Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN62769

Encrypted Data Communication Using Cypress PLC Solution

Author: Aditya Yadav
Associated Project: Yes
Associated Part Family: CY8CPLC20
Software Version: PSoC Designer 5.4
Related Application Notes: AN54416

If you have a question, or need help with this application note, contact at roit@cypress.com.

AN62769 describes the implementation of an AES-128 encryption algorithm for the Cypress Powerline Communication (PLC) solution. The associated code example can be used to encrypt, transmit, receive, and decrypt the data.

Contents

Introduction

Powerline communication (PLC) has received tremendous attention in recent years as an alternative and cost-effective last-mile access technology. Because powerline outlets are available nearly everywhere, power lines form an all-encompassing network and the largest infrastructure for communication. This has enabled many applications to use power lines as a communication medium.

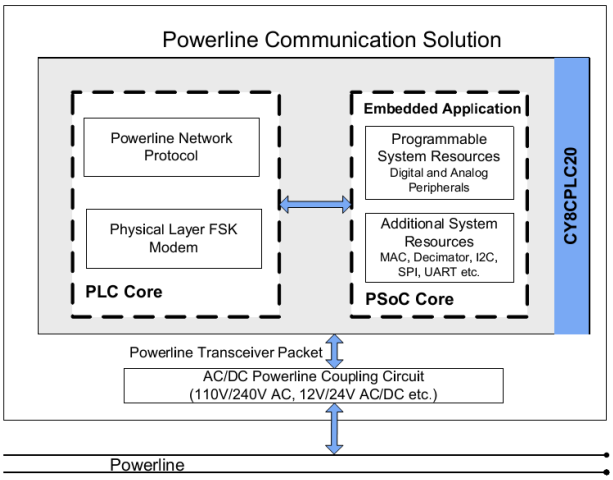
PLC is based on a bus topology, which means that packets sent by one node can be received by any other node. To overcome security and confidentiality issues, encryption is often used with PLC. Because the data is encrypted, only the node with the same key can decrypt the packet. This application note describes the encryption implementation on the Cypress PLC solution, which can be evaluated on the CY3274 High Voltage (HV) 110-240VAC PLC DVK.

Cypress PLC Solution Overview

Cypress provides a robust solution to implement PLC for low-bandwidth applications. The CY8CPLC20 is an integrated PLC solution with the Powerline Modem PHY based on FSK technology and carrier sense multiple access (CSMA)-enabled Network Protocol Stack running on the same device. This solution complies with key standards governing the use of Powerline for communication in Europe and North America.

A block diagram of the PLC solution is shown in Figure 1. To interface the CY8CPLC20 device to the powerline, a coupling circuit is required. Cypress provides the CY3274 DVK that contains coupling circuits for high voltage (110 to 240V AC) powerlines. These can be used as a reference for developing the end-user system.

Figure 1. Block Diagram of PLC Solution



For more information on CY8CPLC20, refer to the datasheet available at http://www.cypress.com/?rid=38201

Encryption

Encryption is a process in which ordinary data (plain text) is converted into cipher text. Decryption is converting the cipher text back to the plain text. Ciphering is an algorithm that performs encryption and decryption using a key. Cipher text is understood only by the devices sharing the same key, whereas the plain text can be understood by every device. The key is known only to the devices that are communicating, which prevents other devices on the shared medium from decoding correct data from the packet.

This application implements the AES-128 algorithm. The input for the Advanced Encryption Standard (AES) algorithm consists of sequences of 128 bits. AES has three different configurations with respect to the number of rounds and the key size. The following table summarizes these three configurations.

