

# XENSIV™ KIT CSK PASCO2 for Amazon Web Services IoT Core

## Getting started guide

### About this document

#### Scope and purpose

This getting started guide helps you to set up Amazon Web Services (AWS) IoT Core using [XENSIV™ PASCO2V01 Connected Sensor Kit](#) (KIT\_CSK\_PASCO2).

#### Intended audience

This document is intended for customers interested in using AWS IoT Core with Rapid IoT Developer Kit in combination with the [XENSIV™ PASCO2V01 12 V CO<sub>2</sub> sensor](#) or the [XENSIV™ DPS368 barometric pressure sensor](#) to build their own IoT solution for various consumer applications.

## Table of contents

About this document.....	1
Table of contents.....	2
<b>1 Overview.....</b>	<b>3</b>
<b>2 Hardware description .....</b>	<b>4</b>
2.1 Datasheet.....	4
2.2 Standard kit contents.....	4
2.3 User-provided items.....	5
2.4 3 <sup>rd</sup> party purchasable items .....	5
2.5 Additional hardware references .....	5
<b>3 Set up your development environment.....</b>	<b>6</b>
3.1 Tools installation (IDEs, Toolchains, SDKs).....	6
3.1.1 Infineon Rapid IoT Connect Platform.....	6
3.1.2 Code examples in ModusToolbox™ .....	6
3.2 Prerequisites.....	6
<b>4 Set up your hardware .....</b>	<b>7</b>
4.1 Wing board components.....	7
4.2 Adafruit Feather-compatible connectors.....	8
4.3 Power supply .....	9
4.4 Mechanical buttons.....	10
<b>5 Setup your AWS account and permissions .....</b>	<b>11</b>
<b>6 Create resources in AWS IoT .....</b>	<b>12</b>
<b>7 Provision the device with credentials .....</b>	<b>13</b>
<b>8 Build and run the demo.....</b>	<b>14</b>
8.1 Create and build an application .....	14
8.2 Flash program image without creating an application .....	16
8.3 Configure the parameters.....	16
8.3.1 Set Wi-Fi credentials .....	16
8.3.2 Configure AWS IoT Core for MQTT .....	17
8.3.3 Navigate to <i>configs/mqtt_client_config.h</i> in Project Explorer .....	17
8.3.4 Program board using Eclipse IDE for ModusToolbox™ .....	17
8.4 Run the demo .....	18
8.5 Verify messages in AWS IoT Core .....	18
<b>9 Troubleshooting .....</b>	<b>20</b>
<b>References.....</b>	<b>23</b>
<b>Glossary .....</b>	<b>24</b>
<b>Revision history.....</b>	<b>25</b>
<b>Disclaimer.....</b>	<b>26</b>

### Overview

## 1 Overview

The XENSIV™ Connected Sensor Kit is an IoT development kit that helps you design, develop, and manufacture your own product based on a hardware (HW) + software (SW) development ecosystem.

One of the key components on the Wing board is Infineon's XENSIV™ PASCO2V01 sensor. The sensor kit is Adafruit Feather-compatible with a small form factor and capable of operating using battery.

The XENSIV™ Connected Sensor Kit ecosystem is targeted at application segments such as building automation, smart home, and consumer electronics. The development kit is integrated with a SW ecosystem provided through ModusToolbox™ and supports the following functionalities:

- **Sense:** [XENSIV™ PASCO2V01 sensor](#) and [XENSIV™ DPS368 barometric pressure sensor](#)
- **Compute:** [PSOC™ 62 microcontroller](#)
- **Connect:** [AIROC™ CYW43012 dual-band 2.4 GHz and 5 GHz Wi-Fi 4 \(802.11n\) and Bluetooth® 5.0 combo module](#)

## 2 Hardware description

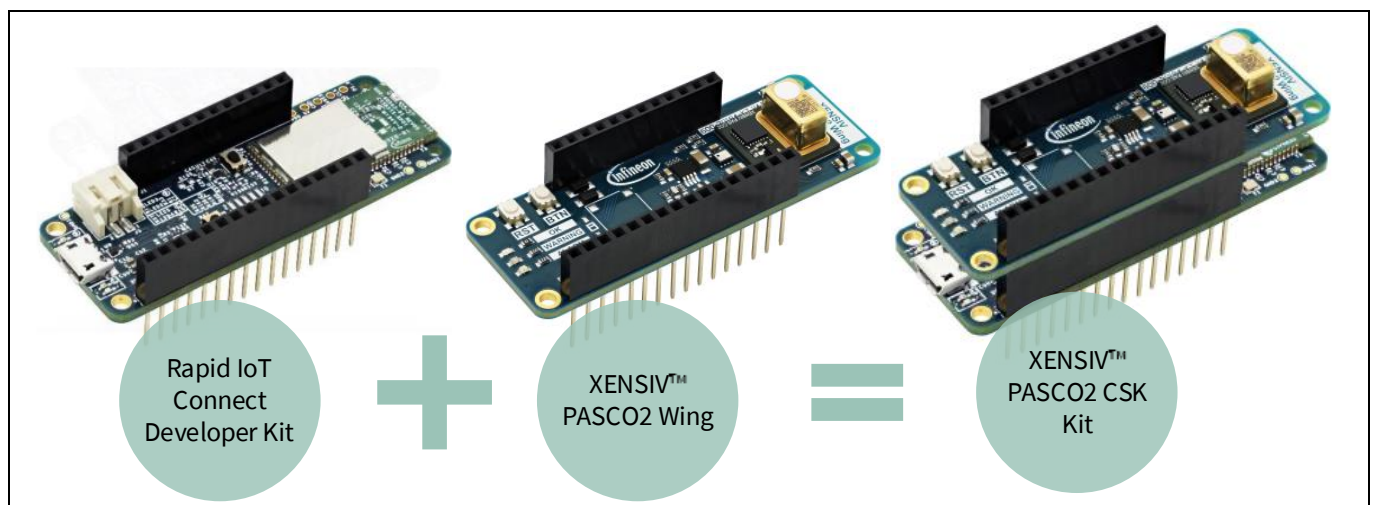
### 2.1 Datasheet

For more details on the [KIT\\_CSX\\_PASCO2](#), refer to the [user guide](#).

### 2.2 Standard kit contents

The XENSIV™ PASCO2 Connected Sensor Kit ([Figure 1](#)) comes with:

- Rapid IoT Connect Developer Kit (CYSBSYSKIT-DEV-01)
- XENSIV™ PASCO2V01 Wing (EVAL\_PASCO2\_WING)



**Figure 1** XENSIV™ PASCO2 Connected Sensor Kit

The Rapid IoT Connect Developer Kit (CYSBSYSKIT-DEV-01) shown in [Figure 2](#) allows the evaluation of the Rapid IoT Connect module (CYSBSYS-RP01) on a standard Feather form factor. The CYSBSYS-RP01 Rapid IoT Connect module is a turnkey module that enables secure, scalable, and reliable compute and connect.

The Rapid IoT Connect Developer Kit carries a CYSBSYS-RP01 Rapid IoT connect system-on-module (SoM), which includes a PSoC™ 6 MCU, a CYW43012 single-chip radio, onboard crystals, oscillators, chip antenna, and passive components.

