

Table of Contents

Purpose
Device Certificate Application
SmartFusion2 Device Certificate
References
SmartFusion2 Device Certificate Service
Using Device Certificate Service
Design Requirements
Design Description
Hardware Implementation
Software Implementation
Setting Up the Design
Running the Design
Viewing Fields in Device Certificate
Conclusion
Appendix A: Design and Programming Files
Appendix B: Decoding Device Certificate Using ASN.1 JavaScript Decoder Open Source Tool 18
List of Changes

Purpose

This application note describes how to read and export device certificate using system services and analyze the content of the device certificate in SmartFusion[®]2 system-on-chip (SoC) field programmable gate array (FPGA) devices.

Introduction

The device certificate includes a digital signature, an electronic analogue of a written signature. The digital signature assures that the claimed signatory signed the information. In addition, a digital signature detects whether or not the information was modified after it was signed.

Microsemi SmartFusion2 device certificate is an X.509 complaint digital certificate that is digitally signed with a Microsemi private key. The X.509 is an ITU-T standard for a public key infrastructure (PKI) and privilege management infrastructure (PMI). It specifies standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.

The device certificate in the SmartFusion2 devices cryptographically binds the device serial number, date code, its model or part number, the device's secret factory key, and a digital signature from Microsemi that is validated internally by the device and externally by the user.

The digital certificate is stored in the device's embedded non-volatile memory (eNVM). The bigger devices (M2S150 and M2S090) support elliptic curve cryptography (ECC). Therefore, the factory ECC public keys are also certified and included in the device certificates.



Device Certificate Application

The primary advantage of the device certificate application is to prevent counterfeiting and fraud. Counterfeiting in electronic parts can take various forms, such as:

- Cloning designs at the transistor level
- · Black-topping and re-marking devices to misrepresent used devices as new
- Changing the date codes
- Improving the speed grade or the temperature grade, and increasing the alleged screening level

As a result, any mismatch between how the device is represented by its shipping paperwork or the label printed on its surface and the digital certificate indicates the possibility of counterfeiting fraud.

One application for a SmartFusion2 device certificate is that if a counterfeiter remarks a device with a faster speed grade, the model number authenticated in the device certificate still reflects the true speed grade. When the user attempts to program such a device with a design that was compiled for the faster speed grade device, the programmer observes that the speed grade reflected in the certificate is incorrect for the design.

SmartFusion2 Device Certificate

The SmartFusion2 device certificate is encoded in the abstract syntax notation one format: ASN.1. It is a standard and notation that describes rules and structures for representing, encoding, transmitting, and decoding data in telecommunications and computer networking. The formal rules enable representation of objects that are independent of machine-specific encoding techniques. Formal notation makes it possible to automate the task of validating whether a specific instance of data representation abides by the specifications.

Field Name	Description		
Version	Contains the version information		
Serial Number	Contains the serial number information		
Signature Algorithm	Provides information about the algorithm that is used to generate the signature		
Issuer	Provides information about certificate issuers information like: Country Name, Organization Unit Name, Organization Name, and Common Name information		
Validity	 Provides information about validity of the certificate Not Before (start time specified for the certificate validity) Not After (end time specified for the certificate validity) <i>Note: The certificate is only valid between these specified time fields.</i> 		
Subject	Provides information about generation qualifier, surname, and given name		
Subject Public Key Info	 Provides the information about the public key generation algorithm and public key information Public Key Algorithm Subject Public Key 		
Issuer Unique Identifier	It contains issuer unique identification string of 9 bytes size		
Subject Unique Identifier	It contains 0 x 00 + factory serial number (FSN) + serial number modifier (SNM). For more information about FSN and SNM descriptions, refer to the UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide.		
Extensions	Reserved		

Table 1 • SmartFusion2 Device Certificate Fields and Descriptions



Field Name	Description		
Certificate Signature Algorithm	Provides information about the algorithm that is being used.		
Certificate Signature	Provides the certificate signature information. The signature of the SmartFusion2 device certificate can be verified using Microsemi public key.		

References

The following list of references is used in this document:

- UG0331: SmartFusion2 Microcontroller Subsystem User Guide
- UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide
- UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide

SmartFusion2 Device Certificate Service

SmartFusion2 device certificate service is a part of device and design information services of the system services. These system services are performed by the system controller block.

The device certificate service provides access to the system controller's device and design information services. This service is accessed through the communication block (COMM_BLK).

There are two COMM_BLK instances:

- Located in the microcontroller sub system (MSS)
- · Located in the system controller

The COMM_BLK consists of an APB interface, eight byte transmit FIFO, and eight byte receive FIFO. The COMM_BLK provides a bi-directional message passing facility between the MSS and the system controller.

