OPL1000

ULTRA-LOW POWER 2.4GHZ WI-FI + BLUETOOTH SMART SOC

System Initialization

Brief Introduction

OPULINKS

REVISION HISTORY

Date	Version	Contents Updated
2018-06-02	0.1	• Initial Release
2018-07-07	0.2	• Updated for better understanding
2018-08-03	0.3	• Update for A1 chip

LIST OF FIGURES

TABLE OF CONTENTS

1. 系统进入点程序说明

\APS\driver\CMSIS\Device\nl1000\Source\ARM\startup_ARMCM3.s

代碼 1

```
; Reset Handler

Reset_Handler   PROC
                EXPORT  Reset_Handler             [WEAK]
                IMPORT  Boot_CheckWarmBoot
                IMPORT  SystemInit
                IMPORT  __main
                IMPORT  Boot_MainPatch

                    …

                LDR     R0, =Boot_CheckWarmBoot
                BLX     R0
                CMP     R0, #1
                BEQ     WarmBoot
                                                        Cold Boot
                LDR     R0, =SystemInit
                BLX     R0
                LDR     R0, =__main
                BX      R0
WarmBoot
                LDR     R0, =Boot_MainPatch
                LDR     R0, [R0]                    Warm Boot
                BX      R0
                ENDP
```

startup_ARMCM3.s

一开始系统上电时， 会使用 **Boot_CheckWarmBoot** 检查是否为 warm boot 或 cold boot。warm boot 和 cold boot 的差别在於系統是否處在定時睡眠的模式 (Timer sleep mode)，假如系統目前是處在於定時睡眠的模式，系統就會用 warm boot 來快速的啟動系統。反之，就是用 cold boot 來啟動系統。

假如是 warm boot，就直接进入 Boot_MainPatch，而 Boot_MainPatch 就会开始进入到主程序的 main() 函式。

假如是 cold boot，会执行 SystemInit 和进入 __main 的动作。 SystemInit 会把蕊片内部振荡器的时钟速率，做一个初始化的动作。 __main 注[1]会 linking 到 ARMCC 所提供 C 组件库里面相对应的 main 函式库，接着开始做一连串的启动流程。 上述的实际动作流程，可以参考代碼 1。

注[1] http://infocenter.arm.com/help/topic/com.arm.doc.dui0377g/DUI0377G_mdk_armlink_user_guide.pdf

2. 主程序说明

\APS\project\opl1000\startup\main.c
Main.c

代碼 2

```
1. void Main_AppRun_impl(void) {
2.         osKernelInitialize();
3.         Sys_DriverInit();
4.         osKernelRestart();
5.         Sys_ServiceInit();
6.         Sys_AppInit();
7.         Sys_PostInit();
8.         osKernelStart();
9.         while (1);
10.}
11.
12./* * main: initialize and start the system */
13.int main(void) {
14.    Boot_Sequence();
15.    Main_AppRun();
16.}
```

main() :

进入 main function 之后会执行 Boot Sequence，但由于有分 Cold Boot 和 Warm Boot，所以后续的上电过程当中，就会产生不同的动作分支流程。

随后会执行 Main_AppRun ()，Main_AppRun () 是一個 function pointer, 真正建構的函式在於 Main_AppRun_impl 為主要的主體。 可参考代碼 2，实际动作的流程，分别为图 1 和图 2。OS init 到 OS scheduling 的部份是屬於 Main_AppRun_impl 裡面的調用。 warm boot 相对于 cold boot 省略了若干步骤，在 warm boot 時，OS init 在 cold boot 時，已有初始化過，故在 warm boot 流程中並不會在執行一次，但是在 warm boot 最後一個流程中，會有 OS Rescheduling 的動作。其目的是為了讓 OS 回復到還沒有休眠之前的設定。Service init 和 App init 在 cold boot 已有啟動程序裡面部份的 task，故在 warm boot 時，並不會在重新執行，所以在 warm boot 時，並沒有這二個流程。

图 1: Cold Boot

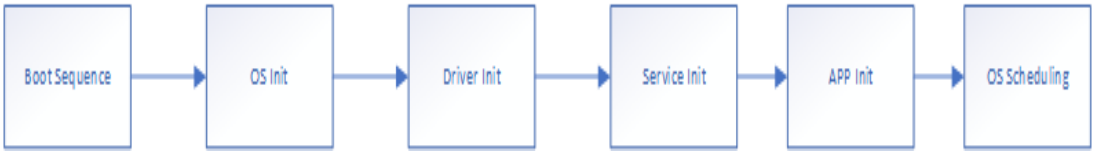

图 2: Warm Boot

Boot Sequence: Boot_Sequence()

- Cold Boot:

