1 Introduction

The software utilizes uC/OS-III as the embedded operating system. Lwip as the network protocol stack, STM32F4xx as the microprocessor. We port and optimize the MediaTek's USB Wireless LAN Driver DPO_RT5572_LinuxSTA_2.6.1.3_20121022 for embedded systems. To reduce the difficulty of porting process. We implement the Linux kernel interface for USB Wireless Driver(*See also:USB_Host_Linux_Convert*). The software provides a demo of using iPerf to test network performance.

The software features:

- HotPlug.
- Support authentication and encryption like WEP、WPAPSK-AES、WPAPSK-TKIP、WPA2PSK-AES、WPA2PSK-TKIP.
- Support 802.11b/g/n
- Utilize iwpriv to configure the Wireless LAN.
- Tested USB Wireless LAN type RT3070.

The third-party software code:

- 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

Issue Date:

(last modified: Nov 17, 2015)

Author: Kaiqin Lin, linkaiqin@sina.com
Jianguo Zhang, jgzhang.cross@foxmail.com

Catalogue

1 Introduction 1

2 Hardware Connection 3

3 Sampled Demo 4

3.1 Preparation 4

3.2 Configuration 4

3.3 Iperf Network Performance Test 5

4 Wireless Network Configuration 7

4.1 Configure through modifying RT2870STA.dat 7

4.2 Configuration Through iwpriv 8

4.3 Configuration Through wireless_exec_cmd( ) 10

5 Usage Restrictions of iperf 11

6 Usage of RAM 12

6.1 Heap Allocation 12

6.2 Thread Stack Allocation 12

7 How to port from DPO_RT5572_LinuxSTA_2.6.1.3_20121022 14

7.1 Obtain Compiler Parameters - D Macro Definition 14

7.2 Optimize The Size of Structure 14

7.3 Dynamic Allocation->Global Static Allocation 15

7.4 Other optimization 16

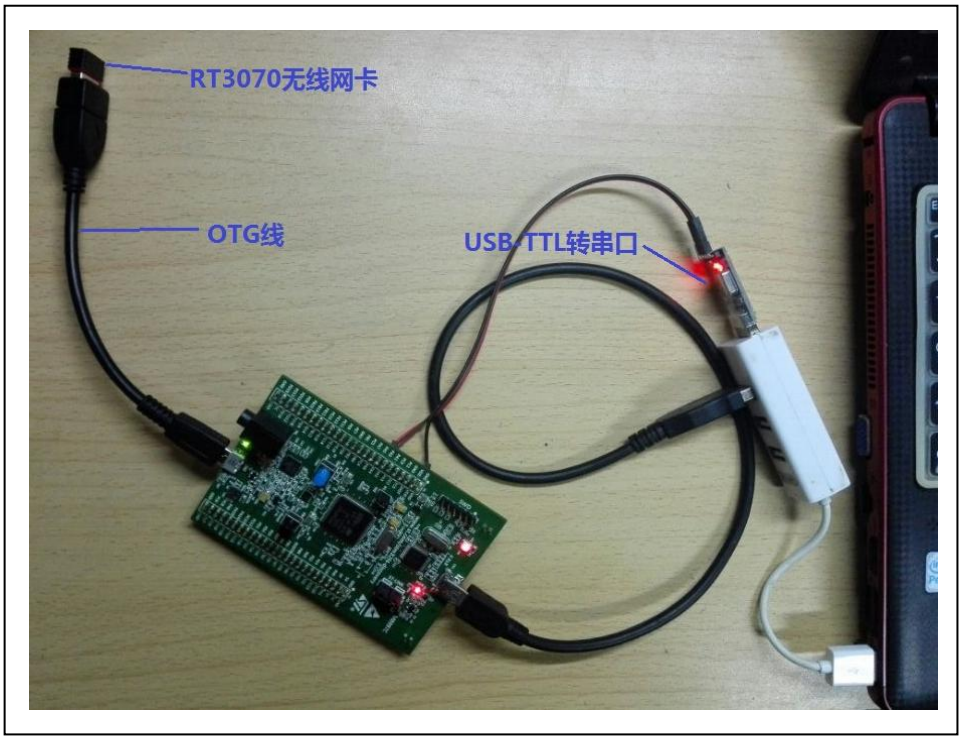
8 USB_Host_Linux_Convert. 18

2 Hardware Connection

Hardwares for sampled demo:

- STM32F4-Discovery Board.
- OTG cable.
- RT3070WirelessNetworkCard.
- Wirelessrouter
- USB-TTL

Fig. 1 STM32F4-Discovery Hardware Connection



Procedure

1. Connect USB-TTL to the USART2(PA2 and PA3) lied in STM32F4-Discovery board.
2. Connect STM32F4-Discovery board to RT3070 Wireless Network Card through a OTG cable.
3. Connect STM32F4-Discovery board to PC through a USB cable(type A to mini-B).

