AN12445

Asymmetric Cryptographic Accelerator CASPER

Rev. 5 — 20 September 2023 | Application note

Document information

Information	Content
Keywords	AN12445, CASPER, cryptographic accelerator, security feature, asymmetric cryptography, LPC55, LPC55Sxx
Abstract	This application note introduces CASPER on LPC5500 series security devices such as LPC55S6x, LPC55S2x, LPC55S1x, and LPC55S0x.

1 Introduction

The cryptographic accelerator and signaling processing engine with RAM-sharing (CASPER) peripheral provides acceleration to asymmetric cryptographic algorithms and certain signal processing algorithms.

For specific operations, the accelerator is faster, more memory efficient, and uses less energy compared to a general-purpose CPU. It accomplishes the resource expensive tasks of large-scale mathematical computations by combining speed and resource efficiency. While the accelerator is running, the processor can be idle or sleeping, or it can be performing other related or unrelated tasks. In addition, the system can run on a slower clock to reduce power consumption and improve energy efficiency.

This application note introduces CASPER on LPC5500 series security devices such as LPC55S6x, LPC55S2x, LPC55S1x, and LPC55S0x. Because the same CASPER is used by various devices, for the simplicity purpose, the examples in this document are based on the SDK for the leading part LPC55S69.

1.1 Acronyms

Table 1 lists the acronyms used in this document.

Table 1. Acronyms

Acronym	Meaning
CASPER	Cryptographic accelerator and signaling processing engine with RAM-sharing
RAM	Random access memory
SDK	Software development kit
RSA	Rivest-Shamir-Adleman
ECC	Elliptic curve cryptography
SIMD	Single-instruction multiple data
FFT	Fast Fourier transform
DCT	Discrete cosine transform
iFFT	Inverse Fast Fourier transform
AHB	Advanced high-performance bus
CP	Co-processor
MAC	Message authentication code
HW	Hardware
API	Application programming interface

1.2 Asymmetric cryptographic algorithms

The CASPER is intended to be a general-purpose engine that can be applied to a wide variety of cryptographic algorithms with a software package. The software includes asymmetric public keys (for example, RSA, elliptic-curve cryptography (ECC)), Diffie-Hellman key exchange, generator exponentials, and non-standard large number algorithms, among others.

1.3 Signal processing algorithms

The CASPER can be parameterized to perform most matrix operations, SIMD-based blending and scaling for graphics, and signal processing operations including FFT, DCT, and iFFT.

1.4 CASPER accelerator model

This section explains the model of the CASPER accelerator and the facilities provided by this accelerator. The block layout of how CASPER fits into a system is shown in Figure 1.

To improve the efficiency/speed of algorithms, the accelerator provides the following facilities:

- An AHB bus and Armv8-M Co-Processor (CP) interface used for loading information to operate.
- Fast shared memory access, allowing 128 bits to be moved at a time, as shown in Figure 1.
- Two 32×32 multipliers.
- A secondary bank of adders and registers for MAC-type operations (multiply and then accumulate).
- A mask facility that allows for side-channel countermeasure by never storing plain values in flops.
- A state machine that performs operations as required.

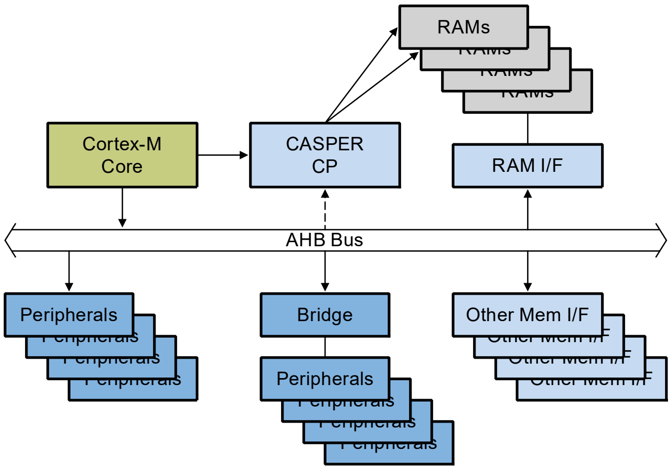


Figure 1. Showing how CASPER fits into a system

2 Approach

The approach to the CASPER accelerator is based on the fundamental properties shown in Figure 2.

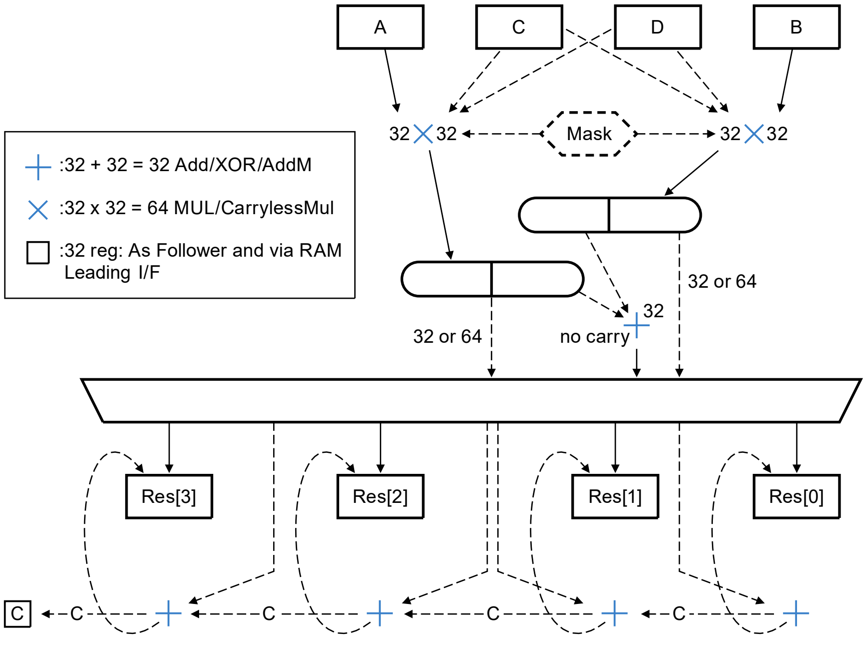


Figure 2. Block diagram of CASPER accelerator

These fundamental properties for the CASPER accelerator are listed as follows:

