bin_info.c: only "version (8)" and "releasedate (8)" for vendor to modify. Others are reversed to exchange data between loader, uitron, uboot.
3. pinmux_cfg.c: pinmux table to configure NT96680's pinmux.
4. intdir_cfg.c: interrupt direction table to decide where engine's interrupt send to.
5. emb_partition_info.c: flash partition table indicate how to divide the used flash.
6. gpio_info.c: set gpio's direction (input / ouput) or initial state (pull up / pull down)
7. dram_partition_info.c: DRAM partition table, indicate where loader, uitron, u-boot, linux located.
8. model_cfg.c: DO NOT EDIT IT. It reversed to store MODELCONFIG_CFLAGS for debug.

Configure DRAM and flash are more complicated, that describe as following section,

3.1 Setup DRAM partition

Don't edit dram_partition_info.c directly. Configure ModelConfig_$MODEL.txt instead to layout the DRAM partition. For example, like below,

```
# [BOARD DRAM PARTITION]
# 1. DO NOT EDIT the partitions of DRAM, REV, IPC, LOADER.
# 2. u-boot must be in the bottom of linux memory.
# 3. rebuild uITRON, u-boot, linux-kernel after changing partition.
# 4. iTron will reserve a region to the Linux system memory usage. (BOARD_UITRON_RESV_SIZE)
# 5. linux automatically use the memory from the bottom of uitron memory to the dram end if following
condition:
#        5.1 there is no dsp
#        5.2 there is a gap between the bottom of uitron memory and dsp starting address.
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 = 0x03000000
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 = 0xF07F0000
BOARD_LOADER_SIZE = 0x0000A000
BOARD_EXTDRAM_ADDR = 0x40000000
BOARD_EXTDRAM_SIZE = 0x20000000
```

Note:

1. uboot memory space will reuse by Linux after uboot finished.
2. loader may need modifying after uitron space is adjusted.
3. Normally, for increate linux memory space, just adjust BOARD_UITRON_RESV_SIZE.

3.2 Setup embedded flash partition

Partition table is defined at emb_partition_info.c. There can see the sample. The valid

partition number is from 0 to 15. User can create his own partitions by using EMBTYPE_USER0~ EMBTYPE_USER5, and modify u-boot to handle ones.
If many partitions with the same type are require, use OrderIdx to indicate them (e.g. DSP1 and DSP2).

EMBTYPE_DSP, EMBTYPE_PSTORE, EMBTYPE_FAT can be removed if not used.
NvtPack's partition order also refer to emb_partition_info.c.

3.2.1 Partition Structure

```
typedef struct _EMB_PARTITION{
   unsigned short  EmbType;        /* EMBTYPE_ */
   unsigned short  OrderIdx;       /* Order index of the same EmbType based on '0' */
   unsigned int    PartitionOffset; /* Phyical offset of partition */
   unsigned int    PartitionSize;   /* Size of this partition */
   unsigned int    ReversedSize;    /* Reserved size for bad block */
}EMB_PARTITION, *PEMB_PARTITION;
```

- *EmbType*: partition type.
- *OrderIdx*: a certain type has more the same partitions. For example, the 2nd PStore has to set it to 1, the 3rd PStore has to set it to 2, and so on.
- *PartitionOffset*: Physical offset in partition.
- *PartitionSize*: Partition size.
- *ReversedSize*: This field is only for FAT that needs the space to exchange bad block. The size will be reversed from your assigned PartitionSize.
- *Offset and Size* is stored by block unit in modext bin file.

3.2.2 Partition Adjustment

There some rules on partition adjustment.
1. Partition[0] is only for load, Partition[1] is only for modelext.
2. Each partition region cannot be overlapped each other.
3. Partition offset most be ascending order.
4. After adjusting partition, better to rebuild whole BSP.
5. Partition size should be block size alignment.
6. For calculating each partition size, the bad block should be considered in it. The more

partition size you set, more bad blocks can be accepted.

4 System Programming

This section describes the general development features: Operating System (OS), Debug message & Performance measure (Debug).

4.1 OS

The OS library provides features to configure code region, tasks, memory pools, semaphores and flags. It also provides some utilities for dumping kernel information. More detailed description will be introduced in ***NT96680 AN - OS.pdf*** application note.

4.2 Debug

The Debug library provides three major functions: debug message shown on UART screen, debug -command typed via UART, and performance measure functions.
Please refer to ***NT96680 AN - Debug.pdf*** application note for details.

5 Application Programming

5.1 Photo

About Preview and Capture: Single, Burst, Multi-sensor, HDR… etc. Please refer to the following application note.

- ***NT96680 AN Application – Capture.pdf***
- ***NT96680 AN Application – ImageApp_Photo.pdf***

5.2 Movie

About Movie Preview and Record: Normal, High-Speed, Time-lapse, Cyclic, Power-Off-Protect… etc. Please refer to ***NT96680 AN Application – ImageApp_Movie.pdf*** application note.

About Movie Playback: Normal/High-Speed Forward & Backward. Please refer to ***NT96680 AN Application – ImageApp MoviePlay.pdf*** application note.

5.3 Playback

About Playback mode: Single View, Browser View, Image View, Image Edit ...etc. Please refer to the following application note.

- ***NT96680 AN Application –Playback.pdf***
-

5.4 UI Framework

Please refer to the following application note.

- ***NT96680 AN Application – UIControl.pdf***

6 Library Programming

6.1 Storage

- FileSystem (FAT32, exFAT)
  - ☐ Please refer to ***NT96680 AN Library – FileSystem.pdf*** application note.
- PStore
  - ☐ Please refer to ***NT96680 AN Library – PStore.pdf*** application note.
- FWStore
  - ☐ Please refer to ***NT96680 AN Library – FwSrv.pdf*** application note.

6.2 Image Pipeline

About IPL Control: Mode, IQ Adjust, Scaling, RSC, HDR… etc. Please refer to the following application note.

- ***NT96680 AN Library - IPLCmd.pd***

6.3 FileList Lib

- DCF
  - ☐ Please refer to ***NT96680 AN Library – DCF.pdf*** application note.
- FileDB
  - ☐ Please refer to ***NT96680 AN Library – FileDB.pdf*** application note.

6.4 Multi-Core communication

Please refer to the following application note.

- ***NT96680 AN Library – NvtIPC.pdf***

7 Driver Programming

Please refer to the following application note.

- ***NT96680_AN Driver – Driver.pdf***

●