USER GUIDE

Software Programming Guide for ATWINC3400 Wi-Fi using SAM D21 Xplained Pro

Atmel SmartConnect

Introduction

This software programming guide describes the Atmel® ATWINC3400 Wi-Fi® Network Controller to build state-of-the-art Internet of Things (IoT) applications.

The following topics will be covered:

- How examples are organized
- Target board information
- Instruction for each example

Prerequisites

- Hardware Prerequisites
  - Atmel SAM D21 Xplained Pro Evaluation Kit
  - Atmel ATWINC3400 extension
  - Atmel IO1 extension
  - Micro-USB Cable (Micro-A/Micro-B)
- Software Prerequisites
  - Atmel Studio 7.0
  - Wi-Fi IoT Examples

Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic.
TIP	Highlights useful tips and techniques.
TO DO	Highlights objectives to be completed.
RESULT	Highlights the expected result of an assignment step.
WARNING	Indicates important information.
EXECUTE	Highlights actions to be executed out of the target.

1 How the Examples are Organized

This example package consists of several example codes and projects. The examples are organized in different levels of codes to explain ATWINC3400 API usage – from basic Wi-Fi operations to advanced topics. Here are the three category levels:

- Basic Examples
- IoT Protocol Examples
- Advanced App Scenario

These example materials are delivered by Atmel Software Framework, or by Atmel FAE (such together with hands-on documentation, datasheets, application notes, software, and tools).

1.1 Basic Examples

These examples describe basic Wi-Fi operation in 'how-to' manner:

- How to read Chip ID (to identify ATWINC3400 H/W revision difference)
- How to adjust debug message level
- How to get MAC address of the Wi-Fi module
- How to start Wi-Fi in specific operation mode, such as:
  - STA Mode (Station mode, known as a Wi-Fi client)
  - AP mode (Access Point mode)
- How to scan AP list that is nearby
- How to set deep sleep mode
- How to connect to secure Wi-Fi with using WEP/WPA/WPA2 security
- How to connect to enterprise security network
- How to connect to security WPS
- How to get RF signal status by reading RSSI value
- How to provision via BLE
- How to provision via AP
- How to provision via HTTP

1.2 Protocol Examples

After basic code examples, the user may want to explore how to send and receive network packets. Here are some protocol examples that can be extended for IoT application:

- UDP protocol example
  - Server and Client
  - Client
  - Server
- TCP protocol example
  - Client
  - Server
- NTP Time client – retrieves network time for IoT application
- Send email – send email from SMTP server
- Location client – get the current location of the network provider using HTTP

1.3 Advanced Examples

These examples demonstrate more complex examples like:

- Growl client – demonstrates using RESTful API over SSL (essential for IoT application)
- MQTT Chat client – demonstrate how to send and receive IoT information using MQTT protocol
- Weather client – get the current weather information of the network provider and utilize the IO1 sensor device
- Wi-Fi serial - useful for chatting or controlling a remote device
- OTA Firmware Upgrade – ATWINC3400 Firmware upgrade via OTA server
- SSL connection - Set up an SSL connection

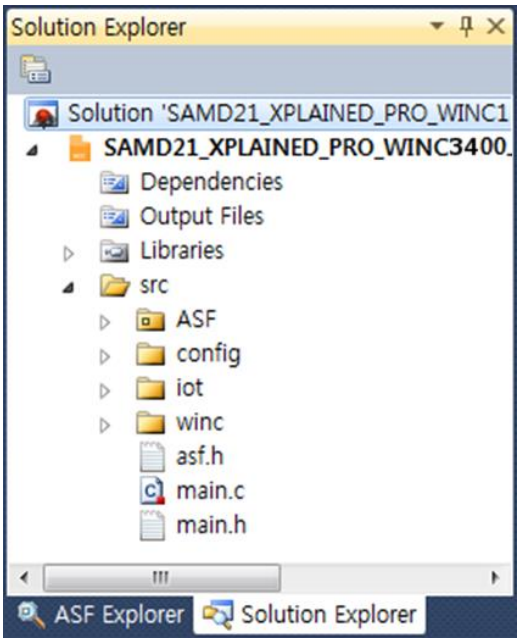
For customer's IoT application, these examples are useful to how to use ATWINC3400 APIs and implement a feature for IoT application.

2 Source Organization

There are some folders which are allocated automatically according to user configurations, and example source consists of main.c and main.h.

Here is the structure of application source codes:

- **ASF**
  All source codes of ASF modules are located in this folder. You can select various modules with ASF wizard and it will configure the content in this folder.
- **config**
  This folder consists of configuration header files for SAM D21 and extension boards.
- **iot**
  Some of the protocol/advanced examples have this folder. It contains the source codes of IoT protocol like HTTP, MQTT, and so forth.
- **winc**
  This is the driver source folder of ATWINC3400 Wi-Fi module.



INFO Some examples may have additional source files, but the structure is similar across the samples.

3 Basic Operation Code

This chapter explains the basic code for using SAM D21 and ATWINC3400. These codes can be different according to the purpose of your example.

3.1 Initialization

Initialize SAM D21 board.

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

1. Initialize UART console to print debug messages.

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

2. Initialize board support package to use ATWINC3400.

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

3. Initialize Wi-Fi driver. You don't need to set the Wi-Fi callback function if you don't use Wi-Fi connection. (E.g. Get Chip ID example, Get MAC Address example.)

```
/* 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) {
	}
}
```

3.2 Wi-Fi Connection

1. Initialize socket module and set socket callback functions to receive socket events and connect ATWINC3400 module to an AP with given information. These are normally defined in main.h.

```
/* Initialize Socket module */
socketInit();
registerSocketCallback(socket_event_handler, socket_resolve_handler);

/* Connect to router. */
ret = m2m_wifi_connect((char *)MAIN_WLAN_SSID, sizeof(MAIN_WLAN_SSID),
	MAIN_WLAN_AUTH, (char *)MAIN_WLAN_PSK, M2M_WIFI_CH_ALL);
```

4 Examples

4.1 Basic Example: How to Get Chip ID

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to retrieve the chip information of the Wi-Fi module. This is a basic operation to identify which HW version is used. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and retrieve information.

1. Code summary.
   nmi_get_chipid() function returns the chip ID of ATWINC3400.
   nmi_get_rfrevid() function returns RF revision ID.

```
/* Display WINC3400 chip information. */
printf("Chip ID : \r\t\t\t%x\r\n", (unsigned int)nmi_get_chipid());
printf("RF Revision ID : \r\t\t\t%x\r\n", (unsigned int)nmi_get_rfrevid());
```

2. Build the program and download it into the board.
3. Start the application.

RESULT

The application is now programmed and running. Following information will be displayed on the terminal window. In the below result, you can see the chip ID of 3000d0 and RF revision ID of 6. The user must be aware of which version of ATWINC3400 module is used.

```
-- ATWINC3400 chip information example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx--
Chip ID :               3000d0
RF Revision ID :        6
Done.
```

TIPS

ATWINC3400 behavior and corresponding log messages can be different according to the revision.

4.2 Basic Example: How to Set Debug Level

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to configure debug level and print debugging messages of the Wi-Fi module. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and print debugging messages according to the debug level.

1. Code summary.
   Enable debug mode by configuring the below code in config\conf_winc.h:

```
/** Debug Options */
#define CONF_WIFI_M2M_DEBUG                    (1)
```

   This example demonstrates setting debug level by M2M_DEBUG_LEVEL(...) macro.

```
for (int8_t i = 0; i < 5; i++) {
	/* Set debug level. */
	M2M_DEBUG_LEVEL(i);

	printf("\r[set debug level : %d]\r\n", (unsigned int)i);

	/* Display debug information. */
	printf("\r");
	M2M_PRINT("test message.\n");
	M2M_ERR("test message.\n");
	M2M_INFO("test message.\n");
	M2M_REQ("test message.\n");
	M2M_DBG("test message.\n");

	printf("\r\n");
}
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 debug level example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
[set debug level : 0]
test message.

[set debug level : 1]
test message.
(APP)(ERR)[main][186]test message.

[set debug level : 2]
test message.
(APP)(ERR)[main][186]test message.
(APP)(INFO)test message.

[set debug level : 3]
test message.
(APP)(ERR)[main][186]test message.
(APP)(INFO)test message.
(APP)(R)test message.

[set debug level : 4]
test message.
(APP)(ERR)[main][186]test message.
(APP)(INFO)test message.
(APP)(R)test message.
(APP)(DBG)[main][189]test message.

Done.
```

TIPS For a debugging session, the user must turn on the debug option, and analyze the situation. The engineering team also requires the logs to review.