- The two multipliers receive their input from a set of four data registers (A/B/C/D) of 32 bits each. For side-channel protection, the multipliers can apply an XOR mask.
- A group of four result registers `Res[3]/Res[2]/Res[1]/Res[0]` can be used with four adders, and can also perform Add-Mask and XOR operations.
- Special access is provided to two or four RAMs (up to 8 kB) in parallel.
  - To allow the application to access the RAMs normally and at any time, the block uses a RAM interface that also supports AHB.
  - The AHB bus sees pairs of RAMs as interleaved, that is, one pair contains even words and the other pair contains odd words. The accelerator, on the other hand, sees them separately, enabling one-step access to 64-bit word pairs.
  - The block can access two or four banks simultaneously, allowing for two or four operations in parallel, that is, 64 bits or 128 bits at a time.
- Two control words (not shown in Figure 2) are used to launch the accelerator.
- For unmasking ABCD and masking output for side channel protection, an optional mask is registered for XOR mask creation.
- Two multipliers, each with a 64-bit output and supporting 32 bit × 32 bit, are used.
- Four adders using ADD and XOR.
- A carry bit (C) is used when a full-sum is performed.

3 Operations

From the fundamental operations, some larger operations are constructed, called modes. These modes can operate on up to 128 bits of the input registers and act as the lowest level of functions called as an API function. Larger computations can be constructed from these modes for efficient computation.

3.1 Modes

The following operation modes are supported:

- 64 bit × 64 bit: (MUL)
- 64 bit × 64 bit + 64 bit × 64 bit: (MUL+ADD)
- 64 bit + 64 bit: (ADD)
- 64 bit - 64 bit: (SUB)
- 64 bit ^ 64 bit: (XOR)
- 32 bit >>: (Shift right)
- 32 bit <<: (Shift left)
- Others: Copy, Remark, Fill, ZERO, Compare

3.2 Internal steps and flow by two example modes

Commonly used operations are the multiplication of two 64-bit values (MUL) and a double multiplication of 64-bit values followed by addition (MUL + ADD). The fundamental operations only include multiplication of 32-bit registers, therefore 64-bit multiplication is constructed from several 32-bit multiplications and additions.

3.2.1 MUL (64 bit × 64 bit = 128 bit)

The steps for MUL (64 bit × 64 bit = 128 bit) are as follows:

1. Read ABCD
2. Step 1:
   a. Compute DB and DA, sum = DBH+DAL
   b. Res[0] = DBL
   c. Res[1] = Sum
   d. Res[2] = DAH
   e. Res[3] = 0
   Figure 3 shows the Step 1 flow for 64 bit × 64 bit.

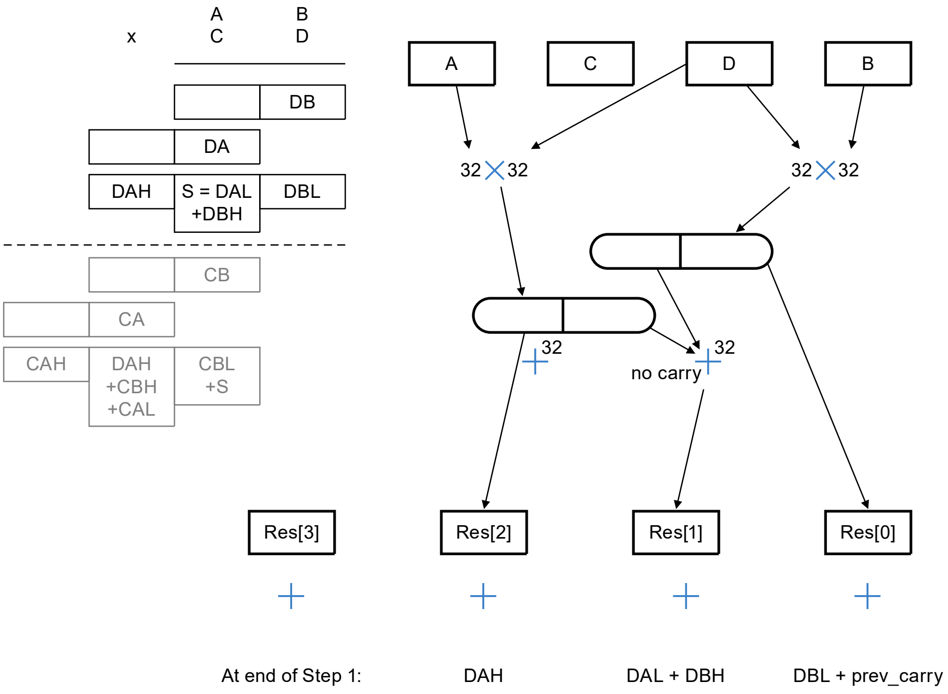


Figure 3. Step 1 flow for 64 bit × 64 bit

3. Step 2:
    a. Compute CB and CA, sum = CBH+CAL
    b. Res[1] += CBL (generate carry bit)
    c. Res[2] += Sum + carry bit (generate carry bit)
    d. Res[3] = CAH + carry bit
    Figure 4 shows the Step 2 flow for 64 bit × 64 bit