The device certificate service is initiated using the COMM_BLK in the MSS, which can be read or written by any master on the AMBA high performance bus (AHB) matrix; typically either the ARM[®] Cortex[®]-M3 processor or a design in the FPGA fabric (also known as a fabric master).

The system controller receives the command through the COMM_BLK in the system controller. On completion of the requested service, the system controller returns a status message through the COMM_BLK. The responses generated are based on the selected command.



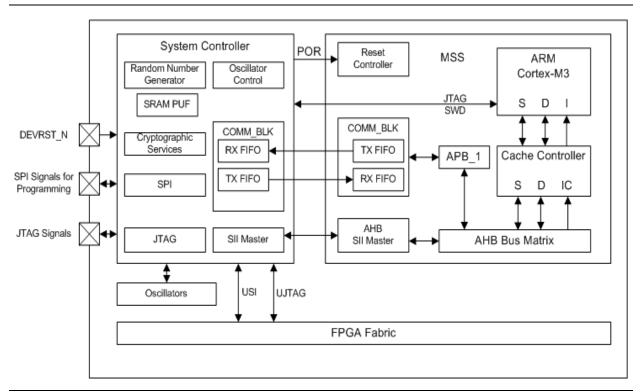


Figure 1 shows the system controller block in SmartFusion2.

Figure 1 • System Controller Interface with MSS and FPGA Fabric



Using Device Certificate Service

The device certificate service is initiated using the COMM_BLK. The COMM_BLK base address resides at 0x40016000 and extends to address 0x40016FFF in the Cortex-M3 processor memory map. Table 2 summarizes the control and status registers for the COMM_BLK.

For more information about COMM_BLK registers description, refer to the "Communication Block" chapter in the UG0331: SmartFusion2 Microcontroller Subsystem User Guide.

Register Name	Address Offset	R/W	Reset Value	Description
CONTROL	0 x 00	R/W	0 x 00	Control Register
STATUS	0 x 04	R/W	0 x 00	Status Register
INT_ENABLE	0 x 08	R/W	0 x 00	Interrupt Enable
DATA8	0 x 10	R/W	0 x 00	Byte Data Register
DATA32	0 x 14	R/W	0 x 00000000	Word Data Register
FRAME_START8	0 x 18	R/W	0 x 00	Frame/Command Byte Register
FRAME_START32	0 x 1c	R/W	0 x 00000000	Frame/Command Word Register

Table 2 • COMM_BLK Register Map

The following are the basic steps to use the device certificate system service in the SmartFusion2 devices:

- Disable the COMM BLOCK loop back mode by writing "1" to LOOPBACK bit (bit-5) of the CONTROL register (0x40016000).
- 2. Enable the COMM block by writing "1" to ENABLE bit (bit-4) of the CONTROL register.
- 3. Enable the receive interrupt by writing "1" to RCVOKAY bit (bit-1) of the INT_ENABLE register (0x40016008).
- 4. Enable the COMM_BLK_INTR (INTISR[19]) in Cortex-M3 Processor interrupts.
- 5. Set up the COMM_BLK register in byte mode by writing "0" to SIZETX bit (bit-2) of the CONTROL register.
- 6. Wait for the TXTOKAY bit (bit-0) of STATUS register(0X40016004) to become 1.
- 7. Send the device certificate command by writing the register FRAME_START8 with the command value. The command value of the device certificate service is 0x00.
- 8. Set up the COMM_BLK in 4 bytes mode by writing "1" to SIZETX bit (bit-2) of the CONTROL register.
- 9. Wait for the TXTOKAY status bit to become 1.
- 10. Send the DEVICECERTPTR address, by writing the register DATA32 with the DEVICECERTPTR value. For more information about the device certificate service request details, refer to Table 3 on page 6.
- 11. After completion of the device certificate service, system controller returns a response through the COMM_BLK instance.
- 12. The service response includes the 1 byte command, 1 byte STATUS, and 4 bytes DEVICECERTPTR value. The 768 Bytes device certificate is stored in the location pointed by the DEVICECERTPTR pointer. For more information about device certificate service response details, refer to Table 4 on page 6.
- Note: Microsemi recommends using system services driver provided in the firmware core configurator for detailed implementation of the device certificate service.



а 🔊 Міскоснір сотрапу

Using Device Certificate System Service in SmartFusion2 - Libero SoC v11.7

Table 3 shows the device certificate service request details.

Table 3 • Device Certificate Service Request

Offset	Length (bytes)	Field	Description
0	1	CMD = 0	Command
1	4	DEVICECERTPTR	Pointer to 768-byte buffer to receive the device certificate.

Table 4 shows the device certificate service response details.

