eLearning

Wi-Fi® SoC Module *WFI32E01*
介紹及軟體應用篇

MICROCHIP

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

SMART | CONNECTED | SECURE

King Chou, Pr. ESE

2021

Microchip - Chinese [Traditional]

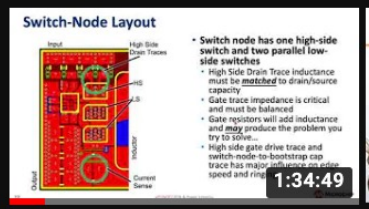
199 位訂閱者

已訂閱

首頁 影片 播放清單 頻道 討論 簡介

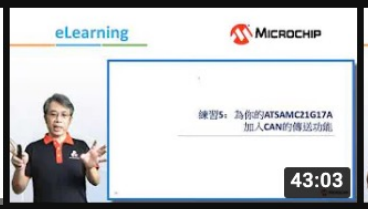
上傳的影片

全部播放



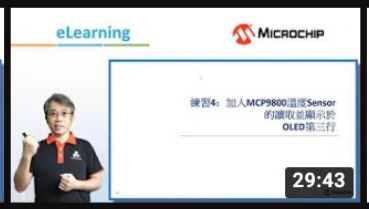
ePOW007——電源分佈網路與電源完整性

觀看次數：29次・3 週前



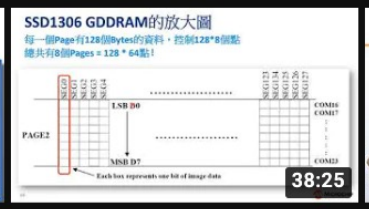
ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：97次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：28次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：31次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：39次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：27次・1 個月前

Featured Channels

Microchip Technology

5.38萬 位訂閱者

Microchip - Chinese [Simplified]

540 位訂閱者

Microchip - Japanese

343 位訂閱者

Microchip - Korean

1310 位訂閱者

Microchip Makes

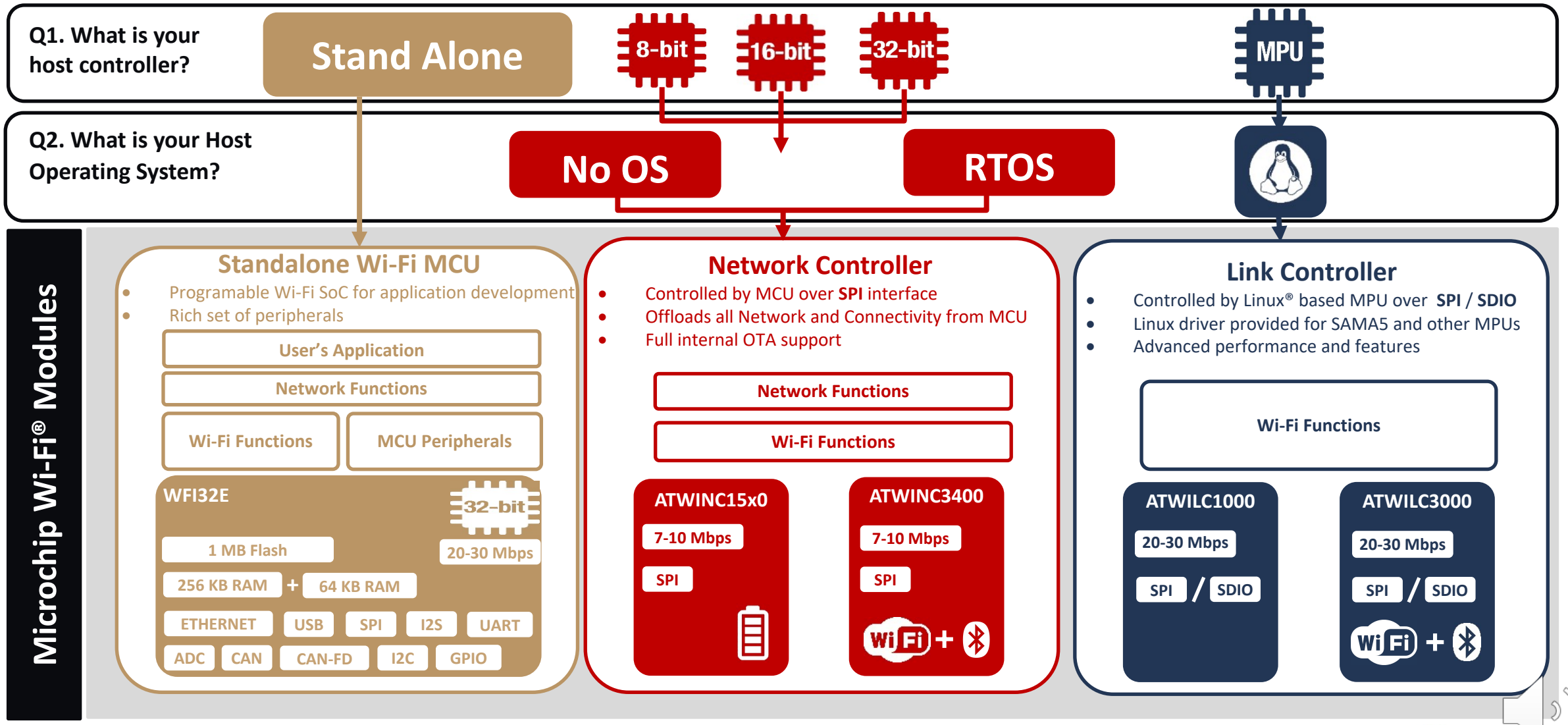
1.87萬 位訂閱者

Agenda

eLearning

- **Hardware Feathers**
- **Software Introduction**
  - Software Architecture
  - WLAN Driver
  - Wireless Service

Microchip Wi-Fi® Overview



eLearning

WFI32E01

Hardware Introduction

MICROCHIP

Hardware Feathers

- **200 MHz, MIPS 32-bit M-Class Microprocessor Core**
  - I-Cache (16 KB) & D-Cache (16 KB)
  - DSP-enhanced Core
  - microMIPS™ Mode for Up to 35% Smaller Code Size
- **1 MB Embedded Flash, 256 KB SRAM & 64 KB Wi-Fi Data Buffer**
- **USB FS2.0, CAN, CAN-FD & 10/100 Ethernet， 12 Bit dual channel A/D**
- **GPIO (Silicon: 62, Module: 37)**
- **Operating conditions: - 3.0V to 3.6V, -40˚C to +85˚C**
- **WLAN：**
  - 802.11 b/g/n
  - SISO (2.4 GHz), 20 MHz Bandwidth
  - WPA/WPA2/WPA3 Personal, TLS 1.3 / SSL (wolfssl)

Hardware Feathers

- **Hardware Accelerated Security Modes (with Built-in DMA Support)**
  - Crypto Engine with TRNG for Data Encryption/decryption
  - AES, 3DES, SHA, MD5 and HMAC
  - AES Modes: ECB, CBC, CTR, CFB, OFB, GCM
  - HW Public Key Cryptography with Support for:
    - 16-DSP multipliers configuration
    - 256-bit ECC/ECDH/ECDSA/Curve25519
    - 256-bit Ed25519 - 512-bit ECC/ECDH/ECDSA generation

- **Integrated ECC608 Trust&GO**

PIC32MZ1025W104

Block Diagram for 132-Pin QFN Chip

1 MB Embedded Flash

WWDT / DMT

320 KB RAM
- **256 KB SRAM**
- **64 KB DBF**

MIPS uAptiv UP Core
- 200 MHz
- FPU / DSP / DMA
- ICSP™ / Debug
- Secure Boot
- XLP Low Power Controller

Encryption Engine

Data Crypto – AES, TDES
Public Key – ECC Curve25519 (ECDH, ECDSA)

Peripherals / Clocks
- 8 MHz Internal Osc. LPRC, 4xREFO, RTC
- 2x12-bit ADC 20 channels w/CVD
- 4x Capture/Compare/PWM
- 6x32-bit Timers
- 16-bit Timer
- Peripheral Trigger Gen.
- Die Temperature Sensor

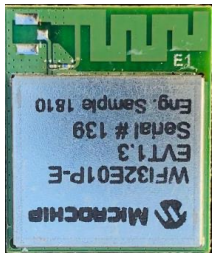
Interfaces

Wi-Fi® 802.11 b/g/n w/ Low Power Modes	USB 2.0 FS	Serial I/O 3xUART I^2C 2xSPI SQI 2xI^2S	10/100 Ethernet RMII IEEE1588	CAN & CAN-FD	62 GPIO

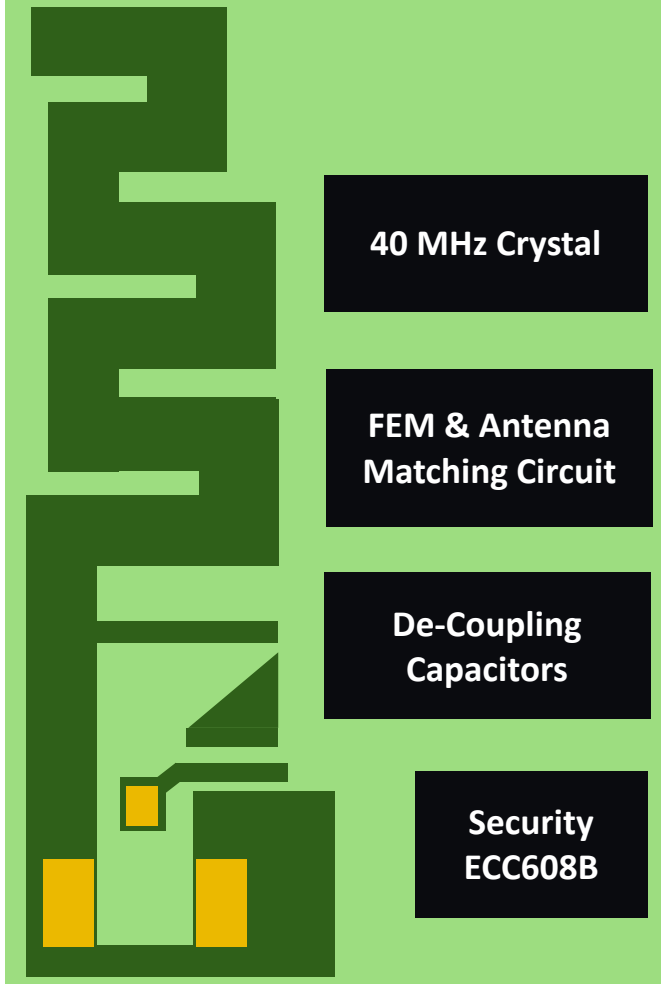
- **IC Package:**
  - 132 pins DQFN
  - 10 x 10 x 0.9 mm
- **Single Supply Voltage**
  - Operating Voltage: 3.0V to 3.6V
- **Operating Temperature:** -40°C to +105°C

WFI32E01UE

Block Diagram for 54-Pad Module



- **Module Package:**
  - 54 pins SMD
  - 24.5 x 20.5 x 2.5 mm
- **Single Supply Voltage**
  - Operating Voltage: 3.0V to 3.6V
- **Operating Temperature:** -40°C to +85°C



Encryption Engine
Data Crypto – AES, TDES
Public Key – ECC, ECHD, ECDSA, Curve25519, Ed25519

MIPS uAptiv UP Core
- 200 MHz
- FPU / DSP / DMA
- ICSP™ / Debug

Wi-Fi® 802.11 b/g/n
w/ Low Power Modes

1 MB Embedded Flash

320 KB RAM
- 256 KB SRAM
- 64 KB Wi-Fi Data Buffer

Die Temp Sensor

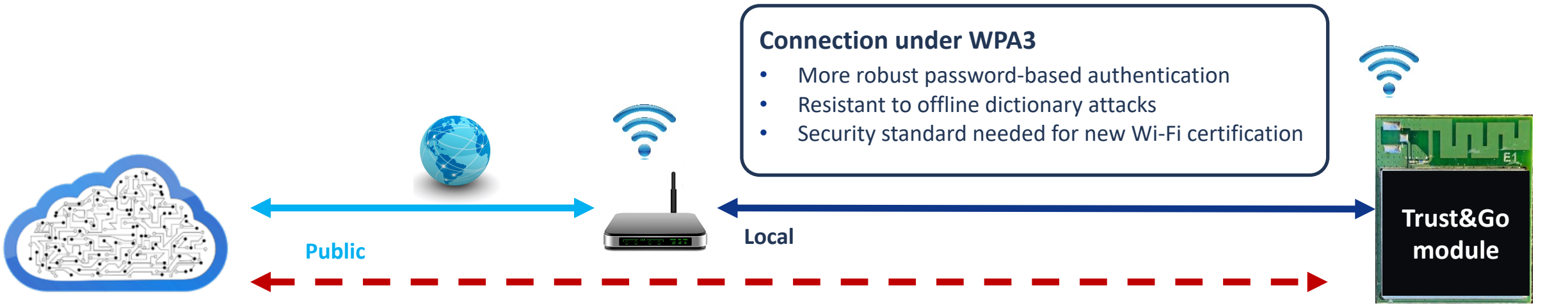
WWDT / DMT

Peripherals & Interfaces

- 8 MHz Internal Osc.
- LPRC, REFO
- 2x12-bit ADC, 1 MSPS, 12 ch.
  - Supports CVD Touch
- 6x32-bit / 16-bit Timers
- 4xCapture/Compare/PWM
- Peripheral Trigger Generator
- USB 2.0 Full Speed
- 10/100 Ethernet (RMII, IEEE1588)
- 3xUART/2xSPI/I²C/SQI/2xI²S
- CAN & CAN-FD
- 37 GPIO

