# TrustFLEX Step by Step Guide
# Firmware Validation

# Table of Contents

# 1 Introduction

This document gives a detailed walk through of the Firmware Validation use case implementation. If familiar with Jupyter Notebook, can skip this section and move to Section 2.

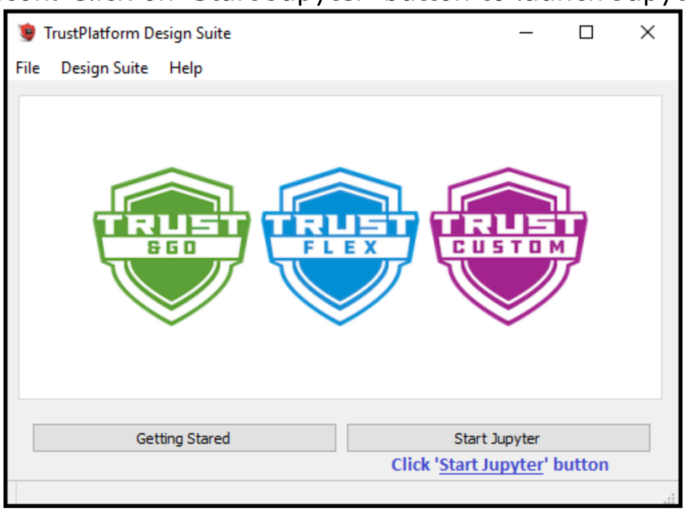
## 1.1 Getting started with Jupyter Notebook Tutorials

Jupyter Notebook is open source web application which allows you to create documents that contain code that you can execute in place as well as narrative text. It provides GUI elements, ability to execute code in place, ability to add images and gives it the look and feel that normal code files lack.

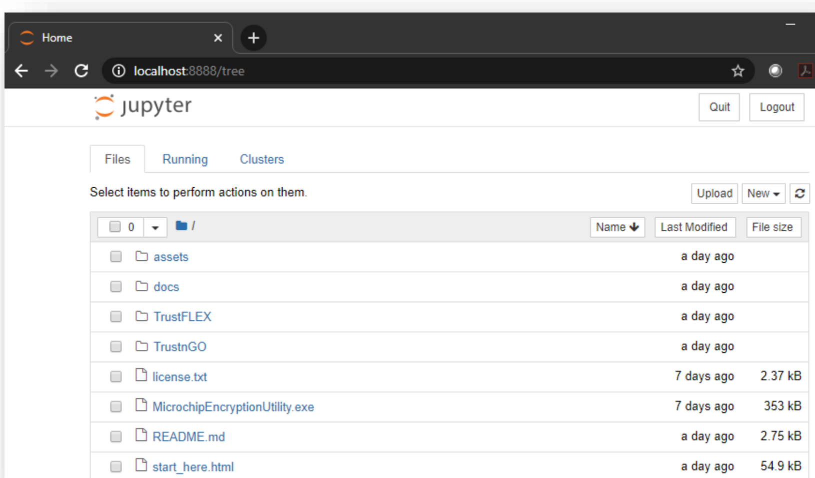
Jupyter notebooks are mainly used to explain/evaluate code in an interactive way.

### 1.1.1 Starting Jupyter Notebook

Jupyter notebook can be launched from Trust Platform GUI Main window. Run START -> Trust Platform x.x.x icon. Click on 'Start Jupyter' button to launch Jupyter local server.




Clicking on Start Jupyter should be web browser tab like below,



## 1.2 Jupyter Notebook Basics

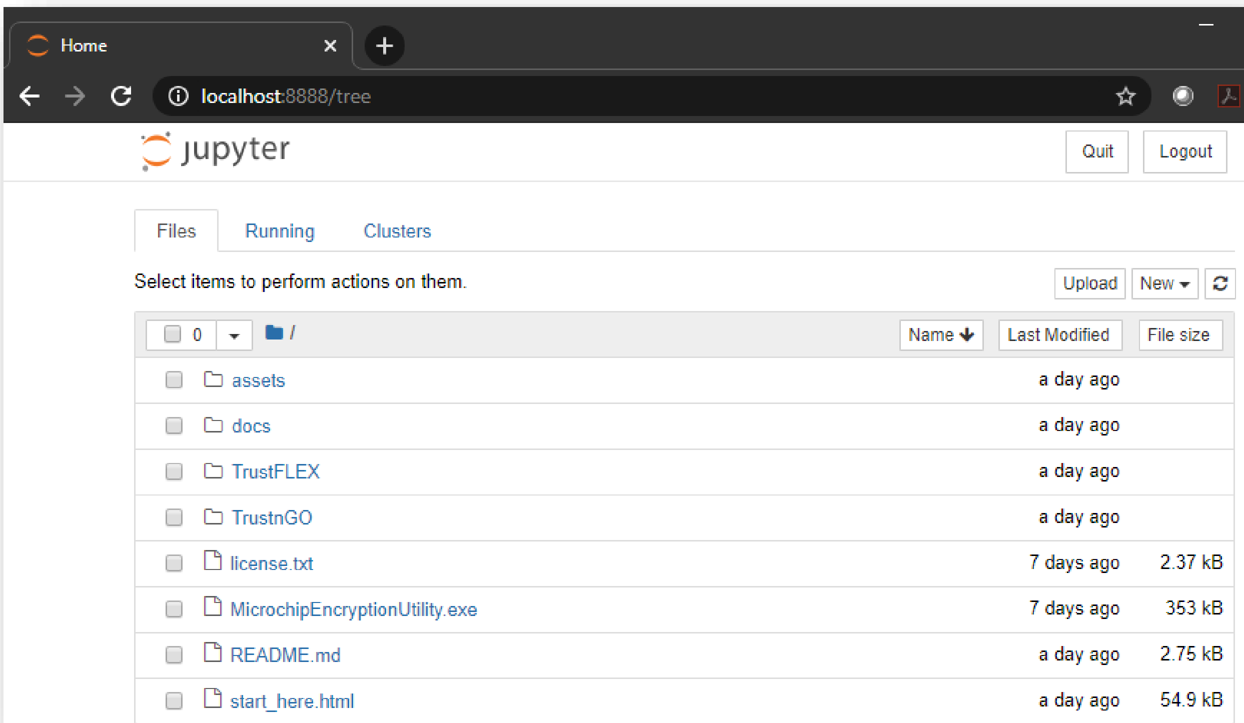
It is recommended to become familiar with Jupyter basic concepts with the online documentation, [https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html](https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html)

Some of the content is duplicated here for convenience. The online documentation should always be used as a reference.

### 1.2.1 The Notebook dashboard

When you first start the notebook server, your browser will open to the notebook dashboard. The dashboard serves as a home page for the notebook. Its main purpose is to display the notebooks and files in the current directory.

For example, here is a screenshot of the Jupyter dashboard. The top of the notebook list displays clickable breadcrumbs of the current directory. By clicking on these breadcrumbs or on sub-directories in the notebook list, you can navigate your file system.