4.3 Basic Example: How to Get MAC Address

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to retrieve the MAC address of the Wi-Fi module. The example uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

1. Code summary.
   MAC address is mostly stored in the OTP-ROM. You can get it via the m2m_wifi_get_otp_mac_address() function.

```
/* Get MAC Address from OTP. */
m2m_wifi_get_otp_mac_address(mac_addr, &u8IsMacAddrValid);
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 MAC ADDRESS example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
OTP MAC Address : XX:XX:XX:XX:XX:XX
(Or)
USER MAC Address : XX:XX:XX:XX:XX:XX
```

TIPS Default MAC address: The MAC address in OTP ROM. (One Time Programmable ROM.)

TIPS User Define MAC address: If you want to use a custom MAC address, you should set the user defined MAC address.

TIPS In this example result, you can see the OTP MAC address or USER MAC address.

4.4 Basic Example: How to Run STA Mode

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to behave as a station. The example uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and connect to AP as a station mode.

1. Code summary.
   Configure the below code in main.h for AP information to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Connect ATWINC3400 to the AP via m2m_wifi_connect() function.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, sizeof(MAIN_WLAN_SSID),
       MAIN_WLAN_AUTH, (void *)MAIN_WLAN_PSK, M2M_WIFI_CH_ALL);
```

   wifi_cb() function is called with M2M_WIFI_RESP_CON_STATE_CHANGED message and then it request an IP address via the m2m_wifi_request_dhcp_client() function.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
       switch (u8MsgType) {
       case M2M_WIFI_RESP_CON_STATE_CHANGED:
       {
              tstrM2mWifiStateChanged *pstrWifiState = ...;
              if (pstrWifiState->u8CurrState == M2M_WIFI_CONNECTED) {
                     m2m_wifi_request_dhcp_client();
              }
```

   wifi_cb() function is called with M2M_WIFI_REQ_DHCP_CONF message and finally get an IP address.

```
       case M2M_WIFI_REQ_DHCP_CONF:
       {
              uint8_t *pu8IPAddress = (uint8_t *)pvMsg;
              printf("Wi-Fi connected\r\n");
              printf("Wi-Fi IP is %u.%u.%u.%u\r\n", pu8IPAddress[0],
                            pu8IPAddress[1], pu8IPAddress[2], pu8IPAddress[3]);
              break;
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 station mode example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
Connecting to XXXXXX.
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
```

4.5 Basic Example: How to Run AP Mode

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to behave as an AP. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and act as an AP.

1. Code summary.
   Configure the below code in main.h for AP information.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_OPEN
#define MAIN_WLAN_CHANNEL       (6)
```

   In main() function, initialize AP mode configuration structure (strM2MAPConfig) as shown below. You can enable AP mode via m2m_wifi_enable_ap function().

```
tstrM2MAPConfig strM2MAPConfig = {
        MAIN_WLAN_SSID,                     /* Access Point Name. */
        MAIN_WLAN_CHANNEL,                  /* Channel to use. */
        MAIN_WLAN_WEP_KEY_INDEX,            /* Wep key index. */
        MAIN_WLAN_WEP_SIZE,                 /* Wep key size. */
        MAIN_WLAN_WEP_KEY,                  /* Wep key. */
        MAIN_WLAN_AUTH,                     /* Security mode. */
        MAIN_WLAN_SSID_MODE,                /* SSID visible. */
        MAIN_WLAN_DHCP_SERVER_IP            /* DHCP Server IP */
};

ret = m2m_wifi_enable_ap(&strM2MAPConfig);
```

2. Build the program and download it into the board.
3. Start the application.

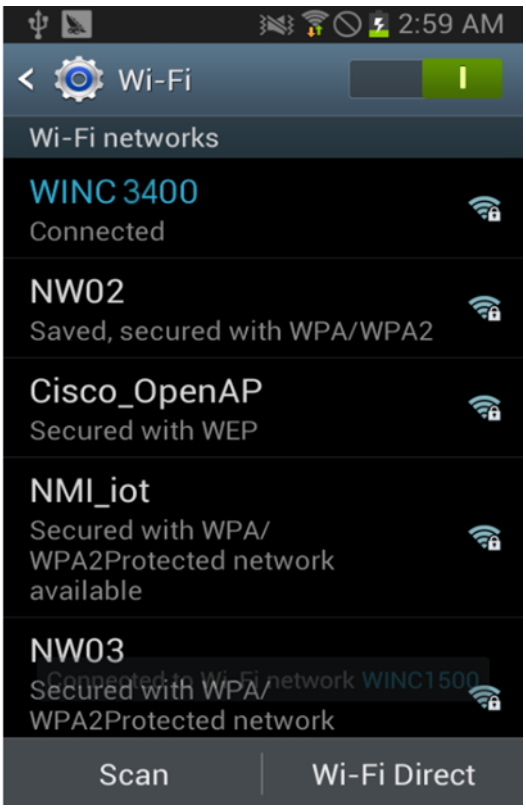
RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 AP mode example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
AP mode started. You can connect to XXXXXX.
Station connected
Station IP is xxx.xxx.xxx.xxx
```

INFO The ATWINC3400 supports AP mode operation with the following limitations:

1. Only ONE associated station is supported. After a connection is established with a station, further connections are rejected.
    a. OPEN and WEP security modes.
    b. The device could not work as a station in this mode (STA/AP Concurrency is not supported).

EXECUTE Now the AP mode is ready. You can use a smart phone to connect to the ATWINC3400 that is running as AP mode.



4.6 Basic Example: How to Scan APs

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to how to scan AP as a station. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and scan AP until the defined AP is found.

1. Code summary.
   Configure the below code in main.h for the AP to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Request to scan for all channels.

```
m2m_wifi_request_scan(M2M_WIFI_CH_ALL);
```

   wifi_cb() function is called with M2M_WIFI_RESP_SCAN_DONE message when scanning is done. You can get the number of found APs as below and request the scan result with a specific channel by calling m2m_wifi_req_scan_result() function.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
        case M2M_WIFI_RESP_SCAN_DONE:
                tstrM2mScanDone *pstrInfo = (tstrM2mScanDone *)pvMsg;
                if (pstrInfo->u8NumofCh >= 1) {
                        m2m_wifi_req_scan_result(scan_request_index);
                        scan_request_index++;
                }
```

   wifi_cb() function will be called again with M2M_WIFI_RESP_SCAN_RESULT message. You can get the information of the AP for the specific channel number you gave.

   If scan result is the same with the AP information in main.h then the device will connect to the AP.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
        case M2M_WIFI_RESP_SCAN_RESULT:
                printf("[%d] SSID:%s\r\n",
                        scan_request_index, pstrScanResult->au8SSID);
                ...
                if ((demo_ssid_len == scan_ssid_len) &&
                                (!memcmp(pstrScanResult->au8SSID,
                                        (uint8_t *)MAIN_WLAN_SSID, demo_ssid_len))
                        ) {
                m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 AP scan example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
[1] SSID:DEMO_AP1
[2] SSID:DEMO_AP2
[3] SSID:DEMO_AP
Found DEMO_AP
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
```

4.7 Basic Example: Security with WEP/WPA

This example demonstrates how to connect ATWINC3400 Wi-Fi device to AP with WEP, WPA Security. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and connect to AP using security mode WEP, WPA.

1. Code summary.

   Case 1: WEP

   To test WEP security, modify MAIN_WLAN_DEVICE_NAME, MAIN_WLAN_WEP_KEY_INDEX, and MAIN_WLAN_WEP_KEY_40 or MAIN_WLAN_WEP_KEY_104 in main.h.

```
#define MAIN_WLAN_DEVICE_NAME           "DEMO_AP"
#define MAIN_WLAN_WEP_KEY_INDEX         1
#define MAIN_WLAN_WEP_KEY_40            "1234567890"
#define MAIN_WLAN_WEP_KEY_104           "1234567890abcdef1234567890"
```

   Use case 1 in main() function and select wep64_parameters or wep128_parameters as security parameters.

```
tstrM2mWifiWepParams wep64_parameters = { MAIN_WLAN_WEP_KEY_INDEX,
              sizeof(MAIN_WLAN_WEP_KEY_40), MAIN_WLAN_WEP_KEY_40};
tstrM2mWifiWepParams wep128_parameters = { MAIN_WLAN_WEP_KEY_INDEX,
              sizeof(MAIN_WLAN_WEP_KEY_104), MAIN_WLAN_WEP_KEY_104};
```

```
m2m_wifi_connect((char *)MAIN_WLAN_DEVICE_NAME,
              strlen((char *)MAIN_WLAN_DEVICE_NAME), M2M_WIFI_SEC_WEP,
              &wep64_parameters, M2M_WIFI_CH_ALL);
```

   Case 2: WPA

   To test WPA security, use case 2 in main() function and modify MAIN_WLAN_PSK in main.h.

```
#define MAIN_WLAN_PSK          "12345678"
```

   Connect to the AP with the given information.

```
m2m_wifi_connect((char *)MAIN_WLAN_DEVICE_NAME, ...
```

2. Prepare an AP that supports WEP and WPA/WPA2 Security and configure Wi-Fi Security. For more information, refer to the AP manufacturer's manual.
3. Run the application. If the device connected successfully, the IP address, which is assigned by DHCP, will be displayed on the terminal program.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 security connection with WEP,WPA security example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
Connecting to XXXXXX.
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
```

4.8 Basic Example: Connection to Security WPS

This example demonstrates how to connect ATWINC3400 Wi-Fi device to AP with WPS Security. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1

main.c: Initialize the ATWINC3400 and connect AP using WPS.

1. Code summary.

   Case 1: Button method

   To test the WPS button method, configure the WPS push button feature in main.h as below and use case 1 in main() function.

