THIS SPEC IS OBSOLETE

Spec No: 001-52478

Spec Title: AN52478 - DESIGNING AN EXTERNAL HOST APPLICATION FOR CYPRESS'S POWERLINE COMMUNICATION IC CY8CPLC10

Sunset Owner: Rohit Kumar (ROIT)

Replaced by: None

AN52478

Designing an External Host Application for Cypress's Powerline Communication IC CY8CPLC10

Author: Aditya Yadav
Associated Project: Yes
Associated Part Family: CY8CPLC10
Software Version: PSoC® Designer™ 5.4
Related Application Notes: AN55427

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

AN52478 describes how to configure the device CY8CPLC10 with a microcontroller, so the user will be able to transmit and receive data over the powerline with an external microcontroller. The CY8CPLC10 is an integrated Powerline Communication chip with the Powerline Modem PHY, Powerline Network Protocol Stack, and I^2C interface. An External Host can control CY8CPLC10 to communicate with different nodes on the powerline.

Contents

Introduction

The Cypress PLC family is a single chip solution for powerline communication (PLC). It has a robust FSK modem with a user-friendly powerline network protocol. Cypress's PLC solution and a simple powerline coupling circuit create low-cost communication interface using the existing power lines. This interface can be used for intelligent command and control systems such as:

- Lighting control
- Smart energy management,
- Automatic meter reading
- Home automation
- Smart Energy Management

Cypress's PLC solutions are:

- **CY8CPLC10**- Integrated PLC solution with I2C
- **CY8CPLC20**- Programmable PSoC PLC solution

The device CY8CPLC10 is meant for systems that have a microcontroller and need a PLC interface. The microcontroller with an I^2C interface can configure the modem and network protocol to transmit and receive data over the powerline. This application note explains how to do this. For more details on this device, refer the CY8CPLC10 datasheet.

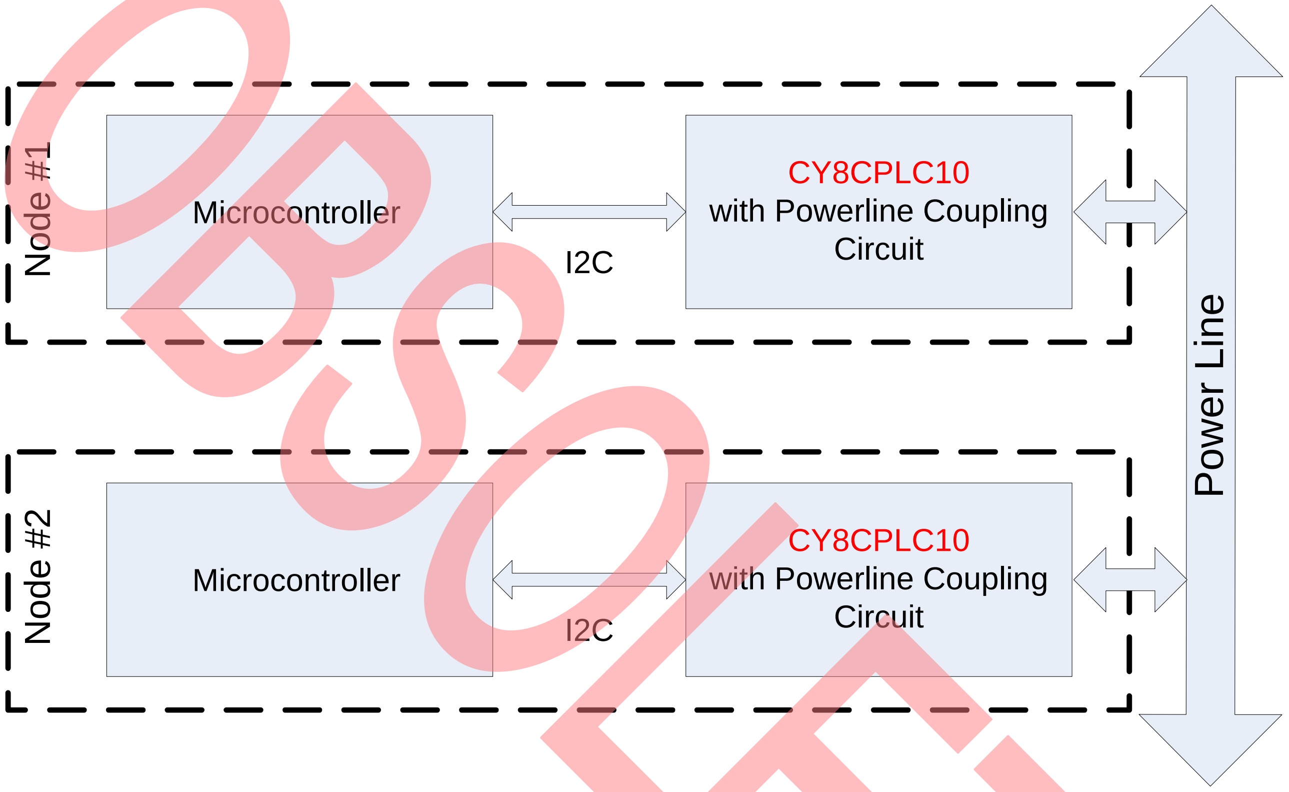
The device CY8CPLC20 is suitable for systems that need to have a more integrated solution. The CY8CPLC20 device combines the FSK modem and network protocol with the Cypress PSoC® 1 core. PSoC 1 contains a

24 MHz CPU with configurable digital and analog blocks (for example, ADCs, filters, counters, multiple communication interfaces, and so on). Therefore, the host application and PLC transceiver can be integrated in one device. For more details on this device, refer to the CY8CPLC20 datasheet.

Complete PLC evaluation kits, compliant with PLC standards in Europe and North America, are available at http://www.cypress.com/go/plc.

Figure 1 shows a system with two PLC nodes.

