



AN1089: Using Installation Codes with Zigbee Devices

This application note provides an overview of using installation codes with zigbee devices. It also explains (with the help of examples) how to use the EM3xx utilities or Simplicity Commander to check, write, verify, and erase installation codes on Silicon Labs EM3x and Wireless Gecko (EFR32™) devices.

KEY FEATURES

- Concepts of the zigbee installation code
- Programming examples for installation codes on EM3x and EFR32 devices
- Checking, writing, verifying, and erasing installation codes on target devices

1 Installation Code Overview

1.1 What Is an Installation Code?

Zigbee installation codes, sometimes also referred to as “install codes,” are provided as a means for a device to join a zigbee network in a reasonably secure fashion. The installation code itself is a random value installed on the joining device at manufacturing time, and is used to encrypt the initial message exchange between it and the Trust Center zigbee network’s centralized Trust Center device (the coordinator). With the creation of the zigbee 3.0 standard in late 2016, all zigbee devices capable of joining networks (as opposed to forming them) must support the use of installation codes during joining as this is a requirement for zigbee 3.0 compliance.

The installation code can be thought of as similar to the PIN code on Bluetooth devices when two devices are paired. The PIN code is provided as an authorization code for the parent device so that the joining device knows it is receiving information securely, such as when a hands-free headset is paired to a smartphone.

The installation code is typically printed on the case or packaging of the device, either as a hexadecimal string or in an encoded fashion such as a barcode or QR code, and provided via an out-of-band mechanism to the Trust Center device or its associated web/cloud interface, along with the 64-bit IEEE MAC address (“EUI64”) of the device. If this device-specific data is stored on a remote web server or cloud-based system, that remote system then securely transports that information to the Trust Center to establish security credentials for the joining device in advance of the in-band joining process.

1.2 Caveats for Zigbee Smart Energy (ZSE) Devices

The Trust Center and the joining device use the installation code as a shared key to establish an initial bond of trust allowing the new device to join the zigbee network. Once the device has successfully joined the network for which it is authorized, zigbee requires that the node negotiate a new Trust Center link key for future secure exchanges with the Trust Center. In traditional zigbee 3.0 networks, this occurs via a key request directly to the Trust Center. However, in zigbee smart energy networks, which behave differently from normal zigbee 3.0 networks, the new Trust Center link key is derived through a special process known as Certificate-Based Key Establishment (CBKE). For more information about the CBKE process, refer to *UG103.5: Fundamentals of Zigbee Security*. Note that the CBKE process requires installing CBKE data certificates signed by Certicom during the manufacturing process. Refer to *AN708: Setting Smart Energy Certificates for Zigbee Devices* for details about how to set these certificate data. Also consult *AN714: Smart Energy ECC-Enabled Device Setup Process* for more information about the requirements for preparing zigbee smart energy (ZSE) devices to be able to join a network and for troubleshooting this process.

This document outlines common practices relating to installation codes for either a standard zigbee 3.0 device or a ZSE device.

2 Security Use

An installation code is used to create a preconfigured, link key. The installation code is transformed into a link key by use on an AES-MMO hash algorithm. For more information and sample code, consult the Install Codes section of the Security chapter of the zigbee alliance's Base Device Behavior Specification (zigbee document #13-0402).

The installation code, while not exactly a secret, cannot be easily guessed by a malicious device that hears the initial exchange between the joining device and the Trust Center. Without knowledge of the installation code and thus the key, the malicious device cannot decrypt the messages.

The derived zigbee link will be known only by the Trust Center and the joining device. The Trust Center uses that key to securely transport the zigbee network key to the device. Once the device has the network key, it can communicate at the network layer to the zigbee network. It has the ability to perform service discovery and begin the application's initialization process. In zigbee 3.0 (non-ZSE) networks, having the network key is generally enough for standard messaging across various clusters. However, ZSE networks have additional restrictions as discussed below.

The initial link key derived from the installation code does not have full access privileges on a ZSE network. Attempts to use it for Smart Energy messaging are not allowed and will be ignored by other ZSE devices. Shortly after joining a network, a device must use the Key Establishment cluster to establish a new link key with the Trust Center via the CBKE process. Only when key establishment completes successfully will a device have full privileges on the network and be able send and receive certain ZSE messages.