Build a Secure and Trustful IoT Connection

Trust&Go Wi-Fi® module: WFI32E01PC-I



Pre-provisioned Certs for multiple Clouds

amazon web services | Partner Network
ADVANCED TECHNOLOGY PARTNER
IoT COMPETENCY

Microsoft Azure

Google Cloud Platform

Hardened TLS with Cryptographic key pairs

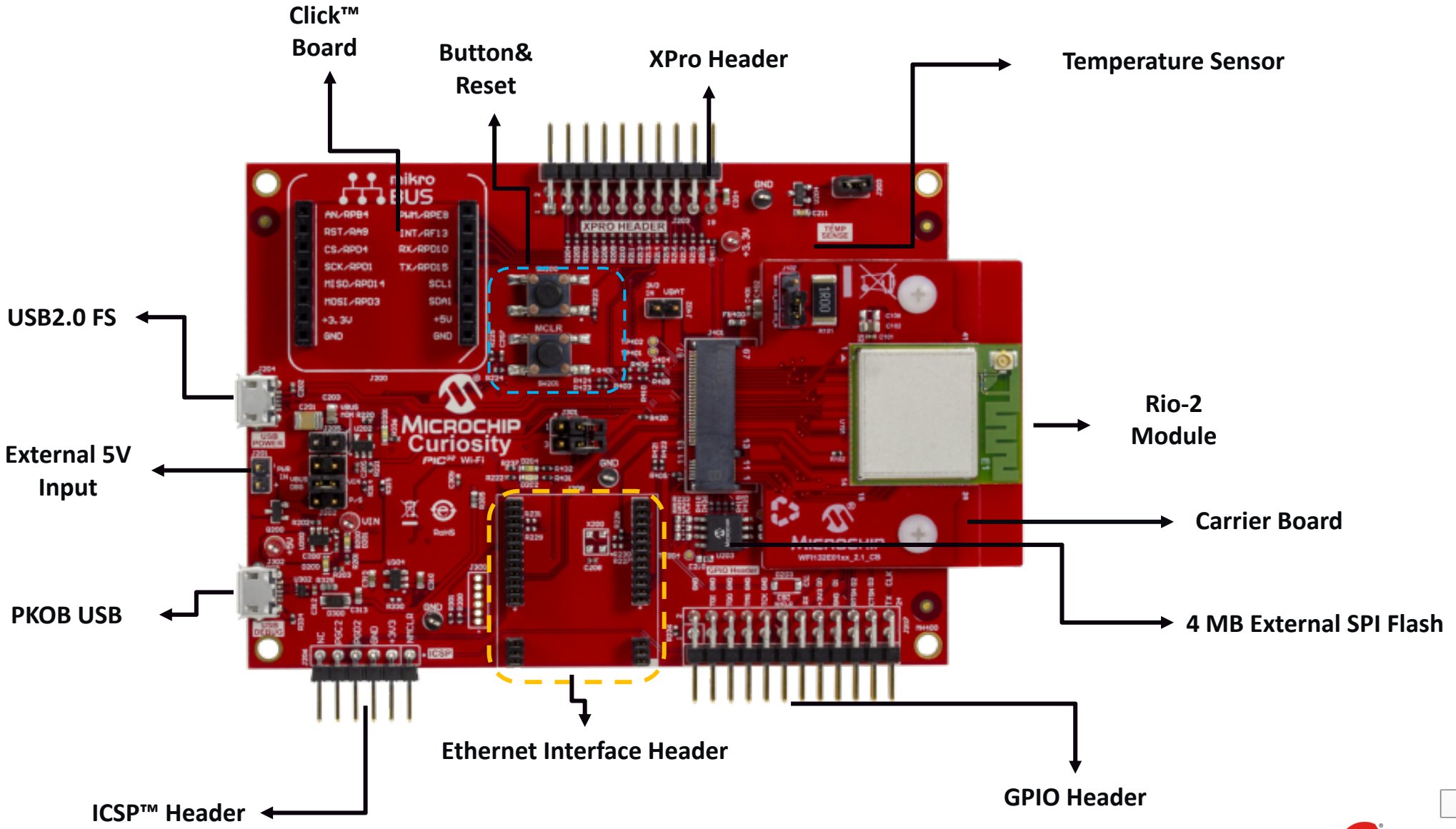
Immutable Public Key

Invisible Private key, Anti-physical Attack

Unique, **Verifiable**, and **Trustable** Device Identity

MICROCHIP

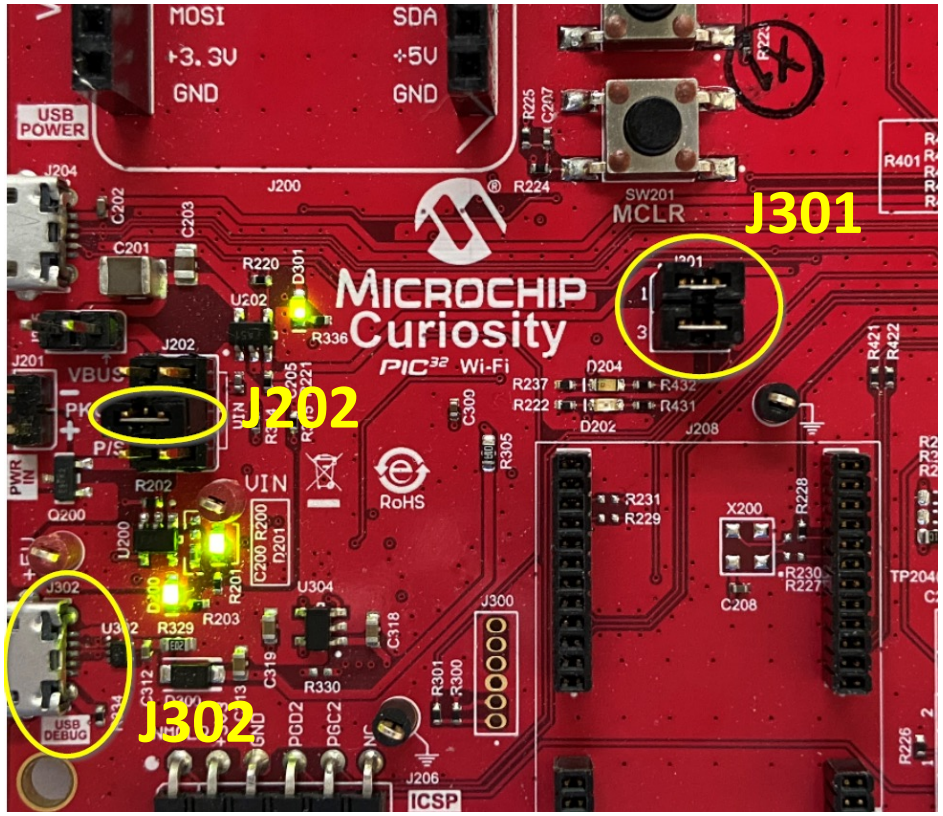
PIC32MZ-W1 Curiosity Board



PIC32MZ-W1 Curiosity Board

On-board debugger setup

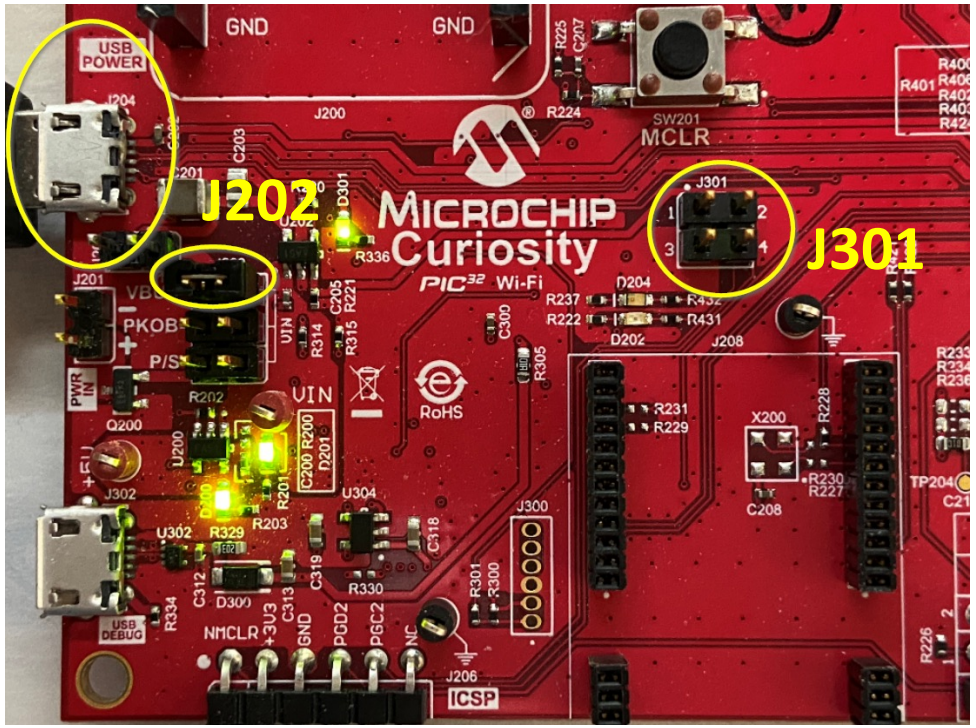
- The PKOB debugger enables the user to program/debug through Debug USB (J302) on PIC32WFI32E Curiosity Board.
  1. Make sure J202 jumper is connected to PKOB
  2. Make sure J301 jumpers are shorted (Pins 1-2 shorted, Pins 3-4 shorted)
  3. Connect the USB Cable between *USB Debug J302* and your PC



PIC32MZ-W1 Curiosity Board

External debugger setup (optional)

- MPLAB® ICD 4 In-Circuit Debugger/Programmer is Microchip's fastest, cost-effective debugging and programming tool for PIC32MZ W1 family.
  1. Make sure that J202 jumper is connected to VBUS
  2. Make sure J301 jumpers are open (Pins 1-2 open, Pins 3-4 open)
  3. To power supply the board , connect the USB Cable between *USB Power J204* and your PC
  4. Connect the MPLAB ICD 4 tool between ICSP™ J206 connector and your PC



PIC32MZ-W1 Curiosity Board

External debugger setup (optional)

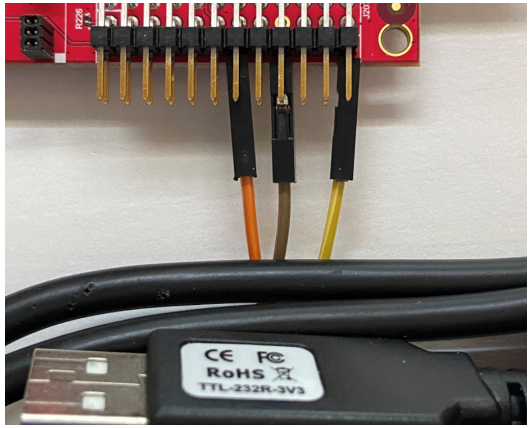
- Connect the MPLAB® ICD 4 tool between ICSP™ J206 connector and your PC

PIC32MZ-W1 Curiosity Board

USB-to-UART converter

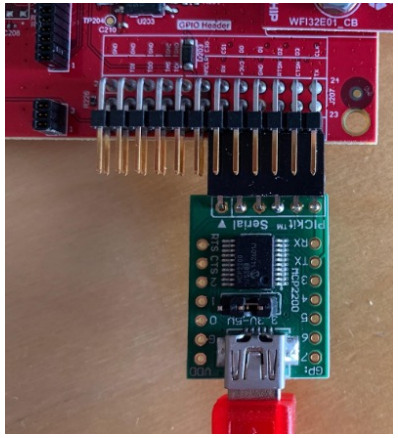
- To observe the application logs, you can connect a USB-to-UART converter to J207 (GPIO Header) of the Curiosity Board. Application debug logs can be captured from UART1.
  - The UART1 pins (Tx and Rx) are marked on the GPIO Header silkscreen.
    Serial port settings: *115200 8N1*

Setup with FDTI cable



USB-UART Cable	GPIO Header J207 of the Curiosity Board
Tx	**UART1**_RX (Pin 13)
Rx	UART1_TX (Pin 23)
Ground	GND (Pin 17)

Setup with PICKIT Serial



PICKIT Serial header	GPIO Header J207 of the Curiosity Board
Tx (Pin 1)	UART1_RX (Pin 13)
Rx (Pin 6)	UART1_TX (Pin 23)
Ground (Pin 3)	GND (Pin 17)

Product Families

ICs	
IC with dual-CAN, Crypto, 1 MB Flash	PIC32MZ1025W104132-V/NX
Modules	
Module with PCB and Shield, CAN, 1 MB Flash	WFI32E01PE-I
Module with U.FL and Shield, CAN, 1 MB Flash	WFI32E01UE-I
Module with PCB and Shield, CAN, 1 MB Flash and Trust&Go	WFI32E01PC-I
Module with U.FL and Shield, CAN, 1 MB Flash and Trust&Go	WFI32E01UC-I

eLearning

Software Architecture of WFI32E01

Software Introduction

MICROCHIP

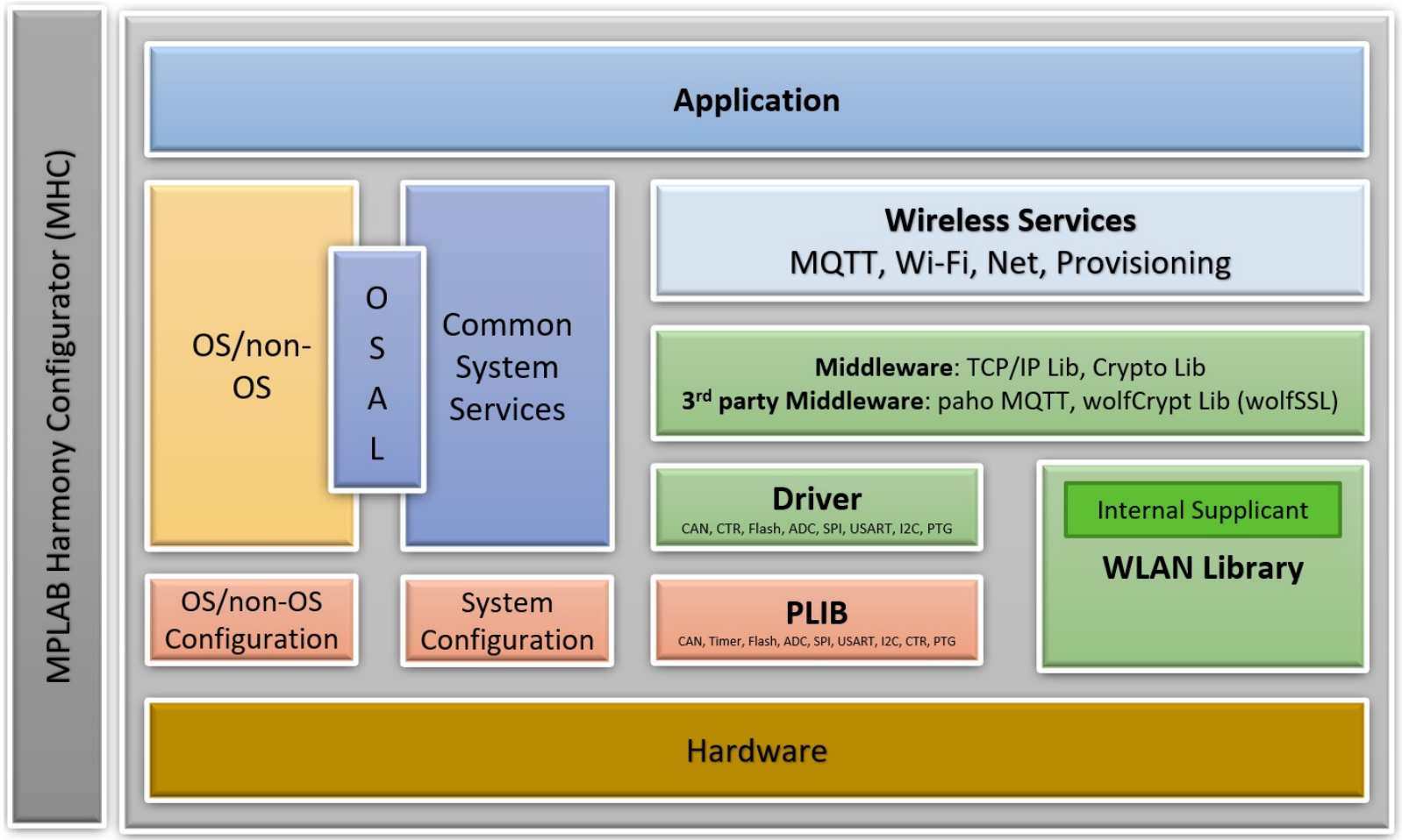
Software Overview

- Support MPLAB® Harmony 3 Framework
- WLAN subsystem support STA (**STA**tion) and SoftAP (**A**ccess **P**oint) mode
- Up to 8 STAs supported in SoftAP mode
- Wi-Fi® Security protocols supported: WPA/WPA2/WPA3 Personal, TLS 1.3 / SSL (wolfssl)
- TCP and UDP can support application like HTTP Server, DHCP Client and Server, DNS, FTP, SNMP, SNTP, SMTP, NDP, NBNS, iPerf, reboot