Figure 1. Cypress PLC Solution with External Microcontrollers



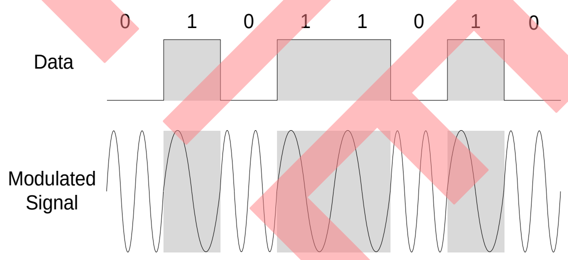
CY8CPLC10 Device Description

The CY8CPLC10 is an integrated PLC chip with the Powerline FSK Modem PHY, Powerline Network Protocol Stack, and I^{2}C Host Interface.

Powerline FSK Modem PHY

The heart of the CY8CPLC10 device is the frequency shift keying (FSK) modem. The FSK modulator sends digital data through two distinct frequencies; one frequency represents a digital 1 and the other represents a digital 0 (see Figure 1). The FSK demodulator must receive the transmitted analog signals and demodulate them to determine the correct sequence of 1s and 0s.

Figure 2. Sample FSK Waveform



Powerline Network Protocol Stack

The network protocol that runs on the processor supports -

- Bidirectional half-duplex communication
- Addressing

- 8-bit logical addressing supports up to 256 powerline nodes
- 16-bit extended logical addressing supports up to 65536 powerline nodes
- 64-bit physical addressing supports up to 2^64 powerline nodes
- Individual broadcast or group mode addressing

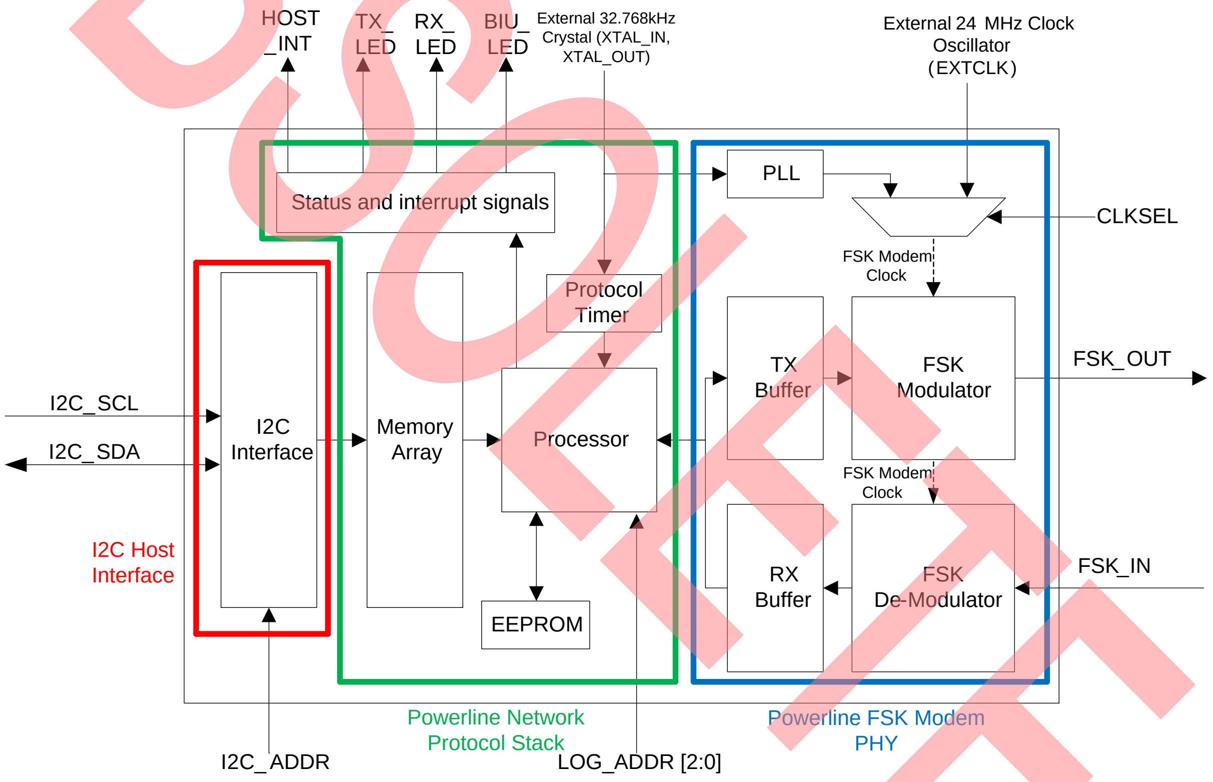
- Bidirectional half-duplex communication
- **B**and-**I**n-**U**se (BIU) and **C**arrier **S**ense **M**ultiple **A**ccess (CSMA)
- Verifies address and packet integrity (CRC) of received packets
- Transmits acknowledgments after receiving a valid packet, and automatically retransmits if a packet is dropped

I^{2}C Host Interface

The modem and network protocol are configured by a memory array that is controlled by a host microcontroller through an I^{2}C interface.

For a full list of features and description of device functionality, refer to the CY8CPLC10 datasheet. Refer Figure 3 for a block diagram of the CY8CPLC10 device.

Figure 3. CY8CPLC10 Block Diagram



PLC Memory Array

The following registers in the device's memory array control the modem and network protocol.

0x00 - 0x05: Network Configuration

0x06 - 0x2F: Transmitter Settings

0x30 - 0x33: Modem Configuration

0x34 - 0x3F: Reserved

0x40 - 0x68: Receiver Settings

0x69 - 0x71: Local Physical Address

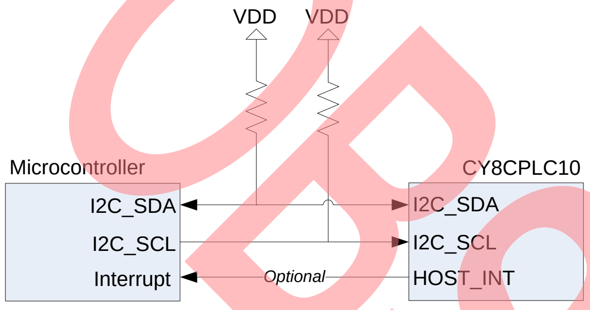
0x72 : Local Firmware Version Number

Refer CY8CPLC10 datasheet for details of all of the registers. This application note will focus on the registers for setting up a complete system.