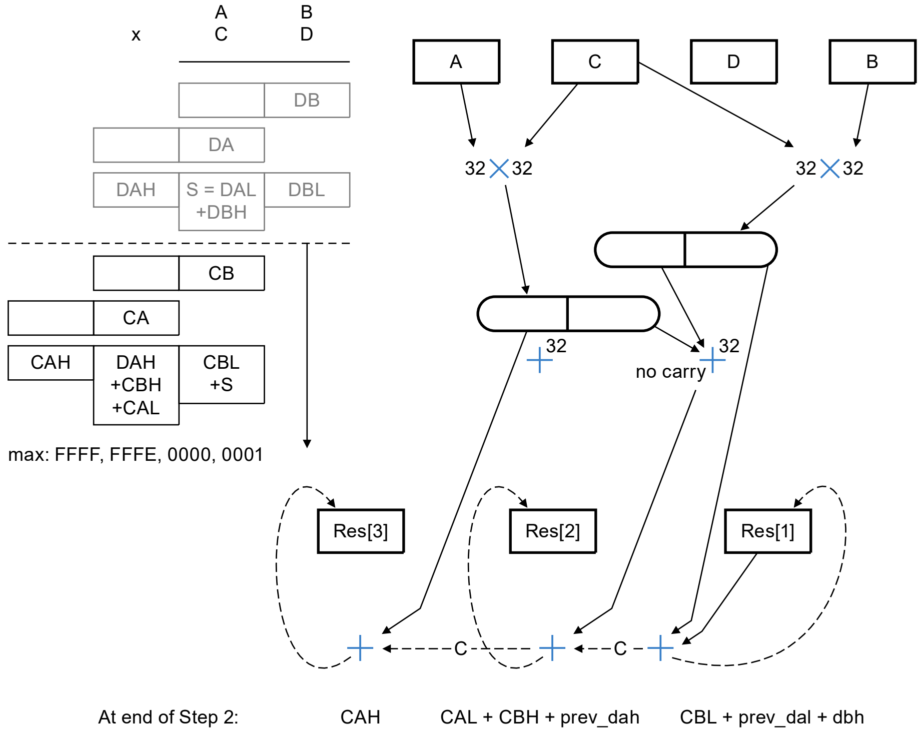


Figure 4. Step 2 flow for 64 bit × 64 bit

4. Step 3:
   a. Write Res[3:0]
   b. Done

3.2.2 MUL + ADD (64 bit × 64 bit + 64 bit × 64 bit)

The steps for MUL + ADD (64 bit × 64 bit + 64 bit × 64 bit) are as follows:

1. Read ABCD
2. Step 1:
   a. Compute DB and DA, sum = DBH+DAL
   b. Res[0] += DBL (generate carry bit)
   c. Res[1] += Sum + carry bit (generate carry bit)
   d. Res[2] += DAH + carry bit
   e. Res[3] += Carry bit
   Figure 5 shows the Step 1 flow of (64 bit × 64 bit + 64 bit × 64 bit).

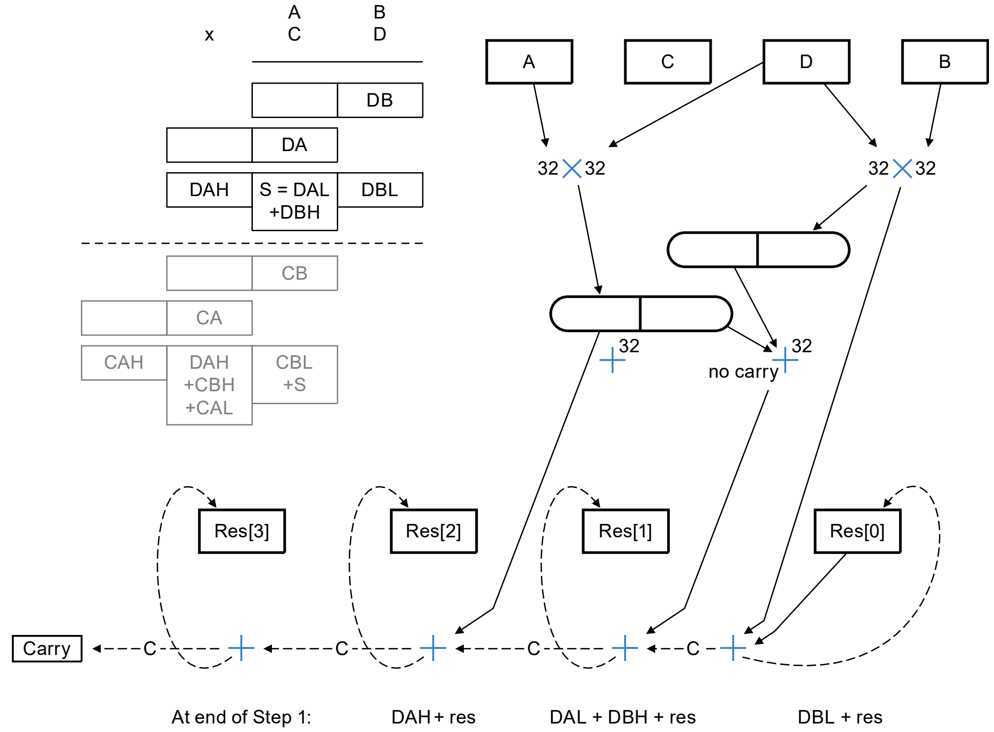


Figure 5. Step 1 flow of (64 bit × 64 bit + 64 bit × 64 bit)

3. Step 2:
   a. Compute CB and CA, sum = CBH+CAL
   b. Res[1] += CBL (generate carry bit)
   c. Res[2] += Sum + carry bit (generate carry bit)
   d. Res[3] += CAH + carry bit
   Figure 6 shows the Step 2 flow of (64 bit × 64 bit + 64 bit × 64 bit).