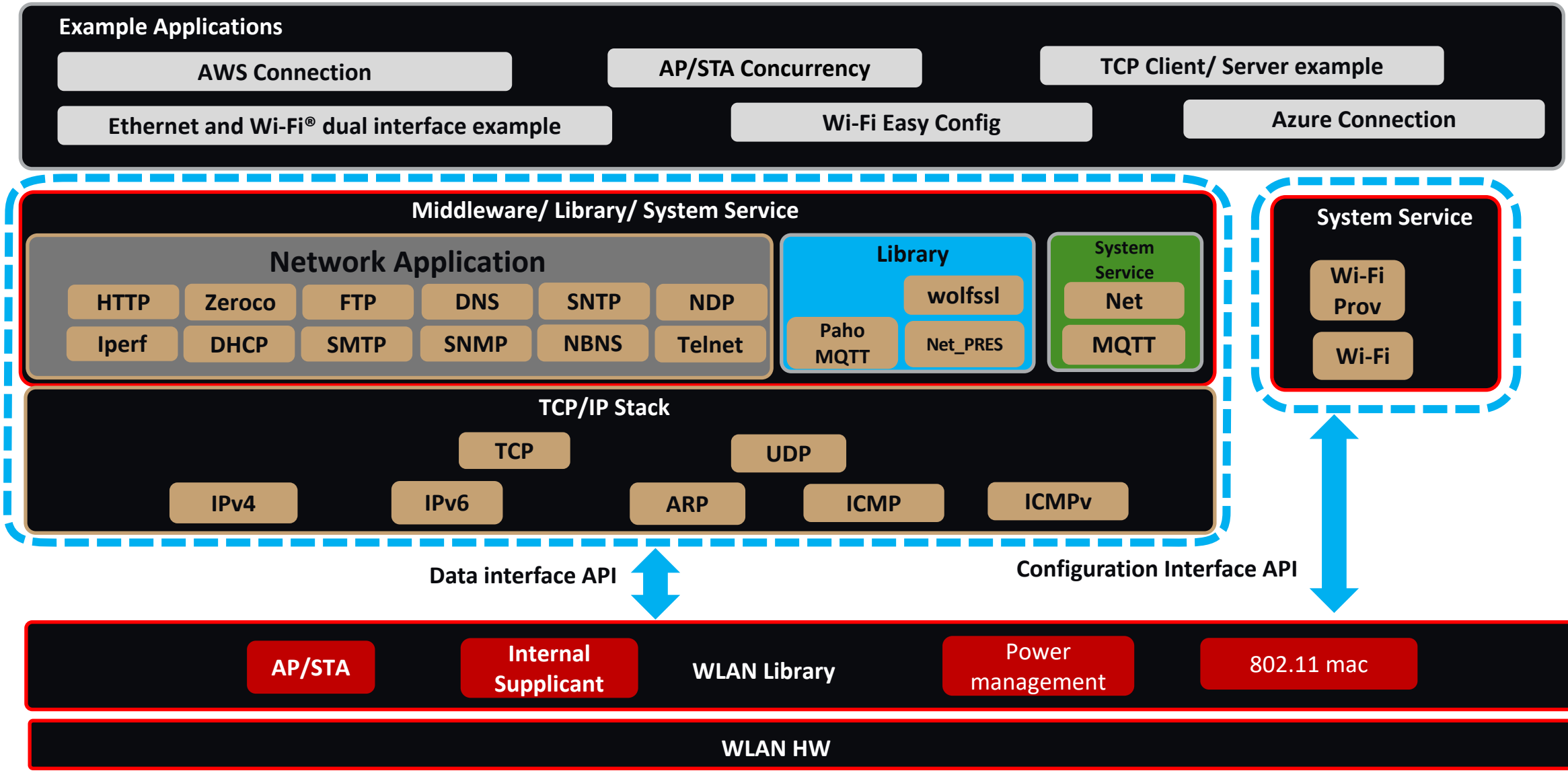
Software Overview

- Support Network layer IPv4 and IPv6
- Support FreeRTOS Operating System or other 3rd party OS that supported in MPLAB® Harmony 3, and non-OS environment as well
- Support of Wi-Fi® related System Service for easy development
- Support system shell over serial interface UART (CLI)
- Support “Wi-Fi Easy Connect” feature, using web-browser to do network provisioning

Software Architecture



Software Architecture



eLearning

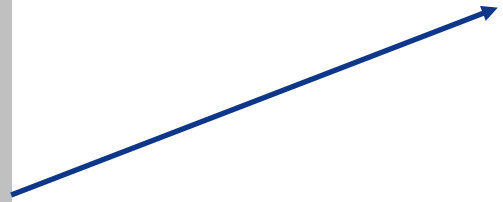
System Initialization

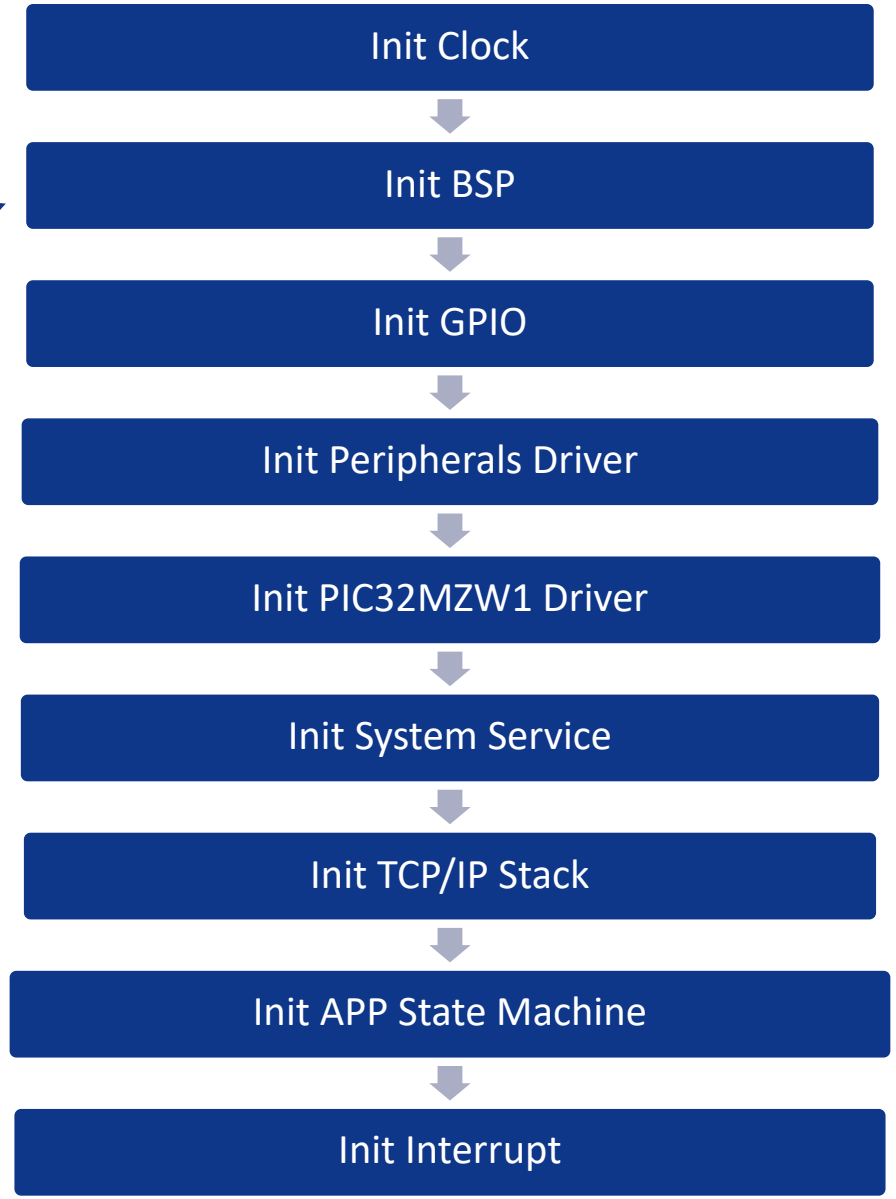
Software Introduction

MICROCHIP

System Initialization

```
int main ( void )
{
   SYS_Initialize ( NULL );
   while ( true )
   {
      SYS_Tasks ( );
   }
   return ( EXIT_FAILURE );
}
```

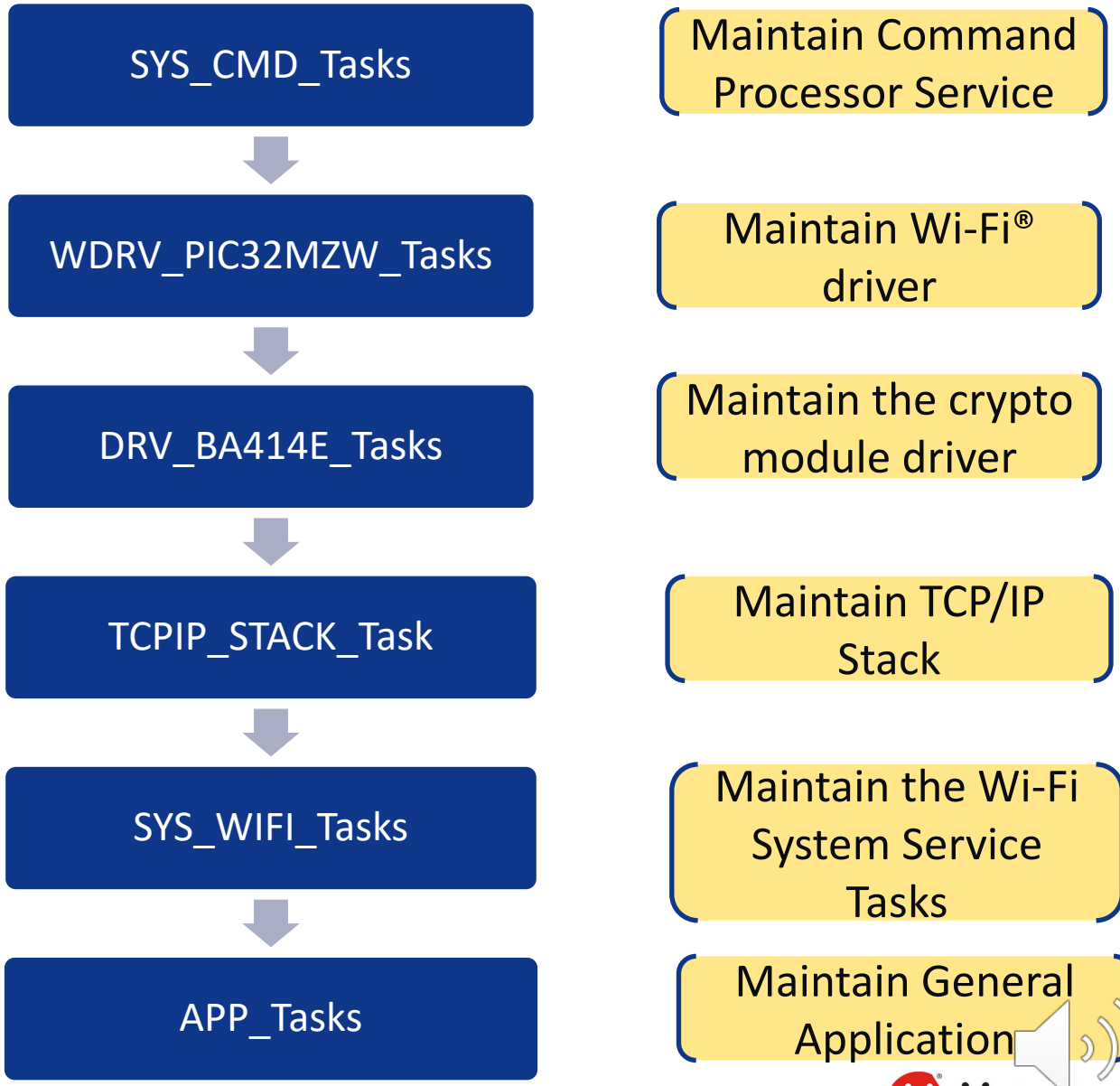




System Tasks

```
int main ( void )
{
   SYS_Initialize ( NULL );
   while ( true )
   {
      SYS_Tasks ( );
   }
   return ( EXIT_FAILURE );
}
```

Run in a while loop to maintain the state machine of each task



- These are the major tasks. Other task may exist depending on configuration
- For FreeRTOS projects, the tasks are created by xTaskCreate function
- Customer **only** need to develop the application in the **APP_Tasks**. All the other task are generated by MHC and run in background

