

# 1 介绍

本软件使用uC/OS-III作为嵌入式操作系统，lwip作为网络协议栈，STM32F4xx作为微处理器，移植了联发科的USB无线网卡驱动DPO\_RT5572\_LinuxSTA\_2.6.1.3\_20121022，并针对嵌入式系统做了优化。为了减少移植过程的难度，我简单地实现了无线驱动所调用的Linux内核接口 [参见:USB Host Linux Convert](#)。本软件提供了使用iPerf进行网络性能测试的演示例子。

本软件特性：

- 支持热插拔
- 支持 WEP、WPAPSK-AES、WPAPSK-TKIP、WPA2PSK-AES、WPA2PSK-TKIP 等认证和加密方式
- 支持 802.11b/g/n
- 使用 iwpriv 工具对无线网卡进行配置
- 支持 USB 无线网卡型号 RT5370、RT3070

第三方软件代码：

- uC/OS-III V3.03.01  
<http://micrium.com/downloadcenter/download-results/?searchterm=mp-uc-os-iii-1&supported=true>
- lwip-1.4.1  
<http://download.savannah.gnu.org/releases/lwip/>
- DPO\_RT5572\_LinuxSTA\_2.6.1.3\_20121022  
<http://www.mediatek.com/en/downloads1/downloads/rt8070-rt3070-rt3370-rt3572-rt5370-rt5372-rt5572-usb-usb/>
- iPerf V2.0.5  
<https://iperf.fr/iperf-download.php>

Author: Kaiqin Lin

Email: linkaiqin@sina.com

(last modified: Jan 15, 2016)

# 目录

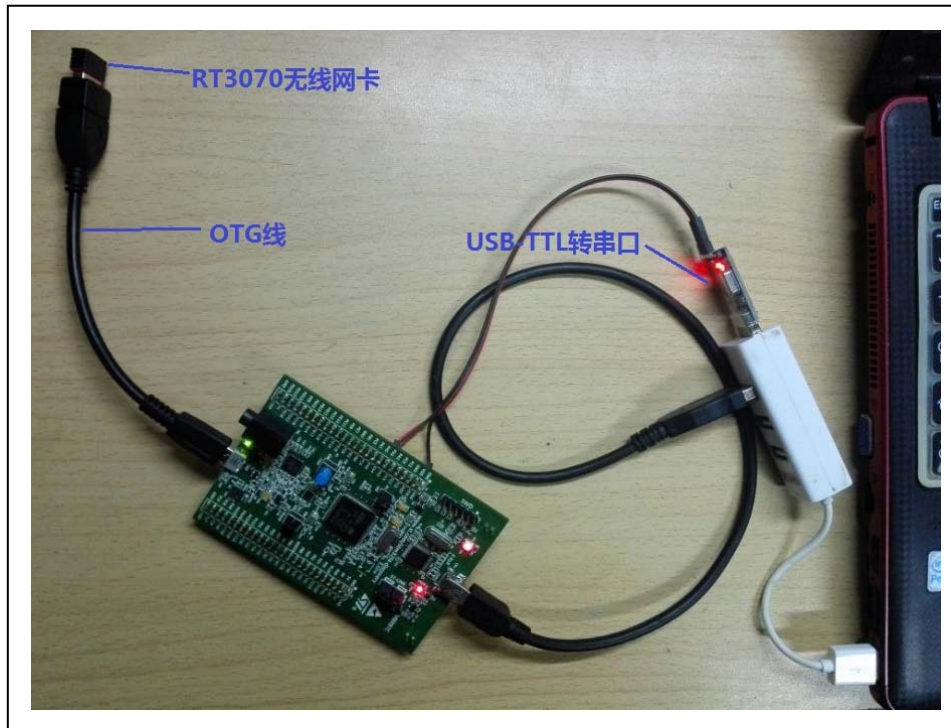
1	介绍 .....	1
2	硬件连接.....	3
3	演示例子.....	4
3.1	准备.....	4
3.2	配置.....	4
3.3	iPERF网络性能测试.....	5
4	无线网络配置.....	8
4.1	通过RT2870STA_CON.H配置.....	8
4.2	通过IWPRIV工具配置.....	8
4.3	通过函数WIRELESS_EXEC_CMD()配置.....	10
5	关于IPERF的使用限制.....	12
6	关于RAM的使用.....	13
6.1	关于堆的分配.....	13
6.2	关于线程栈的分配.....	13
7	DPO_RT5572_LINUXSTA_2.6.1.3_20121022 的移植过程.....	15
7.1	获取编译参数-D宏定义.....	15
7.2	优化结构体大小.....	15
7.3	动态分配->全局分配.....	16
7.4	其它.....	17
8	USB_HOST_LINUX_CONVERT .....	18

