Carlsbad Spi image gen user manual

Rev 0.1	03/04/2020

MICROCHIP CONFIDENTIAL

The information contained herein is confidential and proprietary to Microchip, shall be used solely in accordance with the agreement pursuant to which it is provided, and shall not be reproduced or disclosed to others without the prior written consent of Microchip. Although the information is believed to be accurate, no responsibility is assumed for inaccuracies. Microchip reserves the right to make changes to this document and to specifications and product descriptions at any time without notice. Neither the provision of this information nor the sale of the described semiconductor devices conveys any licenses under any patent rights or other intellectual property rights of Microchip or others. The product may contain design defects or errors known as anomalies, including but not necessarily limited to any which may be identified in this document, which may cause the product to deviate from published specifications. Microchip products are not designed, intended, authorized or warranted for use in any life support or other application where product failure could cause or contribute to personal injury or severe property damage. Any and all such uses without prior written approval of an officer of Microchip will be fully at the risk of the customer.

MICROCHIP DISCLAIMS AND EXCLUDES ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND AGAINST INFRINGEMENT AND THE LIKE, AND ANY AND ALL WARRANTIES ARISING FROM ANY COURSE OF DEALING OR USAGE OF TRADE. IN NO EVENT SHALL SMSC BE LIABLE FOR ANY DIRECT, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES; OR FOR LOST DATA, PROFITS, SAVINGS OR REVENUES OF ANY KIND; REGARDLESS OF THE FORM OF ACTION, WHETHER BASED ON CONTRACT; TORT; NEGLIGENCE OF SMSC OR OTHERS; STRICT LIABILITY; BREACH OF WARRANTY; OR OTHERWISE; WHETHER OR NOT ANY REMEDY OF BUYER IS HELD TO HAVE FAILED OF ITS ESSENTIAL PURPOSE, AND WHETHER OR NOT SMSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TABLE OF CONTENTS

1 INTRODUCTION 3

1.1 PURPOSE 3
1.2 SETUP REQUIREMENTS 3
1.2.1 Desktop PC 3
1.3 REFERENCES 3

2 BLOCK DIAGRAM OF SPI FLASH MEMORY LAYOUT 4

3 CARLSBAD SPI IMAGE HEADER FORMAT 5

4 CARLSBAD SPI IMAGE GENERATION 7

5 KEY GENERATION USING OPENSSL 11

6 REVISION HISTORY 13

1 Introduction

1.1 Purpose

This document is to provide the Spi image gen User manual.