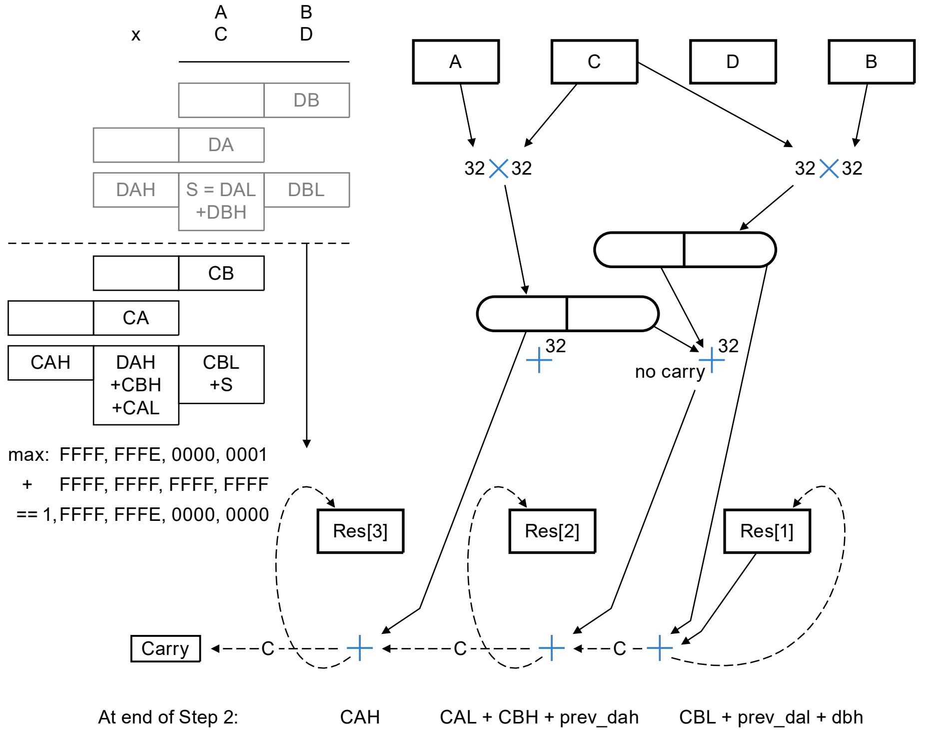


Figure 6. Step 2 flow of (64 bit × 64 bit + 64 bit × 64 bit)

4. Step 3:
   a. Write Res[3:0]
   b. Done

4 RAM interface

The RAM model is set up to support two and four RAMs, as shown in Figure 7. This indicates that the accelerator has simultaneous access to two or four banks, allowing for two or four parallel accesses to those RAMs and up to 128 bits of reading and writing. However, the AHB bus only sees one 32-bit access.

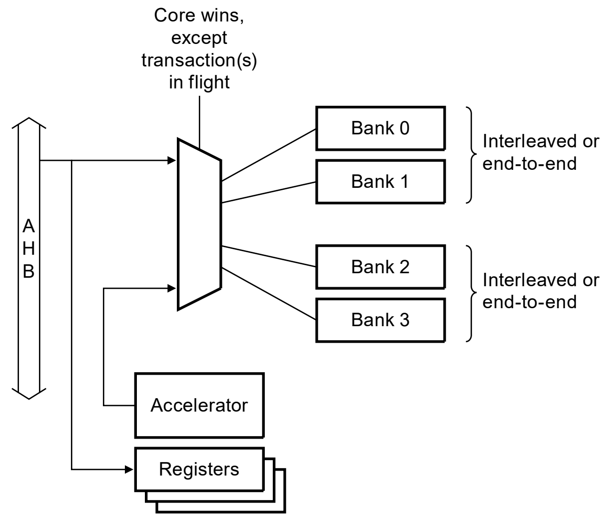


Figure 7. RAM in system with accelerator

5 Performance numbers

The CASPER accelerator is about several times faster than a pure multiplier for crypto-graphic purposes. Actual speed for various uses varies based on the algorithm, number of RAMs, whether interleaved, and how the software has placed its buffers.

The performance between CASPER accelerated and pure software implementation on LPC55S69 is shown in Figure 8.

Operation	System clock:150MHz IDE: IAR8.32, Optimizations: high-speed-no size constraints	SW only		CASPER accelerated		Improvement(times)	
		RAM	Flash	RAM	Flash	RAM	Flash
Signing	ECDSA-secp256r1(ms/sign)	136.43	333.33	76.92	142.86	1.77	2.33
Verification	ECDSA-secp256r1(ms/verify)	250.00	598.80	81.10	149.93	3.08	3.99
Key exchange	ECDHE-secp256r1(ms/handshake)	250.00	500.00	136.43	250.00	1.83	2.00
Key exchange	ECDH-secp256r1(ms/handshake)	136.43	300.30	71.43	130.38	1.91	2.30
Signing	RSA-1024(ms/private)	130.38	250.00	130.38	272.48	-	-
Verification	RSA-1024(ms/public)	4.24	8.90	1.31	1.81	3.24	4.93
Signing	RSA-2048(ms/private)	598.80	1000.00	598.80	1000.00	-	-
Verification	RSA-2048(ms/public)	15.54	31.92	4.14	5.03	3.76	6.35

Figure 8. Performance comparison for asymmetric cryptographic algorithms implemented by software and CASPER

Remarks:

- The CASPER performance is similar on other security parts of the LPC5500 series, for example, LPC55S2x, LPC55S1x, and LPC55S0x.
- IDE version: IAR8.32.2

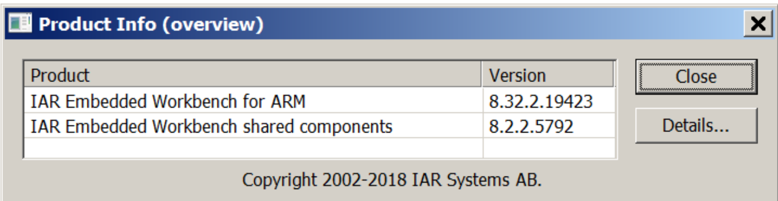


Figure 9. IAR version

- Optimizations:
  1. Select **High** under the **Level** option, as shown in Figure 10.
  2. Select **Speed** from the dropdown and select **No size constraints**.

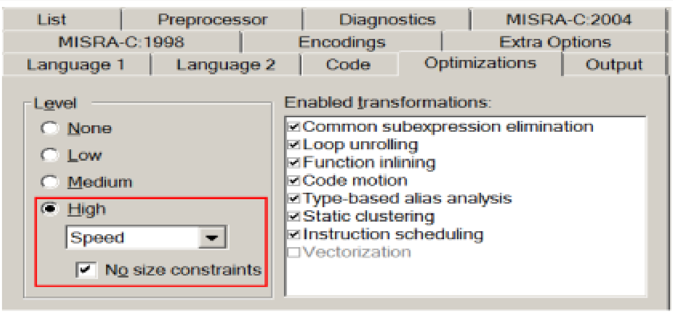


Figure 10. IAR optimizations

- SDK version: 2.6.3 (2019-10-11)
  SDK tag: `REL_SDK_NIOBE4_2.6.3_RFP3.0_RC2_3`
- System clock: 150 MHz

- Example version: Release version of project.
- "-" stands for no CASPER accelerated. Hardware and software efficiency are the same.
- If running in the flash, `LPC55S69_cm33_core0_flash.icf` must be installed.
  If running in the RAM, `LPC55S69_cm33_core0_ram.icf` must be installed.
- Board: LPCXpresso55S69, LPC55S69-EVK
- Silicon revision: 1B

6 SDK implementations

The SDK package includes the CASPER example.

