AN3923

AES Cipher Modes Using MPLAB Harmony v3 for the SAM E54

Introduction

The Advanced Encryption Standard (AES), also known as the Rijndael algorithm, is a symmetric block cipher algorithm that can process data blocks of 128 bits, using a three key lengths of 128, 192, and 256 bits in compliance to the NIST specification (FIPS PUB. 197). The symmetric-key algorithm requires the same key for both encryption and decryption. AES can be used with various modes of operation, designed specifically to use with block cipher algorithms as recommended in the NIST Special Publication 800-38A/B/C/D.

The SAM D5x/E5x family of devices support AES cryptography hardware. This document describes the AES implementation of the following four confidentiality modes of operation in the MPLAB® Harmony v3 framework for SAME54:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter (CTR)
- Galois Counter mode (GCM)

MPLAB Harmony v3 uses a Cryptographic library which is built on top of the third party wolfSSL library. The wolfCrypt portion of the wolfSSL library provides an interface for the configuration and management of the AES block for the device.

© 2021 Microchip Technology Inc.

Table of Contents

1. Glossary

The following terminology is used throughout this document.

AES	Advanced Encryption Standard.
Block Cipher	A cryptographic algorithm that works on a fixed size of data. AES transfers in blocks of 16 bytes.
CBC	Cipher Block Chaining.
Cipher Text (CT)	Encrypted data.
CTR	Counter.
ECB	Electronic Codebook.
GCM	Galois Counter Mode.
Decryption	The process of a confidentiality mode that transforms encrypted data into the original usable data.
Encryption	The process of a confidentiality mode that transforms usable data into an unreadable form.
FIPS	Federal information Processing Standard.
Forward Cipher Function	One of the two functions of the block cipher algorithm that is selected by the cryptographic key.
Initialization Vector (IV)	A data block that some modes of operation require as an additional initial input.
Inverse Cipher Function	The function that reverses the transformation of the forward cipher function when the same cryptographic key is used.
Output Block	A data block that is an output of either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Input Block	A data block that is an input to either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Plain Text (PT)	Usable data that is formatted as input to a mode.
S-box	Non-linear substitution table used in several byte substitution transformations and in the Key Expansion Routine to perform a one-for-one substitution of a byte value.

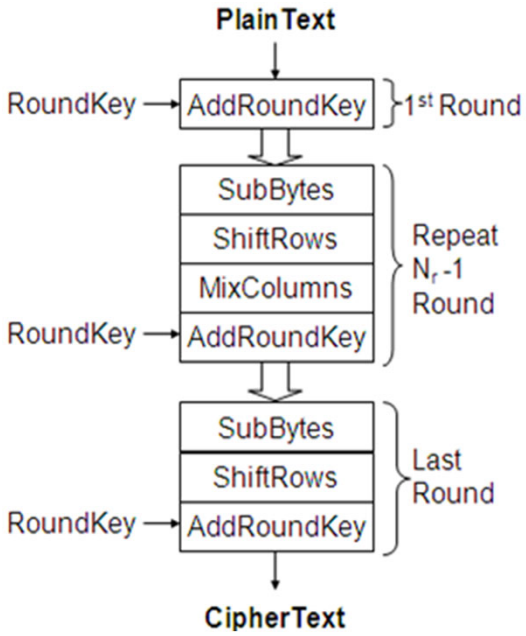
2. AES Overview

The Advanced Encryption Standard Algorithm (AES) implements the FIPS approved (FIPS Pub.197) cryptographic algorithm that can be used to protect electronic data. AES is a symmetric key algorithm that operates on a 128-bit block of input data for a specified number of times. The symmetric key means that same key is used for both encryption and decryption. The encryption process converts the data to an unintelligible form called ciphertext. The decryption process converts the data back to plaintext from the ciphertext. The length of the ciphertext is the same as that of the plaintext. The key size used for an AES encryption and decryption can be 128, 192, or 256 bits for a fixed input block size of 128 bits. The size of the key determines the number of rounds to be performed, and the duration the E54 hardware takes to conduct a block operation. The larger the key, the longer the hardware takes to conduct a block operation.

2.1 AES Encryption

The AES-128 encryption process involves 10 rounds of encryption along with an initial round for the 128-bit data encryption. To begin with, the 128-bit key is expanded into a set of eleven 128-bit round keys using the Key expansion routine. Each of these keys are used for the rounds, finally resulting in the cipher text output. The overall process involved in AES encryption is illustrated in the following figure.

Figure 2-1. AES Encryption



The steps involved in an AES Encryption algorithm are as follows:

1. Key Expansion – The AES key expansion routine gets the symmetric cipher key from the user, and generates a set of Round keys known as the Key Schedule. Each of these keys are used by each round, finally resulting in the ciphertext output.
2. Initial Round – The initial round in the AES Encryption comprises of the Add Round key step in which the plaintext is XOR-ed with the Cipher Key. In the process, each byte of the state is combined with the round key using bit-wise XOR.
3. Rounds – The next rounds go through a different byte-oriented transformation. A typical round consists of these following sub-processes:
   - Sub Byte - Byte substitution using S-box.
   - Shift Rows – Shifting rows of the state array by different offsets.

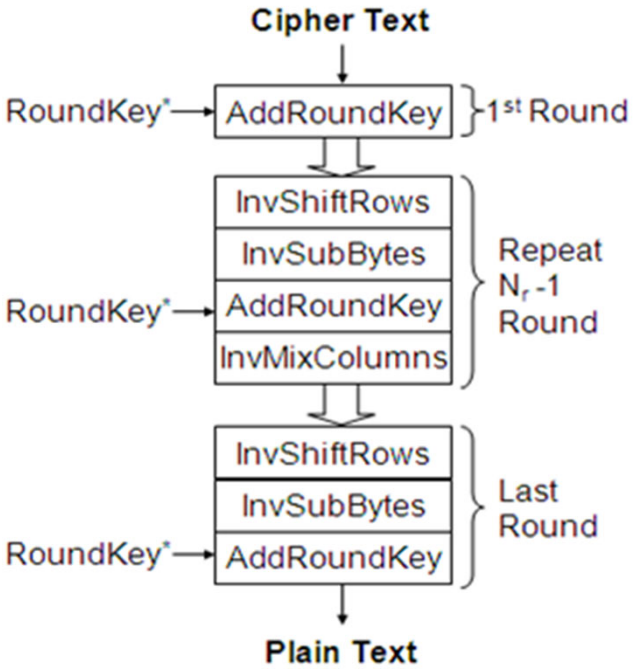
- Mix Columns – Mixing the data within each column.
- AddRoundKey – Adding Round key to the state.

4. Final Round – The final round (nth round) involves the same subprocess; however, it does not use the MixColumn process.

2.2 AES Decryption

The process of decryption of an AES ciphertext is the encryption process performed in the reverse order. The AES-128 decryption process involves a similar number of rounds as the AES-128 encryption process with corresponding inverse transformations.