Table 1. Key Size and Number of Rounds

Size of Plain Text	Number of Rounds	Key Size
128 bits	10	128 bits
	12	192 bits
	14	256 bits

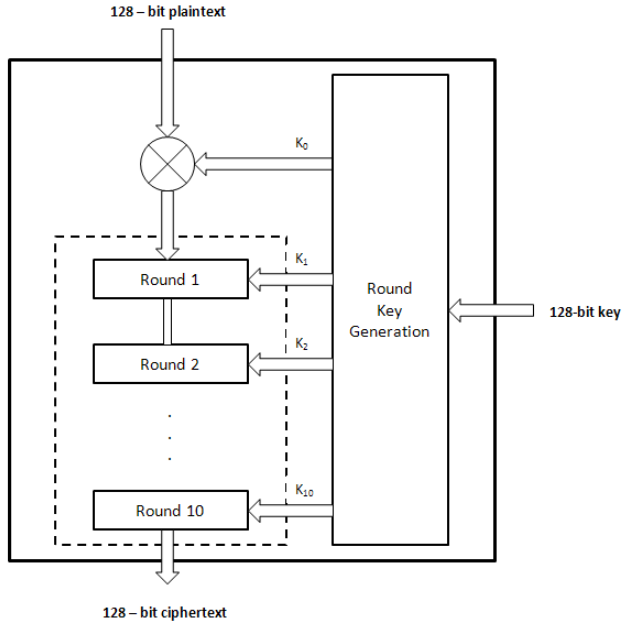
In this application, the cipher key for the AES algorithm is a sequence of 128 bits. The data (128 bits = 16 bytes) is arranged as a 4 x 4 matrix.

The encryption algorithm is executed on the 4 x 4 matrix that includes the following:

1. `SubBytes`—a nonlinear substitution step where each byte is replaced with another according to a lookup table.
2. `ShiftRows`—a transposition step where each row of the state is shifted cyclically a certain number of steps.
3. `MixColumns`—a mixing operation that operates on the columns of the state, combining the four bytes in each column.
4. `AddRoundKey`—each byte of the state is combined with the round key using bitwise Ex-OR.

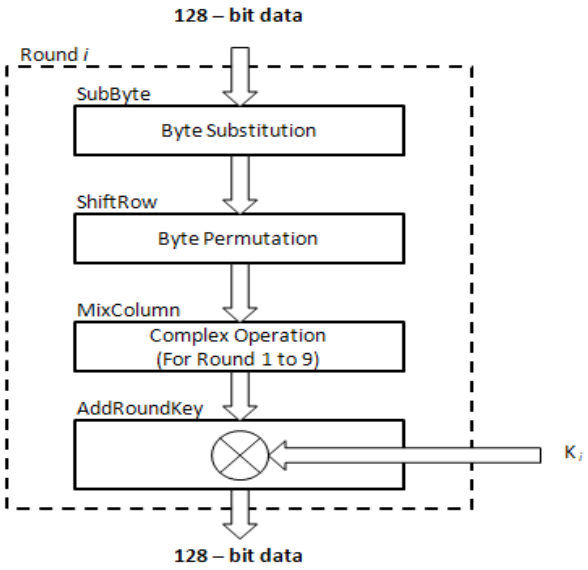
The AES algorithm is executed as shown in the following figure.

Figure 2. Ten Rounds in AES128



In Round 0, the input data is simply XORed with the key. In the next 10 rounds, the data and the key are operated by the SubByte, ShiftRow, MixColumn, and AddRoundKey steps.

Figure 3. Individual Steps in Each Round



The tenth round includes only the first three steps and not the AddRoundKey. Although all 10 rounds are almost identical, each round operates on a different key derived from the original key. After the tenth round, the key generated is used as the decryption key. The details of the execution are provided in the *encrypt.asm* and *decrypt.asm* files in the associated code example. For more information on the encryption, refer to http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

Application

In this application, a PLC node starts sending packets on the powerline when a switch is pressed. The packet contains encrypted data that corresponds to the analog voltage that is read from an input pin. When the user again presses the switch, the PLC node stops transmitting packets. Users can specify the address of the receiving node through DIP switches. The receiver decrypts a valid packet and displays the received data on the LCD.

This application is based on the peer-to-peer topology. Any node in this application can send the encrypted packet to any other node on the powerline network. The Cypress PLC solution has an on-chip network layer that can support 8-bit logical, 16-bit logical, and 64-bit physical addressing. Due to the addressing, the packet sent to one node is received only by that particular node and not by other nodes on the network. For example, if the node sends the encrypted packet to the node with logical address = 0x02, then a node with logical address = 0x02 only receives the packet.