## 2 硬件连接

演示例子所需要的硬件有

- STM32F4-Discovery Board
- OTG 线
- RT3070 或 RT5370 无线网卡
- 无线路由器
- USB-TTL 转串口

图 1 STM32F4-Discovery 硬件连接



## 步骤

1. 连接 USB-TTL 转串口到 STM32F4-Discovery 板子上的 USART2 (PA2 和 PA3)
2. 用一根 OTG 线连接 STM32F4-Discovery 板子到 RT3070 无线网卡。
3. 用一根 USB 线 (type A to mini-B) 连接 STM32F4-Discovery 板子到 PC。

## 3 演示例子

### 3.1 准备

演示例子之前请先安装好

- Keil MDK V4.x
- STLINK V2
- SecureCRT（配置成 115200 波特率 8 位无校验）
- iPerf for Windows

### 3.2 配置

演示例子默认的无线路由器配置为

SSID:mytest

密码: 12345678

认证方式: WPA2PSK

加密方式: AES

如果你的无线路由器不是以上配置话, 请按以下步骤修改默认配置[具体参见: 无线网络配置](#)

1. 编辑 EvalBoards\ST\STM32F4-Discovery\uCOS-III\rt2870sta\_conf.h
2. 修改键 AuthMode, 根据你的无线路由器认证类型, 填入以下值(避免空格)  
OPEN        开放系统  
SHARED      共享密钥系统  
WPAPSK      基础型 WPA 预共享密钥  
WPA2PSK     基础型 WPA2 预共享密钥
3. 修改键 EncrypType, 根据你的无线路由器加密类型, 填入以下值(避免空格)  
NONE        当 AuthMode=OPEN 时  
WEP         当 AuthMode=SHARED 时  
TKIP        当 AuthMode=WPAPSK 或者 WPA2PSK 时  
AES         当 AuthMode=WPAPSK 或者 WPA2PSK 时
4. 修改键 WPAPSK 或 Key1Str, 这里填入你的无线路由器密码。  
当 AuthMode=SHARED 时, 在 Key1Str 处设置密码。  
当 AuthMode=WPAPSK 或 WPA2PSK 时, 在 WPAPSK 处设置密码。
5. 修改键 SSID, 这里填入你的无线路由器 SSID。
6. 运        行        EvalBoards\ST\STM32F4-Discovery\uCOS-III\KeilMDK\

uCOS-III.uvproj, 编译并下载程序。

7. 有 关 配 置 的 更 多 细 节 , 见  
Ralink-WIFI\DP0\_RT5572\_LinuxSTA\_2.6.1.3\_20121022\_highly\_optimize\  
README\_STA\_usb

### 3.3 iPerf网络性能测试

程序下载完成后, 软件将尝试连接 rt2870sta\_conf.h 中配置的 SSID。一旦连接成功后, 会启动 DHCP 获取 IP 地址, 在 SecureCRT 中会输出以下内容。现假设 STM32F4xx 获取到的 IP 为 192.168.1.117, PC 获取到的 IP 为 192.168.1.101。

```
wireless_send_event[35842] (RT2860) BSS(ra0) Scanning
wireless_send_event[35842] (RT2860) BSS(ra0) scan completed
wireless_send_event[35842] (RT2860) BSS(ra0) had associated successfully
wireless_send_event[35842] (RT2860) STA(f4:ee:14:54:4d:2e) connects with our
wireless client
dhcp_start...
DHCP IP:192.168.1.117
DHCP GW:192.168.1.1
DHCP MASK:255.255.255.0
```