Table 4 • Device Certificate Service Response

Offset	Length (bytes)	Field	Description
0	1	CMD = 0	Command
1	1	STATUS	Command status
2	4	DEVICECERTPTR	Pointer to original buffer from request

For more information about System Controller, refer to the UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide.

For more information about COMM_BLK, refer to the Communication Block chapter in the UG0331: SmartFusion2 Microcontroller Subsystem User Guide.



Design Requirements

Table 5 shows the design requirements.

Table 5 • Design Requirements

Design Requirements	Description		
Hardware Requirements			
SmartFusion2 Security Evaluation Kit:	Rev D or later		
12 V adapter			
FlashPro4 programmer			
USB A to Mini-B cable			
Host PC or Laptop	Any 64-bit Windows Operating System		
Software Requirements			
Libero [®] SoC	v11.7		
SoftConsole	v3.4 SP1*		
FlashPro programming software	v11.7		
USB to UART drivers	-		
One of the following serial terminal emulation programs:	-		
HyperTerminal			
• TeraTerm			
• PuTTY			
Note: *For this application note, SoftConsole v3.4 SP1 is SoftConsole v4.0 and Libero SoC v11.7 Tutorial.	s used. For using SoftConsole v4.0, see TU0546:		

Design Description

The design is implemented on the SmartFusion2 Security Evaluation Kit board using the M2S090TS-1FGG484 device.

The design example consists of:

- RC oscillator
- Fabric CCC
- CORERESET
- MSS (DeviceCertificate_MSS_0)

The fabric PLL is used to provide the base clock for the MSS. The system services are run using various C routines in the MSS, as shown in the following sections. In addition, a universal asynchronous receiver/transmitter (UART1) in the MSS is used to display the device certificate information.



Hardware Implementation

Figure 2 shows a block diagram of the design example. The RC oscillator generates a 50 MHz input clock and the fabric PLL generates a 100 MHz clock from the RC oscillator. This 100 MHz clock is used as the base clock for the MSS (DeviceCertificate_MSS_0).

The MMUART_1 signals are used for communicating with the host PC serial terminal program.

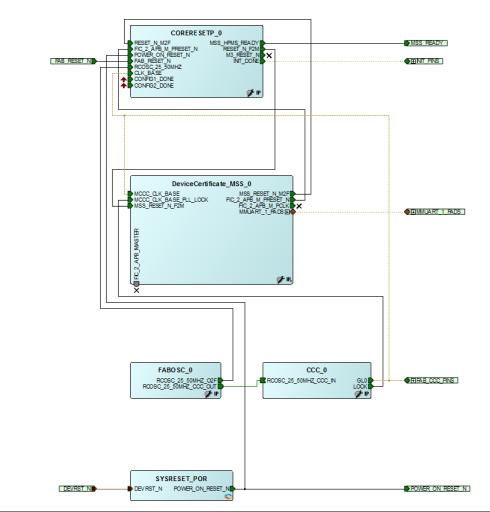


Figure 2 • Block Diagram of SmartFusion2 Device Certificate Design Example



Software Implementation

The software design example is used to display the device certificate information.

Firmware Drivers

The following firmware drivers are used in this application:

- MSS MMUART driver: To communicate with serial terminal program on the host PC.
- MSS system services driver: Provides access to SmartFusion2 system services.

API to Access the Device Certificate Service

The MSS_SYS_get_device_certificate() API is used in software design to access the device certificate service.

Setting Up the Design

•

Ensure that power supply switch SW7 is switched OFF before setting up the SmartFusion2 Security Evaluation Kit, then proceed with the following steps:

- 1. Plug the FlashPro4 ribbon cable into the connector J5 (JTAG Programming Header) of the SmartFusion2 Security Evaluation Kit board.
- 2. Connect FlashPro4 and the USB port of the PC using the mini USB cable.
- 3. Connect the power supply to the J6 connector.
- 4. Connect the J18 connector provided on the SmartFusion2 Security Evaluation Kit to the host PC using the USB mini cable.
- 5. Ensure that the USB to UART bridge drivers are automatically detected by verifying the Device Manager.

Figure 3 shows an example device manager window. If USB to UART bridge drivers are not installed, download and install the drivers from the following location: www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip

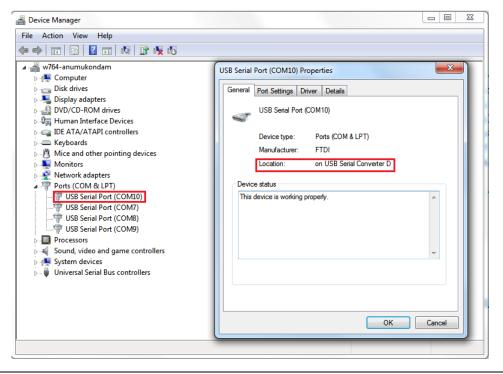


Figure 3 • Device Manager Window



6. Connect the jumpers on the SmartFusion2 Security Evaluation Kit, as shown in Table 6.

Note: Ensure that power supply switch, **SW7** is switched OFF while connecting the jumpers on the SmartFusion2 Security Evaluation Kit.