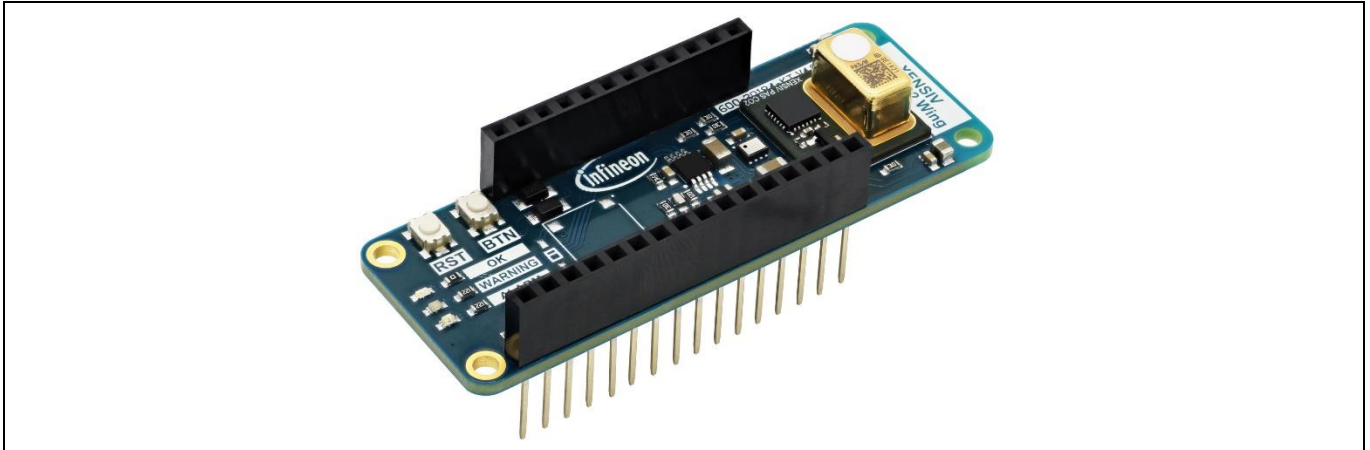


**Figure 2** CYSBSYSKIT-DEV-01

## Getting started guide

### Hardware description

The XENSIV™ PASCO2V01 Wing board shown in [Figure 3](#) is based on the PASCO2V01 CO<sub>2</sub> sensor. The sensor is based on the photoacoustic spectroscopy (PAS) principle, where CO<sub>2</sub> molecules within the sensor cavity absorb the infrared light, generating small pressure changes that are detected by an acoustic detector. The integrated microcontroller provides a direct readout of CO<sub>2</sub> concentration in parts per million (ppm). Precise CO<sub>2</sub> readings are guaranteed.



**Figure 3** XENSIV™ PASCO2V01 Wing board

The board also comprises the XENSIV™ DPS368 digital barometric pressure sensor. This high-precision pressure sensor can detect very small changes in barometric pressure, which makes it an ideal device where accurate pressure event detection is required (for example, opening of doors or windows or fall detection).

Having both the CO<sub>2</sub> sensor and pressure sensor on the board gives the possibility to develop and test more complex scenarios where data from both sensors is combined for more reliable event detection.

### 2.3 User-provided items

- Rapid IoT Connect Developer (CYSBSYSKIT-DEV-01) V3.0
- XENSIV™ PASCO2V01 Wing (EVAL\_PASCO2\_WING) V4.4

### 2.4 3<sup>rd</sup> party purchasable items

- Micro-USB cable

### 2.5 Additional hardware references

- [XENSIV™ PASCO2V01 sensor](#)
- [XENSIV™ DPS368 barometric pressure sensor](#)
- [PSOC™ 62 MCU](#)
- [AIROC™ CYW43012](#)

## 3 Set up your development environment

### 3.1 Tools installation (IDEs, Toolchains, SDKs)

#### 3.1.1 Infineon Rapid IoT Connect Platform

Infineon Rapid IoT Connect Platform provides hardware, firmware, and cloud artifacts to enable rapid onboarding of Infineon's customers to the IoT world. This platform enables bidirectional communication between Infineon devices, sensors, and the cloud, and provides flexibility to provision, monitor, and manage Infineon devices remotely. It also provides device management capabilities that include managing the device state, live fleet monitoring, and firmware over-the-air (FOTA) updates.

Refer to the [user guide](#) for a detailed description and step-by-step instructions.

#### 3.1.2 Code examples in ModusToolbox™

The XENSIV™ PASCO2V01 Wing is supported by the *sensor-xensiv-pasco2*, *sensor-xensiv-dps3xx*, and *mqtt-client* libraries. Code examples using these libraries are available in ModusToolbox™.

- [XENSIV™ PASCO2 Sensor Library](#)

This library provides functions for interfacing with the XENSIV™ PAS CO2 sensor that enables you to read the CO<sub>2</sub> concentration. This library can be set up to use the ModusToolbox™ HAL interface or using user-provided communication functions. See the *README.md* file for more details.

- [XENSIV™ DPS3xx Pressure Sensor Library](#)

This library provides functions for interfacing with the XENSIV™ DPS-310/368 barometric pressure sensors. This library can be set up to use the ModusToolbox™ HAL interface or using user-provided communication functions. See the *README.md* file for more details.

- Code examples

The code examples demonstrate the MQTT client use case implemented using the *mqtt-client* library. See the *README.md* file for more details.

- [mtb-example-sensors-pasco2](#)
- [mtb-example-sensors-pasco2-anycloud-mqtt-client](#)

### 3.2 Prerequisites

It's recommended to use:

- [ModusToolbox™](#) v3.0 or later
- Serial terminal. For example, [Tera Term](#) (for interacting with the code example programs)

## 4 Set up your hardware

This section introduces you to the various features of the XENSIV™ PASCO2V01 Wing board. Apart from the headers, all components are mounted on the top-side of the Wing board, which has male headers facing downwards to either plug the board directly on the Rapid IoT baseboard or on top of another Wing board such as the Infineon XENSIV™ PAS CO2 Wing board.

### 4.1 Wing board components

Figure 4 and Table 1 provide a description of the components mounted on the XENSIV™ PASCO2V01 Wing board.

