Socket Wake-up

# Introduction

This application note describes the basics of sleep management in InnoOS™ with an example illustrating Talaria TWO wake-up from sleep mode.

# System Sleep Enable & Disable

## System Sleep APIs

### os\_suspend\_enable()

Suspends the system when idle.

|  |
| --- |
| void os\_suspend\_enable(void) |

Calling os\_suspend\_enable() will suspend the system or enable deep sleep, when the processor is idle. Enabling and disabling suspend mode takes additional time, which will affect the real-time response of the system. When an interrupt occurs, the system will wake up even if it is in a suspended state. However, the latency will be more as compared to when the system operates in a non-suspended mode.

### os\_suspend\_disable()

Disables system suspend.

|  |
| --- |
| void os\_suspend\_disable(void) |

When the system is idle, the kernel will place the CPU in low-power mode, ready to swiftly resume execution if an interrupt occurs.

### os\_avail\_heap()

Returns size of the remaining heap.

|  |
| --- |
| size\_t os\_avail\_heap(void) |

This is the total size including internal overhead. However, this may not represent the amount that can be allocated by an application.

# Code Walkthrough

## Sock\_wake.c

### Overview

The sample code is the path: examples/socket\_wakeup/src/sock\_wake.c is a simple application which demonstrates sleep mode.

### Sample Code Walkthrough

While programming the ELF to Talaria TWO using boot.py, bootargs is used to pass the SSID and Passphrase parameters to the program:

|  |
| --- |
| ssid = os\_get\_boot\_arg\_str("ssid");  passphrase = os\_get\_boot\_arg\_str("passphrase") ?: ""; |

### Connect to a Wi-Fi network

To connect to a Wi-Fi network the following APIs from the Wi-Fi Connection Manager are used:

1. wcm\_create()

This function creates the Wi-Fi network interface using the wcm\_handle pointer.

1. wcm\_notify\_enable()

Enables callbacks of the link and IP address changes.

1. wcm\_add\_network\_profile ()

Asynchronously adds a Wi-Fi network to connect. Currently only one network can be added.

1. wcm\_auto\_connect ()

Enables start or stop auto connection of the device with Wi-Fi.

|  |
| --- |
| my\_wcm\_handle = wcm\_create(NULL);  if(my\_wcm\_handle != NULL)  {  wcm\_notify\_enable(my\_wcm\_handle, my\_wcm\_notify\_cb, NULL);  if (np\_conf\_path != NULL) {  /\* Create a Network Profile from a configuration file in  \*the file system\*/  rval = network\_profile\_new\_from\_file\_system(&profile, np\_conf\_path);  } else {  /\* Create a Network Profile using BOOT ARGS\*/  rval = network\_profile\_new\_from\_boot\_args(&profile);  }  if (rval < 0) {  pr\_err("could not create network profile %d\n", rval);  return 0;  }  rval = wcm\_add\_network\_profile(my\_wcm\_handle, profile);  if (rval < 0) {  pr\_err("could not associate network profile to wcm %d\n", rval);  return 0;  }  wcm\_cnt = wcm\_auto\_connect(my\_wcm\_handle, 1);  os\_sem\_wait(&connect\_lock);  os\_sleep\_us(1000000, OS\_TIMEOUT\_WAKEUP);  if (wcm\_cnt == WCM\_SUCCESS) {  auot\_sucss = 1; } |

### Wi-Fi Connection Callback Function

The app\_wcm\_notify\_cb() function enables the callbacks of link and IP address.

|  |
| --- |
| switch(msg->msg\_type) {  case(WCM\_NOTIFY\_MSG\_LINK\_UP):  break;  case(WCM\_NOTIFY\_MSG\_LINK\_DOWN):  os\_sem\_post(&app\_wcm\_lock);  break;  case(WCM\_NOTIFY\_MSG\_ADDRESS):  break;  case WCM\_NOTIFY\_MSG\_CONNECTED:  wcm\_connect\_success = 1;  os\_sem\_post(&app\_wcm\_lock);  break;  }  os\_msg\_release(msg); |

### Server Socket Function

On successfully connecting to the Wi-Fi, the device creates a UDP server in Talaria TWO with server port number 8000. The server will initially be in sleep mode. Here, we register a call- back function to wake up Talaria TWO from sleep mode.