```
#define MAIN_WPS_PUSH_BUTTON_FEATURE        "true"
```

```
if (MAIN_WPS_PUSH_BUTTON_FEATURE) {
        /* case 1 WPS Push Button method */
        if (!gbPressButton) {
                btn_init();
        }
}
```

   When pressing the SW0 button on the SAM D21, it will trigger WPS in btn_press() function.

```
m2m_wifi_wps(WPS_PBC_TRIGGER, NULL);
```

   wifi_cb() will receive M2M_WIFI_REQ_WPS message and it can connect to the AP with given information.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
        case M2M_WIFI_REQ_WPS:
        {
                m2m_wifi_connect((char *)pstrWPS->au8SSID, ...
```

   Case 2: PIN method

   To test the WPS PIN method, configure the WPS PIN number and the WPS push button feature in main.h as below and use case 2 in main() function.

```
#define MAIN_PIN_NUMBER                     "12345670"
#define MAIN_WPS_PUSH_BUTTON_FEATURE        "false"
```

```
if (!MAIN_WPS_PUSH_BUTTON_FEATURE) {
        /* case 2 WPS PIN method */
        m2m_wifi_wps(WPS_PIN_TRIGGER, (const char *)MAIN_WPS_PIN_NUMBER);
}
```

2. Prepare an AP that supports Wi-Fi Protected Setup (WPS).

3. Press the WPS button on the AP when using the WPS button method or enter the WPS PIN number in the AP setup menu and start the AP. (For more information, refer to AP product documentation.)
4. Run the application. Press the SW0 button on the SAM D21 when using the WPS button method. The ATWINC3400 will be connected to the AP automatically without security information.

RESULT In the WPS button method, the following information will be displayed on the terminal window.

```
-- ATWINC3400 security connection with Wi-Fi Protected Setup(WPS) example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
SW0 button pressed
Device is connecting using WPS Push Button option
Wi-Fi request WPS
SSID : xxxxxx, AuthType : x, PW : xxxxxxxx
Request Wi-Fi connect
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
```

RESULT In the WPS PIN method, the following information will be displayed on the terminal window.

```
-- ATWINC3400 security connection with Wi-Fi Protected Setup(WPS) example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Wi-Fi request WPS
SSID : xxxxxx, AuthType : x, PW : xxxxxxxx
Request Wi-Fi connect
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
```

4.9 Basic Example: Get Signal Status

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to check the signal strength such as RSSI. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and get the RSSI for the connected AP.

1. Code summary.
   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Connect to the AP with the given information.

```
m2m_wifi_connect((char *) MAIN_WLAN_SSID, ...
```

   Call m2m_wifi_req_curr_rssi() to receive RSSI.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
       case M2M_WIFI_REQ_DHCP_CONF:
              m2m_wifi_req_curr_rssi();
```

   You can get the RSSI value when the wifi_cb() function is called with the M2M_WIFI_RESP_CURRENT_RSSI message.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
       case M2M_WIFI_RESP_CURRENT_RSSI:
       {
              int8_t *rssi = (int8_t *)pvMsg;
              printf("RSSI for the current connected AP (%d)\r\n",
                            (int8_t)(*rssi));
       }
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 signal statistics example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
RSSI for the current connected AP (-48)
```

4.10 BLE Provisioning

The ATWINC3400 supports provisioning using a BLE interface. The release package includes a BLE provisioning demo application for running on the SAM D21 Xplained Pro board, along with Android and IPhone applications for use on a smart phone. For understanding and usage instructions, refer to Atmel ATWINC3400-BLE-Provisioning-Setup-and-Usage_UserGuide.pdf.

4.11 Basic Example: AP Provision

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to start Provision Mode. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and start Provision Mode until one of various APs is selected.

1. Code summary.

   Initialize the socket module and create the TCP server socket.

```
addr.sin_family = AF_INET;
addr.sin_port = _htons((MAIN_WIFI_M2M_SERVER_PORT));
addr.sin_addr.s_addr = 0;
socketInit();
registerSocketCallback(socket_cb, NULL);
...
while (1) {
       m2m_wifi_handle_events(NULL);
              if (tcp_server_socket < 0) {
              /* Open TCP server socket */
                     if ((tcp_server_socket = socket(AF_INET, SOCK_STREAM, 0))
```

   Enable the AP mode before the main loop. (Refer to “How to Run AP Mode” example.)

```
ret = m2m_wifi_enable_ap(&strM2MAPConfig);
```

   After your Android device is connected to ATWINC3400 sends AP configuration, disable AP mode and connect to the AP with the given information.

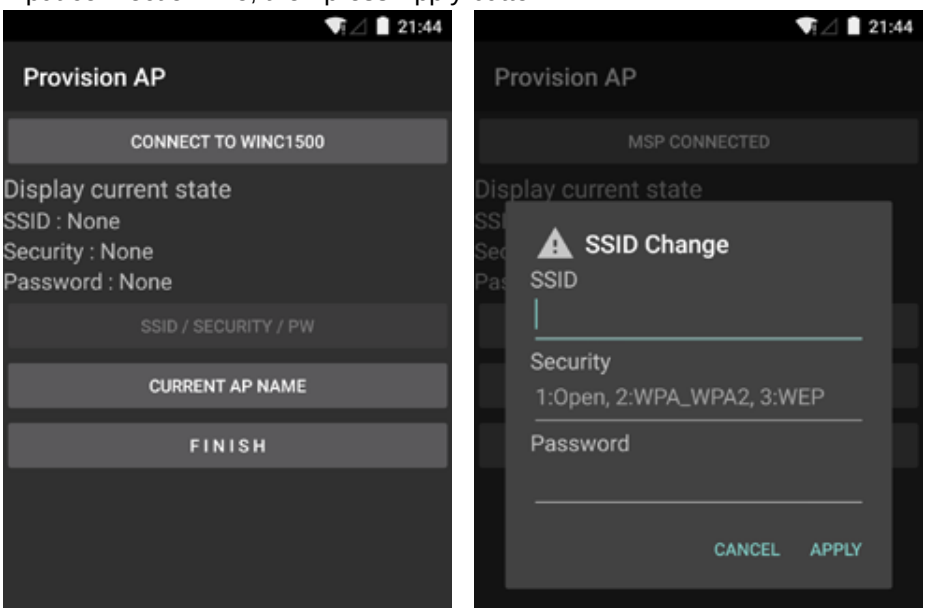
```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
       case SOCKET_MSG_RECV:
              m2m_wifi_disable_ap();
              nm_bsp_sleep(500);
              m2m_wifi_connect((char *)str_ssid, …
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 AP Provision example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
AP Provision mode started.
On the android device, connect to WINC3400_PROVISION_AP then run setting app.
socket_cb: Ready to listen.
```

4. Install provision_ap.apk in the source package to your Android device. You can also build the Android application source and install it.
5. Connect your Android device to the ATWINC3400.
6. Launch the Android application to configure AP, press the Connect button, and then the SSID button will be available.
7. Input connection info, then press Apply button.



8. ATWINC3400 will be connected to the AP which you configured.

```
Wi-Fi connected. IP is xxx.xxx.xxx.xxx
socket_cb: Client socket is created.
Disable to AP
Connecting to XXXXXX.
wifi_cb: DISCONNECTED
wifi_cb: CONNECTED
Wi-Fi connected. IP is xxx.xxx.xxx.xxx
```

4.12 Basic Example: HTTP Provision

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to start Provision Mode. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and start the Provision Mode until one of various AP is selected.