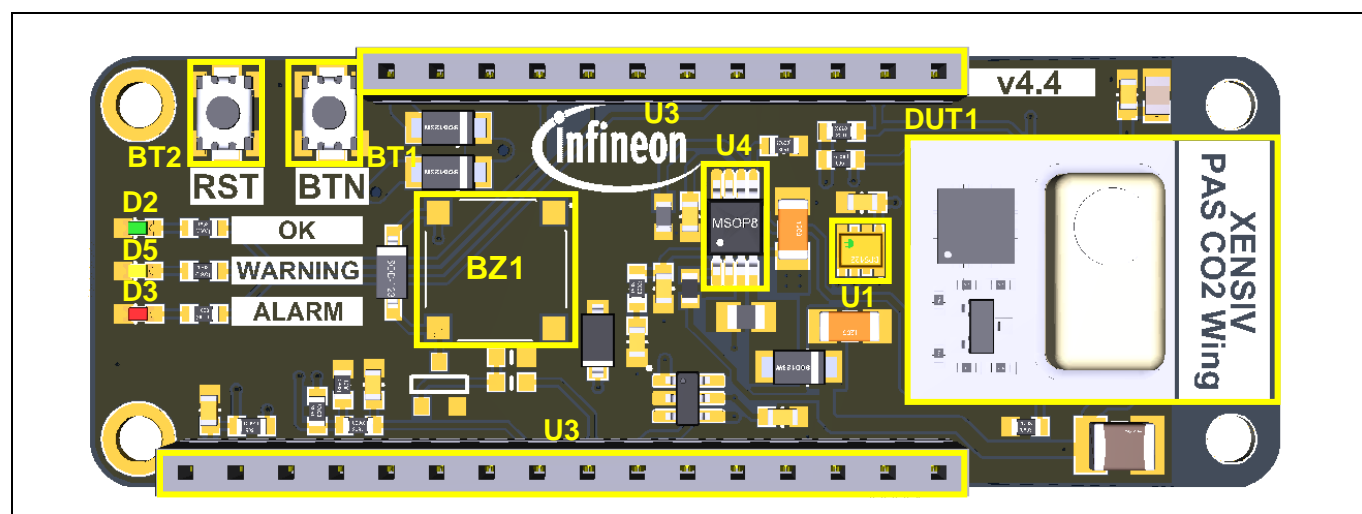


Figure 4 Front view of the XENSIV™ PASCO2V01 Wing board

Table 1 Onboard hardware

Designator	Function
U1	XENSIV™ DPS368 digital barometric pressure sensor
U2	2:1 (SPDT), 1-channel analog switch, programmable control of the on and off output 3 V
U3	28-pin Adafruit Feather-compatible adapter headers
U4	Boost/inverting DC-DC converter with 2 A switch, soft-start, and synchronization
DUT1	XENSIV™ PASCO2V01 device
BT1	User button; active LOW
BT2	System reset button; active LOW
D2	Green LED for acceptable level of CO <sub>2</sub> concentration
D3	Red LED for alarm level of CO <sub>2</sub> concentration
D5	Yellow LED for warning level of CO <sub>2</sub> concentration
BZ1	Option for buzzer (not populated by default)

## 4.2 Adafruit Feather-compatible connectors

Figure 5 highlights the 28-pin Adafruit Feather-compatible headers. The function of the respective header pins is described in Table 2. The image also shows the test points which were used for testing the boards in the lab or production.

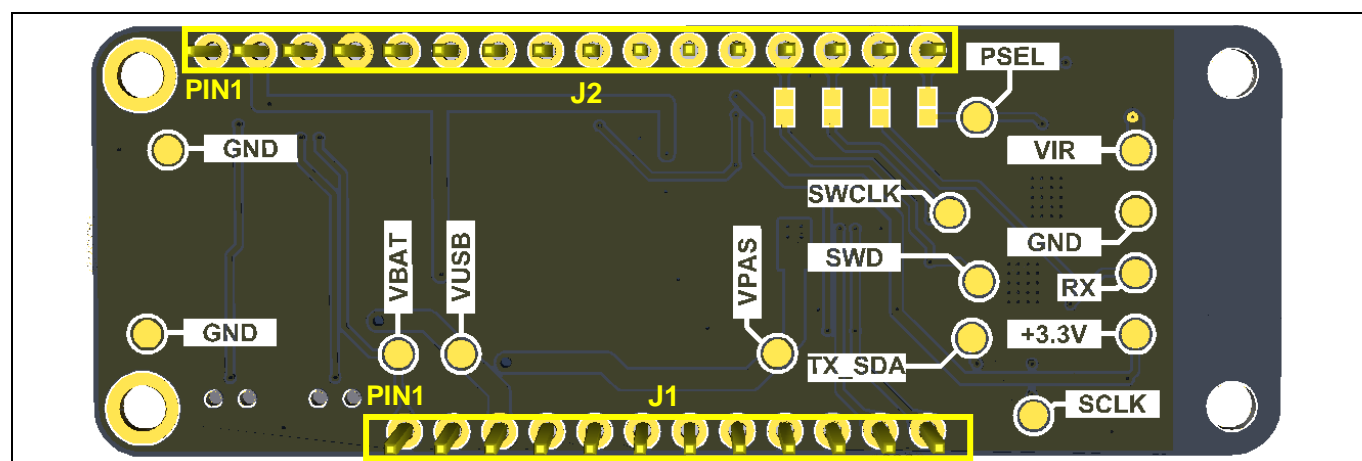


Figure 5 Adafruit headers and test points on bottom of the XENSIV™ PASCO2V01 Wing board

Table 2 Adafruit Feather-compatible pinout

Header mapping	Primary onboard function	PSOC™ 6 MCU pin (Rapid IoT baseboard)	Adafruit Feather-compatible mapping (Rapid IoT baseboard)	Adafruit Feather-compatible mapping (PASCO2V01 Wing Board)	Details
J1.1	VBAT	–	–	VBAT	LiPo battery voltage
J1.2	EN	–	–	–	Not connected
J1.3	VBUS	–	–	VUSB	USB power
J1.4	GPIO	P9_0	GPIO13	–	Not connected
J1.5	GPIO	P9_1	GPIO12	–	Not connected
J1.6	GPIO	P9_2	GPIO11	–	Not connected
J1.7	GPIO	P9_3	GPIO10	–	Not connected
J1.8	GPIO	P9_4	GPIO9	RED_LED	Red LED
J1.9	GPIO	P9_7	GPIO6	–	Not connected
J1.10	GPIO	P8_4	GPIO5	–	Not connected
J1.11	I <sup>2</sup> C SCL	P6_0	SCL	I2C_SCL_Feather	Connected to KitProg3. Note that this pin has a 4.7 kΩ pull-up for I <sup>2</sup> C communication
J1.12	I <sup>2</sup> C SDA	P6_1	SDA	I2C_SDA_Feather	Connected to KitProg3. Note that this pin has a 4.7 kΩ pull-up for I <sup>2</sup> C communication



Header mapping	Primary onboard function	PSOC™ 6 MCU pin (Rapid IoT baseboard)	Adafruit Feather-compatible mapping (Rapid IoT baseboard)	Adafruit Feather-compatible mapping (PASCO2V01 Wing Board)	Details
J2.1	XRES	XRES	XRES	RST	Reset button
J2.2	3.3 V	VDDA, VDDIO	VCC	+3V3	Analog voltage for PSOC™ 6 MCU
J2.3	NC	–	NC	–	Not connected
J2.4	GND	–	GND	GND	Ground
J2.5	Analog GPIO	P10_0	A0	–	Not connected
J2.6	Analog GPIO	P10_1	A1	–	Not connected
J2.7	Analog GPIO	P10_2	A2	–	Not connected
J2.8	Analog GPIO	P10_3	A3	BTN	User button/VBAT voltage monitoring
J2.9	Analog GPIO	P10_4	A4	BUZZER	PASCO2 buzzer
J2.10	Analog GPIO	P10_5	A5	POWERDOWN	PASCO2 board power-down
J2.11	SPI Clock	P5_2	SCK	–	Not connected
J2.12	SPI MOSI	P5_0	MOSI	–	Not connected
J2.13	SPI MISO	P5_1	MISO	SWD	SWD
J2.14	UART RX	P6_4	RX	SWCLK	SWCLK
J2.15	UART TX	P6_5	TX	RX	RX
J2.16	SPI CS	P5_3	GPIO	PSEL	PSEL at PASCO2 sensor

## 4.3 Power supply

The kit can be powered from a 3.7 V LiPo battery or via a USB cable from an external 5 V power supply. The battery is automatically charged when the system is connected to an external power supply.

