Atmel SmartConnect

AT12181: ATWINC1500 Wi-Fi Network Controller - AP Provision Mode

APPLICATION NOTE

Introduction

This application note explains how to build the state-of-art Internet of Things (IoT) applications using the Wi-Fi® Access Point Provision Mode with the Atmel® ATWINC1500 Wi-Fi Network Controller.

The following topics are covered:

- Organization of demo application
- Information about target boards
- Flow of demo application
- Step-by-step execution of the API

Features

- ATWINC1500 host MCU driver architecture
- ATWINC1500 internal architecture
- Application description with code snippets
- Events handled in the Wi-Fi callback function with appropriate structures used for each events
- Steps to execute the AP provision mode application demo using SAM D21 Xplained Pro board and ATWINC1500

Table of Contents

1. Application Overview

What is Wi-Fi Provisioning?

Wi-Fi provisioning is the process of connecting a new Wi-Fi device (station) to a Wi-Fi network. The provisioning process involves loading the station with the network name (often referred to as SSID) and its security credentials.

What is SoftAP?

SoftAP is an abbreviated term for "Software enabled Access Point". This is a software that enables a computer that is not designed to be a router into a wireless access point. It is often used interchangeably with the term "virtual router".

What is Wi-Fi Station?

In IEEE® 802.11 (Wi-Fi) terminology, a station (STA) is a device that has the capability to use the 802.11 protocol. For example, a station may be a laptop, a desktop PC, PDA, access point, or Wi-Fi phone. A STA may be fixed, mobile, or portable. In wireless networking terminology a station is also called wireless client and node. It is referred as transmitter or receiver based on its transmission characteristics. IEEE 802.11-2007 formally defines station as: Any device that contains an IEEE 802.11-conformant Media Access Control (MAC) and physical layer (PHY) interface to the Wireless Medium (WM). WLAN station can operate in Infrastructure mode and Ad-Hoc or Peer to Peer mode.

What is WINC1500 AP Provisioning?

Wi-Fi AP Provision mode primarily demonstrates on how to configure the credentials (such as, SSID and Passphrase) in ATWINC1500 remotely. The configured credentials are used to connect with a desired access point. This demo uses the Android device with apps to configure the credentials and verify the connection using simple ping operation. For the ping operation, ping-free apps can be used.

Remote configuration facilities such as AP provisioning and HTTP provisioning modes are available. In HTTP provisioning mode, the HTTP page is used to configure the credentials. The HTTP server is running in the WINC firmware and the HTTP page is also stored in the WINC flash memory.

The demo provides the following capabilities:

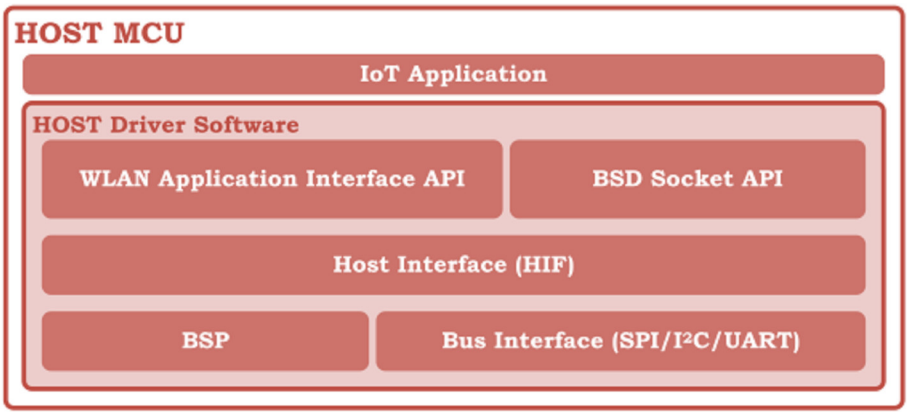
- To configure the credentials to ATWINC1500
- The ATWINC1500 Wi-Fi network will start as an access point and any Android device with WLAN device can be connected to it as a station
- After a successful connection using provision app, credentials are configured to the ATWINC1500
- The ATWINC1500 disables the AP mode and initialize the station mode. It starts scanning with given credentials to establish a connection with the desired access point
- The ATWINC1500 host application parses the scan results to determine the SSID and establish the connection with the desired access point

Figure 1-1. Sample Demo Setup

2. Host Driver Architecture

The ATWINC1500 host driver software is a C library. It provides the host MCU application with necessary APIs to perform WLAN and socket operations. It shows the architecture of the ATWINC1500 host driver software which runs on the host MCU. The components of the host driver are described in ATWINC1500 Wi-Fi Network Controller - Software Design Guide.