1. Code summary.

   Configure the below code in main.h for provision information.

```
#define MAIN_M2M_DHCP_SERVER_IP             {192, 168, 1, 1}
#define MAIN_HTTP_PROV_SERVER_DOMAIN_NAME   "atmelconfig.com"
#define MAIN_M2M_DEVICE_NAME                "WINC3400_00:00"
```

   Start provision mode before the main loop.

```
m2m_wifi_start_provision_mode((tstrM2MAPConfig *)&gstrM2MAPConfig,
                              (char *)gacHttpProvDomainName, 1);
```

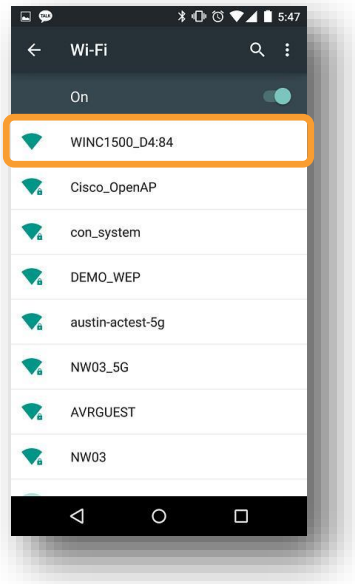
   When your mobile device sends configuration information, the wifi_cb() function will be called with M2M_WIFI_RESP_PROVISION_INFO message and you can connect to the AP with the given information.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
       case M2M_WIFI_RESP_PROVISION_INFO:
              tstrM2MProvisionInfo *pstrProvInfo = (tstrM2MProvisionInfo *)pvMsg;
              if (pstrProvInfo->u8Status == M2M_SUCCESS) {
                     m2m_wifi_connect((char *)pstrProvInfo->au8SSID,
                            strlen((char *)pstrProvInfo->au8SSID),
                            pstrProvInfo->u8SecType,
                            pstrProvInfo->au8Password, M2M_WIFI_CH_ALL);
              }
```

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 HTTP Provision example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Provision Mode started.
Connect to [atmelconfig.com] via AP[WINC3400_xx:xx] and fill up the page
```

1. Connect your mobile device to ATWINC3400 AP (ATWINC3400_xx:xx).



2. Browse the webpage (atmel.com) to setup AP, complete the page, and then press Connect.

3. ATWINC3400 will be connected to the AP what you configured.

```
wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: DISCONNECTED.
wifi_cb: M2M_WIFI_RESP_PROVISION_INFO:
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
```

4.13 Protocol Example: UDP (Server and Client)

This program demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to test UDP socket. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the Wi-Fi module and test the UDP server and the client.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID                    "DEMO_AP"
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK                     "12345678"
#define MAIN_WIFI_M2M_PRODUCT_NAME        "NMCTemp"
#define MAIN_WIFI_M2M_SERVER_IP           0xFFFFFFFF
#define MAIN_WIFI_M2M_SERVER_PORT         (6666)
#define MAIN_WIFI_M2M_REPORT_INTERVAL     (1000)
```

   Initialize socket module.

```
addr.sin_family = AF_INET;
addr.sin_port = _htons(MAIN_WIFI_M2M_SERVER_PORT);
addr.sin_addr.s_addr = _htonl(MAIN_WIFI_M2M_SERVER_IP);
socketInit();
registerSocketCallback(socket_cb, NULL);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   In the main loop, after the device is connected to the AP, create an RX socket and bind it.

```
if (rx_socket < 0) {
	if ((rx_socket = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
		continue;
	}
	bind(rx_socket, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
}
```

   In socket_cb() function, prepare a buffer to receive data.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	if (u8Msg == SOCKET_MSG_BIND) {
		recvfrom(sock, gau8SocketTestBuffer, MAIN_WIFI_M2M_BUFFER_SIZE, 0);
```

Create a TX socket in the main loop.

```
if (tx_socket < 0) {
	uint32 u32EnableCallbacks = 0;
	if ((tx_socket = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
		continue;
	}
	setsockopt(tx_socket, SOL_SOCKET, SO_SET_UDP_SEND_CALLBACK,
		&u32EnableCallbacks, 0);
}
```

After binding is completed, send data from the TX socket to the RX socket.

```
ret = sendto(tx_socket, &msg_temp_report, sizeof(t_msg_temp_report), 0,
		(struct sockaddr *)&addr, sizeof(addr));
```

You can receive data in the socket_cb() function with the SOCKET_MSG_RECVFROM message.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	} else if (u8Msg == SOCKET_MSG_RECVFROM) {
		tstrSocketRecvMsg *pstrRx = (tstrSocketRecvMsg *)pvMsg;
		if (pstrRx->pu8Buffer && pstrRx->s16BufferSize) {
```

2. Build the program and download it into the board, both device A and device B.
3. Start the application of device A and device B.

RESULT The application of device A is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 UDP example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF : IP is xxx.xxx.xxx.xxx
socket_cb: bind success!
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
UDP test Complete!
```

RESULT The application of device B is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 UDP example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF : IP is xxx.xxx.xxx.xxx
socket_cb: bind success!
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
main: message sent
socket_cb: received app message
UDP test Complete!
```

4.14 Protocol Example: UDP Client

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to test UDP socket. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the Wi-Fi module and test UDP server.

1. Code summary.
   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID                    "DEMO_AP"
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK                     "12345678"
#define MAIN_WIFI_M2M_PRODUCT_NAME        "NMCTemp"
#define MAIN_WIFI_M2M_SERVER_IP           0xFFFFFFFF
#define MAIN_WIFI_M2M_SERVER_PORT         (6666)
#define MAIN_WIFI_M2M_REPORT_INTERVAL     (1000)
```

   Initialize socket module.

```
addr.sin_family = AF_INET;
addr.sin_port = _htons(MAIN_WIFI_M2M_SERVER_PORT);
addr.sin_addr.s_addr = _htonl(MAIN_WIFI_M2M_SERVER_IP);
socketInit();
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, create a TX socket in the main loop.

```
if (tx_socket < 0) {
	if ((tx_socket = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
		continue;
	}
}
```

   After binding is completed, send data from the TX socket to the RX socket.

```
ret = sendto(tx_socket, &msg_temp_report, sizeof(t_msg_temp_report), 0,
		(struct sockaddr *)&addr, sizeof(addr));
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 UDP client example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED : CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF : IP is xxx.xxx.xxx.xxx
main: message sent
. . .
main: message sent
UDP client test Complete!
```

4.15 Protocol Example: UDP Server

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to test UDP socket. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the Wi-Fi module and test the UDP client.

1. Code summary.
   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID                    "DEMO_AP"
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK                     "12345678"
#define MAIN_WIFI_M2M_PRODUCT_NAME        "NMCTemp"
#define MAIN_WIFI_M2M_SERVER_IP           0xFFFFFFFF
#define MAIN_WIFI_M2M_SERVER_PORT         (6666)
#define MAIN_WIFI_M2M_REPORT_INTERVAL     (1000)
```

   Initialize the socket module and create the UDP server socket.

```
addr.sin_family = AF_INET;
addr.sin_port = _htons(MAIN_WIFI_M2M_SERVER_PORT);
addr.sin_addr.s_addr = _htonl(MAIN_WIFI_M2M_SERVER_IP);
socketInit();
registerSocketCallback(socket_cb, NULL);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, create an RX socket and bind it in the main loop.

```
if (rx_socket < 0) {
	if ((rx_socket = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
		continue;
	}
	bind(rx_socket, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
}
```

   In the socket_cb() function, prepare a buffer to receive data.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	if (u8Msg == SOCKET_MSG_BIND) {
		recvfrom(sock, gau8SocketTestBuffer, MAIN_WIFI_M2M_BUFFER_SIZE, 0);
```

You can receive data in the socket_cb() function with the SOCKET_MSG_RECVFROM message when a client device sends data. (Use “UDP Client” example.)

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	} else if (u8Msg == SOCKET_MSG_RECVFROM) {
		tstrSocketRecvMsg *pstrRx = (tstrSocketRecvMsg *)pvMsg;
		if (pstrRx->pu8Buffer && pstrRx->s16BufferSize) {
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 UDP server example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED : CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF : IP is xxx.xxx.xxx.xxx
socket_cb: bind success!
socket_cb: received app message.(1)
. . .
socket_cb: received app message.(10)
UDP server test Complete!
```

4.16 Protocol Example: TCP Client

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to test the TCP client. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the Wi-Fi module and test the TCP client.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID                    "DEMO_AP"
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK                     "12345678"
#define MAIN_WIFI_M2M_PRODUCT_NAME        "NMCTemp"
#define MAIN_WIFI_M2M_SERVER_IP           0xFFFFFFFF
#define MAIN_WIFI_M2M_SERVER_PORT         (6666)
#define MAIN_WIFI_M2M_REPORT_INTERVAL     (1000)
```

   Initialize the socket module and register socket callback function.