3 Sampled Demo

3.1 Preparation

Requirement for Sampled Demo.

- Keil MDK V4.x
- STLINK V2
- SecureCRT (115200 baudrate, 8 data bits and no parity)
- iPerf for Windows

3.2 Configuration

The default configuration of WirelessRouter:

SSID:mytest

Password: 12345678

Authentication type:WPA2PSK

Encryption type:AES

If your wireless router default configuration is not the same with above, you should modify it in the following steps. For more information in detail, refer to *See also: Wireless Network Configuration*

1. Edit EvalBoards\ST\STM32F4-Discovery\uCOS-III\RT2870STA.dat
2. Modify AuthMode key, fill in the following values(avoid spaces) according to your wireless router authentication type.

   OPEN open system

   SHARED WEP

   WPAPSK WPA pre-shared key.

   WPA2PSK WPA2 pre-shared key.
3. Modify EncrypType key, fill in the following values(avoid spaces) according to your wireless router Encryption type.

   NONE if AuthMode=OPEN

   WEP if AuthMode=SHARED

   TKIP if AuthMode=WPAPSK or WPA2PSK

   AES if AuthMode=WPAPSK or WPA2PSK
4. Modify WPAPSK key or Key1Str key, fill in your wireless router cipher.

   If AuthMode=SHARED, set cipher in Key1Str.

If AuthMode=WPAPSK or WPA2PSK, set cipher in WPAPSK.

5. Modify SSID, fill in your wireless router SSID.
6. Run EvalBoards\ST\STM32F4-Discovery\uCOS-III\ RT2870STA_to_c.exe，it will convert RT2870STA.dat to RT2870STA.c which will be compiled in compiling.
7. Run EvalBoards\ST\STM32F4-Discovery\uCOS-III\KeilMDK\ uCOS-III.uvproj，compile and download programs.
8. More configurations and details related to RT2870STA.dat, refer to Ralink-WIFI\DPO_RT5572_LinuxSTA_2.6.1.3_20121022_highly_optimize\ README_STA_usb.

3.3 Iperf Network Performance Test

After programs have been downloaded, the software will try to connect the SSID configured in RT2870STA.dat. Once connection established, it will start DHCP to get IP address and output the following information in SecureCRT. We assume the IP address obtained by STM32F4xx is 192.168.1.100 and IP address obtained by PC is 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 IP:192.168.1.100
DHCP GW:192.168.1.1
DHCP MASK:255.255.255.0
```

iPerf TCP client Test

Test time is 10 seconds,STM32F4xx reports results once every second.

Under Windows system, input following command through cmd.exe terminal:

```
iperf.exe -s
```

Under STM32F4xx, input following command through SecureCRT:

```
>iperf -c 192.168.1.101 -t 10 -i 1
------------------------------------------------------------
Client connecting to 192.168.1.101, TCP port 5001
TCP window size: 2.85 KByte (default)
------------------------------------------------------------
[  0]  0.0- 1.0 sec   324 KBytes  2.65 Mbits/sec
[  0]  1.0- 2.0 sec   340 KBytes  2.79 Mbits/sec
[  0]  2.0- 3.0 sec   364 KBytes  2.98 Mbits/sec
[  0]  3.0- 4.0 sec   366 KBytes  3.00 Mbits/sec
[  0]  4.0- 5.0 sec   376 KBytes  3.08 Mbits/sec
[  0]  5.0- 6.0 sec   372 KBytes  3.05 Mbits/sec
[  0]  6.0- 7.0 sec   366 KBytes  3.00 Mbits/sec
[  0]  7.0- 8.0 sec   372 KBytes  3.05 Mbits/sec
[  0]  8.0- 9.0 sec   376 KBytes  3.08 Mbits/sec
[  0]  9.0-10.0 sec   372 KBytes  3.05 Mbits/sec
[  0]  0.0-10.0 sec  3.54 MBytes  2.97 Mbits/sec
```

iPerf TCP server Test

Test time is 15 seconds, STM32F4xx report results once every second.

Under STM32F4xx, input the following command through SecureCRT line:

```
>iperf -s -i 1
Spawning a listener.
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 2.85 KByte (default)
------------------------------------------------------------
```

Under Windows system, input following command through cmd.exe terminal:

```
iperf.exe -c 192.168.1.100 -t 15
```

STM32F4xx outputs:

```
Spawning a server.
[  1] local 192.168.1.100 port 5001 connected with 192.168.1.101 port 35670
[  1]  0.0- 1.0 sec   256 KBytes  2.10 Mbits/sec
[  1]  1.0- 2.0 sec   256 KBytes  2.10 Mbits/sec
[  1]  2.0- 3.0 sec   256 KBytes  2.10 Mbits/sec
[  1]  3.0- 4.0 sec   256 KBytes  2.10 Mbits/sec
[  1]  4.0- 5.0 sec   256 KBytes  2.10 Mbits/sec
[  1]  5.0- 6.0 sec   238 KBytes  1.95 Mbits/sec
[  1]  6.0- 7.0 sec   235 KBytes  1.92 Mbits/sec
[  1]  7.0- 8.0 sec   236 KBytes  1.93 Mbits/sec
[  1]  8.0- 9.0 sec   245 KBytes  2.00 Mbits/sec
[  1]  9.0-10.0 sec   243 KBytes  1.99 Mbits/sec
[  1] 10.0-11.0 sec   233 KBytes  1.91 Mbits/sec
[  1] 11.0-12.0 sec   235 KBytes  1.92 Mbits/sec
[  1] 12.0-13.0 sec   256 KBytes  2.10 Mbits/sec
[  1] 13.0-14.0 sec   256 KBytes  2.10 Mbits/sec
[  1] 14.0-15.0 sec   256 KBytes  2.10 Mbits/sec
```

4 Wireless Network Configuration

Wireless network can be configured in three ways

- Configure through modifying RT2870STA.dat.
- Configure through iwpriv.
- Configure through calling wireless_exec_cmd().

4.1 Configure through modifying RT2870STA.dat

The configuration in RT2870STA.dat is default configuration。You can modify it according to your wireless router configuration.

1. Edit EvalBoards\ST\STM32F4-Discovery\uCOS-III\RT2870STA.dat
2. Modify AuthMode key, fill in the following values(avoid spaces) according to your wireless router authentication type.
   OPEN open system
   SHARED WEP
   WPAPSK WPA pre-shared key.
   WPA2PSK WPA2 pre-shared key.
3. Modify EncrypType key, fill in the following values(avoid spaces) according to your wireless router Encryption type.
   NONE if AuthMode=OPEN
   WEP if AuthMode=SHARED
   TKIP if AuthMode=WPAPSK or WPA2PSK
   AES if AuthMode=WPAPSK or WPA2PSK
4. Modify WPAPSK key or Key1Str key, fill in your wireless router cipher.
   If AuthMode=SHARED, set cipher in Key1Str.
   If AuthMode=WPAPSK or WPA2PSK, set cipher in WPAPSK.
5. Modify SSID, fill in your wireless router SSID.
6. Run EvalBoards\ST\STM32F4-Discovery\uCOS-III\ RT2870STA_to_c.exe，it will convert RT2870STA.dat to RT2870STA.c which will be compiled in compiling.
7. Run EvalBoards\ST\STM32F4-Discovery\uCOS-III\KeilMDK\ uCOS-III.uvproj，compile and download programs.
8. More configurations and details related to RT2870STA.dat, refer to

Ralink-WIFI\DPO_RT5572_LinuxSTA_2.6.1.3_20121022_highly_optimize\README_STA_usb.

4.2 Configuration Through iwpriv

Input iwpriv command in serial terminal.

Usage: iwpriv set [parameter]=[value].

Main configuration parameters:

Parameter	Value		Remarks
SSID	SSID name		
AuthMode	OPEN	open system	
	SHARED	wep	
	WPAPSK	WPA pre-shared key	
	WPA2PSK	WPA2 pre-shared key	
EncrypType	NONE	no encryption	if AuthMode=OPEN
	WEP	wired equivalent privacy	if AuthMode=SHARED
	TKIP	temporal key integrity protocol	if AuthMode=WPAPSK or WPA2PSK
	AES	Advanced encryption standard	if AuthMode=WPAPSK or WPA2PSK
WPAPSK	Pre-shared Key(8-63ASCII)		if AuthMode=WPAPSK or WPA2PSK
KEY1	WEP key(5 ASCII or 13 ASCII)		If AuthMode=SHARED

The table below may help you understand the above relations:

Parameter	AuthMode	EncrypType	WPAPSK	KEY1
Value	OPEN	NONE		
	SHARED	WEP		set key
	WPAPSK	TKIP	set key	
		AES	set key	

	WPA2PSK	TKIP	set key	
		AES	set key	

Examples：

a) Assume the wireless router authentication type is OPEN, encryption type is NONE, SSID is mytest.

   iwpriv set AuthMode=OPEN

   iwpriv set EncrypType=NONE

   iwpriv set SSID=mytest

b) Assume the wireless router authentication type is SHARED, encryption type is WEP, SSID is mytest, WEP key is 12345.

   iwpriv set AuthMode=SHARED

   iwpriv set EncrypType=WEP

   iwpriv set Key1=12345

   iwpriv set SSID=mytest

c) Assume the wireless router authentication type is WPA2PSK, encryption type is TKIP, SSID is mytest, pre-shared Key is 12345678.

   iwpriv set AuthMode= WPA2PSK

   iwpriv set EncrypType= TKIP

   iwpriv set WPAPSK=12345678

   iwpriv set SSID=mytest

d) Assume the wireless router authentication type is WPA2PSK, encryption type is AES, SSID is mytest, pre-shared Key is 12345678.

