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.

AN54416

Using CY8CPLC20 in Powerline Communication (PLC) Applications

Author:Aditya Yadav
Associated Project: Yes
Associated Part Family: CY8CPLC20
Software Version: PSoC® Designer™5.4 CP1
Related Application Notes: For a complete list of the application notes, click here.

AN54416 describes how to use the Powerline Transceiver (PLT) User Module on the CY8CPLC20 device for PLC applications. The application note also includes a spreadsheet to estimate the power consumption by CY8CPLC20 and focuses on four code examples. The first example provides steps to develop an example project to communicate between two nodes on the powerline. The second example discusses how to develop a UART Host interface for CY8CPLC20. The third example discusses how to develop an I^2C Host interface for CY8CPLC20. The fourth shows how to use CY8CPLC20 with average low-power consumption of less than 50 mW.

Contents

Introduction

The Cypress PLC family is a single-chip solution for powerline communication (PLC). Cypress's PLC solution combined with a simple powerline coupling circuit form a low-cost communication interface using the existing power lines. This interface can be used for intelligent command and control systems such as:

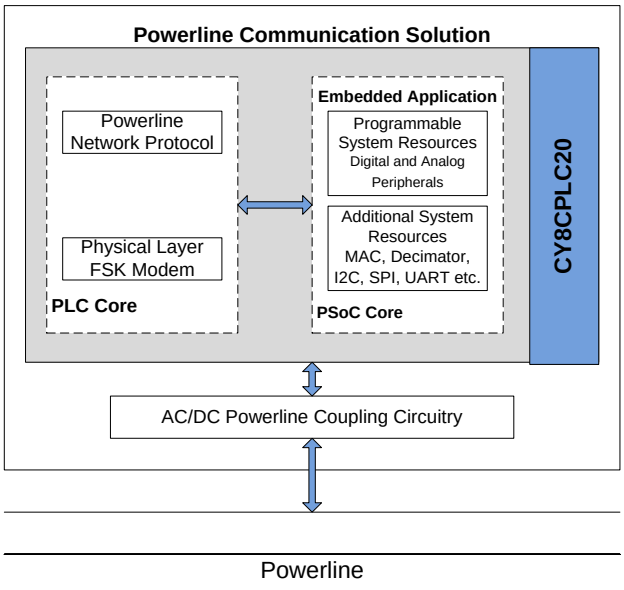
- Lighting control
- Automatic meter reading
- Home automation

The CY8CPLC20 device combines the FSK modem and network protocol with the Cypress PSoC® core. The PSoC contains a 24-MHz CPU with configurable digital and analog blocks.

Figure 1 shows a block diagram of the Cypress CY8CPLC20 PLC solution. To interface the CY8CPLC20 device to the powerline, a coupling circuit is required.

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

Figure 1. Cypress CY8CPLC20 PLC Solution



This application note also explains how to use the Powerline Transceiver (PLT) User Module (UM) on the CY8CPLC20 device with the help of an example project to communicate between two nodes on the powerline.

CY8CPLC20 Device Description

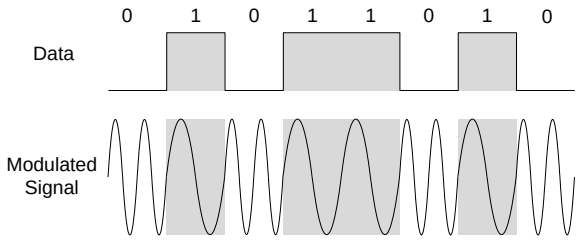
The CY8CPLC20 is a Programmable Powerline Communication chip with a:

- Powerline FSK Modem PHY
- Powerline NetworkProtocol Stack.

Powerline FSK Modem PHY

The heart of the CY8CPLC20 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 2). 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*



* This diagram is only for conceptual understanding and is not to scale

Powerline NetworkProtocol Stack

The network protocol that runs on the processor supports:

- Bidirectional half-duplex communication
- Master-slave or peer-to-peer network topologies
- Multiple masters on powerline network
- 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
- **C**arrier **S**ense **M**ultiple **A**ccess (CSMA)

  The protocol provides the random selection of a period between 85 ms and 115 ms (out of seven possible values in this range) in which the band-in-use (BIU) detector must indicate that the line is not in use before attempting a transmission.