After downloading the latest version of the SDK from the NXP website, open the CASPER example at the following location:

```
SDK_2.6.3_LPCXpresso55S69\boards\lpcxpresso55s69\driver_examples\casper\cm33_core0
```

Compile the project and open the `casper.c` file. Some application functions can be divided into two applications as follows:

- Modular exponentiation algorithm
- Elliptic curve multiplication (`secp256r1`, `secp384r1`, and `secp521r1`)

6.1 Modular exponentiation algorithm

Modular exponentiation (ModExp) is a computation where exponentiation is combined with a modular reduction. It is useful in computer science, especially in the field of public-key cryptography.

The formula in Equation 1 explains how to verify a signature by using a public key (including E and N):

$$Plaintext = Signature^{E} mod \;\; N \tag{1}$$

The function code example is as follows:

```
/* ModExp test */
CASPER_ModExp(CASPER , (void *)signature0, (void *)pubkey0, sizeof(plaintext) / sizeof(uint32_t),
 pub_e , plaintext);
TEST_ASSERT (memcmp (plaintext0, plaintext, sizeof(plaintext)) == 0);
PRINTF("ModExp Test pass. \r \n");
```

The implementation process includes a series of complicated data conversions. It is based on the classic ModExp algorithm, including Montgomery modular multiplication and so on. Finally, the algorithm uses basic multiply, addition, and subtraction algorithms. These algorithms can be achieved by CASPER.

Some basic application codes are as follows:

```
Accel_SetABCD_Addr(CA_MK_OFF(&v[0]), CA_MK_OFF(u));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_wordlen / 2 - 1, kCASPER_OpmMu16464NoSum,
 CA_MK_OFF(&w_out[0])));
Accel_done();

Acce]_SetABCD_Addr(CA_MK_OFF(&v[j]}), CA_MK_OFF(u));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_wordlen / 2 - 1, kCASPER_OpmMu16464Sum,
 CA_MK_OFF(&w_out[j])));
Accel_done();

Accel_SetABCD_Addr(CA_MK_OFF(Nmod), 0);
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpSub64, CA_MK_OFF(Rp)));
Accel_done();

carry = GET_DWORD(&w64[N_dwordlen]);
```

```
Accel_SetABCD_Addr(CA_MK_OFF(&b64[i]), CA_MK_OFF(a64));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpmMul6464FullSum,
 CA_MK_OFF(w64)));
Accel_done();
```

Due to the accelerator function of CASPER, RSA signature verification is fast. In the functions, there are some CASPER operations. These operations correspond to the operation modes described in Section 3 and are as follows:

```
typedef enum _casper_operation
{
  kCASPER_OpMul6464NoSum = 0x01, /*! Walking 1 or more of J Loop, doing r=a*b using 64x64=128*/
  kCASPER_OpMul6464Sum = 0x02, /*! Walking 1 or more of J loop, doing c, r=r+a*b using 64x64=128,
 but assume inner j Loop*/
  kCASPER_OpMul6464FullSum = 0x03, /*! Wa1Lking 1 or more of J loop, doing c,r=r+a*b using
 64x64=128, but sum all of w. */
  kCASPER_OpMul6464Reduce = 0x04, /*! Wa1Lking 1 or more of J loop, doing c, r[-1]=r+a*b using
 64x64=128, but skip 1st write*/
  kCASPER_OpAdd64 = 0x08, /*! Walking add with off_AB, and in/out off_RES doing c,r=r+a+c using
 64+64=65*/
  kCASPER_OpSub64 = 0x09, /*! Walking subtract with off_AB, and in/out off _RES doing r=r-a using
 64-64=64, with last borrow implicit if any*/
  kCASPER_OpDouble64 = 0x0A, /*! Walking add to self with off_RES doing c,r=r+r+c using 64+64=65*/
  kCASPER_OpXor64 = 0x0B, /*! Walking XOR with off_AB, and in/out off_RES doing r=r^a using
 64^64=64*/
  kCASPER_OpShiftLeft32 = 0x10, /*! Walking shift left doing r1,r=(b*D)/r1, where D is 2^amt and is
 loaded by app (off_CD not used)*/
  kCASPER_OpShiftRight32 = 0x11, /*! Walking shift right doing r,r1=(b*D)/r1, where Dis 2^(32-amt)
 and is loaded by app (off_CD not used) and off_RES starts at MSW*/
  kCASPER_OpCopy = 0x14, /*! Copy from ABoff to resoff, 64b at a time*/
  kCASPER_OpRemask = 0x15, /*! Copy and mask from ABoff to resoff, 64b at a time*/
  kCASPER_OpCompare = 0x16, /*! Compare two arrays, running all the way to the end*/
  kCASPER_;OpCompareFast = 0x17, /*! Compare two arrays, stopping on 1st !=*/
} casper_operation_t;
```

6.2 Elliptic curve multiplication

The functions perform ECC point single scalar multiplication `[resX; resY] = scalar * [X; Y]` and ECC point double scalar multiplication `[resX; resY] = scalar1 * [X1; Y1] + scalar2 * [X2; Y2]` for the curves secp256r1, secp384r1, and secp521r1. They are the basis of the ECC.

