

# **MAX32550 SecureROM package installer and quick start guide**

**Contents**

<b>1</b>	<b>Version</b>	<b>1</b>
<b>2</b>	<b>Purpose</b>	<b>1</b>
<b>3</b>	<b>Release Notes</b>	<b>1</b>
<b>4</b>	<b>Tested configuration</b>	<b>1</b>
<b>5</b>	<b>3rd parties packages</b>	<b>1</b>
<b>6</b>	<b>Installation</b>	<b>2</b>
6.1	Windows users only . . . . .	2
6.2	Linux users only . . . . .	2
<b>7</b>	<b>Use</b>	<b>3</b>
7.1	program the CRK . . . . .	3
7.2	program the application . . . . .	4
7.3	Execution . . . . .	6
<b>8</b>	<b>Notes</b>	<b>6</b>
<b>9</b>	<b>Glossary</b>	<b>7</b>
<b>10</b>	<b>Tools purpose</b>	<b>7</b>

## 1 Version

- this document version is 1.1.2 (18-sep-2014)

## 2 Purpose

- the MAX32550 SecureROM package is a set of tools, examples and documents enabling user to securely program, load and run applications on the MAX32550 secure microcontroller .
- this package enables users to set a MAX32550 in phase 3, program the CRK on the MAX32550, move the MAX32550 to phase 4, program an application and runs it on the MAX32550.
- this package requires installation of 3rd parties tools.
- This is run on a PC, connected to the MAX32550.
- this package demonstrates:
  - how to set up the MAX32550 life-cycle,
  - how to program the customer CRK, using a test CRK,
  - how to program a demo application in the internal flash, to be run in the internal RAM.
  - a glossary is available at the end of this document

## 3 Release Notes

- this package version supports now MAX32550 A3: this is the default target
- this package version is validated on Windows Seven 64-bit and not on any 32-bit Windows OS

## 4 Tested configuration

- on PC side, this package has been tested on:
  - Windows Seven 64-bit, using Cygwin 64-bit 1.7,
  - Ubuntu 12.04 LTS,
- on embedded side, this package has been tested on
  - MAX32550 A3
  - Evkit 2.0

## 5 3rd parties packages

- the Maxim Integrated delivered package contains documentation, examples and tools, from Maxim Integrated.
- 3rd party tools to be installed are:
  - For Windows users only:
    - \* **Cygwin**
      - a collection of tools which provide a Linux look and feel environment for Windows
      - needed for tools execution
      - download and install guide

## 6 Installation

### 6.1 Windows users only

1. copy the SecureROM package in /cygdrive/c/securerom (i.e. c:/securerom)
2. cygwin installation
  - a. use the Cygwin\_install.pdf document
3. set up package application
  - a. goto secureROM package home directory, e.g. securerom
  - b. select the targeted MAX32550 version, A2 or A3, by using the option --soc=A3 or --soc=A2 (default is A3)

```
$ cd securerom
$ bash setup.sh --soc=A3
Setting up MAX32550 rev.A3's SecureROM Package... for cygwin
dos2unix: converting file Host/customer_scripts/lib/romversion to Unix ↵
format ...
Info:: New environment variable ANGELA_SCRIPTS_PATH added by
      ANGELA PACKAGE's setup.sh script into /home/<username>/. ↵
      bash_profile.
>>> You need to logout then login again to make this variable available ↵
.
>>> You can also do "source /home/<username>/.bash_profile" to make ↵
      this variable available (in this terminal session only).
```

- c. once this is done, the ANGELA\_SCRIPTS\_PATH variable can be used for locating the package directory.