   iwpriv set AuthMode= WPA2PSK

   iwpriv set EncrypType= AES

   iwpriv set WPAPSK=12345678

   iwpriv set SSID=mytest

e) Assume the wireless router authentication type is WPAPSK, encryption type is AES, SSID is mytest, pre-shared Key is 12345678.

   iwpriv set AuthMode= WPAPSK

   iwpriv set EncrypType= AES

   iwpriv set WPAPSK=12345678

   iwpriv set SSID=mytest

More usages related to iwpriv tool, refer to:

Ralink-WIFI\DPO_RT5572_LinuxSTA_2.6.1.3_20121022_highly_optimize\

iwpriv_usage.txt

4.3 Configuration Through wireless_exec_cmd( )

Header File：wlan.h

Function：int wireless_exec_cmd(char *cmd).

wireless_exec_cmd() is used to income iwpriv command line, its usage is the same as chapter 4.2.

Examples:

a) Assume the wireless router authentication type is WPA2PSK, encryption type is AES, SSID is mytest, pre-shared Key is 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 Usage Restrictions of iperf

STM32F4xx iPerf is ported from the iPerf in Linux. It has following restrictions.

- Currently STM32F4xx iPerf only has a single thread.
- If STM32F4xx iPerf runs in the client mode, if it cannot connect to the server,STM32F4xx iPerf will abort, unless reset.
- Some iPerf parameters cannot change like TCP window size and max segment size.
- If STM32F4xx iPerf is in server mode, there is no key can come back to the command terminal, unless reset.
- If iPerf runs in the server mode, parameters can only be changed through reset.
- If STM32F4xx iPerf runs in the client mode, -P and -d are not available for STM32F4xx iPerf only has a single thread.

6 Usage of RAM

STM32F4xx has 192KB RAM, 128KB kept in 0x20000000 area, the other 64KB kept in 0x10000000area. Usages are shown in table.

Memory Usage of STM32F4xx

MemoryAddress	Size	Memory has been used
0x20000000	128KB	73.8KB
0x10000000	64KB	59.1KB

More detailed usages related to RAM refer to:
EvalBoards\ST\STM32F4-Discovery\uCOS-III\KeilMDK\Lst\uCOS-III.map

6.1 Heap Allocation

STM32F4xx has two areas including 0x20000000 and 0x10000000. Given it is inconvenient to manage two discrete heap allocations in the same time. We define some variables that are relatively large storage allocation as global variables in 0x10000000 area(approximately taken 59.1k RAM).

We use the lwip memory allocation function for the storage allocation of wireless driver. Wireless driver and lwip use the same heap whose size is 26KB. As wireless driver only takes up a large memory neared 17 KB in the initialization and destruction process(USB HotPlug), it just takes up a small memory neared 7.2KB in the TCP or UDP data transfer process. Thus we design wireless driver and lwip to use the same heap in order to save memory. If you want to use different heaps for allocation, cancel the macro definition of USE_LWIP_MALLOC in memory.h and add the lwip\mem.c to Keil MDK project.

6.2 Thread Stack Allocation

Thread stack takes up most of the memory. The software establishes 12 tasks in total

Usage of Thread Stack

Thread Name	Stack Size(Byte)	Remarks
RtmpCmdQTask	2048	Task of staring a USB wireless driver
RtmpMlmeTask	2048	Task of staring a USB wireless driver
RtmpTimerTask	2048	Task of staring a USB wireless driver
USBH Probe Task	1024	USB host search task. Used for HotPlug
USBH_Task	1024	USB host scheduling task
Tasklet Action	1536	Tasklet under Linux
TCP/IP	2048	Task of starting lwip
App Task Start	2048	The first startup program task. Used for process Shell command
uC/OS-III Timer Task	512	uC/OS-III kernel timer task
uC/OS-III Stat Task	512	uC/OS-III kernel stat task
uC/OS-III Tick Task	512	uC/OS-III kernel tick timer task
uC/OS-III Idle Task	512	uC/OS-III kernel idle task

7 How to port from DPO_RT5572_LinuxSTA_2.6.1.3_20121022

7.1 Obtain Compiler Parameters - D Macro Definition

1. EditDPO_RT5572_LinuxSTA_2.6.1.3_20121022\Makefile.
2. Add following information on command line 387：
   @echo $(WFLAGS) > define.txt.
3. Compile DPO_RT5572_LinuxSTA_2.6.1.3_20121022 source code under Linux.
4. Get define.txt after compiling.
5. Add macro-DVENDOR_FEATURE2_SUPPORT, it will help you find out if the porting process has causes a memory leak in time.
6. Add macro-DMEMORY_OPTIMIZATION, this macro can reduce the utilization of memory.
7. Delete macro-DIQ_CAL_SUPPORT for it often output bothering information during the debugging.

7.2 Optimize The Size of Structure

7.2.1 Optimize RTMP_ADAPTER Structure