eLearning

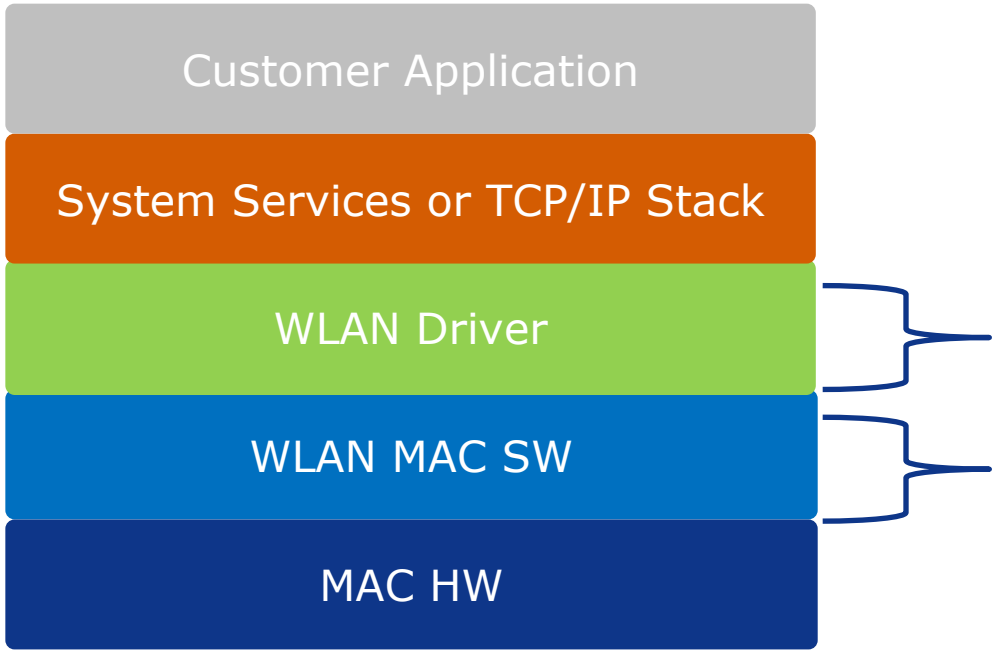
WLAN Driver

Software Introduction

MICROCHIP

Wi-Fi® Application SW

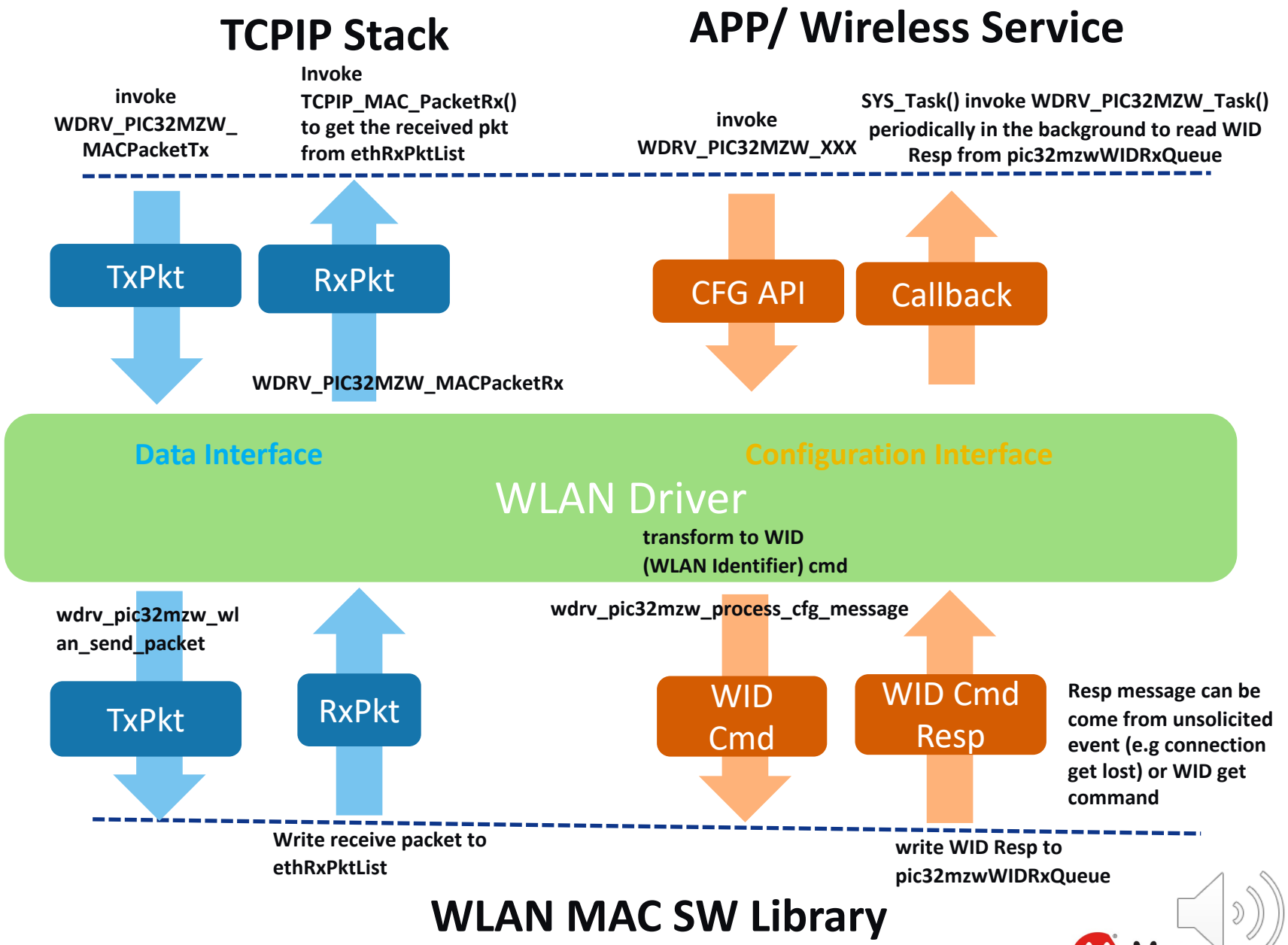
- **WLAN driver is acting as the interface layer between the application, system service, TCP/IP Stack and WLAN MAC SW, HW**



Driver codes are located at driver\wifi\pic32mzw1

pic32mzw1.a
Located at driver\wifi\pic32mzw1\lib

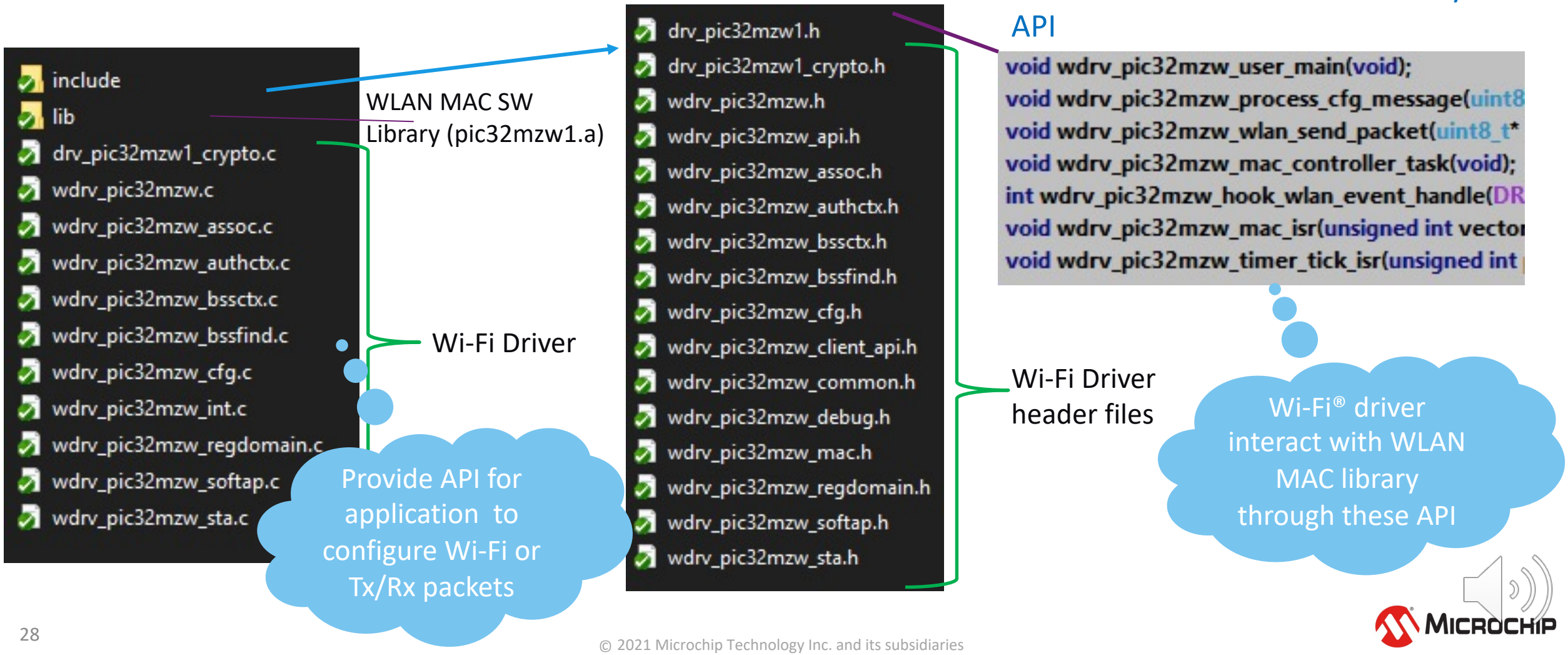
WLAN Driver



WLAN Driver

Folder Structure

Location: driver\wifi\pic32mzw1



WLAN Driver

- **Details of the WLAN Driver API can be found in below document:**
  https://microchip-mplab-harmony.github.io/wireless/
- **Client interface API can be separate into below category:**

API Category	Description
system Interface	Init/ Deinit the PIC32MZW1 Wi-Fi® module instance or get the driver status
Open/ Close	Open or close the driver
STA	Connect/ Disconnect STA
Soft-AP	Start/ Stop AP mode
Authentication Context	Configure the authentication context for Wi-Fi Authentication, used when start AP mode or connect STA
BSS Context	Set the BSS context such as SSID and channel for Wi-Fi connection
BSS Find	Use for scan Wi-Fi channels
Association	Provide information about the current association with a peer device like RSSI, peer mac addr
Information	interface provides general information about the device like device mac addr, operating channel

WLAN Driver

- **Some Highlighted Configuration API:**

Interface API	Description
WDRV_PIC32MZW_BSSCtxSetSSID	The SSID string is copied into the BSS context.
WDRV_PIC32MZW_BSSCtxSetChannel	The supplied channel value is copied into the BSS context.
WDRV_PIC32MZW_AuthCtxSetPersonal	The context are configured appropriately for WPA-PSK authentication.
WDRV_PIC32MZW_AuthCtxSetOpen	The context are configured appropriately for Open authentication.
WDRV_PIC32MZW_BSSConnect	Connect PIC32MZW1 to the BSS as an infrastructure station.
WDRV_PIC32MZW_BSSDisconnect	Disconnects from an existing BSS.
WDRV_PIC32MZW_APStart	Creates and starts a Soft-AP instance.
WDRV_PIC32MZW_APStop	Stops an instance of Soft-AP.
WDRV_PIC32MZW_AssocRSSIGet	Retrieve the RSSI of the current association.
WDRV_PIC32MZW_AssocPeerAddressGet	Retrieve the current association peer device network address.
WDRV_PIC32MZW_BSSFindFirst	A scan is requested. An optional callback can be provided to receive notification of the first BSS discovered.
WDRV_PIC32MZW_BSSFindNext	Requested to get next BSS information structure.

WLAN Driver

API Calling Sequence

Connect STA

Set SSID by calling
WDRV_PIC32MZW_BSSCtxSetSSID()

↓

Set Channel by calling
WDRV_PIC32MZW_BSSCtxSetChannel ()

↓

Set WPA Security mode by calling
WDRV_PIC32MZW_AuthCtxSetPersonal ()

↓

Start connection by calling
WDRV_PIC32MZW_BSSConnect ()

↓

Disconnect/ Connect callback event is received

Scan AP, with callback function

Start scan by calling
WDRV_PIC32MZW_BSSFindFirst ()