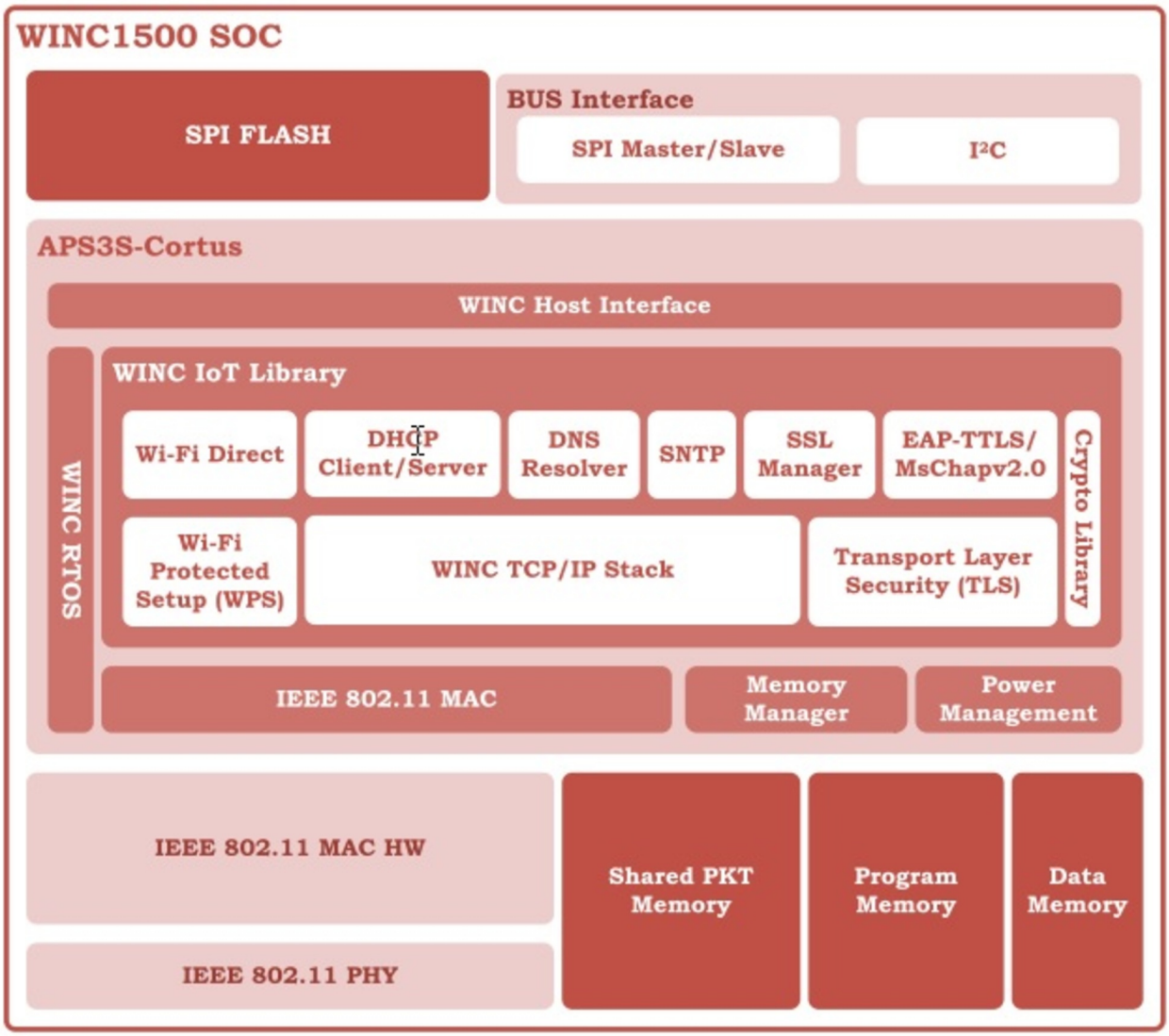
Figure 2-1. Host Driver Architecture



3. ATWINC1500 System Architecture

ATWINC1500 has a built-in Wi-Fi, IEEE®-802.11 physical layer and RF front end, and ASIC has an embedded APS3S-Cortus 32-bit CPU to run the ATWINC1500 firmware. The firmware comprises the Wi-Fi IEEE-802.11 MAC layer and embedded protocol stacks which offload the host MCU. The components of the system are described in the sub-sections of ATWINC1500 Wi-Fi Network Controller - Software Design Guide.

Figure 3-1. ATWINC System Architecture



4. Application Description

This section describes the ATWINC1500 host driver AP provision mode application in detail.