1.2 Setup Requirements

1.2.1 Desktop PC

Desktop PC running x64 based Windows operating system

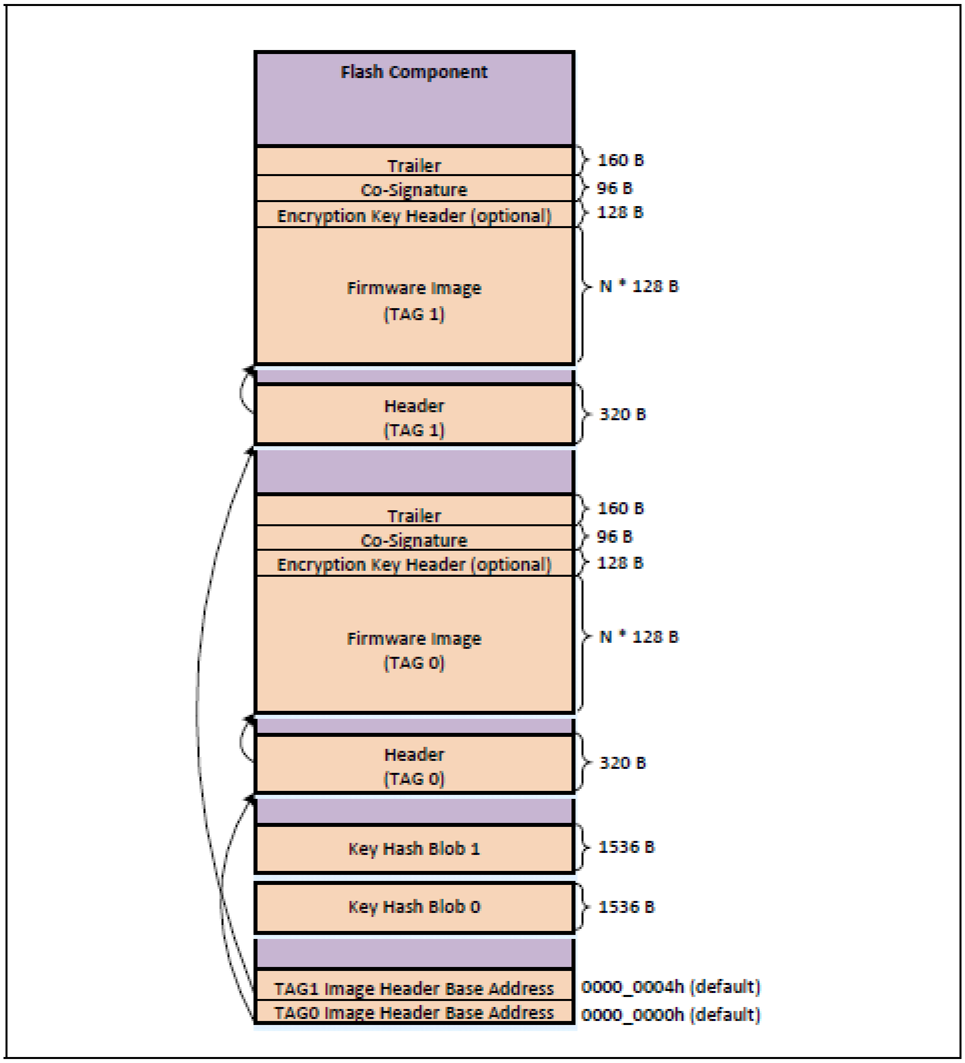
1.3 References

Refer the Datasheets of the Carlsbad to get the EC Firmware map ,EC configuration information is to provided in the spi_cfg.txt to generate the spi_image

Refer the release.txt and spi_cfg.txt for the SPI image generation for the Carlsbad

Spi_cfg.txt = configuration for the EC image to be available in the spi image

2 Block diagram of SPI flash memory layout



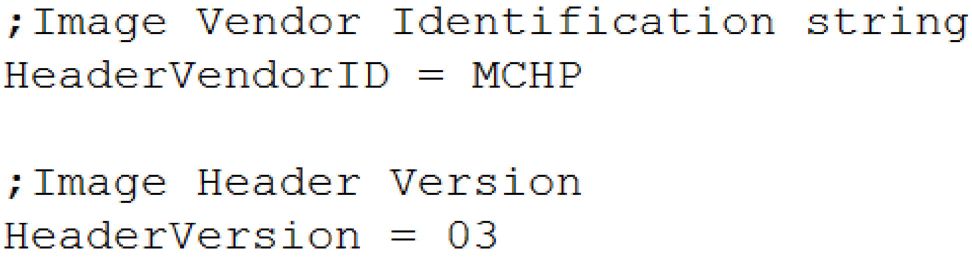
EC configuration :
FW image stores based on the Imagelocation provided in the spi_Cfg.txt

Tag0 is the address of the EC image 0 into the spi image
Tag1 is the address of the EC image 1 into the spi image

3 Carlsbad SPI image header format

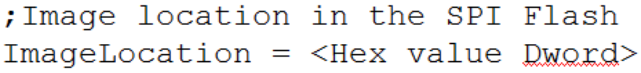
EC FW header provided in the spi_cfg.txt

1) EC FW header string as shown below in the spi_cfg.txt



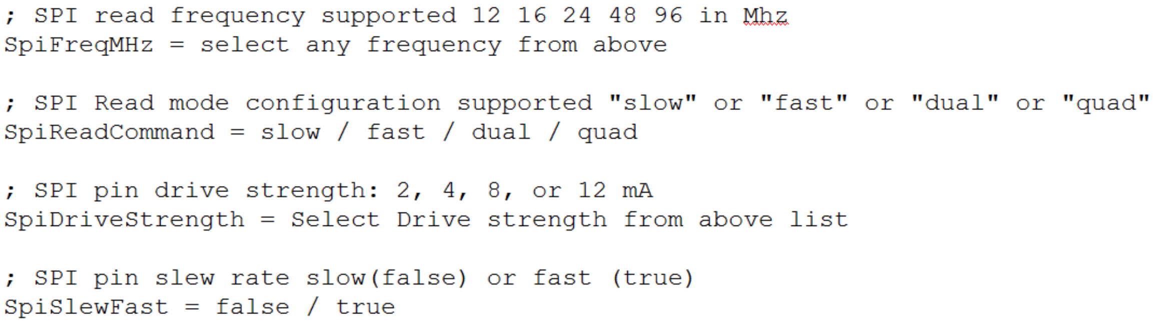
```
;Image Vendor Identification string
HeaderVendorID = MCHP

;Image Header Version
HeaderVersion = 03
```