3 Installation Code Format

While zigbee smart energy networks allow the installation code to be comprised of either 6-, 8-, 12-, or 16-byte random, hexadecimal number with a 2-byte CRC appended to the end, zigbee 3.0 networks specifically require 16-byte hexadecimal installation codes, also accompanied by a 2-byte CRC. Note that the CRC16 should be delivered to the user in least significant byte (LSB) order, as this is what is expected when the code is entered into the device that performs the AES-MMO hash algorithm. As far as the user is concerned, the CRC is part of the installation code and they do not need to know that it is there or why. Therefore, from the user's point of view, the length of the install code is 18 bytes (with potentially 8-, 10-, or 14-byte variants possible in ZSE devices).

Manufacturing and managing the list of installation codes will play a part in choosing the size, security, and user experience in installing the device. A larger installation code size will mean less of a chance of an attacker “guessing” the installation code and eavesdropping on the initial join. However smaller installation code is much easier for a user to read off the device during installation.

Note: It is required by the zigbee 3.0 Base Device Behavior Specification that you only use a 16-byte installation code. While this may be more difficult to enter, it provides sufficient strength against an attacker from guessing the installation code and gaining unauthorized access to network or device.

4 Installation Code CRC

The installation code CRC is mechanism used to verify the integrity of an installation code when it is transmitted via an out-of-band mechanism to the utility. This transport mechanism involves human interaction in some way. As a result, the CRC was designed as a way to verify that an installation code is valid and was not mistakenly changed during transport.

The zigbee installation model enables users to install a device themselves. Users simply read the installation code on the back of the device and enter it into a webpage or provide it over the phone to a utility service. Because the number is a hexadecimal value, it is easy to transpose digits or read the wrong value.

4.1 Validation

The zigbee specification expects that the server processing the out-of-band installation code entry from the installer will perform basic checking of the installation code for validity. The server then calculates the CRC over all bytes in the installation code except the final two. It then compares the final two bytes of the installation code with the calculated CRC to see if they match. If they do not match, the user entering the installation code can be informed immediately that it does not look valid. The user should then double-check the value.

Zigbee specifications do not require the Trust Center to validate the installation code directly. (Any validation can be done on a remote web- or cloud-based server if the Trust Center doesn't have this capability locally.) The Trust Center expects to receive a pre-configured link key along with the EUI64 of the new joining device. It does not need to have any knowledge about how that key was derived. It is up to the particular utility how it wishes to manage and transport the link key to the Trust Center.

For details on how the CRC is calculated, including sample code, consult the Install Codes section of the Security Chapter of the zigbee 3.0 Base Device Behavior Specification (zigbee document #13-0402).

4.2 Generation

Silicon Labs recommends that the installation code be a random number. This reduces the chances of an attacker guessing the installation code and compromising the initial join procedure. The installation code should not be based on the manufacturing process, such as tied to the EUI64 or sequential numbering based on the manufacturing lot.

If that were the case, an attacker with knowledge about the type of device being joined would have a known range of installation codes it could try to compromise the network and clone the device's identity. An installation code does not have to be unique across all zigbee devices for all manufacturers.

4.3 Labels

The device's installation code should be printed on a label on the outside of the device along with its EUI64. Both elements should be identified with text indicating what they are. The installation code should not be printed on the outside of the box because that makes it easier for an attacker to gain knowledge of the installation code and potentially compromise the device. It is recommended that the installation code be printed in 2-byte blocks (for example, 83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5).

Note: The CRC should be appended to the installation code in little endian format on the label.

4.4 Example

The following is an 18-byte installation code label (16-byte random code with a 2-byte CRC):

```
83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5 C3B5
```

The random number portion of the code is the first 16 sequential bytes. The calculated CRC value is 0xB5C3, but it is appended in little-endian format.

5 Programming the Installation Code on a Zigbee Device

5.1 Format of the Installation Code File

To program the installation code, create a simple text file with the value of the installation code (without the CRC). This file is passed into Simplicity Commander.

The format of the file is as follows:

```
Install Code: <ascii-hex>
```

Here is a sample installation code file. The CRC for that code is 0xB5C3 and does not need to be present in the file.

```
Install Code: 83FED3407A939723A5C639B26916D505
```

The installation code must be 16 bytes in length (not including the two-byte CRC).

