Table of Contents

1. Introduction................................................................................................................3
2. Features.....................................................................................................................4
3. Application Overview................................................................................................. 5
4. Application Description.............................................................................................. 6
   4.1. SmartConnect 6LoWPAN Directory Structure in ASF Application.................................6
   4.2. ATWILC1000 Host Driver Directory Structure...............................................................7
   4.3. Integration of ATWILC1000 Driver with IP64 Driver.........................................................8
   4.4. Hardware Pinmux Configuration for ATWILC1000.........................................................8
   4.5. ATWILC1000 and 6LoWPAN Integration Structure.......................................................9
   4.6. ATWILC1000 Initialization.............................................................................................9
   4.7. Firmware Download to ATWILC1000...........................................................................10
   4.8. ATWILC1000 Connection to AP..................................................................................10
   4.9. Managing Wi-Fi operations in Contiki OS.....................................................................10
   4.10. Handling Data Transfer in ATWILC1000...................................................................11
   4.11. Executing the BR Node Process..................................................................................11
   4.12. Joining Node with Border Router...............................................................................12
   4.13. Accessing Atmel Web Server.....................................................................................12
5. Executing ATWILC1000 and 6LoWPAN BR Application......................................... 13
   5.1. Board Setup.............................................................................................................13
   5.2. ATWILC1000 Initialization and Connection................................................................13
6. ATWILC1000 Firmware File Format Conversion.....................................................15
7. Revision History........................................................................................................16
1. **Introduction**

Atmel® SmartConnect ATWILC1000 is an IEEE® 802.11b/g/n link controller SoC for applications in the Internet-of-Things. It is an ideal add-on to existing powerful MCU/MPU solutions bringing Wi-Fi® and Ethernet interface capabilities through a UART-to-Wi-Fi or SPI-to-Wi-Fi interface. Integrating ATWILC1000 WLAN module into Atmel SmartConnect 6LoWPAN Software is a complete solution for IP-mesh connectivity over 802.15.4. The WLAN interface provides the gateway interface to the border router of the IP-mesh network.

This document describes.

- The Integration of ATWILC1000 WLAN driver to RF212B / RF233 6LoWPAN border router application.
- The Example application based on the border router with web browser provided in the ASF release.
- The Atmel Studio application project files.
2. **Features**

- Portable, low-power 6LoWPAN solution
- Support for ATMEL Cortex-M0 microcontrollers – SAM D21
- Support for ATMEL transceivers - AT86RF212B / AT86RF233B
- WLAN interface support - ATWILC1000 WLAN module
3. **Application Overview**

This application primarily demonstrates the integration of Wi-Fi interface using ATWILC1000 WLAN module. Wi-Fi module connects with desired access point (AP) to provide the gateway access to the 6LoWPAN border router application.

The ASF 6LoWPAN border (bridge device) router with Web browser application used with ATWILC1000 bare metal WLAN driver. The ATWILC1000 bare-metal driver is available in ASF example application. This application note integrates the 6LoWPAN border router application with ATWILC1000 ASF application based on SAM D21 MCU.
4. Application Description

This topic elaborates the software integration of 6LoWPAN border router application with ATWILC1000 bare metal WLAN driver for providing gateway interface. In this ATWILC1000 WLAN module replaces the ethernet interface in 6LoWPAN application border router application.

This software integration consists of,

- Integrating the ATWILC1000 Free RTOS driver in to Contiki OS driver
- Replacing the LwIP network stack from Contiki OS providing Micro-IP network stack
- Modifying required hardware configuration for ATWINC1000 WLAN module

4.1. SmartConnect 6LoWPAN Directory Structure in ASF Application

The SmartConnect 6LoWPAN driver is located in ./src/ASF/thirdparty/wireless/SmartConnect_6LoWPAN folder of Atmel Software Framework (ASF) code base explains directory structure in the SmartConnect 6LoWPAN module of ASF.