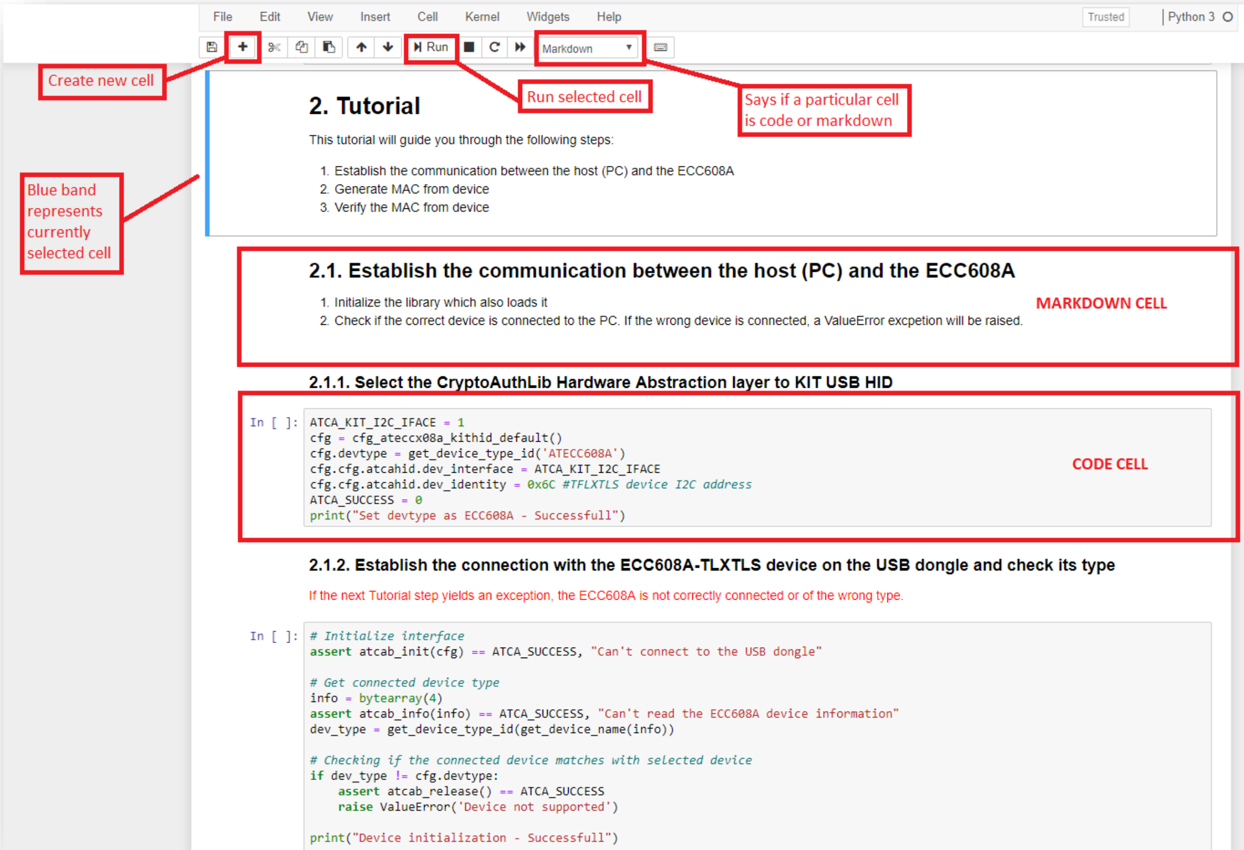



## 1.3 Introduction to Jupyter Notebook GUI.

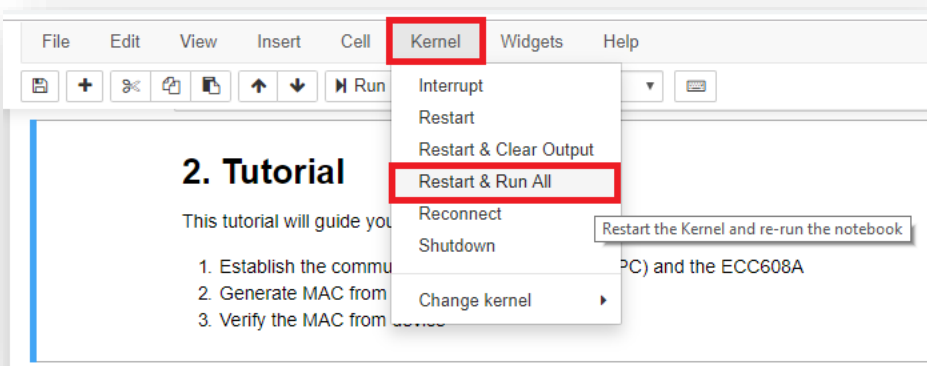
Jupyter Notebooks contain cells where you can either write code or markdown text. Notebooks contain multiple cells, some set as code and others markdown. Code cells contain code that can be executed live, and markdown contains text and images to explain the code.

Below image shows some options in a typical Jupyter Notebook. Individual cells can be executed by pressing on the RUN button as shown in the below image.

All cells in the Notebook can be executed in order by **Kernel->Restart & Run All**.




To run all cells in sequence.



# 2 Jupyter Notebook Tutorials

The TrustPlatform Design Suite comes with Notebook Tutorials to easily prototype popular use cases for TrustFLEX. Here is the list of Jupyter Notebook Tutorials.

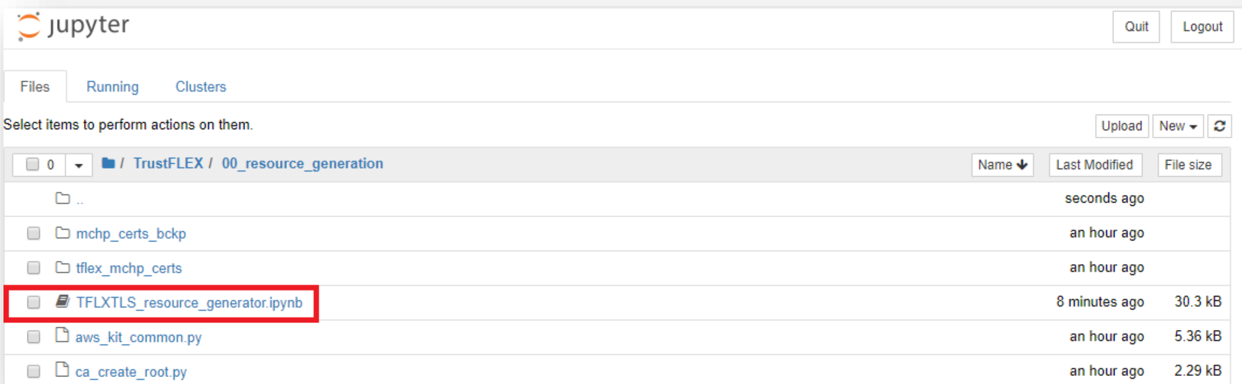
| Jupyter Notebook Tutorials | Relative Path | Applicable Devices |
|---|---|---|
| Manifest Generation | TrustnGO\00_resource_generation\TNGTLS_manifest_file_generation.ipynb | TrustnGO |
| Resource Generation | TrustFLEX\00_resource_generation\TFLXTLS_resource_generator.ipynb | TrustFLEX |
| Accessory Authentication | TrustFLEX\01_accessory_authentication\notebook\ TFLXTLS_accessory_authentication.ipynb | TrustFLEX |
| Firmware Validation | TrustFLEX\02_firmware_validation\notebook\ TFLXTLS_firmware_validation.ipynb | TrustFLEX |
| GCP Connect | TrustFLEX\03_gcp_connect\notebook\TFLXTLS_GCP_connect.ipynb | TrustFLEX |
| IP Protection | TrustFLEX\04_ip_protection\notebook\ TFLXTLS_IP_protection.ipynb | TrustFLEX |
| Secure Public Key Rotation | TrustFLEX\05_public_key_rotation\notebook\ TFLXTLS_public_key_rotation.ipynb | TrustFLEX |
| AWS Custom PKI | TrustFLEX\06_custom_pki_aws\notebook\ TFLXTLS_aws_connect.ipynb | TrustFLEX |
| Azure Connect | TrustFLEX\07_custom_pki_azure\notebook\ TLFXTLS_azure_connect.ipynb | TrustFLEX |

# 3 Resource Generation Notebook

TFLXTLS device is one of the three devices available on CryptoAuth Trust Platform Development Kit. TrustFlex devices come with pre-programmed certificates in slots 10, 11 and 12, also slots 0-4 have pre-generated private keys, other than the mentioned slots all the other slots have no data in them.

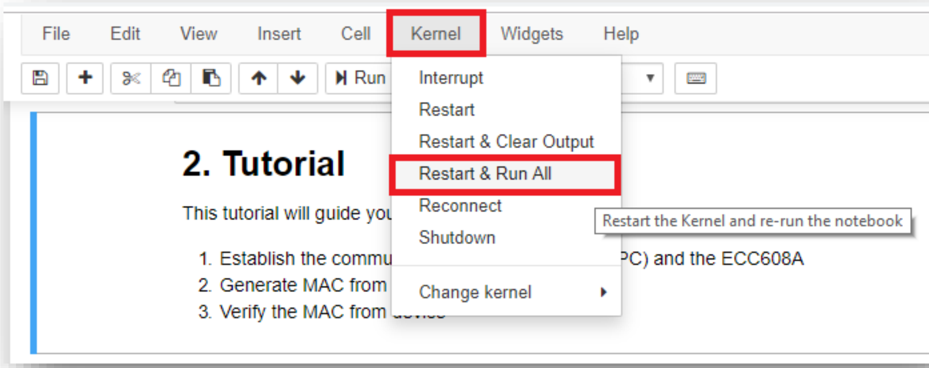
The Resource Generator Notebook will create development keys and certificates for all slots that can be updated. Generated Keys and Certificate chains are stored in the PC file system. These keys should never be used for production purposes as their generation is not handled in a secure environment. These development keys will be later used by the other notebooks to implement the various pre-defined use cases.

Within the Jupyter Dashboard, navigate **TrustFLEX\00_resource_generation** folder to open **TFLXTLS_resource_generator.ipynb** notebook



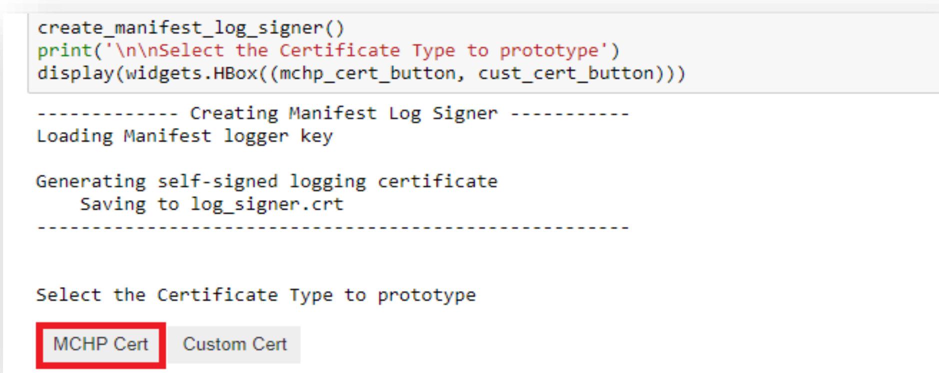
Run all cells of the Crypto Resource Generator Notebook: Kernel->Restart & Run All

**Note:** Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to [CryptoAuth Trust Platform Development Kit Factory reset]() section for reloading default program.




Resource Generator Notebook is common for all the use cases which comes with option to load the signer certificate and device certificate. It will execute and prompt you to choose between MCHP certificate and a custom certificate chain, press “MCHP Cert” option for this use case.