*Note: The PASCO2V01 Wing board must be manually switched to either battery or external 5 V supply (switch S3 in [Figure 4](#)).*

## 4.4 Mechanical buttons

**Table 3** Onboard hardware

Button	Function
RST	System reset
BTN	User button — the function executed on button press can be individually programmed

## 5 Set up your AWS account and permissions

If you do not have an existing AWS account, see the AWS documentation at [Set up your AWS Account](#). To get started, follow the steps outlined in the sections below:

- [Sign up for an AWS account](#)
- [Create a user with administrative access](#)
- [Open the AWS IoT console](#)

## 6 Create resources in AWS IoT

See the AWS documentation at [Create AWS IoT resources](#). Follow the steps outlined in these sections to provision resources for your device:

- [Create an AWS IoT policy](#)
- [Create a Thing object](#)

## 7 Provision the device with credentials

This section requires you to use the AWS signed certificates.

1. Sign in to the [AWS IoT console](#).
2. In the left navigation pane, select **Secure > Certificates > Create**.
3. Select **One-click certificate creation** (recommended) to create the certificate.
4. From the **Certificate created!** page, download the client certificate files for the Thing, public key, and private key to a secure location. These certificates generated by AWS IoT are only available for use with AWS IoT services.

A client certificate has now been created and registered with AWS IoT. You must activate the certificate before you use it in a client.

*Note: If you also need the Amazon Root CA certificate file, this page also has the link to the page from where you can download it.*

5. Click **Activate**.

If you don't want to activate the certificate now, [Activate a client certificate \(console\)](#) describes how to activate the certificate later.

6. If you want to attach a policy to the certificate, click **Attach a policy**.
  - a) Sign up for the Amazon Partner Central using a valid ID.
  - b) Go to the device listing on the portal.
  - c) Add a new device.
  - d) Select the service to which the AWS services are validated.
  - e) List the product device qualification test to be done. Upload the certificate for the same along with the relevant SKU number.

## 8 Build and run the demo

The following example illustrates how to build a demo application in ModusToolbox™, a development environment for Infineon microcontrollers for embedded and IoT applications.

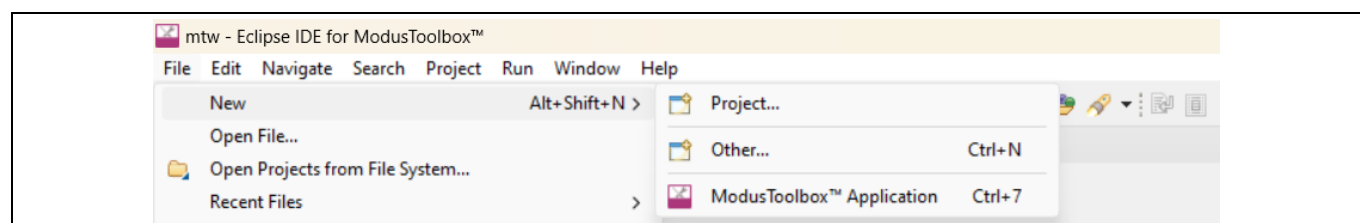
### 8.1 Create and build an application

The following workflow creates the example applications for the Connected Sensor Kit (CSK). Once finished, you can get further instructions in the respective *README.md* file which will open in the Eclipse IDE after creation.

Do the following:

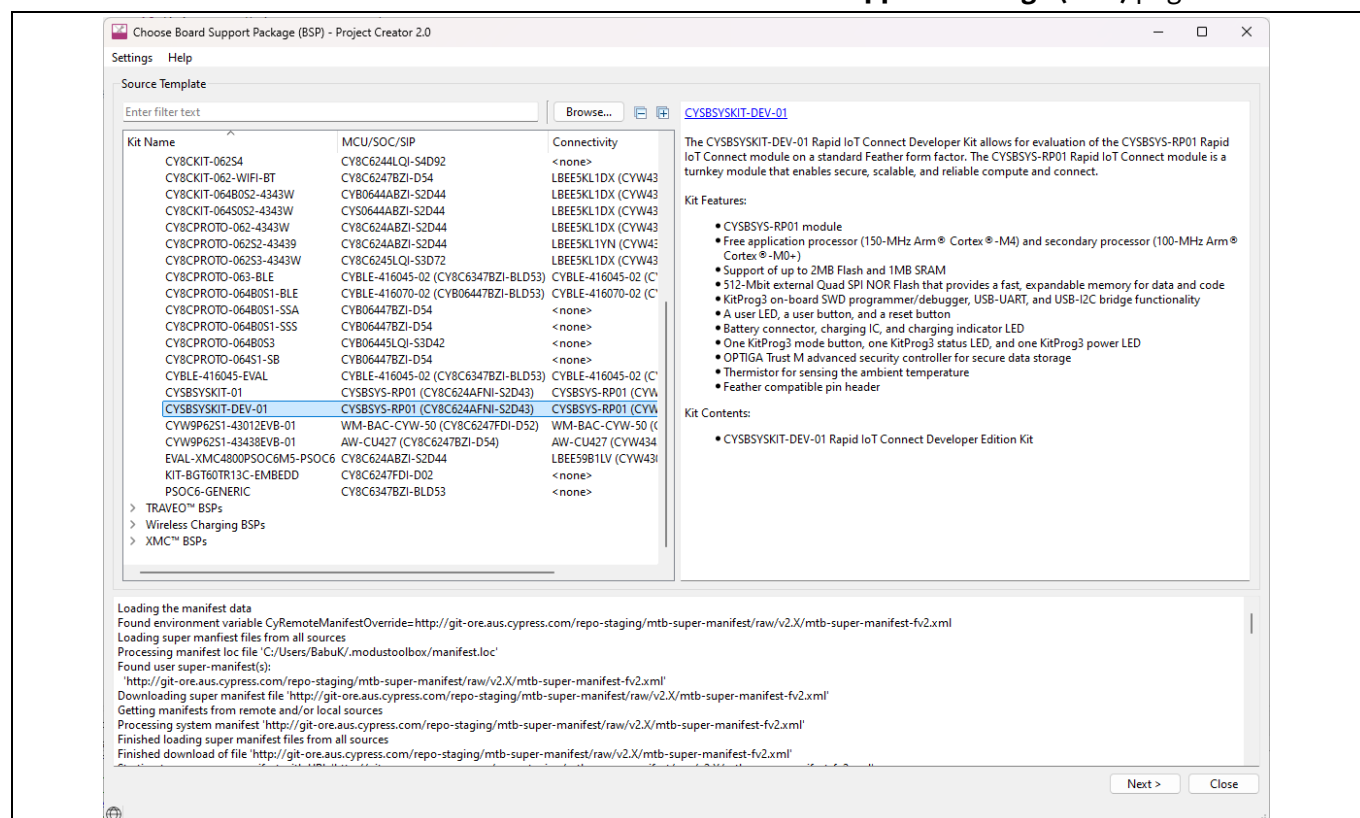
1. Launch Eclipse IDE for ModusToolbox™.
2. Click the **New Application** link in the **Quick Panel** (or use **File > New > ModusToolbox™ Application**).