Figure 2-2. AES Decryption



The steps involved in an AES decryption algorithm are explained as follows:

1. Initial Round – The initial round in the AES decryption comprises of the Add Round key step, which is the same as in the AES encryption process.
2. Rounds – The subsequent rounds go through an inverse transformation. The inverse transformations for the subsequent rounds are as follows:
   - Inverse ShiftRows – Inverse of ShiftRows transformation.
   - Inverse SubBytes – Inverse of the SubBytesTransformation using the inverse S-Box.
   - AddRoundKey – Same as in the encryption step as this involves only XOR operation between an input and output blocks.
   - Inverse MixColumn – Inverse of MixColumnstransformation.
3. Final Round – In the final round, the above steps are required, excluding the MixColumns step.

Note: The steps involved in AES encryption and decryption are handled by the hardware AES engine in the SAME54 device. The user does not need to use software for this process. For additional information on the AES standard, refer to the AES standard document FIPS PUB. 197.

3. AES Cipher Modes in MPLAB Harmony v3

3.1 Cryptographic Library

The crypto library of MPLAB Harmony v3 provides a cryptographic library with a convenient C- language interface. The crypto library offers many functions to perform encryption, decryption, hashing, authentication, and compression within the embedded application. AES, a symmetric block cipher, deploys the encryption and decryption APIs of the crypto library for various cipher modes, such as ECB, CBC, CTR, and GCM.

The crypto library lies on top of the wolfCrypt library of wolfSSL. The wolfCrypt cryptography engine is a lightweight crypto library written in ANSI 'C' and targeted for embedded, RTOS, and resource-constrained environments; primarily because of its small size, speed, and feature set. Microchip has an agreement with WolfSSL for royalty-free use of WolfSSL. The wolfSSL libraries can be used with no additional licensing charge when using Microchip products. More details regarding the license can be found here.

The Crypto repository contains the crypto module for the MPLAB Harmony v3 framework. The crypto module contains the MPLAB Harmony v3 APIs for cryptography, the SAM E54 driver for hardware cryptographic acceleration (other devices are also supported), and the deltas/patch for the wolfCrypt or wolfSSL library to use the hardware accelerators. The following files are associated with the AES crypto operations.

`configuration.h`	Configuration of cryptographic Library

Table 3-1. Cryptographic Library Files

`crypto.h`	Interface to the Crypto Library
`crypto.c`	Definitions of all Crypto library functions

Table 3-2. wolfCrypt Library Files

`aes.h`	Interface to Wolfcrypt Library functions
`crypt_aes_u2238.h`	Crypto Framework Library interface file for hardware AES
`crypt_aes_u2238.c`	Crypto Framework Library source for cryptographic functions

Notes:

1. The SAM E54 carries a hardware AES crypto engine. The wolfCrypt library supports hardware cryptography and acceleration. The `crypt_aes_u2238.h/c` file includes both interface and source functions for AES hardware acceleration.
2. The APIs of Crypto and wolfSSL or wolfCrypt libraries for AES can be found at Crypto Library Help and wolfCrypt Library Help.

3.2 AES Modes of Operation

NIST recommends different confidentiality modes of operation to use with an underlying symmetric key block cipher algorithm. The mode of operation is an algorithm that describes how to repeatedly apply a cipher's single-block operation to encrypt data larger than a block. The following modes are covered in this document.

- Electronic Code Book mode (ECB)
- Cipher Block Chaining mode (CBC)
- Counter mode (CTR)
- Galois Counter mode (GCM)

3.2.1 Electronic Code Book (ECB)

This is the simplest of all modes. In this mode, the input is divided into separate blocks of 128 bits. Each block is encrypted or decrypted independently. In ECB encryption, the cipher function is directly applied to each input

block resulting in a ciphertext. In ECB decryption, the inverse cipher function is applied to each block to retrieve the plaintext. It is not recommended to use ECB mode, as ECB mode will always encrypt the same plain text to the same ciphertext with the same key, providing an easier attack vector.

Table 3-3. Harmony APIs for ECB

Cryptographic Library APIs	`CRYPT_AES_ECB_Encrypt`
	`CRYPT_AES_ECB_Decrypt`
wolfcrypt Library APIs	`wc_AesEcbEncrypt`
	`wc_AesEcbDecrypt`

Note: The `CRYPT_AES_ECB` encrypt and `CRYPT_AES_ECB` decrypt functions are not automatically generated in the `crypto.h/c` files for this project. Refer to Things to Consider for additional information.

3.2.2 Cipher Block Chaining Mode (CBC)

In CBC mode, each block of plaintext is XOR-ed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector (IV) must be used in the first block. The IV is not secret, but must be unpredictable. For decryption, the inverse cipher is applied over the input ciphertext and the resulting output block is XOR-ed with IV to get the first block of plaintext. The CBC mode and the various methods of generation of the initialization vector is explained in FIPS Pub 800-38A.

Table 3-4. Harmony APIs for CBC

Cryptographic Library APIs	CRYPT_AES_CBC_Encrypt
	CRYPT_AES_CBC_Decrypt
wolfcrypt Library APIs	wc_AesCbcEncrypt
	wc_AesCbcDecrypt

3.2.3 Counter Mode (CTR)

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Counter mode turns a block cipher into a stream cipher. Each block in the sequence is different from every other block in the sequence of counters. All the counters must be unique. In CTR mode, an initial counter is chosen by the user, which is incremented on all subsequent intermediate results till the counter overflows. In both CTR encryption and CTR decryption, the forward cipher functions can be performed in parallel. Various methods of counter generation are explained in Appendix B of FIPS Pub 800-38A.

Table 3-5. Harmony APIs for CTR

Cryptographic Library APIs	CRYPT_AES_CTR_Encrypt
wolfcrypt Library APIs	wc_AesCtrEncrypt

3.2.4 Galois Counter Mode (GCM)

GCM is comprised of the AES engine in CTR mode along with a universal hash function (GHASH engine) that is defined over a binary Galois field to produce a message authentication tag. The operation is an authenticated encryption algorithm designed to provide both data authenticity (integrity) and confidentiality. The GHASH engine processes data packets after the AES operation. GCM provides assurance of the confidentiality of data through the AES Counter mode of operation for encryption. Authenticity of the confidential data is assured through the GHASH engine.

AES-GCM is a stream like cipher mode that generates two outputs: a ciphertext and a message authentication code (also known an authentication tag). The encryption and decryption use forward cipher function in CTR mode. An additional Authentication data (AAD) is used to send non-confidential data like protocol version, recipient information, and so on. AAD is authenticated but not encrypted. Therefore, the AAD is not included in the AES-GCM output. It can be used to authenticate plaintext packet headers. It is recommended to have an Initialization vector of 12 bytes

for GCM as using other greater lengths will need more time for processing and will in turn slow down the operation. The security of GCM is strongly dependent on the authentication tag size. The larger the tag, the more security. It is preferable to use the maximum possible tag length of 16 bytes for good security. For additional information on GCM mode, refer to NIST SP 800-38D.