The function code example that performs the ECC single scalar multiplication of secp256r1 is as follows:

```
CASPER_ECC_SECP256R1_Mul(CASPER, X1, Y1, &test_ecmulans256[0][0], &test_ecmulans256[0][8],
 test_ecmulscalar256[i]);
```

The double scalar multiplication is as follows:

```
CASPER_ECC_SECP256R1_MulAdd(CASPER, c3, c4, &test_ecddoublemul_base256[0][0],
&test_ecddoublemul_base256[0][256U / 32U], &test_ecddoublemul_scalars256[i][0],
&test_ecddoublemul_base256[1][0], &test_ecddoublemul_base256[1][256U / 32U],
&test_ecddoublemul_scalars256[i][256U / 32U]);
```

Similar to ModExp, the fundamental implementation of ECC multiplication is based on the CASPER API and is as follows:

```
/* we are reducing, so the 1st [0th] 64 bit value product is tossed, but we*/
/* need its carry. We let the accel do this separately - really need a mode to */
/* do this "reduce" since it is natural */
carry = GET _DWORD(&w64[ N_dwordlen]);
Accel_SetABCD_Addr(CA_MK_,OFF(m64), CA_MK_OFF(&N64[0]));
A1c,cel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpMul6464FullSum,
 CA_MK_OFF(&w64[0])));
Accel_done();
Ca1rry = (GET_DWORD(&w64[N_dwordlen]) < carry);
```

```
Accel_SetABCD_Addlr (CA_MK_OFF(&w64[1], 0);
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpCopy, CA_MK_OFF(&w64[0])));
Accel_done();
SET_DWORD(&w64[N_dwordlen], (GET_DWORD(&w64[N_dwordlen + 1]) + carry));
```

Finally, after running the example code, you can get the print string on the CommAssistant as follows:

```
ModExp Test pass.
Casper ECC Demo P256
Round: 0
Round: 1
Round: 2
Round: 3
Round: 4
Round: 5
Round: 6
Round: 7
All EC scalar multiplication tests were successful.
Round:
Round: 0
Round: 1
Round: 2
Round: 3
Round: 4
Round: 5
Round: 6
Round: 7
All EC double scalar multiplication tests were successful.
```

7 CASPER usage in mbedTLS

This section describes CASPER usage in mbedTLS.

7.1 Introduction

MbedTLS has been modified using the port capabilities ("_alt") to replace, among others, some of the software implementations of asymmetric algorithms. This modification is done by directly accessing the CASPER driver for accelerating the multiply and multiply add operations.

7.2 Running the demo

The location of the mbedTLS example in the SDK package is as follows:

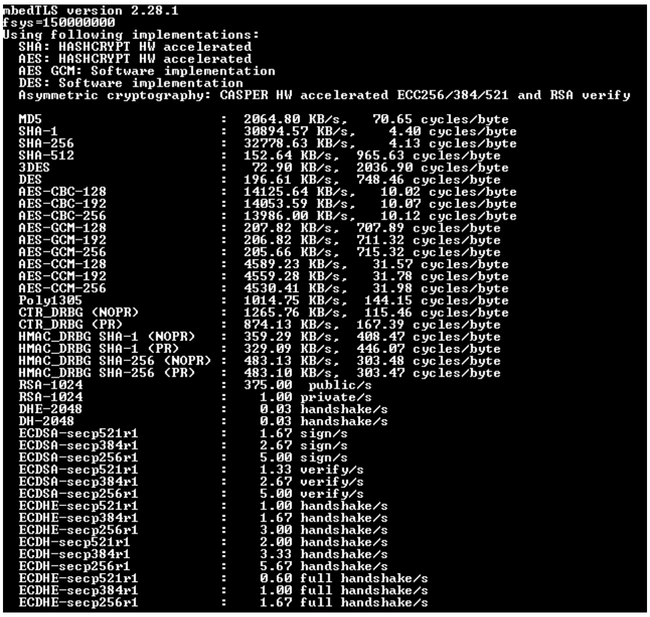
```
SDK_2.6.3_LPCXpresso55S69\boards\lpcxpresso55s69\mbedtls_examples\mbedtls_benchmark\cm33_core0
```

The demo application performs a benchmark of the cryptographic algorithm, which includes the following:

- Symmetric and asymmetric encryption
- CASPER hardware accelerated in the RSA-1024 encryption
- ECDSA-secp256r1, ECDSA-secp384r1, and ECDSA-secp521r1 signing and verification
- ECDHE-secp256r1, ECDHE-secp384r1, and ECDHE-secp521r1 key exchange
- ECDH-secp256r1, ECDH-secp384r1, and ECDH-secp521r1 key exchange

After downloading and running the code, the CASPER accelerator is used. The output of the accelerated benchmark is shown in Figure 11.