This launches the **Project Creator** tool.



**Figure 6** Eclipse IDE for ModusToolbox™

3. Select the **CYSBSYSKIT-DEV-01** kit shown in the **Choose Board Support Package (BSP)** page.

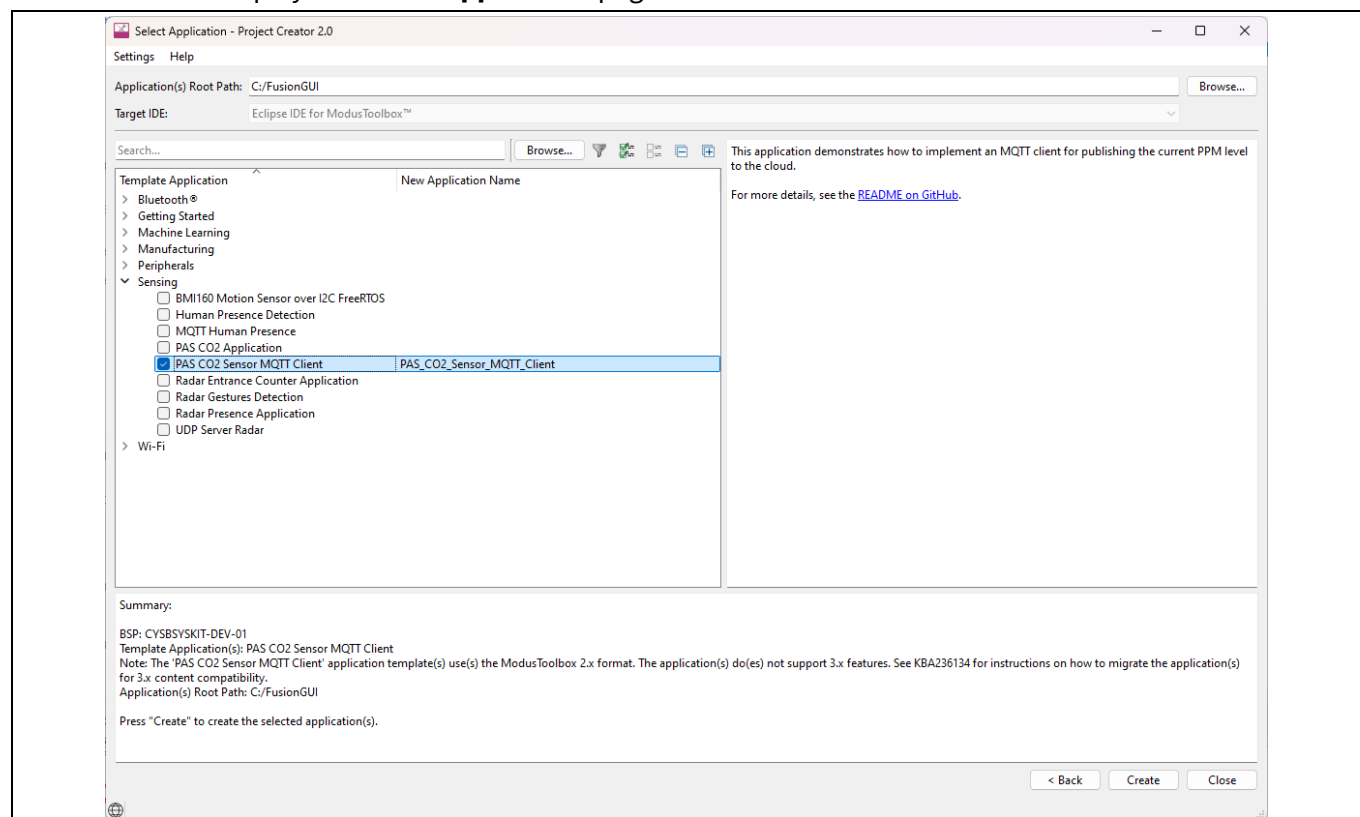


**Figure 7** Board Support Package

## Getting started guide

## Build and run the demo

4. Click **Next** to display the **Select Application** page.



**Figure 8** Select Application

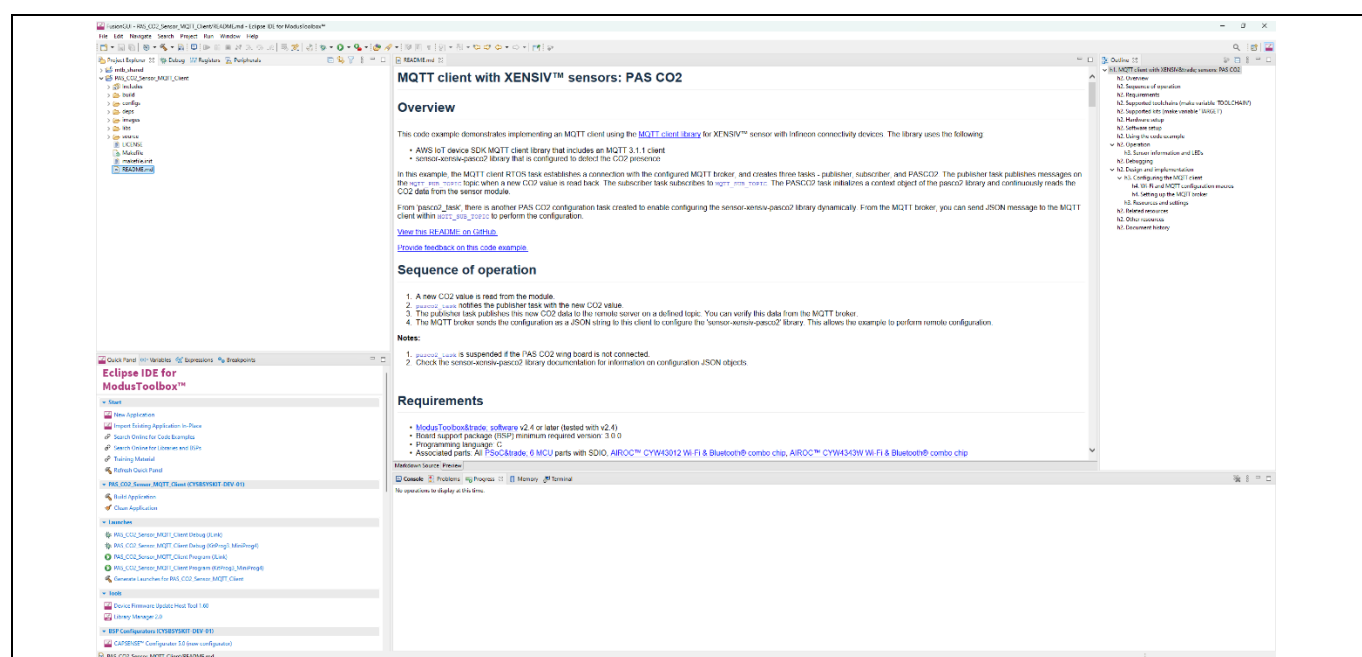
5. Optionally, enter the local path in the **Application(s) Root Path** field to indicate where the application needs to be created. Applications that can share libraries can be placed in the same root path.
6. Choose an example by enabling the check box. Optionally, change the suggested **New Application Name**.
7. Click **Create** to complete the application creation process.