The Notebook will generate several keys and certificates. Make sure you have an error free output before continuing to the next steps of the training.



The output log should look like this.

```
------------------------------------------
MCHP Certs processing...
MCHP certificates found in the device

Backing up certificates from device

Backing up certificates from device - Success

----------------------------------------------------
Root Certificate:
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            77:d3:6d:95:6e:c8:ae:62:05:e5:8e:3a:cb:98:5a:81
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: O = Microchip Technology Inc, CN = Crypto Authentication Root CA 002
        Validity
            Not Before: Nov  8 19:12:19 2018 GMT
            Not After : Nov  8 19:12:19 2058 GMT
        Subject: O = Microchip Technology Inc, CN = Crypto Authentication Root CA 002
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:bd:54:e6:6d:e3:87:54:84:00:6b:53:ae:15:80:
                    d5:0a:a0:69:e7:8a:df:55:78:d8:5c:e2:d5:4d:d5:
                    b8:30:29:6b:ff:dd:6e:6f:72:56:fb:d9:9e:f1:a1:
                    16:b1:1d:33:ad:49:10:3a:a1:85:87:39:dc:fa:e4:
                    37:e1:9d:63:4e
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                7A:ED:7D:6D:C6:B7:78:9D:B2:38:01:A5:E8:4A:8C:B0:A4:0E:2A:8C
            X509v3 Authority Key Identifier:
                keyid:7A:ED:7D:6D:C6:B7:78:9D:B2:38:01:A5:E8:4A:8C:B0:A4:0E:2A:8C

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:21:00:a1:dc:63:45:90:ec:81:9e:e1:de:5b:81:12:
         65:51:ad:d4:c2:c4:f8:e5:95:28:2e:e0:4b:e7:68:ec:7c:02:
         73:02:20:3e:6b:a7:4e:9e:4c:0a:d6:8c:24:b0:fb:2e:e7:93:
         d2:e6:be:94:65:ca:15:d0:ea:5b:c8:7f:55:79:99:5c:ad
```

-----BEGIN CERTIFICATE-----
MIIB8TCCAZegAwIBAgIQd9NtlW7IrmIF5Y46y5hagTAKBggqhkjOPQQDAjBPMSEw
HwYDVQQKDBhNaWNyb2NoaXAgVGVjaG5vbG9neSBJbmMxKjAoBgNVBAMMIUNyeXB0
byBBdXRoZW50aWNhdGlvbiBSb290IENBIDAwMjAgFw0xODExMDgxOTEyMTlaGA8y
MDU4MTEwODE5MTIxOVowTzEhMB8GA1UECgwYTWljcm9jaGlwIFRlY2hub2xvZ3kg
SW5jMSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gUm9vdCBDQSAwMDIw

WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS9VOZt44dUhABrU64VgNUKoGnnit9V
eNhc4tVN1bgwKWv/3W5vclb72Z7xoRaxHTOtSRA6oYWHOdz65DfhnWNOo1MwUTAd
BgNVHQ4EFgQUeu19bca3eJ2yOAGl6EqMsKQOKowwHwYDVR0jBBgwFoAUeu19bca3
eJ2yOAGl6EqMsKQOKowwDwYDVR0TAQH/BAUwAwEB/zAKBggqhkjOPQQDAgNIADBF
AiEAodxjRZDsgZ7h3luBEmVRrdTCxPjllSgu4EvnaOx8AnMCID5rp06eTArWjCSw
+y7nk9LmvpRlyhXQ6lvIf1V5mVyt
-----END CERTIFICATE-----

Validate Root Certificate:
OK

Signer Certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            79:0a:a7:d5:7d:73:dc:e9:6d:65:db:66:8b:76:b2:5e
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: O = Microchip Technology Inc, CN = Crypto Authentication Root CA 002
        Validity
            Not Before: Dec 14 19:00:00 2018 GMT
            Not After : Dec 14 19:00:00 2049 GMT
        Subject: O = Microchip Technology Inc, CN = Crypto Authentication Signer F600
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:76:47:41:70:b2:63:e7:99:54:bc:85:bb:12:e9:
                    fe:70:0c:5b:8d:d4:d6:93:45:98:c2:29:a7:68:02:
                    0e:4e:0b:6d:48:75:d0:ed:a1:ee:f6:5f:91:5f:c6:
                    b1:16:46:c5:a1:ca:63:1f:62:55:68:74:47:69:c5:
                    de:83:b5:89:6a
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Subject Key Identifier:
                FB:DC:AA:12:8A:FA:C1:B5:92:8F:CD:AB:11:DB:09:3E:CF:4D:BE:F6
            X509v3 Authority Key Identifier:
                keyid:7A:ED:7D:6D:C6:B7:78:9D:B2:38:01:A5:E8:4A:8C:B0:A4:0E:2A:8C

    Signature Algorithm: ecdsa-with-SHA256
         30:46:02:21:00:c6:30:31:e9:a9:8b:30:4e:68:7e:06:c5:39:
         79:2a:c5:7a:5c:01:4d:30:17:de:dc:d2:7d:d5:1d:cd:86:37:
         ff:02:21:00:c6:a2:2c:6e:b1:ae:5f:85:91:49:cb:5d:e7:77:
         8b:a3:f3:0b:e9:3d:9b:80:6f:94:bf:3d:90:a5:84:78:61:dc
```

-----BEGIN CERTIFICATE-----
MIICBTCCAaqgAwIBAgIQeQqn1X1z3OltZdtmi3ayXjAKBggqhkjOPQQDAjBPMSEw
HwYDVQQKDBhNaWNyb2NoaXAgVGVjaG5vbG9neSBJbmMxKjAoBgNVBAMMIUNyeXB0
byBBdXRoZW50aWNhdGlvbiBSb290IENBIDAwMjAgFw0xODEyMTQxOTAwMDBaGA8y

MDQ5MTIxNDE5MDAwMFowTzEhMB8GA1UECgwYTWljcm9jaGlwIFRlY2hub2xvZ3kg
SW5jMSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gU2lnbmVyIEY2MDAw
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAR2R0FwsmPnmVS8hbsS6f5wDFuN1NaT
RZjCKadoAg5OC21IddDtoe72X5FfxrEWRsWhymMfYlVodEdpxd6DtYlqo2YwZDAO
BgNVHQ8BAf8EBAMCAYYwEgYDVR0TAQH/BAgwBgEB/wIBADAdBgNVHQ4EFgQU+9yq
Eor6wbWSj82rEdsJPs9NvvYwHwYDVR0jBBgwFoAUeu19bca3eJ2yOAGl6EqMsKQO
KowwCgYIKoZIzj0EAwIDSQAwRgIhAMYwMempizBOaH4GxTl5KsV6XAFNMBfe3NJ9
1R3Nhjf/AiEAxqIsbrGuX4WRSctd53eLo/ML6T2bgG+Uvz2QpYR4Ydw=
-----END CERTIFICATE-----

Validate Signer Certificate:
OK

Device Certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5a:cb:a3:f7:cf:bf:c5:28:92:cd:e1:9f:a3:ac:9d:17
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: O = Microchip Technology Inc, CN = Crypto Authentication Signer F600
        Validity
            Not Before: Aug 21 22:00:00 2019 GMT
            Not After : Aug 21 22:00:00 2047 GMT
        Subject: O = Microchip Technology Inc, CN = 0123867D566FFB7701 ATECC
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:fc:57:67:b6:fb:ae:50:60:ca:96:5a:ef:41:b1:
                    c5:d6:a1:60:61:87:8e:a4:78:f4:4d:18:d0:76:9d:
                    ad:62:24:b3:68:c2:1a:62:cb:0a:fd:ef:f5:b4:0c:
                    e3:55:ec:f0:40:bb:41:83:61:02:ef:20:3c:63:93:
                    32:d4:90:41:ab
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage: critical
                Digital Signature, Key Agreement
            X509v3 Subject Key Identifier:
                43:9E:4F:45:79:35:CE:DC:D4:35:B9:4F:4A:23:69:E1:2D:89:33:04
            X509v3 Authority Key Identifier:
                keyid:FB:DC:AA:12:8A:FA:C1:B5:92:8F:CD:AB:11:DB:09:3E:CF:4D:BE:F6

    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:21:00:83:32:78:25:9c:5a:07:7c:4a:04:f8:b5:c4:
         57:d6:08:70:ee:c3:d4:79:9c:b6:14:8e:5e:86:54:38:50:cf:
         ec:02:20:58:e1:cf:e1:f6:e2:17:08:c3:5a:fc:86:91:31:ef:
         65:09:e0:e4:ba:7e:02:8e:4c:49:d1:4b:e3:ac:35:33:f7
```

-----BEGIN CERTIFICATE-----
MIIB9TCCAZugAwIBAgIQWsuj98+/xSiSzeGfo6ydFzAKBggqhkjOPQQDAjBPMSEw

HwYDVQQKDBhNaWNyb2NoaXAgVGVjaG5vbG9neSBJbmMxKjAoBgNVBAMMIUNyeXB0
byBBdXRoZW50aWNhdGlvbiBTaWduZXIgRjYwMDAgFw0xOTA4MjEyMjAwMDBaGA8y
MDQ3MDgyMTIyMDAwMFowRjEhMB8GA1UECgwYTWljcm9jaGlwIFRlY2hub2xvZ3kg
SW5jMSEwHwYDVQQDDBgwMTIzODY3RDU2NkZGQjc3MDEgQVRFQ0MwWTATBgcqhkjO
PQIBBggqhkjOPQMBBwNCAAT8V2e2+65QYMqWWu9BscXWoWBhh46kePRNGNB2na1i
JLNowhpiywr97/W0DONV7PBAu0GDYQLvIDxjkzLUkEGro2AwXjAMBgNVHRMBAf8E
AjAAMA4GA1UdDwEB/wQEAwIDiDAdBgNVHQ4EFgQUQ55PRXk1ztzUNblPSiNp4S2J
MwQwHwYDVR0jBBgwFoAU+9yqEor6wbWSj82rEdsJPs9NvvYwCgYIKoZIzj0EAwID
SAAwRQIhAIMyeCWcWgd8SgT4tcRX1ghw7sPUeZy2FI5ehlQ4UM/sAiBY4c/h9uIX
CMNa/IaRMe9lCeDkun4CjkxJ0UvjrDUz9w==
-----END CERTIFICATE-----

Validate Device Certificate:
OK

------------------------------------------------------
Generated the manifest file 0123867d566ffb7701_manifest.json
MCHP Certificate processing completed successfully

------------------------------------------------------

The Notebook will also generate a manifest file to be uploaded into the public cloud of your choice (Google GCP, AWS IoT and Microsoft Azure).