4.1. Wi-Fi Host Driver Initialization

- System Initialization of SAM D21 Xplained Pro board consists of MCU's clock initialize, hardware events and external hardware interfaces.

```
/* Initialize the board. */
    system_init();
```

- Configuration of console UART interface used for debug log output. Debug log level value can be set using `M2M_LOG_LEVEL` macro in the `nm_debug.h` file.

```
/* Initialize the UART console. */
    configure_console();
    printf(STRING_HEADER);
```

- The BSP driver initialization will follow the ATWINC1500 bring-up sequence. The sequence of chip enable and reset pin of ATWINC1500 is followed. Refer the `nm_bsp_init` API definition.

```
/* Initialize the BSP. */
    nm_bsp_init();
```

- The Wi-Fi host driver initialization starts with the API `m2m_wifi_init()` and the structure `tstrWifiInitParam`. The Wi-Fi initialization sequence configures the SPI communication interface and external interrupt with respect to the host MCU peripherals. Apart from this, the Wi-Fi host application layer callback function and the `wifi_cb()` function is registered during the initialization sequence.
- When the SPI interface initialization is completed, host MCU reads chip-id of ATWINC1500. Its indicates that, WLAN module is ready for initialization process which needs the module reset and waits for the confirmation from the module.
- The purpose of the Wi-Fi application callback function is to indicate the events such as connect and disconnect status. The function operates by obtaining the IP address from the DHCP server with respect to the DHCP request from the ATWINC1500 station. In SoftAP mode, DHCP server providing IP address to the connected WLAN client device will also notified by the event.

```
/* Initialize Wi-Fi parameters structure. */
    memset((uint8_t *)&param, 0, sizeof(tstrWifiInitParam));

    /* Initialize Wi-Fi driver with data and status callbacks. */
    param.pfAppWifiCb = wifi_cb;
    ret = m2m_wifi_init(&param);
    if (M2M_SUCCESS != ret) {
        printf("main: m2m_wifi_init call error!(%d)\r\n", ret);
        while (1) {
        }
    }
```

4.2. Socket Layer Callback Registration

During socket layer initialization, a general IP layer event callback and specific socket layer event handling callback function are registered.

General IP layer call back function is triggered whenever host application receives an event from the TCP/IP stack of the ATWINC1500 WLAN module. Corresponding event specific data is read from the

WLAN module. After reading the socket event data's from ATWINC1500, socket layer callback function is called.

While initializing the socket, the`sockaddr_in` structure is filled with *address family* belongs to IPv4 or IPv6, port number, and IP address.

```
/* Initialize socket address structure. */
    addr.sin_family = AF_INET;
    addr.sin_port = _htons((MAIN_WIFI_M2M_SERVER_PORT));
    addr.sin_addr.s_addr = 0;
    /* Initialize Socket module */
    socketInit();
    registerSocketCallback(socket_cb, NULL);
```

4.3. AP Mode Initialization

The `m2m_wifi_enable_ap` API enables the SoftAP mode in ATWINC1500 with the configurations such as SSID, Channel, Open security mode. IP address of the SoftAP is used for DHCP server configuration. When ATWINC1500 is in AP provision mode, Andoid device can be connected to configure the credentials using Android provisioning App by provided by Atmel. ATWINC1500 DHCP server provides IP address for the connecting WLAN device.

```
       /* Initialize AP mode parameters structure with SSID, channel and OPEN
security type. */
    memset(&strM2MAPConfig, 0x00, sizeof(tstrM2MAPConfig));
    strcpy((char *)&strM2MAPConfig.au8SSID, MAIN_WLAN_SSID);
    strM2MAPConfig.u8ListenChannel = MAIN_WLAN_CHANNEL;
    strM2MAPConfig.u8SecType = MAIN_WLAN_AUTH;
    strM2MAPConfig.au8DHCPServerIP[0] = 0xC0; /* 192 */
    strM2MAPConfig.au8DHCPServerIP[1] = 0xA8; /* 168 */
    strM2MAPConfig.au8DHCPServerIP[2] = 0x01; /* 1 */
    strM2MAPConfig.au8DHCPServerIP[3] = 0x01; /* 1 */

    /* Bring up AP mode with parameters structure. */
    ret = m2m_wifi_enable_ap(&strM2MAPConfig);
    if (M2M_SUCCESS != ret) {
        printf("main: m2m_wifi_enable_ap call error!\r\n");
        while (1) {
        }
    }
```

4.4. Socket Layer Initialization

To connect with Atmel provided Android application, the TCP server socket must be created and bound to ATWINC1500 AP's IP address and port number. IP address is used by the TCP client during the connection process. Android Apps is connected to TCP server socket of ATWINC1500 while starting the Android applition.

After connecting to the ATWINC1500 TCP server socket, AP's credentials can be entered and transferred to the ATWINC1500. When ATWINC1500 receives the credentials from Android apps, the AP mode is disabled using `m2m_wifi_disable_ap()`. Switch to the station mode to connect with matching AP using `m2m_wifi_connect()`.

The `socket()` function uses the arguments such as `AF_INET` to specify the IPv4 address format and `SOCK_STREAM` to specify the socket type as a TCP. Return value of the `socket()` is socket ID, which is used for further socket configuration and connection process.

The `bind()` function uses the structure `sockaddr_in` to specify the address family of the socket as a IPv4, port number, IPv4 network address. When INADDR_ANY is specified in the bind call, the socket is bound to all local interfaces.

```
   /* Initialize socket address structure. */
addr.sin_family = AF_INET;
addr.sin_port = _htons((MAIN_WIFI_M2M_SERVER_PORT));
addr.sin_addr.s_addr = 0;
if (tcp_server_socket < 0) {
/* Open TCP server socket */
if ((tcp_server_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("main: failed to create TCP server socket error!\r\n");
    continue;
}
/* Bind service*/
bind(tcp_server_socket, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
```

Note: The address and port number are always stored in network byte order format. To convert the host byte order to network byte order using `htons()`.