2) EC fw binary to be copied into the SPI image using the below fields



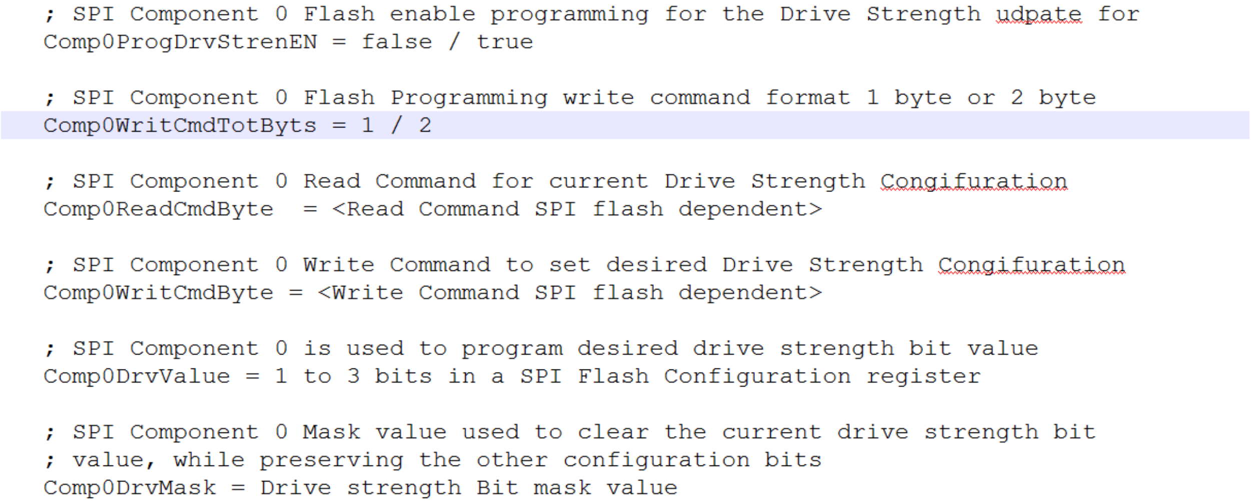
```
;Image location in the SPI Flash
ImageLocation = <Hex value Dword>
```

3) SPI related fields available in the EC FW header



```
; SPI read frequency supported 12 16 24 48 96 in Mhz
SpiFreqMHz = select any frequency from above

; SPI Read mode configuration supported "slow" or "fast" or "dual" or "quad"
SpiReadCommand = slow / fast / dual / quad

; SPI pin drive strength: 2, 4, 8, or 12 mA
SpiDriveStrength = Select Drive strength from above list

; SPI pin slew rate slow(false) or fast (true)
SpiSlewFast = false / true
```

4) SPI components fields



```
; SPI Component 0 Flash enable programming for the Drive Strength udpate for
Comp0ProgDrvStrenEN = false / true

; SPI Component 0 Flash Programming write command format 1 byte or 2 byte
Comp0WritCmdTotByts = 1 / 2

; SPI Component 0 Read Command for current Drive Strength Congifuration
Comp0ReadCmdByte  = <Read Command SPI flash dependent>

; SPI Component 0 Write Command to set desired Drive Strength Congifuration
Comp0WritCmdByte = <Write Command SPI flash dependent>

; SPI Component 0 is used to program desired drive strength bit value
Comp0DrvValue = 1 to 3 bits in a SPI Flash Configuration register

; SPI Component 0 Mask value used to clear the current drive strength bit
; value, while preserving the other configuration bits
Comp0DrvMask = Drive strength Bit mask value
```

```
; SPI Component 1 Flash enable programming for the Drive Strength udpate for
Comp1ProgDrvStrenEN = false / true

; SPI Component 0 Flash Programming write command format 1 byte or 2 byte
Comp1WritCmdTotByts = 1 / 2

; SPI Component 0 Read Command for current Drive Strength Congifuration
Comp1ReadCmdByte  = <Read Command SPI flash dependent>

; SPI Component 0 Write Command to set desired Drive Strength Congifuration
Comp1WritCmdByte = <Write Command SPI flash dependent>

; SPI Component 0 is used to program desired drive strength bit value
Comp1DrvValue = 1 to 3 bits in a SPI Flash Configuration register

; SPI Component 0 Mask value used to clear the current drive strength bit
; value, while preserving the other configuration bits
Comp1DrvMask = Drive strength Bit mask value
```

5) Use the Authentication to be set True/False to keep the EC fw image for authentication
Authentciation key to be set to True/False
Autentication key to be selected
Rollback protection key to be selected are provide in the spi_cfg.txt sample and release.txt

```
; Enable Authentication of Header and Firmware FW
; Generate ECDSA signature of 64-byte Header, padded FW binary + optional
; encryption key header. If false bytes[31:0] of the signatures contain the
; SHA256(object)
UseECDSA = false / true
```

6) Key to be provided in this below fields

```
; This EC key pair is used to sign and verify/authenticate the FW Image Header,
; FW + optional key header optional key header.
; EC Private Key in PEM encoded Openssl SSLeay encrypted format
; This key is used to sign the Header and is NOT stored in the MEC chip.
ECDSAPrivKeyFile = Authentication Key.pem
ECDSAPrivKeyPassword = PASSWORD for the Private Key
```