## 测试 iPerf UDP 客户端

现以测试时间为 10 秒，测试带宽 4Mbps，PC 每 1 秒钟报告一次结果为例。

STM32F4xx 下，通过 SecureCRT 命令行输入：

```
>iperf -c 192.168.1.101 -u -b 4Mbps -t 10  
-----  
Client connecting to 192.168.1.101, UDP port 5001  
Sending 1470 byte datagrams  
UDP buffer size: 2.85 KByte (default)  
-----
```

Windows 下，运行 cmd.exe，进入到 iperf-2.0.5-cygwin/ 目录中，输入：

```
F:\iperf-2.0.5-win32\iperf-2.0.5-cygwin>iperf.exe -s -u -i 1  
-----  
Server listening on UDP port 5001  
Receiving 1470 byte datagrams  
UDP buffer size: 64.0 KByte (default)  
-----  
[ 3] local 192.168.1.100 port 5001 connected with 192.168.1.117 port 49154  
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagram  
[ 3] 0.0- 1.0 sec   449 KBytes  3.68 Mbits/sec  1.352 ms  186/ 499 (37%)  
[ 3] 1.0- 2.0 sec   446 KBytes  3.66 Mbits/sec  1.195 ms   45/ 356 (13%)  
[ 3] 2.0- 3.0 sec   446 KBytes  3.66 Mbits/sec  1.541 ms   46/ 357 (13%)  
[ 3] 3.0- 4.0 sec   449 KBytes  3.68 Mbits/sec  1.073 ms   43/ 356 (12%)  
[ 3] 4.0- 5.0 sec   446 KBytes  3.66 Mbits/sec  1.353 ms   47/ 358 (13%)  
[ 3] 5.0- 6.0 sec   444 KBytes  3.63 Mbits/sec  1.520 ms   48/ 357 (13%)  
[ 3] 6.0- 7.0 sec   449 KBytes  3.68 Mbits/sec  1.432 ms   43/ 356 (12%)  
[ 3] 7.0- 8.0 sec   446 KBytes  3.66 Mbits/sec  1.392 ms   44/ 355 (12%)  
[ 3] 8.0- 9.0 sec   448 KBytes  3.67 Mbits/sec  1.175 ms   47/ 359 (13%)  
[ 3] 0.0- 9.6 sec   4.20 MBytes 3.66 Mbits/sec  3.188 ms  572/ 3570 (16%)
```

## 测试 iPerf UDP 服务器

现以测试带宽 8Mbps，STM32F4xx 每 1 秒钟报告一次结果为例。

STM32F4xx 下，通过 SecureCRT 命令行输入：

```
>iperf -s -u -i 1
Spawning a listener.
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 2.85 KByte (default)
-----
```

Windows 下，运行 cmd.exe，进入到 iperf-2.0.5-cygwin/目录中，输入：

```
F:\iperf-2.0.5-win32\iperf-2.0.5-cygwin>iperf.exe -c 192.168.1.117 -u -b 8Mbps
-----
Client connecting to 192.168.1.117, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
-----
[ 3] local 192.168.1.100 port 49678 connected with 192.168.1.117 port 5001
```