- **B**and-**I**n-**U**se (BIU)

  A BIU detector, as defined under CENELEC EN 50065-1, is active whenever a signal that exceeds 86 dBuVrms anywhere in the range 131.5 kHz to 133.5 kHz is present for at least 4 ms. This threshold can be configured for different end-system applications not requiring CENELEC compliance. The modem tries to retransmit after every 85 ms to 115 ms when the band is in use. The transmitter times out after 1.1 seconds to 3.5 seconds (depending on the noise on the Powerline) and generates an interrupt to indicate that the transmitter was unable to acquire the Powerline.

  Note that for non-CENELEC compliant systems, the BIU interval can be modified for improved performance by modifying the Timing_Config register. Refer the PLT UM datasheet for more details.

- Verifies the address and packet integrity (CRC) of received packets

- Transmits acknowledgments after receiving a valid packet, and automatically retransmits if a packet is dropped

Estimating CY8CPLC20 Power Consumption

This section describes how to estimate power consumption for a CY8CPLC20 PSoC Designer™ project, which is done using the spreadsheet available with this document. The spreadsheet estimates power consumption of the CY8CPLC20 device in different configurations.

PLT User Module

Cypress provides the pre-configured PLT UM for powerline communication implementation in its development software, PSoC Designer. A developer can place and run this PLT UM in a CY8CPLC20 device.

The PLT UM occupies nine analog and eight digital blocks in the CY8CPLC20 device. It has the following settings that affect power consumption:

Analog Power = **SC On/Ref High**

OpAmpBias = **1 (High)**

A_Buffer Power = **1 (High)**

When placed and running, the PLT UM consumes ~35 mA in receive mode and ~29 mA in transmit mode. This does not include the CPU current, which ranges from 5.8 mA to 8.8 mA, depending on the CPU frequency.

To run the PLT UM with less power consumption, see Low-Power Firmware for CY8CPLC20.

Spreadsheet Description

The first tab of the associated spreadsheet (AN54416_Power_Consumption.xls) lists the available resources when the PLT UM is placed in a project for a CY8CPLC20 device. For additional resources, the calculator adds the additional current to the PLT current to provide the total current. See Figure 3.

The second tab, shown in Figure 4, of the associated spreadsheet lists the available resources when the PLT user module is not placed. This is useful when implementing dynamic reconfiguration in a project. In this case, all resources are available and settings are configurable. For more information on dynamic reconfiguration, see application note AN2104 - PSoC®1 - Dynamic Reconfiguration With PSoC® Designer™.

The fields in the spreadsheet are color coded as follows:

- Data entry cells are yellow
- Resultant current cells are green
- If the value is out of range, the cell turns red

Each field in the spreadsheet is explained as follows. As the power of the PLT user module is already calculated, these fields (except CPU current) only apply when adding additional user modules (for example, PWM, ADC, and so on).

CPU Current

Enter the frequency of the CPU in the CPU Frequency field. The estimated CPU current will appear in the corresponding green cell. Unused clock dividers (VC1, VC2, and VC3) should be set to the maximum value to minimize power.

Note When using the PLT user module, the allowed values are between 3 MHz (SysClk/8) and 24 MHz (SysClk/1) and the clock dividers (VC1, VC2, and VC3) cannot be changed.

Digital Block Current

Enter the input clock value for each of the digital blocks into the corresponding cells. For example, if a PWM is added and has an input clock of VC1, which is set to SysClk/6, then the frequency would be 4 MHz.

Row Current

Enter the frequency values of the used output rows into the corresponding cells. Note that 12 MHz is the maximum clock speed for rows.