Table 4-1. 6LoWPAN Directory Structure

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.apps</td>
<td>This directory contains the application modules that are used in conjunction with core stack</td>
<td></td>
</tr>
<tr>
<td>.core/lib</td>
<td>This module contains management modules that are required by stack to use like memory management, list management, etc</td>
<td></td>
</tr>
<tr>
<td>.core/sys</td>
<td>This module contains system modules like timers, process, auto-start module, etc</td>
<td></td>
</tr>
<tr>
<td>.core/net</td>
<td>Contains the protocol stack</td>
<td></td>
</tr>
<tr>
<td>.dev</td>
<td>This folder contains transceiver related files</td>
<td></td>
</tr>
<tr>
<td>.examples</td>
<td>This folder contains example projects and related application files</td>
<td></td>
</tr>
<tr>
<td>.services</td>
<td>This folder contains the modules that are used in conjunction with stack</td>
<td></td>
</tr>
<tr>
<td>./ASF/common/components/wifi/wilc1000</td>
<td>ATWILC1000 WLAN driver module</td>
<td></td>
</tr>
<tr>
<td>./services/wilc1000</td>
<td>ATWILC1000 interface function to the 6LoWPAN IP64 stack</td>
<td></td>
</tr>
</tbody>
</table>
ATWILC1000 Host Driver Directory Structure

ATWILC1000 WLAN driver is located in `.:/src/ASF/common/components/wifi/wilc1000` folder of Atmel Software Framework (ASF) code base. Explaining the directory structure of ATWILC1000 driver with respect to the SAM D21 Cortex-M0+ MCU.

**Table 4-2. ATWILC1000 Host Driver Directory Structure**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.bsp</code></td>
<td>This directory contains initialization of SAM D21 peripherals such as GPIO, Interrupt and Timer control API for WLAN module.</td>
<td></td>
</tr>
<tr>
<td><code>.bus_wrapper</code></td>
<td>This folder contains communication interface between the ATWILC1000 and host MCU initialization and data transfer API.</td>
<td></td>
</tr>
<tr>
<td><code>.common</code></td>
<td>This folder contains wrapper functions for C library and string API.</td>
<td></td>
</tr>
<tr>
<td><code>.driver</code></td>
<td>This module contains the ATWILC1000 WLAN host MCU driver API. This includes the HIF(host interface), WLAN module ASIC, Wi-Fi connection and configuration API's.</td>
<td></td>
</tr>
<tr>
<td><code>.image_storage</code></td>
<td>This folder contains the files system access function from storage memory (SD/MMC card). These functions uses the FATFS API.</td>
<td></td>
</tr>
</tbody>
</table>
4.3. Integration of ATWILC1000 Driver with IP64 Driver

This folder `/src/ASF/thirdparty/wireless/SmartConnect_6LoWPAN/apps/ip64` contains the interface layer between ATWINC1000 WLAN driver and ip64 driver. The ip64 driver converts the IPv4 format packets from IPv6 packet generated in the 6LoWPAN driver. This Interface layer handles the Web browser data transfer activity through this interface.

- ./ip64-wifi-interface.c
- ./ip64-wifi-interface.h
- ./ip64-wifi.c
- ./ip64-wifi.h