Jumper	Pin (From)	Pin (To)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

Figure 4 shows the board setup for running the ECC services design on the SmartFusion2 Security Evaluation Kit.

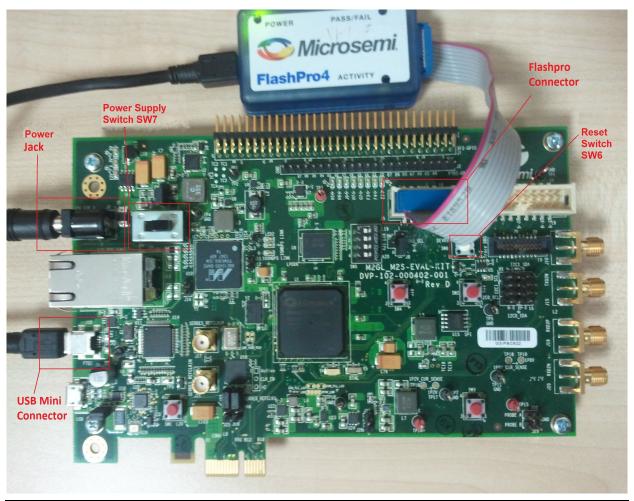


Figure 4 • SmartFusion2 Security Evaluation Kit



Running the Design

The following steps describes how to run the design on the SmartFusion2 Security Evaluation Kit board using the M2S090TS-1FGG484 device:

- 1. Switch ON the power supply switch, SW7.
- 2. Start a PuTTY session with 115200 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. Use any free serial terminal emulation program such as: HyperTerminal or TeraTerm, if the computer does not have the PuTTY program. For more information about configuring HyperTerminal, TeraTerm, or PuTTY, refer to the *Configuring Serial Terminal Emulation Programs Tutorial*.
- 3. Program the SmartFusion2 Security Evaluation Kit board with the provided STAPL file using FlashPro4. Refer to "Appendix A: Design and Programming Files" on page 17 for more information.
- 4. After programming, press the reset switch, **SW6** (DEVRST), then HyperTerminal displays a message and the device certificate information, as shown in Figure 5.

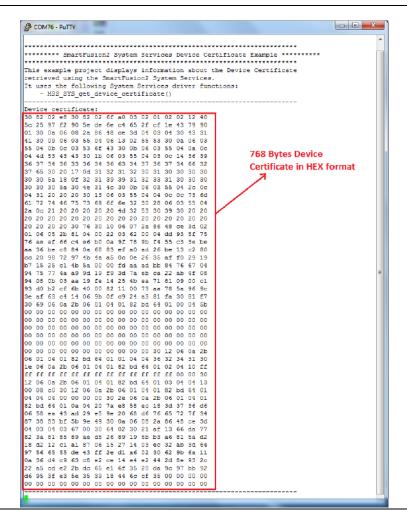


Figure 5 • Device Certificate in Hexadecimal Format (also known as base 16 or hex)



Viewing Fields in Device Certificate

The SmartFusion2 device certificates are encoded in the ASN.1 format. To view the content, the certificates need to be decoded to a user readable format. The content of a device certificate can be decoded in many ways, such as:

- · Use windows utility certutil.exe
- Use open source or other third-party online tools

For more information about how to decode device certificate using ASN.1 JavaScript decoder online tool, refer to "Appendix B: Decoding Device Certificate Using ASN.1 JavaScript Decoder Open Source Tool" on page 18.

This application note uses the <code>certutil.exe</code> windows command tool utility to decode the device certificate. The following steps describe how to decode the device certificate:

- 1. Copy the device certificate HEX values (768 bytes) from the serial terminal program (PuTTY/HyperTerminal) to a text file. For example, copy the device certificate HEX values to the DC_HEX.txt file and save the file in the C\D\E: drive.
- 2. Remove the Padded tailing zeros inserted at the end of the device certificate.
- Note: The actual device certificate length can be found from 3rd and 4th bytes of the certificate, for example in this case the total certificate length is 0x02e9 (3rd and 4th bytes value) + 0x4 i.e. 745+4 = 749 bytes. so remove the last 15 bytes (768-749) of padded zeros from the certificate.
 - 3. Open the command prompt window and type the command E:\>certutil.exe -asn E:\DC_HEX.txt and click Enter key.
 - 4. The command prompt displays the decoded device certificate in a user readable format.

