Sign App - User Guide

UG98S08

Maxim Integrated

2018-09-24

1 Description

sign_app build binary file including signature and SLA header from customer application binary file.

2 Usage

```
sign_app [OPTION] [PARAMETERS] [APP [KEYFILE]]
```

3 General Option

3.1 c - chip part number

`-c CHIP_NAME` - Use default configuration of CHIP_NAME.

3.2 Help

`-h` - Print this help and quit.

3.3 Version

`-v` - output software and libraries versions and quit.

3.4 Debug

`-d` - Activate debug output.

4 Parameters

Parameters are used by priority in the following order :

1. Command line
2. Configuration file ” INIFILE ” in the current folder
3. Chip default parameters selected by the -c option or the MAXIM_SBT_DEVICE env variable.
4. Software default values.

4.1 Signing algorithm

```
algo=algo
```

algo - Algorithm to be used to sign the application Please refers to CHIP documentation to select the corect one. Available algorithms are :

- rsa
- rsa_paola
- none
- ecdsa
- crc32
- sha256

4.2 Key file

```
key_file=file.key
```

UCL format private key file for SCP packet signing. For more information see UCL Key Format.

4.3 Signature Only

```
signonly=yes
signonly=no
```

yes - Only a sig file containing the signature will be generated no - A signed binary (binary + signature) file will be also generated.

4.4 Generate SLA header

```
header=yes
header=no
```

yes - SLA header will be generated according to parameters and added at the begining of binary no - No header will be egenrate, it is supposed that the header is already present in binary

4.5 Verbose

```
verbose=level
```

verbose level (0-5)

5 Header Parameter

5.1 App version

```
application_version=version
```

`version` - Version of the application - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

5.2 Bootloader Version

```
rom_version=version
```

`version` - Version of the targeted Bootloader, Please refers to CHIP documentation to select the corect one. - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

5.3 Load Address

```
load_address=address
```

`address` - Address of the location where the application will be copy before executed. - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

5.4 Jump Address

```
jump_address=address
```

`address` - Address of the instruction where the bootloader will jump. - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

5.5 Arguments

```
arguments=args
```

`args` - Argument for the application to include inside the header. The pointer arguments will be store in r0 and the length in r1 before jumping in the application. - String (ex : `argument1 argument2`)

5.6 Boot Method

```
boot_method=cmsis
boot_method=direct
```

`cmsis` - The Jump address points to the value ot the *Stack pointer* followed by the address of the *reset handler*. The boot loader will setup the Stack pointer and then jump to the ”reset handler*. `direct` - The bootloader will directly jump to the Jump address and the application is responsible to setting up the stack.

6 HSM Parameter

This application can use a Thales(R) HSM for key storage and cryptographics operation. By default the application use it’s builtin cryptographics functions.

6.1 HSM

```
hsm=yes
hsm=no
```

Use or not an HSM to manage key and perform cryptographics operations

6.2 HSM Key Name

```
hsm_key_name=name
```

`name` - name of the key to use stored inside the HSM.

6.3 HSM Thales DLL Location

```
hsm_thales_dll=dll_path
```

`dll_path` - path to the Thales cknfast DLL.

6.4 HSM SLot Number

```
hsm_slot_nb=nb
```

`nb` - number of the HSM slot to use (usually : 1).

7 MAX3259x Parameter

7.1 SDRAM Power Down

```
sr_papd=value
```

`value` - DMC Primary SDRAM Power down register value - 1 bytes hexadecimal encoded : (ex : `0A 0xA1`)

7.2 LPDDR Mode

```
sr_pext=value
```

`value` - DMC Primary LPDDR Mode register value - 1 bytes hexadecimal encoded : (ex : `0A 0xA1`)

7.3 SDRAM Refresh

```
sr_prfsh=value
```

`value` - DMC Primary SDRAM Refresh register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

7.4 SDRAM Configuration

```
sr_pcfg=value
```

`value` - DMC Primary SDRAM Configuration register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

7.5 DMC Global configuration

```
dmc_gcfg=value
```

`value` - DMC Global config register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

7.6 DMC Clock

```
dmc_clk=value
```

`value` - DMC Clock config register value - 1 bytes hexadecimal encoded : (ex : `0A 0xA1`)

7.7 UCI AES Key

```
uci0_ksrc_configencint=value
```

`value` - UCI AES Encryption key 0 register value - 1 bytes hexadecimal encoded : (ex : `0A 0xA1`)

7.8 UCI Area Config 0

```
uci0_ac1r_so=value
```

`value` - UCI Area Config 0 Start Offset register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

```
uci0_ac1r_eo=value
```

`value` - UCI Area Config 0 End Offset register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8` `0x0123ACE8`)

7.9 UCI DDR Region 0 Config

```
uci0_ddr_r0=value
```

`value` - UCI DDR Region 0 Config register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8` `0x0123ACE8`)