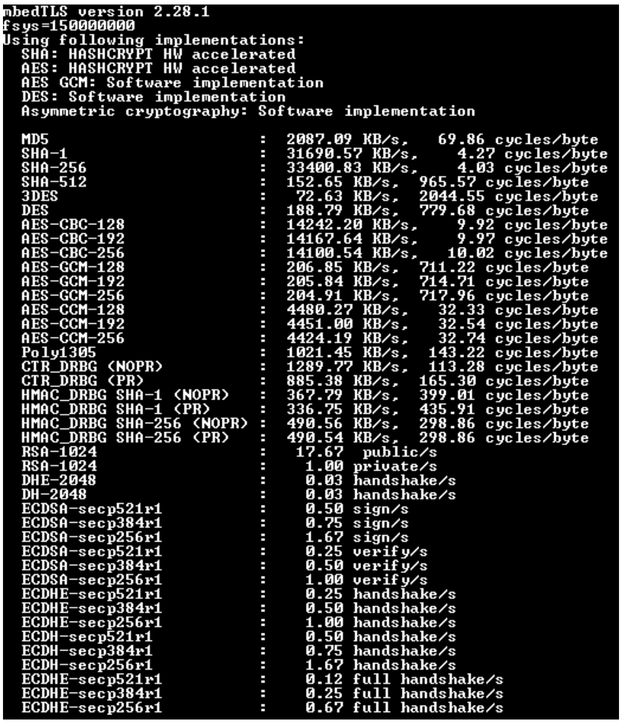
If `FSL_FEATURE_SOC_CASPER_COUNT` is set to **0** in the `device\LPC55S69_cm33_core0_features.h`, it changes to the software implementation. The result is shown in Figure 12.



```
mbedTLS version 2.28.1
fsys=150000000
Using following implementations:
  SHA: HASHCRYPT HW accelerated
  AES: HASHCRYPT HW accelerated
  AES GCM: Software implementation
  DES: Software implementation
  Asymmetric cryptography: CASPER HW accelerated ECC256/384/521 and RSA verify

  MD5                      :  2064.80 KB/s,   70.65 cycles/byte
  SHA-1                    :  30894.57 KB/s,    4.40 cycles/byte
  SHA-256                  :  32778.63 KB/s,    4.13 cycles/byte
  SHA-512                  :  152.64 KB/s,  965.63 cycles/byte
  3DES                     :   72.90 KB/s,  2036.90 cycles/byte
  DES                      :  196.61 KB/s,  748.46 cycles/byte
  AES-CBC-128              :  14125.64 KB/s,   10.02 cycles/byte
  AES-CBC-192              :  14053.59 KB/s,   10.07 cycles/byte
  AES-CBC-256              :  13986.00 KB/s,   10.12 cycles/byte
  AES-GCM-128              :  207.82 KB/s,  707.89 cycles/byte
  AES-GCM-192              :  206.82 KB/s,  711.32 cycles/byte
  AES-GCM-256              :  205.66 KB/s,  715.32 cycles/byte
  AES-CCM-128              :  4589.23 KB/s,   31.57 cycles/byte
  AES-CCM-192              :  4559.28 KB/s,   31.78 cycles/byte
  AES-CCM-256              :  4530.41 KB/s,   31.98 cycles/byte
  Poly1305                 :  1014.75 KB/s,  144.15 cycles/byte
  CTR_DRBG (NOPR)          :  1265.76 KB/s,  115.46 cycles/byte
  CTR_DRBG (PR)            :  874.13 KB/s,  167.39 cycles/byte
  HMAC_DRBG SHA-1 (NOPR)   :  359.29 KB/s,  408.47 cycles/byte
  HMAC_DRBG SHA-1 (PR)     :  329.09 KB/s,  446.07 cycles/byte
  HMAC_DRBG SHA-256 (NOPR) :  483.13 KB/s,  303.48 cycles/byte
  HMAC_DRBG SHA-256 (PR)   :  483.10 KB/s,  303.47 cycles/byte
  RSA-1024                 :  375.00  public/s
  RSA-1024                 :    1.00 private/s
  DHE-2048                 :    0.03 handshake/s
  DH-2048                  :    0.03 handshake/s
  ECDSA-secp521r1          :    1.67 sign/s
  ECDSA-secp384r1          :    2.67 sign/s
  ECDSA-secp256r1          :    5.00 sign/s
  ECDSA-secp521r1          :    1.33 verify/s
  ECDSA-secp384r1          :    2.67 verify/s
  ECDSA-secp256r1          :    5.00 verify/s
  ECDHE-secp521r1          :    1.00 handshake/s
  ECDHE-secp384r1          :    1.67 handshake/s
  ECDHE-secp256r1          :    3.00 handshake/s
  ECDH-secp521r1           :    2.00 handshake/s
  ECDH-secp384r1           :    3.33 handshake/s
  ECDH-secp256r1           :    5.67 handshake/s
  ECDHE-secp521r1          :    0.60 full handshake/s
  ECDHE-secp384r1          :    1.00 full handshake/s
  ECDHE-secp256r1          :    1.67 full handshake/s
```

Figure 11. CASPER hardware accelerated result



```
mbedTLS version 2.28.1
fsys=150000000
Using following implementations:
  SHA: HASHCRYPT HW accelerated
  AES: HASHCRYPT HW accelerated
  AES GCM: Software implementation
  DES: Software implementation
  Asymmetric cryptography: Software implementation

  MD5                      :  2087.09 KB/s,   69.86 cycles/byte
  SHA-1                    :  31690.57 KB/s,    4.27 cycles/byte
  SHA-256                  :  33400.83 KB/s,    4.03 cycles/byte
  SHA-512                  :  152.65 KB/s,  965.57 cycles/byte
  3DES                     :   72.63 KB/s,  2044.55 cycles/byte
  DES                      :  188.79 KB/s,  779.68 cycles/byte
  AES-CBC-128              :  14242.20 KB/s,    9.92 cycles/byte
  AES-CBC-192              :  14167.64 KB/s,    9.97 cycles/byte
  AES-CBC-256              :  14100.54 KB/s,   10.02 cycles/byte
  AES-GCM-128              :  206.85 KB/s,  711.22 cycles/byte
  AES-GCM-192              :  205.84 KB/s,  714.71 cycles/byte
  AES-GCM-256              :  204.91 KB/s,  717.96 cycles/byte
  AES-CCM-128              :  4480.27 KB/s,   32.33 cycles/byte
  AES-CCM-192              :  4451.00 KB/s,   32.54 cycles/byte
  AES-CCM-256              :  4424.19 KB/s,   32.74 cycles/byte
  Poly1305                 :  1021.45 KB/s,  143.22 cycles/byte
  CTR_DRBG (NOPR)          :  1289.77 KB/s,  113.28 cycles/byte
  CTR_DRBG (PR)            :  885.38 KB/s,  165.30 cycles/byte
  HMAC_DRBG SHA-1 (NOPR)   :  367.79 KB/s,  399.01 cycles/byte
  HMAC_DRBG SHA-1 (PR)     :  336.75 KB/s,  435.91 cycles/byte
  HMAC_DRBG SHA-256 (NOPR) :  490.56 KB/s,  298.86 cycles/byte
  HMAC_DRBG SHA-256 (PR)   :  490.54 KB/s,  298.86 cycles/byte
  RSA-1024                 :   17.67  public/s
  RSA-1024                 :    1.00 private/s
  DHE-2048                 :    0.03 handshake/s
  DH-2048                  :    0.03 handshake/s
  ECDSA-secp521r1          :    0.50 sign/s
  ECDSA-secp384r1          :    0.75 sign/s
  ECDSA-secp256r1          :    1.67 sign/s
  ECDSA-secp521r1          :    0.25 verify/s
  ECDSA-secp384r1          :    0.50 verify/s
  ECDSA-secp256r1          :    1.00 verify/s
  ECDHE-secp521r1          :    0.25 handshake/s
  ECDHE-secp384r1          :    0.50 handshake/s
  ECDHE-secp256r1          :    1.00 handshake/s
  ECDH-secp521r1           :    0.50 handshake/s
  ECDH-secp384r1           :    0.75 handshake/s
  ECDH-secp256r1           :    1.67 handshake/s
  ECDHE-secp521r1          :    0.12 full handshake/s
  ECDHE-secp384r1          :    0.25 full handshake/s
  ECDHE-secp256r1          :    0.67 full handshake/s
```

Figure 12. Software implementation result

7.3 Call stacks of implementations

If `FSL_FEATURE_SOC_CASPER_COUNT` is set to **1** in the `device\LPC55S69_cm33_core0_features.h` and debugging is done step by step, function calls are as follows:

1. The implementation of **RSA-1024: 196.33 public/s**.
   `CASPER_ModExp()` is the CASPER low driver API and is used in the RSA-1024 encryption, as shown in **Call Stack** in Figure 13.