STM32F4xx 输出结果：

```
Spawning a server.
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagrams
[ 0] 0.0- 1.0 sec   846 KBytes  6.93 Mbits/sec  0.634 ms    0/ 589 (0%)
[ 0] 1.0- 2.0 sec   841 KBytes  6.89 Mbits/sec  0.613 ms    1/ 587 (0.17%)
[ 0] 2.0- 3.0 sec   846 KBytes  6.93 Mbits/sec  0.633 ms   46/ 635 (7.2%)
[ 0] 3.0- 4.0 sec   844 KBytes  6.91 Mbits/sec  0.770 ms  104/ 692 (15%)
[ 0] 4.0- 5.0 sec   846 KBytes  6.93 Mbits/sec  0.586 ms  133/ 722 (18%)
[ 0] 5.0- 6.0 sec   846 KBytes  6.93 Mbits/sec  0.671 ms  134/ 723 (19%)
[ 0] 6.0- 7.0 sec   846 KBytes  6.93 Mbits/sec  0.523 ms  134/ 723 (19%)
[ 0] 7.0- 8.0 sec   846 KBytes  6.93 Mbits/sec  0.729 ms   97/ 686 (14%)
[ 0] 8.0- 9.0 sec   846 KBytes  6.93 Mbits/sec  0.749 ms  139/ 728 (19%)
[ 0] 9.0-10.0 sec   848 KBytes  6.95 Mbits/sec  0.876 ms  143/ 734 (19%)
[ 0] 0.0-10.5 sec   8.64 MBytes  6.92 Mbits/sec  0.713 ms  977/ 7138 (14%)
[ 0] 0.0-10.5 sec   1 datagrams received out-of-order
```

## 4 无线网络配置

无线网络配置可以通过以下三种方式配置

- 通过修改 `rt2870sta_con.h` 进行配置
- 通过 `iwpriv` 工具进行配置
- 通过调用函数 `wireless_exec_cmd()` 进行配置

### 4.1 通过 `rt2870sta_con.h` 配置

`rt2870sta_con.h` 的配置是默认配置，默认将连接这个配置文件的 SSID。根据你无线路由器的配置修改这个文件。

1. 编辑 `EvalBoards\ST\STM32F4-Discovery\uCOS-III\rt2870sta_con.h`
2. 修改键 `AuthMode`，根据你的无线路由器认证类型，填入以下值(避免空格)  

OPEN	开放系统
SHARED	共享密钥系统
WPAPSK	基础型 WPA 预共享密钥
WPA2PSK	基础型 WPA2 预共享密钥
3. 修改键 `EncrypType`，根据你的无线路由器加密类型，填入以下值(避免空格)  

NONE	当 <code>AuthMode=OPEN</code> 时
WEP	当 <code>AuthMode=SHARED</code> 时
TKIP	当 <code>AuthMode=WPAPSK</code> 或者 <code>WPA2PSK</code> 时
AES	当 <code>AuthMode=WPAPSK</code> 或者 <code>WPA2PSK</code> 时
4. 修改键 `WPAPSK` 或 `Key1Str`，这里填入你的无线路由器密码。  
当 `AuthMode=SHARED` 时，在 `Key1Str` 处设置密钥。  
当 `AuthMode=WPAPSK` 或 `WPA2PSK` 时，在 `WPAPSK` 处设置密钥。
5. 修改键 `SSID`，这里填入你的无线路由器 SSID。
6. 运 行 `EvalBoards\ST\STM32F4-Discovery\uCOS-III\KeilMDK\uCOS-III.uvproj`，编译并下载程序。
7. 有 关 配 置 的 更 多 细 节 ， 见 `Ralink-WIFI\DP0_RT5572_LinuxSTA_2.6.1.3_20121022_highly_optimize\README_STA_usb`

### 4.2 通过 `iwpriv` 工具配置

通过串口使用 `iwpriv` 命令进行配置



使用方法: iwpriv set [参数]=[值]

主要的配置参数有:

参数	值		备注
SSID	信道名称		
AuthMode (认证模式)	OPEN	开放系统	
	SHARED	共享密钥系统	
	WPAPSK	基础型 WPA 预共享密钥	
	WPA2PSK	基础型 WPA2 预共享密钥	
EncrypType (加密类型)	NONE	无加密	当 AuthMode=OPEN 时
	WEP	有线等效加密	当 AuthMode=SHARED 时
	TKIP	临时密钥完整性协议	当 AuthMode=WPAPSK 或者 WPA2PSK 时
	AES	高级加密标准	当 AuthMode=WPAPSK 或者 WPA2PSK 时
WPAPSK	预共享密钥 (8-63 个 ASCII 码)		当 AuthMode=WPAPSK 或者 WPA2PSK 时
KEY1	WEP 密钥(5 个 ASCII 码或 13 个 ASCII 码)		当 AuthMode=SHARED 时