7) FW binary file to be provided in the below fiels in the spi_cfg.txt

```
;Firmware Application binary image
FwBinFile = <Application Binary file . bin>
```

8) MCHP cosignature to be use with the help of spi_cfg.txt in the below fields

```
;Zero means get entry address from offset 0x4 of input Application binary which is the
;If FWEntryAddress is non-zero then use it as entry point.
FwEntryAddress = 0 or <Entry address in Hex if known>
;MCHP Dual signature enable
UseMCHPECDSA = false / true

;If Dual signature is enabled provide the key and password for the keys
MCHPECDSAPrivKeyFile = Signature Private Key.pem
MCHPECDSAPrivKeyPassword = PASSWORD for the Private Key
```

4 Carlsbad SPI image generation

The below instructions is to provide to the spi_cfg.txt which is based on the Datasheet.

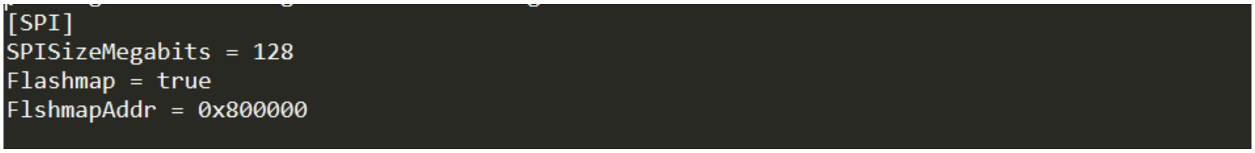
Refer the release.txt in this directory which has information to be provided to the spi_cfg.txt to generate the spi image

Sample configuration is provided in the spi_cfg.txt to generate the SPI image.

Release.txt has the information of the configuration to generate the spi image to be provided in the spi_cfg.txt

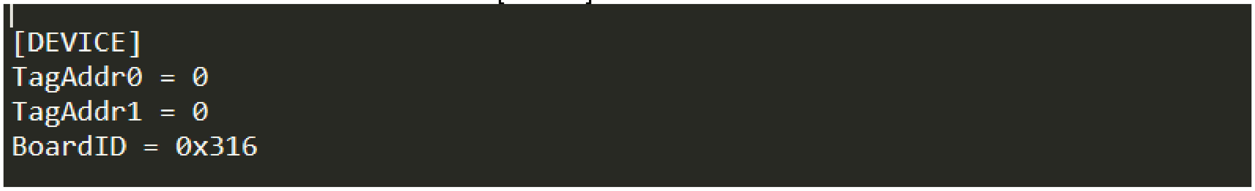
Quick run of carlsbad_spi_gen.exe :
Follow the instruction to run the carlsbad_spi_gen.exe

1) Open the spi_cfg.txt as default file
2) Provide the size of the spi image , it has default with 128 (16MB)
   For ex : SPI image want to be in 32 MB provide the "SPISizeMegabits =256"
   SPI image want to be in 16 MB provide the "SPISizeMegabits =128"



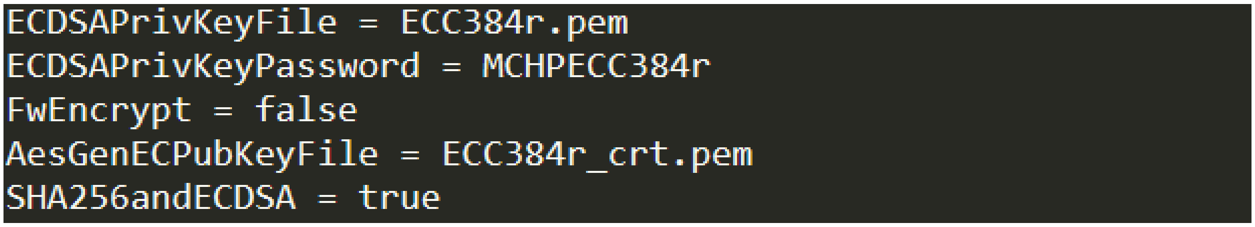
```
[SPI]
SPISizeMegabits = 128
Flashmap = true
FlshmapAddr = 0x800000
```

3) Provide the TaAddr0/1 under the section [Device]