Sending Encrypted Packets Over the Powerline Network

Table 2. PLT Packet Structure

Byte Offset	Bit Offset							
	7	6	5	4	3	2	1	0
0x00	SA Type	DA Type		Service Type	RSVD		Response	RSVD
0x01	Destination Address (8-bit Logical, 16-bit Extended Logical or 64-bit Physical)							
0x02	Source Address (8-bit Logical, 16-bit Extended Logical or 64-bit Physical)							
0x03	Command							
0x04	RSVD			Payload Length				
0x05	Seq Num				Powerline Packet Header CRC			
0x06	Payload (0 to 31 Bytes)							
	Powerline Transceiver Packet CRC							

This application uses the Cypress-proprietary network protocol that has the network frame shown in Figure 2. The encrypted data is sent using the Payload field. As the encrypted packet is always 128 bits, the payload is always 16 bytes in length. The command ID is fixed to 0xF0 so that the receiving node understands that the payload received has been encrypted. For more information on all the fields in the network frame, refer to the CY8CPLC20 datasheet.

In this application, the communication is always initiated by the node that sends the encrypted data. The following tables summarize the transmitter parameters for the packet. For more information on the memory array registers, refer to the PLC20 datasheet.

Table 3. Transmitter Memory Array Settings

Offset	Register Name	Value	7	6	5	4	3	2	1	0
0x06	**Tx Message Length**	0x10	0	0	0	1	0	0	0	0
0x07	**Tx_Config**	0x10*	0	0	0	1	0	0	0	0
0x08	**Tx_DA**		Set by the DIP switches							
0x10	**Tx Command ID**	0xF0	1	1	1	1	0	0	0	0
0x11:0 x21	**Tx Data**		Encrypted Data							

*The Tx_Config register contains a value of 0x10. This means that the node that transmits a packet also expects an acknowledgement back from the receiver. Acknowledgments are automatically sent and have no payload (therefore, no encryption).

Whenever a node receives a packet over powerline, it extracts the Payload Length, Source Address, Command ID, and the Data from the packet and copies the values from the memory array at the registers shown in the following table.

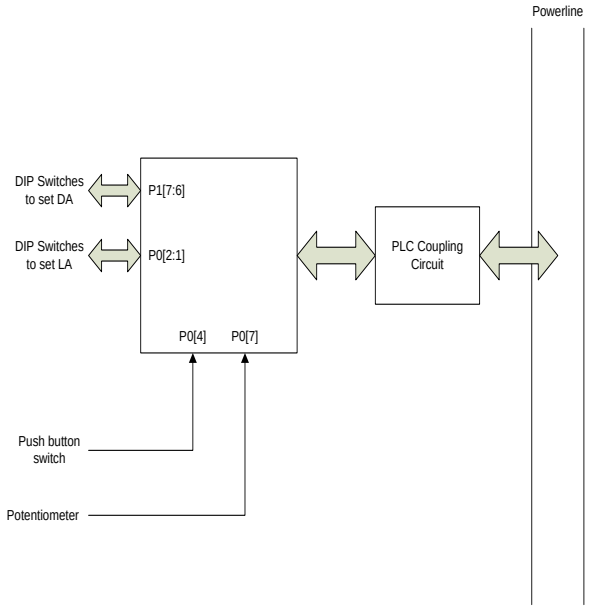
Table 4. PLT Memory Array Receiver Registers

Offset	Register Name	7	6	5	4	3	2	1	0
0x40	RX_Message_INFO	New_RX_Msg	RX_DA_Type	RX_SA_Type	RX_Msg_Length				
0x41	RX_SA	Remote Node Source Address(8 Bytes)							
0x49	RX_CommandID	RX Command ID							
0x4a	RX_Data	RX Data (31 bytes)							

Application Development (UM Information)

The application is developed using Cypress's CY8CPLC20 – Programmable PLC chip. The following figure shows the pin configuration for this application.