The device certificate fields are highlighted in Figure 6 on page 13, Figure 7 on page 14, Figure 8 on page 15, and Figure 9 on page 16.



Figure 6 shows the device certificate - screen 1.

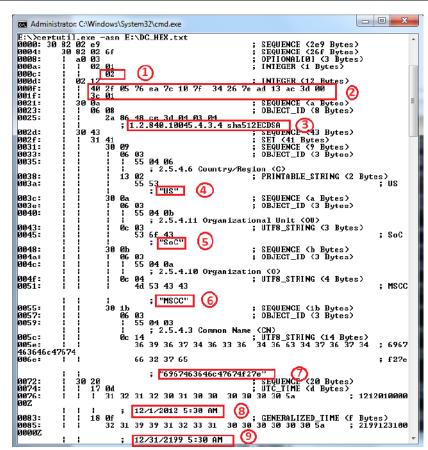


Figure 6 • Device Certificate - Screen 1

The following is the description of labels in Figure 6:

Version Information

- 1. Version Number: 02
- Serial Number Information
 - 2. Certificate Serial Number: 40 c9 39 5e 32 b1 01 33 63 15 3c a4 ae 08 55 c3 da 01

Algorithm ID Information

3. Algorithm ID: SHA512ECDSA

Issuer Information

- 4. Country / Region: US
- 5. Organizational Unit: SoC
- 6. Organization: MSCC
- 7. Common Name: 6967463646c47674f27e used to point to the public key for signature check.

Validity information

Not Before

8. 12/1/2012 5:30 AM

Not After

9. 12/31/2199 5:30 AM



Figure 7 shows the device certificate - screen 2.

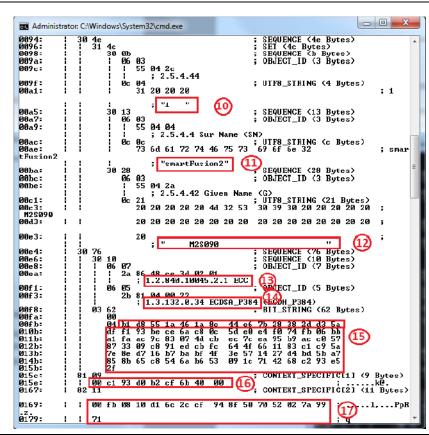


Figure 7 • Device Certificate - Screen 2

The following is the description of labels in Figure 7:

Subject Information

- 10. Rev (Generation Qualifier) 4 bytes fixed length: "1 '
- 11. Family (Surname): "SmartFusion2"
- 12. Product ID 33 characters (Given name): " M2S090

Subject Public Key

- Public Key Algorithm Information
 - 13. ECC
 - 14. ECDSA_P384

Subject Public Key information

- 15. 96 Byte ECC public key
- Issuer Unique Identifier
 - 16. 9 byte bit string: 00 c1 93 d0 b2 cf 6b 40 00

Subject Unique Identifier

17. (0x00+Factory Serial Number + Serial Number Modifier): 00 a5 54 aa 38 fd fc 34 b3 7a ae 36 33 07 cc 10 38

,,



Figure 8 shows the device certificate - screen 3.

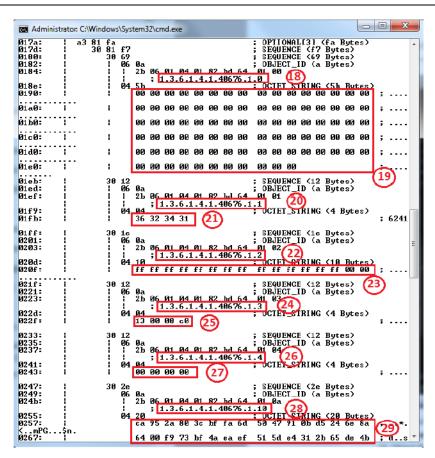


Figure 8 • Device Certificate - Screen 3

The following is the description of labels in Figure 8:

Extension Information

- 18. Object ID 1.3.6.1.4.1.40676.1.0: padding
- 19. 91 bytes zero padding
- 20. Object ID 1.3.6.1.4.1.40676.1.1: Date code
- 21. Date code value: 36 32 34 31
- 22. Object ID 1.3.6.1.4.1.40676.1.2: Temp, Speed, and Voltage Grade
- 24. Object ID 1.3.6.1.4.1.40676.1.3: Reserved
- 25. Reserved
- 26. Object ID 1.3.6.1.4.1.40676.1.4: Reserved
- 27. Reserved
- Object ID 1.3.6.1.4.1.40676.1.10: Certificate Validator. This field is used by the Libero SoC/Flashpro software to validate the Device certificate
- 29. Certificate Validator Value (256 bit validator): "00 31 f0 ab f6 6c 0a 63 c5 27 ef 1f 12 8e 5a 20 8a a8 6c c0 b8 3e 12 ec 19 d0 87 51 1f e0 7c 45"



Figure 9 shows the device certificate - screen 4.