Figure 4-3. Wi-Fi Integration with IP64 Driver

4.4. Hardware Pinmux Configuration for ATWILC1000

To connect the ATWILC1000 Xplained PRO board on EXT1 header of SAM D21 Xplained Pro board, SERCOM SPI interface, and three GPIO’s for RESET, CHIP_EN, and WAKEUP pins are required. Apart from that to handle the ATWILC1000 interrupt services, one of the external IRQ pin to be configured. This pinmux configuration are available in the `/src/config/conf_wilc.h`.

```c
#define CONF_WILC_PIN_RESET             PIN_PB06
#define CONF_WILC_PIN_CHIP_ENABLE    PIN_PB05
#define CONF_WILC_PIN_WAKE        PIN_PB07
/** SPI pin and instance settings. */
#define CONF_WILC_SPI_MODULE         EXT1_SPI_MODULE
#define CONF_WILC_SPI_SERCOM_MUX     EXT1_SPI_SERCOM_MUX_SETTING
#define CONF_WILC_SPI_PINMUX_PAD0    EXT1_SPI_SERCOM_PINMUX_PAD0 /* in */
#define CONF_WILC_SPI_PINMUX_PAD1    PINMUX_UNUSED /* cs driven from software */
#define CONF_WILC_SPI_PINMUX_PAD2    EXT1_SPI_SERCOM_PINMUX_PAD2 /* out */
#define CONF_WILC_SPI_PINMUX_PAD3    EXT1_SPI_SERCOM_PINMUX_PAD3 /* sck */
#define CONF_WILC_SPI_CS_PIN         EXT1_PIN_SPI_SS_0
/** SPI interrupt pin. */
#define CONF_WILC_SPI_INT_PIN        EXT1_IRQ_PIN
#define CONF_WILC_SPI_INT_MUX        EXT1_IRQ_MUX
#define CONF_WILC_SPI_INT_EIC       (4)
```
To connect the IO XPRO board on EXT3 header of the SAM D21 board, required pinmux configurations are mentioned. This pinmux configuration are available in the ./src/config/conf_sd_mmc.h

```c
#define SD_MMC_SPI                        EXT3_SPI_MODULE
#define SD_MMC_SPI_PINMUX_SETTING      EXT3_SPI_SERCOM_MUX_SETTING
#define SD_MMC_SPI_PINMUX_PAD0         EXT3_SPI_SERCOM_PINMUX_PAD0
#define SD_MMC_SPI_PINMUX_PAD1         EXT3_SPI_SERCOM_PINMUX_PAD1
#define SD_MMC_SPI_PINMUX_PAD2         EXT3_SPI_SERCOM_PINMUX_PAD2
#define SD_MMC_SPI_PINMUX_PAD3         EXT3_SPI_SERCOM_PINMUX_PAD3
#define SD_MMC_CS                      EXT3_PIN_15
#define SD_MMC_0_CD_GPIO               EXT3_PIN_10
```

4.5. **ATWILC1000 and 6LoWPAN Integration Structure**

In file ip64-conf.h IP64_CONF_WIFI_DRIVER and UIP_FALLBACK_INTERFACE macros are assigned to the structure wilc1000_ip64_driver.c. This structure variables are assigned to the ATWILC1000 initialization (Init) and data transfer function (Output) functions.

This wilc1000_ip64_driver structure is defined in the file wifi-ip64-driver.c file.

```c
const struct ip64_driver wilc1000_ip64_driver = {
    init, /* Dummy Initialization function */
    output /* Data transfer function (Tx)*/
};
```

4.6. **ATWILC1000 Initialization**

From main() function routine of 6LoWPAN border router application wifi_init() initiates the ATWILC1000 initialization process. To synchronize 6LoWPAN application process ip64 thread starts after the successful Wi-Fi connection.