4.5. Wi-Fi Host Driver Event and Callback Handling

All the Wi-Fi host driver event is handled in the `m2m_wifi_handle_events()`. The HIF (Host communication Interface) layer API's is used to monitor the external interrupt which registered using host MCU configuration.

```
while (1) {
        /* Handle pending events from network controller. */
        while (m2m_wifi_handle_events(NULL) != M2M_SUCCESS) {
        }
    }
```

When ATWINC1500 external interrupt occurs, the host interface ISR layer reads the ATWINC1500 control register to identify the type of event which triggered the external interrupt. Depending on the event, if any data is available in the ATWINC1500, registered callback functions are called with appropriate data.

Host MCU Wi-Fi application driver handles the event with various categories such as:

- **`M2M_REQ_GRP_WIFI`**:`m2m_wifi_cb` handle all the Wi-Fi configuration and connection events.
- **`M2M_REQ_GRP_IP`**:`m2m_ip_cb` handle all the socket, and network application event callbacks.
- **`M2M_REQ_GRP_OTA`**:`m2m_ota_cb` handle all the *Over The Air* firmware upgrade events.

4.5.1. Wi-Fi Callback Function

Wi-Fi callback function is called depending on the success or failure state of the connection status and DHCP request confirmation.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
   switch (u8MsgType) {
   case M2M_WIFI_RESP_CON_STATE_CHANGED:{
    tstrM2mWifiStateChanged *pstrWifiState =
                (tstrM2mWifiStateChanged *)pvMsg;
    ..............................
   case M2M_WIFI_REQ_DHCP_CONF:{
    ..............................
}
```

Wi-Fi callback function is called in the various scenarios. List of events handled during the `wifi_cb()` function are provided in the following table.

Table 4-1. Wi-Fi Callback Events

Wi-Fi Callback Events	Structure used for the Events	Comments
M2M_WIFI_RESP_SCAN_DONE	tstrM2mScanDone	Scan complete notification response for requested Scan command
M2M_WIFI_RESP_SCAN_RESULT	tstrM2mWifiscanResult	Response for the requested Scan results command
M2M_WIFI_RESP_CON_STATE_CHANGED	tstrM2mWifiStateChanged	WLAN connection state whether station or SoftAP mode
M2M_WIFI_RESP_CURRENT_RSSI	char *	Response to M2M_WIFI_REQ_CURRENT_RSSI with the RSSI value
M2M_WIFI_RESP_CONN_INFO	tstrM2MConnInfo	Connected AP information response
M2M_WIFI_RESP_PROVISION_INFO	tstrM2MProvisionInfo	Received provisioning information from the HTTP web page
M2M_WIFI_RESP_ETHERNET_RX_PACKET	char *	Receiving 802.3 type ethernet packet in bypass mode
M2M_WIFI_REQ_DHCP_CONF	tstrM2MIPConfig	Response indicating that IP address obtained and Netmask, Gateway, DNS addresses of the network
M2M_WIFI_RESP_IP_CONFLICT	unsigned int	Response indicating a conflict in obtained IP address. The user should re attempt the DHCP request
M2M_WIFI_RESP_GET_SYS_TIME	tstrSystemTime	Response of the time of day from network
M2M_WIFI_RESP_WIFI_RX_PACKET	tstrM2MWifiRxPacketInfo	Indicate that a packet was received in monitor mode
M2M_WIFI_RESP_DEFAULT_CONNECT	tstrM2MDefaultConnResp	Response for the connection information in default connect

Note: To set the static IP address, refer the FAQ. To get the gateway, DNS, and netmask address - refer the FAQ.

4.5.2. Socket Callback Function

- Socket call back mechanism is used to handle the socket events. It uses appropriate structure to parse the socket event information such as the status or received data. while creating `socket()` function returns the socket ID.
- The `bind()` function specifies the address & port on the *local* side of the connection to monitor the corresponding socket activity.

- In response to the SOCKET_MSG_BIND event, listen() function is called using corresponding socket ID. This informs the system that it is willing to accept the incoming connection or it turns this socket in to *server socket*.
- In response to the SOCKET_MSG_LISTEN event, accept() functn is called to accept the any TCP client connect request do to the further data communication between the TCP client and the server. The system places any new connection request into a queue. The accept() function process the queue in First-In-First-Out (FIFO) order. Without accepting the connection request, server and client cannot communicate each other.
- The revc() function corresponding to the receive socket buffer handles the data transferred from the TCP client. When the data is received, socket will trigger the SOCKET_MSG_RECV event. Using the structure tstrSocketRecvMsg received data is parsed and passed to application usage. To handle further data transfer from client, recv() function must be called in SOCKET_MSG_RECV callback event or in the main loop.

Table 4-2. Socket layer Callback Events

Socket Events	Structures Used
SOCKET_MSG_BIND	tstrSocketBindMsg
SOCKET_MSG_LISTEN	tstrSocketListenMsg
SOCKET_MSG_ACCEPT	tstrSocketAcceptMsg
SOCKET_MSG_CONNECT	tstrSocketConnectMsg
SOCKET_MSG_RECV	tstrSocketRecvMsg
SOCKET_MSG_SEND	*(int16_t*)
SOCKET_MSG_SENDTO	*(int16_t*)
SOCKET_MSG_RECVFROM	tstrSocketRecvMsg

4.6. Handling Provisioning Credentials

When ATWINC1500 receives the credentials from the Android apps using TCP server, it will be parsed as shown. During the provisioning process, SSID,Security method, and passphrase are received and passed to this m2m_wifi_connect() function.