**Note:** *When you select a supported kit, the example is reconfigured automatically to work with the kit. To work with a different supported kit later, use the Library Manager to choose the BSP for the supported kit.*

# XENSIV™ KIT CSK PASCO2 for Amazon Web Services IoT Core

## Getting started guide

### Build and run the demo



**Figure 9 Build and Program**

8. Build and program as you would for any ModusToolbox™ application in the Eclipse IDE.

## 8.2 Flash program image without creating an application

Do the following to program a HEX/BIN/ELF file already compiled for the CYSBSYSKIT-DEV-01 device:

1. Open a command console and navigate to the OpenOCD folder in the ModusToolbox™ installation directory. Default location: `<user-home>/ModusToolbox/tools_2.2/openocd`
2. Execute the following command. Replace `<file-path>` with the actual path of the HEX/BIN/ELF file on your disk:

```
bin\openocd.exe -s scripts -f interface/kitprog3.cfg -f target/psoc6_2m.cfg -c "program <file-path> verify reset exit"
```

For example:

```
bin\openocd.exe -s scripts -f interface/kitprog3.cfg -f target/ psoc6_2m.cfg -c "program c:/testing/myproject.hex verify reset exit"
```

## 8.3 Configure the parameters

Modify the user configuration files in the `configs` directory as follows:

### 8.3.1 Set Wi-Fi credentials

1. Navigate to the `configs/wifi_config.h` file in the ModusToolbox™ Project Explorer.
2. Modify the following macros to match with that of the Wi-Fi network that you want to connect:

- WIFI\_SSID
- WIFI\_PASSWORD
- WIFI\_SECURITY



### 8.3.2 Configure AWS IoT Core for MQTT

Set up the MQTT Client and configure:

1. Open an account and sign in as described in the [Getting Started with AWS IoT Core](#) developer guide.
2. Create a policy.
3. Ensure that the policy associated with this device permits all MQTT operations (\*iot: Connect\*, \*iot: Publish\*, \*iot: Receive\*, and \*iot: Subscribe\*)
  - Action: iot: \*
  - Resource ARN: \*
  - Effect: Allow
4. Set up the MQTT device (also known as a ‘Thing’) in AWS IoT Core.
5. Add a device certificate for your thing.
6. Activate and download a certificate for the AWS IoT thing.
  - Certificate: xxxxxxxxxx.certificate.pem.crt
  - Public key: xxxxxxxxxx.public.pem.key
  - Private key: xxxxxxxxxx.private.pem.key
  - Root CA ‘RSA 2048 bit key: Amazon Root CA 1’ for AWS IoT from CA certificates for server authentication - xxxxxxxxxx.AmazonRootCA1.pem
7. Add a policy for your Thing:
  - a) Select the policy you created to the certificate.
  - b) Register the policy: Go to **Manage > Things > Interact**.

### 8.3.3 Navigate to *configs/mqtt\_client\_config.h* in Project Explorer

1. Set `MQTT_BROKER_ADDRESS` to your custom endpoint on the settings page of the AWS IoT console.  
``ABCDEF1234567.iot.<region>.amazonaws.com``.
2. Using these certificates and keys, enter the following parameters in *mqtt\_client\_config.h* in Privacy-Enhanced Mail (PEM) format:
  - `CLIENT_CERTIFICATE` – xxxxxxxxxx.cert.pem
  - `CLIENT_PRIVATE_KEY` – xxxxxxxxxx.private.key
  - `ROOT_CA_CERTIFICATE` – Root CA certificate

You can either convert the values to strings manually following the format shown in *mqtt\_client\_config.h* or you can use the HTML utility available in the following link to convert the certificates and keys from PEM format to C string format. You need to clone the repository from [GitHub](#) to use the utility.

### 8.3.4 Program board using Eclipse IDE for ModusToolbox™

1. Select the application project in **Project Explorer**.
2. In the **Quick Panel**, scroll down, and click **Application Name > Program (KitProg3\_MiniProg4)**.
3. Open a terminal program, select the **KitProg3 COM** port and set the serial port parameters to **8N1** and **115200** baud.

## 8.4 Run the demo

Install a terminal emulator. Instructions in this document use Tera Term. If using Eclipse IDE for ModusToolbox™, do the following:

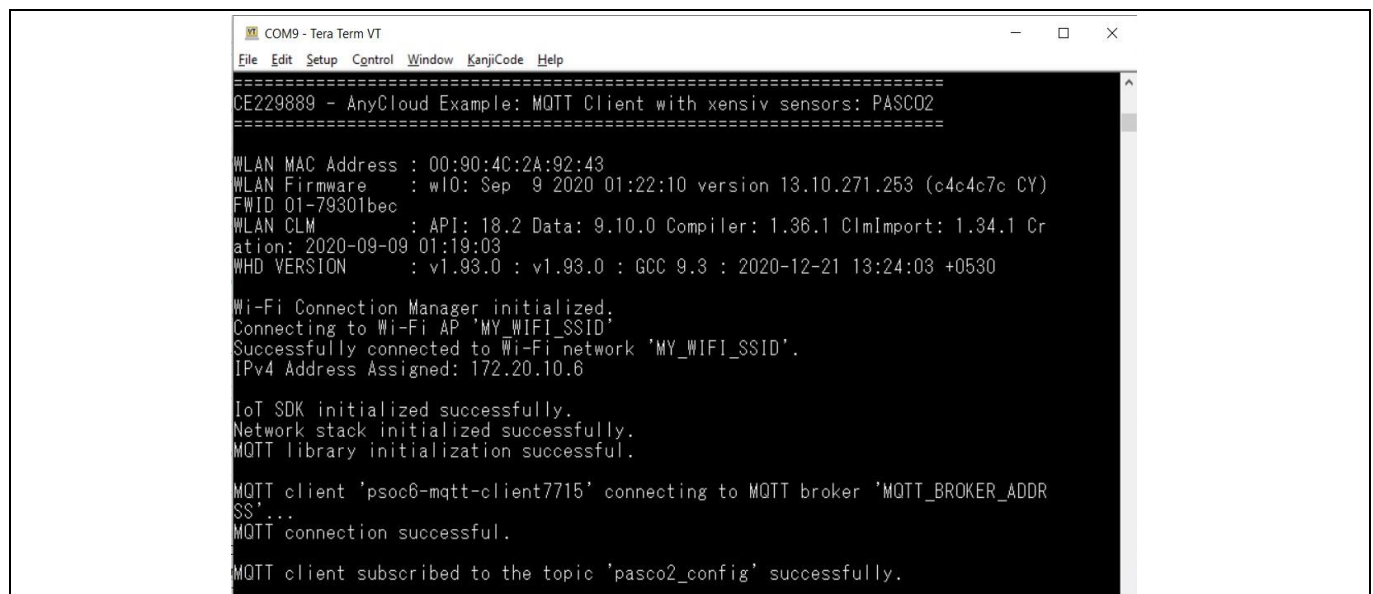
1. Open a new application. Select **File > New > ModusToolbox™ Application**.

