XRADIO WLAN Cold-Start Fast-Connection Developer Guide

Revision 1.3

Nov 19, 2019

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Date	Summary of Changes
1.0	2019-8-30	Initial Version
1.1	2019-10-11	Fix file name
1.2	2019-10-11	Fix format
1.3	2019-11-19	Add command to connect AP,or clear BSS info

Table 1- 1 Revision History

Contents

Figures

1 概述

Cold-Start Fast-Connection（冷启动快连）是 WLAN 模块用于适配一些特定的应用场景所使用的功能，其主要目的是加快设备从启动到连接到 AP 的过程时间，使设备在低功耗情况下可以迅速连接上 AP 进行通信。

冷启动快连是一种特殊的工作模式，此模式的使用需要经过特殊的流程。下面以冷启动快连示例工程（工程位置：project\example\cold_start_fast_connection）为例子，描述此流程的实现。

1.1 功能实现

为了实现启动之后快速连接 AP，需要在第一次连接 AP 之后，将 AP 对应的信息保存到 flash 中，等待下次启动的时候可以从 flash 中读取 AP 信息，省去了扫描和获取 IP 信息的过程。

因此，要完成快速连接的功能，必须有一次先连接上 AP 的过程。

1.2 实现流程

冷启动快连的大致实现流程如下：

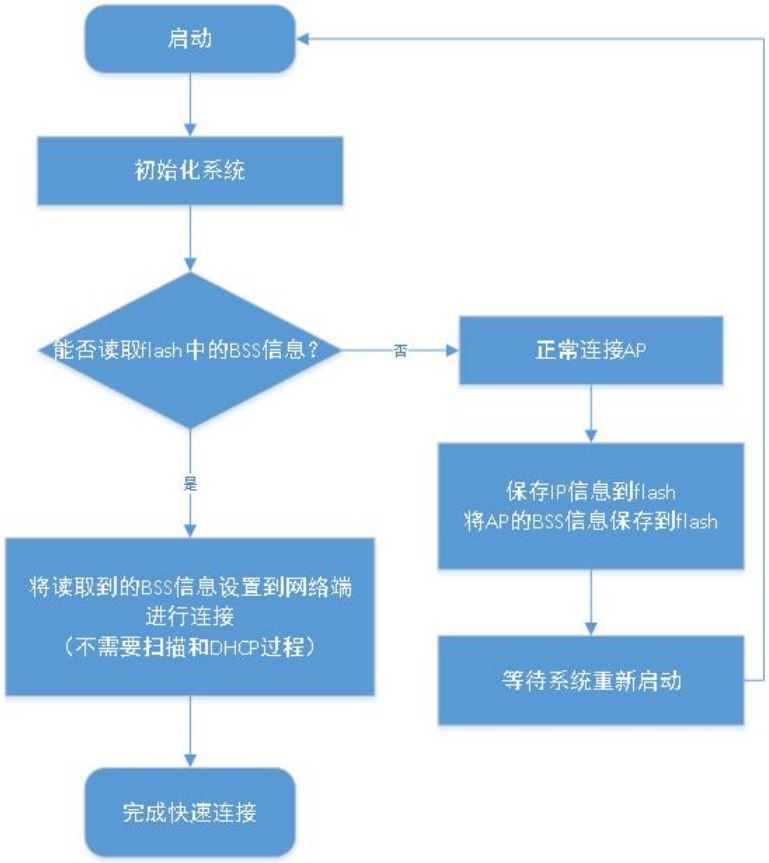

图 1-1 冷启动快连流程图

流程总结如下：

1）按下 reset 键，系统启动；

2）初始化系统；

3）从 flash 指定位置读取保存的 BSS 信息；

4）如果没有读到，则认为是首次启动，执行 5）步骤，否则认为是快速启动，执行 8）步骤；

5）进行普通连接，输入如下命令连接 AP：

```
net fc config your_ssid your_password
net fc enable
```

6）等待连接上 AP，之后，将 IP 信息保存到 flash，将 BSS 信息也保存到 flash 的指定位置，提示如下：

```
Connect AP success!
Save bss info done!
The first connection is complete, please reboot to run fast connection!

begin app:      021881us        delta:21881us
end pf init:    110687us        delta:88806us
end connection: 172222062us     delta:172111375us
delta of begin app to end connection:   172200181us
Try to ping www.baidu.com
```