```
case SOCKET_MSG_RECV:{
tstrSocketRecvMsg *pstrRecv = (tstrSocketRecvMsg *)pvMsg;
        if (pstrRecv && pstrRecv->s16BufferSize > 0) {
            char *p;
            p = strtok((char *)pstrRecv->pu8Buffer, ",");
            if (p != NULL && !strncmp(p, "apply", 5)) {
                char str_ssid[M2M_MAX_SSID_LEN], str_pw[M2M_MAX_PSK_LEN];
                uint8 sec_type = 0;
                p = strtok(NULL, ",");
                if (p) {
                    strcpy(str_ssid, p);
                }
                p = strtok(NULL, ",");
                if (p) {
                    sec_type = atoi(p);
                }
                p = strtok(NULL, ",");
                if (p) {
                    strcpy(str_pw, p);
```

```
                }
                printf("Disable to AP.\r\n");
                m2m_wifi_disable_ap();
                nm_bsp_sleep(500);
                printf("Connecting to %s.\r\n", (char *)str_ssid);
                m2m_wifi_connect((char *)str_ssid, strlen((char *)str_ssid),
sec_type, str_pw, M2M_WIFI_CH_ALL);
                break;
            }
```

5. How to Run the AP Provision Mode Application

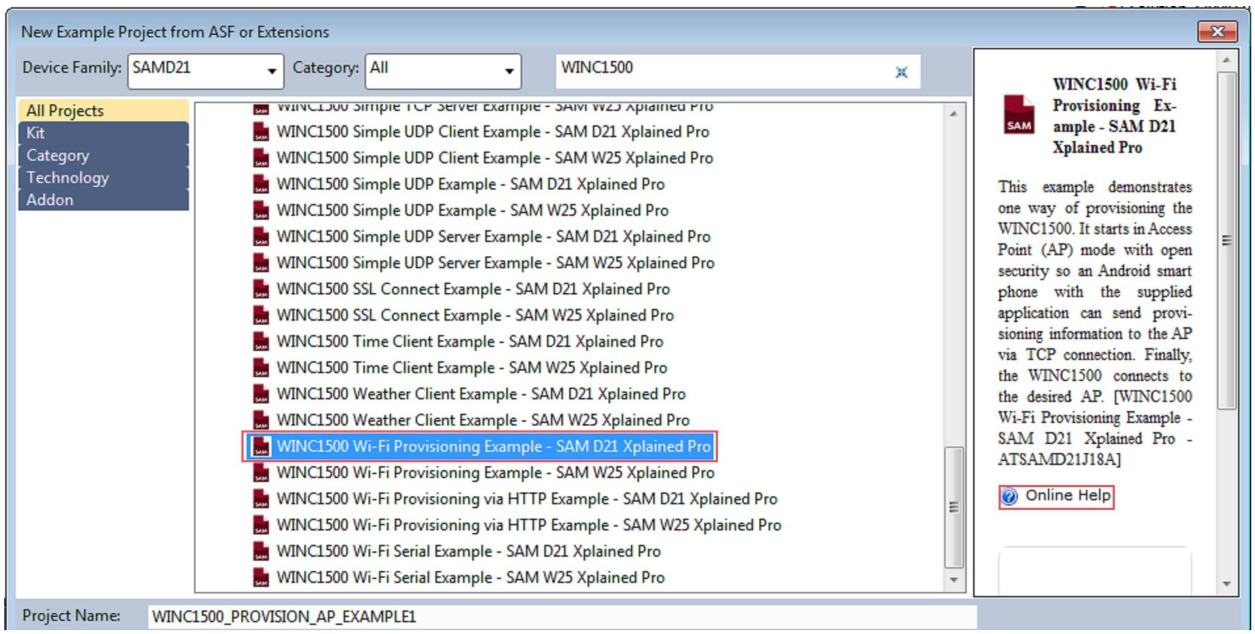
This section elaborates the steps to run the AP provisioning mode application using SAM D21 XPlained Pro board with ATWINC1500 WLAN module.

5.1. Getting Started ASF ATWINC1500 Provisioning Mode Demo

Tto demonstrate ATWINC1500 projects using Atmel Studio ASF example application,

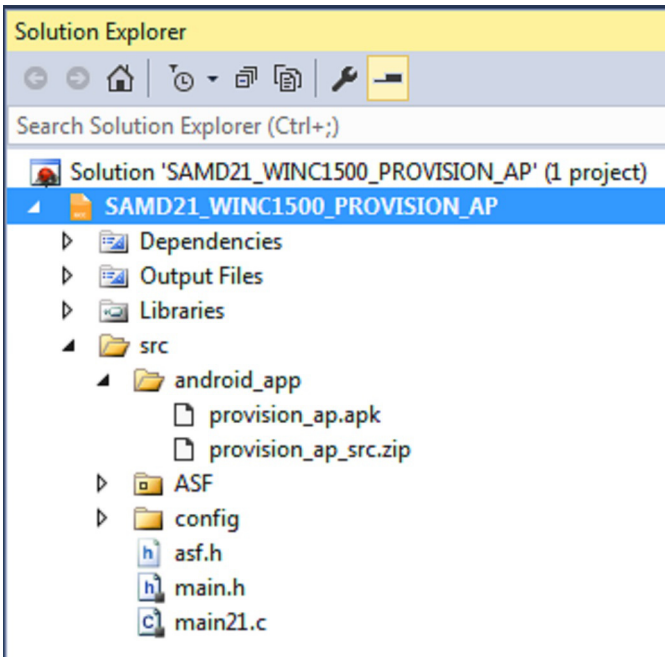
1. Open Atmel Studio 7. Go to **File** > **New** > **Example Projects**.