Table 3-6. Harmony APIs for GCM

Application APIs	`int aesgcm_default_test(void);`
	`static int aesgcm_default_test_helper(const uint8_t*, int, const uint8_t*, int, const uint8_t*, int, const uint8_t*, int ,const uint8_t*, int, const uint8_t*, int);`
Cryptographic Library APIs	CRYPT_AES_GCM_SetKey
	CRYPT_AES_GCM_Encrypt
	CRYPT_AES_GCM_Decrypt
wolfcrypt Library APIs	wc_AesGcmSetKey
	wc_AesGcmEncrypt
	wc_AesGcmDecrypt

Other supporting APIs used in ECB, CBC, and CTR modes.

Table 3-7. APIs for Key and IV

Application APIs	`void aes_test(void)`
Cryptographic Library APIs	CRYPT_AES_KeySet
	CRYPT_AES_IvSet

4. AES Encryption and Decryption Example: Application Implementation

4.1 Prerequisites

The following hardware and software prerequisites to be considered:

Hardware

- One SAM E54 Xplained Pro Evaluation Kit
- One Micro USB Cable

Software

- MPLAB X IDE v5.30 and higher (Download MPLABX latest version)
- MPLAB XC32/32++ Compiler v2.30 and higher (Download latest XC32 compiler)
- MPLAB Harmony v3 Configurator (v3.6.2) (Install MHC)
- MPLAB Harmony v3 package:
  - MPLAB Harmony v3 bsp repository, v3.7.0 (Github bsp)
  - MPLAB Harmony v3 csp repository, v3.7.0 (Github csp)
  - MPLAB Harmony v3 core repository, v3.7.0 (Github core)
  - MPLAB Harmony v3 dev-packs repository, v3.8.0 (Github dev-packs)
  - MPLAB Harmony v3 Crypto repository, v3.6.0 (Github Crypto Repo)
  - MPLAB Harmony v3 wolfSSL repository, v4.5.0 (Github wolfSSL)
- Terminal Application (Tera Term)

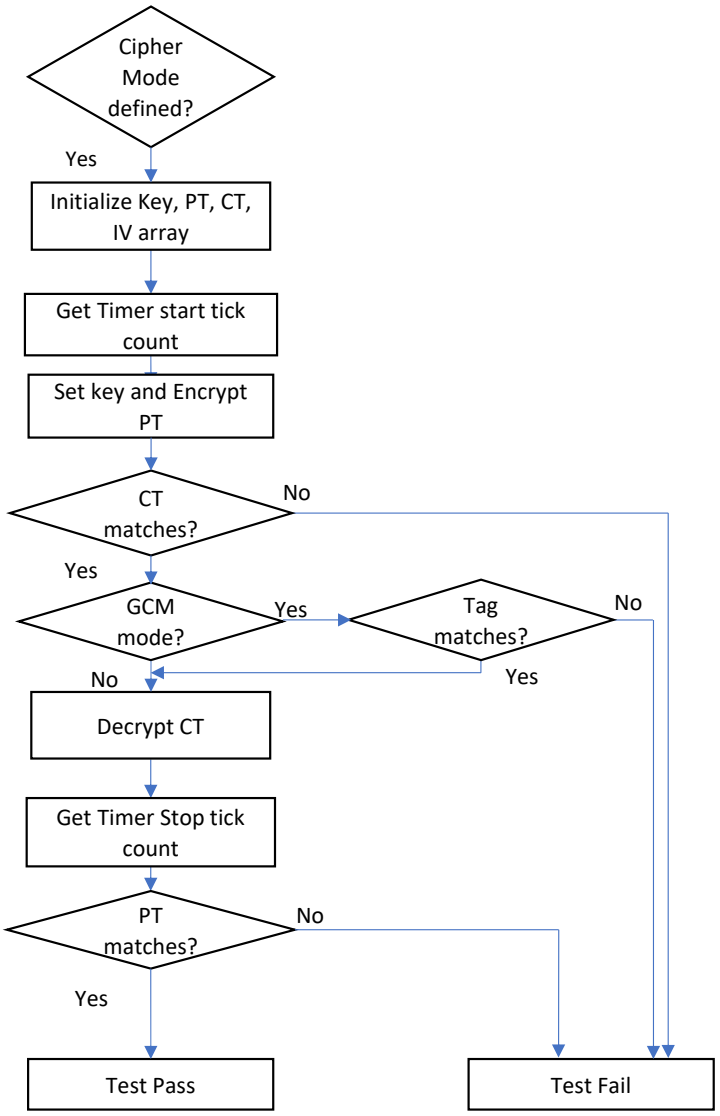
4.2 Application Synopsis

The application example uses AES as the underlying symmetric block cipher function of various block cipher operation modes, such as ECB, CBC, CTR, and GCM. AES supports three key-lengths of 128, 192, and 256 bits. This application uses the 128-bit key for all modes. The encrypted plaintext (PT) is verified against the ciphertext (CT), and the decrypted ciphertext (CT) is verified against the plaintext (PT) in all these modes. The tests are passed when the checks return success.

GCM mode needs both authenticity and confidentiality validation. Therefore, the authentication tag generated after encryption is verified before decrypting the confidential data and non-confidential additional authentication data (AAD).

The MPLAB Harmony v3 Time system service is enabled to compute the number of system clock cycles it takes for key setting, encryption, and decryption processes of each of the cipher modes.