↓

Callback function is invoking scanned AP information is available

↓

Call WDRV_PIC32MZW_BSSFindGetInfo () to the information of the scanned AP

Callback function is invoked until all scan results are passed to application

eLearning

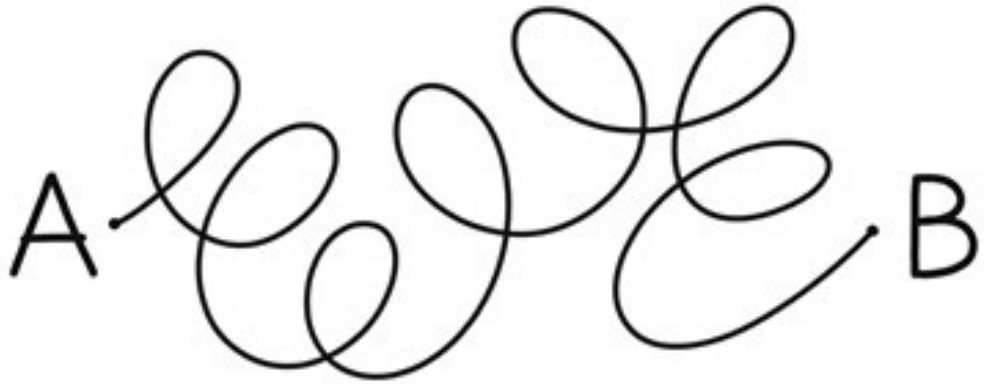
Wireless Service

Software Introduction

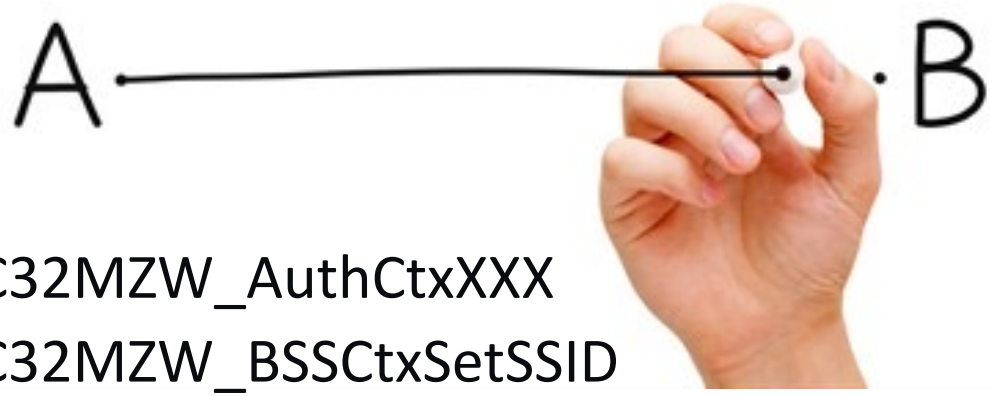
MICROCHIP

Wireless Service

Without Wireless Services



With Wireless Services



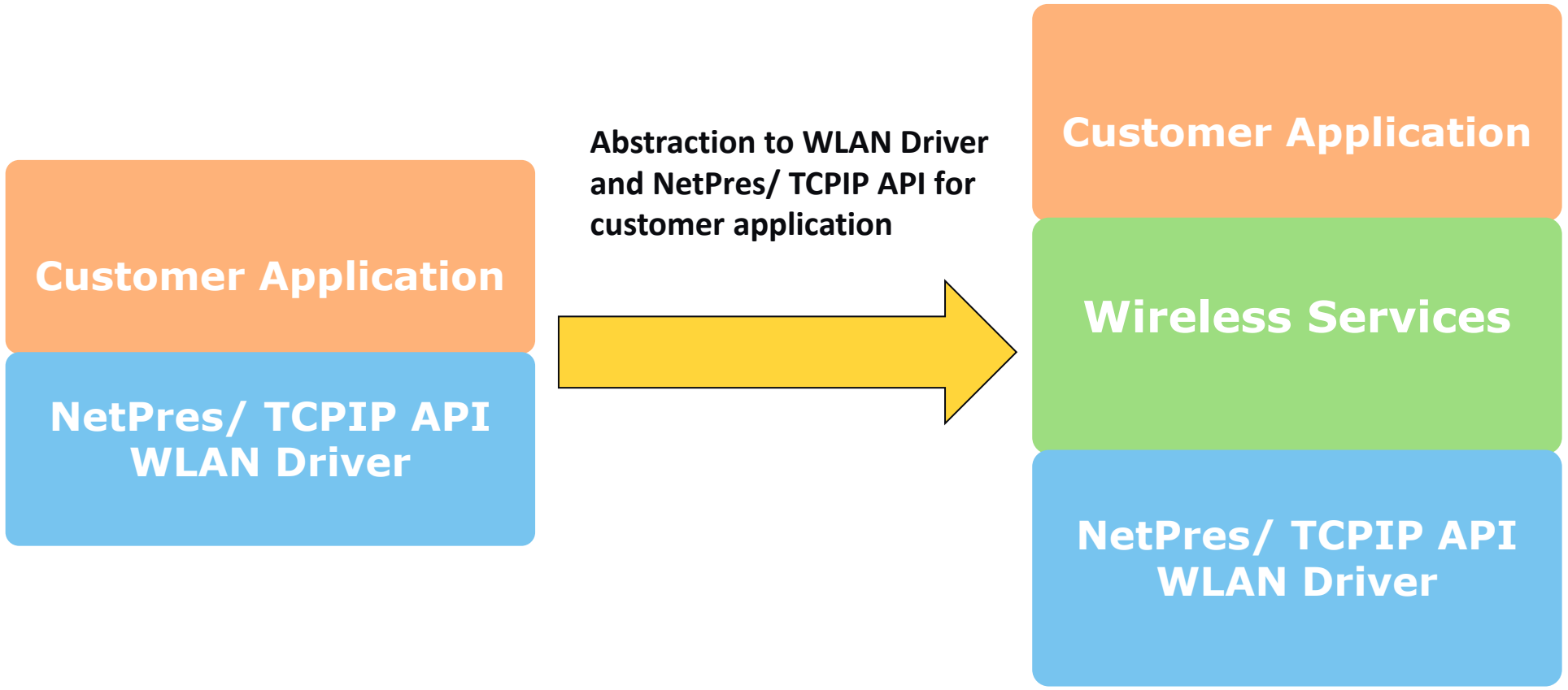
WDRV_PIC32MZW_AuthCtxXXX

WDRV_PIC32MZW_BSSCtxSetSSID

WDRV_PIC32MZW_BSSCtxSetChannel

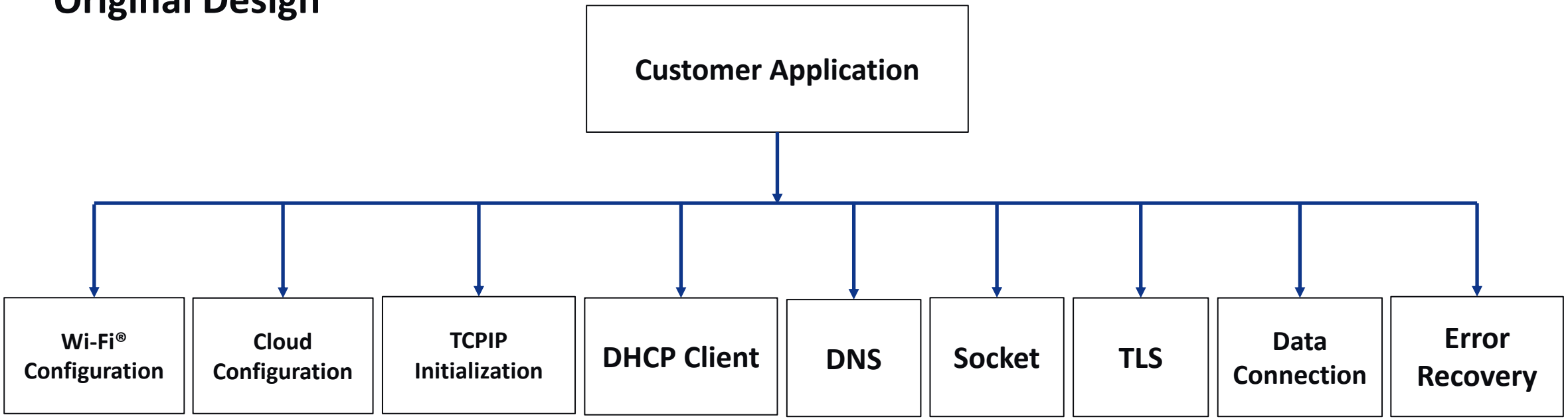
WDRV_PIC32MZW_BSSConnect

Wireless Service

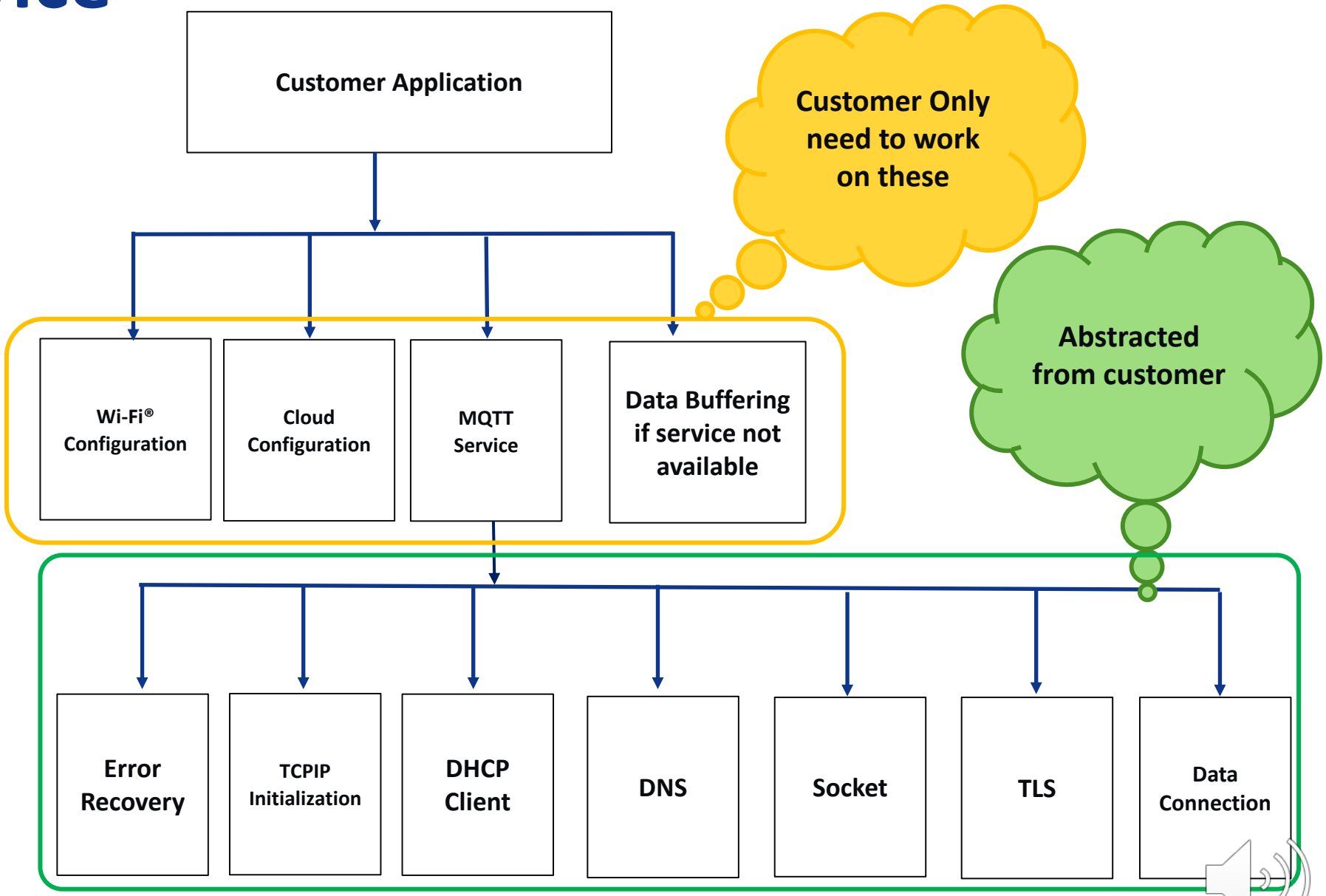


Wireless Service

Original Design



Wireless Service



Delivered in source code, customer can customize the code if they need

Wireless Services

- **Customer with limited Wi-Fi® and networking knowledge are also easy to develop Wi-Fi project**

Customer Already Has Domain Expertise

Limited technology expertise required to get started

~~Networking~~

~~Security~~

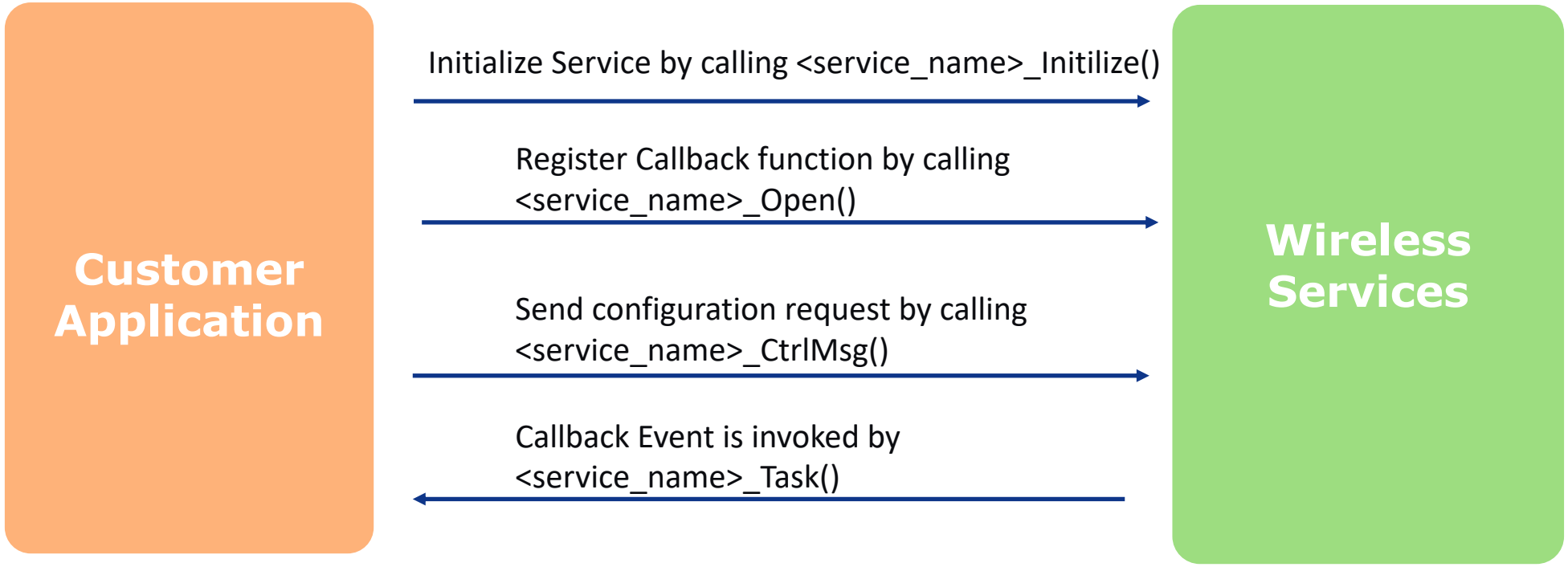
~~None~~

Framework like H3

Wireless Services

- Wi-Fi® System Service – Provide simple interface to manage Wi-Fi functionalities like enable/disable AP/ STA mode, self healing, etc.
- Wi-Fi Provisioning System Service – Provide simple interface to enable Wi-Fi Provisioning with TCP socket or command line
- NET System Service – Provide simple interface for user to request TCP/IP Network connectivity functionalities like TCP/UDP Client/Server with/ without TLS, support self healing
- MQTT System Service – Provide simple interface to have MQTT related features, support self healing
- APP Debug Service – manage and configure debug log printout

Wireless Service Architecture



Notes

NET and MQTT System Service include some other API for data exchange, like below:
SYS_NET_SendMsg() is used to send data to the network
SYS_NET_RecvMsg() is used to receive data to the network
SYS_MQTT_Publish() () is used to publish MQTT message to the network

eLearning

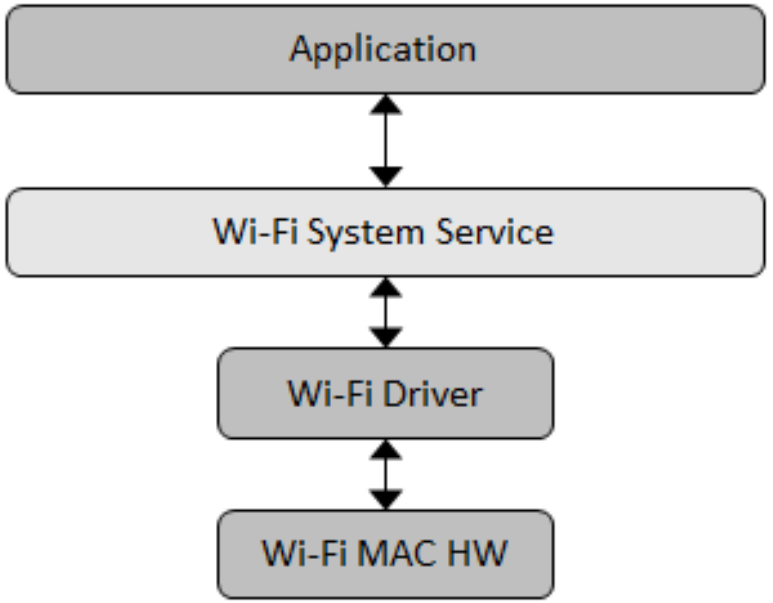
Wireless System Service

Wireless Service

MICROCHIP

Wi-Fi® System Service

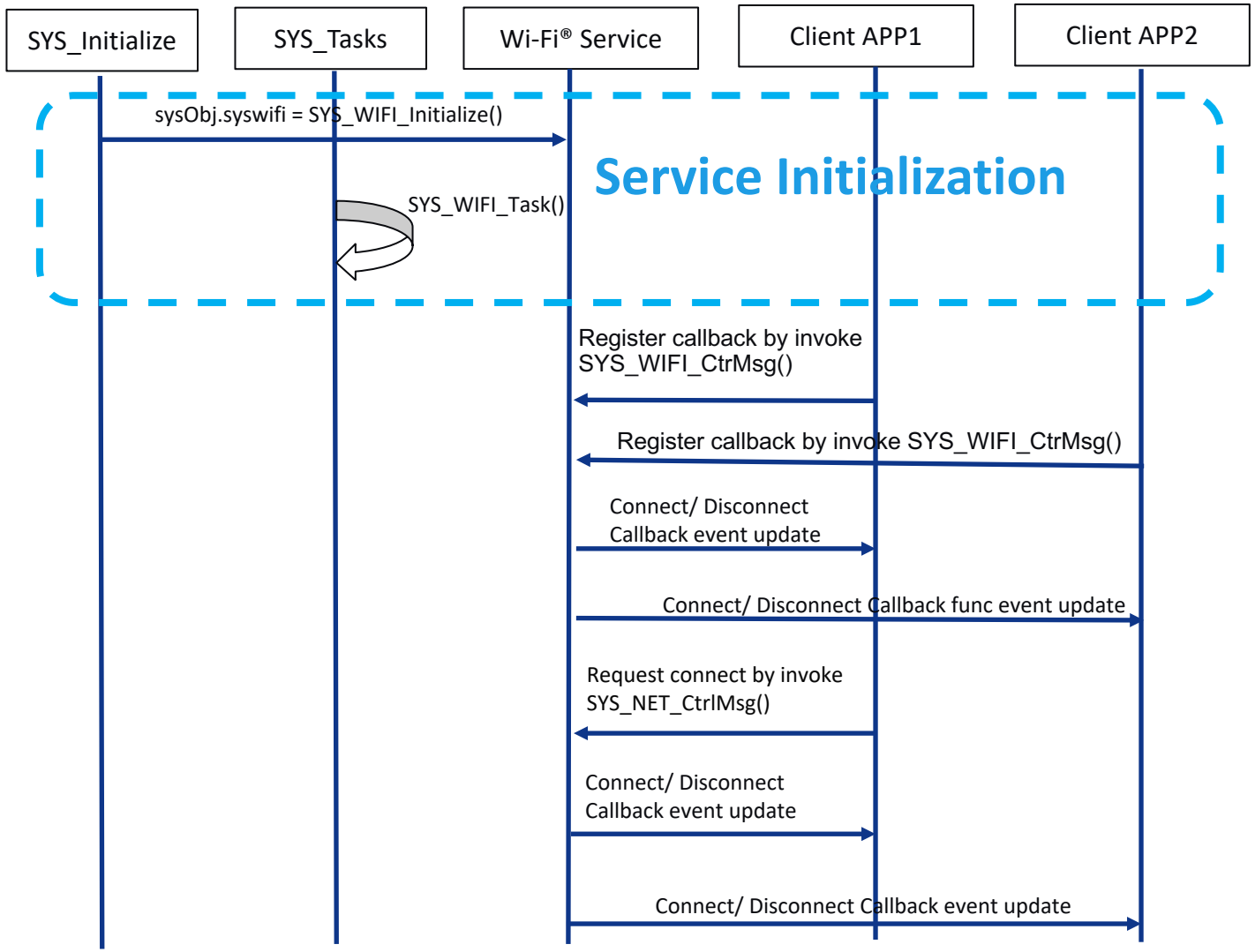
- Simple interface to manage Wi-Fi basic connectivity functionalities
- Features:
  - Configuration of Station mode (STA)
  - Configuration of Soft Access point mode (AP)
  - Scan AP
  - Self Healing: The connection for some reason breaks, the service shall take care of reconnecting the same internally