2. Search for ATWINC1500 sample application for other MCU.
3. Select the Wi-Fi Provisioning Example `WINC1500_AP_PROVISION_EXAMPLE` project for SAM D21 and open it.

Figure 5-1. Atmel Studio ATWINC1500 Project Creation



The directory structure for AP provision mode application is as follows.

Figure 5-2. AP Provision Mode Directory Structure

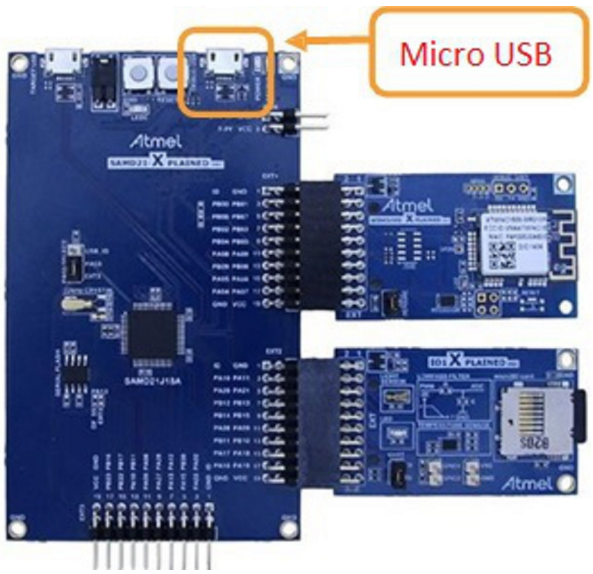


5.2. Programming the SAM D21 Xplained Pro

To dowload the firmware from PC to ATWINC1500, use the firmware update application provided in the ASF.

1. Connect the SAM D21 Xplained Pro board using the ATWINC1500 EXT1 header as shown.

Figure 5-3. SAM D21 Xplained Pro Board with ATWINC1500



2. Connect the USB cable to EDBG port of the SAM D21 Xplained Pro board.
3. Compile and program the ATWINC1500 ASF application using Atmel Studio.
4. To download or upgrade new release firmware into ATWINC1500 module, follow the steps specified in the Quick Start Guide.

5.3. Executing AP Provision Mode Application

This example demonstrates how to execute the ATWINC1500 Wi-Fi AP provisioning mode application using the SAM D21 Xplained Pro board as host MCU.

Following hardware are used in this example:

- The SAM D21 Xplained Pro
- The ATWINC1500 on EXT1 header
- The 802.11 b/g/n supported AP or router
- Android mobile device

Figure 5-4. Demo Setup

1. In the AP provision mode demo, the ATWINC1500 starts as a SoftAP using open security mode (no security method) and broadcasts the beacon frames with SSID `WINC1500_PROVISION_AP`. These macros are defined in the `main.h` file.