  在 Cold Boot 阶段执行 Boot Sequence 时，会开始把所有的 function point 都设定到对应的 function，诸如系统函式初始化，任务控制单元初始化，WiFi API 初始化，BLE 控制单元初始化，LwIP (TCP/IP Stack) 初始化，AT command 初始化，HAL driver API 初始化，OS 初始化。此步骤也包涵了频率的设定，UART 波特率的初始化，SPI 的预设振荡频率与启动的初始化，Flash 也指定到 SPI0 为内部使用的 Flash 等等。

- Warm Boot:

  在 Warm Boot 阶段，目前没有做任何设定。

OS Init: `osKernelInitialize()`

- Cold Boot:

  在 Cold Boot 阶段，memory pool 的初始化。

- Warm Boot:

  在 Warm Boot 阶段，目前没有做任何设定。

Driver Init: `Sys_DriverInit()`

电源设定初始化，设定 Power 的参数，系统频率初始化，GPIO 功能(pin-mux)初始化，可用 Opulinks 所提供的 pinmux 工具来设定 GPIO 的脚位功能。SPI0、SPI1、SPI2 分别撷取系统定义的频率来做初始化，Flash 初始化，FIM (Flash Item Management) 初始化，UART0 / UART1 初始化，PWM (Pulse Width Modulation) 初始化

Service Init: `Sys_ServiceInit()`

AT command 任务单元初始化， WiFi mac 任务单元初始化，LWIP (TCP/IP Stack) 任务单元初始化，控制任务单元初始化，IPC 单元初始化，WPA_Supplicant 的初始化，自动联机的初始化设定，会根据 flash 读取设定来自动联机，都在此步骤完成。

APP Init: `Sys_AppInit()`

在 App Init 這個步驟，客戶可以在這裡新增想要的初始化功能。

Post Init: `Sys_PostInit()`

在 Post Init 这个步骤，会把 Log module (Tracer) 做一个初始化的动作。Tracer 顾名思意追踪信息，用于 debug，实现追踪 OS 中函数事件的框架。

3. 客户的客制化初始化设定

\APS_PATCH\project\nl1000\startup\main_patch.c

客户可以新增 APP Init 的相对初始化设定，从这里开始。使用者可以初始化變數, 也可以產生相對應的 APP task.

\APS_PATCH\project\nl1000\startup\sys_init_patch.c

针对 Driver Init ,Service Init，客户端如有需要新增相对应的初始化动作，可以做相对应的 Patch，如 Sys_DriverInit_patch , Sys_ServiceInit_patch。

4. 實際量測數據

根據本文針對 Cold Boot 和 Warm Boot 的流程說明，用戶可以了解到二者的動作流程之差異，並且透過實際的量測數據，更加可以讓用戶了解到 OPL1000 在 Cold Boot 和 Warm Boot 在不同應用情境之下，所量測到的真正應用數據。用戶在了解理論和實用上有更進一步的感受之後，用戶更可以在不同的場景情況之下，選擇適合用戶本身的使用情境。

4.1. Cold Boot

場景定義: Power on 到與 AP 建立通信拿到 IP 為止。

量測數據: 1821ms (Cold Boot + Auto Connect)

	Timestamp(ms)	Duration(ms)	Comment
Wakeup	0		System reset button release
Power on sequence	3.08	3.08	
End of Boot_Sequence()	564.08	561	Load & apply patches
Start of Main_WaitforMsqReady()	608.88	44.8	Switch to XTAL, Driver init, excluding ps
End of Main_WaitforMsqReady()	815.88	207	Wairt for M0 ready
Start of osKernelStart()	882.68	66.8	Start running in main loop
Auto connect start	0		at+cwmode=1
Connect start	5	5	Fast connect trigger
Connect done	47	42	
Got IP	938	891	
	Total duration	1820.68	

說明:

(1) Device 從 Power on Wakeup 到進 main loop 需要 883ms。系統 Cold Boot 小於 1 s。

(2) 從 Auto Connect Start 開始，timestamp 重新計算。
執行 AT command 進行 auto connect to AP，其中 connect AP = 47ms， Got IP = 891ms。

(3) Cold Boot + Auto Connect 共需時間總和: 883+47+891 = 1821ms。

4.2. Warm Boot

場景定義: 在 Device 已建立 WiFi 連線的情況下，從 Timer Sleep Wakeup 到喚醒的時間。

量測數據: 1.6ms

	Timestamp(us)	Duration(us)
Wakeup	0	
Power on sequence	831	**831**
End of Boot_Sequence()	991	160
Start of Sys_DriverInit()	997.4	6.4
End of ps_init()	1533.4	**536**
Start of osKernelRestart()	1581.4	48
	Total duration	**1581.4**

CONTACT

sales@Opulinks.com