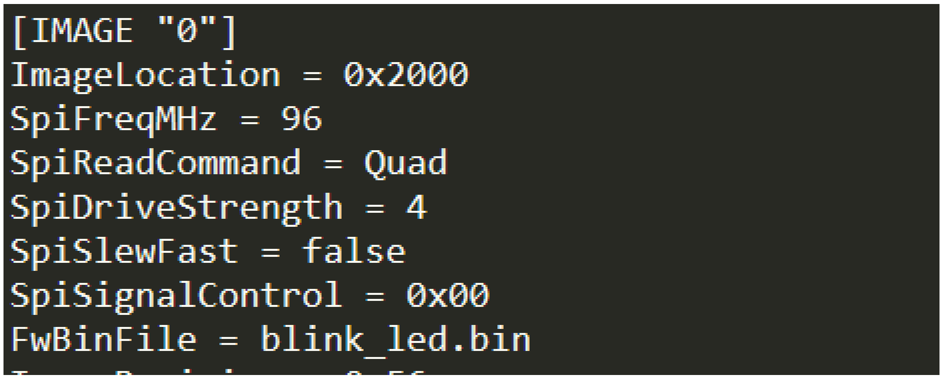
```
[DEVICE]
TagAddr0 = 0
TagAddr1 = 0
BoardID = 0x316
```

4) Provide the path of the ECDSA384 key with the password option to be provided in the spi_cfg.txt



```
ECDSAPrivKeyFile = ECC384r.pem
ECDSAPrivKeyPassword = MCHPECC384r
FwEncrypt = false
AesGenECPubKeyFile = ECC384r_crt.pem
SHA256andECDSA = true
```

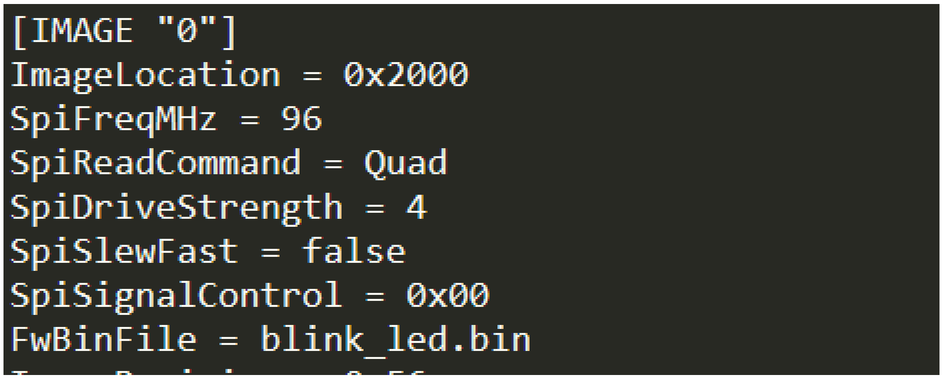
5) Provide the path of the EC FW image in the spi_cfg.txt as shown below
FWimage should be generated using the srec_cat.exe tool to convert the hex into bin file using the below command as
Command as below:
srec_cat.exe <IDE generated HEX file> -intel -offset -0xE0000 -o bin_file.bin -binary



```
[IMAGE "0"]
ImageLocation = 0x2000
SpiFreqMHz = 96
SpiReadCommand = Quad
SpiDriveStrength = 4
SpiSlewFast = false
SpiSignalControl = 0x00
FwBinFile = blink_led.bin
```

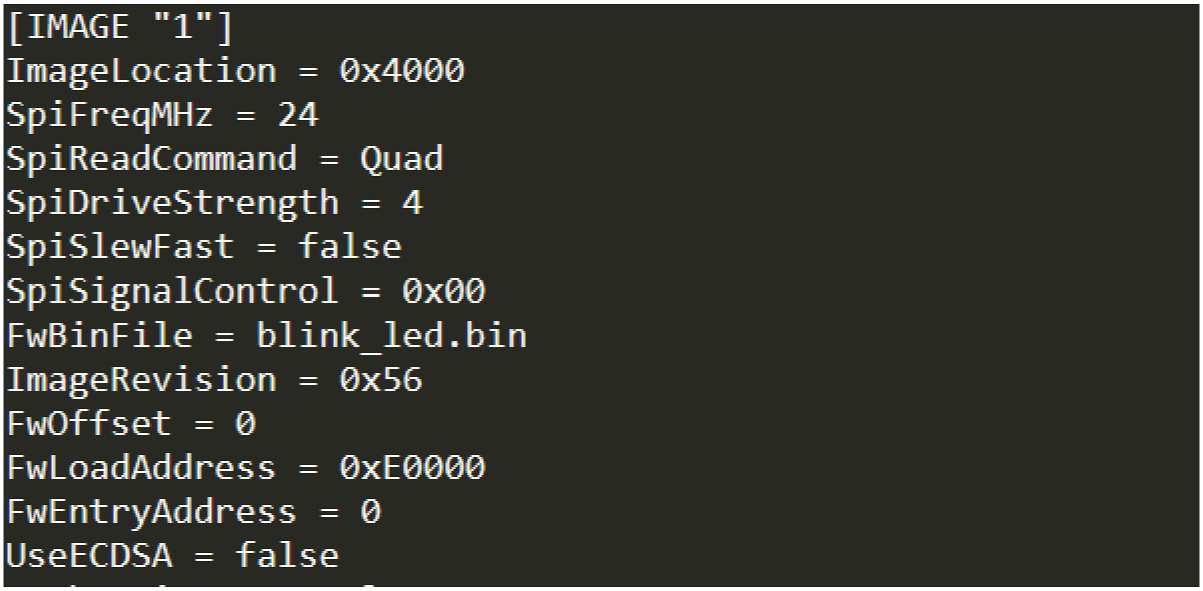
6) If there are two EC image
Repeat the [IMAGE “0”] as [IMAGE “1”] section has two times to be provided in the spi_cfg.txt