下面的表格也许能更好帮助你理解上述关系:

参数	AuthMode	EncrypType	WPAPSK	KEY1
值	OPEN	NONE		
	SHARED	WEP		此参数设置密钥
	WPAPSK	TKIP	此参数设置密钥	
		AES	此参数设置密钥	
	WPA2PSK	TKIP	此参数设置密钥	
		AES	此参数设置密钥	

例子:

a) 假设路由器认证和加密类型分别是 OPEN/NONE, SSID 为 mytest

```
iwpriv set AuthMode=OPEN
```

```
iwpriv set EncrypType=NONE
```

```
iwpriv set SSID=mytest
```

- b) 假设路由器认证和加密类型分别是 SHARED/WEP, SSID 为 mytest, WEP 密钥为 12345

```
iwpriv set AuthMode=SHARED
```

```
iwpriv set EncrypType=WEP
```

```
iwpriv set Key1=12345
```

```
iwpriv set SSID=mytest
```

- c) 假设路由器认证和加密类型分别是 WPA2PSK/TKIP, SSID 为 mytest, 预共享密钥为 12345678

```
iwpriv set AuthMode=WPA2PSK
```

```
iwpriv set EncrypType=TKIP
```

```
iwpriv set WPAPSK=12345678
```

```
iwpriv set SSID=mytest
```

- d) 假设路由器认证和加密类型分别是 WPA2PSK/AES, SSID 为 mytest, 预共享密钥为 12345678

```
iwpriv set AuthMode=WPA2PSK
```

```
iwpriv set EncrypType=AES
```

```
iwpriv set WPAPSK=12345678
```

```
iwpriv set SSID=mytest
```

- e) 假设路由器认证和加密类型分别是 WPAPSK/AES, SSID 为 mytest, 预共享密钥为 12345678

```
iwpriv set AuthMode=WPAPSK
```

```
iwpriv set EncrypType=AES
```

```
iwpriv set WPAPSK=12345678
```

```
iwpriv set SSID=mytest
```

有关该驱动 iwpriv 工具的更多使用方法, 见  
Ralink-WIFI\DP0\_RT5572\_LinuxSTA\_2.6.1.3\_20121022\_highly\_optimize\  
iwpriv\_usage.txt

### 4.3通过函数wireless\_exec\_cmd()配置

头文件: wlan.h

函数原型: int wireless\_exec\_cmd(char \*cmd);

wireless\_exec\_cmd()只是简单传入 iwpriv 的命令行,它的使用方法和 4.2 节一样。

例子:

- a) 假设路由器认证和加密类型分别是 WPA2PSK/AES, SSID 为 mytest, 预共享密钥为 12345678

```
wireless_exec_cmd("iwpriv set AuthMode=WPA2PSK");  
wireless_exec_cmd("iwpriv set EncrypType=AES");  
wireless_exec_cmd("iwpriv set SSID=mytest");  
wireless_exec_cmd("iwpriv set WPAPSK=12345678");
```

## 5 关于iPerf的使用限制

STM32F4xx iPerf 是移植的 Linux 下的 iPerf，目前它有以下限制：

- 目前 STM32F4xx iPerf 只有单一线程。
- 当 STM32F4xx iPerf 运行在客户端模式时，如果不能连接上服务器，STM32F4xx iPerf 将会中止，除非复位。
- 许多 iPerf 的参数不能改变，例如 TCP window size 和 max segment size
- 当 STM32F4xx iPerf 处于服务器模式时，没有任何键会回到命令终端的模式，使用堆分配的内存也始终占用着，除非复位。
- 当 iPerf 运行在服务器模式时，如果想改变参数，只有复位。
- 当 STM32F4xx iPerf 运行在客户端模式时，-P 和-d 参数不可用，因为 STM32F4xx iPerf 只有单一线程。