After running this Notebook, it generates the required resources and program data zone with required secrets, keys and certificates. For this use case, IO protection key and firmware validation public key are loaded into TrustFLEX device in the slot 6 and 15 respectively.

# 4 Use Case Prototyping

This hands-on lab is intended to demonstrate the usage of TrustFLEX device to validate firmware that going to run on HostMCU. It uses asymmetric authentication.

To validate the firmware, following steps to be followed

1. Generating a firmware Signing Key pair
2. Signing the firmware
3. Updating the firmware to product
4. Verifying the firmware image

OEM to take care of first 2 things in a controlled environment. To have firmware validation functionality, once the firmware implementation is completed it should be signed by the OEM firmware signer to make the image authentic. Typically, OEM firmware signer's public key will be loaded to secure element and locked permanently.

On the product side, the digest and signature generated in the previous step will be provided to secure element using Secure boot command. Secure boot command will be executed on secure element with option to store (Full Copy) on successful validation of the digest and signature.

On TrustFLEX device secure boot configuration is set as "FullDig", which stores the firmware digest on the device (slot 7 on TrustFLEX). On subsequent boots, the digest is compared without ECC verify operations. While sending the digest to TrustFLEX device, the digest is encrypted with IO protection key to avoid man in the middle attack.

This lab is setup such a way firmware sign operation taken care by notebook, update and verify operations can be done both in notebook and embedded project. Firmware sign operations are NOT done in embedded project as it's the role of OEM but not the product.

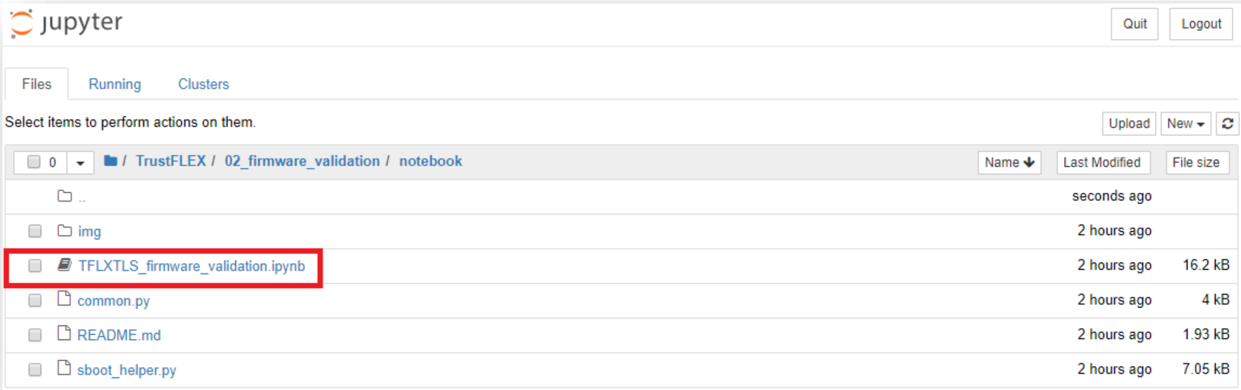
The resource generation for TrustFLEX device will load

1. A prototyping firmware signing key
2. A prototyping IO protection key to Slot6
3. Signers public key to Slot15 respectively

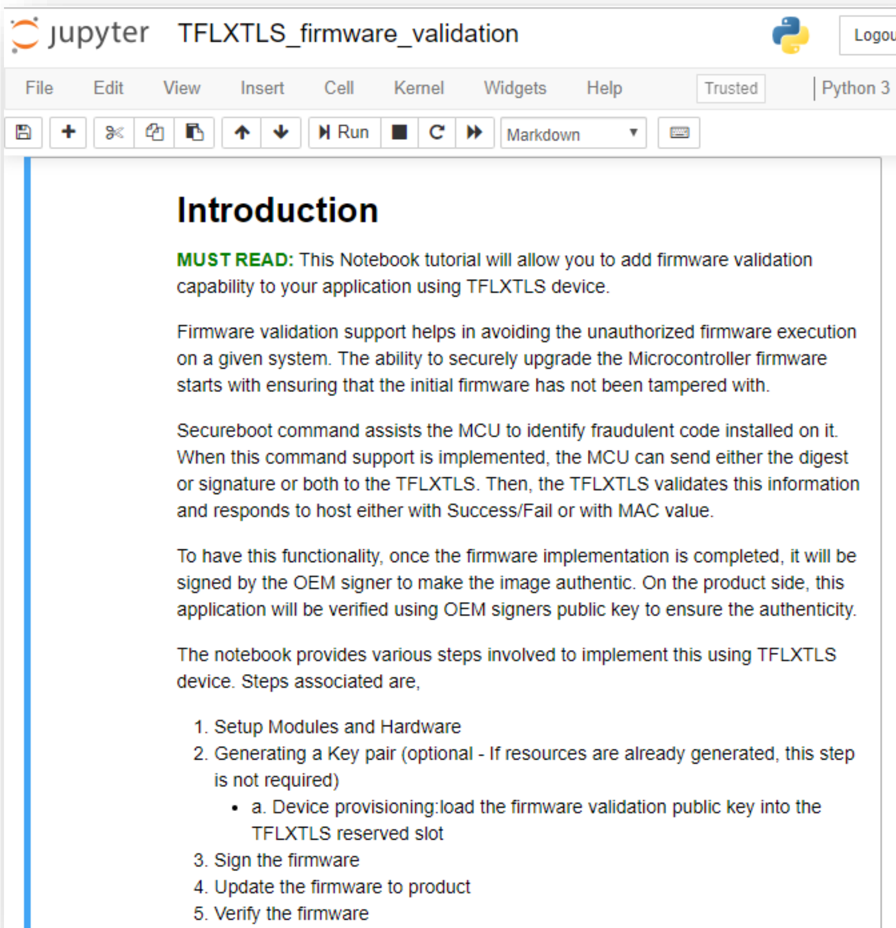
Following sections provide detail steps to execute the Usecase both on Jupyter Notebook and on Embedded project

## 4.1 Running Firmware Validation example on Jupyter Notebook:

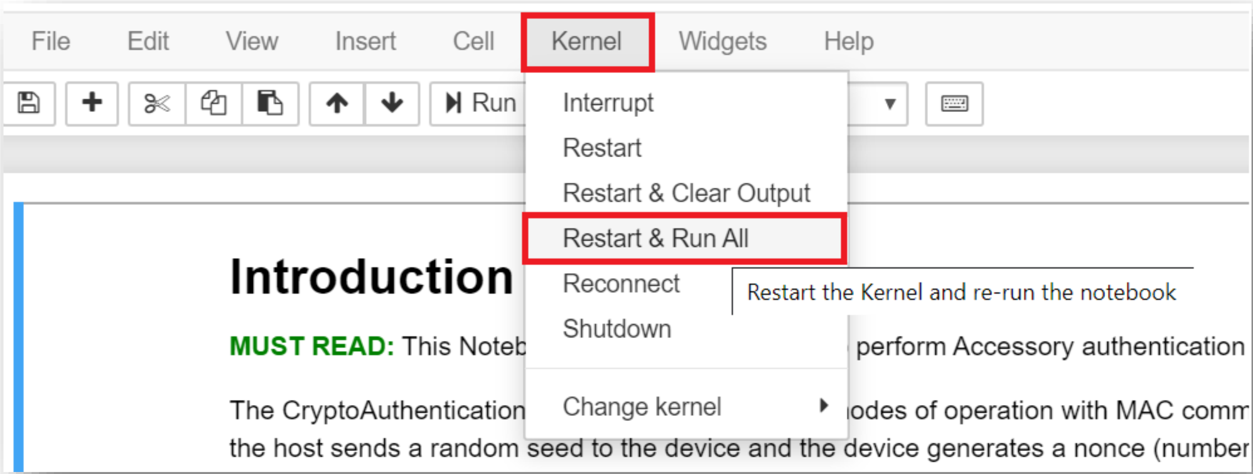
1. From the Jupyter Home page, navigate to **TrustFLEX\02_firmware_validation\notebook\TFLXTLS_firmware_validation.ipynb** notebook file and open it.