Interfacing the Host Microcontroller to the CY8CPLC10 Device

The microcontroller controls the CY8CPLC10 device through an I^{2}C interface. Figure 4 shows the I^{2}C connection between the devices.

Figure 4. Microcontroller to PLC Device Interface



In addition to the I^{2}C interface, the CY8CPLC10 device has a digital output HOST_INT that indicates that a PLC event has occurred. The microcontroller can wait for this pin to go HIGH, instead of continuously polling the status register through the I^{2}C bus. More information on using this pin is provided in the section "Application Details".

I^{2}C Write Messages

The PLC memory array registers are modified by sending I^{2}C messages from the host. The structure of an I^{2}C write packet is shown in Table 1.

Table 1. I^{2}C Write Packet for Writing Data

	7	6	5	4	3	2	1	0
0	I^{2}C Address							W/R
1	Memory Array Offset							
2+	Write Data Payload							

BYTE 0

The least significant bit is the Read/Write bit. This indicates whether the packet is for writing data (logic '0') or for reading data (logic '1'). The remaining 7 bits contain the I^{2}C address of the slave device. The I^{2}C address for the CY8CPLC10 device can be set to -

- 0x01 when I2C_ADDR pin is High
- 0x7A when I2C_ADDR pin is Low

BYTE 1

This byte contains the offset of the memory array where the first byte of the data payload needs to be written.

BYTE 2

The third byte contains the data that will be written. Note that multiple bytes can be written at one time by sending additional bytes after the third byte. The PLC device will know that the I^{2}C write is complete when it receives the stop bit.

Example

To set the TX_Gain register (offset 0x32) to 0x0B and set the RX_Gain register (offset 0x33) to 0x06, you would send the packet shown in Table 2. The I^{2}C address used is 0x01 and the read/write bit is 0. Since TX_Gain and RX_Gain are in consecutive registers, the starting offset is set to the first register offset (0x32) and then the two data bytes are written.

Table 2. I^{2}C Write Packet to Registers 0x32-0x33

Byte	7	6	5	4	3	2	1	0
0	0b0000001							0
1	0x32							
2	0x0B							
3	0x06							

I^{2}C Read Messages

To read from the memory array, perform the following two steps.

First, the offset to read from must be set. To do this, send an I^{2}C write packet that contains the starting offset, as shown in Table 3.

Table 3. I^{2}C Write Packet for Setting Up to Read Data

Byte	7	6	5	4	3	2	1	0
0	I^{2}C Address							0
1	Memory Array Offset							

The second step is to read the data from the CY8CPLC10 device, by sending an I^{2}C Read Packet, as shown in Table 4. When the CY8CPLC10 receives the first byte of the I^{2}C read packet, it will see that the read/write bit is '1'. The CY8CPLC10 device will respond by sending the data.

Table 4. I^{2}C Read Packet

Byte	7	6	5	4	3	2	1	0
0	I^{2}C Address							1
1+	Read Data (from CY8CPLC10)							

As an example, assume that the CY8CPLC10 device has just received two bytes of data (0x0B and 0x06) at offset 0x32 to set the Tx_Gain (offset 0x32) and Rx_Gain (offset 0x33) registers. To read these two bytes of data, first send an I^{2}C write packet to set the starting offset to the Tx_Gain register (offset 0x32), as shown in Table 5.

Table 5. I^{2}C Write Packet for Reading RX_Data (0x32)

Byte	7	6	5	4	3	2	1	0
0	0b0000001							0
1	0x32							

Now read two bytes of data by sending an I^{2}C read packet as shown in Table 6. Recall that the first byte is sent by the host. The I^{2}C address is 0x01 and the read/write bit is '1'. The subsequent bytes are sent by the CY8CPLC10 device and contain the two bytes of data that is stored in Tx_Gain (offset 0x32) and Rx_Gain (offset 0x33) registers.

Table 6. I^{2}C Read Packet of Two Bytes

Byte	7	6	5	4	3	2	1	0
0	0b0000001							1
1	0x0B							
2	0x06							

External Host for CY8CPLC10

Controlling the CY8CPLC10 device to send and receive messages is simply a matter of writing to and reading from the PLC memory array. This section explains how to configure the modem and network protocol, how to send messages, check the transmission status, and how to check for received messages and read the received data.

Configuring the Device

Follow the steps below to configure the modem and network protocol:

1. Set Local_LA_LSB register (offset 0x01):

   Set the logical address of the device. This value should be unique for every node in the network. The logical address can also be set through hardware with the 3-bit LOG_ADDR (Logical Address) Port; however, it is overwritten when this register is set in software.

2. Set PLC_Mode register (offset 0x05):

   Enable the transmitter and receiver (TX_Enable and RX_Enable). The other settings in this register are explained in the CY8CPLC10 datasheet.

3. Set Threshold_Noise register (offset 0x30):

   Set the BIU detection threshold (BIU_Threshold_Constant). The BIU threshold determines the level to differentiate noise from a real PLC signal when acquiring the powerline (i.e., **CSMA**). The recommended starting value is 87 dBuVrms. Increase this threshold if the device is unable to acquire the powerline (Status_Busy bit in the INT_Status register).

   This threshold can be set automatically by setting the Auto_BIU_Threshold bit. The BIU_Threshold_Constant value will automatically update to one value above the noise measured on the FSK_IN pin. Note that there should not be any PLC traffic on the line while running the automatic threshold function.

4. Set Modem_Config register (offset 0x31):

   Ensure that the baud rates (Modem_BPS) and FSK deviation (Modem_FSKBW) are the same for all nodes.