Figure 4. Pin Configuration



Setting the Logical and the Destination Address of the PLC Node

The logical address of the node is set through P0[2:1]. The pins are active LOW and are internally pulled HIGH. To set the LSB of the address, the P0[1] should be grounded. Similarly, to set the MSB of the address, P0[2] should be grounded. This way, logical address between 0 to 3 can be assigned to a PLC node.

The destination address can be set through P1[6:5]. The logic required to set the destination address is the same as that of the logical address.

The DIP switches available on the PLC development kits can be used to set the logical and the destination addresses.

Firmware Development

The PLC device in this application performs two tasks:

- Polls to check if any packet is received over the powerline
- Transmits a packet when the user presses the switch.

The firmware for the CY8CPLC20 PLC part is developed using Cypress's PSoC® Designer™ 5.4.

The following User Modules are used in this application.

1. PLT (PLC Modem + Network Stack)
   This User Module performs all PLC tasks including transmitting and receiving packets over Powerline. The properties for the PLT User Module are set in the code.
2. ADCINC
   This User Module converts the analog signal to its equivalent digital value. The ADC is configured to output an 8-bit unsigned value.

3. PGA
   The signal is fed to the ADCINC User Module through this User Module. The gain for the PGA is set to 1. The input for the PGA is through the analog column input MUX, which is connected to P0[7]. The output of the PGA goes to the ADCINC.

4. Counter
   This User Module is used to generate a delay. The input clock for the counter is 32 kHz in frequency and the period is set to 6400. This provides the counter output period to be 6400/32000 = 200 ms.

   The same counter is used to generate a delay of 2 ms by setting the counter period to 64 in the code. The ISR for this User Module is the `Counter_ISR` function, which is in the *main.c* file. It is called from the *CounterINT.asm* file, which is located in the \lib directory.

5. LCD
   This User Module is used to interface the PLC chip to the LCD. Port 4 is assigned for the LCD.

6. LED (three instances)
   The three instances of the LED User Modules correspond to the Tx, Rx and BIU. They are HIGH when the PLT user module is transmitting, receiving, and has a Band-In-Use timeout, respectively. All the LEDs are configured to be active HIGH. The following table summarizes the pins used by the LED User Module instances.

LED	Port Pin
Tx	P2.5
Rx	P2.3
BIU	P2.1

7. GPIO interrupt (Not a User Module)
   An interrupt has been enabled on a falling edge on P0[4]. The ISR for this User Module is the `GPIO_ISR` function, which is in the *main.c* file. It is called from the file PSoCGPIOINT.asm, which is located in the \lib directory.