```
/** AP mode Settings */
#define MAIN_WLAN_SSID                    "WINC1500_PROVISION_AP" /* < SSID */
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_OPEN /* < Security manner
*/
#define MAIN_WLAN_CHANNEL                 (6) /* < Channel number */
```

2. Open the serial port terminal application using COM port configuration 115200,8,none,1,none.
3. Compile and download the image into the SAM D21 Xplained Pro board.

Figure 5-5. Atmel Studio Debug Button

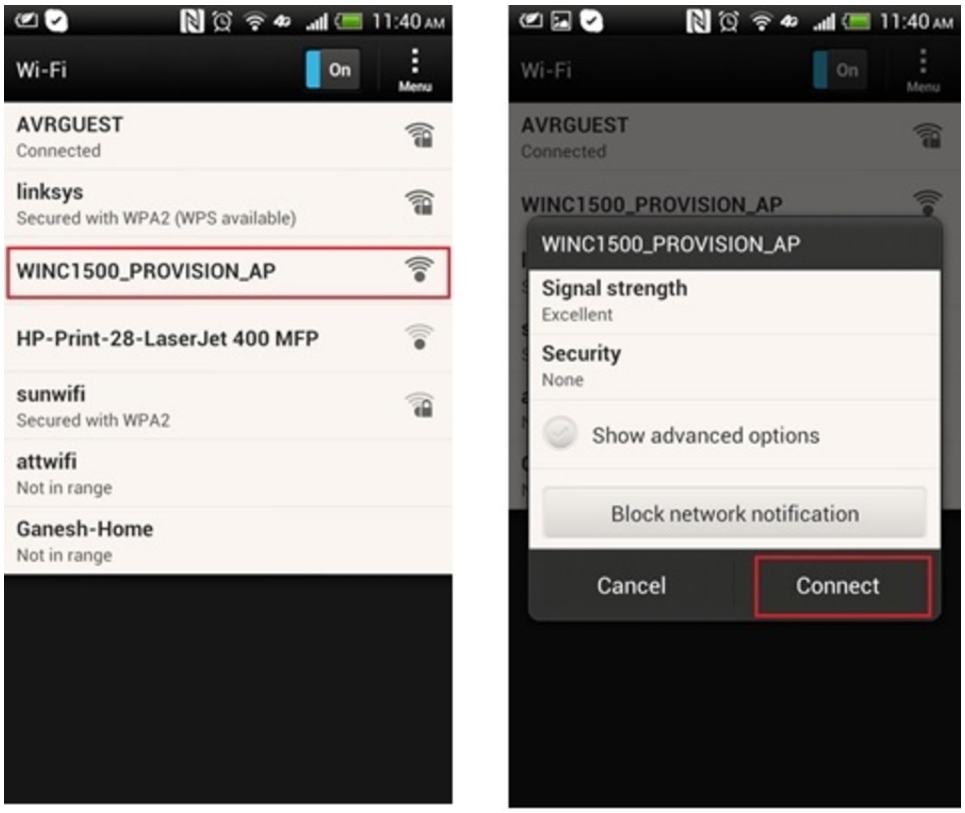
4. Run the application. Success or error messages appear in the serial port terminal.
5. The TCP server socket is listed in the terminal log. The socket will be ready to listen and waits for an incoming connection request from the TCP client.

Figure 5-6. AP Provision Mode Output