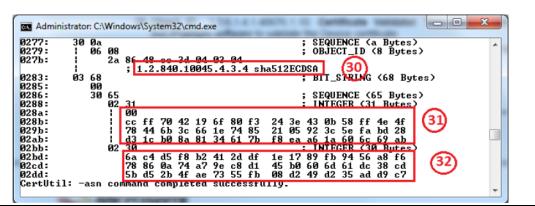


Figure 9 • Device Certificate - Screen 4

The following is the description of labels in Figure 9:

Certificate Signature Algorithm Information

30. Algorithm: sha512ECDSA (Object ID 1.2.840.10045.4.3.4)

Certificate Signature Information: As shown in Figure 9, the highlighted circles 31 and 32 display the certificate signature that is stored in the bit string format.

Conclusion

This application note describes how to implement the device certificate using the system services in the SmartFusion2 SoC FPGAs and view the content of the device certificate.



Appendix A: Design and Programming Files

Download the design files from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_ac436_liberov11p7_df

The design file consists of Libero SoC Verilog project, SoftConsole software project, and programming files (*.stp) for the SmartFusion2 Security Evaluation Kit board. Refer to the Readme.txt file included in the design file for the directory structure and description.

Download the programming files from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_ac436_liberov11p7_pf

The programming file consists of STAPL programming fle (*.stp) for the SmartFusion2 Security Evaluation Kit board.



Appendix B: Decoding Device Certificate Using ASN.1 JavaScript Decoder Open Source Tool

ASN.1 JavaScript decoder is a web tool capable of parsing and showing any valid ASN.1 DER or BER data structure as both a tree and a cross-linked hex-dump.

1. Open any standard web browser (for example, Internet Explorer) and enter the following URL in the address bar: *http://lapo.it/asn1js/*#

ASN.1 online decoder page is displayed, as shown in Figure 10.

2. Click clear.

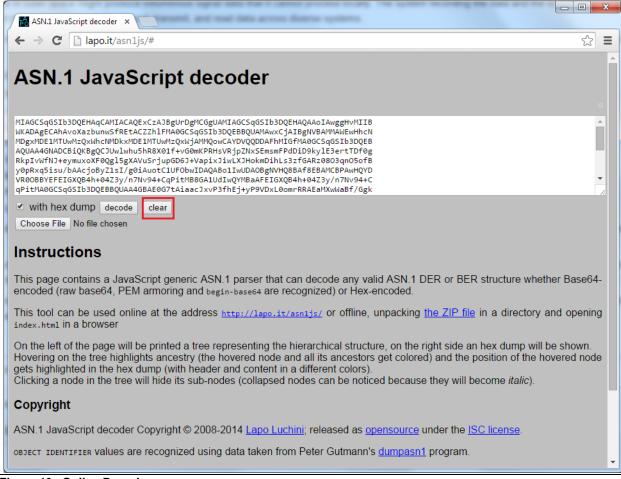


Figure 10 • Online Decoder

- 3. Copy the 768 Bytes HEX format device certificate from any serial terminal program and paste it in the online decoder, as shown in the Figure 11.
- 4. Click decode.



Appendix B: Decoding Device Certificate Using ASN.1 JavaScript Decoder Open Source Tool

ASN.1 JavaScript decoder ×	
← → C [] lapo.it/asn1js/#	☆ =
ASN.1 JavaScript decoder	
24 03 04 03 67 00 30 64 02 30 49 31 57 05 0f 0b 55 b1 a1 37 40 54 64 52 9d 56 b1 p1 37 40 54 64 5a 9d 56 58 57 82 05 66 79 0d 5c 3a 22 e0 58 69 e4 32 94 53 3e 72 74 26 3c ea 02 30 35 f0 87 93 26 d6 89 c3 ae b6 78 61 89 c3 ae ba 56 ad c6 38 55 ad ae ba 56 ad cf e8 63 28 5f a8 ba 56 ad cf e8 63 28 56 ad cf e8 63 28 56 ad	•
✓ with hex dump decode clear Choose File No file chosen	
nstructions	
This page contains a JavaScript generic ASN.1 parser that can decode any valid ASN.1 DER or BER structure whet encoded (raw base64, PEM armoring and begin-base64 are recognized) or Hex-encoded.	ther Base64-
This tool can be used online at the address <u>http://lapo.it/asnljs/</u> or offline, unpacking <u>the ZIP file</u> in a directory .ndex.html in a browser	and opening
On the left of the page will be printed a tree representing the hierarchical structure, on the right side an hex dump will dovering on the tree highlights ancestry (the hovered node and all its ancestors get colored) and the position of the highlighted in the hex dump (with header and content in a different colors). Clicking a node in the tree will hide its sub-nodes (collapsed nodes can be noticed because they will become <i>italic</i>).	
Copyright	
ASN.1 JavaScript decoder Copyright © 2008-2014 Lapo Luchini; released as opensource under the ISC license.	
BJECT IDENTIFIER Values are recognized using data taken from Peter Gutmann's <u>dumpasn1</u> program.	