Figure 5. Interconnect View for Digital Blocks

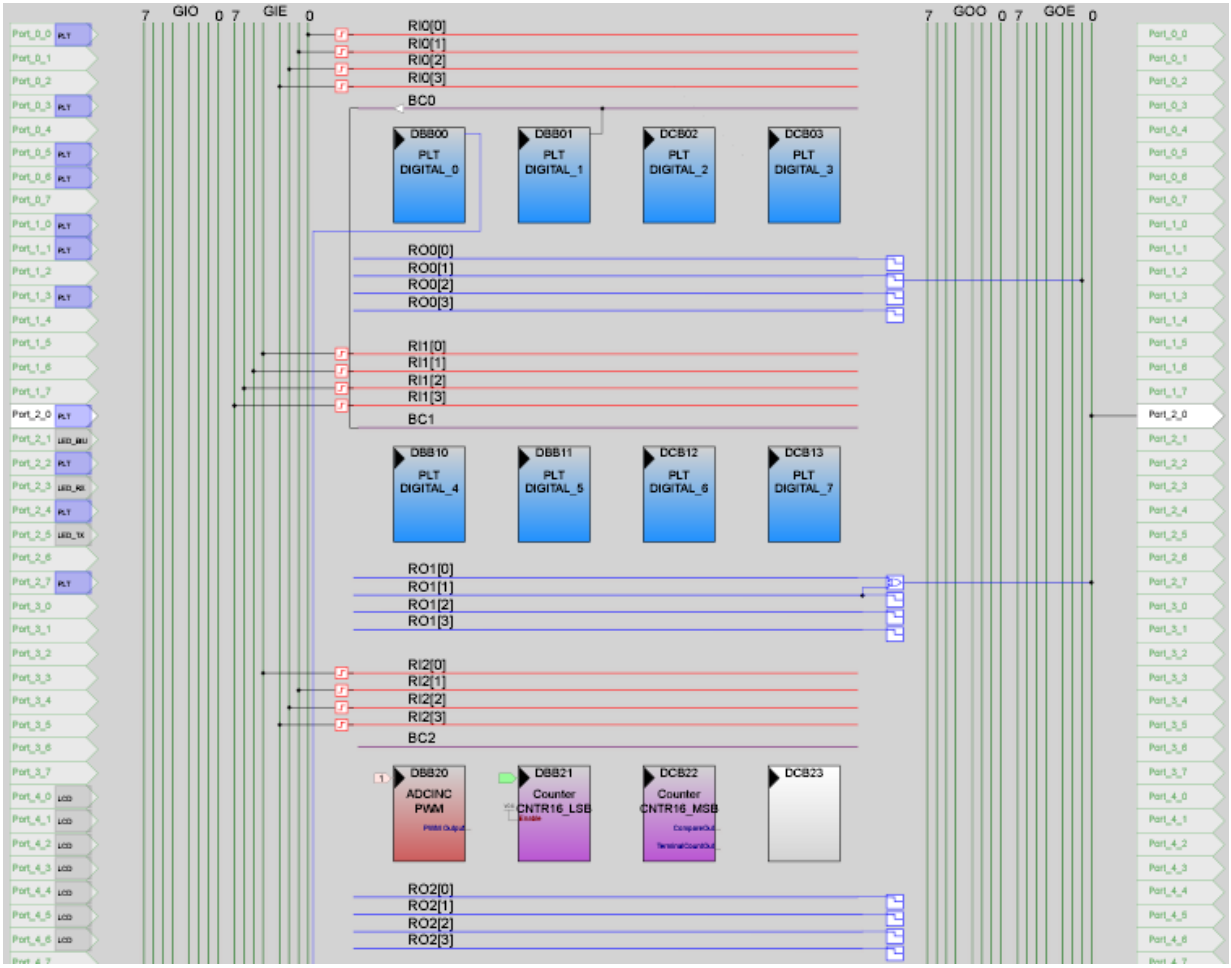
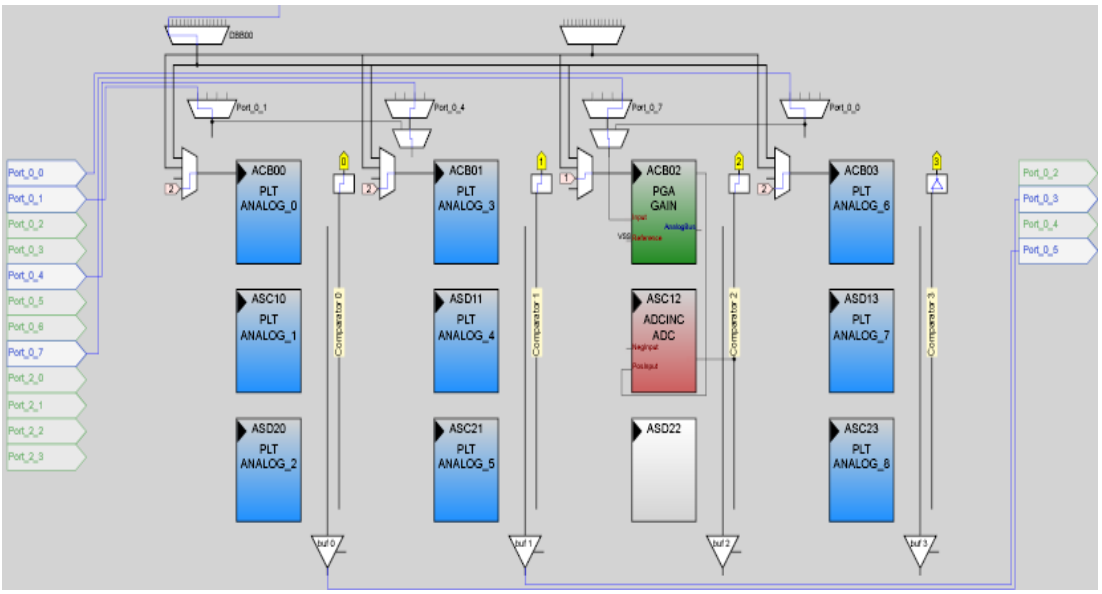


