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. PROCEDURE DESCRIPTION OF POINT OF ENTRY TO SYSTEM

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

Code 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

Once system is powered up, Boot_CheckWarmBoot should be used to check whethere it is warm boor or cold boot. The difference between warm boot and cold boot is whether the system is in Timer Sleep Mode, and if the system is indeed in Timer Sleep Mode, the system shall quickly activate rapid system-activation with warm book, and vice versa, it would use cold boot for system activation.

If warm boot is in effect, it means that the system directly enters into Boot_MainPatch, and Boot_MainPatch will enter the main () function of the main procedure.

If cold boot is in effect, it means that the system will execute “SystemInit” and enter “_main”. “SystemInit” will initialize the clock speed of the oscillator inside the chip. “_main”will link to the corresponding main function base in C library provided by ARMCC, before executing a series of activation procedures. The actual execution procedure outlined above can refer to Code 1.

2. DESCRIPTION OF MAIN PROCEDURE

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

Main.c

Code 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() :

Once having entered main function, Boot Sequence shall be executed, however, since there are differences between Cold Boot and Warm Boot, there would be difference in action branching-out in procedures during the subsequent power-up process.

The subsequently executed Main_AppRun (), Main_AppRun () is a function pointer as the actually constructed function is due to the main body comprised of Main_AppRun-impl. By referring to Code 2, the flow of the actual actions is respectively illustrated in Figure 1 and Figure 2. The part between OS init and OS scheduling is of designation within Main_AppRun_impl. Relative to cold boot, warm boot is without certain steps, as in warm boot, OS init has been initialized during cold boot, so it will not be carried out again in warm boot procedure, however, in the last procedure of warm boot, there will be OS Rescheduling. The purpose is to allow OS to be restored to the setting prior to sleep mode. Service init and App init would have enabled some tasks in the procedure during cold boot, so that they will not be repeated during warm boot, so these two procedures are not executed during warm boot.

Figure 1: Cold Boot

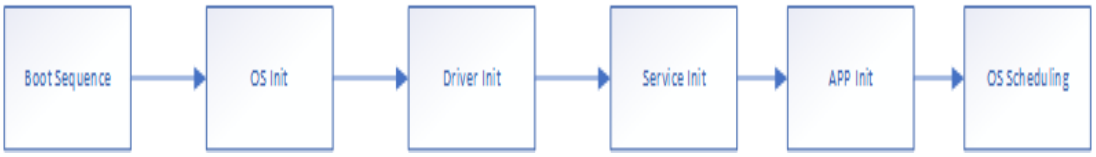
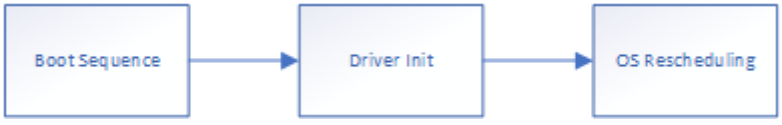


Figure 2: Warm Boot



Boot Sequence: Boot_Sequence()

- Cold Boot:

  When executing Boot Sequence in Cold Boot stage, all function points will be set to correspond with particular function, such as system function initialization, task control unit initialization, WiFi API initialization, BLE control unit initialization, LwIP (TCP/IP Stack) initialization, AT command initialization, HAL driver APT initialization, and OS initialization. This step also includes frequency setting, UART Baud rate initialization, the initialization of SPI default oscillation frequency and activation, as Flash is also designated to Internal Flash of SPI0.

- Warm Boot:

  In Warm Boot stage, there is currently no setup to be done.

OS Init: `osKernelInitialize()`

- Cold Boot:

  In Cold Boot stage, memory pool initialization.

- Warm Boot:

  In Warm Boot stage, there is currently no setup to be done.

Driver Init: `Sys_DriverInit()`

For Power supply setup initialization, setting Power parameters, system frequency initialization, and GPIO function (pin-mux) initialization, pinmux tool provided by Opulinks can be used to set up GPIO pin function. SPI0, SPI1 and SPI2 will adopt system-defined frequency to enable initialization, Flash initialization FIM (Flash Item Management) initialization, UART0/UART1 initialization, and PWM (Pulse Width Modulation) initialization.

Service Init: `Sys_ServiceInit()`

In this step, there shall be completions of AT command task unit initialization, WiFi mac task unit initialization, LWIP(TCP/IP Stack) task unit initialization, control task unit initialization, IPC unit initialization, WPA_Supplicant Initialization, and auto device-connection initialization, according to flash read setup to automatically connect with device.

APP Init: `Sys_AppInit()`

In this step of App Init, customers can add the needed initialization function here.

Post Init: `Sys_PostInit()`

In this step of Post Init, Log module (Tracer) will be subject to an initialization action. Tracer, for its name sake, will trace message to be used for debug, in realizing the tracing framework of function events in OS.

3. INITIALIZATION SETUP FOR CUSTOMIZATION

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

Customers can add new APP Init relative initialization setup, and start from there. Users can initialize variables, and also produce corresponding APP task.

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

Regarding Driver Init, Service Init, if customer side needs to add new corresponding initialization action, corresponding Patch can be created such as Sys_DriverInit_patch and Sys_ServiceInit_patch.

4. ACTUAL MEASUREMENT DATA

Regarding the procedure description of this document on Cold Boot and Warm Boot, users can understand the difference between the two action procedures, and through actual measured data, user can further understand the actual application data measured under different application scenarios. After users further understand the theory and applications, they can choose the fitting usage scenarios under different scenarios.

4.1. Cold Boot

Scenario Definition: Power on until AP establishes communication, before IP is obtained.

Measure data: 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	

Remark:

(1) Time for device from Power on Wakeup to main loop takes 883ms, as system Cold Boot takes less than 1s.
(2) Starting from Auto Connect Start, timestamp restating count。
run AT command to execute auto connect to AP， connect AP = 47ms， Got IP = 891ms。
(3) Time needed for Cold Boot + Auto Connect : 883+47+891 = 1821ms

4.2. Warm Boot

Scenario Definition: When Device has established WiFi connection, the time from Timer Sleep Wakeup to actual Wakeup.

Measured Data: 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