   The recommended values are 2400 bps and 3 kHz deviation. For lower baud rates, check the CY8CPLC10 datasheet for the settings required for the Modem_TXDelay parameter.

5. Set TX_Gain register (offset 0x32):

   This value sets the amplitude of the transmitted signal on the FSK_OUT pin. The recommended value for CENELEC compliant high voltage reference designs is 125 mVp-p; otherwise the recommended value is 1.55 Vp-p.

6. Set RX_Gain register (offset 0x33):

   Set the receiver gain for amplifying the received signal on the FSK_IN pin. The recommended sensitivity value is 250 uVrms. If the system is noisy, reduce the sensitivity (e.g., 600 dBuVrms).

Transmitting Messages

After configuring the device, you are now ready to send data. This section describes how to transmit messages.

1. Set TX_Config register (offset 0x07):

   Configure the addressing mode (TX_SA_Type, TX_DA_Type). To enable acknowledgement set the TX_Service_Type bit. In the field set the field TX_Retry for the number of retransmissions.

2. Set TX_DA register (offset 0x08 – 0x0F):

   Set the destination address. Ensure the appropriate addressing type (TX_DA_Type) in the TX_Config (Offset 0x07) is set.

3. Set TX Command ID register (offset 0x10):

   Set the type of message that needs to be sent. To send normal data, the command ID is 0x09.

The list of command IDs are in the CY8CPLC10 data sheet. Custom user-defined command IDs (0x30-0xFF) can also be used. The remote configuration commands are explained in the section "Remote Node Configuration".

4. Set TX_Data register (Offset 0x11 – 0x2F):

   Write the payload data here.

5. Set TX_Message_Length register (offset 0x06):

   Set the length of the payload. To initiate transmission, set the Send_Message bit (bit 7) to '1'.

6. Read INT_Status register(offset 0x69):

   This register indicates the status of the transmission.

   - If the message was successfully sent, the Status_TX_Data_Sent bit will be '1'.
   - If an acknowledgment was expected, but was not received, the Status_TX_NO_ACK bit will be '1'.
   - If the device was unable to acquire the powerline to transmit, the Status_BUSY bit will be '1'.

   Optional: Instead of polling the INT_Status register, the status of the PLC device can be checked by reading the HOST_INT pin of the device. When an event occurs (for example, transmit success), the HOST_INT pin will be set to '1'. Then, the INT_Status register can be checked via I^2C. The HOST_INT pin is set back to '0' when the INT_Clear bit in the INT_Enable register (offset 0x00) is set to '0', or when the New_RX_Msg bit in the RX_Message_INFO register (offset 0x40) is set to '0'.

Receiving Messages

1. Read INT_Status register (offset 0x69):

   The receive status bits in this register indicate if a new message is received.

   - If a message was received, the Status_RX_Data_Available bit will be '1'.
   - If the receive buffer is full and another message is received, the Status_RX_Packet_Dropped will be set to '1'.

2. Read RX_Message_INFO register (offset 0x40):

   Read this register when a message is received.

   - RX_SA_Type: Indicates the source address type
   - RX_DA_Type: Indicates the destination address type
   - RX_Msg_Length: Indicates the message length

3. Read RX_SA (offset 0x41 – 0x47):

   Read this register to get the source address of the message. The number of bytes is determined by the RX_SA_Type that is read in the previous step.

4. Read RX_CommandID (offset 0x49):

   This indicates the received Command ID.

5. Read RX_Data (offset 0x4A – 0x68):

   Read this register to get the received data. The number of bytes to read depends on the RX_Msg_Length bits read earlier.

6. Set RX_Message_INFO register (offset 0x40):

   Set the New_RX_Msg bit to '0' to clear the receive buffer to be able to receive a new message. When this bit is cleared, the receive status bits in the INT_Status byte will also be cleared.

   Optional: Instead of polling the INT_Status register, the status of the PLC device can be checked by reading the HOST_INT pin of the device. When an event occurs (for example, new message received), the HOST_INT pin will be set to '1'. Then, the INT_Status register can be checked for a specific receiver event via I^2C. The HOST_INT pin is set back to '0' when the INT_Clear bit in the INT_Enable register (offset 0x00) is set to '0', or when the New_RX_Msg bit in the RX_Message_INFO register (offset 0x40) is set to '0'.

Remote Node Configuration

A PLC device can be configured remotely by using the Remote Commands. For example, Node-1 can change the logical address of another device, Node-2 by sending a SetRemote_LogicalAddr command (TX_Command_ID = 0x04) with the new intended address as the payload. These messages are sent in the same way as explained in the "Transmitting Messages" section.

Data can also be requested from a remote node. For example, remote node's Physical Address can be requested by sending the command GetRemote_PhysicalAddr (TX_Command_ID = 0x06). In this case, after the transmission is successful, follow the steps in "Receiving Messages" to get the requested data.

In systems, where remote node configuration is not allowed, the PLC device can be locked by setting the Lock_Configuration bit in the PLC_Mode register (offset 0x05).