Figure 6. Interconnect View for Analog Blocks



Code Example Encryption Files

The attached code example uses the following files for encryption and decryption. These files are modified for the large memory model devices that use 2 KB of RAM.

aescopy.asm contains functions for copying to and from the `AES_Key` and `AES_Text` arrays. The encryption and the decryption functions operate only on these two arrays. Therefore, it is important to copy the correct data to `AES_Text` and correct key to `AES_Key` array.

common.asm contains common variables, tables and functions used by both encrypt and decrypt.

encrypt.asm contains functions required to encrypt the data available in `AES_Text`. After the encryption is done, the output data is returned in the `AES_Text` buffer and the key stored in `AES_Key` is also modified. Therefore, it is important to load the encryption key every time before the encrypt function is called.

decrypt.asm contains functions required to decrypt the data available in `AES_Text`. Once the decryption is done, the output data is returned in the `AES_Text` buffer and the key stored in `AES_Key` is also modified. Therefore, it is important to load the decryption key every time before the decrypt function is called.

wraps.asm file contains wrappers to enable C calls to the assembly functions.

Updated Library Files

8. *CounterINT.asm* (located in the \lib directory) is modified to call the ISR for the counter. In this file, the following interrupt code is added:

```
_Counter_ISR:
   ljmp  _WaitCounter_ISR
   reti
```

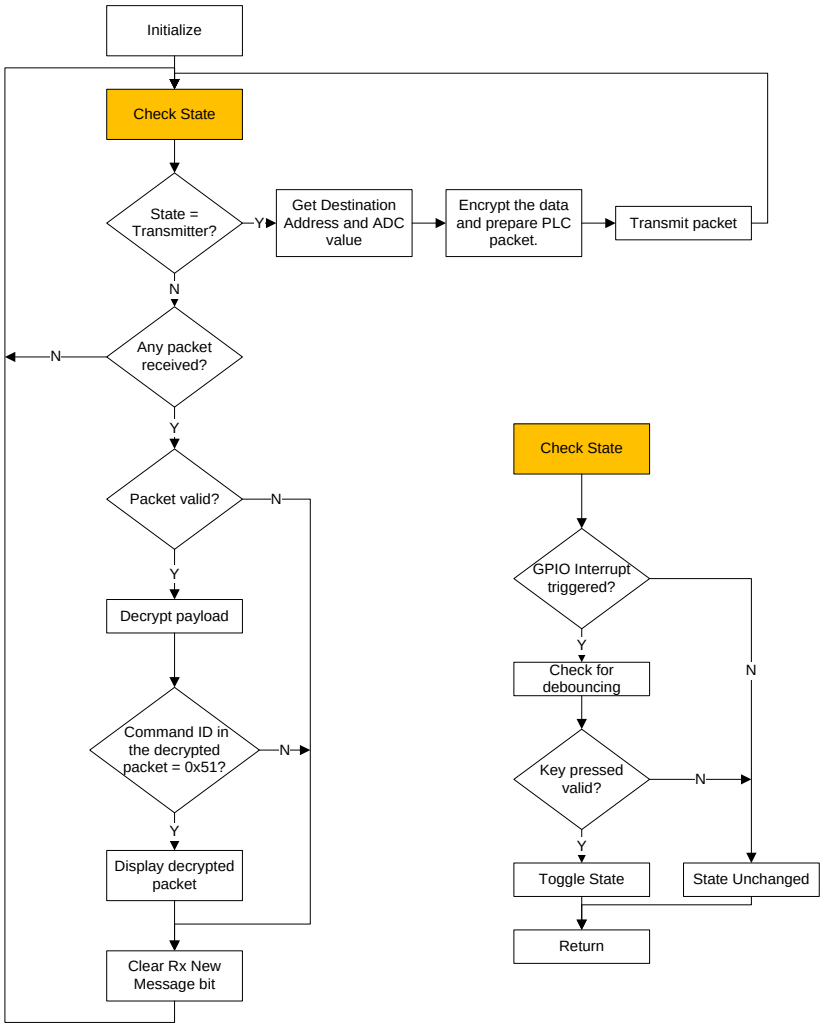
9. *PSoCGPIOINT.asm* (located in the \lib directory) is modified to call the ISR for the GPIO interrupt on pin P0[7]. In this file, the following interrupt code is added:

```
PSoC_GPIO_ISR::
   ljmp  _GPIO_ISR
   reti
```