Wi-Fi® System Service

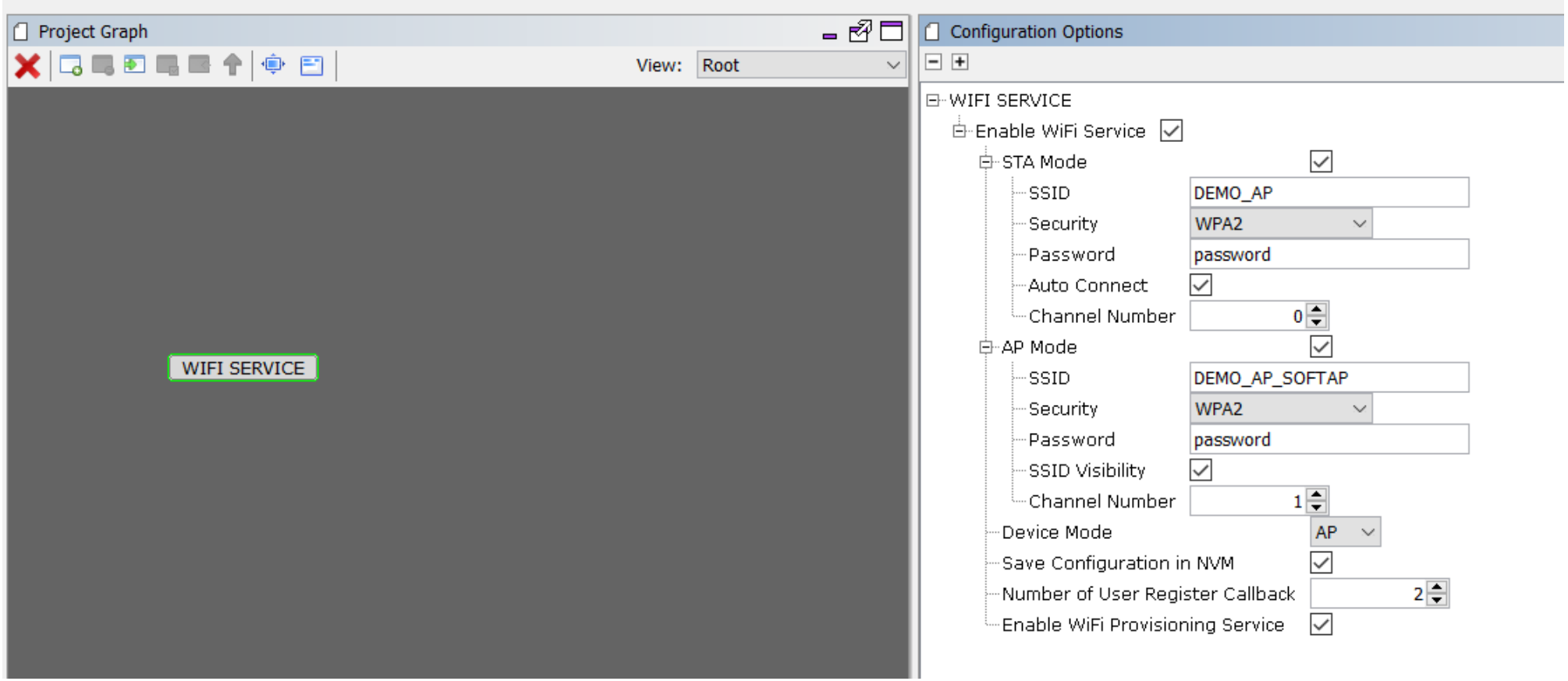
- **How the library works?**

1. SYS_Initialize() invoke SYS_WIFI_Initialize() to **initialize the Wi-Fi System Service module**
2. SYS_Tasks() invoke SYS_WIFI_Tasks() every a period of time to **maintain the Wi-Fi System module functionalities**
3. **Client App 1 register the callback** by calling SYS_WIFI_CtrMsg() with SYS_WIFI_REGCALLBACK event ID
4. **Client App 2 register the callback** by calling SYS_WIFI_CtrMsg() with SYS_WIFI_REGCALLBACK event ID
5. **Client APP1 send connect request message** to Wi-Fi System Service by calling SYS_NET_CtrMsg() with SYS_WIFI_CONNECT event ID
6. **Connect/ Disconnect Event** is sent from Wi-Fi System Service module to Client APP1 and Client APP 2 through the callback function



Wi-Fi® System Service

- **Configure Wi-Fi System Service via MHC**
  - User can select the station (STA) or access point (AP) and configure the detail settings



eLearning

Wireless Provisioning System Service

Wireless Service

Wi-Fi® Provisioning System Service

- **Simple interface to manage Wi-Fi Provisioning**
- **Features:**
  - Wi-Fi Provisioning by below methods
    - **using command line (CLI)**
    - **Wi-Fi Provisioning using TCP socket**
      - A socket server is activated when initialize the service
      - Use a laptop or mobile phone as a TCP client to connect to the device's socket server
      - Send JSON format data to socket server for network provisioning
      - We have a mobile APP (named “Wi-Fi Provisioning” from Android™ Play Store) for Wi-Fi Provisioning by using this method
    - **HTTP Pages**
      - HTTP Socket Number:User configuration for HTTP Server Socket
      - Default port number is 80
    - MHC GUI Based

Wi-Fi® Provisioning System Service

- **CLI based and GUI based are good for development while SoftAP based is good for production**

Provisioning Method	Easy to develop	Compilation required for providing credentials	Mass Production Ready
MHC	Yes	Yes	No
CLI	Yes	No	No
Soft AP	No	No	Yes

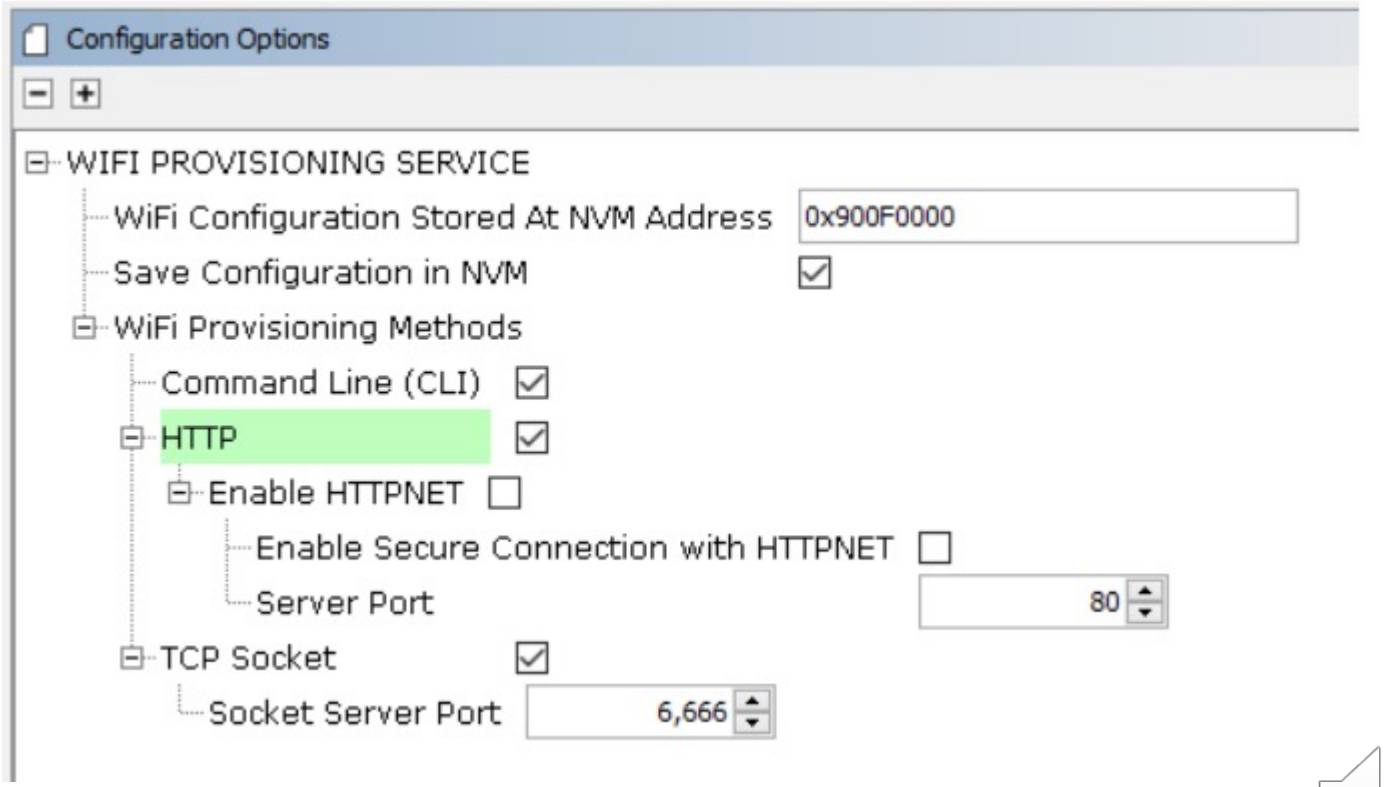
Wi-Fi® Provisioning System Service

- **Command Line:**

Command	Details	Example
wifiprovhelp	Wi-Fi Provision System Service help command	wifiprovhelp
wifiprov set <bootmode> <save config> <channel> <auto_connect> <authtype> <ssid_name> <psk_name>	Set Wi-Fi Configuration for Station(STA) mode	wifiprov set 0 1 1 1 1 "DEMO_AP" "password"
wifiprov set <bootmode> <save config> <channel> <ssid_visibility> <authtype> <ssid_name> <psk_name>	Set Wi-Fi Configuration for Access point(AP) mode	wifiprov set 1 1 1 1 1 "DEMO_SOFTAP" "password"
wifiprov get	Get Wi-Fi Configuration	wifiprov get

Wi-Fi® Provisioning System Service

- **Configure Wi-Fi Provisioning System Service via MHC**
  - User can select provisioning with Command line or TCP Socket method or HTTP method
  - User can input the service server port number



eLearning

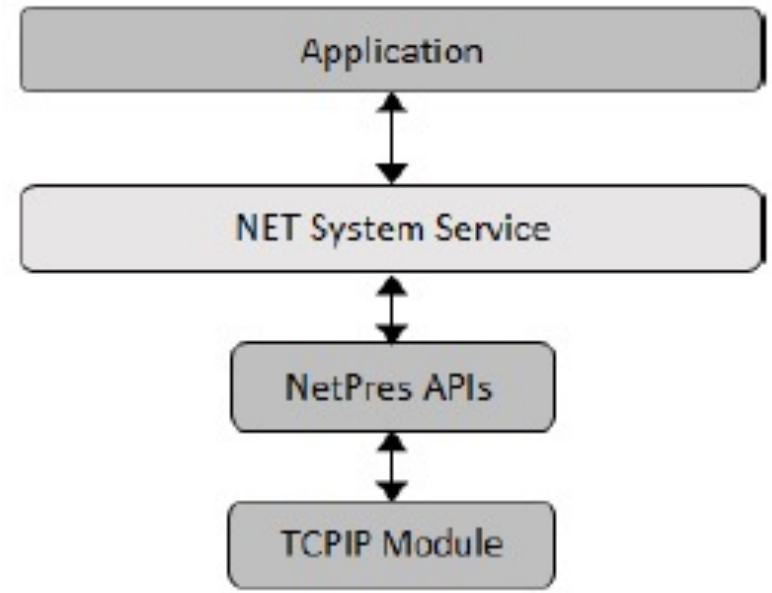
Net System Service

Wireless Service

MICROCHIP

NET System Service

- **Simple interface to manage TCPIP Networking functionalities**
- **Features:**
  - Supports Client/ Server Mode for IP Network Connectivity
  - Supports TCP and UDP Protocols of IP
  - Supports TLS for TCP Connection
  - Supports Self Healing, that is if the connection for some reason breaks, the service shall take care of reconnecting the same internallv
  - Support command line



NET System Service

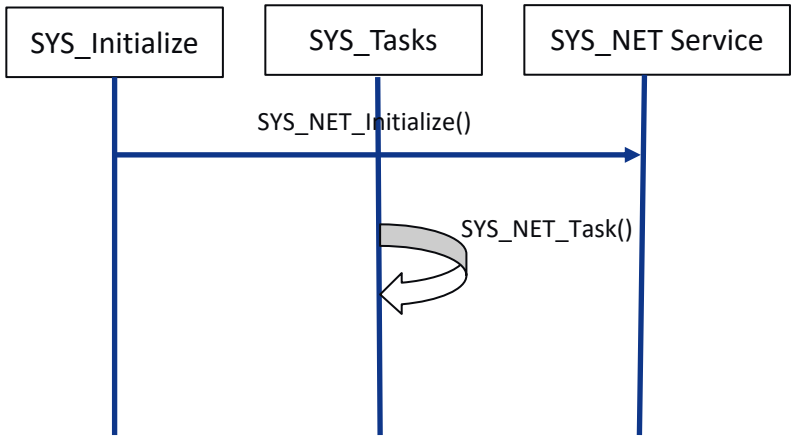
- **Command Line:**

Command	Details	Example
Sysnethelp	NET System Service help command	>sysnethelp
Sysnet get info	Command for knowing the Current Information for all the instances of NET System Service	>sysnet get info Output: ********** NET Service Instance: 0 Status: SYS_NET_STATUS_IDLE Mode: SYS_NET_MODE_CLIENT Socket ID: 0 Host: Remote IP: 0.0.0.0 Remote Port: 0 Local Port: 0 hNet: 0
Sysnet open	Command for Reconfiguring an already open instance of Net System Service	>sysnet 0 1 google.com 443 tls_enable 1 auto_reconnect 1
Sysnet send	Command to send a message on an already open Instance of NET System Service	>sysnet send 0 hello

NET System Service

- **How the library works? (Service Initialization)**

1. SYS_Initialize() invoke *SYS_NET_Initialize()* to initialize the Net Service module
2. SYS_Tasks() invoke SYS_NET_Task() every a period of time in a while loop to maintain the Net Service module functionalities

