AVR1317: Using the XMEGA built-in DES accelerator

Features

- **Instruction set extension to the XMEGA™ CPU performing DES iterations.**
  - **Capable of both decoding and encoding 64-bit data blocks according to the Data Encryption Standard (DES)**

1 Introduction

The XMEGA family has an instruction set extension that is performing DES iterations. This application note describes the basic functionality of the XMEGA DES instructions with code examples to get up and running quickly. A driver interface written in C and Assembler is included as well.

8-bit AVR® Microcontrollers

Application Note

Rev. 8105A-AVR-04/08

2 Theory

Cryptography is the art or science of keeping information secret and is based on either hiding the cryptographic method or securing the cryptographic key. Algorithms only based on the secrecy of the method used are mainly of historical interest and do not meet the needs of the real world. Modern algorithms use a key to control encryption and decryption. Without the matching key, the scrambled message or data cannot be arranged into plaintext.

Algorithms based on cryptographic keys are divided in two classes; symmetric and asymmetric. Symmetric algorithms use the same key for encryption and decryption while asymmetric algorithms use different keys. The most studied and probably the most widely spread symmetric algorithm is DES.

2.1 Data Encryption Standard – DES

The Data Encryption Standard (DES) was originally developed in the 1970's and was later turned into a standard by the US National Institute of Standards (NIST). DES is a symmetric cryptographic algorithm using a 64-bit key, including 8 parity bits. DES is a block cipher, operating on blocks of 64 bits of data. Each input block is processed as illustrated in Figure 2-1.The DES algorithm is no longer considered to be secure, and is therefore not recommended to use. DES itself can be adapted and reused in a more secure scheme presented later in this document.

The DES algorithm has an efficient key length of 56-bit because of the parity bits, meaning that the number of possible key combinations is:

$$2^{56} = 72{,}057{,}594{,}037{,}927{,}936 = 7.206 \times 10^{16}$$

The security is further reduced because of week keys.

Figure 2-1 the Encryption Flow According to the DES Algorithm

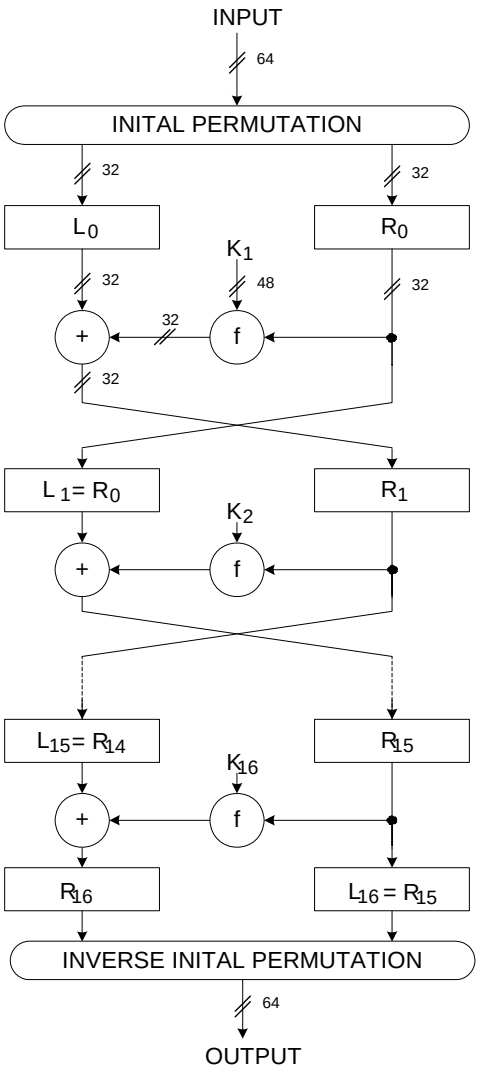


Figure 2-1 illustrates how a single block of data is being encrypted. First, the order of the input bits is changed according to the permutation function. The low 32 bits (R_0) are then processed separately from the high 32 bits (L_0). There are 16 process steps, each step using a different subset (K_n) of the encryption key (only steps 1, 2 and 16 are illustrated in Figure 2). Finally, the bit order is changed inversely with respect to the initial permutation function.

The decryption algorithm is the same as the encryption algorithm; only the sequence of the key subsets K_n is reversed.

A complete description of the DES algorithm itself is outside the scope of this application note. The reader is referred to the FIPS standard for a complete specification of the algorithm:

http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

2.2 Triple Data Encryption Standard (3DES)

Triple Data Encryption Standard (3DES) is based on using DES three times, thus increasing the key length from 56 to 168 bits. 3DES is very much stronger than DES but 3 times slower. The newer Advanced Encryption Standard (AES) offers markedly higher security margins and is recommended used if it is possible.

