ABSTRACT

Often times, microcontrollers need to transmit secure data over an RF or wired communications channel. For secure communication, the data needs to be encrypted for transmission and decrypted upon reception. The Tiny Encryption Algorithm (TEA) is designed for use on microcontrollers and allows this. This application report explains how to implement this algorithm using the MSP430 microcontroller.

1 Introduction

Data encryption is useful in microcontroller designs that are handheld and/or are battery operated. Some examples are key fobs that are used for operating garage doors, automobile locks, starters or alarms, and residential keyless entry systems and alarms. Given enough computing power, a highly secure encryption algorithm can be implemented. The challenge on a microcontroller is trying to match the computing power with the appropriate security level. In order to fit the appropriate security level into a microcontroller, the Tiny Encryption Algorithm (TEA) is used as the basis for the crypto-engine. A user-configurable 128-bit key is used. This is demonstrated in hardware that consists of two MSP430F1232 development boards communicating via a one-way serial link. The focus of this application report is on the setting up the transmitter and receiver to demonstrate how to communicate securely between two systems.

2 Hardware Description

This encryption application makes use of two development boards, one called the Transmitter, which sends the encrypted data, and one is called the Receiver, which decrypts that data. For the purposes of this demonstration, a wire is used to connect the Transmitter with the Receiver. Refer to Appendix B for details on setting up and using the demonstration. The communication channel between the Transmitter and Receiver is a serial link implemented using port I/O hardware and a software-implemented universal asynchronous receiver/transmitter (UART). Figure 1 shows how the software could be incorporated into a larger system.
3 Software Description

3.1 Software Overview

Executing encryption software on a microcontroller is a fine balance between security level, size, and cycles. In general, the more secure an algorithm, the more cycles that are required. The TEA was chosen because it offers the best security-to-cycles ratio. As written, it does 32 iterations, with any single bit change in the input data being fully propagated in 4 iterations. The security is similar to DES, but it is done with one-third the cycles of DES. This is accomplished by using less-complicated equations but executing more iterations. Refer to the paper by David Wheeler and Roger Needham, *TEA, a Tiny Encryption Algorithm* for more detail.

This demonstration software implements a very simple key-fob application. The Transmitter sends encrypted commands, and the Receiver decrypts the command and takes appropriate action. It is intended that this software serve as the basis for an application by adding application-specific commands.

3.1.1 Code Size and Performance

The Transmitter code base is 1888 bytes and uses 159 bytes of RAM for data and 131 bytes of flash for const data. The encoding requires 5800 cycles per 64-bit encoding. The Receiver code base is 2512 bytes and uses 117 bytes of RAM for data and 727 bytes of flash for const data. The decoding requires 5800 cycles for each 64-bit data.

<table>
<thead>
<tr>
<th>FILE NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cw.c</td>
<td>Code-word generation in C</td>
</tr>
<tr>
<td>enc.c</td>
<td>Encryption algorithm in C</td>
</tr>
<tr>
<td>main.c</td>
<td>Transmitter demo application in C</td>
</tr>
<tr>
<td>io.c</td>
<td>Hardware-dependant I/O routines in C</td>
</tr>
<tr>
<td>transmitter.c</td>
<td>Transmitter-specific functions in C</td>
</tr>
</tbody>
</table>
### Table 2. File Overview – Receiver

<table>
<thead>
<tr>
<th>FILE NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cw.c</td>
<td>Code-word generation in C</td>
</tr>
<tr>
<td>dec.c</td>
<td>Decryption algorithm in C</td>
</tr>
<tr>
<td>main.c</td>
<td>Receiver demo application in C</td>
</tr>
<tr>
<td>io.c</td>
<td>Hardware-dependant I/O routines in C</td>
</tr>
<tr>
<td>receiver.c</td>
<td>Receiver-specific functions in C</td>
</tr>
</tbody>
</table>

### 3.2 Transmitter Setup

The Transmitter software flow diagram is shown in Figure 2. The Transmitter has two operating modes: Learn Mode and Button Mode. Learn Mode allows the Transmitter to be added to the system. See Section 3.6 for details on the Learn Mode. Button Mode is the normal transmitting mode, in which the input data is encrypted and transmitted to the Receiver when button sw2 is pressed. The mode is selected by placing a jumper from P2.2 to either ground or V<sub>CC</sub>. When the jumper is connected to ground, the device is in Learn Mode. When the jumper is connected to V<sub>CC</sub>, the device is in Button Mode. The mode of the Transmitter must match the mode of the Receiver for the demonstration to work correctly. When button sw2 is pressed, the eight bits associated with P1IN are transmitted. This data is encrypted, formed into a code word (see section 3.5), then transmitted over the serial link to the receiver.
Software Description

Figure 2. Software Flow Diagram – Transmitter
3.3 Receiver Setup

The Receiver software flow diagram is shown in Figure 3. The Receiver has two operating modes: Learn Mode and Button Mode. Learn Mode allows a transmitter to be added to the system. See Section 3.6 for details on the Learn Mode. Button Mode is the normal receiving mode, in which transmitted data is received. The mode is selected by placing a jumper from P2.1 to either ground or $V_{CC}$. When the jumper is connected to ground, the device is in Learn Mode. When the jumper is connected to $V_{CC}$, the device is in Button Mode. The mode of the Receiver must match the mode of the Transmitter for the demonstration to work correctly. When data is received, it must be validated before being accepted. Validation is done by comparing the serial number to a known good list and matching the Event Count within a valid range. If both these tests pass, the data is accepted.