1. Link node address configured in the 6LoWPAN initialization sequence is copied in to the EUI-64 (IEEE-defined 64-bit Extended Unique Identifier) address of the device.
2. Mangle the EUI-64 address in to 48-bit Wi-Fi module MAC address. Leaving the 3rd and 4th fields of the EUI-64 address combining first and last three bytes to form the MAC address.
3. ATWILC1000 WLAN module MAC address is replaced with this MAC address using m2m_wifi_set_mac_address() function after the WLAN module initialization.
4. The structure tstrWifiInitParam parameters filled with required parameters and passed to arguments of m2m_wifi_init().
   ```c
   /* Initialize Wi-Fi driver with data and status callbacks. */
   param.strEthInitParam.pfAppEthCb = winc1500_recv_cb;
   param.strEthInitParam.au8ethRcvBuf = ip64_packet_buffer;
   param.strEthInitParam.u16ethRcvBufSize = ip64_packet_buffer_maxlen;
   param.pfAppWifiCb = wifi_cb;
   ret = m2m_wifi_init(&param);
   if (M2M_SUCCESS != ret) {
       printf("main: m2m_wifi_init call error!(%d)\n", ret);
       while (1) {}  
   }
   
   m2m_wifi_init(6param);
   ```
5. The structure tstrEthInitParam is passed with 802.3 frame receive callback function wilc1000_recv_cb to the function pointer tpfAppEthCb.
6. Buffer pointer, au8ethRcvBuf and buffer size u16ethRcvBufSize variable is passed ip64_packet_buffer, and ip64_packet_buffer_maxlen respectively.
7. These variables are ip64 driver global variables to handle the received data’s directly.
8. The receive callback buffer function called whenever the data received from the WLAN module. Remaining length of the data to be received is also passed with the callback function.
9. The function pointer tpfAppWifiCb carried with Wi-Fi callback function notification wifi_cb().

10. The callback function triggered for,
- Wi-Fi connection status change notification with AP
- DHCP obtain IP address event
- RSSI value indication
- Scan request and results notification

4.7. Firmware Download to ATWILC1000

ATWILC1000 WLAN module requires a firmware to execute the 802.11 WLAN MAC functionalities. The firmware stored in the SD/MMC card as a hex file. The SD card connected to the SAMD21 MCU on EXT3 header. The firmware downloaded from the SD card through MCU to ATWILC1000 WLAN module as part of initialization. FATFS file system services is used to access the firmware.

ATWILC1000 link WLAN controller also requires a wpa_supplicant to handle the WLAN security. The Contiki OS provides the network stack (micro TCP/IP stack) and applications like DNS, DHCP, etc.

4.8. ATWILC1000 Connection to AP

To connect with the desired access point m2m_wifi_connect() function called with argument like SSID, security method, passphrase. This initiates the connection process to AP and wifi_cb() function called when connection process status success or failure.

The macros are defined in the wifi-ip64-driver.h file. User must modify the macros with appropriate AP parameters.

```c
/** Wi-Fi Settings */
#define MAIN_WLAN_SSID        "DEMO_AP" /* < Destination SSID / Access Point Name */
#define MAIN_WLAN_AUTH        M2M_WIFI_SEC_WPA_PSK /* < Security manner */
#define MAIN_WLAN_PSK        "1234567890" /* < Password of the Access Point*/
printf("Connecting to %s.\r\n", (char *)MAIN_WLAN_SSID);
/* Connect to defined AP. */
m2m_wifi_connect((char *)MAIN_WLAN_SSID, sizeof(MAIN_WLAN_SSID),
 MAIN_WLAN_AUTH, (void *)MAIN_WLAN_PSK, M2M_WIFI_CH_ALL);
```

**Note:** For more details about the AP connection procedure refer the station mode application notes.

4.9. Managing Wi-Fi operations in Contiki OS

In the gateway application, two OS tasks are handling the WLAN operations such as "WILC1000 IP64 driver" and "WILC1000 Events IP64 driver".

```c
PROCESS(wilc1000_ip64_driver_process, "WILC1000 IP64 driver");
PROCESS(WILC1000_Events_ip64_driver_process, "WILC1000 Events IP64 driver");
```