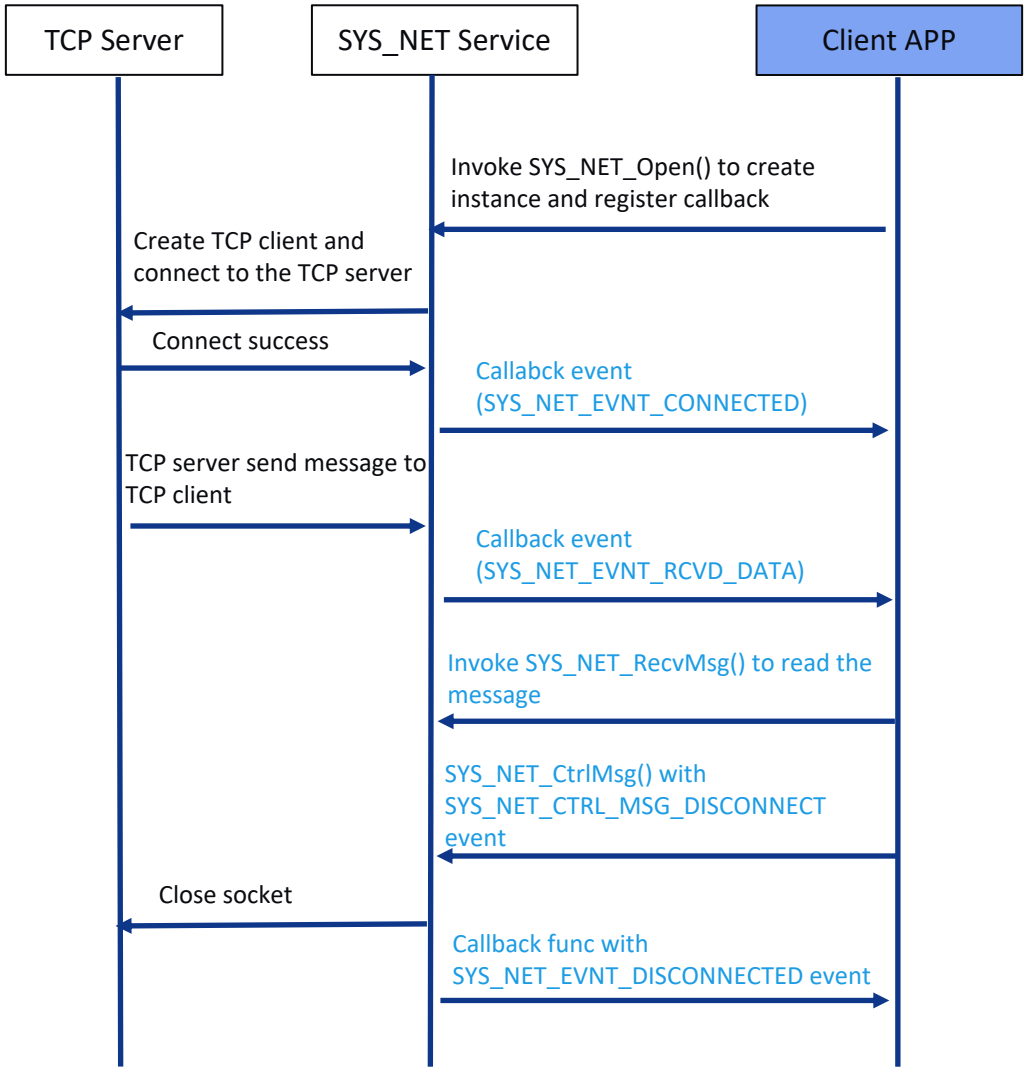


The code of NET System Service Initialization is generated by MHC and run-in background. Customer no need to develop these code.

NET System Service

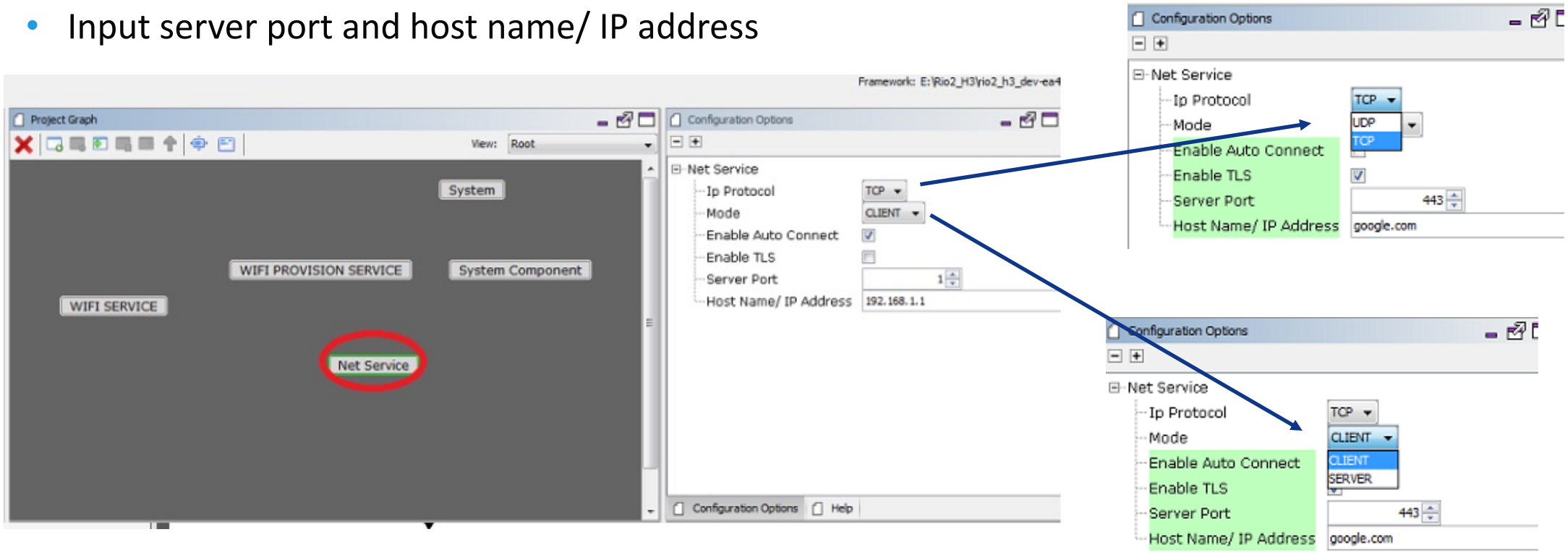
- **How the library works?**
  (TCP Client Example)

1. Client APP invoke SYS_NET_Open () to create the Net service instance and register the callback func
2. Net Service module create the TCP client and connect to the TCP server. If the connection is success, callback event (SYS_NET_EVENT_CONNECTED) is return to client APP
3. When Net service receive message from TCP server, it send callback function with SYS_NET_EVNT_RCVD_DATA event to Client APP
4. Client APP invoke SYS_NET_RecvMsg () to get the message from NET Service module
5. Client APP request Net service module to disconnect TCP socket by invoke SYS_NET_CtrlMsg () with SYS_NET_CTRL_MSG_DISCONNECT event
6. When socket disconnection get success, Net service send callback event SYS_NET_EVNT_DISCONNECTED event to Client APP



NET System Service

- **Configure NET System Service via MHC**
  - User Select TCP or UDP mode
  - Select Client/ Server mode
  - Enable/ disable TLS and self-healing
  - Input server port and host name/ IP address



eLearning

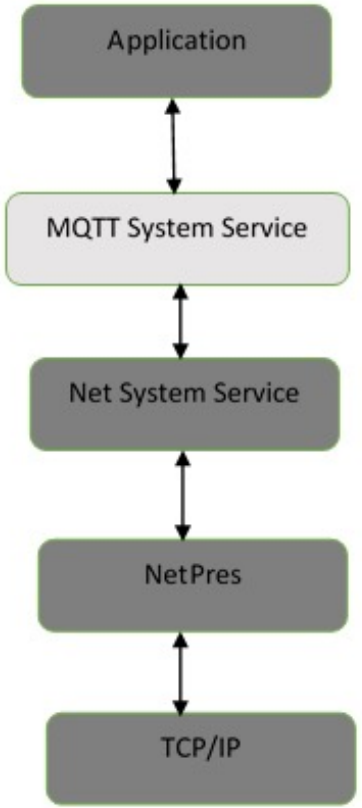
MQTT System Service

Wireless Service

MICROCHIP

MQTT System Service

- Simple interface to manage MQTT functionalities
- The MQTT System Service internally uses the third party Paho MQTT software for MQTT support
- Features:
  - Supports MQTT Client
  - Supports TLS for MQTT Connection
  - Supports Self Healing, that is if the connection for some reason breaks, the service reconnecting the same internally.
  - Supports command line (CLI)