```
addr.sin_family = AF_INET;
addr.sin_port = _htons(MAIN_WIFI_M2M_SERVER_PORT);
addr.sin_addr.s_addr = _htonl(MAIN_WIFI_M2M_SERVER_IP);
socketInit();
registerSocketCallback(socket_cb, NULL);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, create a TCP client socket and connect to the server in the main loop.

```
if (tcp_client_socket < 0) {
       if ((tcp_client_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
              continue;
       }
}
ret = connect(tcp_client_socket, (struct sockaddr *)&addr, ...);
```

Connect, send and recv operations will be executed sequentially in the socket_cb() function.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	case SOCKET_MSG_CONNECT:
	{
		if (pstrConnect && pstrConnect->s8Error >= 0)
			send(tcp_client_socket, &msg_wifi_product, ...);
	}
	...
	case SOCKET_MSG_SEND:
	{
		recv(tcp_client_socket, gau8SocketTestBuffer, ...);
	}
	...
	case SOCKET_MSG_RECV:
	{
		tstrSocketRecvMsg *pstrRecv = (tstrSocketRecvMsg *)pvMsg;
		if (pstrRecv && pstrRecv->s16BufferSize > 0) {
			printf("socket_cb: recv success!\r\n");
			printf("TCP Client Test Complete!\r\n");
		}
	}
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 TCP client example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
m2m_wifi_state: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
socket_cb: connect success!
socket_cb: send success!
socket_cb: recv success!
TCP Client Test Complete!
```

4.17 Protocol Example: TCP Server

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to test the TCP server. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the Wi-Fi module and test the TCP server.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID                    "DEMO_AP"
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK                     "12345678"
#define MAIN_WIFI_M2M_PRODUCT_NAME        "NMCTemp"
#define MAIN_WIFI_M2M_SERVER_IP           0xFFFFFFFF
#define MAIN_WIFI_M2M_SERVER_PORT         (6666)
#define MAIN_WIFI_M2M_REPORT_INTERVAL     (1000)
```

   Initialize the socket module and register socket callback function.

```
addr.sin_family = AF_INET;
addr.sin_port = _htons(MAIN_WIFI_M2M_SERVER_PORT);
addr.sin_addr.s_addr = _htonl(MAIN_WIFI_M2M_SERVER_IP);
socketInit();
registerSocketCallback(socket_cb, NULL);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, create a TCP server socket and bind it in the main loop.

```
if (tcp_server_socket < 0) {
	if ((tcp_server_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
		continue;
	}
	bind(tcp_server_socket, (struct sockaddr *)&addr, ...);
}
```

Five operations (bind / listen / accept / recv / send) will be executed sequentially in socket_cb() function.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	case SOCKET_MSG_BIND:
	{
		tstrSocketBindMsg *pstrBind = (tstrSocketBindMsg *)pvMsg;
		if (pstrBind && pstrBind->status == 0)
			listen(tcp_server_socket, 0);
	}
	...
	case SOCKET_MSG_LISTEN:
	{
		tstrSocketListenMsg *pstrListen = (tstrSocketListenMsg *)pvMsg;
		if (pstrListen && pstrListen->status == 0)
			accept(tcp_server_socket, NULL, NULL);
	}
	...
	case SOCKET_MSG_ACCEPT:
	{
		tstrSocketAcceptMsg *pstrAccept = (tstrSocketAcceptMsg *)pvMsg;
		if (pstrAccept) {
			accept(tcp_server_socket, NULL, NULL);
			tcp_client_socket = pstrAccept->sock;
			recv(tcp_client_socket, gau8SocketTestBuffer, ..., 0);
		}
	}
	...
	case SOCKET_MSG_RECV:
	{
		tstrSocketRecvMsg *pstrRecv = (tstrSocketRecvMsg *)pvMsg;
		if (pstrRecv && pstrRecv->s16BufferSize > 0)
			send(tcp_client_socket, &msg_wifi_product, ..., 0);
	}
	...
	case SOCKET_MSG_SEND:
	{
		printf("socket_cb: send success!\r\n");
		printf("TCP Server Test Complete!\r\n");
		printf("close socket\n");
	}
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal.

```
-- ATWINC3400 TCP server example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
socket_cb: bind success!
socket_cb: listen success!
socket_cb: accept success!
socket_cb: recv success!
socket_cb: send success!
TCP Server Test Complete!
close socket
```

4.18 Protocol Example: NTP Time Client

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to retrieve the time information from the time server. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the chip and retrieve info.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Initialize the socket module and register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, create a UDP socket and bind it in the main loop.

```
if (udp_socket < 0) {
        udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
        if (udp_socket < 0) {
                continue;
        }

        /* Initialize default socket address structure. */
        addr_in.sin_family = AF_INET;
        addr_in.sin_addr.s_addr = _htonl(MAIN_DEFAULT_ADDRESS);
        addr_in.sin_port = _htons(MAIN_DEFAULT_PORT);

        bind(udp_socket, (struct sockaddr *)&addr_in, ...);
}
```

   Initialize the socket module and send an NTP time query to the NTP server in resolve_cb() function.

```
static void resolve_cb(uint8_t *pu8DomainName, uint32_t u32ServerIP)
{
        ...
        if (udp_socket >= 0) {
                addr.sin_family = AF_INET;
                addr.sin_port = _htons(MAIN_SERVER_PORT_FOR_UDP);
                addr.sin_addr.s_addr = u32ServerIP;

                ret = sendto(udp_socket, (int8_t *)&cDataBuf, ...);
        }
```

Receive NTP time from the server and convert it in the socket_cb() function.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	case SOCKET_MSG_BIND:
	{
		if (pstrBind && pstrBind->status == 0)
			ret = recvfrom(sock, gau8SocketBuffer, ..., 0);
	}
	...
	case SOCKET_MSG_RECVFROM:
	{
		uint32_t secsSince1900 = packetBuffer[40] << 24 |
			packetBuffer[41] << 16 | packetBuffer[42] << 8 |
			packetBuffer[43];

		const uint32_t seventyYears = 2208988800UL;
		uint32_t epoch = secsSince1900 - seventyYears;

		printf("socket_cb: The GMT time is %lu:%02lu:%02lu\r\n",
				(epoch  % 86400L) / 3600, (epoch  % 3600) / 60,
				epoch % 60);
	}
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 time client example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
m2m_ip_resolve_handler : DomainName pool.ntp.org
socket_cb: The GMT time is xx:xx:xx
```

WARNING If the server connection is unstable, the operation may fail to produce the result as illustrated.

4.19 Protocol Example: SMTP Send Email

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to send email for SMTP server. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the chip and send an email.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"

#define MAIN_SENDER_RFC         "<sender@gmail.com>"
#define MAIN_RECIPIENT_RFC      "<recipient@gmail.com>"

#define MAIN_TO_ADDRESS         "recipient@gmail.com"
#define MAIN_FROM_ADDRESS       "sender@gmail.com"
#define MAIN_FROM_PASSWORD      "12345678"
```

   Initialize the socket module and register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, try to connect the SMTP server. After then, smtpStatehandler will be executed sequentially until socket status become SocketComplete.

```
if (gu8SocketStatus == SocketInit) {
	if (tcp_client_socket < 0) {
		gu8SocketStatus = SocketWaiting;
		if (smtpConnect() != SOCK_ERR_NO_ERROR) {
			gu8SocketStatus = SocketInit;
		}
	}
} else if (gu8SocketStatus == SocketConnect) {
	gu8SocketStatus = SocketWaiting;
	if (smtpStateHandler() != MAIN_EMAIL_ERROR_NONE) {
		...
	}
} else if (gu8SocketStatus == SocketComplete) {
	printf("main: Email was successfully sent.\r\n");
	close_socket();
}
```

Connect to the socket and receive data following SMTP status.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	...
	case SOCKET_MSG_CONNECT:
	{
		if (pstrConnect && pstrConnect->s8Error >= SOCK_ERR_NO_ERROR)
			recv(tcp_client_socket, gcHandlerBuffer, ..., 0);
	}
	...
	case SOCKET_MSG_RECV:
	{
		switch (gu8SmtpStatus) {
			case SMTP_INIT:
				...
			case SMTP_HELO:
				...
			case SMTP_AUTH:
				...
			case SMTP_AUTH_USERNAME:
				...
			case SMTP_AUTH_PASSWORD:
				...
			case SMTP_FROM:
				...
			case SMTP_RCPT:
				...
			case SMTP_DATA:
				...
			case SMTP_MESSAGE_DATAEND:
				...
		}
	}
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 send email example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
Host IP is 173.194.72.108
Host Name is smtp.gmail.com
Recipient email address is recipient@gmail.com
main: Email was successfully sent.
```

WARNING For using the Gmail, the root certificate must be installed. For more details about downloading the root certificate, refer to the “Atmel-42640-Getting-Started-Guide-for-ATWINC3400WiFi-using-SAMD21-Xplained-Pro_UserGuide” document.

WARNING If the server connection is unstable, the operation may fail to produce the result as illustrated.

TIPS Limitations/known issues:
1. Email is sent to only one recipient.
2. Only plain text Email is supported.