Figure 4-1. Application Flow Chart



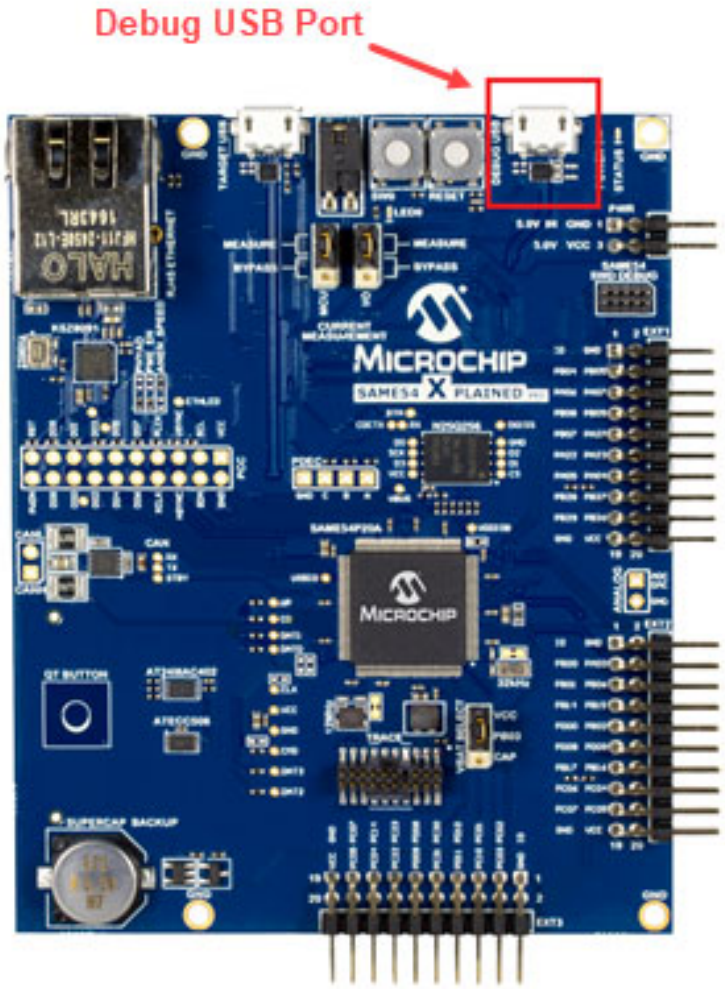
4.3 Running the Demo Application

The demo application for SAM E54 can be found at /crypto_apps_encrypt_decrypt/apps/encrypt_decrypt/firmware/sam_e54_xplained_pro.X of the H3 crypto repository.

Follow these steps to program the demo application:

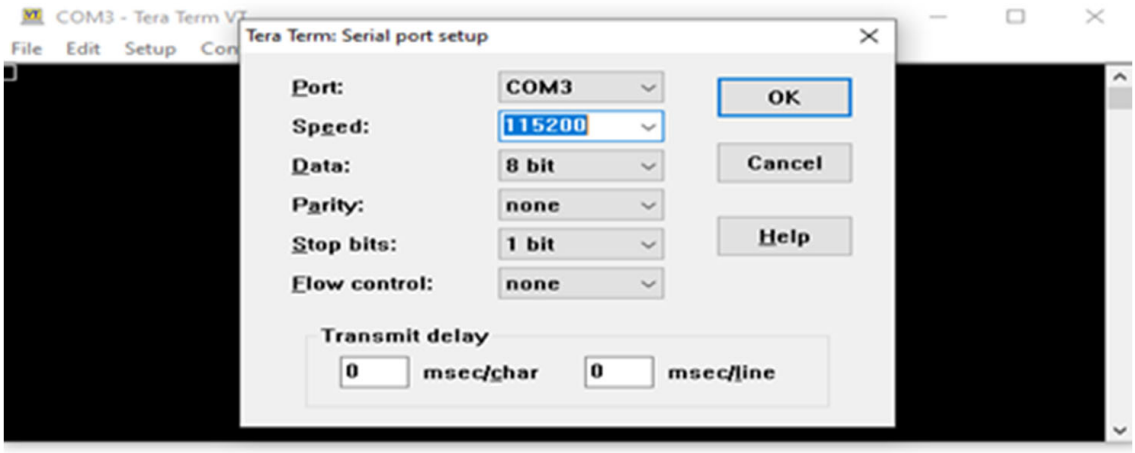
1. Connect the micro-USB cable to the DEBUG USB connector of the SAM E54 Xplained Pro board, and then power-up the board.

Figure 4-2. SAM E54 Xpro Debug USB Connection



2. Open the Terminal Application (Tera Term) and set the baud rate to 115200.

Figure 4-3. Baud Rate Setting in TeraTerm

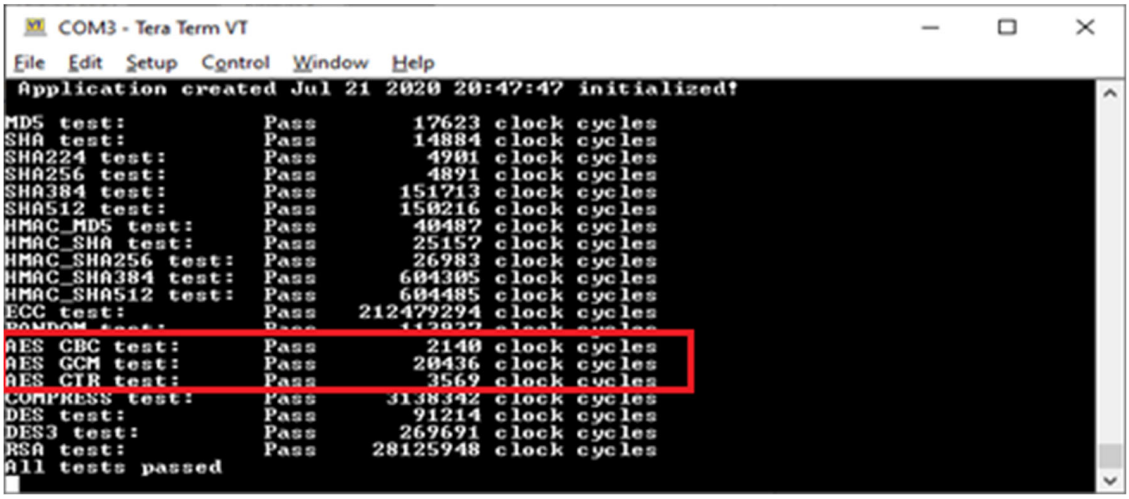


3. Open the project in MPLAB X IDE.

4. Select *File* > *Open Project*, and then from the H3 Crypto repository choose the following location: `(\crypto_apps_encrypt_decrypt\apps\encrypt_decrypt\firmware\sam_e54_xplained_pro.X)`.

5. To build and program the project to the target board, click the  icon (Make and Program device icon). The COM3 - Tera Term window will display the cryptographic ciphers and compression tests results, as shown below.

Figure 4-4. Output Window



Notes:

1. The output window will show the results of all cryptographic ciphers and compression tests. AES CBC, AES GCM, and AES CTR depicts the results of CBC, GCM, and CTR modes of the block cipher operations.
2. Refer to the section Things to Consider for including ECB mode into the demo.

4.4 Creating the Application

The application is created in MPLAB Harmony v3 Configurator (MHC) as the GUI drastically simplifies the peripheral and core configuration of 32-bit devices. It is an user-friendly tool, and enables users to configure specific peripheral drivers which makes application development easy. For additional information on the MHC, refer to the GitHub wiki page.