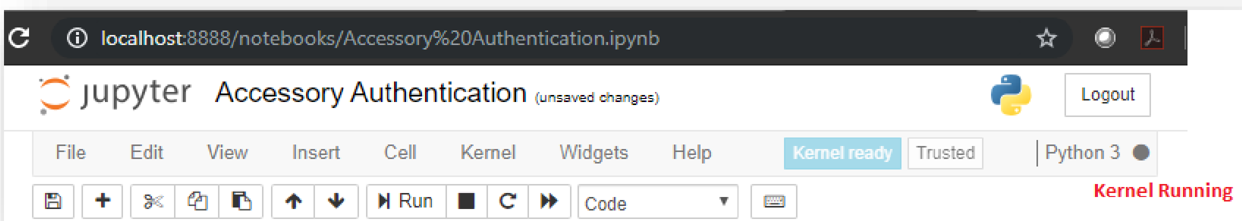

Opening the notebook from Jupyter home page should load the following on the browser,

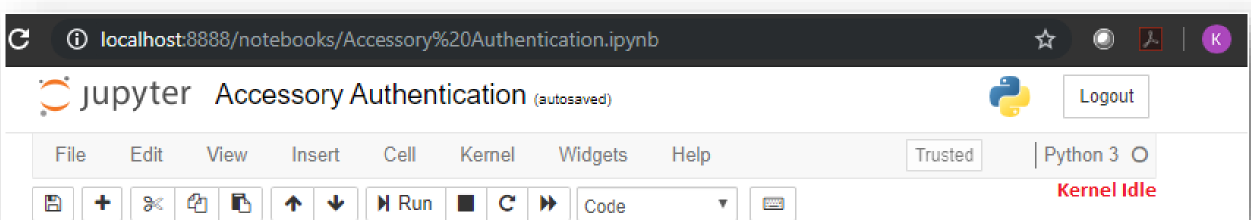



2. Run All Cells by using Kernel -> Restart & Run All




3. It may take a while to complete, wait for the kernel to complete all processing i.e. from Kernel Running to Kernel Idle state (Check circle above **RED** text)




4. Navigate through different cells output for the description of the step and result from the execution.

5. There are 4 major steps in this lab

   **Generating a Firmware Validation key pair**
   This step setups a temporary firmware signer to perform firmware validation process. This key generation is already taken care part of resource generation.

   **Sign the Firmware**
   This step generates firmware digest by hash the example firmware image with SHA 256 algorithm and get it signed with firmware signer's private key. Then digest will

be encrypted with IO protection key to avoid man in the middle attack before host send digest to the device.
Here is how the memory of the Microcontroller is portioned. Microcontroller has a 256KB flash starting from 0x0000 0000, supporting address range from 0x0000 0000 to 0x0003 FFFF.

| | |
|---|---|
| Firmware validation image | 0x0000 0000 to 0x0000 BFFF |
| Application image | 0x0000 C000 to 0x0003 FBFF |
| Signature data | 0x0003 FC00 to 0x0003 FFFF |

The firmware validation image and the application image can be obtained by building (compile + link) the respective projects in the correct address spaces, the signature will be calculated and stitched with the other images through Jupyter Notebooks.

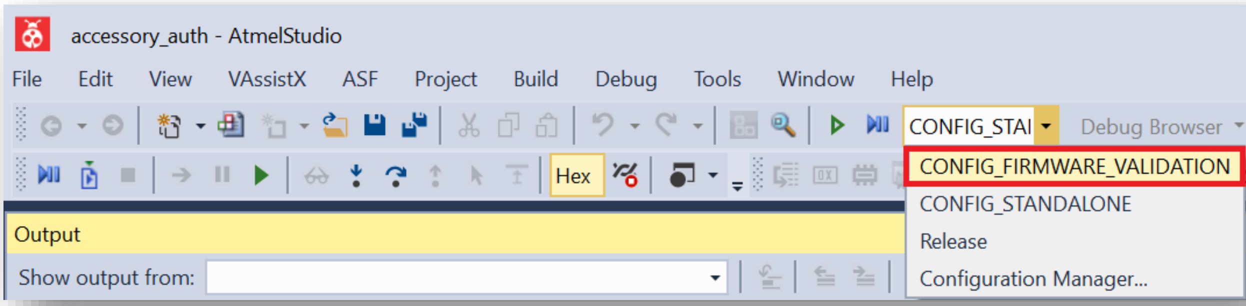
To get firmware validation hex, just navigate to **TrustFLEX\02_firmware_validation\c.** Open either MPLAB project or Atmel studio project and build the project. After successful build, it will create .hex file under **TrustFLEX\02_firmware_validation\notebook\firm_valid*.hex.** We will be using this hex file in future steps.

To get Application hex, open any of the use case example project either in MPLAB or Atmel studio. As discussed earlier, the application hex start address should be 0xC000. So, we need to change the build configuration to get output hex that starts from 0xC000.

The example applications in the DesignTools have two build configurations, one is **CONFIG_STANDALONE** where application image starts from 0x00000000 and another one is **CONFIG_FIRMWARE_VALIDATION** where application image starts from 0xC000. When Firmware validation feature is used, example application should be compiled using **CONFIG_FIRMWARE_VALIDATION** configuration.

ATMEL STUDIO:
Let’s use Accessory Authentication example as an application. To open project just navigate to **TrustFLEX\01_accessory_authentication\c\studio** and select **accessory_auth.atsln.**

Here we need to select the CONFIG_FIRMWARE_VALIDATION configuration to get application image start from 0xC000. Below screenshot display how to change the configuration,

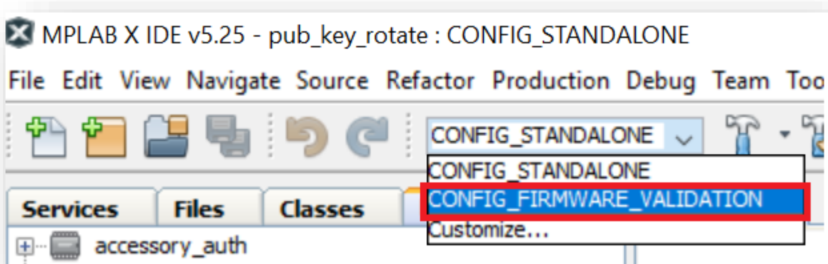



After changing the configuration, build the project. Once build is successful, it will create .hex file under **TrustFLEX\02_firmware_validation\notebook\accessory_auth*.hex.**

**Note**: Before reusing the application in standalone mode this configuration should be set back to CONFIG_STANDALONE**.**

MPLAB:
Let's use Accessory Authentication example as an application. To open project just navigate to **TrustFLEX\01_accessory_authentication\c\mplab** and select **accessory_auth.X.**

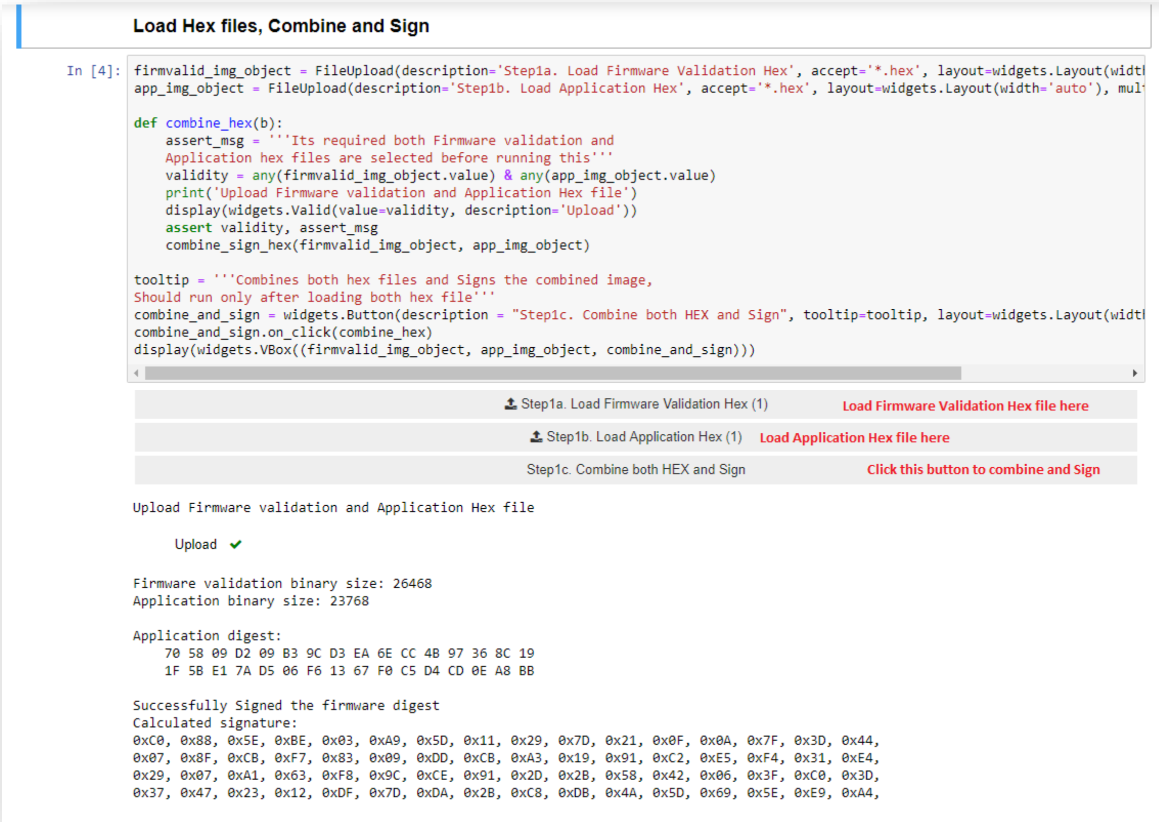
Here we need to select the CONFIG_FIRMWARE_VALIDATION configuration to get application image start from 0xC000. Below screenshot display how to change the configuration,




After changing the configuration, build the project. Once build successful, it will create .hex file under **TrustFLEX\02_firmware_validation\notebook\accessory_auth*.hex.**

**Note**: Before reusing the application in standalone mode this configuration should be set back to CONFIG_STANDALONE**.**

Now that we have all the binaries available, go back to the Firmware Validation Juyter Notebook. Go to step 2.3, this step accepts two hex files, combines then and appends signature to it. Follow the below snapshots for reference. Make sure the correct images are selected on Upload buttons.