[IMAGE “0”] as shown below



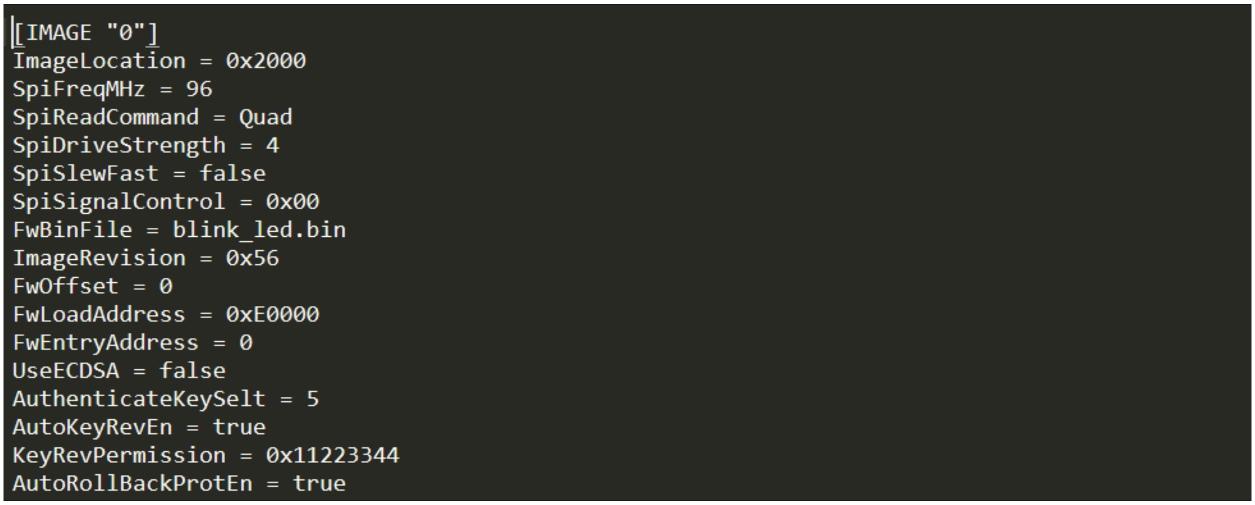
```
[IMAGE "0"]
ImageLocation = 0x2000
SpiFreqMHz = 96
SpiReadCommand = Quad
SpiDriveStrength = 4
SpiSlewFast = false
SpiSignalControl = 0x00
FwBinFile = blink_led.bin
```

[IMAGE “1”] as shown below

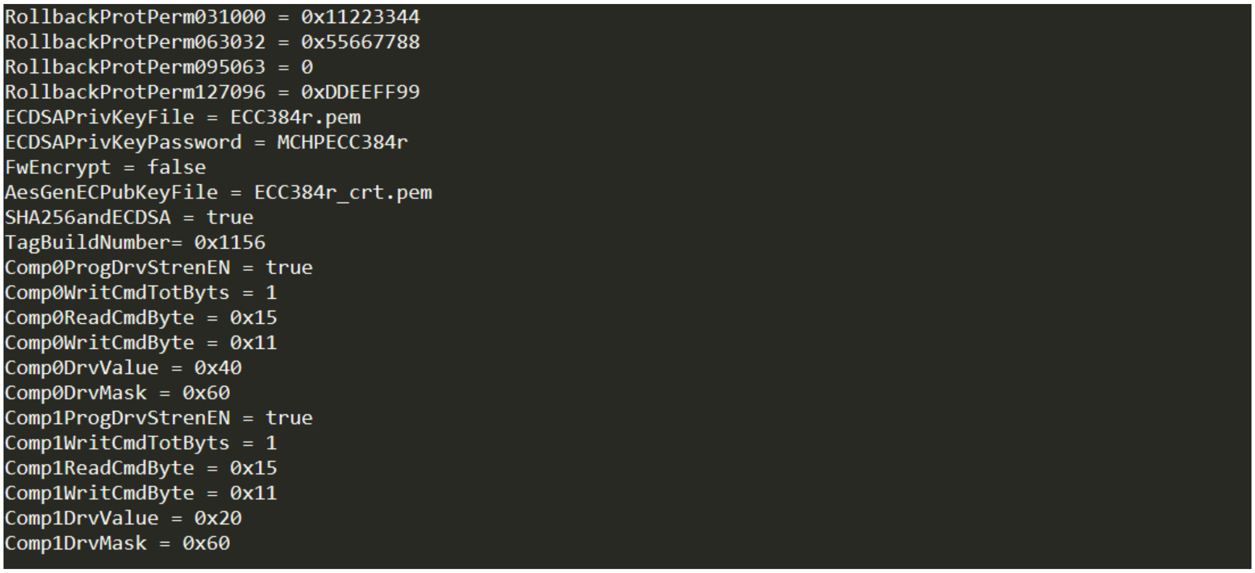


```
[IMAGE "1"]
ImageLocation = 0x4000
SpiFreqMHz = 24
SpiReadCommand = Quad
SpiDriveStrength = 4
SpiSlewFast = false
SpiSignalControl = 0x00
FwBinFile = blink_led.bin
ImageRevision = 0x56
FwOffset = 0
FwLoadAddress = 0xE0000
FwEntryAddress = 0
UseECDSA = false
```