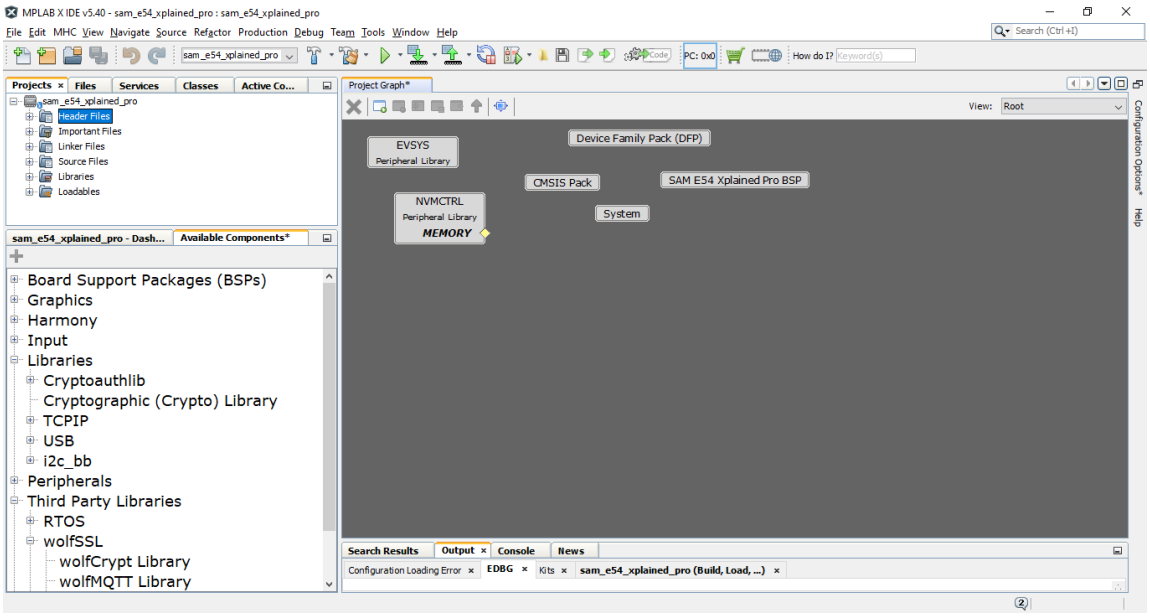
Users can create the AES application by following the steps mentioned in the following section. The MPLAB Harmony v3 AES demo application, discussed in the previous sections, is used as a reference here.

MPLAB Harmony v3 Configuration

Before following the steps below, ensure that the latest version of the MPLABX IDE and MPLAB Harmony v3 plugins are installed.

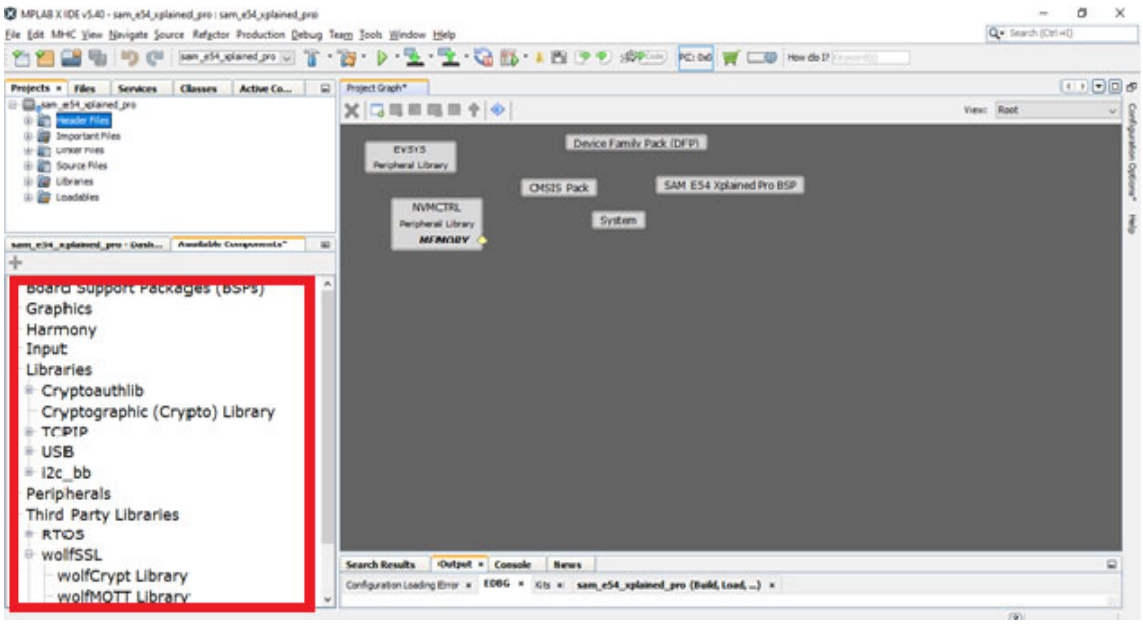
1. To create a MPLAB Harmony v3 project for SAME54P20A using MPLAB X IDE, refer to the link Create Harmony v3 project.
2. By default, the Device Family Pack (DFP), System, CMSIS Pack, and SAM E54 Xpro BSP are present when the configurator is launched.

Figure 4-5. MHC Default Modules



3. Cryptographic and wolfCrypt libraries, USART peripheral driver, and Time System service are to be added into the project graph for this demo application. To add relevant libraries and peripherals click the **Available Components** tab on the bottom left of the MPLAB X IDE