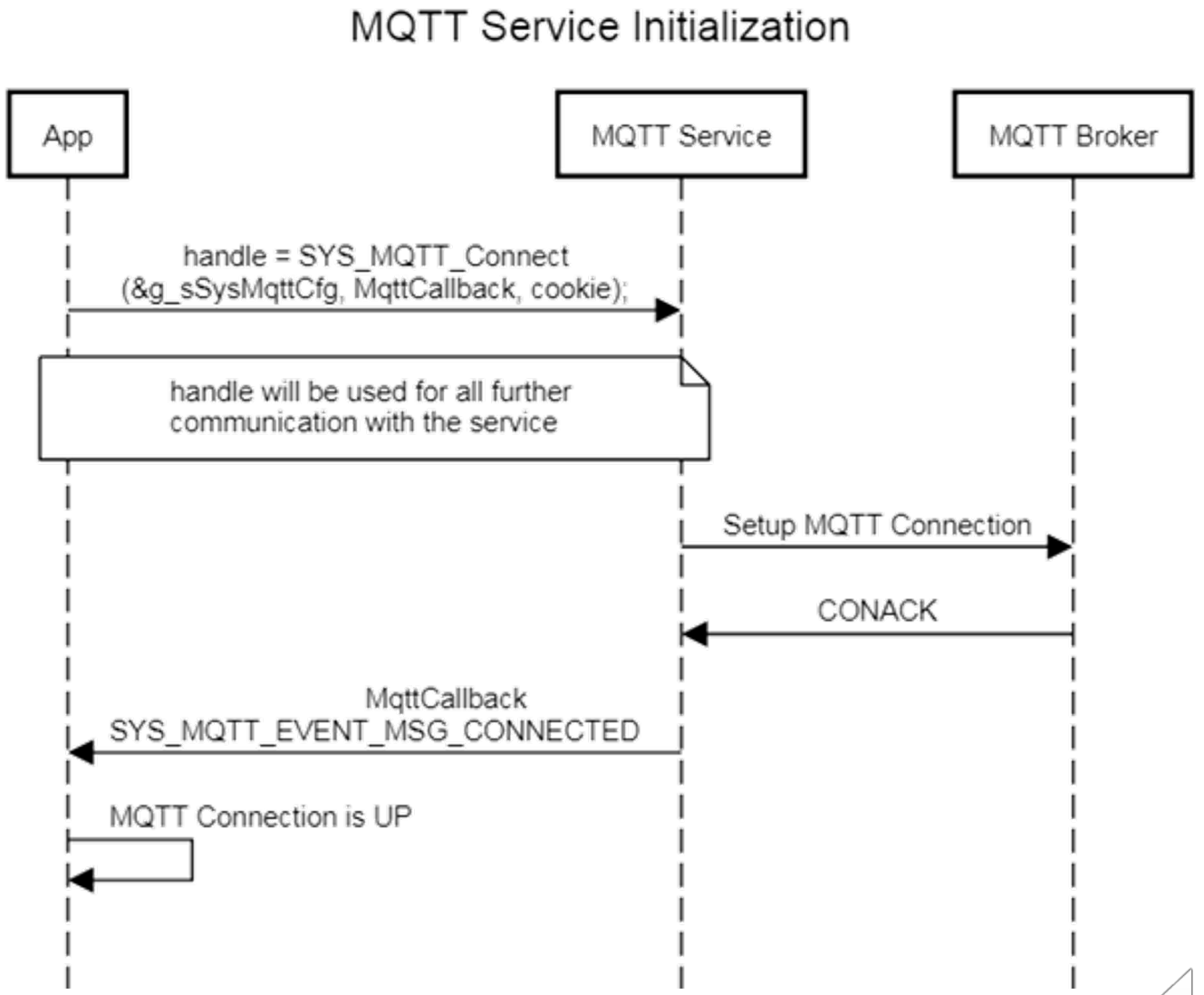


MQTT System Service

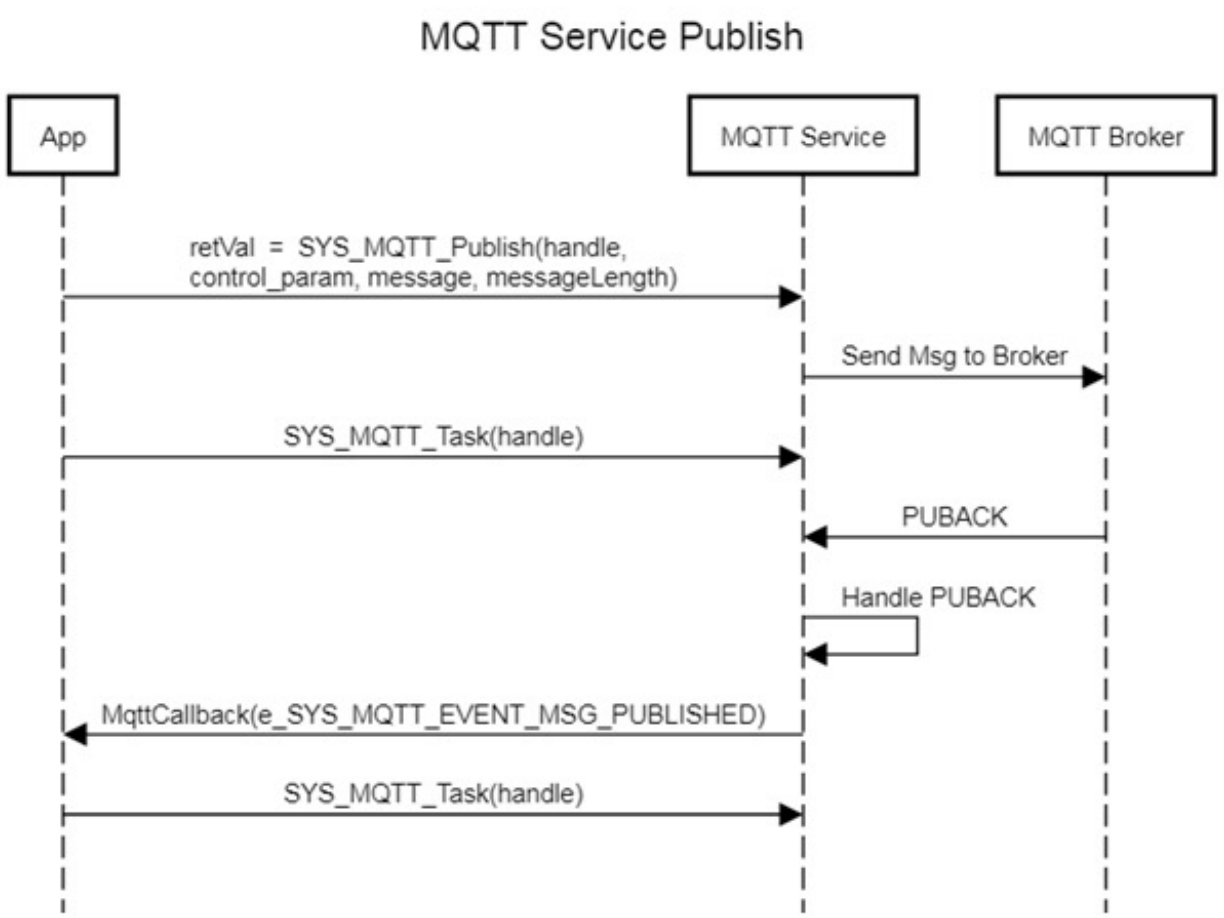
- **Command Line:**

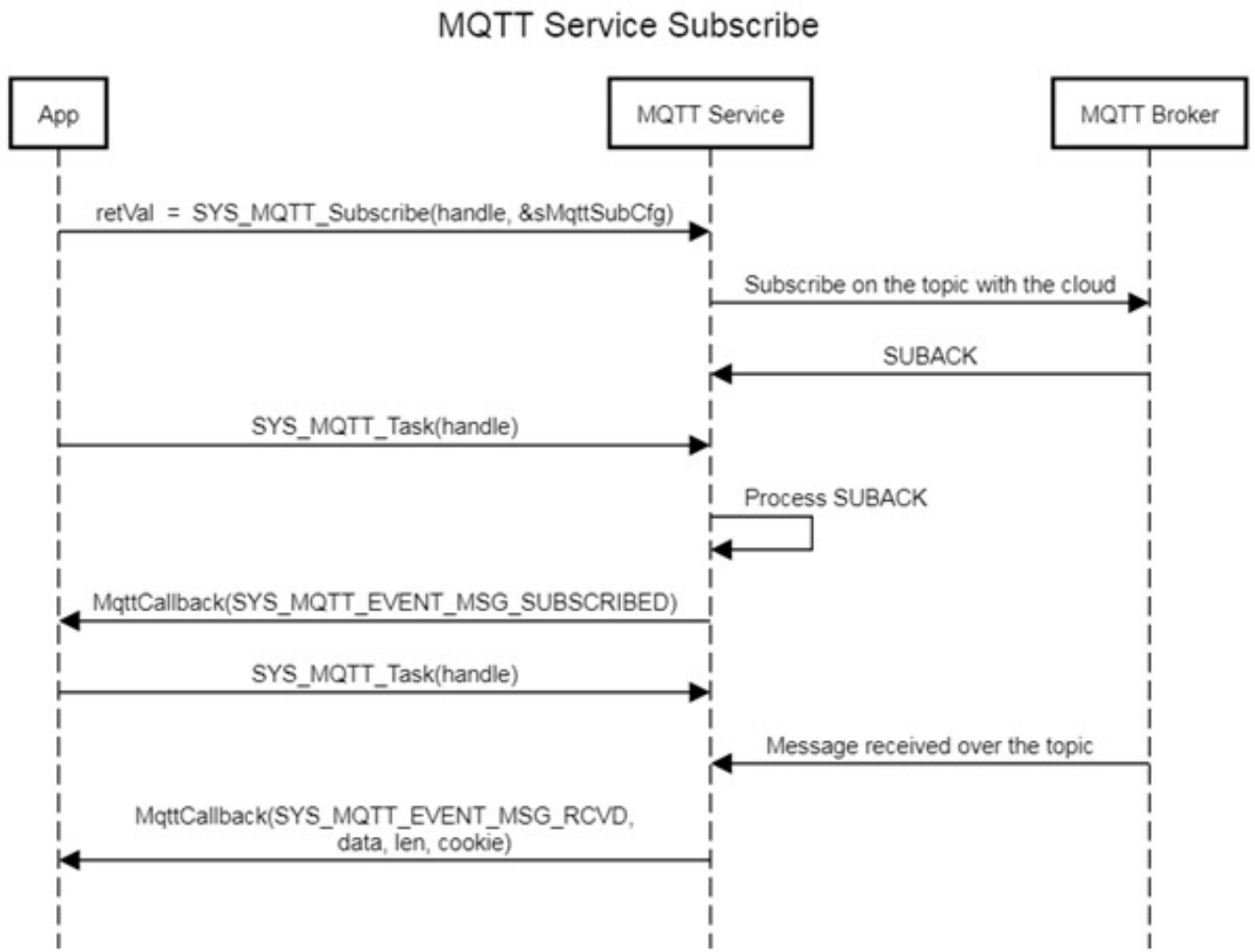
Command	Details	Example
sysmqtthelp	Open the sysmqtt service instance	>sysmqtthelp
sysmqtt get info	Get the list of sysmqtt service instances	>sysmqtt get info Output: ********** MQTT Service Instance: 0 Status: IDLE BrokerName: BrokePort: 0 ClientId: TlsEnabled:0 AutoReconnect: 0 Username: Password:
sysmqtt open	Open sysmqtt service instance	>sysmqtt open 0 mqttbroker <broker_name>
sysmqtt close	Close sysmqtt service instance	>sysmqtt close 0
sysmqtt send	Send message on a topic	>sysmqtt send 0 MCHP/Sample/a 1 1 hello world
sysmqtt subscribe	Subscribe to a topic	>sysmqtt subscribe 0 MCHP/Sample/b 1
sysmqtt unsubscribe	Unsubscribe from a topic	>sysmqtt unsubscribe 0 MCHP/Sample/b

MQTT System Service



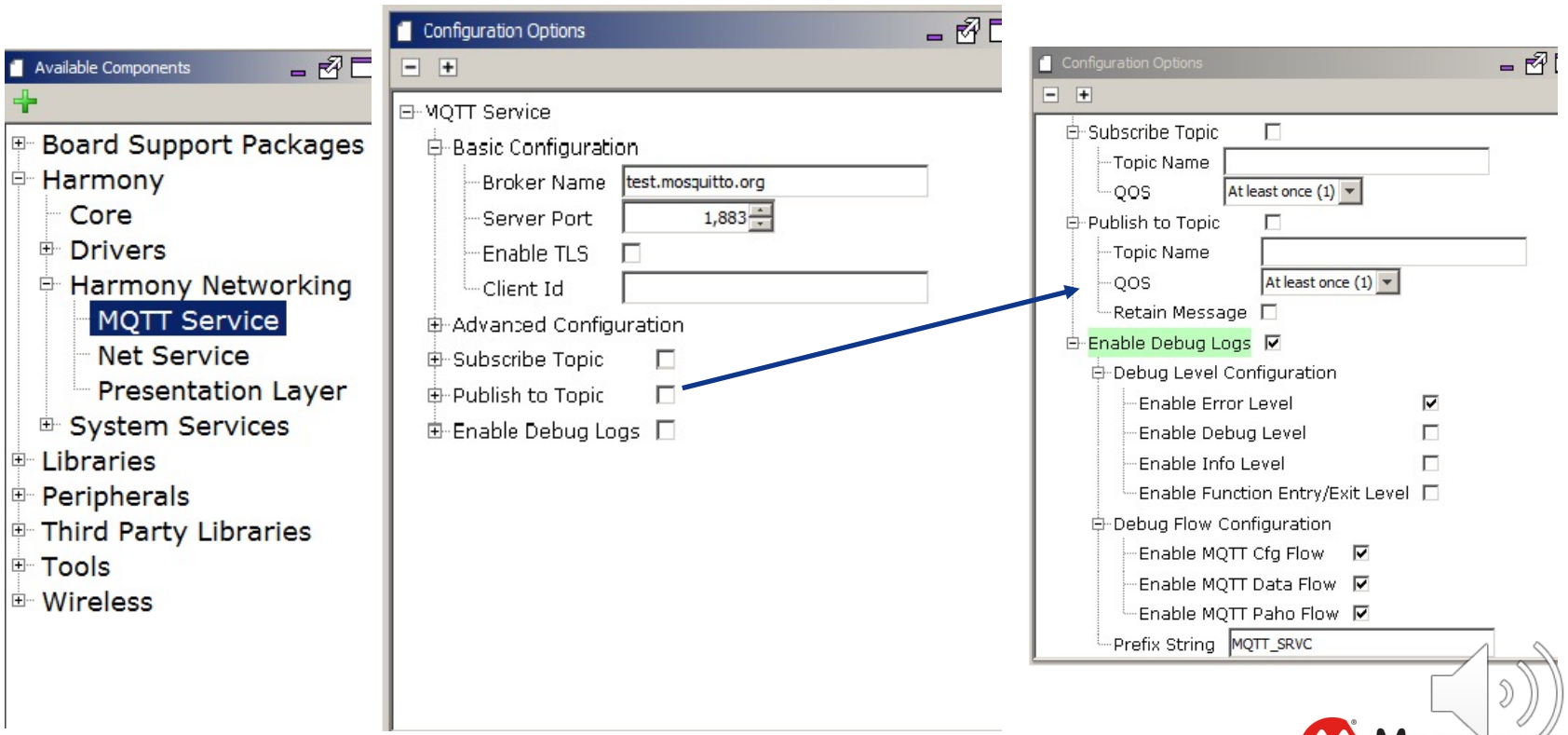
MQTT System Service





MQTT System Service

- Configure MQTT System Service via MHC
  - Broker server configurations
  - MQTT Topic which publish or subscribe
  - Enable/ disable TLS and self-healing



Example Code

- MQTT application code **with** System Services (project: /wireless/apps/paho_mqtt_client)

```
int32_t MqttCallback(SYS_MQTT_EVENT_TYPE eEventType, void *data, uint16_t len, void* cookie)
{
  switch(eEventType)
  {
    case SYS_MQTT_EVENT_MSG_RCVD:
    {
                SYS_MQTT_PublishConfig    *psMsg = (SYS_MQTT_PublishConfig      *)data;
      psMsg->message[psMsg->messageLength] = 0;
      psMsg->topicName[psMsg->topicLength] = 0;
      SYS_CONSOLE_PRINT("\nMqttCallback(): Msg received on Topic: %s ; Msg: %s\r\n",
```

```
void APP_Initialize ( void )
{

    g_sSysMqttHandle = SYS_MQTT_Connect(NULL, MqttCallback, NULL);

}
```

```
void APP_Tasks ( void )
{
  Publish_PeriodicMsg();
  SYS_MQTT_Task(g_sSysMqttHandle);
}
```

```
void Publish_PeriodicMsg(void)
{
  if(checkTimeOut(MQTT_PUB_TIMEOUT_CONST, g_lastPubTimeout))
  {
    char      message[32] = {0};
    SYS_MQTT_PublishTopicCfg    sMqttTopicCfg;
    int32_t retVal = SYS_MQTT_FAILURE;

    //reset the timer
    g_lastPubTimeout = 0;

    /* All Params other than the message are initialized by the config provided in MHC*/
    strcpy(sMqttTopicCfg.topicName, SYS_MQTT_DEF_PUB_TOPIC_NAME);
    sMqttTopicCfg.topicLength = strlen(SYS_MQTT_DEF_PUB_TOPIC_NAME);
    sMqttTopicCfg.retain = SYS_MQTT_DEF_PUB_RETAIN;
    sMqttTopicCfg.qos = SYS_MQTT_DEF_PUB_QOS;

    sprintf(message, "message_%d\r\n", PubMsgCnt);

    retVal = SYS_MQTT_Publish(g_sSysMqttHandle,
                &sMqttTopicCfg,
                message,
                sizeof(message));
    if(retVal != SYS_MQTT_SUCCESS)
    {
      SYS_CONSOLE_PRINT("\nPublish_PeriodicMsg(): Failed
    }
    PubMsgCnt++;
  }
}
```

All around **150** lines in app.c

Easy for customers to get started

Example Code

- MQTT application code **without** System Services (project: /net/apps/wolfmqtt_demo)

```
bool APP_MQTT_Init(void)
{
        // initialize the MQTT context used in the application
        …
}
Void APP_MQTT_Task()
{
switch (mqttCtx->currState)
   {
        case APP_MQTT_STATE_INIT:
              MqttClient_Init()
              …
        case APP_MQTT_STATE_NET_CONNECT:
              MqttClient_NetConnect()
             …
        case APP_MQTT_STATE_CLIENT_CONNECT:
              MqttClient_Connect()
              …
         case APP_MQTT_STATE_SUBSCRIBE:
              MqttClient_Subscribe()
              …
         case APP_MQTT_STATE_PUBLISH:
              MqttClient_Publish()
              …
              …
}
```

A lot of configuration and condition handling in each state
Over **900** lines
in app_mqtt_task.c

Summary

- **Fast and highly efficient MIPS core – WFI32E01**
- **Best peripheral options in the Wi-Fi® MCU market:**
  - **Largest number** of free-to-use GPIOs
  - **Rich peripherals** like Ethernet, CAN/CAN-FD, USB, CVD
- **Premium analog performance**
- **Excellent product quality, supporting tools, and services**
- **MPLAB® Harmony v3**
  - Example applications
  - Peripheral drivers
  - Other system services – **Wi-Fi Service**
  - GitHub support – **Microchip Github**

What's Next….

MICROCHIP UNIVERSITY

Microchip has been delivering superior learning for our clients over the last 23 years at our annual users' conference. Our goal for the new online Microchip University Program is to provide you with all the same information you need to design robust embedded control systems with Microchip solutions.

All Microchip University courses are being offered for free at this time.

Microchip - Chinese [Traditional]

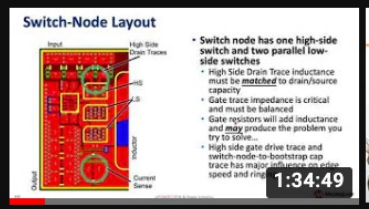
199 位訂閱者

已訂閱

首頁 影片 播放清單 頻道 討論 簡介

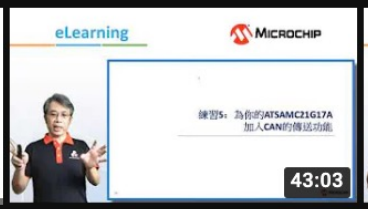
上傳的影片

全部播放



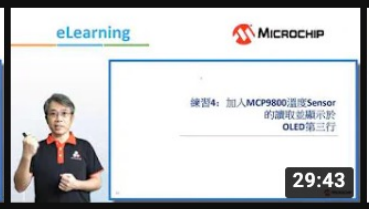
ePOW007——電源分佈網路與電源完整性

觀看次數：29次・3 週前



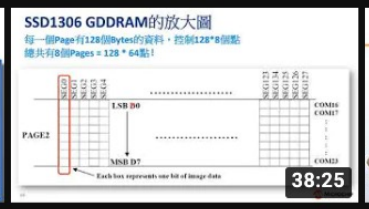
ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：97次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：28次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：31次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：39次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：27次・1 個月前

Featured Channels

Microchip Technology

5.38萬 位訂閱者

Microchip - Chinese [Simplified]

540 位訂閱者

Microchip - Japanese

343 位訂閱者

Microchip - Korean

1310 位訂閱者

Microchip Makes

1.87萬 位訂閱者

eLearning

Wi-Fi® SoC Module *WFI32E01*
介紹及軟體應用篇

Thank you!

Next....

MICROCHIP