7) Tag0 EC image field which has TagAddr0 in the section [Device] ,Provide the ImageLocation, EC firmware image file name under the fields “FwBinFile” and and ECDSA test key specified for TAG0 and TAG1 , select the authentication/Encryption is to be enable or not by providing the true/false



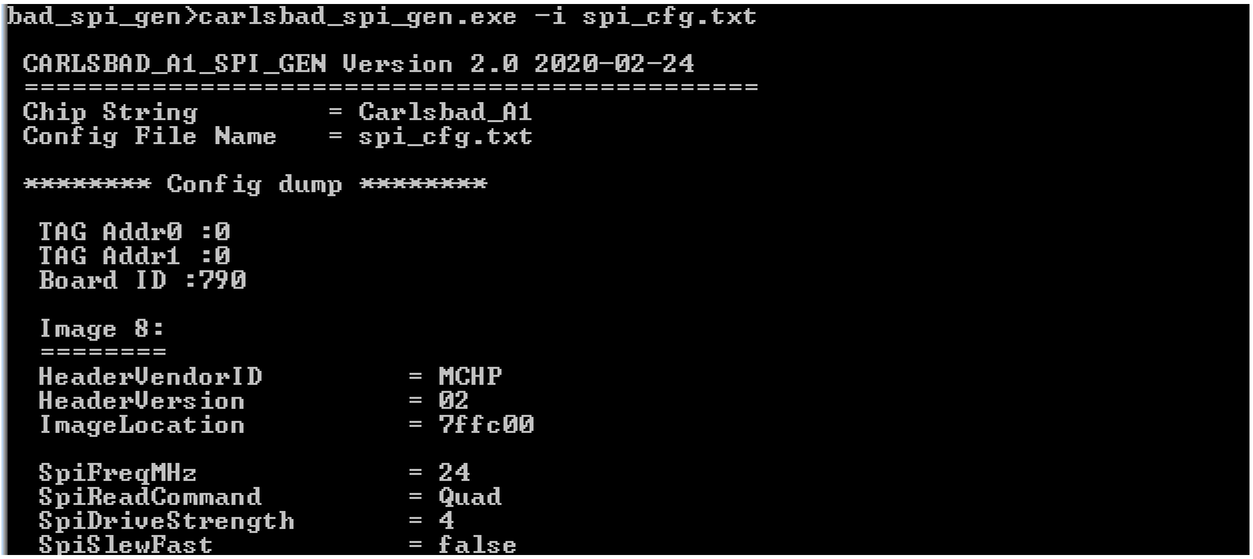
```
[IMAGE "0"]
ImageLocation = 0x2000
SpiFreqMHz = 96
SpiReadCommand = Quad
SpiDriveStrength = 4
SpiSlewFast = false
SpiSignalControl = 0x00
FwBinFile = blink_led.bin
ImageRevision = 0x56
FwOffset = 0
FwLoadAddress = 0xE0000
FwEntryAddress = 0
UseECDSA = false
AuthenticateKeySelt = 5
AutoKeyRevEn = true
KeyRevPermission = 0x11223344
AutoRollBackProtEn = true
```



```
RollbackProtPerm031000 = 0x11223344
RollbackProtPerm063032 = 0x55667788
RollbackProtPerm095063 = 0
RollbackProtPerm127096 = 0xDDEEFF99
ECDSAPrivKeyFile = ECC384r.pem
ECDSAPrivKeyPassword = MCHPECC384r
FwEncrypt = false
AesGenECPubKeyFile = ECC384r_crt.pem
SHA256andECDSA = true
TagBuildNumber= 0x1156
Comp0ProgDrvStrenEN = true
Comp0WritCmdTotByts = 1
Comp0ReadCmdByte = 0x15
Comp0WritCmdByte = 0x11
Comp0DrvValue = 0x40
Comp0DrvMask = 0x60
Comp1ProgDrvStrenEN = true
Comp1WritCmdTotByts = 1
Comp1ReadCmdByte = 0x15
Comp1WritCmdByte = 0x11
Comp1DrvValue = 0x20
Comp1DrvMask = 0x60
```