GPIO Current

The GPIO current depends on the GPIO frequency (“GPIO Clk”) and the number of GPIOs at the same frequency (“GPIO Count”). GPIO frequency has a 12-MHz limit similar to the row frequency. For example, if three pins have a 2-MHz signal, then enter ‘3’ for “GPIO Count” and ‘2’ for “GPIO Clock”.

Analog Block Current

The analog block power level and opamp bias level determine the current consumed by analog blocks. Enter the opamp bias level and power level of the occupied analog blocks into the corresponding yellow cells. The power level is set when the user module is started in code.

Figure 3. Power Consumption with PLT User Module

Estimated Power Calculations for CY8CPLC20 Family Devices (with the PLT UM placed and running)

Version 1.1

Enter parameters in yellow
See estimated result in green
Indicates something is wrong

Fixed Settings for PLT UM

Reference Power	SC On/Ref High
OpAmpBias	1 (High)
A_Buffer Power	1 (High)

CPU Frequency	3.000	MHz	**SysClkx2Disable**	1	
Vdd	5.00	V			
CPU Current	5.833	mA			

Digital Block	**Block Clk (MHz)**			**GPIO Count**	**GPIO Clk (MHz)**		
DBB20	0.000	0.000	mA	0	0.000	0.000	mA
DBB21	0.000	0.000	mA	0	0.000	0.000	mA
DCB22	0.000	0.000	mA	0	0.000	0.000	mA
DCB23	0.000	0.000	mA	0	0.000	0.000	mA
DBB30	0.000	0.000	mA	0	0.000	0.000	mA
DBB31	0.000	0.000	mA	0	0.000	0.000	mA
DCB32	0.000	0.000	mA	0	0.000	0.000	mA
DCB33	0.000	0.000	mA	0	0.000	0.000	mA
Digital Block Current		0.000	mA	0	0.000	0.000	mA
				0	0.000	0.000	mA
				0	0.000	0.000	mA
				0	0.000	0.000	mA
				0	0.000	0.000	mA
Row	**Row Clk (MHz)**			0	0.000	0.000	mA
Row_2_Output_0	0.000	0.000	mA	0	0.000	0.000	mA
Row_2_Output_1	0.000	0.000	mA	GPIO Current		0.000	mA
Row_2_Output_2	0.000	0.000	mA				
Row_2_Output_3	0.000	0.000	mA				
Row_3_Output_0	0.000	0.000	mA				
Row_3_Output_1	0.000	0.000	mA				
Row_3_Output_2	0.000	0.000	mA				
Row_3_Output_3	0.000	0.000	mA				
Row Current		0.000	mA				

Analog Block	**Power Level (0-3)**		
ACB02	0	0.000	mA
ASC12	0	0.000	mA
ASD22	0	0.000	mA
Analog Block Current		0.000	mA
Analog Output Buffer			
AnalogOutBuf_2	0	0.000	mA
Analog Output Buffer Current		0.000	mA

	RX Mode	**TX Mode**	
Estimated PLT Current	35.000	29.000	**mA**
Estimated Additional Current	5.833	5.833	**mA**
Estimated Total Device Current	40.833	34.833	**mA**
Estimated Total Device Power	204.165	174.165	**mW**

Figure 4. Power Consumption without PLT User Module

Estimated Power Calculations for the CY8CPLC20 Device (with the PLT UM not placed)

Version 1.1

Enter parameters in yellow
See estimated result in green
Indicates something is wrong