|  |
| --- |
| static bool add\_server\_socket(struct hiosock \*hsock, struct hiosock\_uri  \*hs\_uri)  {  os\_printf("add\_server\_socket\n"); bool status = false;  hsock->conn = netconn\_new\_with\_callback(NETCONN\_UDP, sock\_event\_cb); os\_printf("conn created\n");  if(hsock->conn != NULL)  {  if (hs\_uri->tos != -1) {  conn->pcb.tcp->tos = hs\_uri->tos;  }  if (hsock->type == SOCK\_STREAM) { netconn\_set\_nonblocking(hsock->conn, 1);  }  IP\_SET\_TYPE\_VAL(hs\_uri->rem\_addr, IPADDR\_TYPE\_ANY);  if (netconn\_bind(hsock->conn, &hs\_uri->rem\_addr, hs\_uri->port) == ERR\_OK) {  if (hsock->type == SOCK\_STREAM) {  if (netconn\_listen(hsock->conn) == ERR\_OK) { status = true;  //rsp->status = 0;  //rsp->socket = hsock->fd; hsock->state = HIOSOCK\_LISTEN;  list\_add\_tail(&hsock\_tbl, &hsock->list);  }  } else {  status = true;  //rsp->status = 0;  //rsp->socket = hsock->fd; hsock->state = HIOSOCK\_RD\_WRT;  netconn\_set\_nonblocking(hsock->conn, 1); list\_add\_tail(&hsock\_tbl, &hsock->list);  }  }  if (status == false) {  //rsp->status = ERR\_VAL; netconn\_delete(hsock->conn);  os\_printf("server at %s failed\n", ipaddr\_ntoa(&hs\_uri-  >rem\_addr));  } else {  os\_printf("%s server at %s port %d\n",  (hsock->type == SOCK\_STREAM)? "tcp": "udp", ipaddr\_ntoa(&hs\_uri->rem\_addr), hs\_uri->port);  } } } |

### Socket Wake-up Function

The thread will be in sleep mode. If any message is sent to the UDP server from any client, Talaria TWO will wake up for 500ms and go back to sleep mode.

|  |
| --- |
| static void\* sock\_wake\_thread(void\* arg)  {  os\_printf("sock wake thread\n"); while(1)  {  os\_msleep(1000);  }  return NULL;  } |

The os\_avail\_heap() function reveals the amount of space available on the heap.

### Socket Event Callback Function

This is the callback function, where Talaria TWO is enabled from sleep and put back to sleep after a brief period of 500ms.

|  |
| --- |
| static void  sock\_event\_cb(struct netconn \*conn, enum netconn\_evt event, u16\_t len)  {  if (event == NETCONN\_EVT\_RCVPLUS/\* || event == NETCONN\_EVT\_SENDPLUS\*/){  //int evt = SOCK\_EVT\_RECVPLUS;  os\_printf("NETCONN\_EVT\_RCVPLUS\n");  os\_printf("Waking up\n");  os\_suspend\_disable();  os\_sleep\_us(500000, OS\_TIMEOUT\_WAKEUP);  os\_printf("sleeping\n");  os\_suspend\_enable();  }  } |

### Running the Application

Program sock\_wake.elf (sdk\_x.y\examples\socket\_wakeup\bin) using the Download tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down
   2. ELF Input: Load the sock\_wake.elf by clicking on Select ELF File.
   3. AP Options: Provide the SSID and Passphrase under AP Options to connect to an Access Point.
   4. Programming: Prog RAM or Prog Flash as per requirement.

For more details on using the Download tool, refer to the document: UG\_Download\_Tool.pdf (path: *sdk\_x.y/pc\_tools/Download\_Tool/doc*).

**Note**: x and y refer to the SDK release version. For example: sdk\_2.5/doc.

Launch the Hercules tool for Windows and provide the port number along with the IP address and send the data.

Graphical user interface, application

Description automatically generated

Figure : Hercules Tool - Data transfer

### Expected Output

sock\_wake.elf is created when compiling the code which provides the following console output when programmed to Talaria TWO.

|  |
| --- |
| UART:SNWWWWWAEBuild $Id: git-ba65998b7 $  np\_conf\_path=/data/nprofile.json ssid=InnoPhase passphrase=43083191  $App:git-73e7f910  SDK Ver: sdk\_2.5  Wake From Sock App  addr e0:69:3a:00:13:90  [2.077,676] CONNECT:00:5f:67:cd:c5:a6 Channel:11 rssi:-45 dBm  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_LINK\_UP  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_ADDRESS  [2.808,043] MYIP 192.168.0.102  [2.808,207] IPv6 [fe80::e269:3aff:fe00:1390]-link  ip:192.168.0.102  URI: udp://192.168.0.102:8000  hiosock\_alloc  add\_server\_socket  udp server at 192.168.0.102 port 8000  conn created  sock wake thread  NETCONN\_EVT\_RCVPLUS  Waking up  sleeping  NETCONN\_EVT\_RCVPLUS  Waking up  sleeping  NETCONN\_EVT\_RCVPLUS  Waking up  sleeping |