4.20 Protocol Example: Location Client

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to get the location of the network provider. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the ATWINC3400 and get location information.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Configure the HTTP client.

```
configure_http_client();
```

   Get the default config data and specify the user configuration. Then http_client_init() and http_client_register_callback() function will be executed sequentially.

```
static void configure_http_client(void)
{
	...
	http_client_get_config_defaults(&httpc_conf);

	httpc_conf.recv_buffer_size = 256;
	httpc_conf.timer_inst = &swt_module_inst;
	httpc_conf.user_agent = "curl/7.10.6";

	ret = http_client_init(&http_client_module_inst, &httpc_conf);
	if (ret < 0) {
		while (1) {
		} /* Loop forever. */
	}

	http_client_register_callback(&http_client_module_inst, ...);
}
```

   Initialize the socket module and register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

After the device is connected to the AP, HTTP request will be sent.

```
static void wifi_callback(uint8_t msg_type, void *msg_data)
{
	...
	case M2M_WIFI_REQ_DHCP_CONF:
	{
		http_client_send_request(&http_client_module_inst, ...);
	}
```

Four operations will be executed sequentially.

```
static void http_client_callback(...)
{
	switch (type) {
	case HTTP_CLIENT_CALLBACK_SOCK_CONNECTED:
		printf("Connected\r\n");
		break;
	case HTTP_CLIENT_CALLBACK_REQUESTED:
		printf("Request complete\r\n");
		break;
	case HTTP_CLIENT_CALLBACK_RECV_RESPONSE:
		if (data->recv_response.content != NULL) {
			if (json_create(...) == 0 && json_find(...) == 0) {
				printf("Location : %s\r\n", loc.value.s);
			}
		}
		break;
	case HTTP_CLIENT_CALLBACK_DISCONNECTED:
		printf("Disconnected reason:%d\r\n", data->disconnected.reason);
		...
	}
}
```

The first sequence beginning with socket connected.

After request complete, third sequence will be executed and you can get the location data.

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 location client example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
Connected
Request complete
Received response 200 data size 178
Location : {latitude},{longitude}
Disconnected reason:-104
```

WARNING If disconnect reason is equal to -ECONNRESET(-104), it means the Server disconnected your connection due to the keep alive timeout. This is a normal operation.

This example obtains the location of your network provider, not your current position.

WARNING If the server connection is unstable, the operation may fail to produce the result as illustrated.

4.21 Advanced Example: Growl Notification

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro.

This example transmits a notification from the ATWINC3400 device (based on a certain trigger) to a public remote server, which in turn sends a phone application.

The initiated notification from the ATWINC3400 device is directed to a certain subscriber on the server. The supported applications are PROWL (for iPhone notifications) and NMA (for ANDROID notifications).

It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize growl and send notification message.

1. Code summary.
   Configure the below code in main.h for your account.

```
#define PROWL_API_KEY       "6ce3b9ff6c29e5c5b8960b28d9e987aec5ed603a"
#define NMA_API_KEY         "0757fe93214fc2cdf2ad42a5005ee0aa83a7a8ea242c0b80"
```

   Get the MAC address and set the device name with MAC address.

```
m2m_wifi_get_mac_address(gau8MacAddr);

set_dev_name_to_mac((uint8_t *)gacDeviceName, gau8MacAddr);
set_dev_name_to_mac((uint8_t *)gstrM2MAPConfig.au8SSID, gau8MacAddr);
m2m_wifi_set_device_name((uint8_t *)gacDeviceName, ...);
```

   Start provision mode.

```
m2m_wifi_start_provision_mode((tstrM2MAPConfig *)&gstrM2MAPConfig,
                                  (char *)gacHttpProvDomainName, 1);
```

   When your mobile device sends configuration information, the wifi_cb() function will be called with M2M_WIFI_RESP_PROVISION_INFO message and you can connect to the AP with the given information.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
       case M2M_WIFI_RESP_PROVISION_INFO:
              tstrM2MProvisionInfo *pstrProvInfo = (tstrM2MProvisionInfo *)pvMsg;
              if (pstrProvInfo->u8Status == M2M_SUCCESS) {
                     m2m_wifi_connect((char *)pstrProvInfo->au8SSID,
                            strlen((char *)pstrProvInfo->au8SSID),
                            pstrProvInfo->u8SecType,
                            pstrProvInfo->au8Password, M2M_WIFI_CH_ALL);
              }
```

After the device is connected to the AP, initialize the growl key and execute the message handler.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
	...
	case M2M_WIFI_REQ_DHCP_CONF:
	{
		...
		NMI_GrowlInit((uint8_t *)PROWL_API_KEY, (uint8_t *)NMA_API_KEY);
		growl_send_message_handler();
	}
	...
}
```

Notification message will be sent thru the below function.

```
static int growl_send_message_handler(void)
{
	...
	NMI_GrowlSendNotification(NMA_CLIENT, (uint8_t *)"Growl_Sample",
	(uint8_t *)"Growl_Event", (uint8_t *)"growl_test", NMA_CONNECTION_TYPE);
	return 0;
}
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 simple growl example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Provision Mode started.
Connect to [atmelconfig.com] via AP[WINC3400_08:CA] and fill up the page.
```

4. Connect your mobile device to the ATWINC3400 AP [WINC3400_08:CA].
5. Browse the webpage (atmel.com) to setup AP, complete the page, and then press Connect.
6. ATWINC3400 will be connected to the AP you entered.
7. Growl message will be sent.

```
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
wifi_cb: M2M_WIFI_RESP_PROVISION_INFO.
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
send Growl message
Growl CB : 20
```

This example supports sending GROWL notifications to the following servers.

- PROWL for iOS push notifications (https://www.prowlapp.com/)

- NMA for Android push notifications (http://www.notifymyandroid.com/)

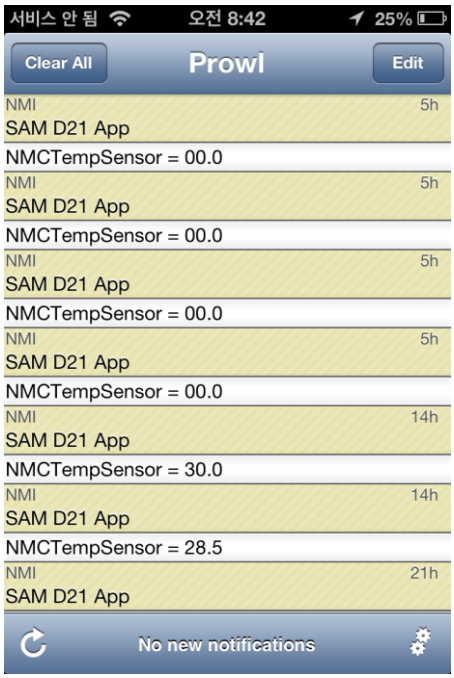
In order to enable the GROWL application (for sending notifications) working, you need to set your own API key to represent your account. Create your own key by following these instructions:

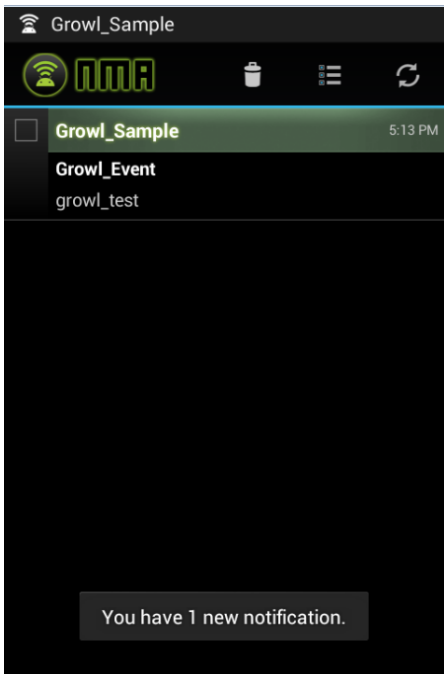
- Create an NMA account at http://www.notifymyandroid.com/ and create an API key. Copy the obtained key string in the file main.h in the macro NMA_API_KEY as the following.
- Create a PROWL account at https://www.prowlapp.com/ and create an API key. Copy the obtained API key string in the file main.h in the macro PROWL_API_KEY as the following.

```
#define NMA_API_KEY "f8bd3e7c9c5c10183751ab010e57d8f73494b32da73292f6"
#define PROWL_API_KEY "117911f8a4f2935b2d84abc934be9ff77d883678"
```

WARNING For using the growl, the root certificate must be installed. For more details about down-loading the root certificate, refer to the “Atmel-42640-Getting-Started-Guide-for-AT-WINC3400WiFi-using-SAMD21-Xplained-Pro_UserGuide” document.

Launch the Growl or NMA application to receive notification.





4.22 Advanced Example: MQTT Chat

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to chat using MQTT protocol. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the board, connect to an MQTT broker and chat with other devices.

1. Code summary.

   Configure the below code in main.h for the MQTT broker and the AP information to be connected.