This launches the **Project Creator** tool.

2. Pick **CYSBSYSKIT-DEV-01** from the list shown in **Project Creator**.
3. Select **PAS CO2 Sensor MQTT Client**.

## 8.5 Verify messages in AWS IoT Core

After programming, the application starts automatically. Observe the messages on the UART terminal and wait for the device to make all the required connections.



```
COM9 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
=====
CE229889 - AnyCloud Example: MQTT Client with xensiv sensors: PASCO2
=====
WLAN MAC Address : 00:90:4C:2A:92:43
WLAN Firmware   : wl0: Sep  9 2020 01:22:10 version 13.10.271.253 (c4c4c7c CY)
FWID 01-79301bec
WLAN CLM        : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport: 1.34.1 Cr
ation: 2020-09-09 01:19:03
WHD VERSION     : v1.93.0 : v1.93.0 : GCC 9.3 : 2020-12-21 13:24:03 +0530

Wi-Fi Connection Manager initialized.
Connecting to Wi-Fi AP 'MY_WIFI_SSID'
Successfully connected to Wi-Fi network 'MY_WIFI_SSID'.
IPv4 Address Assigned: 172.20.10.6

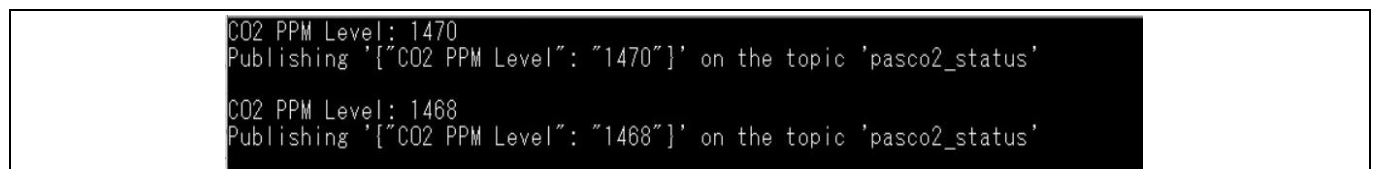
IoT SDK initialized successfully.
Network stack initialized successfully.
MQTT library initialization successful.

MQTT client 'psoc6-mqtt-client7715' connecting to MQTT broker 'MQTT_BROKER_ADDR
SS'...
MQTT connection successful.

MQTT client subscribed to the topic 'pasco2_config' successfully.
```

**Figure 10** Initialization status on the serial terminal

Once the initialization is complete, confirm that subscription to the topic is successful. If the PASCO2V01 Wing board is connected and detect a sensor event, the following messages are displayed.



```
CO2 PPM Level: 1470
Publishing '{"CO2 PPM Level": "1470"}' on the topic 'pasco2_status'

CO2 PPM Level: 1468
Publishing '{"CO2 PPM Level": "1468"}' on the topic 'pasco2_status'
```

**Figure 11** Event detection message from MQTT

## Getting started guide

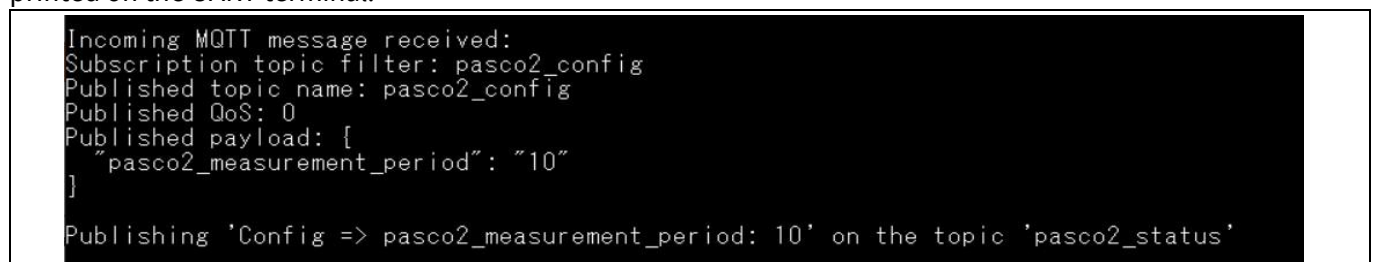
### Build and run the demo

Event detected message on MQTT broker:



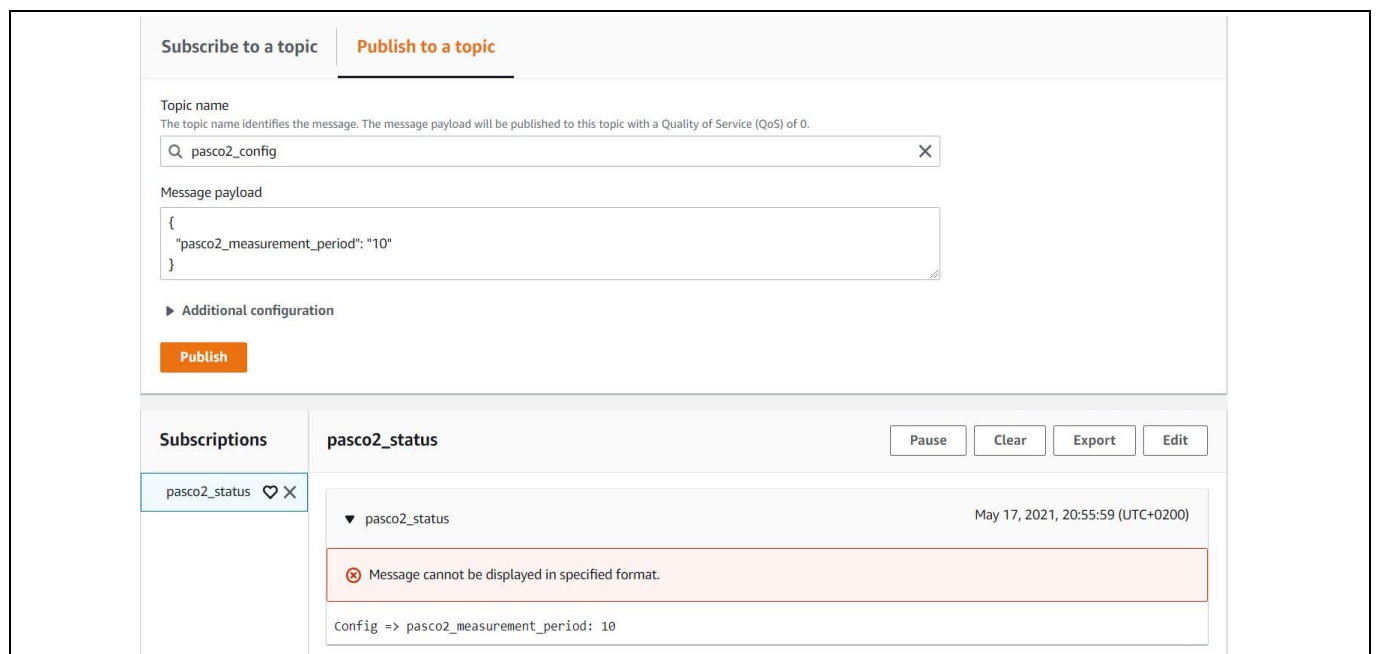
**Figure 12** CO<sub>2</sub> PMM level value on the MQTT broker

When the configuration is received from the MQTT broker, the messages received on the subscribed topic are printed on the UART terminal.