Figure 11 • Online Decoder with Decode Option



The following window is displayed.

ASN.1 JavaScript decoder ×	
 C lapo.it/asn1js/#308202E83082026FA003020102021240C9395E3 	32B1013363153CA4AE0855C3DA01300A060 🔶
QUENCE (3 elem)	30 82 02 E8 30 82 02 6F A0 03 02 01 02 02 12 40
SEQUENCE (10 elem) [0] (1 elem)	C9 39 5E 32 B1 01 33 63 15 3C A4 AE 08 55 C3 DA
INTEGER 2	01 30 0A 06 08 2A 86 48 CE 3D 04 03 04 30 43 31
INTEGER (143 bit) 5643659310483573095574361427006075246467585	41 30 09 06 03 55 04 06 13 02 55 53 30 0A 06 03 55 04 08 0C 03 53 6F 43 30 08 06 03 55 04 0A 0C
SEQUENCE (1 elem) OBJECT IDENTIFIER 1.2.840.10045.4.3.4	04 4D 53 43 43 30 1B 06 03 55 04 03 0C 14 36 39
SEQUENCE (1 elem)	36 37 34 36 33 36 34 36 63 34 37 36 37 34 66 32
SET (4 elem)	37 65 30 20 17 0D 31 32 31 32 30 31 30 30 30 30
SEQUENCE (2 elem) OBJECT IDENTIFIER 2.5.4.6	30 30 5A 18 0F 32 31 39 39 31 32 33 31 30 30 30
PrintableString US	30 30 30 5A 30 4E 31 4C 30 0B 06 03 55 04 2C 0C
SEQUENCE (2 elem)	04 31 20 20 20 30 13 06 03 55 04 04 0C 0C 73 6D
OBJECT IDENTIFIER 2.5.4.11 UTF8String SoC	61 72 74 46 75 73 69 6F 6E 32 30 28 06 03 55 04 2A 0C 21 20 20 20 20 20 4D 32 53 30 39 30 20 20
SEQUENCE (2 elem)	20 20 20 20 20 20 20 20 20 20 20 20 20 2
OBJECT IDENTIFIER 2.5.4.10	20 20 20 20 30 76 30 10 06 07 2A 86 48 CE 3D 02
UTF8String MSCC SEOUENCE (2 elem)	01 06 05 28 81 04 00 22 03 62 00 04 B8 41 38 BF
OBJECT ÌDENTIFÍER 2.5.4.3	D9 D0 79 3D 67 24 50 E0 41 CA AC 74 8D 3F 7E AD
UTF85tring 6967463646c47674f27e	F4 85 06 F9 F1 07 24 76 03 F8 62 D2 C1 08 A4 47 CE C4 8E 2D 50 71 CD 06 9A 83 50 BB 58 88 20 08
SEQUENCE (2 elem) UTCTime 2012-12-01 00:00:00 UTC	17 36 88 CD 55 57 57 9E F6 F9 90 0B D9 13 AF 0B
GeneralizedTime 2199-12-31 00:00:00 UTC	3D 09 C7 5E A0 95 02 F9 A7 75 98 70 15 2D C1 64
SEQUENCE (1 elem) SET (3 elem)	F7 49 A9 27 0A 04 AB DD 63 C3 6A AB 81 09 00 C1
SEQUENCE (2 elem)	93 D0 B2 CF 6B 40 00 82 11 00 A5 54 AA 38 FD FC
OBJECT ÌDENTIFÍER 2.5.4.44	34 B3 7A AE 36 33 07 CC 10 38 A3 81 FA 30 81 F7
UTF8String 1	30 69 06 0A 2B 06 01 04 01 82 BD 64 01 00 04 5B
SEQUENCE (2 elem) OBJECT IDENTIFIER 2.5.4.4	00 00 00 00 00 00 00 00 00 00 00 00 00
UTF8String smartFusion2	
SEQUENCE (2 elem) OBJECT IDENTIFIER 2.5.4.42	00 00 00 00 00 00 00 00 00 00 00 00 00
UTF8String M2S090	00 00 00 00 00 00 00 00 00 00 00 00 00
SEQUENCE (2 elem)	00 00 00 00 00 00 00 00 00 00 00 30 12 06 0A 2B
SEQUENCE (2 elem)	06 01 04 01 82 BD 64 01 01 04 04 36 32 34 31 30
OBJECT IDENTIFIER 1.2.840.10045.2.1 OBJECT IDENTIFIER 1.3.132.0.34	1E 06 0A 2B 06 01 04 01 82 BD 64 01 02 04 10 FF FF FF
BIT STRING (776 bit) 110101010101011011000011110001101011101111	⁰¹ 12 06 0A 2B 06 01 04 01 82 BD 64 01 03 04 04 13