Code Example

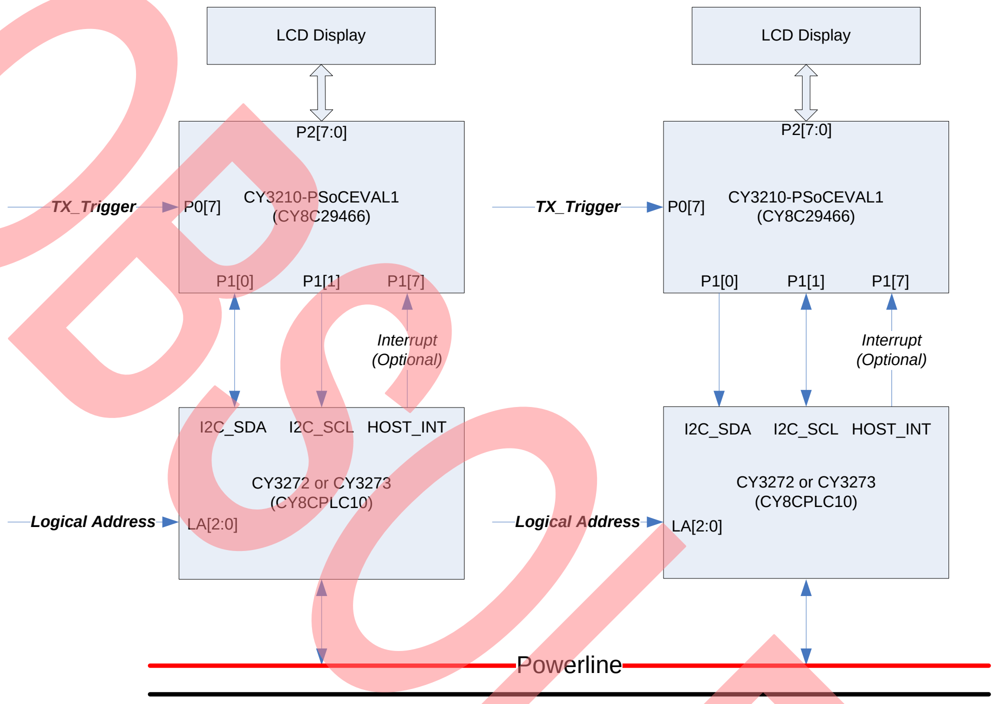
This section describes how to develop a code example that uses a Cypress Programmable System-on-Chip (PSoC) device as the host to control the CY8CPLC10 device. Other third party micro-controllers with an I^2C interface can also use this code with the required modifications.

The system diagram for evaluating this code example is shown in Figure 5. Two nodes that both have a CY3210-PSoCEVAL1 evaluation board (which is running this code example) connected to a CY3272 or CY3273 board (which performs the PLC communication). One node will transmit data, while the other receives the data. The CY3210-

PSoCEVAL1 kit is available at the following: www.cypress.com/go/CY3210-PSoCEval1.

To skip the application's detailed description and start evaluating the application and hardware, go to section "Evaluating on Hardware".

Figure 5. Code Example System Diagram



Application Details

The host microcontroller (PSoC-1 in this case) should have the following resources:

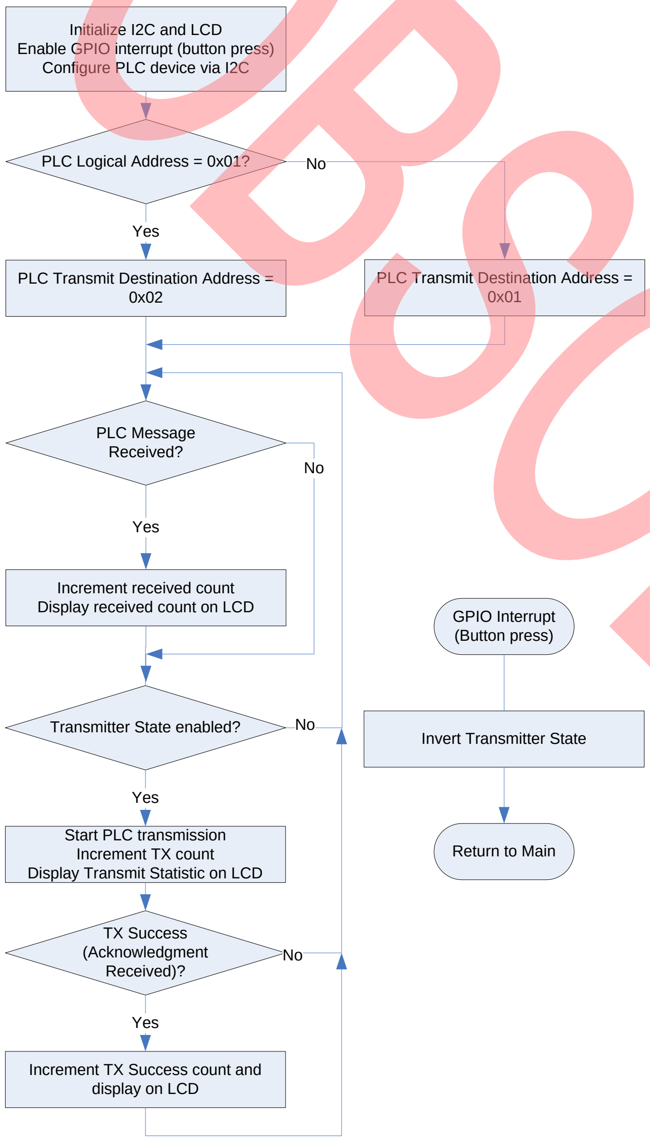
- I^{2}C Master: Interfaces the Host device (PSoC-1 in this case) to the CY8CPLC10 device
- LCD driver (optional): Controls an external LCD, which is used for displaying the transmit and receive statistics.
- GPIO interrupt: When the signal on pin P0[7] transitions from logic '1' to logic '0', an interrupt will occur to trigger the device to toggle the transmitter state.

A detailed flow of the application's algorithm is shown in Figure 5.

1. Host's resources are initialized.
2. The CY8CPLC10 device is configured (with the I^{2}C interface) to be able to transmit and receive messages.
3. The 3 GPIO pins LOG_ADDR_0, LOG_ADDR_1 and LOG_ADDR_2 set the logical address of the CY8CPLC10 device. The host reads this logical address. If the logical address is 0x01, then the destination address that data packets will be sent to is set to 0x02. If the local logical address is other than 0x01, the destination address will be set to 0x01. This way, all data traffic will be sent to address 0x01, which will resemble a star network.
4. After the initial configuration, the Host will continuously loop to check for a received message. If a packet is received, the RX_Count is incremented and the LCD is updated. The source address of a received packet is copied to the TX_DA register. For example, if CY8CPLC10 device 0x01 receives a message from device 0x03, it will transmit its next message to 0x03.
5. When a GPIO interrupt occurs (e.g. a button press) on pin P0[7], the transmitter state is inverted. If the transmitter is enabled then a packet without any

payload is transmitted to the destination address in the TX_DA register (offset 0x08 – 0x0F). TX_Count is incremented the LCD is updated. If an acknowledgement is received, then the TX_Success_Count is incremented the LCD is updated. The PSoC Host will continue to initiate data transmission until the GPIO interrupt occurs again.