10. *Custom.lkp*: The `AES_Encrypt` and `AES_Decrypt` functions operate on `AES_Text` and `AES_Key`. However, the current page pointer (`CUR_PP`) and the index page pointer (`IDX_PP`) are set to `AES_Text` only. Therefore, it is required to have `AES_Text` and `AES_Key` on the same page. This is done by adding the following line in this file.

    Line 2 – `Bmyarea:`

Figure 7. Flowchart



Description

On start up, the firmware initializes all the UMs. The PLC device then reads the logical address set using the DIP switches. The encryption key is stored in the EEPROM, which is then loaded into RAM. The PLC device then calculates the decryption key from the encryption key. The decrypt key schedule is in reverse order and therefore, the decryption key is not the same as the encryption key. The key used for decryption is the same as the key left after an execution of encryption algorithm.

After the initialization, the PLC device keeps checking the state. The state can be either a transmitter or a receiver. If the state is a receiver, then the PLC device polls for the New_Rx_Msg bit (Bit 7) in the Rx Message INFO register (Memory Array Offset 0x40) to check if any packet is received over Powerline.

If the PLC node has received a packet over Powerline, then the PLC device checks if the received parameters are valid. The PLC device checks the payload length and the command ID in the received packet.

If the payload length in the Rx_Message_INFO register (Memory array register 0x40) is 0x10 and the Rx_CommandID (Memory array register 0x49) is 0xF0, then the packet is valid. If any of the two conditions are not met, then the packet is discarded.

The PLC device decrypts the valid packet, which contains the User Command ID (different from the RX_CommandID), a 4-bit serial number and 4-bit data length, and the data (Refer to Table 5). If the User Command ID in the decrypted data is not 0x51 then the packet is discarded. Otherwise, the PLC device displays the decrypted data on the LCD. The data displayed on the LCD consists of the command ID, Serial ID, and the data value of the decrypted packet.

Note The command ID in the PLC packet and the command ID in the decrypted data are different. The command ID in the PLC packet is fixed to 0xF0, so that the receiving PLC node understands that the received payload is encrypted. The User Command ID in the decrypted packet is used so that the receiving node understands the decrypted data. In this application, 0x51 specifies that the data correspond to the ADC value.

A GPIO interrupt has been enabled for the P0[4], which is connected to the push-button switch. As the interrupt type is set to the falling edge, the user has to release the key before the interrupt is triggered. On the interrupt, the PLC device runs the key debouncing algorithm.

The debouncing mechanism works as follows:

Once the key is released, the GPIO interrupt is triggered. In the ISR, a flag is set indicating that the GPIO interrupt has been processed. The PLC device polls for this flag and when the PLC device finds the flag HIGH, it polls for the interrupt pin P0[4]. The PLC device waits until it finds the pin LOW. When the pin is LOW, the PLC device starts a counter for 2 ms and at the end of the 2 ms, the PLC device again polls for the interrupt pin. If the pin is found to be LOW, then it confirms that the pin has been released.

After the PLC device detects the key press, it toggles the state. If the state is a transmitter state, the PLC device calculates the Destination Address based on the DIP switch's position. The PLC device also gets the ADC value and encrypts the packet. The data for the encryption is passed in the PLT Memory array TX_Data registers as shown in the following table.