CPU Frequency	3.000	MHz		**SysClkx2Disable**	1									
Vdd	5.00	V												
CPU Current	5.833	mA												
										OpAmp Bias	0			
	Block Clk			**GPIO**	**GPIO Clk**						**Power**			
Digital Block	**(MHz)**			**Count**	**(MHz)**					**Analog Block**	**Level (0-3)**			
DBB00	0.000	0.000	mA	0	0.000	0.000	mA			ACB00	0	0.000	mA	
DBB01	0.000	0.000	mA	0	0.000	0.000	mA			ACB01	0	0.000	mA	
DCB02	0.000	0.000	mA	0	0.000	0.000	mA			ACB02	0	0.000	mA	
DCB03	0.000	0.000	mA	0	0.000	0.000	mA			ACB03	0	0.000	mA	
DBB10	0.000	0.000	mA	0	0.000	0.000	mA			ASC10	0	0.000	mA	
DBB11	0.000	0.000	mA	0	0.000	0.000	mA			ASD11	0	0.000	mA	
DCB12	0.000	0.000	mA	0	0.000	0.000	mA			ASC12	0	0.000	mA	
DCB13	0.000	0.000	mA	0	0.000	0.000	mA			ASD13	0	0.000	mA	
DBB20	0.000	0.000	mA	0	0.000	0.000	mA			ASD20	0	0.000	mA	
DBB21	0.000	0.000	mA	0	0.000	0.000	mA			ASC21	0	0.000	mA	
DCB22	0.000	0.000	mA	0	0.000	0.000	mA			ASD22	0	0.000	mA	
DCB23	0.000	0.000	mA	0	0.000	0.000	mA			ASC23	0	0.000	mA	
DBB30	0.000	0.000	mA	0	0.000	0.000	mA			Analog Block Current		0.000	mA	
DBB31	0.000	0.000	mA	0	0.000	0.000	mA							
DCB32	0.000	0.000	mA	0	0.000	0.000	mA							
DCB33	0.000	0.000	mA	0	0.000	0.000	mA							
Digital Block Current		0.000	mA	GPIO Current		0.000	mA							
	Row Clk			**Reference Power**						**Analog Buffer Power**	0			
Row	**(MHz)**			All Off	1									
Row_0_Output_0	0.000	0.000	mA	SC Off/Ref Low	0	0.000	mA			**Analog**				
Row_0_Output_1	0.000	0.000	mA	SC Off/Ref Med	0	0.000	mA			**Output Buffer**				
Row_0_Output_2	0.000	0.000	mA	SC Off/Ref High	0	0.000	mA			AnalogOutBuf_0	0	0.000	mA	
Row_0_Output_3	0.000	0.000	mA	SC On/Ref Low	0	0.000	mA			AnalogOutBuf_1	0	0.000	mA	
Row_1_Output_0	0.000	0.000	mA	SC On/Ref Med	0	0.000	mA			AnalogOutBuf_2	0	0.000	mA	
Row_1_Output_1	0.000	0.000	mA	SC On/Ref High	0	0.000	mA			AnalogOutBuf_3	0	0.000	mA	
Row_1_Output_2	0.000	0.000	mA	Reference Circuit Current		0.000	mA			Analog Output Buffer Current		0.000	mA	
Row_1_Output_3	0.000	0.000	mA											
Row_2_Output_0	0.000	0.000	mA							**Estimated Total Device Current**		**5.833**	**mA**	
Row_2_Output_1	0.000	0.000	mA											
Row_2_Output_2	0.000	0.000	mA							**Estimated Total Device Power**		**29.165**	**mW**	
Row_2_Output_3	0.000	0.000	mA											
Row_3_Output_0	0.000	0.000	mA											
Row_3_Output_1	0.000	0.000	mA											
Row_3_Output_2	0.000	0.000	mA											
Row_3_Output_3	0.000	0.000	mA											
Row Current		0.000	mA											

For example, “PGA_1_Start(PGA_1_HIGHPOWER);” sets the PGA’s analog block current to high power. The opamp bias level is set in the Global Resources window.

To minimize power consumption, use the lowest power level that allows the analog block to meet system requirements at the specified column clock frequency. Refer to the user module’s datasheet (for example, PGA) to determine the performance impact of different power levels.

Note You cannot modify the opamp bias level when the PLT UM is placed.

Analog Output Buffer Current

Analog output buffer power consumption is affected by the analog buffer power level. Select the analog output buffer power by writing ‘1’ to the appropriate cell. To minimize power, use high power for the buffer only if the output frequency is greater than 50 kHz.