Figure 6. Code Example Flow Chart


Initialize I2C and LCD
Enable GPIO interrupt (button press)
Configure PLC device via I2C
PLC Logical Address = 0x01?
No
Yes
PLC Transmit Destination Address = 0x02
PLC Transmit Destination Address = 0x01
PLC Message Received?
No
Yes
Increment received count
Display received count on LCD
GPIO Interrupt (Button press)
Transmitter State enabled?
No
Invert Transmitter State
Yes
Start PLC transmission
Increment TX count
Display Transmit Statistic on LCD
Return to Main
TX Success (Acknowledgment Received)?
No
Yes
Increment TX Success count and display on LCD

The application contains an option for using another digital input (pin P1[7]) for checking event status updates. If enabled, connect this pin to the HOST_INT pin on the PLC device. To check if a message is received or if a transmission is complete, the Host device will first wait until pin P1[7] is a logic '1'. Then, it will read the PLC device's status register (offset 0x69) using I^{2}C.

This is optional and is used to reduce the traffic on the I^{2}C bus. If bus traffic is not a concern, it is okay to continuously poll the PLC device's status register until it updates. By default, this HOST_INT monitoring function is disabled (see PLC_INT_BYPASS in the code for details).

Evaluating on Hardware

To evaluate this system, the following equipment and software is required:

1. 2 x CY3272 PLC HV Evaluation Kits

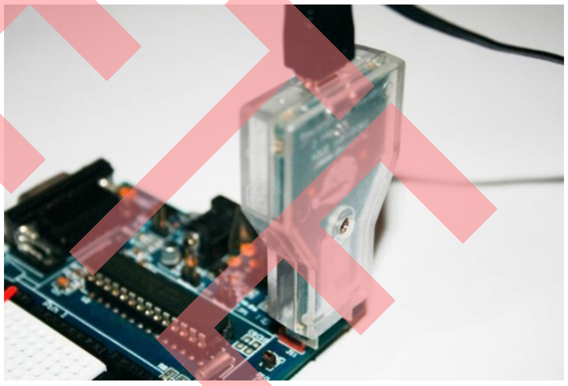
   or

   2 x CY3273 PLC LV Evaluation Kits
2. 2 x CY3210-PSoCEVAL1 PSoC Evaluation kits
3. 1 x PC running PSoC Programmer

To set up the first node for PLC communication:

1. On the CY3210, connect a wire from SW to P07. This connects the push button for toggling the transmitter state.
2. Program the CY3210-PSoCEVAL1
   a. Connect the USB cable from the PC to the MiniProg programmer (available with the CY3210-PSoCEVAL1 Kit).
   b. Connect the MiniProg programmer to the 5-pin header on the CY3210-PSoCEVAL1 board (see Figure 7).

   Figure 7. MiniProg Connection



   c. Open PSoC Programmer. Make sure the settings are as follows (see Figure 9):
      - Programming Mode = Power Cycle
      - AutoDetection = On
      - In PSoC Programmer, open the file "CY8CPLC10_Host.hex", which is in the attached code example.
   d. Click the program button.

3. Once the device is programmed, connect the 5-pin ribbon cable (included in the CY3272/CY3273 PLC kit) from the 5-pin header on the PLC board to the 5-pin header on the CY3210 board. Make sure that the red stripe on the cable lines up with the 'V' on the PLC board connector and the '+' on the CY3210 connector.
4. On the PLC board, connect three jumper shunts across the 2-pin headers SCL, SDA, and PWR. Note that the orientation of the SCL and SDA headers is different from the other headers.
5. On the PLC board, set all of the DIP switches to the OFF position. Then, set LA0 to the ON position. This will set the logical address to 0x01.
6. Connect the power supply cable from the mains to the PLC board. The blue LED on the PLC board should turn ON, the red PWR LED on the CY3210 board should turn ON. On the CY3210, the LCD should briefly display "I2C Test" followed by "LA=01 RX#=   ".
7. To setup the second board, follow the same steps as above, except in step 9, set LA0 to OFF and LA1 to ON. This will set the logical address to 0x02. The LCD will display "LA=02 RX#=   ".

Note If the CY3210 PWR LED does not turn on, make sure that the ribbon cable is oriented correctly and that there is a jumper shunt on the PLC board's PWR header. If the "I2C Test" message is still present after 5 seconds, make sure that the jumper shunts on the PLC board's SCL and SDA headers are oriented correctly.