## 6 关于RAM的使用

STM32F4xx 有 192KB 的 RAM，其中 128KB 在 0x20000000 区域，另外 64KB 在 0x10000000 区域。使用情况见表。

STM32F4xx 内存使用情况		
内存地址	大小	已使用
0x20000000	128KB	73.8KB
0x10000000	64KB	52KB

有关内存的更详细使用情况见编译完成后的 `EvalBoards\ST\STM32F4-Discovery\uCOS-III\KeilMDK\Lst\uCOS-III.map`

### 6.1 关于堆的分配

STM32F4xx 有 0x20000000 和 0x10000000 两个内存区域，由于同时管理两个不连续内存的堆分配会比较麻烦，所以我将一些分配内存比较大的变量以全局变量的方式定义在 0x10000000 区域（大概占用了 59.1KB 的内存）。

无线驱动的内存分配使用的是 lwip 的内存分配函数，无线驱动和 lwip 共用一个堆，堆的大小为 26KB，定义在 0x20000000 区域。之所以共用，而不是为无线驱动单独定义堆用来分配内存，是因为无线驱动只是在初始化和销毁过程（USB 网卡的插和拔）会占用比较大的内存（大概 17KB），而在进行 TCP 或 UDP 数据传输时只占用了 7.2KB 左右的内存，因此和 lwip 共用堆进行内存分配是比较节省内存的。当然如果你想使用不同的堆进行分配，可以在 memory.h 中取消对 USE\_LWIP\_MALLOC 的宏定义，并在 Keil MDK 工程目录中将 lwip\mem.c 包含进来。

### 6.2 关于线程栈的分配

线程的栈占用了大部分内存，本软件总共创建了 12 个任务。使用情况见表。

线程栈使用情况		
线程名	栈大小(Byte)	备注
RtmpCmdQTask	2048	无线驱动程序启动的任务
RtmpMlmeTask	2048	无线驱动程序启动的任务
RtmpTimerTask	2048	无线驱动程序启动的任务

USBH Probe Task	2048	USB 主机探寻任务, 用于热插拔
USBH_Task	512	USB 主机调度任务
Tasklet Action	1024	类型 Linux 下的 Tasklet 小任务
TCP/IP	2048	lwip 启动的任务
App Task Start	2048	第一个启动的应用程序任务, 现用于处理 shell 命令
uC/OS-III Timer Task	512	uC/OS-III 内核定时任务
uC/OS-III Stat Task	512	uC/OS-III 内核统计任务
uC/OS-III Tick Task	512	uC/OS-III 内核滴答定时器任务
uC/OS-III Idle Task	512	uC/OS-III 内核空闲任务

## 7 DPO\_RT5572\_LinuxSTA\_2.6.1.3\_20121022 的移植过程

### 7.1 获取编译参数-D宏定义

1. 编辑 DPO\_RT5572\_LinuxSTA\_2.6.1.3\_20121022\Makefile
2. 在 387 行增加如下内容：  
@echo \$(WFLAGS) > define.txt
3. 在 Linux 下编译 DPO\_RT5572\_LinuxSTA\_2.6.1.3\_20121022 源代码
4. 编译完成后得到 define.txt
5. 增加宏-DVENDOR\_FEATURE2\_SUPPORT，它会帮助你及时发现移植过程是否造成了内存泄露。
6. 增加宏-DMEMORY\_OPTIMIZATION，这个宏会减少内存的使用。
7. 删除宏-DIQ\_CAL\_SUPPORT，调试时它总是输出些烦人的消息。