Table 5. Data Before Encryption

Offset	Data Buffer
0	User Command ID
1	Serial Number (7:4) Data Length (3:0)
2	Data

The first byte in the data corresponds to the command ID. The Command ID field in the memory array (offset 0x10) is always 0xF0. However, many applications require using custom command IDs. As the command ID byte in the PLC packet is accessible to all the devices on the network, it reduces the security. Therefore, the command ID in the PLC packet is always 0xF0 and the real command ID is sent as the first byte in the data buffer.

The second byte in the data buffer contains the serial number at bits 7:4, which increments with every packet transmitted. It enables the receiver to know if it has missed any packet. In the data buffer, it is possible that consecutive packets have the same values for command ID, data length and the data. If the serial number is not used, then the data after encryption would always be the same, which degrades the confidentiality.

The second byte in the data buffer contains the data length at bits 3:0. The encrypted packet is always 16 bytes in length, but the actual data need not be 16 bytes. The data length specifies the length of the useful data in the remaining packet. In this application, the data length is always 1 as it transmits an 8-bit ADC value. The maximum length according to this data frame can be 14 bytes.

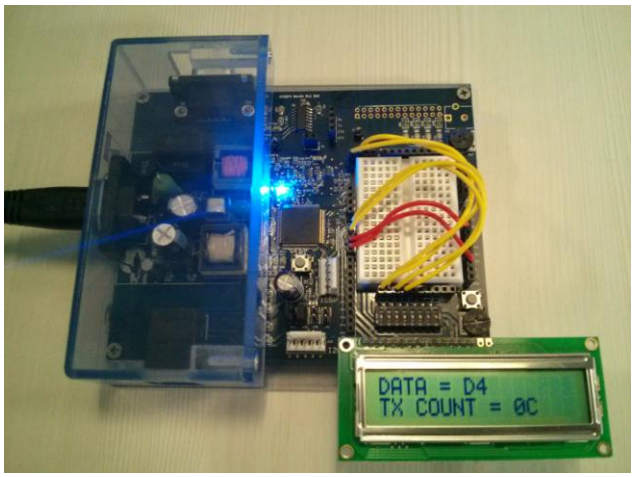
The data buffer contains the data at offset 0x02 onwards. In this application, the data is an 8-bit ADC value. The ADC is connected to P0[7] through a PGA. P0[7] is connected to the potentiometer on the development kit. Different ADC values can be obtained by changing the potentiometer position.

After encryption is over, the encrypted packet is again placed back to the PLT Memory Array TX_Data registers. Then, the PLC device transmits the packet to the destination node.

Hardware Implementation

This application is developed using the Cypress CY3274 kit, which has the CY8CPLC20 device. To set the logical and the destination address, P0[2:1] and P1[7:6] should be connected to the onboard DIP switches. The potentiometer output (marked as "VR") should be connected to P0[7]. The potentiometer (R47) is used to vary the input to the ADC. This ADC value is encrypted in the packet and also displayed on the LCD. The on-board push button switch (marked as 'SW') should be connected to P0[4]. The push-button switch is used to generate an interrupt so that the node starts/stops the transmission. The following figures show the connections for this application on the transmitter side.

Figure 8. Hardware Connections



The LCD should be connected to the LCD header on both the transmitter and the receiver.

Summary

This application note shows an example where packets communicated over the Powerline contain an encrypted payload (using AES128), which adds security and confidentiality. The encryption implementation is such that the encryption code can be reused in any other application. The same firmware is capable of encrypting and decrypting the data and therefore it can be used to transmit and receive data over the Powerline.

Document History

Document Title: Encrypted Data Communication Using Cypress PLC Solution - AN62769

Document Number: 001-62769

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2968402	ROSG	07/08/2010	New application note
*A	3210215	FRE	03/29/2011	Updated to PSoC Designer 5.1 SP1
*B	4076935	ADIY	07/242013	Updated Software Version as “PSoC Designer 5.4”. Updated Figure 1 under Cypress PLC Solution Overview. Updated in new template. Completing Sunset Review.
*C	4402430	ROIT	06/09/2014	Removed reference to CY3275 kit

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com