There are two Wi-Fi tasks running in this application such as wilc1000_ip64_driver_process and wilc1000_Events_ip64_driver_process. The task wilc1000_Events_ip64_driver_process starts before the Wi-Fi initialization process. The task used to service the ATWILC1000 external interrupt (ISR) events. The task triggered or woke up using post_processs(&wilc1000_Events_ip64_driver_process)when ever ISR ever occurred.

```c
PROCESS_THREAD(WILC1000_Events_ip64_driver_process, ev, data){
  PROCESS_BEGIN();
  while(1) {
    PROCESS_WAIT_EVENT();
    m2m_wifi_handle_events(NULL);
  }
}
```
External interrupts are triggered from the ATWILC1000 module, the data ready to received from the ATWILC1000 receive buffer and for event notifications. The task `wilc1000_ip64_driver_process` starts only after the Wi-Fi connection success notified in the callback function `wifi_cb()`. The data received from the ATWILC1000 passes to the ip64 driver to process the request or response from the Web browser. To receive the consecutive data from the ATWILC1000 `m2m_wifi_set_receive_buffer()` function registered with `ip64_packet_buffer` and `ip64_packet_buffer_maxlen` global buffer of ip64 driver.

```c
PROCESS_THREAD(wilc1000_ip64_driver_process, ev, data){
    PROCESS_BEGIN();
    while(1) {
        PROCESS_WAIT_EVENT();
        if(sz != 0){
            IP64_INPUT(ip64_packet_buffer, sz);
            sz = 0;
        }
        m2m_wifi_set_receive_buffer(ip64_packet_buffer,ip64_packet_buffer_maxlen);
    }
    PROCESS_END();
}
```

**Note:** The OS tasks are named as a process in Contiki OS.

### 4.10. Handling Data Transfer in ATWILC1000

The API `m2m_wifi_send_ethernet_pkt()` used to transfer the data to ATWILC1000 WLAN module firmware. The ip64 driver uses the API to transfer the 802.3 ethernet packet to ATWILC1000 module firmware.

```c
static int output(uint8_t *packet, uint16_t len){
    m2m_wifi_send_ethernet_pkt(packet, len);
    return len;
}
```

The API `wilc1000_recv_cb()` registered with ip64 driver buffer pointer and size to received in the ATWILC1000 IP layer driver callback function. Whenever the data is available in the IP layer, the callback function will return the buffer pointer and size of the buffer.

```c
void wilc1000_recv_cb(uint8 u8MsgType, void * pvMsg,void * pvCtrlBuf){
    tstrM2mIpCtrlBuf *ctrl = (tstrM2mIpCtrlBuf *)pvCtrlBuf;
    if (u8MsgType == M2M_WIFI_RESP_ETHERNET_RX_PACKET) {
        sz = ctrl->u16DataSize;
        rem = ctrl->u16RemainingDataSize;
    }
    process_post(&wilc1000_ip64_driver_process,WIFI_DATA_EVENT,NULL);
}
```

### 4.11. Executing the BR Node Process

The tasks `router_node_process`, `blinker_process`, and `network_reboot_process` are initiated from the Wi-Fi connection success callback function. These tasks are used to handle the ip64 driver and 6LoWPAN operations.

The task `router_node_process` initiates the ip64 driver initialization. The DHCP request to obtain the IP address initiated to the connected access point network DHCP server. The access point network provides the IP address for the requested WLAN client. The DHCP network application uses the micro IP TCP/IP stack of Contiki OS.
The task `blinker_process` used to manipulate the LED On/Off or Blink LED of the selected 6LoWPAN node in the 802.15.4 network. These operations are accessed using Web browser of the 6LoWPAN network.

**Figure 4-4. Web Browser Page**

![Web Browser Page](image)

The optional process `network_reboot_process` to handle the 6LoWPAN reboot process. This process is used depending upon the application usage.

### 4.12. Joining Node with Border Router

The 6LoWPAN network application consists of three 802.15.4 nodes. In this application AT86RF212B / AT86RF233 802.15.4 RF modules are supported. One of the device node starts as a border router and other two device nodes starts as end device. The router starts to advertise the beacon frames in the channel 1 of 802.15.4 network. Other 6LoWPAN end device nodes join with the border router using 6LoWPAN connection process.

**Note:** The 6LoWPAN connection process explained in the SmartConnect 6LoWPAN user guide part of the ASF application.

### 4.13. Accessing Atmel Web Server

The IPv4 address of the device used to browse the Web browser using any of the web browser application.