Figure 8. Board Hardware Setup with CY3272 HV PLC EVK

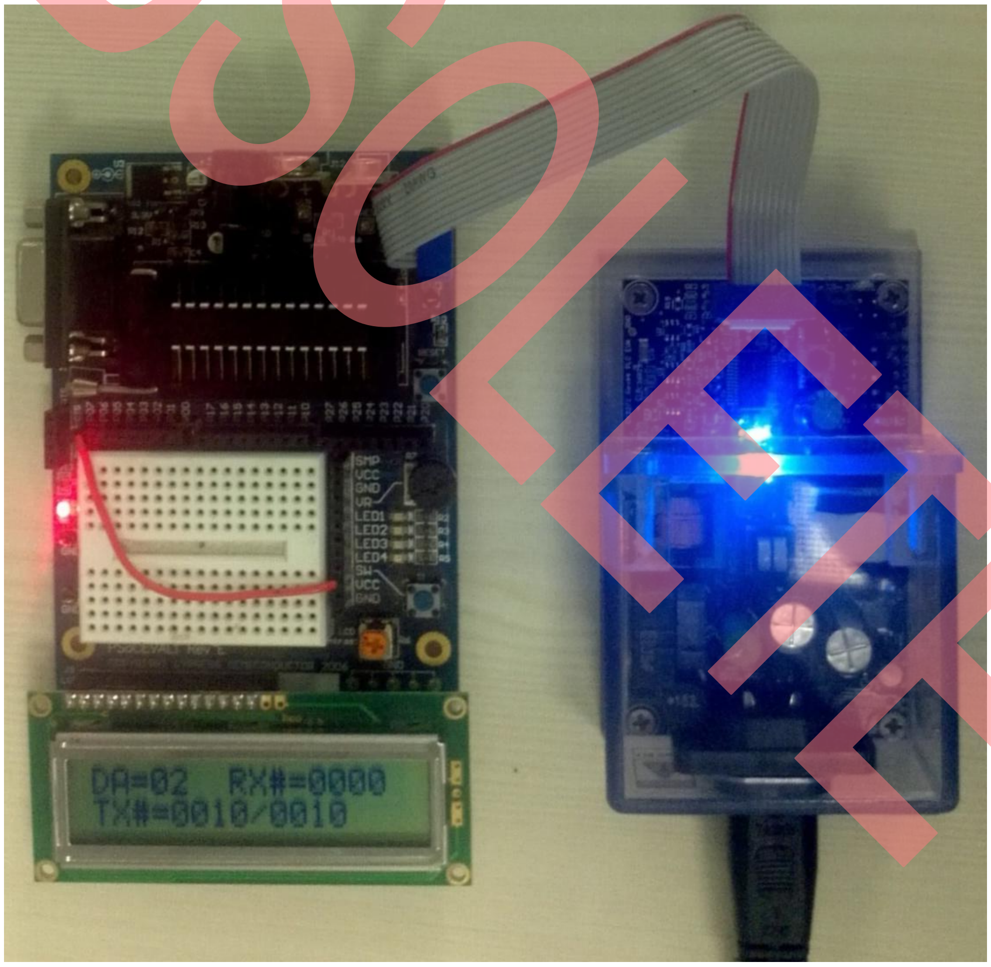
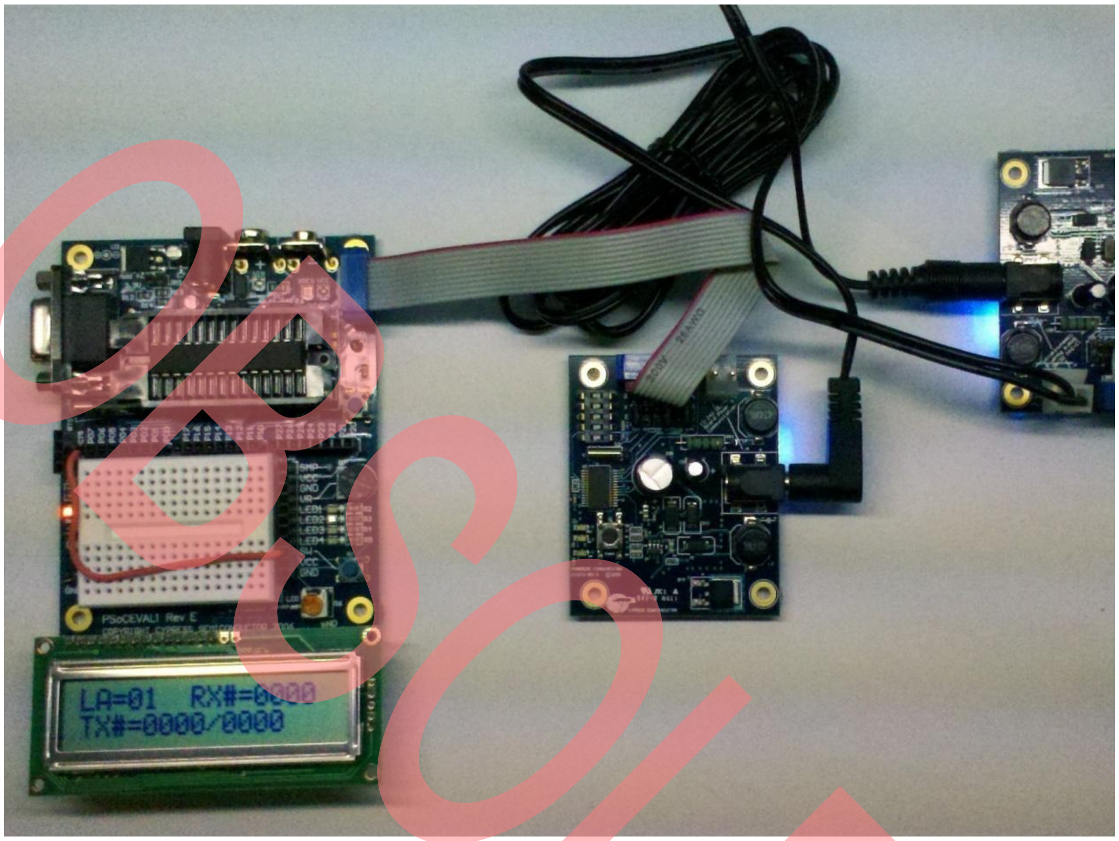


Figure 9. Board Hardware Setup with CY3273 LV PLC EVK



Note If using the CY3273 boards, instead of connecting a second power supply adapter, connect the daisy chain cable (included in the kit) from the first board to the second board (see the CY3273 quick start guide for pictures).

To send a message from the first board, press and release the SW push button on the CY3210 board. The number of transmitted packets 'ZZZZ' is updated on the LCD. If the messages are successfully acknowledged by the second board, the number of successfully transmitted packets 'YYYY' is updated. On the receiver board, the LCD should display the same value XX that is transmitted by the first board. The bottom row of the LCD should display the number of received packets 'XXXX'. To stop transmitting, press and release the SW button on the transmitter board or wait for 1000 packets to be transmitted.