**Figure 13** Configuration message (publisher)

The configuration message is received on the MQTT broker (Ignore the warning message, as it indicates that the received message is not in JSON format.):




**Figure 14** Configuration message (receiver)

## 9 Troubleshooting

### Password missing after new user sign-up

- Possible cause:
  - Email has gone to junk/spam folder of your mailbox.
  - You may have entered an incorrect email ID while registering.
- Possible solution:
  - Check your spam/junk folders for an email from [no-reply@infineon.com](mailto:no-reply@infineon.com) with your username and temporary password. You will be prompted to change the password upon logging for the first time.
  - Verify the email address entered during user registration. If it is incorrect, register again with the correct email address.

### Build failed or status not updated

- Possible cause:
  - Mismatch in the serial number of the Rapid IoT Connect Developer Kit.
  - Device is already provisioned.
  - 505 error ('HTTP Version Not Supported').
- Possible solution:
  - Click the Ellipsis icon  at the end of the devices list to open the drop-down menu and click **Edit** to restart the **Quick IoT Experience Wizard**. Verify the details that you entered, such as the kit serial number, Wi-Fi SSID, and Wi-Fi password.
  - Check whether the device is already registered by anyone with a different email address. If so, contact [Infineon Support](#) to delete the old entry.

### Unable to program the kit after downloading the package from the dashboard


- Possible cause:
  - You tried to run the `program_kit` command without extracting the ZIP package.
  - You do not have permission to execute/run the script.
  - The path where you extracted the package to has a blank space.
  - Another program has control of the Rapid IoT Connect Developer Kit's KitProg3 interface.
  - Two or more devices are connected to the PC.
  - USB device is not recognized because the USB cable is incorrect.
  - Problem unknown.
- Possible solution:
  - Ensure that you extract the ZIP package first and then run the `program_kit` from the extracted folder based on the operating system and bundle. Check whether the device is connected to the PC before running the script.
  - Linux and macOS users: Ensure that you have permissions to run the script.
  - Check for empty space in the extracted folder name. If so, rename the folder without empty space.
  - Close CYPRESS™ programmer, PSOC™ Programmer, or ModusToolbox™ if running in the background and retry programming.
  - Remove all other Infineon kits, KitProg3 programmers, MiniProg4 programmers and ensure that the correct device is connected and retry the programming.

## Getting started guide

### Troubleshooting

- Try with a different USB cable.
- For more details, check the log file generated inside the *log* folder created in the same path as the script.

#### Device not connected to cloud after successful programming

- Possible cause:
  - Wi-Fi connection timed-out.
  - Incorrect SSID/password.
  - Wi-Fi network does not have internet connectivity.
  - SDIO error.
  - User LED blinking, indicating a hardware fault.
- Possible solution:
  - Open the debug serial interface in your PC and press the **RST\_BTN** button on the Rapid IoT Connect Developer Kit to reset the node. The live connection status is established. Mostly, it may solve the problem.
  - Click on the ellipsis icon  at the end of the devices list and select **Edit** from the drop-down menu. This will open the **Quick IoT Experience Wizard** with all data entered earlier. Verify that the entered kit serial number, Wi-Fi SSID, Wi-Fi password, etc. are correct. If not, update the details and try with the updated **Quick IoT Experience Wizard**.
  - Open the debug serial interface (for example, Tera Term) in your PC and press the **RST\_BTN** button on the Rapid IoT Connect Developer Kit to reset the node. On the serial terminal, an error message appears which says, “connection to Wi-Fi failed”, which implies that the device does not have internet access. Create a hotspot with the username as ‘IFX\_Sensor’ and password as ‘S66M14022021’ from your mobile network. Check whether your Wi-Fi network has internet access from another connected device and then resolve the firewall settings.
  - Open the debug serial interface in your PC and press the **RST\_BTN** button on the Rapid IoT Connect Developer Kit to reset the node. If the message “Unable to connect” appears, it implies that the device is not getting sufficient power from the USB port. Connect the node to a USB 3.0 port or to a USB charger with at least 1 A output capacity and retry the connection.
  - User LED blinking while booting up/after reset. This indicates that the device build has issues. Reflash the node by running the script again. If the problem persists, rebuild the package by restarting the **Quick IoT Experience Wizard**. Contact Infineon if the issue is unresolved.

#### Sensor not found

- Possible cause:
  - You have selected an incorrect application (other than ‘Thermistor’), which does not have Sensor\_Solution attributes with an Infineon XENSIV™ PASCO2V01 Wing board.
- Possible solution:
  - Select **Thermistor** from the drop-down menu of the Sensor\_Solution under the **Attributes** tab to resolve the issue.

#### Node not appearing in fleet monitoring

- Possible cause:
  - “Unable to obtain semaphore for Wi-Fi-scan for CP” messages may appear rarely on the UART terminal. Note that when enabling the location service feature on the cloud, the software on the node may rarely fail to get a semaphore lock for the Wi-Fi scan. The custom processor (CP) and software are running on

### Troubleshooting

the CM4 core. The Wi-Fi scan function is shared by both the cores; this situation arises when the CM4 core is unable to get access to Wi-Fi scan.

- Possible solution:
  - This issue affects only the location service feature. If this issue occurs, reset the node so that the CM4 node can successfully scan the Wi-Fi networks, and the fleet monitoring service can get the device details.

### SSID 'IFX\_Sensor' cannot be altered

- Possible cause:
  - 'IFX\_Sensor' is the default SSID and password created to create a quick hotspot.
- Possible solution:
  - Do one of the following:
    - Create a personalized hotspot.
    - Add your local network and select it while building the package.

### 'program\_kit.command' does not close after programming

- Possible cause:
  - macOS is not allowing the window to close on its own.
- Possible solution:
  - Close the terminal after "[Process Completed]" appears on the screen.

### References

- [1] Infineon Technologies AG: *XENSIV™ KIT CSK PASCO2 user guide*; [Available online](#)
- [2] Infineon Technologies AG: *Getting started with PSOC™ 6 MCU on ModusToolbox™ application note*; [Available online](#)
- [3] Infineon Technologies AG: *Code examples for ModusToolbox™*; [Available online](#)
- [4] Amazon: *AWS IoT Core Documentation*; [Available online](#)

### Glossary

### Glossary

**AA**

*active authentication (AA)*

**SoM**

*system-on-module (SoM)*

**CSK**

*connected sensor kit (CSK)*

**HAL**

*hardware abstraction layer (HAL)*

**PAS**

*photoacoustic spectroscopy (PAS)*

**MQTT**

*Message Queuing Telemetry Transport (MQTT)*

**IoT**

*Internet of Things (IoT)*

**SSID**

*Service Set Identifier (SSID)*

**SDIO**

*Secure Digital Input Output (SDIO)*



#### Revision history

Document revision	Date	Description of changes
1.00	2024-10-01	Initial release

#### Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2024-10-01**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**

**© 2024 Infineon Technologies AG.  
All Rights Reserved.**

**Do you have a question about this document?**

**Email:**

[erratum@infineon.com](mailto:erratum@infineon.com)

**Document reference**

**UG094313**

#### Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

#### Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.