### 6.2 Linux users only

1. copy the SecureROM package in ~/securerom
2. set up package application
  - a. goto secureROM package home directory, e.g. securerom
  - b. select the targeted MAX32550 version, A2 or A3, by using the option --soc=A3 or --soc=A2 (default is A3)

```
$ cd securerom
$ bash setup.sh linux ./
Setting up MAX32550 rev.A3's SecureROM Package... for linux
Info:: New environment variable ANGELA_SCRIPTS_PATH added by
      ANGELA PACKAGE's setup.sh script into /home/<username>/. ↵
      bash_profile.
>>> You need to logout then login again to make this variable available ↵
.
>>> You can also do "source /home/<username>/.bash_profile" to make ↵
      this variable available (in this terminal session only).
```

- c. once this is done, the ANGELA\_SCRIPTS\_PATH variable can be used for locating the package directory.

## 7 Use

### 7.1 program the CRK

1. program the test CRK and move MAX32550 life cycle phase from 3 to 4 (to be done once per chip)

- a. Note: the SCP packets have already been generated (and stored in prod\_p3\_write\_crk) for this test CRK, this is why programming can be done immediately. These packets are an example of what Maxim Integrated generates from the CRK public key supplied to Maxim Integrated.
- b. the connection between the Host and the MAX32550 is a serial connection, 115200 8N1,
- c. connect the EvKit UART0 connector to the targeted COM port,
- d. run writecrk.sh with the chosen COM port (in our example, COM1 for Windows, /dev/ttyS0 for Linux) and prod\_p3\_write\_crk as parameters

```
$ cd $ANGELA_SCRIPTS_PATH/Host/customer_scripts/scripts
$ bash ./writecrk.sh COM1 ../../../../SCP_Packets/prod_p3_write_crk
Ready to execute /cygdrive/h/PACKAGE_ANGELA_1.0.1/SCP_Packets/ ↵
prod_p3_write_crk
Power cycle the MAX32550 system then press [Enter] IMMEDIATELY!
Please wait...
Open file: packet.list
Open serial port: COM1 (timeout: 2s)
Start SCP session (use -v for details)

0000001 SEND> connection_request
0000002 WAIT< connection_reply
0000003 SEND> ack
0000004 SEND> hello_request
0000005 WAIT< ack
0000006 WAIT< hello_reply
error: received packet is not the expected one
=====
Expected Command
ScpCmd:
ScpCmdHdr:
- SYNC: ('\xbe', '\xef', '\xed')
- CTL: 5
- DL: 54
- ID: 145
- CKS: 98
Data:
2000003248454 ↵
c4c4f20484f5354010000000000000c00500abcdef01000102abcdaf600000000000
00000000000000000000000000000000
Checksum: a836acf6
-----
Received Command
ScpCmd:
ScpCmdHdr:
- SYNC: ('\xbe', '\xef', '\xed')
- CTL: 5
- DL: 54
- ID: 145
- CKS: 98
Data:
```

```

2000003248454 ←
    c4c4f20484f535406000102030000400500abcdef01000102abcdf6ae0000000000
    00000000000000000000000000000000
Checksum: ae138608
=====
00000007 SEND> ack
00000008 SEND> write_crk
00000009 WAIT< ack
00000010 WAIT< write_crk_response
00000011 SEND> ack
00000012 SEND> disconnection_request
00000013 WAIT< disconnection_reply
SCP session OK
SUCCESS.

```

- e. Note1: the MAX32550 is now in phase 4, with the test CRK programmed in its OTP. Any SCP command will now require a signature with this key to be approved.
- f. Note2: the error message displayed during the CRK programming is related to the MAX32550 (varying) USN which is different from the (fixed) USN used in the packet. It is not an issue but an expected warning.

## 7.2 program the application

1. program test application in the MAX32550 internal flash,
  - a. signed application
    - i. the application signing operation is automatically performed within the COBRA environment, in the Eclipse IDE, usually using the "format binary" target
    - ii. in the SLA/build, there is a binary example of such signed application, sla\_template.sbin
    - iii. this .sbin has been signed using the test CRK private key.
    - iv. the version of the application shall be consistent with the MAX32550 version. For example, the version to be used with the A3 is 1.0.3, coded 01000003. This has been set up automatically by the setup.sh script. The hexadecimal parameters are the load address and the jump address. These parameters have also been provided in the COBRA environment, at the signature step.