![Software Flow Diagram](image-url)

Figure 3. Software Flow Diagram – Receiver
3.4 **TEA Implementation**

The TEA was selected for use in this application report because it was designed for a small memory footprint and low MIPS requirement. Instead of using a complex mathematical formula for encryption, TEA relies on a simple formula with many iterations. The formulas follow in the next two sections.

3.4.1 **Encryption Formula**

```c
delta=0x9e3779b9, n=32; /* a key schedule constant */
while (-->0) {
    sum -= delta;
    y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1];
    z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3];
}
```

3.4.2 **Decryption Formula**

```c
unsigned long n=32;
delta=0x9e3779b9;
sum=delta<<5;

while (-->0) {
    z-= (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3];
    y-= (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1];
    sum-=delta;
}
```

3.5 **Creating a Unique Transmission Code Word**

One of the requirements for secure data transmission is that each transmission be unique, even if the data being sent is the same. This is implemented in the application using a simple incrementing counter algorithm to generate an Event Count. Each time data is prepared for transmission, the counter is incremented and its value is incorporated into the code word. The code word is made up of both encrypted and nonencrypted information. The definition for the code word is application dependant. For the purpose of this demonstration, only encrypted data is used to make up the code word. The format of the code word can be seen in Appendix B.

3.6 **Learn Mode**

This section discusses how it is possible to securely add a new transmitter to the system without reprogramming the Receiver. When a new Transmitter must be added to an in place system, a process must be in place for the Receiver to accept it. To do this, each unit is placed in Learn Mode, and the Transmitter sends its serial number to the receiver. If accepted, the serial number is stored in a lookup table, and all subsequent transmissions have the serial number compared to this table.

4 **Conclusion**

This application report describes how to implement an encryption and matching decryption algorithm on two MSP430 microcontrollers communicating via serial connection. Although not implemented here, this technique could be extended to use wireless transmission via RF signals.

5 **References**

1. *TEA, a Tiny Encryption Algorithm*, David Wheeler and Roger Needham, Computer Laboratory, Cambridge University, England
2. *MSP430x1xx Family User's Guide* (SLAU049)
3. MSP430x12x2 data sheet (SLAS361)
Appendix A Code Word Formats

This section describes the format of the code word that is transmitted over the serial communication link between the Transmitter and Receiver. In this example, the CRC byte is not generated or transmitted.

A.1 Button Mode Code-Word Format

<table>
<thead>
<tr>
<th>Port P1 Data 8 Bits</th>
<th>32-Bit Serial Number</th>
<th>24-Bit Event Counter</th>
<th>8-Bit CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypted Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A.2 Learn Mode Code-Word Format

<table>
<thead>
<tr>
<th>Encrypted Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>64-Bit Data</td>
<td></td>
</tr>
<tr>
<td>8-Bit CRC</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B Details on Using the Demonstration Software

1. Connect the FET boards as shown in Figure B-1.
2. Port 2 determines which mode the software executes in. Set each board to match either Learn Mode or Button Mode\(^{(1)}\) (see Figure B-1).
3. Load Transmitter and Receiver programs to respective boards.
4. Run the Receiver program. The terminal emulation program should display some Receiver status information.
5. Run the Transmitter program.
6. LED (P1.0) indicates mode (ON = Learn, OFF = Button).
7. Press SW2 to:
   - Transmit P1 data to Receiver if in Button Mode
   - Transmit key code to Receiver if in Learn Mode
8. Observe terminal emulation screens for status and transmitted data

\(^{(1)}\) Whenever there is a mode change, the boards must be reset.

Figure B-1. Demo Hardware Setup
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplifiers</td>
<td>Audio</td>
</tr>
<tr>
<td>Data Converters</td>
<td>Automotive</td>
</tr>
<tr>
<td>DSP</td>
<td>Broadband</td>
</tr>
<tr>
<td>Interface</td>
<td>Digital Control</td>
</tr>
<tr>
<td>Logic</td>
<td>Military</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>Optical Networking</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>Security</td>
</tr>
<tr>
<td>Low Power Wireless</td>
<td>Telephony</td>
</tr>
<tr>
<td></td>
<td>Video &amp; Imaging</td>
</tr>
<tr>
<td></td>
<td>Wireless</td>
</tr>
</tbody>
</table>

Mailing Address: Texas Instruments  
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2006, Texas Instruments Incorporated