Load Hex files, Combine and Sign

```
In [4]: firmvalid_img_object = FileUpload(description='Step1a. Load Firmware Validation Hex', accept='*.hex', layout=widgets.Layout(widtl
        app_img_object = FileUpload(description='Step1b. Load Application Hex', accept='*.hex', layout=widgets.Layout(width='auto'), mul

        def combine_hex(b):
            assert_msg = '''Its required both Firmware validation and
            Application hex files are selected before running this'''
            validity = any(firmvalid_img_object.value) & any(app_img_object.value)
            print('Upload Firmware validation and Application Hex file')
            display(widgets.Valid(value=validity, description='Upload'))
            assert validity, assert_msg
            combine_sign_hex(firmvalid_img_object, app_img_object)

        tooltip = '''Combines both hex files and Signs the combined image,
        Should run only after loading both hex file'''
        combine_and_sign = widgets.Button(description = "Step1c. Combine both HEX and Sign", tooltip=tooltip, layout=widgets.Layout(widtl
        combine_and_sign.on_click(combine_hex)
        display(widgets.VBox((firmvalid_img_object, app_img_object, combine_and_sign)))
```



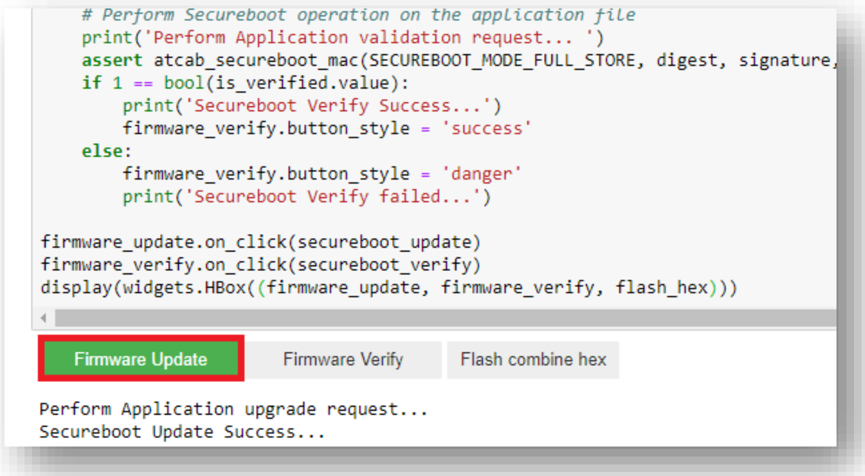
```
Upload Firmware validation and Application Hex file

      Upload ✔

Firmware validation binary size: 26468
Application binary size: 23768

Application digest:
    70 58 09 D2 09 B3 9C D3 EA 6E CC 4B 97 36 8C 19
    1F 5B E1 7A D5 06 F6 13 67 F0 C5 D4 CD 0E A8 BB

Successfully Signed the firmware digest
Calculated signature:
0xC0, 0x88, 0x5E, 0xBE, 0x03, 0xA9, 0x5D, 0x11, 0x29, 0x7D, 0x21, 0x0F, 0x0A, 0x7F, 0x3D, 0x44,
0x07, 0x8F, 0xCB, 0xF7, 0x83, 0x09, 0xDD, 0xCB, 0xA3, 0x19, 0x91, 0xC2, 0xE5, 0xF4, 0x31, 0xE4,
0x29, 0x07, 0xA1, 0x63, 0xF8, 0x9C, 0xCE, 0x91, 0x2D, 0x2B, 0x58, 0x42, 0x06, 0x3F, 0xC0, 0x3D,
0x37, 0x47, 0x23, 0x12, 0xDF, 0x7D, 0xDA, 0x2B, 0xC8, 0xDB, 0x4A, 0x5D, 0x69, 0x5E, 0xE9, 0xA4,
```

"Combine HEX" will combine the firmware validation hex, ip protection hex and will append the signature to it. The combined hex file will be store in the PC at DesignTool\ TrustFLEX\02_firmware_validation\notebook \combined_image.hex

At this step, we have the combined image available for firmware validation update and verify operations. Both update and verify can be performed on the Notebook itself or on embedded projects. Refer to section 4.2 for instructions on embedded projects.

**Update the firmware to product**

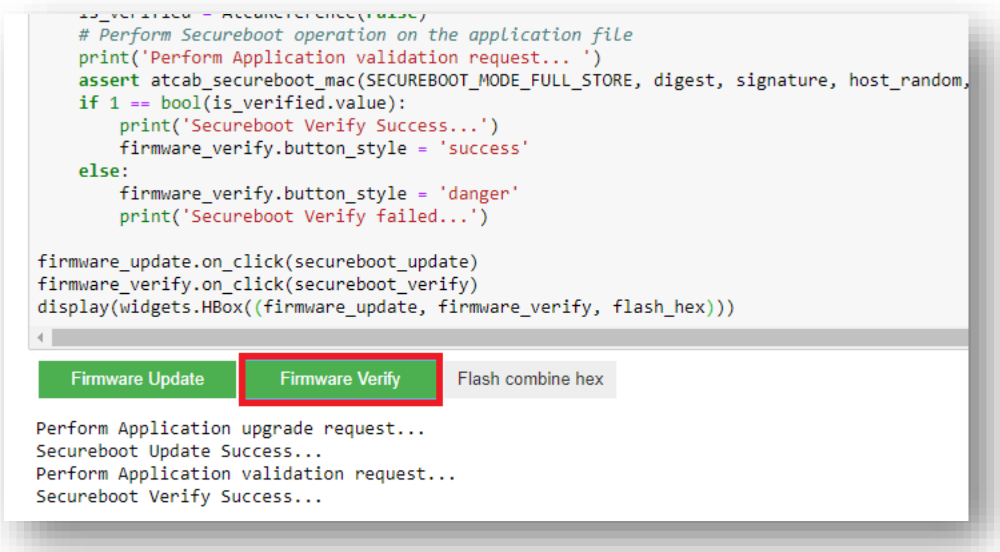
Before verifying the firmware's validity, the firmware digest should be verified and stored to secure element. In this step host sends the encrypted firmware digest and signature to device to validate the firmware. Here the firmware is validated by verifying the signature using firmware signer's public key. Upon successful validation, the device stores the digest to Secureboot digest slot i.e. slot7.




Clicking on "**Firmware Update"** will perform the above steps between host (PC) and the TrustFLEX device. Once firmware update is completed successfully, current firmware digest will be stored in the Secureboot digest slot.

**Verifying the firmware image**

This step recalculates the digest from the example bin (secureboot_test_app.bin). The encrypted digest will be sent to TrustFLEX. Upon successful validation, the device returns MAC value corresponding to this verify request.




Clicking on "**Firmware Verify"** will perform the above steps between host (PC) and the TrustFLEX device.

Pressing “Firmware Update” and “Firmware Verify” should turn to green to indicate successful firmware update and verify operations.

**Flash Combine Hex**

This step programs the CryptoAuth Trust Platform with combined hex (combined_image.hex) using MPLABX IPE. To use this option, its required to provide **ipecmd.jar** file location in trustplatform.config file and update “**path_set**” variable to “**True**” in it and which can be found under your Home directory. Below screenshot display json file,

trustplatform_config.json

```
{
    "MPLABX": {
      "path_set": "False",
      "ipe_path": "C:/Program Files (x86)/Microchip/MPLABX/v5.30/mplab_platform/mplab_ipe/ipecmd.jar"
      }
}
```

Clicking on “Flash combine hex” will program CryptoAuth Trust Platform with combined hex.

Firmware Update | Firmware Verify | Flash combine hex

```
Perform Application upgrade request...
Secureboot Update Success...
Perform Application validation request...
Secureboot Verify Success...
Programming combined hex into CryptoAuth Trust Platform board
Invalid interface selected in the project properties, choosing to use default swd
Invalid interface selected in the project properties, choosing to use default swd
*****************************************************
Invalid interface selected in the project properties, choosing to use default swd
Invalid interface selected in the project properties, choosing to use default swd
Currently loaded versions:
Application version..........1.14.464 (0x01.0x0e.0x01d0)
Target voltage detected
Invalid interface selected in the project properties, choosing to use default swd
*****************************************************
Invalid interface selected in the project properties, choosing to use default swd
Invalid interface selected in the project properties, choosing to use default swd
*****************************************************
Invalid interface selected in the project properties, choosing to use default swd
Invalid interface selected in the project properties, choosing to use default swd
Invalid interface selected in the project properties, choosing to use default swd
Configuration memory will not be programmed because no configuration bits settings have been defined in your code.
To program configuration memory, either define the settings in your code or use the Program Configuration Bits button on the co
nfiguration memory window.
Erasing...
The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x67ff
program memory: start address = 0xc000, end address = 0x123ff
program memory: start address = 0x3fc00, end address = 0x3fdff
Programming complete
 Program Report
23-Dec-2019, 18:22:33
Device Type:ATSAMD21E18A
Program Succeeded.
Operation Succeeded

Programmed hex successfully
```

Pressing “Flash combine hex” should turn green to indicate that, it programmed combined hex successfully.