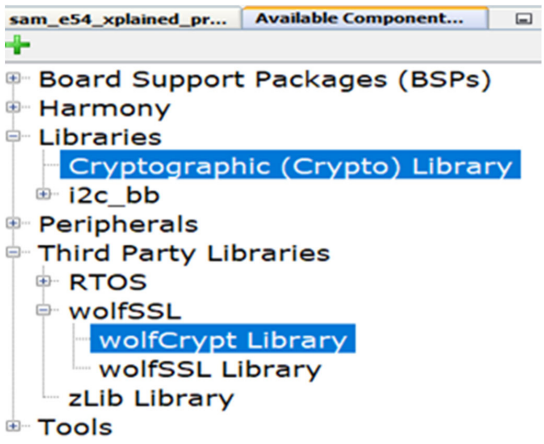
Figure 4-6. Available Components



4. The Crypto library has dependency over the wolfCrypt Library.
Libraries > Cryptographic (Crypto) Library

Third Party Libraries > wolfSSL - wolfCrypt Library

Figure 4-7. Harmony v3 and wolfCrypt Libraries

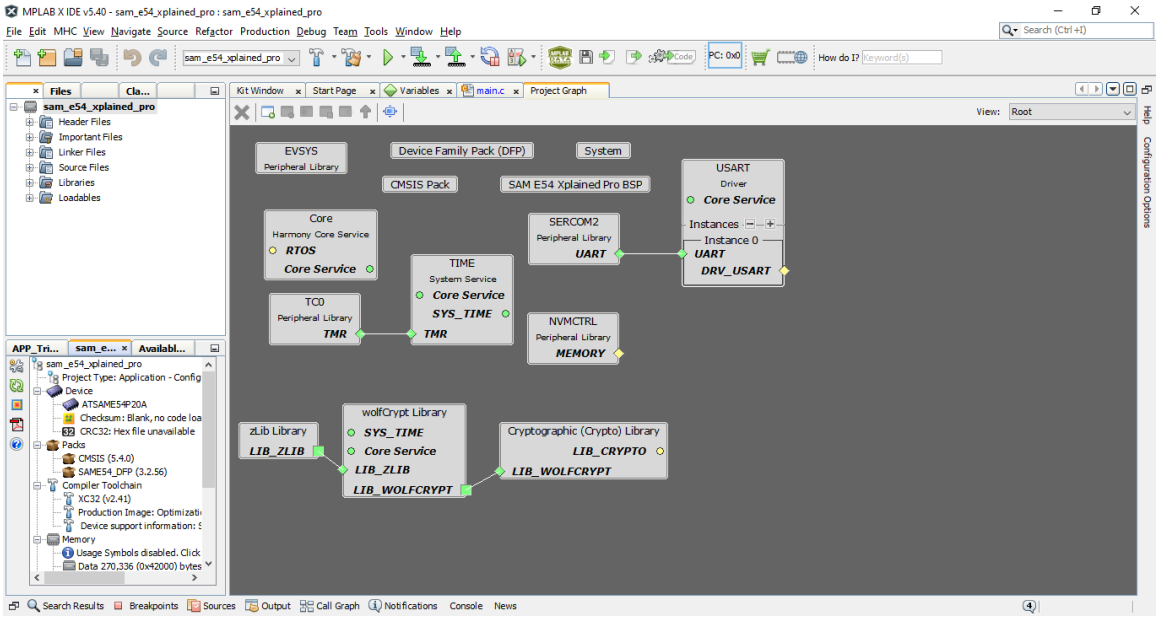


5. The USART Harmony driver has dependency over SERCOM2 instance.
   Harmony > Drivers > USART

   Peripherals > SERCOM > SERCOM2
6. TC0 instance is used by Time system service.
   Harmony > System services > Time

   Peripherals > Timer > TC0
7. The harmony configuration file (`.xml`) with these components configured can be found in the following path of the H3 Crypto repository:
   `\crypto_apps_encrypt_decrypt\apps\encrypt_decrypt\firmware\src\config\sam_e54_xplained_pro`.
   Load this file from the MPLAB Harmony v3 Configurator by navigating through *MHC > File > Load State*.
8. The following figure illustrates the demo application project graph:

Figure 4-8. Application Project Graph

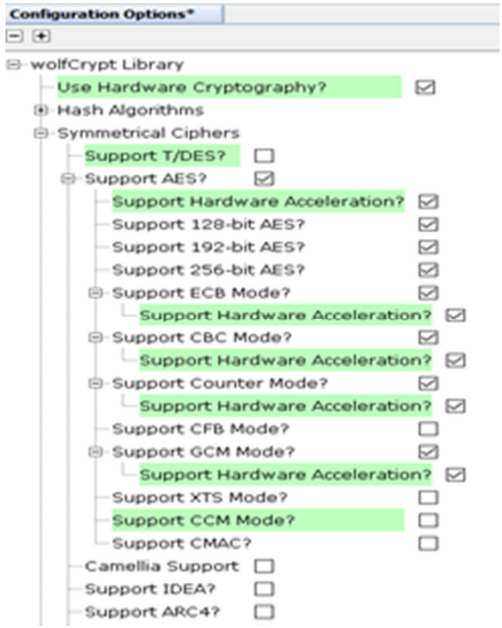


9. Because this document focuses primarily on the Cryptographic features, users can refer to the figure below while configuring the wolfCrypt Library for AES.

Notes:

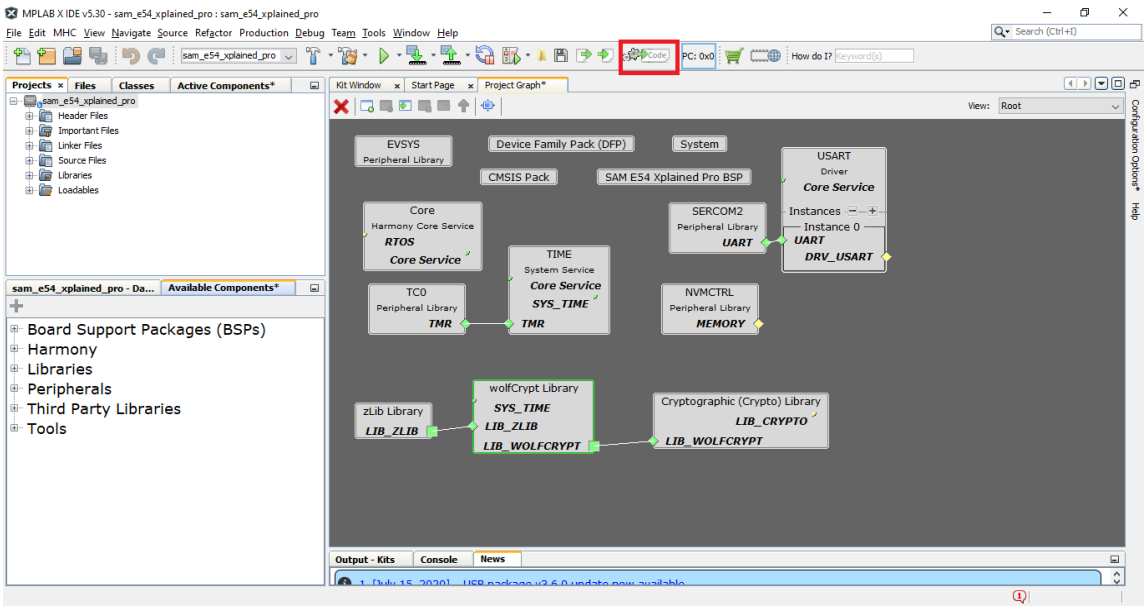
1. The demo application (`encrypt_decrypt`) has many crypto features, such as Hashing, Asymmetric ciphers, Random Number generation, and compression enabled. The scope of this document is limited to AES, hence other features are not described in this document.
2. The zLib library contains APIs that support data encoding, decoding, and compression used by other algorithms of the example. However, the zLib Library is not required for the AES operations, hence it is not within the scope of this document.

Figure 4-9. wolfCrypt Library Configuration Options



10. If any configuration changes made, the corresponding code can be generated by clicking the (Generate Code button).

Figure 4-10. Generate Code



5. Appendices

5.1 Appendix A: Example Vectors for AES Cipher Modes

The following are some of the test vector examples used for the various modes discussed in this document. Users can refer to the NIST Publication to find more sample vectors. Users can download the NIST test vectors from NIST Block Vectors and NIST GCM Vectors.