5.2 Checking the Installation Code on an EM3x Device

To get started, it is best to verify there is connectivity with the device to be programmed, and what information is currently stored on the node. To do this, execute the following command to print all manufacturing token data from an EM3x-based device:

```
$ ./em3xx_load.exe --cibtokensprint
```

You should see output similar to the following, where the highlighted portion below reflects the significant fields related to the installation code:

```
$ em3xx_load.exe --cibtokensprint
```

```
em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3588

'General' token group
TOKEN_MFG_CIB_OBS [16 byte array ] : A55AFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64 [8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_STRING [16 byte string] : "" (0 of 16 chars) FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME [16 byte string] : "" (0 of 16 chars) FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG [16-bit integer] : 0xFF26
TOKEN_MFG_BOOTLOAD_AES_KEY [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE [8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM [16-bit integer] : 0xFFFF
TOKEN_MFG_SYNTH_FREQ_OFFSET [16-bit integer] : 0xFFFF
TOKEN_MFG_OSC24M_SETTLE_DELAY [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURITY_CONFIG [16-bit integer] : 0xFFFF
TOKEN_MFG_CCA_THRESHOLD [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURE_BOOTLOADER_KEY [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
Device Private Key [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
CBKE Flags [1 byte array ] : FF
```

```
'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC [16-bit integer] : 0xFFFF

'Smart Energy 1.2 CBKE (TOKEN_MFG_CBKE_283K1_DATA)' token group
Device Implicit Cert (283k1) [74 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFF
CA Public Key (283k1) [37 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFF
Device Private Key (283k1) [36 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFF
CBKE FLAGS (283k1) [ 1 byte array ] : FF
DONE
```

Note: The same installation code token is used for both zigbee smart energy devices and zigbee 3.0 devices, so it is acceptable that the installation code token appears with a note about “Smart Energy Install Code” in the above output.

5.3 Writing the Installation Code into the Manufacturing Area on an EM3x Device

To write the installation code into the manufacturing area, execute the following command:

```
$ em3xx_load.exe --cibtokenspatch install-code-file.txt
```

5.4 Verifying the Stored Installation Code on an EM3x Device

After writing the installation code, it is best to verify the information by executing the following command again:

```
$ ./em3xx_load.exe --cibtokensprint
```

Output of this command should be similar to that shown in [section 5.2, Checking the Installation Code on an EM3x Device](#), but with the installation code fields updated according to the `install-code-file.txt` file.

5.5 Checking the Installation Code on an EFR32 Device

To get started, it is best to verify there is connectivity with the device to be programmed, and what information is currently stored on the node. To do this, execute the following command to print all manufacturing token data from an EFR32-based device. The `tokendump` command prints manufacturing token data as key-value pairs. Simplicity Commander supports more than one group of tokens. In this example, the token group named ‘znet’ is used.

```
$ commander tokendump --tokengroup znet
```

You should see the following output, where the highlighted portion below reflects the significant fields related to the installation code:

```
#
# The token data can be in one of three main forms: byte-array, integer, or string.
# Byte-arrays are a series of hexadecimal numbers of the required length.
# Integers are BIG endian hexadecimal numbers.
# String data is a quoted set of ASCII characters.
#
# MFG_EMBER_EUI_64      : A8D417FEFF570B00
MFG_CUSTOM_VERSION     : 0xFFFF
MFG_CUSTOM_EUI_64      : FFFFFFFFFFFFFFFF
MFG_STRING              : " "
MFG_BOARD_NAME         : " "
MFG_MANUF_ID           : 0xFFFF
MFG_PHY_CONFIG         : 0xFFFF
MFG_SYNTH_FREQ_OFFSET  : 0xFFFF
MFG_CCA_THRESHOLD      : 0xFFFF
MFG_EZSP_STORAGE       : FFFFFFFFFFFFFFFF
MFG_CTUNE              : 0xFFFF
```