**Figure 4-5. Border Router Web Page**

![Border Router Web Page](image)

In this Web browser, user control the other 6LoWPAN node in the 802.15.4 network. Other details are explained in the 6LoWPAN ASF application document.
5. Executing ATWILC1000 and 6LoWPAN BR Application

5.1. Board Setup

This example demonstrates how to execute the ATWILC1000 WLAN module as gateway interface to the ATZB-RF-212B based 802.15.4 6LoWPAN network using the SAM D21 Xplained Pro board as host MCU. Following hardware are used in this example:

- The SAM D21 Xplained Pro
- The ATWILC1000 XPRO on EXT1 header
- The 802.11 b/g/n supported AP or router
- The ATZB-RF-212B-0-CN XPRO board on EXT2
- The IO Xplained Pro board with micro SD card on EXT3
- Platform is Atmel Studio 7 and ASF applications

Figure 5-1. Board Setup with ATWILC1000

Note: Currently ATWILC1000 XPRO board is not available. Use the ATWINC1500 Xplained Pro board instead of ATWILC1000 XPlained Pro board. Both are pin to pin compatible.

5.2. ATWILC1000 Initialization and Connection

Wi-Fi and storage initialization for mounting SD card with file system are called in the main function itself.

- Firmware file named, wilc1000_1000b.bin, provided as part of the application note software package. The file path is ASF\common\components\wifi\wilc1000\driver\include.
- Copy the firmware in to the microSD card and insert in to the IO Xplained pro board and connect it with EXT3 header of the SAM D21 Xplained Pro board.
- Connect the micro USB cable in to the EDBG port of SAM D21 Xplained Pro board with PC USB port.
- Open the serial port terminal application using COM port configuration 115200, 8, none, 1, none.
- Open the SAMD21_WILC1000_6LOWPAN_BR-WEBSERVER-NODE1 application in Atmel Studio 7 from the provided software package.
- Modify the access point (AP) SSID and security password in the file wifi-ip64-driver.h as mentioned in the ATWILC1000 Connection to AP.
- Compile and download the image into the SAM D21 Xplained Pro board.
- When 6LoWPAN application starts along with the ATWILC1000 WLAN initialization.
- SD card is initialized and mounted with FATFS file system to access the firmware.
• As part of the ATWILC1000 WLAN initialization sequence firmware downloaded to the WLAN module. This firmware download process takes some time to complete, due to SD card file access process.
• After completing the firmware download, WLAN connection process is initiated to the desired access point.
• The ip64 driver node process starts followed by the WLAN connection process success.
• The DHCP client request initiated through ip64 driver to connected AP network.
• DHCP server in the network provides the IP address to WLAN client.
• Simultaneously 802.15.4 RF212B starts the border router and become the route in the network.
• RF212B advertise the beacon frames in the channel 1 of the sub-GHz network.
• Any other 6LoWPAN node in the 802.15.4 network joins with the border router.
• Obtained IP address (XX.XX.XX.XX) is used to open the Web browser of the 6LoWPAN Border Router application as shown.
• Web browser application usage is explained in the 6LoWPAN user guide as part of the ASF application itself.
6. **ATWILC1000 Firmware File Format Conversion**

In this application, ATWILC1000 firmware used as binary format. Default ATWILC1000 bare metal driver application using the firmware as a hex array format. Attached the script in the application `/src/script` to convert from hex array format to binary format. Run the script in command line using the original firmware file.

```bash
command >> array_to_bin.py wifi_firmware_1000.h
```

Figure 6-1. Script Directory

The ATWILC1000 firmware file available in ATWILC1000 ASF application.

Figure 6-2. ATWILC1000 Firmware Directory in ASF Application
7. **Revision History**

<table>
<thead>
<tr>
<th>Doc Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42790A</td>
<td>10/2016</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>