```
-- ATWINC1500 AP Provision example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
AP Provision mode started.
On the android device, connect to WINC1500_PROVISION_AP then run setting
app.
socket_cb: Ready to listen.
```

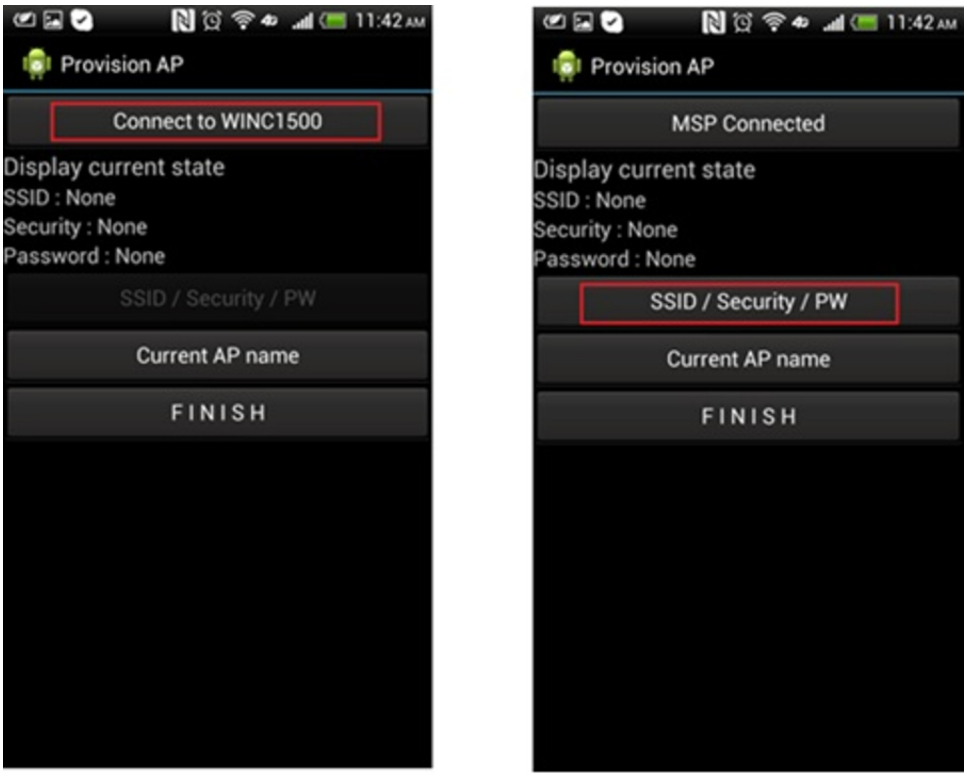
6. Connect to ATWINC1500 SoftAP.

Figure 5-7. ATWINC1500 AP Connection



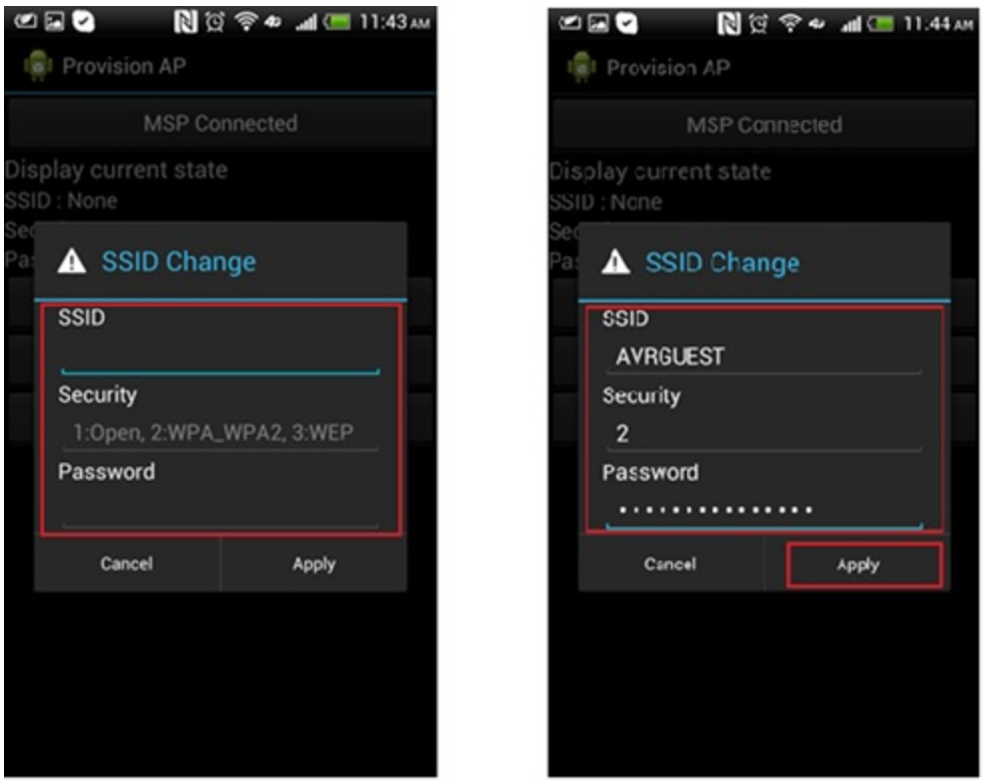
7. Install the Android app provided by Atmel. This is available in the AP provision mode ASF example application, `provision_ap.apk` in the `\src` directory.
8. Press **Connect to WINC1500** to establish the connection with the TCP server of the ATWINC1500. The *provision_ap.apk* Android app will pick the ATWINC1500 softAP IP address and port number 80 from the WLAN client connected to the ATWINC1500 softAP.

Figure 5-8. Connect to TCP Server and Enter Credentials



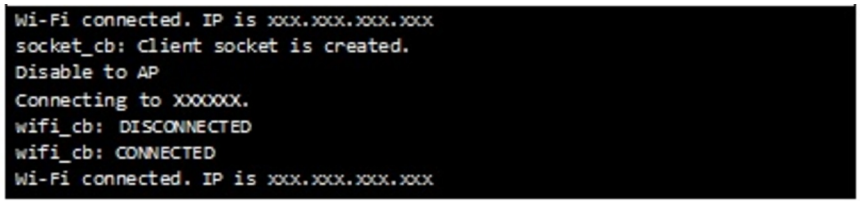
9. If the *socket connect error* does not appear, enter the AP credentials using **SSID / Security / PW**.
10. Enter the AP credentials such as SSID, security method, and passphrase as shown. click **Apply**.

Figure 5-9. Entering AP Credentials



11. When the ATWINC1500 TCP server receives the credentials from the Android Apps, the ATWINC1500 will disable the AP mode.
12. Switch to station mode to connect with AP using the credentials received. Aa successful connection terminal log appears.

Figure 5-10. Provision Mode Log



6. Revision History

Doc Rev.	Date	Comments
42636A	01/2016	Initial document release

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**