Unless, the trustplatform.config file is not updated this option cannot be used. To program CryptoAuth Trust Platform refer section 4.2.

## 4.2 Running Firmware Validation on Embedded platform

This usecase can also be executed on Embedded platform. Once the resources are generated as described in previous section, Atmel Studio project provided can be used to run the usecase on CryptoAuth Trust Platform.

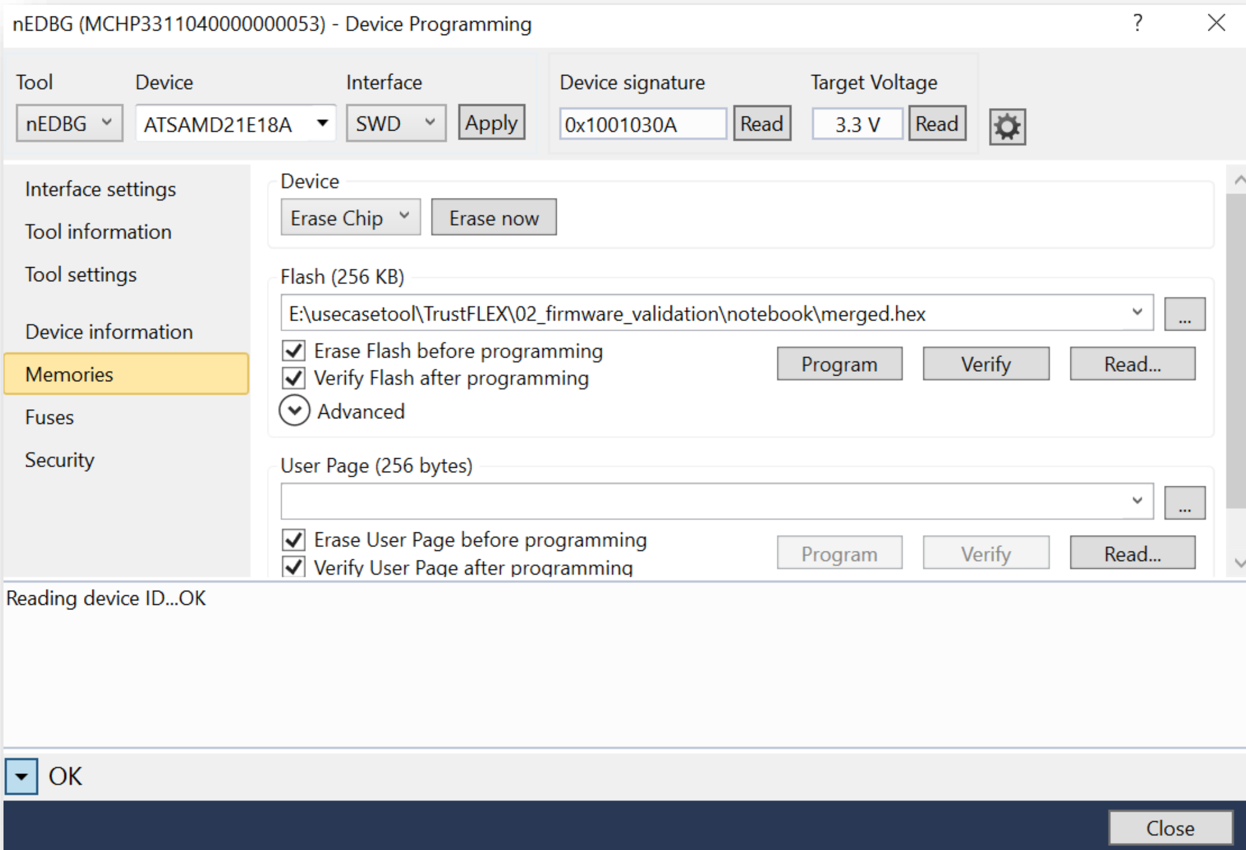
### 4.2.1 Atmel Studio:

All the necessary build steps are done as part of the previous steps. All that needs be done is to program the generated .hex file available at DesignTool\TrustFLEX\02_firmware_validation\notebook \combined_image.hex

The combined_image.hex contains firmware_validation, application images and the signature.

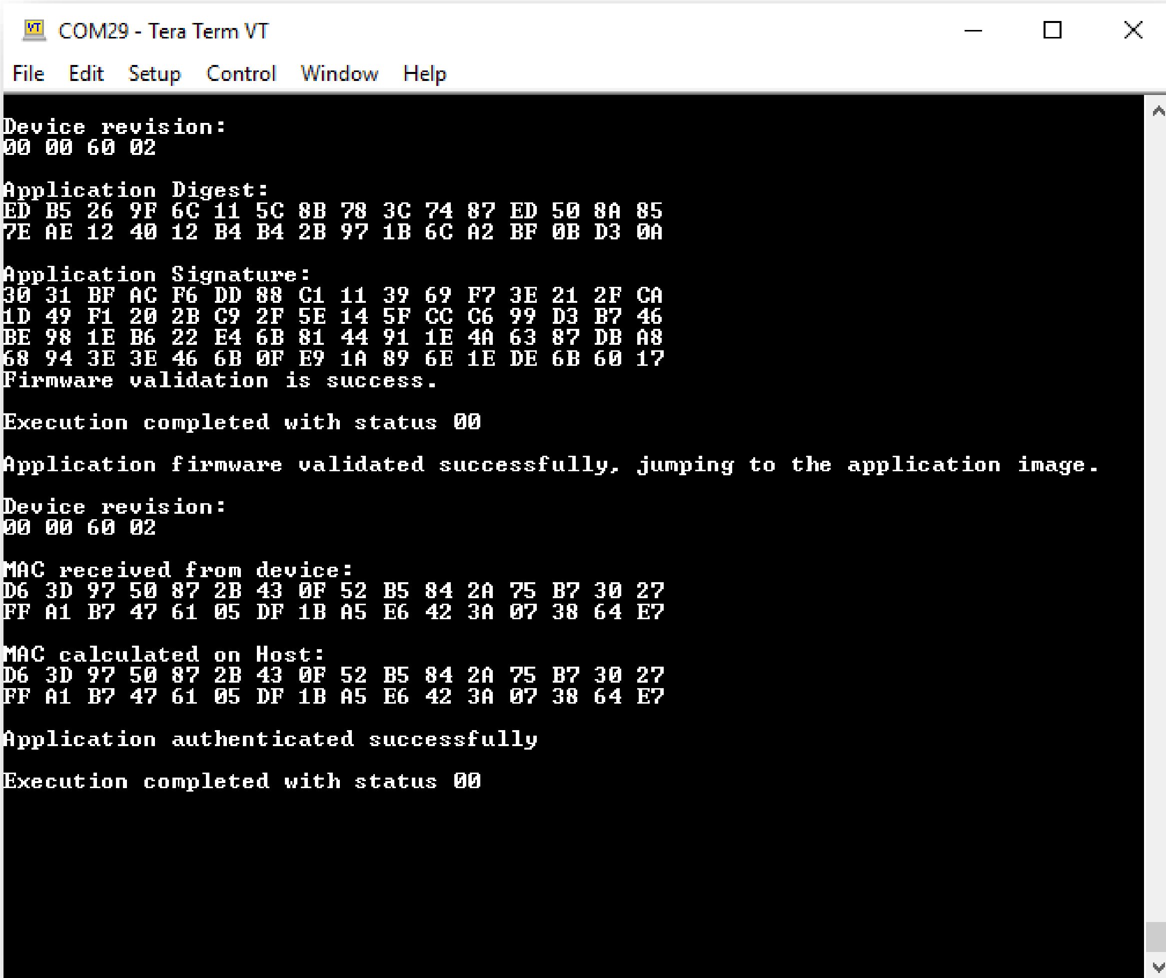
To program using Atmel Studio:

1. Navigate to AtmelStudio -> Tools -> Device Programming
2. Select Tool as nEDBG and Apply
3. Go to Memories and navigate to above path under Flash dropdown
4. Select combined_image.hex file
5. Check both Erase Flash and Verify Flash
6. Click on Program




The program output can be viewed using a serial terminal. Terminal needs to be opened with 115200-8-N-1 settings.

Output on the serial terminal would look like the image below,




On any error, LED blinks five times every second.