RTMP_ADAPTER structure takes up most spaces in all structures, but it is the most important structure as most driver functions will use it. The sequence of optimization is shown in the table.

1	Modify MAX_REORDERING_MPDU_NUM lied in Rtmp_def.h from 256 to 4	Memory reduces about 300KB
2	Modify MAX_NUMBER_OF_MAC lied in Oid.h from 32 to 2	Memory reduces about 60KB
3	Modify MAX_NUMBER_OF_ACL lied in Oid.h from 64 to 2	Memory reduces about 60KB
4	Modify HASH_TABLE_SIZElied in Rtmp_def.h from 256 to 1	Memory reduces about 1KB
5	Modify MGMT_RING_SIZE lied in Rtmp_def.h from 32 to 1	Memory reduces about 1KB
6	Modify TX_RING_SIZE lied in Rtmp_def.h from 8 to 2	Memory does not reduce
7	Modify RIO_RING_SIZE lied in Rtmp_def.h from 8 to 2	Memory does not reduce
8	Modify RX_RING_SIZE lied in Rtmp_def.h from 8 to 1	Memory reduces about 360KB
9	Modify MAX_LEN_OF_MLME_QUEUE lied in Mlme.h from 40	Memory reduces about 800KB

to 20

RTMP_ADAPTER structure takes up 33792KB after Optimization.

7.2.2 Optimize HTTX_BUFFER Structure

HTTX_BUFFER takes up 122884KB before optimization.

Modify BULKAGGRE_SIZE lied in Rt_linux.h from 60 to 7. The structure takes up 14340KB after optimization.

7.2.3 Optimize RX_CONTEXT Structure

RX_CONTEXT->TransferBuffer allocates 24KB before optimization.

Modify RXBULKAGGRE_SIZE lied in Rtmp_usb.h from 12 to 2. The structure takes up 4KB after optimization.

7.2.4 Optimize MLME_QUEUE_ELEM Structure

Other threads will fill the messages to MLME_QUEUE_ELEM->Msg[MGMT_DMA_BUFFER_SIZE] when they send messages to RtmpMlmeTask. The structure size is fixed so it takes up large memory. We need to change fixed memory to dynamic memory to reduce memory usage.

1. Modify MLME_QUEUE_ELEM->Msg[] array to MLME_QUEUE_ELEM->Msg pointer.
2. Allocate memory for MLME_QUEUE_ELEM->Msg pointer.
3. Free the momery taken up by MLME_QUEUE_ELEM->Msg in RtmpMlmeTask thread.

This optimization is valid only if the macro MLMEQUEUE_ALLOC_INUSE is defined. You can search the macro in projects to find which code is modified.

7.3 Dynamic Allocation->Global Static Allocation

Some structures take up a lot of memory, we allocate the memory of these structures in global static at 0x10000000 ram area.

1. Modify the instance of RTMP_ADAPTER structure dynamic allocated by AdapterBlockAllocateMemory() function from dynamic allocation to global static allocation in rt_linux.c.
2. Modify the buffer of RTMP_ADAPTER->pBeaconBuf dynamic allocated by

RTMPAllocAdapterBlock() function from dynamic allocation to global static allocation in rtmp_init.c.

3. Modify the buffer of RTMP_ADAPTER->UsbVendorReqBuf dynamic allocated by RtmpRaDevCtrlInit() function from dynamic allocation to global static allocation in rtmp_init.c.
4. Modify the 802.11 data frame transfer buffer of RTMP_ADAPTER->TxContext[NUM_OF_TX_RING]->TransferBuffer dynamic allocated by NICInitTransmit() function from dynamic allocation to global static allocation in mlme.c. Besides, we find usb wireless driver only use the transfer buffer of RTMP_ADAPTER->TxContext[AC0], so we only define one TxContext structure.
5. Modify the 802.11 null frame transfer buffer of RTMP_ADAPTER-> NullContext[2]->TransferBuffer dynamic allocated by NICInitTransmit() function from dynamic allocation to global static allocation in mlme.c. Besides, we find usb wireless driver only use the transfer buffer of RTMP_ADAPTER->NullContext[0], so we only define one NullContext structure.
6. Modify the 802.11 ps-poll frame transfer buffer of RTMP_ADAPTER-> PsPollContext->TransferBuffer dynamic allocated by NICInitTransmit() function from dynamic allocation to global static allocation in mlme.c.

This optimization is valid only if the macro HIGHLY_OPTIMIZE is defined. You can search the macro in projects to find which code is modified.。

7.4Other optimization

1. Modify TIMER_QUEUE_SIZE_MAX lied in rtmp_timer.h from 128 to 32. We find usb wireless driver cannot use so many timers.
2. Some constant variables don’t have the const prefix that will lead to take up space both ROM and RAM. We prefix const to these constant variables to not take up RAM space.
3. Set WirelessEvent=1 in RT2870STA.dat, this will let us receive wireless event(eg: scan started or completed, connection established or lost) in the wireless_send_event() function.
4. In the MiniportMMRequest() function and line 88:

```
UCHAR rtmpHwHdr[TXINFO_SIZE +    pAd->chipCap.TXWISize];
```

The size of rtmpHwHdr variable is not a constant that will cause compiler use malloc() function to allocate memory instead of stack. After ensuring pAd->chipCap.TXWISize will not exceed 32.We modify the code as follows:

```
UCHAR rtmpHwHdr[TXINFO_SIZE + 32];
```

8 USB_Host_Linux_Convert

Given DPO_RT5572_LinuxSTA_2.6.1.3_20121022 driver is designed for Linux system. To reduce the difficulty in porting, we implement the Linux kernel interface for uC/OS-III embedded system and put them in the USB_Host_Linux_Convert\ Directory.

Linux Kernel Interface

Category	Header File	Interface
USB host	usbh_linux.h	usb_submit_urb() usb_kill_urb() usb_unlink_urb() usb_control_msg() usb_bulk_msg() remark: we currently only implement control transfer and block transfer.
Thread	kthread.h	kill_pid() kernel_thread()
complete	complete.h	init_completion() complete() complete_and_exit() wait_for_completion_timeout() wait_for_completion()
Spinlock	spinlock.h	spin_lock_init() spin_lock() spin_unlock() spin_lock_irqsave() spin_unlock_irqrestore()
Semaphore	sem.h	sema_init() down_interruptible() up() down_trylock() sema_destroy()
Timer	timer.h	mod_timer() init_timer() add_timer()

		timer_pending() del_timer()