### 7.2 优化结构体大小

#### 7.2.1 优化RTMP\_ADAPTER结构体

RTMP\_ADAPTER 结构体占用空间是最大的，但也是最重要的，驱动的大部分函数都会用到它。优化顺序如下：

1	Rtmp_def.h 中将 MAX_REORDERING_MPDU_NUM 由 256 改为 4	内存减少约 300 多 K
2	Oid.h 中将 MAX_NUMBER_OF_MAC 由 32 改为 2	内存减少约 60 多 K
3	Oid.h 中将 MAX_NUMBER_OF_ACL 由 64 改为 2	内存减少约 60 多 K
4	Rtmp_def.h 中将 HASH_TABLE_SIZE 由 256 改为 1	内存减少约为 1K
5	Rtmp_def.h 中将 MGMT_RING_SIZE 由 32 改为 1	内存减少约为 1K
6	Rtmp_def.h 中将 TX_RING_SIZE 由 8 改为 2	内存未减少
7	Rtmp_def.h 中将 PRIO_RING_SIZE 由 8 改为 2	内存未减少
8	Rtmp_def.h 中将 RX_RING_SIZE 由 8 改为 1	内存减少 360 多字节
9	mlme.h 中将 MAX_LEN_OF_MLME_QUEUE 由 40 改为 20	内存减少 800 多字节

RTMP\_ADAPTER 结构体中移除 UsbVendorReqBuf，UsbVendorReqBuf 是 Linux 下专门用于 USB DMA 传输的缓冲区，STM32F4xx 下并不需要。

优化后 RTMP\_ADAPTER 结构体占用内存为 33792 字节。

#### 7.2.2 优化HTTX\_BUFFER结构体

未优化前 HTTX\_BUFFER 占用 122884 字节。

Rt\_linux.h 中将 BULKAGGRE\_SIZE 由 60 改为 7 后，内存占用 14340 字节

### 7.2.3 优化RX\_CONTEXT结构体

未优化前 RX\_CONTEXT->TransferBuffer 将分配 24K 字节的内存。

Rtmp\_usb.h 中将 RXBULKAGGRE\_SIZE 由 12 改为 2 后，只分配 4K 字节内存。

### 7.2.4 优化MLME\_QUEUE\_ELEM结构体

其它线程给 RtmpMlmeTask 线程发送的消息时，会将消息填充到 MLME\_QUEUE\_ELEM->Msg[MGMT\_DMA\_BUFFER\_SIZE]成员中，它的大小是固定的，因此占用了大量内存。为了减少内存使用，需要将固定分配的内存改为动态分配（也就是用到时再分配内存）。

1. 将 MLME\_QUEUE\_ELEM->Msg[] 改为指针 MLME\_QUEUE\_ELEM->Msg
2. 对源代码用到 MLME\_QUEUE\_ELEM->Msg 的地方做修改，动态分配内存。
3. 在 RtmpMlmeTask 线程中释放掉的 MLME\_QUEUE\_ELEM->Msg 所占用的内存。

我将该优化使用宏 MLMEQUEUE\_ALLOC\_INUSE 作为开关。你可以直接在工程中搜索宏 MLMEQUEUE\_ALLOC\_INUSE，以查找该优化修改了哪些源代码。

## 7.3动态分配->全局分配

由于一些变量的结构体比较大，会分配很多内存空间，我将这些变量以全局变量的方式定义在 0x10000000 区域。

1. rt\_linux.c 中将 AdapterBlockAllocateMemory() 函数所分配的结构体 RTMP\_ADAPTER 改为全局的方式定义。
2. rtmp\_init.c 中将 RTMPAllocAdapterBlock() 函数所分配的 Beacon 缓冲区 RTMP\_ADAPTER->pBeaconBuf 改为全局的方式定义。
3. rtmp\_init.c 中将 RtmpRaDevCtrlInit() 函数所分配的缓冲区 RTMP\_ADAPTER->UsbVendorReqBuf 改为全局的方式定义。
4. mlme.c 中将 NICInitTransmit() 函数所分配的发送 802.11 数据帧缓冲区 RTMP\_ADAPTER->TxContext[NUM\_OF\_TX\_RING]->TransferBuffer 改为全局的方式定义。另外在测试时候发现驱动程序只使用到了 TxContext[AC0]的数据帧缓冲区，故只定义 AC0 通道的数据帧缓冲区。
5. mlme.c 中将 NICInitTransmit() 函数所分配的发送 802.11 NULL 帧缓冲区 RTMP\_ADAPTER->NullContext[2]->TransferBuffer 改为全局的方式定义。另外在测试时候发现驱动程序只使用到了 NullContext [0]的数据帧缓冲区，故只定义 0 通道的 NULL 帧缓冲区。