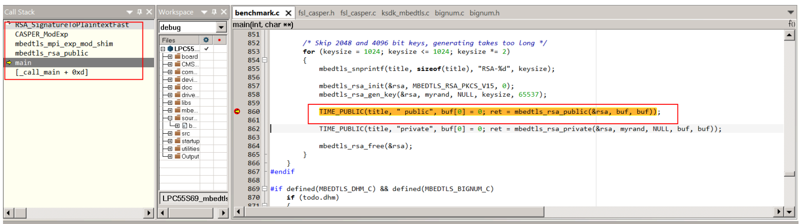


Figure 13. RSA-1024 encryption implementation

2. The implementation of **ECDSA-secp256r1: 3.33 sign/s**.
   `CASPER_ECC_SECP256R1_Mul ()` is the CASPER low driver API and is used in the ECDSA-secp256r1 signing, as shown in **Call Stack** in Figure 14.

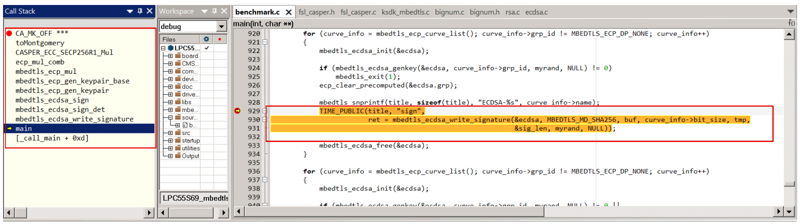


Figure 14. ECDSA-secp256r1 signing implementation

3. The implementation of **ECDSA-secp256r1: 3.33 verify/s**.
   `CASPER_ECC_SECP256R1_MulAdd ()` is the CASPER low driver API and is used in the ECDSA-secp256r1 verification, as shown in **Call Stack** in Figure 15.

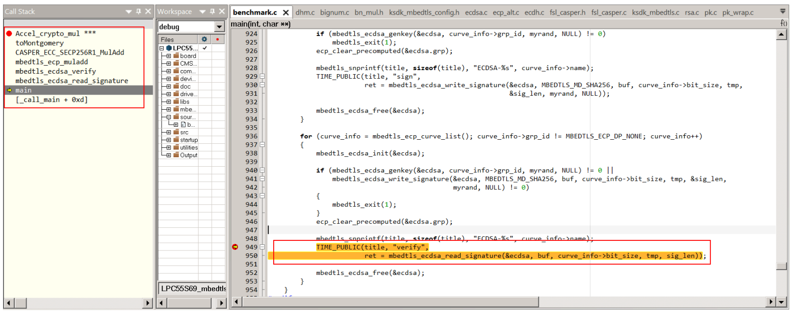


Figure 15. ECDSA-secp256r1 verification implementation

4. The implementation of **ECDHE-secp256r1: 2.00 handshake/s**.
   `CASPER_ECC_SECP256R1_Mul ()` is the CASPER low driver API and is used in the ECDHE-secp256r1 key exchange, as shown in **Call Stack** in Figure 16.

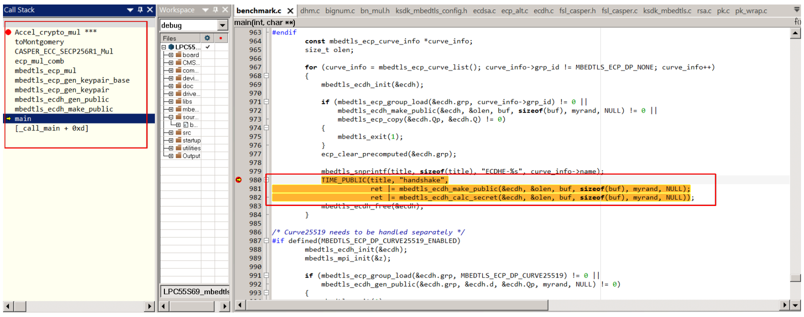


Figure 16. ECDHE-secp256r1 key exchange implementation

5. The implementation of **ECDH-secp256r1: 3.67 handshake/s**.
   `CASPER_ECC_SECP256R1_Mul()` is the CASPER low driver API and is used in the ECDH-secp256r1 key exchange, as shown in **Call Stack** in Figure 17.

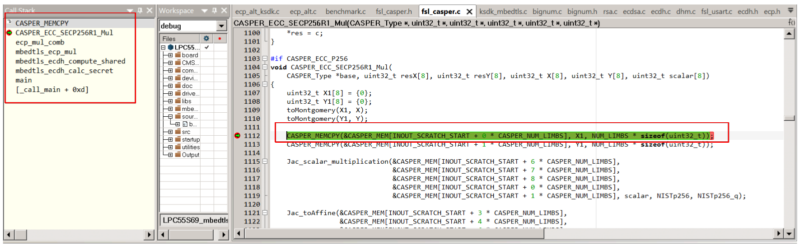


Figure 17. ECDH-secp256r1 key exchange implementation

8 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

9 Revision history

Table 2 summarizes the revisions done to this document.

Table 2. Revision history

Revision history	Release date	Description
5	20 September 2023	• Multiple editorial changes • Figures updated to svg format • Various codeblocks incorporated • Spelling and grammar improvements for the entire document • New topics and content added to Section 7 • Figure 11 updated
4	27 October 2020	Added LPC55S0x
3	7 January 2020	Include LPC55S2x/1x
2	23 October 2019	Parameter update
1	15 July 2019	General updates
0	20 April 2019	Initial public release

10 Legal information

10.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

10.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

IAR — is a trademark of IAR Systems AB.

Contents

Date of release: 20 September 2023
Document identifier: AN12445