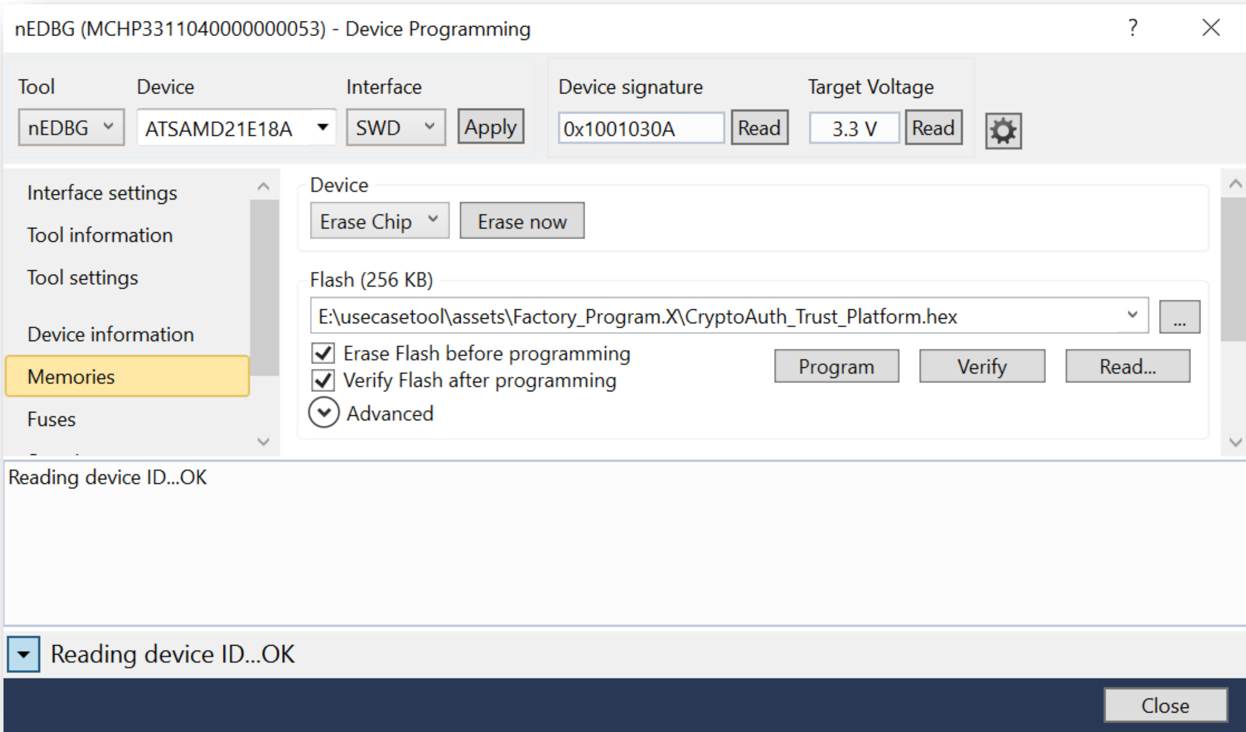
## 4.3 CryptoAuth Trust Platform Development Kit Factory reset

Once any of the embedded project is loaded to CryptoAuth Trust Platform Development Kit, the default program that enables interaction with TrustPlatform tools will be erased.

Before using the Platform with any other notebook or tools on PC, its required to reprogram the default .hex file. Default hex file is available at **assets\Factory_Program.X\CryptoAuth_Trust_Platform.hex**

To reprogram using Atmel Studio:

1. Navigate to AtmelStudio -> Tools -> Device Programming
2. Select Tool as nEDBG and Apply
3. Go to Memories and navigate to above path under Flash dropdown
4. Check both Erase Flash and Verify Flash
5. Click on Program




To reprogram using MPLAB:

1. Open **assets\Factory_Program.X** project in MPLAB IDE
2. Program the Crypto Trust platform by navigating to **CryptoAuth_Trust_Platform_Factory_Program -> Make and Program Device**

Now, CryptoAuth Trust Platform Development Kit contains factory programmed application that enables interactions with Notebooks and/or PC tools.

# 5 FAQ

1. **What are the reasons for "AssertionError: Can't connect to the USB dongle" error?**
   There are many possibilities like,
   1. Crypto Trust Platform is having different application than factory reset firmware. Refer to "CryptoAuth TrustPlatform Factory reset" section any usecase TrustFLEX Guide for reloading it
   2. Check the switch positions on Crypto Trust Platform and/or ATECC608A Trust board
      a. Correct Trust device should be connected and only one device of that type is allowed on the I2C bus. Multiple devices with same address results in error
   3. Check USB connections to Crypto Trust Platform

2. **How to reload factory default application to Crypto Trust Platform?**
   Refer to "CryptoAuth TrustPlatform Factory reset" section any usecase TrustFLEX Guide for reloading it.

3. **Why does my C projects generates No such file or directory with ../../../ TFLXTLS_resource_generation/?**
   C project generates this error when the resources are not generated prior to using embedded projects. Running the resource generation notebook ensures these files and secrets are generated.

4. **Before running any use case notebook and/or C project, why is it mandate to execute resource generation?**
   When resource generation notebook is executed, it generates and programs the required resources like secrets, keys and certificates. These are only prototyping keys and cannot be used for production. These keys will be used part of Usecase notebooks and C projects

5. **How to know the resources being used in a use case?**
   Refer to individual Usecase description html for details on transaction diagrams, resources being used and other details. The resources required for given use case is mentioned in INFER CRYPTOGRAPHIC ASSETS section.

6. **When should I select Custom certificates while doing resource generation?**
   Custom certificates are required when user wants to have their own root, signer instead of MCHP provided. The difference would be organization name, common name and validity are configurable

7. **How to know whether C project is executing on Trust Platform or not after programming?**
   Once the programming is done, the firmware will do use case operation. Depending on the use case operation's output, the Crypto Trust Platform board's status LED will blink at different rates.

It is also possible to view the Console messages by using applications like TeraTerm. Open the application with the COM related to Crypto Trust Platform with 115200-8-N-1 settings

**8. Why is firmware validation project fails with error “Firmware validation is failed! with status 01”?**

There are many possibilities like,

a. The resources on TrustFLEX device and on the host (PC) could be different. Rerun “Resource Generation Notebook” section for reloading it.

b. Firmware digest is not matched. Make sure that firmware Update step is exec uted using Notebook prior to running C project

## The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support.

Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

## Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

## Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE,

ISBN:

## Quality Management System Certified by DNV

**ISO/TS 16949**

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

# Worldwide Sales and Service

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Austin, TX**
Tel: 512-257-3370

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

**Raleigh, NC**
Tel: 919-844-7510

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110
Tel: 408-436-4270

**Canada - Toronto**
Tel: 905-695-1980
Fax: 905-695-2078

## ASIA/PACIFIC

**Australia - Sydney**
Tel: 61-2-9868-6733

**China - Beijing**
Tel: 86-10-8569-7000

**China - Chengdu**
Tel: 86-28-8665-5511

**China - Chongqing**
Tel: 86-23-8980-9588

**China - Dongguan**
Tel: 86-769-8702-9880

**China - Guangzhou**
Tel: 86-20-8755-8029

**China - Hangzhou**
Tel: 86-571-8792-8115

**China - Hong Kong SAR**
Tel: 852-2943-5100

**China - Nanjing**
Tel: 86-25-8473-2460

**China - Qingdao**
Tel: 86-532-8502-7355

**China - Shanghai**
Tel: 86-21-3326-8000

**China - Shenyang**
Tel: 86-24-2334-2829

**China - Shenzhen**
Tel: 86-755-8864-2200

**China - Suzhou**
Tel: 86-186-6233-1526

**China - Wuhan**
Tel: 86-27-5980-5300

**China - Xian**
Tel: 86-29-8833-7252

**China - Xiamen**
Tel: 86-592-2388138

**China - Zhuhai**
Tel: 86-756-3210040

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-3090-4444

**India - New Delhi**
Tel: 91-11-4160-8631

**India - Pune**
Tel: 91-20-4121-0141

**Japan - Osaka**
Tel: 81-6-6152-7160

**Japan - Tokyo**
Tel: 81-3-6880- 3770

**Korea - Daegu**
Tel: 82-53-744-4301

**Korea - Seoul**
Tel: 82-2-554-7200

**Malaysia - Kuala Lumpur**
Tel: 60-3-7651-7906

**Malaysia - Penang**
Tel: 60-4-227-8870

**Philippines - Manila**
Tel: 63-2-634-9065

**Singapore**
Tel: 65-6334-8870

**Taiwan - Hsin Chu**
Tel: 886-3-577-8366

**Taiwan - Kaohsiung**
Tel: 886-7-213-7830

**Taiwan - Taipei**
Tel: 886-2-2508-8600

**Thailand - Bangkok**
Tel: 66-2-694-1351

**Vietnam - Ho Chi Minh**
Tel: 84-28-5448-2100

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**Finland - Espoo**
Tel: 358-9-4520-820

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**France - Saint Cloud**
Tel: 33-1-30-60-70-00

**Germany - Garching**
Tel: 49-8931-9700

**Germany - Haan**
Tel: 49-2129-3766400

**Germany - Heilbronn**
Tel: 49-7131-67-3636

**Germany - Karlsruhe**
Tel: 49-721-625370

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Germany - Rosenheim**
Tel: 49-8031-354-560

**Israel - Ra'anana**
Tel: 972-9-744-7705

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Italy - Padova**
Tel: 39-049-7625286

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Norway - Trondheim**
Tel: 47-7289-7561

**Poland - Warsaw**
Tel: 48-22-3325737

**Romania - Bucharest**
Tel: 40-21-407-87-50

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**Sweden - Gothenberg**
Tel: 46-31-704-60-40

**Sweden - Stockholm**
Tel: 46-8-5090-4654

**UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820