```
static const char main_mqtt_broker[] = "test.mosquitto.org";

#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Configure the MQTT module. You can set the timer instance and register callback for MQTT messages.

```
/* Initialize the MQTT service. */
configure_mqtt();
```

```
static void configure_mqtt(void)
{
        ...
        mqtt_get_config_defaults(&mqtt_conf);

        mqtt_conf.timer_inst = &swt_module_inst;
        mqtt_conf.recv_buffer = mqtt_buffer;
        mqtt_conf.recv_buffer_size = MAIN_MQTT_BUFFER_SIZE;

        result = mqtt_init(&mqtt_inst, &mqtt_conf);

        result = mqtt_register_callback(&mqtt_inst, mqtt_callback);
```

   Set up the user name first and then the topic value will be set with MAIN_CHAT_TOPIC + user name.

```
printf("Enter the user name (Max %d characters)\r\n", MAIN_CHAT_USER_NAME_SIZE);
scanf("%s", mqtt_user);
printf("User : %s\r\n", mqtt_user);
sprintf(topic, "%s%s", MAIN_CHAT_TOPIC, mqtt_user);
```

Initialize the socket module and the register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

After the device is connected to the AP, call the mqtt_connect() function to connect the socket.

```
static void wifi_callback(uint8 msg_type, void *msg_data)
{
	...
	case M2M_WIFI_REQ_DHCP_CONF:
		...
		mqtt_connect(&mqtt_inst, main_mqtt_broker);
}
```

MQTT callback will receive MQTT_CALLBACK_SOCK_CONNECTED message then start sending CONNECT message to the MQTT broker.

```
static void mqtt_callback(struct mqtt_module *module_inst, int type, union
mqtt_data *data)
{
	...
	case MQTT_CALLBACK_SOCK_CONNECTED:
	{
		mqtt_connect_broker(module_inst, 1, NULL, NULL, mqtt_user,
				NULL, NULL, 0, 0, 0);
```

MQTT callback will receive the MQTT_CALLBACK_CONNECTED message then register subscription with a specific topic.

```
static void mqtt_callback(struct mqtt_module *module_inst, int type, union
mqtt_data *data)
{
	...
	case MQTT_CALLBACK_CONNECTED:
	{
		mqtt_subscribe(module_inst, MAIN_CHAT_TOPIC "#", 0);
```

If other device sends a message with this topic then MQTT callback will receive the MQTT_CALLBACK_RECV_PUBLISH message.

```
static void mqtt_callback(struct mqtt_module *module_inst, int type, union
mqtt_data *data)
{
	...
	case MQTT_CALLBACK_RECV_PUBLISH:
```

If the user inputs some string via terminal, publish the MQTT message as below.

```
static void check_usart_buffer(char *topic)
{
	if (uart_buffer_written >= MAIN_CHAT_BUFFER_SIZE) {
		mqtt_publish(&mqtt_inst, topic, uart_buffer,
			MAIN_CHAT_BUFFER_SIZE, 0, 0);
		uart_buffer_written = 0;
	}
```

2. Build the program and download it into the board.
3. Start the application.
4. On the terminal window, enter the user name through the terminal window.

```
-- ATWINC3400 Wi-Fi MQTT chat demo --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Preparation of the chat has been completed.
```

WARNING The user name should not contain a white space.

5. After chatting initialization is completed, enjoy the chat.

TIPS The initialization may take longer than a few minutes according to the network environment.

RESULT Application is now successfully running. Below shows two users talking to each other's device connected to the same MQTT chat server.

```
Enter the user name (Max 64 characters)
User : demo_user
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
Preparation of the chat has been completed.
demo_user >> hi!
demo_user >> anybody there?
other_user >> I'm here
other_user >> hi
```

WARNING Maximum message length should be shorter than 128 bytes.

WARNING If the server connection is unstable, the operation may fail to produce the result as illustrated.

4.23 Advanced Example: Weather Client

This example demonstrates the use of the ATWINC3400 with the SAM D21 Xplained Pro board to retrieve weather information from a weather server (openweathermap.org). It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize the chip and retrieve info.

1. Code summary.
   Configure the below code in main.h for the AP information to be connected.

   Initialize the socket module and register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

   Get the MAC address and set the device name with the MAC address.

```
m2m_wifi_get_mac_address(gau8MacAddr);

set_dev_name_to_mac((uint8_t *)gacDeviceName, gau8MacAddr);
set_dev_name_to_mac((uint8_t *)gstrM2MAPConfig.au8SSID, gau8MacAddr);
m2m_wifi_set_device_name((uint8_t *)gacDeviceName, ...);
```

   Start provision mode.

```
m2m_wifi_start_provision_mode((tstrM2MAPConfig *)&gstrM2MAPConfig,
                              (char *)gacHttpProvDomainName, 1);
```

   When your mobile device sends configuration information, the wifi_cb() function will be called with the M2M_WIFI_RESP_PROVISION_INFO message and you can connect to the AP with the given information.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
       case M2M_WIFI_RESP_PROVISION_INFO:
              tstrM2MProvisionInfo *pstrProvInfo = (tstrM2MProvisionInfo *)pvMsg;
              if (pstrProvInfo->u8Status == M2M_SUCCESS) {
                     m2m_wifi_connect((char *)pstrProvInfo->au8SSID,
                            strlen((char *)pstrProvInfo->au8SSID),
                            pstrProvInfo->u8SecType,
                            pstrProvInfo->au8Password, M2M_WIFI_CH_ALL);
              }
```

After the device is connected to the AP, the gethostbyname() function will be called.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
	...
	case M2M_WIFI_REQ_DHCP_CONF:
	{
		...
		gbConnectedWifi = true;
		gethostbyname((uint8_t *)MAIN_WEATHER_SERVER_NAME);
	}
	...
}
```

Create a TCP client socket and connect to the server in the main loop.

```
if (gbConnectedWifi && !gbTcpConnection) {
	if (gbHostIpByName) {
		if (tcp_client_socket < 0) {
			if ((tcp_client_socket = socket(...)) < 0) {
				continue;
			}
		}

		if (connect(tcp_client_socket, ...) != SOCK_ERR_NO_ERROR) {
			continue;
		}

		gbTcpConnection = true;
	}
}
```

The socket callback function will receive the SOCKET_MSG_CONNECT message. Then it requests weather information to the server with a city name.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	case SOCKET_MSG_CONNECT:
		sprintf(gau8ReceivedBuffer, ..., MAIN_CITY_NAME, ...);
		...
		send(tcp_client_socket, gau8ReceivedBuffer, ...);
		recv(tcp_client_socket, ...);
	break;

	case SOCKET_MSG_RECV:
	{
	}
}
```

The socket callback function will receive the SOCKET_MSG_RECV message with weather information.

You can also get the current temperature via the IO1 sensor board as below.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
	case SOCKET_MSG_RECV:
		...
		getTemperature();
}
```

```
static void getTemperature(void)
{
	...
	double s8SensorTemperature = at30tse_read_temperature();
```

2. Build the program and download it into the board.
3. Start the application.

RESULT The application is now programmed and running. The following information will be displayed on the terminal window.

```
-- ATWINC3400 weather client example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx –
Provision Mode started.
Connect to [atmelconfig.com] via AP[WINC3400_08:CA] and fill up the page.
```

4. Connect your mobile device to the ATWINC3400 AP [WINC3400_08:CA].
5. Browse the webpage (atmel.com) to setup AP, complete the page, and then press Connect.
6. ATWINC3400 will be connected to the AP what you entered.
7. The weather info will be printed.

```
Wi-Fi IP is xxx.xxx.xxx.xxx
wifi_cb: M2M_WIFI_RESP_PROVISION_INFO.
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx
Host IP is 144.76.102.166
Host Name is openweathermap.org
City: Seoul
Weather Condition: sky is clear

Temperature from sensor : 27 degrees
Temperature from server : 3 degrees
Temperature difference : 24 degrees
```

WARNING If the server connection is unstable, the operation may fail to produce the result as illustrated.

4.24 Advanced Example: Wi-Fi Serial

This example demonstrates how to emulate serial ports between two devices. It reads input data from serial interface and sends it via Wi-Fi connection and the terminal window will print the messages which you typed or received. It can be useful for chatting or controlling a remote device. It uses the following hardware and you need to prepare two pairs of SAM D21 and ATWINC3400 boards.

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize devices and USART interface. Create TCP sockets, send/receive messages and print them on the terminal window.

1. Code summary.

   Configure the below code in main.h for the AP information to be connected.

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
```

   Configure the USART module to read the user input data.

```
static void configure_console(void)
{
        ...
        stdio_serial_init(&cdc_uart_module, CONF_STDIO_USART_MODULE,
                        &usart_conf);
        /* Register USART callback for receiving user input. */
        usart_register_callback(&cdc_uart_module, uart_callback,
                        SART_CALLBACK_BUFFER_RECEIVED);
        usart_enable_callback(&cdc_uart_module, USART_CALLBACK_BUFFER_RECEIVED);
        usart_enable(&cdc_uart_module);
}
```

   Initialize the socket module and register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, create TCP server socket and bind it in the main loop.