ECB AES 128
Key 2b7e151628aed2a6abf7158809cf4f3c

Block #1

Plaintext 6bc1bee22e409f96e93d7e117393172a

Ciphertext 3ad77bb40d7a3660a89ecaf32466ef97

Block #2

Plaintext ae2d8a571e03ac9c9eb76fac45af8e51

Ciphertext f5d3d58503b9699de785895a96fdbaaf

Block #3

Plaintext 30c81c46a35ce411e5fbc1191a0a52ef

Ciphertext 43b1cd7f598ece23881b00e3ed030688

Block #4

Plaintext f69f2445df4f9b17ad2b417be66c3710

Ciphertext 7b0c785e27e8ad3f8223207104725dd4

CBC AES 128
Block #1

Key 2b7e151628aed2a6abf7158809cf4f3c

IV 000102030405060708090a0b0c0d0e0f

Plaintext 6bc1bee22e409f96e93d7e117393172a

Ciphertext 7649abac8119b246cee98e9b12e9197d

Block #2

Key 00000000000000000000000000000000

IV 00000000000000000000000000000000

Plaintext f34481ec3cc627bacd5dc3fb08f273e6

Ciphertext 0336763e966d92595a567cc9ce537f5e

CTR 128 AES
Key 2b7e151628aed2a6abf7158809cf4f3c

Init. Counter f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff

Block #1

Plaintext 6bc1bee22e409f96e93d7e117393172a

Ciphertext 874d6191b620e3261bef6864990db6ce

Block #2

Plaintext ae2d8a571e03ac9c9eb76fac45af8e51

Ciphertext 9806f66b7970fdff8617187bb9fffdff

Block #3

Plaintext 30c81c46a35ce411e5fbc1191a0a52ef

Ciphertext 5ae4df3edbd5d35e5b4f09020db03eab

Block #4

Plaintext f69f2445df4f9b17ad2b417be66c3710

Ciphertext 1e031dda2fbe03d1792170a0f3009cee

GCM 128
Block #1

Key 298efa1ccf29cf62ae6824bfc19557fc

IV 6f58a93fe1d207fae4ed2f6d

PT cc38bccd6bc536ad919b1395f5d63801f99f8068d65ca5ac63872daf16b93901

AAD 021fafd238463973ffe80256e5b1c6b1

CT dfce4e9cd291103d7fe4e63351d9e79d3dfd391e3267104658212da96521b7db

Tag 542465ef599316f73a7a560509a2d9f2

Block #2

[AAD = 0]

Key 016dbb38daa76dfe7da384ebf1240364

IV 0793ef3ada782f78c98affe3

PT 4b34a9ec5763524b191d5616c547f6b7

CT 609aa3f4541bc0fe9931daad2ee15d0c

Tag 33afec59c45baf689a5e1b13ae423619

Block #3

[Plaintext, AAD length = 0]

Key b01e45cc3088aaba9fa43d81d481823f

IV 5a2c4a66468713456a4bd5e1

Tag 014280f944f53c681164b2ff

5.2 Appendix B: Things to Consider

The following checks must be made before working on the AES application when using the Crypto library version 3.6.0 and wolfCrypt Library 4.5.0

- Ensure that the APBC bridge masking for AES is configured in the `CLOCK_Initialize()` function found in *(encrypt_decrypt\firmware\src\config\default\peripheral\clock\plib_clock.c)*. If not, add the following line at the end of it:

  ```
  /* Configure the APBC Bridge Clocks */
   MCLK_REGS->MCLK_APBCMASK = 0x2e00;
  ```

  Without unmasking the APB clock for AES, it is not possible to write into the AES registers to make sure the clock is available.
- The ECB functionality is not implemented in the H3 example. To include the ECB test, the following additional modifications must be made to the project:

1. Ensure that ECB Hardware acceleration is enabled. Once it is done, generate the code.
2. Declare the following variables in the APP_DATA structure in the `app.h` file.
   - *\crypto_apps_encrypt_decrypt\apps\encrypt_decrypt\firmware\src\app.h*

```
int aes_ecb_test_result;
uint32_t aes_ecb_timing;
```

3. `ECB APIs CRYPT_AES_ECB_Encrypt` and `CRYPT_AES_ECB_Decrypt` are neither declared nor defined in the generated `crypto.h/c` files with Crypto library version 3.6.0. This must be included by the user into the corresponding files. No changes are required to be made in the wolfCrypt Library files. Add the following code into the `crypto.c` file:

```
../src/config/sam_e54_xpalined_pro/crypto/src/crypto.c

#ifdef HAVE_AES_ECB
int CRYPT_AES_ECB_Encrypt(CRYPT_AES_CTX* aes, unsigned char* out,
                          const unsigned char* in, unsigned int inSz)
{
    if (aes == NULL || out == NULL || in == NULL)
        return BAD_FUNC_ARG;

    return wc_AesEcbEncrypt((Aes*)aes, out, in, inSz);
}



/* AES ECB Decrypt */
int CRYPT_AES_ECB_Decrypt(CRYPT_AES_CTX* aes, unsigned char* out,
                          const unsigned char* in, unsigned int inSz)
{
    if (aes == NULL || out == NULL || in == NULL)
        return BAD_FUNC_ARG;

    return wc_AesEcbDecrypt((Aes*)aes, out, in, inSz);
}
#endif /* HAVE_AES_ECB */
```

4. Add the following code that has been marked in **bold** into the `aes_test(void)` function in the `app.c` file:

```
void aes_test(void)
{
    CRYPT_AES_CTX enc;
    CRYPT_AES_CTX dec;

    int AES_KEYSIZE = 0;
    #ifdef HAVE_AES_ECB


/* Input Message of 128 bit length*/
      const uint8_t msgecb[] = {
      0x6b, 0xc1, 0xbe, 0xe2, 0x2e, 0x40, 0x9f, 0x96,
      0xe9, 0x3d, 0x7e, 0x11, 0x73, 0x93, 0x17, 0x2a
    };
    /*Ciphertext of 128 bit length*/
    const uint8_t verifyecb[] =
    {
      0x3a, 0xd7, 0x7b, 0xb4, 0x0d, 0x7a, 0x36, 0x60,
      0xa8, 0x9e, 0xca, 0xf3, 0x24, 0x66, 0xef, 0x97
    };
    /* AES - ECB 128 bit key*/
    uint8_t keyecb[] =  {0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6,
                         0xab, 0xf7,0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c};
    /* I.V is not required in ECB mode*/
    uint8_t ivecb[]  = {0};  /* align */
    /* arrays to store encryption and decryption results respectively*/
    uint8_t cipherecb[AES_BLOCK_SIZE * 4] = {0};
```