LCD Display: LA=WW RX#=XXXX

TX#=YYYY/ZZZZ

Application Code

The intent of this application code is to provide users with a consistent interface to the CY8CPLC10. The driver was designed to interface with C written applications and consists of the following files:

delay.h	This file defines the function prototypes of the functions in the file *delay.asm* This file is for PSoC 1 and should be updated for other devices.
delay.asm	This file contains the functions that generate the delay This file is for PSoC 1 and should be updated for other devices.
plc_i2c.h	This file contains functions prototypes of the low level APIs for controlling a CY8CPLC10 device via I^{2}C commands, which are specific to Cypress PSoC. Update this file with I^{2}C APIs specific to the device used.

plc_i2c.c	This file contains low level APIs which provide I^{2}C access for register read from and writes to the CY8CPLC10 device. Update this file with I^{2}C APIs specific to the device used.
plc.h	This file contains the global constants and functions prototypes of the high level APIs for controlling a CY8CPLC10 device. This file should not require any update for various Host devices. More High Level functions can be added in this file for specific tasks by the user.
plc.c	This file contains high level APIs for controlling the CY8CPLC10 device This file should not require any update for various Host devices More High Level functions can be added in this file for specific tasks by the user.

Header files

To use the application code, include *delay.h*, *plc.h*, and *plc_i2c.h*.

```
#include "delay.h"
#include "plc.h"
#include "plc_i2c.h"
```

Type Declarations

The type declarations are described below.

BYTE	Used for 8-bit register values
WORD	Used for 16-bit register values
TRUE	(BOOL)1
FALSE	(BOOL)0

Modem Low Level Functions

PLC_I2C_Start()

```
void PLC_I2C_Start(void)
```

Summary:

Initialize the I^{2}C hardware block

Parameters:

None

Return:

None

PLC_I2C_WriteToOffset()

```
BYTE PLC_I2C_WriteToOffset(BYTE bOffset, BYTE *pbData, BYTE bDataLength)
```

Summary:

Write an I^{2}C message of bDataLength to the specified PLC memory offset.

The offset is the first byte written after the I^{2}C address byte. It is followed by the data to be written starting at the memory offset.

Parameters:

bOffset:	PLC memory offset to write to
pbData:	pointer to the data that will be written to the PLC device
bDataLength:	length of the data

Return:
Status of the I^{2}C communication

PLC_I2C_ReadFromOffset()

```
BYTE PLC_I2C_ReadFromOffset (BYTE bOffset, BYTE *pbData, BYTE bDataLength)
```

Summary:

Read an I^{2}C message of bDataLength from the specified PLC memory offset.

The offset is first written as a complete I^{2}C message to set up the memory offset to read from. Then, the data is read from the device.

Parameters:

bOffset:	PLC memory offset to read from
pbData:	pointer to the data that will be stored when read from the PLC device
bDataLength:	length of the data

Return:

Status of the I^{2}C communication

PLC_I2C_IsUpdated()

```
BYTE PLC_I2C_IsUpdated(void)
```

Summary:

Responds TRUE if the PLC device has an event update

Parameters:

None

Return:

TRUE if the PLC device has an event update. FALSE otherwise

Modem High Level Functions

PLC_Init()

```
BYTE PLC_Init(void)
```

Summary:

Initialize the PLC interface

Parameters:

None

Return:

I2C_SUCCESS if I^{2}C communication was successful. I2C_FAIL otherwise

PLC_SetDestinationAddress()

```
BYTE PLC_SetDestinationAddress (BYTE bAddrType, BYTE *pbDestinationAddress)
```

Summary:

Sets the PLC Destination Address according to the address type

Parameters:

bAddrType	Destiation Address Type. Refer to TX_DA_Type constants
pbDestinationAddress	pointer to the destination address

Return:

Status of the I^{2}C communication and address type validity

PLC_TransmitPacket()

`BYTE PLC_TransmitPacket(BYTE bCommand, BYTE *pbTXData, BYTE bDataLength)`

Summary:

Initiates a PLC packet transmission of a data packet of specified length

Parameters:

bCommand:	Command ID of the PLC message
pbTXData:	pointer to the data payload that will be in the PLC message
bDataLength:	length of the data payload

Return:

Status of the PLC communication

PLC_IsPacketReceived()

`BYTE PLC_IsPacketReceived(void)`

Summary:

Responds TRUE if a message has been received by the PLC device

Parameters:

None

Return:

TRUE if a message is received. FALSE otherwise

About the Author

Name:	Aditya Yadav
Title:	Applications Engineer
Contact:	adiy@cypress.com

Document History

Document Title: AN52478 - Designing an External Host Application for Cypress's Powerline Communication IC CY8CPLC10

Document Number: 001-52478

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2725925	ROSG	06/29/2009	New application note.
*A	2739181	ROSG	07/16/2009	Corrected title on the last page
*B	2758927	ROSG	09/02/2009	Updated Figure 4 – Connections for Master and Slave
*C	3175968	FRE	02/17/2011	Added Associated Application Notes in page 1 as AN55427. Updated Software Version in page 1 as PSoC Designer 5.1. Updated Abstract. Renamed the section "CY8CPLC10 – Cypress's Solution for Powerline Communication" as Introduction and updated the section. Deleted the sections "Key Features" and "Designing Using the CY8CPLC10". Added CY8CPLC10 Device Description. Added Interfacing the Microcontroller to the PLC Device. Deleted the section "Local and Remote Commands". Renamed the section "Building the External Host Application" as Writing the External Host Application, merged with the section "Quick Start Guide" and updated the same section. Simplified Code Example and added an option to communicate with the PLC Control Panel GUI. Updated Summary. Improved the flow of the document.
*D	3685329	ADIY	07/19/2012	Updated for PSoC Designer 5.2 SP1 Release. Added application code section. Updated Figure 9 for the new firmware. Removed the ADC from the associated project. Added Figure 2 and Figure 9. Updated Figure 1, Figure 3, and Figure 5. Updated template.
*E	4091038	ADIY	08/08/2013	Updated firmware for PSoC Designer 5.4 release. Updated Worldwide Sales and Design Support.
*F	4443919	ROIT	07/15/2014	Obsolete this AN as there is no PLC kit with CY8CPLC10 device.

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