[illegible]

```
MFG_ASH_CONFIG[0] : 0xFFFF
MFG_ASH_CONFIG[1] : 0xFFFF
MFG_ASH_CONFIG[2] : 0xFFFF
MFG_ASH_CONFIG[3] : 0xFFFF
MFG_ASH_CONFIG[4] : 0xFFFF
MFG_ASH_CONFIG[5] : 0xFFFF
MFG_ASH_CONFIG[6] : 0xFFFF
MFG_ASH_CONFIG[7] : 0xFFFF
MFG_ASH_CONFIG[8] : 0xFFFF
MFG_ASH_CONFIG[9] : 0xFFFF
MFG_ASH_CONFIG[10] : 0xFFFF
MFG_ASH_CONFIG[11] : 0xFFFF
MFG_ASH_CONFIG[12] : 0xFFFF
MFG_ASH_CONFIG[13] : 0xFFFF
MFG_ASH_CONFIG[14] : 0xFFFF
MFG_ASH_CONFIG[15] : 0xFFFF
MFG_ASH_CONFIG[16] : 0xFFFF
MFG_ASH_CONFIG[17] : 0xFFFF
MFG_ASH_CONFIG[18] : 0xFFFF
MFG_ASH_CONFIG[19] : 0xFFFF
```

```
#'MFG CBKE DATA (Smart Energy CBKE)' token group
```

```
Device Implicit Cert :
```

1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525

```
CA Public Key      : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

```
Device Private Key      : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

```
# CBKE Flags : 0xFF
```

#'MFG INSTALLATION CODE (Smart Energy Install Code)' token group

```
# Install Code Flags : 0xFFFF
```

```
Install Code      : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

```
# CRC      : 0xFFFF
```

```
#'MFG_SECURE_BOOTLOADER_KEY (Manufacture token space for storing secure bootloader key.)' token
group
```

MFG SECURE BOOTLOADER KEY : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

```
#'MFG_CBKE_283K1_DATA (Smart Energy 1.2 CBKE)' token group
```

```
Device Implicit Cert (283k1) :
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 104

FFF

CA Public Key (283k1) :

[illegible]

```
-----
Device Private Key (283k1)      :
```

[illegible]

```
# CBKE FLAGS (283k1) : 0xFF
```

```
#'MFG_SIGNED_BOOTLOADER_KEY_X (Manufacture token space for storing ECDSA signed bootloader key (X-point).)' token group
```

[illegible]

```
#'MFG_SIGNED_BOOTLOADER_KEY_Y (Manufacture token space for storing ECDSA signed bootloader key (Y-point).)' token group
```

[illegible]

DONE

The pre-programmed EUI64 is read out by executing the following command.

```
commander tokendump --tokengroup znet --token MFG_EMBER_EUI_64
#
# The token data can be in one of three main forms: byte-array, integer, or string.
# Byte-arrays are a series of hexadecimal numbers of the required length.
# Integers are BIG endian hexadecimal numbers.
# String data is a quoted set of ASCII characters.
#
# MFG_EMBER_EUI_64: A8D417FEFF570B00

DONE
```

5.6 Writing the Installation Code into the Manufacturing Area on an EFR32 Device

To write the installation code into the manufacturing area, execute the following command:

```
$ commander flash --tokengroup znet --tokenfile install-code-file.txt
```

You should see output similar to the following:

```
Writing 2048 bytes starting at address 0x0fe04000
Comparing range 0x0FE04000 - 0x0FE047FF (2 KB)
Programming range 0x0FE04270 - 0x0FE04283 (20 Bytes)
Verifying range 0x0FE04000 - 0x0FE047FF (2 KB)
DONE
```

5.7 Verifying the Stored Installation Code on an EFR32 Device

After writing the installation code, it is best to verify the information by executing the following command again:

```
$ commander tokendump --tokengroup znet
```

You should see output similar to the yellow-highlighted area of the example code in [section 5.5, Checking the Installation Code on an EFR32 Device](#), but with the MFG_INSTALLATION_CODE data now representing your chosen code and LSB CRC.

6 Erasing the Installation Code

If you want to remove this security data from the device, simply create an installation code file with the contents as “!ERASE!” such as the example below, and then program this file into the target per the instructions either [section 5.3, Writing the Installation Code into the Manufacturing Area on an EM3x Device](#) or [section 5.6, Writing the Installation Code into the Manufacturing Area on an EFR32 Device](#), depending on your target device.

```
Install Code: !ERASE!
```

Output it similar to that seen in [section 5.3, Writing the Installation Code into the Manufacturing Area on an EM3x Device](#) and can be confirmed with a procedure similar to [section 5.2, Checking the Installation Code on an EM3x Device](#) but now your MFG_INSTALLATION_CODE token data should reflect all 0xFF bytes.



Smart.
Connected.
Energy-Friendly.



Products

www.silabs.com/products



Quality

www.silabs.com/quality



Support and Community

community.silabs.com

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

<http://www.silabs.com>