6. mlme.c 中将 NICInitTransmit() 函数所分配的发送 802.11 PS-POLL 帧缓冲区 RTMP\_ADAPTER->PsPollContext->TransferBuffer 改为全局的方式定义。
7. rtmp\_init.c RTMPAllocAdapterBlock() 中 RTMP\_ADAPTER->ProbeRespIE.pIe 动态分配改为全局分配。
8. spectrum.c TpcReqTabInit() 中 RTMP\_ADAPTER->CommonCfg.pTpcReqTab 动态分配改为全局分配。
9. spectrum.c MeasureReqTabInit() 中 RTMP\_ADAPTER->CommonCfg.pMeasureReqTab 动态分配改为全局分配。

我将这些优化使用宏 HIGHLY\_OPTIMIZE 作为开关。你可以直接在工程中搜索宏 HIGHLY\_OPTIMIZE，以查找该优化修改了哪些源代码。

## 7.4其它

1. rtmp\_timer.h 中将 TIMER\_QUEUE\_SIZE\_MAX 由 128 改为 32，在测试过程发现驱动程序实际上用不了这么多定时器。
2. 在 DP0\_RT5572\_LinuxSTA\_2.6.1.3\_20121022 的源代码中有些常量变量没有加 const，对嵌入式系统来说这不仅会占用 ROM 空间也会占用 RAM 空间。我在这些常量变量前加了 const 以减少 RAM 的使用。
3. 在 RT2870STA.dat 中设置 WirelessEvent=1，这样 wlan.c 中 wireless\_send\_event() 函数才会接收到无线驱动连接或断开路由器等消息。
4. cmm\_data.c 中 MiniportMMRequest() 函数第 88 行：  
    UCHAR rtmpHwHdr[TXINFO\_SIZE + pAd->chipCap.TXWISize];  
这句代码使用变量来定义局部变量大小，测试过程发现它会导致编译对 rtmpHwHdr 变量使用自带的 malloc 函数进行内存分配。在确保 pAd->chipCap.TXWISize 不会超过 32 后，我对它进行了如下修改：  
    UCHAR rtmpHwHdr[TXINFO\_SIZE + 32];
5. 配置文件中 FragThreshold=2346，lwip 不会发送如此大数据包，因此数据包不会分片。函数 RTMPAllocTxRxRingMemory() 中取消对 pAd->FragFrame.pFragPacket 分配内存。
6. Probe Response 帧响应最大长度 MAX\_VIE\_LEN 1024->256，以减少内存的使用。
7. 修复 BUG，STARxDoneInterruptHandle() 函数 940 行，当 pRxWI->MPDUtotalByteCount < 14 的话，应该释放数据包，否则会造成内存泄露。

## 8 USB\_Host\_Linux\_Convert

由于 DPO\_RT5572\_LinuxSTA\_2.6.1.3\_20121022 的驱动是针对 Linux 操作系统的，为了减小移植的难度，我在 uC/OS-III 下简单地实现该驱动所需要的 Linux 内核接口，我把它们放在 USB\_Host\_Linux\_Convert\目录下。

实现的接口		
类别	头文件	实现的接口
USB 主机	usbh_linux.h	usb_submit_urb() usb_kill_urb() usb_control_msg() usb_bulk_msg() ...
线程	kthread.h	kill_pid() kernel_thread()
complete	complete.h	init_completion() complete() complete_and_exit() wait_for_completion_timeout() wait_for_completion()
自旋锁	spinlock.h	spin_lock_init() spin_lock() spin_unlock() spin_lock_irqsave() spin_unlock_irqrestore()
信号量	sem.h	sema_init() down_interruptible() up() down_trylock() sema_destroy()
定时器	timer.h	mod_timer() init_timer() add_timer() timer_pending() del_timer()