此时会显示整个启动过程的耗时，并且会尝试对“www.baidu.com”进行 3 次 ping 操作，用户如不关心网络情况，可以不用关注此操作；

7）等待用户重启，重新执行 1）步骤；

8）将读到的 BSS 信息设置到网络端；

9）配置 SSID 和 PSK，此时再进行连接，不会有扫描过程和 DHCP 过程；

10）完成快连，显示整个连接过程的时间，同时也会尝试对“www.baidu.com”进行 3 次 ping 操作；

11）如用户想连接其它 AP，则输入以下命令即可清除 flash 上保存的 AP 信息，再重新执行 1）步骤即可：

```
net fc clear_bss
```

1.3 启动时间统计

完成冷启动快连后，提示的信息如下：

```
begin app:      021850us        delta:21850us
end pf init:    110656us        delta:88806us
end connection: 243301us        delta:132645us
delta of begin app to end connection:   221451us
Try to ping www.baidu.com
100 bytes from 14.215.177.38: icmp_seq=1        time=16 ms
100 bytes from 14.215.177.38: icmp_seq=2        time=31 ms
100 bytes from 14.215.177.38: icmp_seq=3        time=96 ms
```

对于以上统计时间说明如下：

- begin app：进入 app 的时间点

- end pf init：平台初始化完成的时间点
- end connection：完成 AP 连接，获取到 IP 的时间点
- delta of begin app to end connection：从进入到 app 的时间到完成 AP 连接的时间差

注意：

1.app 之前的时间为 BROM 和 bootloader 的运行时间，计时存在 ms 级误差；

2.完成 AP 连接的时间点与网络环境和 AP 行为有关，可能会有较大的偏差，以上结果使用的是华为 WS318#3 路由器测试；

2 API 介绍

在本示例中将 BSS 信息以 FDCM 的形式保存到 flash 中，对应的 API 可以参考 FDCM 相关介绍。用户也可以使用自己的接口保存到 flash 中。

2.1 wlan_sta_set

函数原型：int wlan_sta_set(uint8_t *ssid, uint8_t ssid_len, uint8_t *psk)

参数说明：ssid：需要连接的 AP 的 ssid 字符串

ssid_len：需要连接的 AP 的 ssid 字符串长度

Psk：需要连接的 AP 的 psk 字符串，若是不加密，填 NULL 即可

返回值：成功-0，失败-非 0

函数说明：此函数用于设置需要连接的 AP 的 SSID 和 PSK，在调用此函数后，网络端会需要计算出 PSK 对应的一个 32byte 的 key 值，需要大约 1s 的时间。所以在快连应用中，应该在第一次连接的时候计算出该 key 值，然后保存起来，后续快连模式时直接设置该 key 值，就不需要 1s 的计算时间了。

2.2 wlan_sta_enable

函数原型：int wlan_sta_enable(void)

参数说明：无

返回值：成功-0，失败-非 0

函数说明：此函数用于启动 STA 模式，会自动连接 AP，需要在设置了 AP 的 ssid 和 psk 后调用。

2.3 wlan_sta_gen_psk

函数原型：int wlan_sta_gen_psk(wlan_gen_psk_param_t *param);

参数说明：param：struct wlan_gen_psk_param 结构体，包含了用于生成 psk key 值的信息，定义如下：

```
/**
 * @brief Parameter of generating WPA PSK based on passphrase and SSID
 */
typedef struct wlan_gen_psk_param {
    uint8_t ssid[WLAN_SSID_MAX_LEN];
    uint8_t ssid_len;
    char passphrase[WLAN_PASSPHRASE_MAX_LEN + 1];
    uint8_t psk[WLAN_PSK_HEX_LEN]; /* out */
} wlan_gen_psk_param_t;
```

ssid：需要连接的 AP 的 ssid 字符串

ssid_len：需要连接的 AP 的 ssid 字符串长度

passphrase：需要连接的 AP 的 psk 字符串

psk：生成的 32byte key 值

返回值：成功-0，失败-非 0

函数说明：此函数用于生成 32byte key 值，在调用 wlan_sta_set()函数时，将此 key 值作为 psk 设置的话，网络端不会再进行计算，可以直接使用。

2.4 sysinfo_get

函数原型：struct sysinfo *sysinfo_get(void)

参数说明：无

返回值：成功-系统当前使用的 sysinfo 结构体指针，失败-NULL

函数说明：此函数用于获取当前系统使用的 sysinfo 结构体指针，保存了一些用户可以修改的系统配置，其具体定义如下：

```
/**
 * @brief Sysinfo structure definition
 */
struct sysinfo {
    uint32_t version;

#if PRJCONF_NET_EN
    uint32_t sta_use_dhcp : 1;

    uint8_t mac_addr[SYSINFO_MAC_ADDR_LEN];

    enum wlan_mode wlan_mode;

    struct sysinfo_wlan_sta_param wlan_sta_param;
    struct sysinfo_wlan_ap_param wlan_ap_param;

    struct sysinfo_netif_param netif_sta_param;
    struct sysinfo_netif_param netif_ap_param;
#endif
```