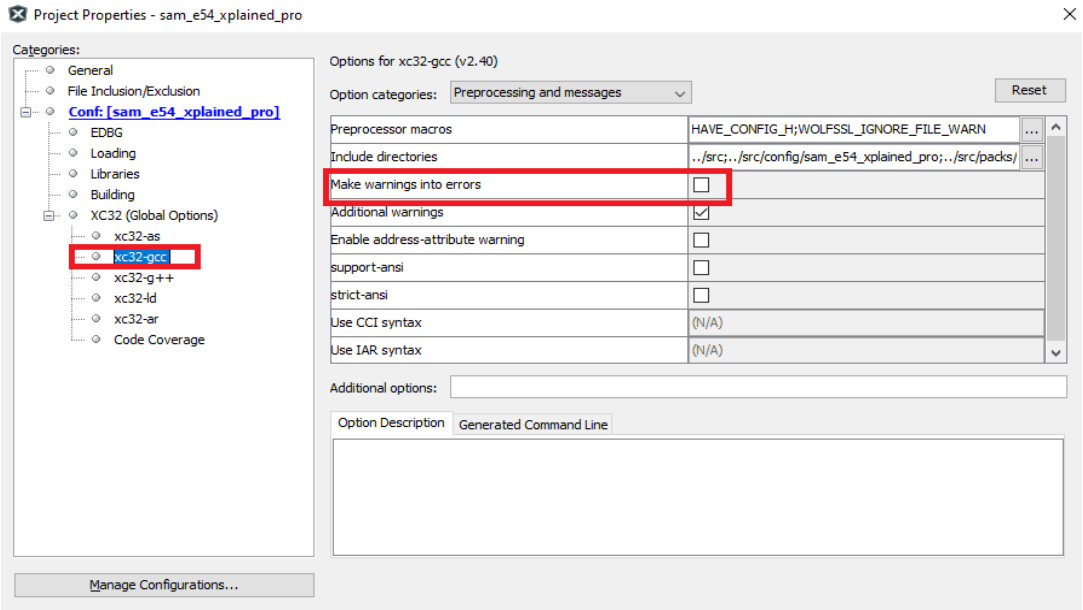
```
    uint8_t plainecb [AES_BLOCK_SIZE * 4] = {0};
    /* Test Pass count*/
    int numEcbSubTests = 2;
    uint32_t hashEcbStart;
    uint32_t hashEcbStop;
    AES_KEYSIZE = sizeof(keyecb);
    hashEcbStart = APP_getTicks();
    /* The above const allocates in RAM, but does not flush out of cache.
Copy
       it back out so it is in physical memory. */
   #if defined(HW_CRYPTO)
    SYS_DEVCON_DataCacheFlush();
#endif
    CRYPT_AES_KeySet(&enc, keyecb, AES_KEYSIZE, ivecb, AES_ENCRYPTION);
    CRYPT_AES_KeySet(&dec, keyecb, AES_KEYSIZE, ivecb, AES_DECRYPTION);
    CRYPT_AES_ECB_Encrypt(&enc, cipherecb, msgecb,   AES_BLOCK_SIZE);
    CRYPT_AES_ECB_Decrypt(&dec, plainecb, verifyecb, AES_BLOCK_SIZE);

    appData.aes_ecb_test_result = numEcbSubTests;

    if (!(memcmp(plainecb, msgecb, AES_BLOCK_SIZE)))
        appData.aes_ecb_test_result--;
    if (!(memcmp(cipherecb, verifyecb, AES_BLOCK_SIZE)))
        appData.aes_ecb_test_result--;
    hashEcbStop = APP_getTicks();
    appData.aes_ecb_timing = hashEcbStop - hashEcbStart;
#endif /* HAVE_AES_ECB */
```

5. To print the ECB test results, include the following code in the `app.c` file:

```
#ifdef HAVE_AES_ECB
sprintf(printBuffer, "%s\n\rAES ECB test:
%s",printBuffer,
(appData.aes_ecb_test_result==expectedResult?"Pass":"FAIL"));
sprintf(printBuffer, "%s\t
%10d clock cycles", printBuffer, (int) appData.aes_ecb_timing);
#endif
```

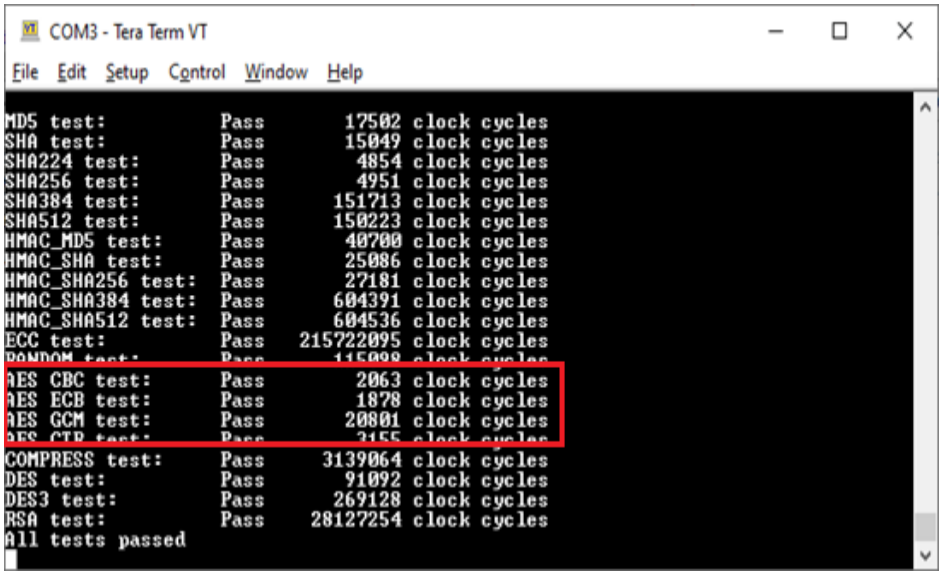
6. After making these changes to the project, in the Project Properties window, under Categories section, select xc32-gcc.
7. In the Preprocessing and messages category, clear **Make warnings into error** checkbox..

Figure 5-1. Project Properties Window



8. Follow the steps given in the section Running the Example Application. The output window will display the details for the ECB test, as shown in the following figure.

Figure 5-2. Output Window with ECB Results



6. References

1. NIST Special Publication 800-38A 2001 Edition - Recommendation for Block Cipher Modes of Operation: nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
2. NIST Special Publication 800-38D November, 2007 - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC: nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
3. How to Build an Application by Adding a New PLIB, Driver, or Middleware to an Existing MPLAB Harmony v3 Project: ww1.microchip.com/downloads/en/DeviceDoc/How_to_Build_Application_Adding_PLIB_%20Driver_or_Middleware%20_to_MPLAB_Harmony_v3Project_DS90003253A.pdf
4. MPLAB Harmony v3 Installation and Usage: github.com/Microchip-MPLAB-Harmony/mhc/wiki
5. MPLAB Harmony v3 Crypto Library: github.com/Microchip-MPLAB-Harmony/crypto/wiki
6. SAM E54 Xplained Pro: www.microchip.com/developmenttools/ProductDetails/atsame54-xpro
7. Create Your First Project with SAME54 Using MPLAB® Harmony v3: www.youtube.com/watch?v=KTEajJQ4ukc

   microchip.wikidot.com/harmony3:same54-getting-started-training-module
8. MPLAB Harmony v3 landing page: www.microchip.com/en-us/development-tools-tools-and-software/embedded-software-center/mplab-harmony-v3

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2021, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-8118-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820