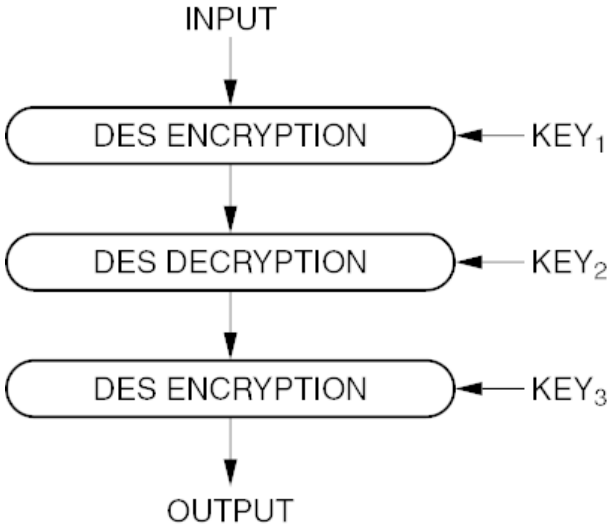
The 3DES algorithm uses three 56-bit encryption keys. The number of combinations is therefore increased to:

$$2^{168} = 3.741 \times 10^{50}$$

But because of weaknesses in the DES algorithm the effective security it provides is only 112 bits.

3DES is defined in ANSI® x9.52. The encryption flow is illustrated below.

Figure 2-2 the Encryption Flow According to the 3DES Algorithm



During encryption, the input is first encrypted with the first key, then decrypted with the second key and finally encrypted with the third key. During decryption, the key sequence and encryption/decryption block sequence is reversed.

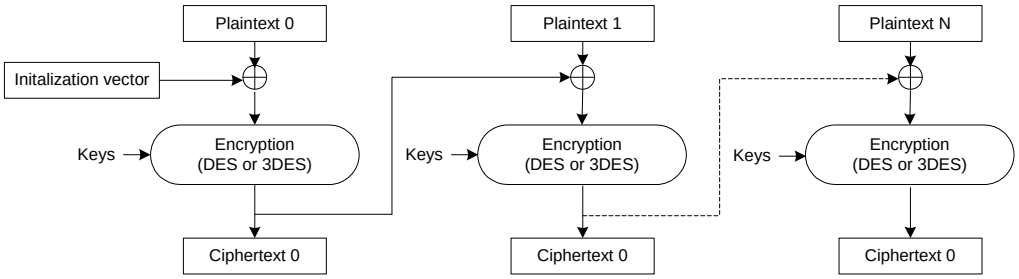
2.3 Cipher Block Chaining (CBC)

DES and 3DES are block ciphers, meaning that the algorithm operates on fixed size blocks of data. For a known input block and a constant encryption key, the output is always the same. This information may provide useful for somebody wanting to attack the cipher system.

There are methods commonly used which cause identical plaintext blocks being encrypted to different ciphertext blocks. One such method is called Cipher Block Chaining (CBC).

CBC is a method that is connecting the cipher blocks such that leading blocks influence all trailing blocks. This is achieved by first performing an XOR operation on the plaintext block and the previous ciphertext block before encrypting the result. This increases the number of plaintext bits one ciphertext bit depends on.

Figure 2-3. CBC Encryption



3 DES accelerator

The DES accelerator is an instruction set extension to the XMEGA CPU, performing DES iterations. The 64-bit data block (plaintext or ciphertext) is placed in the CPU register R0-R7, while the full 64-bit key (including parity bits) is placed in registers R8-R15. Executing one DES instruction performs one round in the DES algorithm. Sixteen rounds must be executed in increasing order to form the correct DES ciphertext or plaintext. Intermediate results are stored in the register file (R0-R15) after each DES instruction. The instruction's operand (K) determines which round is executed, and the half carry flag (H) determines whether encryption or decryption is performed.

Intermediate results in this implementation differ from the FIPS standard because the initial permutation and the inverse initial permutation are performed every iteration. This does not affect the result in the final ciphertext or plaintext. For supplementing information see the XMEGA manual.

4 Driver Implementation

This application note includes a source code package with a basic DES driver implemented in Assembly. It is written for the IAR Embedded Workbench® compiler.

The DES driver supports DES, 3DES and CBC encryption and decryptions.

Note that the DES driver is made in two versions, one with speed optimized code and one with size optimized code. It is designed as a library to get started using the DES accelerator. Please refer to the driver source code and device datasheet for more details.

4.1 Files

The source code package consists of three files:

- *DES_driver_speed.s90* – DES accelerator driver speed optimized source file
- *DES_driver_size.S90* – DES accelerator driver size optimized source file
- *DES_driver.h* – DES accelerator driver header file
- *DES_example.c* – Example code using the driver

For a complete overview of the available driver interface functions and their use, please refer to the source code documentation. Include the driver with the desired optimization option to your project.

4.2 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit http://www.doxygen.org. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

8105A-AVR-04/08