<pre>[1] (9 byte) 00C193D0B2CF6B4000 [2] (17 byte) 00A554AA38FDFC34B37AAE363307CC1038</pre>	00 08 C0 30 12 06 0A 2B 06 01 04 01 82 BD 64 01
[3] (1 elem)	04 04 04 00 00 00 00 30 2E 06 0A 2B 06 01 04 01
SEQUENCE (6 elem)	82 BD 64 01 0A 04 20 00 31 F0 AB F6 6C 0A 63 C5
SEQUENCE (2 elem) OBJECT IDENTIFIER 1.3.6.1.4.1.40676.1.0	27 EF 1F 12 8E 5A 20 8A A8 6C C0 B8 3E 12 EC 19
OCTET STRING (91 byte) 000000000000000000000000000000000000	D0 87 51 1F E0 7C 45 30 0A 06 08 2A 86 48 CE 3D ³⁶ 04 03 04 03 67 00 30 64 02 30 49 31 57 05 0F 0B
SEQUENCE (2 erem)	35 6B A4 9E 36 B5 E3 8B 1B 13 37 40 54 64 5A 9D
OBJECT ÍDENTIFÍER 1.3.6.1.4.1.40676.1.1 OCTET STRING 6241	6E 58 B7 82 05 E6 79 0D 5C 3A 22 E0 B5 86 99 E4
SEQUENCE (2 elem)	3D 39 4E 53 3E 72 74 26 3C EA 02 30 35 F0 87 93
OBJECT ÌDENTIFÍER 1.3.6.1.4.1.40676.1.2	26 D6 89 C3 AE BA 65 78 1F 96 90 0A 63 28 5F A8
OCTET STRING (1 elem) Private 316912650057057350374175801216 (0 elem)	49 E1 E3 D2 AB 56 AD CF E8 06 87 C3 35 A0 D7 77
SEQUENCE (2 elem)	70 79 0D 93 66 AA AF 38 1D 50 B4 E3
OBJECT IDENTIFIER 1.3.6.1.4.1.40676.1.3	
OCTET STRING (4 byte) 130008C0 SEOUENCE (2 elem)	
OBJECT IDENTIFIER 1.3.6.1.4.1.40676.1.4	
OCTET STRING (4 byte) 00000000	
SEQUENCE (2 elem) OBJECT IDENTIFIER 1.3.6.1.4.1.40676.1.10	
OCTET STRING (32 byte) 0031F0ABF66C0A63C527EF1F128E5A208AA86CC0B	383E12EC19D087511FE07C45
SEQUENCE (1 elem)	
OBJECT IDENTIFIER 1.2.840.10045.4.3.4 BIT STDING (1 elem)	
BIT STRING (1 elem) SEQUENCE (2 elem)	
INTEGER (383 bit) 1126539281721270959380850704685393650800303812081454	
INTEGER (382 bit) 8302059454850576228069896355786673203803224952788832	2177112352005694518

Figure 12 • Decoded Device Certificate

The decoded certificate field values in a tree format are on the left and the HEX dump values are on the right.

For more information about the decoded field values, refer to "Viewing Fields in Device Certificate" on page 12.



List of Changes

Revision*	Changes	Page
Revision 4 (May 2020)	Information about device certificate description was updated.	N/A
Revision 3 (April 2016)	Updated the document for Libero SoC v11.7 software release (SAR 78039).	N/A
Revision 2 (October 2015)	Updated the document for Libero SoC v11.6 software release (SAR 71682).	N/A
Revision 1 (February 2015)	Initial release.	N/A

The following table shows the important changes made in this document for each revision.

Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.



а 🔨 Міскоснір company

Microsemi Headquarters One Enterprise, Aliso Viejo, CA 92656 USA Within the USA: +1 (800) 713-4113 Outside the USA: +1 (949) 380-6100 Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996 Email: sales.support@microsemi.com www.microsemi.com

©2020 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners. Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.