8) Run the below command by providing the hex has the input to generate the FW binary file
Command as below:
srec_cat.exe <IDE generated HEX file> -intel -offset -0xE0000 -o bin_file.bin -binary

9) Using the srec_cat.exe to generate the FWbin file.
And call the carlsbad_spi_gen generator tool to generate the spi image
Script to generate the spi image using carlsbad_spi_gen.exe
Command as below :
carlsbad_spi_gen.exe -i spi_cfg.txt



```
bad_spi_gen>carlsbad_spi_gen.exe -i spi_cfg.txt

 CARLSBAD_A1_SPI_GEN Version 2.0 2020-02-24
 ==============================================
 Chip String        = Carlsbad_A1
 Config File Name   = spi_cfg.txt

 ******** Config dump ********

  TAG Addr0 :0
  TAG Addr1 :0
  Board ID :790

  Image 8:
  ========
  HeaderVendorID         = MCHP
  HeaderVersion          = 02
  ImageLocation          = 7ffc00

  SpiFreqMHz             = 24
  SpiReadCommand         = Quad
  SpiDriveStrength       = 4
  SpiSlewFast            = false
```

5 Key Generation using openssl

ECDSA P384 Curve key generation using openssl command :

1) EC private key of P384:

```
openssl ecparam -name secp384r1 -genkey -noout -out key.pem
```

2) EC private key of P384:

```
openssl ecparam -name secp384r1 -genkey -noout -out key.pem
```

3) print private key and public key:

```
  openssl ec -in key.pem -nout -text
```

4) **Create signature**:

```
openssl dgst -sha384 -sign key.pem input.bin > signature.bin
```

5) How to generate the public key from the private key:

```
openssl ec -in key.pem -pubout -out pubkey.pem
```

6) Verify signature:

```
openssl dgst -sha384 -verify pubkey.pem -signature signature.bin input.bin
```

7) Create the key with password:

```
set ecdsa_key_filename=ec384pem
set ecdsa_key_filename_pass=ec384
set csr_file=%ecdsa_key_filename_pass%_csr.pem
set crt_file=%ecdsa_key_filename_pass%_crt.pem


openssl.exe ecparam -name secp384r1 -genkey | openssl.exe ec -out %ecdsa_key_filename%
passout pass:%ecdsa_key_filename_pass% -aes-256-cbc

openssl.exe req -new -key %ecdsa_key_filename% -out %csr_file% -passin
pass:%ecdsa_key_filename_pass% -subj /C=US/ST=NYC/L=Hauppauge/O=MCHP/OU=CPG-
FW/CN=CEC1712
```

8)

```
openssl.exe x509 -req -days 3650 -in %csr_file% -signkey %ecdsa_key_filename% -out
%crt_file% -passin pass:%ecdsa_key_filename_pass%
```

RSA Key generation for the PKCS and PSS for 2K/3K/4K key generation

1) Create a RSA private key of 2k or 3k or 4k using below command

For ex :

RSA 3K

```
openssl genrsa -out key.pem 3072
```

RSA 2K

```
openssl genrsa -out key.pem 2048
```

RSA 4K

```
openssl genrsa -out key.pem 4096
```

2) Signing with PKCS1.5 format

```
openssl dgst -sha256 -sign key.pem -out  sign.bin input.bin
```

3) **To verify the PKCS signed data** is perfect with the openssl command , create a sign.bin from the secure boot spi image generator tool binary , take the signed data has been copied in the spi image in the specified location and verify with the data vs openssl will be same.

4) Signing using PSS method

   ```
   openssl dgst -sha256 -sigopt rsa_padding_mode:pss -sigopt rsa_pss_saltlen:-1 -sign key.pem -out sign.bin  input.bin
   ```

5) **To verify the PSS for the secure boot spi image generated binary** , take the signed part and verify here
   Create a sign.bin from the spi image generator binary where the sign data has been copied to specified location and verify using that data vs openssl to verify it.

   ```
   openssl dgst -sha256 -sigopt rsa_padding_mode:pss -sigopt rsa_pss_saltlen:-1 -verify key.pem -signature sign.bin input.bin
   ```

6 Revision History

Name	Revision Level	Date	Section	Remarks
Prabhakar V	0.1	April 2nd 2020	Document	Initial draft