

Secure 2 click

PID: MIKROE-2760

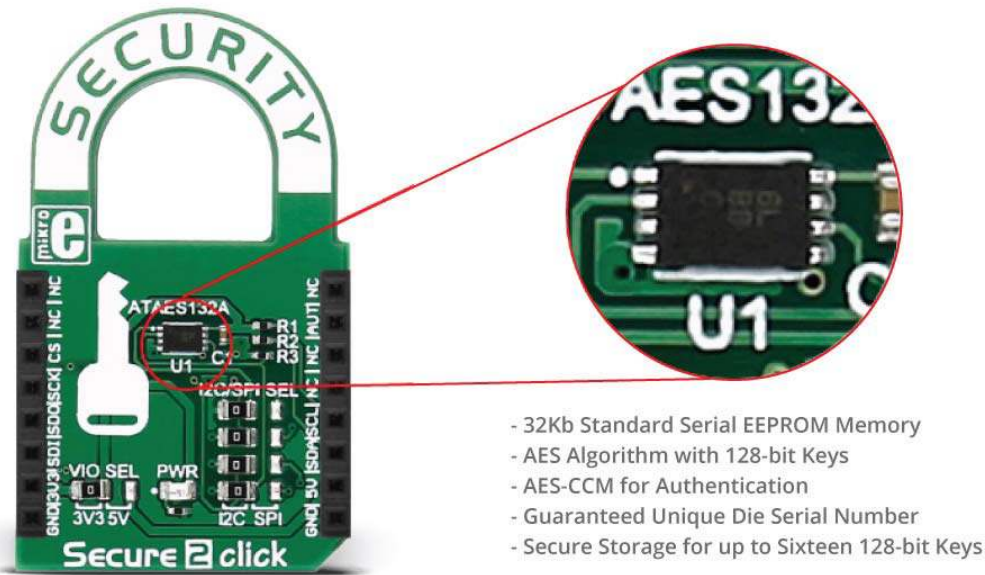


Secure 2 click carries the ATAES132A, a cryptographic coprocessor with secure hardware-based key storage from Microchip. The click is designed to run on either 3.3V or 5V power supply. Secure 2 click communicates with the target microcontroller over SPI and I2C interface, with additional functionality provided by the INT pin on the mikroBUS™ line. The click comes with stackable headers so you can put another click on the top of it.

NOTE: The click comes with stacking headers which allow you to combine it with other clicks more easily by using just one mikroBUS™ socket.

ATAES132A features

The ATAES132A is a high-security, Serial Electrically-Erasable and Programmable Read-Only Memory (Serial EEPROM) providing both authentication and confidential nonvolatile data storage capabilities. Access restrictions for the 16 user zones are independently configured, and any key can be used with any zone. In addition, keys can be used for standalone authentication.



- 32Kb Standard Serial EEPROM Memory
- AES Algorithm with 128-bit Keys
- AES-CCM for Authentication
- Guaranteed Unique Die Serial Number
- Secure Storage for up to Sixteen 128-bit Keys


The AES-128 cryptographic engine operates in AES-CCM mode to provide authentication, stored data encryption/decryption, and Message Authentication Codes. Data encryption/decryption can be performed for internally stored data or for small external data packets, depending upon the configuration. Data encrypted by one ATAES132A device can be decrypted by another, and vice versa.

Specifications

| | |
|------------------|--|
| Type | EEPROM |
| On-board modules | ATAES132A - a cryptographic coprocessor with secure hardware-based key storage from Microchip |
| Key Features | Crypto element device with secure hardware-based key storage, 512 bit OTP (One Time Programmable) Bits for Fixed Information |
| Interface | I2C,SPI |
| Input Voltage | 3.3V or 5V |
| Click board size | M (42.9 x 25.4 mm) |

Pinout diagram

This table shows how the pinout on **Secure 2 click** corresponds to the pinout on the mikroBUS™ socket (the latter shown in the two middle columns).

| Notes | Pin |  mikro™ BUS | | | | Pin | Notes |
|------------------------|--------------|---|------|-----|----|-------------|----------------|
| | | 1 | AN | PWM | 16 | | |
| | NC | 1 | AN | PWM | 16 | NC | |
| | NC | 2 | RST | INT | 15 | AUTH | Auth signaling |
| Chip select | CS | 3 | CS | TX | 14 | NC | |
| SPI clock | SCK | 4 | SCK | RX | 13 | NC | |
| Slave data out for SPI | SDO | 5 | MISO | SCL | 12 | SCL | I2C clock |
| Slave data in for SPI | SDI | 6 | MOSI | SDA | 11 | SDA | I2C data |
| Power supply | +3.3V | 7 | 3.3V | 5V | 10 | +5V | Power supply |
| Ground | GND | 8 | GND | GND | 9 | GND | Ground |

Programming

Code examples for Secure 2 click, written for MikroElektronika hardware and compilers are available on Libstock.

Code snippet

The following code snippet shows functions that will lock the configuration zone of the device and then output a generated random number.

```
01 //Configuration zone locking
02     if (aes132m_execute(AES132_LOCK, 0x02, 0x0000, 0x0000,
03                         0, 0, 0, 0, 0, 0, 0, 0, 0, txBuffer, rxBuffer)
04                         == AES132_FUNCTION_RETCODE_SUCCESS)
05     {
06         LOG( "rnrn Configuration zone locked! " );
07     }
08     else
09     {
10         LOG( "rnrn Configuration zone locking failed" );
11         LOG( " or it is already locked." );
12     }
13     memset (txBuffer, 0, 84);
14     memset (rxBuffer, 0, 36);
15 /*
16     Fourth test - Random number generator
17     Will only return 0xA5 unless config zone is locked first.
18 */
19     if (aes132m_execute(AES132_RANDOM, 0x00, 0x0000, 0x0000,
20                         0, 0, 0, 0, 0, 0, 0, 0, 0, txBuffer, rxBuffer)
21                         == AES132_FUNCTION_RETCODE_SUCCESS)
22     {
23         LOG( "rnrn Generated random number: " );
24         outputHex (&rxBuffer[2], 16);
25     }
26     else LOG( "rnrn Random number generation failed..." );
27
28     memset (txBuffer, 0, 84);
29     memset (rxBuffer, 0, 36);
```