本次仅使用到 sta_use_dhcp 和 netif_sta_param，其说明如下：

sta_use_dhcp：启动 STA 模式时是否使用 DHCP 获取 ip 地址，1-使用 DHCP，0-使用静态 ip

netif_sta_param：保存使用静态 ip 时对应的 ip 地址、mask 和 gateway 地址，具体定义如下：

```
/**
 * @brief Sysinfo net interface parameters definition
 */
struct sysinfo_netif_param {
#ifdef __CONFIG_LWIP_V1
    ip_addr_t ip_addr;
    ip_addr_t net_mask;
```

```
    ip_addr_t gateway;
#elif LWIP_IPV4 /* now only for IPv4 */
    ip4_addr_t ip_addr;
    ip4_addr_t net_mask;
    ip4_addr_t gateway;
#else
    #error "IPv4 not support!"
#endif
};
```

2.5 sysinfo_save

函数原型：int sysinfo_save(void)

参数说明：无

返回值：成功-0，失败- -1

函数说明：此函数用于将当前使用的 sysinfo 数据保存到 flash 中，以便下次系统启动时使用。

2.6 wlan_sta_get_bss_size

函数原型：int wlan_sta_get_bss_size(uint32_t * size);

参数说明：size：返回当前连接的 AP 的 BSS 信息数据大小

返回值：成功-0，失败- -1

函数说明：此函数用于返回当前连接的 AP 的 BSS 信息数据大小，得到这个大小的值后，才能去获取实际 BSS 的数据。

2.7 wlan_sta_get_bss

函数原型：int wlan_sta_get_bss(wlan_sta_bss_info_t * bss_get);

参数说明：bss_get：返回当前连接的 AP 的 BSS 信息数据

返回值：成功-0，失败- -1

函数说明：此函数用于返回当前连接的 AP 的 BSS 信息数据，wlan_sta_bss_info_t 定义如下：

```
/**
 * @brief Wlan station bss infomation definition
 */
typedef struct wlan_sta_bss_info {
    uint8_t *bss;
    uint32_t size;
} wlan_sta_bss_info_t;
```

bss：实际网络端使用的 bss 数据，用户不需要做解析

size：bss 数据的大小，需要调用 wlan_sta_get_bss_size()来获取

2.8 wlan_sta_set_bss

函数原型：int wlan_sta_set_bss(wlan_sta_bss_info_t * bss_set);

参数说明：bss_set：设置当前需要连接的 AP 的 BSS 信息数据

返回值：成功-0，失败- -1

函数说明：此函数用于设置当前需要连接的 AP 的 BSS 信息到网络端，之后再调用 wlan_sta_enable()进行连接时就不会有扫描的流程了。

3 注意事项

3.1 缩短启动时间

由于快速启动对时间要求较高，在必要时候，可以对以下影响因素进行调试，以便缩短启动时间：

3.1.1 打印

关闭多余打印，可以有效地缩短时间。

本例中使用接口 stdout_enable()来动态开关打印，详细请参考代码中的调用。注意，使用此接口只会关闭硬件串口的输出，代码的运行仍然存在，所以还是会有部分时间消耗在代码运行上。

3.1.2 CPU 频率

提高 CPU 频率，也可以一定程度上缩短时间

3.1.3 Flash ctrl 时钟

提高 flash ctrl 的时钟频率也能缩短一些启动时间，但对于大部分代码都存放在 xip 中的情况并不会有很大改善。

3.2 保存 BSS 信息

本例中使用 FDCM 将 BSS 信息保存在 flash 的指定位置，FDCM 区域有其自定义的数据保存格式，有可以多次写入后才进行擦除的优势。

用户如果不想使用 FDCM 的方式也是可以的，只需要保证将 BSS 信息保存好，下次启动时可以快速获取到即可。

推荐使用 FDCM 进行管理。

3.3 保存时间信息

本例中，记录时间点使用的函数为 HAL_RTC_GetFreeRunTime()，然后使用接口 save_time()将时间信息保存在一块指定的内存区域中，最后使用接口 get_time()打印输出统计的各个时间点，以及它们之间的差值。

具体接口不在此处进行介绍了，用户如果需要，可以直接参考代码中的调用。