```
if ((tcp_server_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        continue;
}
...
bind(tcp_server_socket, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
```

If there is user input data then the handle_input_message() function in the main loop calls the parse_command() function to parse the user data and execute the handler function according to the command.

```
uint8_t parse_command(char *buffer, uint8_t remote)
{
	for (i = i_st; i < i_ed; i++) {
		if (!strcmp(cmd_list[i].cmd_string, cmd_buf)) {
			cmd_list[i].cmd_handler(
					buffer + strlen(cmd_list[i].cmd_string) + 1);
			break;
		}
	}
```

Or the handle_input_message() function prints the user data in the terminal window and sends it to the remote device.

```
void handle_input_message(void)
{
	...
		if (tcp_connected == 1) {
			PRINT_LOCAL_MSG(uart_buffer);
			send(tcp_client_socket, uart_buffer, msg_len + 1, 0);
		}
```

When receiving data from the remote device, it handles the received message to print it or executer a proper command.

There are several commands for Wi-Fi serial functionality in this example.

2. Build the program and download it into the board.
3. Start the application.
4. On the terminal window, the following text should appear:

```
-- ATWINC3400 Wi-Fi Serial example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
Wi-Fi connected.
Wi-Fi IP is xxx.xxx.xxx.xxx
socket_cb: bind success.
socket_cb: listen success.
```

5. Check the IP address of each board and execute connection on one device by typing the below command on the terminal window with the other device's address. Use prefix "<<" to execute local commands.

```
<<connect xxx.xxx.xxx.xxx
```

6. If connected, the following text should appear:

```
(Local device)
Connecting to [xxx.xxx.xxx.xxx] ...
socket_cb: connect success.

(Remote device)
socket_cb: accept success.
```

7. Type messages on the terminal window and you will see the sent/received messages.
8. You can control the LED on the remote device by typing the following command. Use prefix ">>" to execute remote commands.

```
>>control ledon
(Or)
>>control ledoff
```

RESULT Application reads data from the serial interface.

TIPS User commands can be modified to execute various actions.

4.25 Advanced Example: OTA Firmware Upgrade

This example demonstrates how to upgrade ATWINC3400 Firmware via OTA. It downloads ATWINC3400 firmware from the OTA Download server. The OTA Download server is a web server. You can upload a new firmware image and download it to your device. It uses the following hardware:

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize devices and set the server information. It connects to the OTA Download Server.

1. Set your OTA Download server.
2. Upload the OTA firmware binary to the root folder in your server. (e.g. http://192.168.0.137/m2m_ota_3400.bin)
3. Code summary.

   Configure the below code in main.h for the AP and for the OTA download server."

```
#define MAIN_WLAN_SSID          "DEMO_AP"
#define MAIN_WLAN_AUTH          M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK           "12345678"
#define MAIN_OTA_URL            "http://192.168.0.137/m2m_ota_3400.bin"
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   Initialize the OTA function.

```
m2m_ota_init(OtaUpdateCb, OtaNotifCb);
```

   After the device is connected to the AP, m2m_ota_start_update() function will be executed in the wifi_cb() function.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
        ...
        case M2M_WIFI_REQ_DHCP_CONF:
                m2m_ota_start_update((uint8_t *)MAIN_OTA_URL);
```

If successfully downloaded, the m2m_ota_switch_firmware() function will be called thru OtaUpdateCb().Then if the switch is successful, the system_reset() function will be called to reset both SAM D21 and ATWINC3400.

```
static void OtaUpdateCb(uint8_t u8OtaUpdateStatusType, uint8_t u8OtaUpdateStatus)
{
	if(u8OtaUpdateStatus == OTA_STATUS_SUCSESS) {
		if(u8OtaUpdateStatusType == DL_STATUS) {
			m2m_ota_switch_firmware();
		} else if(u8OtaUpdateStatusType == SW_STATUS) {
			system_reset();
		}
	}
}
```

4. Build the program and download it into the board.
5. Start the application.
6. The following text should appear on the SAM D21 terminal window:

```
-- ATWINC3400 OTA Firmware upgrade example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --

m2m_wifi_connect.
Wi-Fi connected
Wi-Fi IP is xxx.xxx.xxx.xxx.
```

When you get the IP address the OTA get started.

7. At the start of OTA, the following text should appear on the ATWINC3400 terminal window :

```
(60460)(M2M)(OTA) Start Update http://192.168.2.4:80/m2m_ota_3400.bin
(60460)(M2M)Sock(6) Conn...
(60570)(M2M)Sock created(6)
(60620)(M2M)CS update 14
(60630)(M2M)(OTA) Invalidate RB Image
(60640)(M2M)Start Writing.. at 80000 Size 354312
```

8. At the end of successful OTA, the following text should appear on the ATWINC3400 terminal window:

```
 (22350)(M2M)Verification OK
(22350)(M2M)OTA SW  VERSION 1.0.2
(22350)(M2M)OTA HIF LEVEL 1
(22350)(M2M)OTA Build date Jan 12 2016 Time 16:17:56
(22370)(M2M)CS update 22
(22380)(M2M)(OTA) Update image done successfully
(22380)(M2M)C-S<6>
(22420)(M2M)CS update 24

 (0)NO CORTUS app
```

RESULT After successful OTA download and switch, the ATWINC3400 is restarted by the SAM D21.

4.26 Advanced Example: SSL connection

This example demonstrates how to set up an SSL connection.

- The SAM D21 Xplained Pro
- The ATWINC3400 on EXT1 header

main.c: Initialize devices and connect to a server using SSL.

1. Code summary.

   Configure the below code in the main.h for AP and the server information to be connected.

```
#define MAIN_WLAN_SSID                   "DEMO_AP"
#define MAIN_WLAN_AUTH                   M2M_WIFI_SEC_WPA_PSK
#define MAIN_WLAN_PSK                    "12345678"

#define MAIN_SSL_BUF_LEN                 1024
#define MAIN_HOST_NAME                   "www.google.com"
#define MAIN_HOST_PORT                   443
```

   Initialize the socket module and register socket callback function.

```
socketInit();
registerSocketCallback(socket_cb, resolve_cb);
```

   Connect to the AP.

```
m2m_wifi_connect((char *)MAIN_WLAN_SSID, ...
```

   After the device is connected to the AP, the gethostbyname() function will be executed.

```
static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
        ...
        case M2M_WIFI_REQ_DHCP_CONF:
                gethostbyname((uint8_t *)MAIN_HOST_NAME);
```

   In the main loop, try to connect to the SSL server.

```
if (sslConnect() != SOCK_ERR_NO_ERROR) {
        gu8SocketStatus = SocketInit;
}
```

   If successfully connected, the socket_cb() function will be called with the SOCKET_MSG_CONNECT message.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
{
        switch (u8Msg) {
        case SOCKET_MSG_CONNECT:
                if (pstrConnect && pstrConnect->s8Error >= SOCK_ERR_NO_ERROR)
                        printf("Successfully connected.\r\n");
        break;
```

2. Build the program and download it into the board.
3. Start the application.
4. The following text should appear on the terminal window:

```
-- ATWINC3400 SSL example --
-- SAMD21_XPLAINED_PRO --
-- Compiled: xxx xx xxxx xx:xx:xx --
wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED: CONNECTED
wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is xxx.xxx.xxx.xxx
Host IP is 173.194.127.115
Host Name is www.google.com
Successfully connected.
```

RESULT The device is connected to a server using SSL.

WARNING To set up an SSL connection, a root certificate must be installed. For more details about downloading the root certificate, refer to the “Atmel-42640-Getting-Started-Guide-for-AT-WINC3400WiFi-using-SAMD21-Xplained-Pro_UserGuide” document.

5 Conclusion

This software programming guide explains examples of how to use the Atmel ATWINC3400 Wi-Fi module along with the Wi-Fi Software API for the SAM D21 Xplained Pro board.

The following topics have been covered:

- Basic example of controlling the ATWINC3400 module
- Simple protocol examples
- IoT application example

You have seen how the ATWIN3400 Wi-Fi module makes it easy to add Wi-Fi capabilities to a SAM D21 host MCU.

6 References

Atmel-42497ATWIN3400-BLE-Provisioning-Demo-Setup: http://www.atmel.com/images/atmel-42497-atwinc3400-ble-provisioning-setup-and-usage_userguide.pdf

7 Revision History

Doc Rev.	Date	Comments
42369B	03/2016	1. Globally changed references to Atmel Studio to say version 7.0. 2. Updated description in Section 4.1. 3. Revised Section 4.26. 4. Globally removed referenced of P2P as it is not supported.
42369A	12/2015	Initial document release.

Atmel | Enabling Unlimited Possibilities®

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