- b. generate the packets for the application programming

```
$ cd $ANGELA_SCRIPTS_PATH/Host/customer_scripts/scripts
$ bash ./build_application.sh ../../../../SLA/build/sla_template.bin.sbin ↵
    buildapp ../keys/maximtestcrk.key
SBL/SCP packets builder v3.7.4 (build 1) (c)Maxim IC 2006-2013
--warning: this tool does not handle keys in a manner compliant with PCI ↵
    PTS--
WARNING: <session_build.ini> not found
<buildapp/scp.log> created
Generated SCP packets in: buildapp
Use sendscp.sh <serial_port_spec> buildapp to run the SCP session ↵
    generated by this script.
SUCCESS.
```

- i. buildapp directory has been created and packets have been generated and stored in
- c. program the generated packets
  - i. connect the EvKit to the targeted serial port (COM1 is for Windows, to be replaced by /dev/ttyS0 for Linux)
  - ii. use sendscp.sh script to run the SCP session generated by this script

[illegible]

```
0000018 WAIT< write_mem_response
0000019 SEND> ack
0000020 SEND> write_mem
0000021 WAIT< ack
0000022 WAIT< write_mem_response
0000023 SEND> ack
0000024 SEND> write_mem
0000025 WAIT< ack
0000026 WAIT< write_mem_response
0000027 SEND> ack
0000028 SEND> disconnection_request
0000029 WAIT< disconnection_reply
SCP session OK
SUCCESS.
```

### 7.3 Execution

1. start the application by resetting the board
  - a. after more than 10s (5s for SCP-USB time window, 5s for SCP-Serial time window), the EvKit LED0 blinks.

## 8 Notes

- the setup.sh configures the package for a specific MAX32550 version. For another configuration, just run setup.sh with the accurate version specified with the --soc parameter.
- 32-bit Windows binary files (libucl.dll and session\_build.exe) can be found in Host/session\_build/windows\_32bits. If your system is not 64-bit compliant, copy these files
- the SCP USB does not work on A2
- it is possible to change the SCP ports (UART and USB) timings, using a script and session build tool
  - for no UART timing: write-timeout 0 0000
  - for no USB timing: write-timeout U 0000
  - for no VBUS timing: write-timeout V 0000
  - for USB timing set up at 5s (i.e. 5000ms = 0x1388): write-timeout U 8813
  - the typical use is:

```
$ cd Host/customer_scripts/scripts
$ mkdir session_timeout
$ ../lib/session_build.exe session_mode=SCP_ANGELA_ECDSA verbose=yes ↵
  output_file=session_timeout/session_timeout pp=ECDSA script_file= ↵
  script_timeout.txt ecdsa_file=../keys/crk_ecdsa_angela_test.key
$ cd session_timeout
$ ls -l *.packet >packet.list
$ cd ..
$ bash ./sendscp.sh COM1 session_timeout
```



## 9 Glossary

- CRK: Customer Root Key, the customer ECDSA 256-bit public key, to be written in the OTP, used for signatures verifications,
- OTP: One-Time Programming Memory, an embedded memory used for configuration data storage,
- phase: the MAX32550 life cycle is made of several phases: phase 3 allows to program the CRK, phase 4 allows to program and run applications,
- SCP: Secure Communication Protocol: the secure protocol over UART and USB used for firmware, keys updates, OTP configuration, applets execution
- SLA: Second-Level Application: the final application launched by the SecureROM.
- USN: Unique Serial Number, a value unique per chip.

## 10 Tools purpose

- for more details, see UG25H04 *Secure ROM code User Guide*
  - session\_build: builds the SCP packets from a script (typically the signed CRK or a signed final application)
  - serial\_sender: sends the SCP packets to the chip
-