Note The analog output buffer current is not modifiable when the PLT user module is placed.

Reference Circuit Current

Reference power should be set greater than or equal to the power level of the highest-power analog block occupied. If switched capacitor blocks are not used, the reference power can be set to ‘SC Off/x’, where x is any of the reference level settings.

Note The reference circuit current field is only available when the PLT UM is not placed.

Example Projects

This application note also discusses four example projects. To use these example projects, you should be familiar with PSoC Designer. To learn more about PSoC Designer, you can see the link www.cypress.com/?id=1162. A brief overview of each code example is given here:

Project	Description
Using PLT UM in a Code Example Associated project:AN54416_PLC_Transceiver	Develop a PLC application to communicate between two nodes with the device CY8CPLC20.
UART Interface for CY8CPLC20 Associated project: AN54416_UART_Host_Interface	Explains the UART configuration, packet structure, application, and provides an example algorithm for how the host communicates with the CY8CPLC20 device.
I2C Interface for CY8CPLC20 Associated project: AN54416_I2C_Host_Interface	Explains the I^2C configuration, packet structure, application, and provides an example project.
Low-Power Firmware for CY8CPLC20 Associated project: AN54416_PLC_Low_Power	Shows how the Cypress PLC device can perform with an average low power consumption of < 50 mW.

Using PLT UM in a Code Example

The PLT UM can be implemented on the device CY8CPLC20. This section describes how to develop a PLC application to communicate between two nodes with the device CY8CPLC20.

The user is expected to know how to use PSoC Designer. To learn more about PSoC Designer, you can see the link www.cypress.com/?id=1162.

Software Requirements

- PSoC Designer 5.4 CP1 or higher
- PSoC Programmer 3.14 or higher

These software tools can be downloaded from www.cypress.com/psocdesigner.

Using the PLT UM in an Example Project

This section provides a guide to get started with the PLT UM. The general steps are:

1. Open PSoC Designer and create a new chip-level Project.
2. Select a CY8CPLC20 device (-28PVXI for 28-pin SSOP or -48LFXI for 48-pin QFN) as shown in Figure 5.
3. Both devices have the same number of available hardware blocks and memory but the 48-pin device has more GPIO pins.

Figure 5. Select Project Type

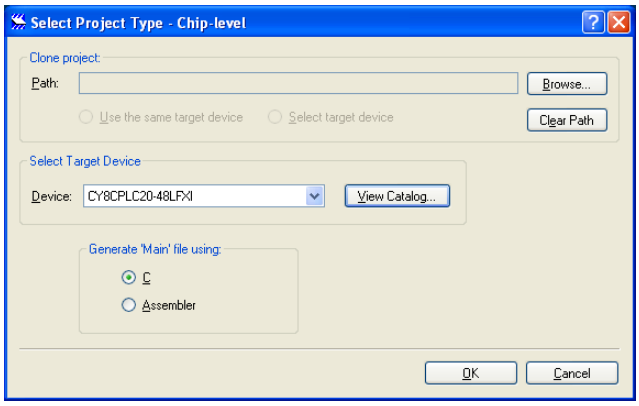
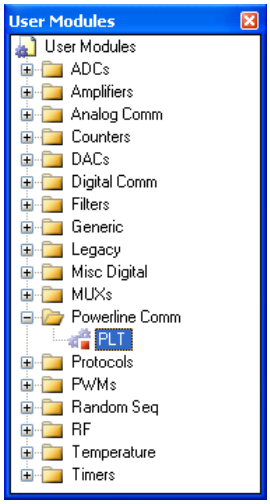


Figure 6. CY8CPLC20 User Modules



4. In the User Modules window, open the **Powerline Communication** folder and double-click the PLT User Module to place it as shown in Figure 6.

   When placing the PLT User Module, a window pops up, showing three options (see Figure 7).

   "**FSK Modem**"

   With this choice, only the FSK Modem is implemented. The user can implement a custom networking protocol to manage the communication over the powerline.

   "**FSK Modem + Network Stack**":

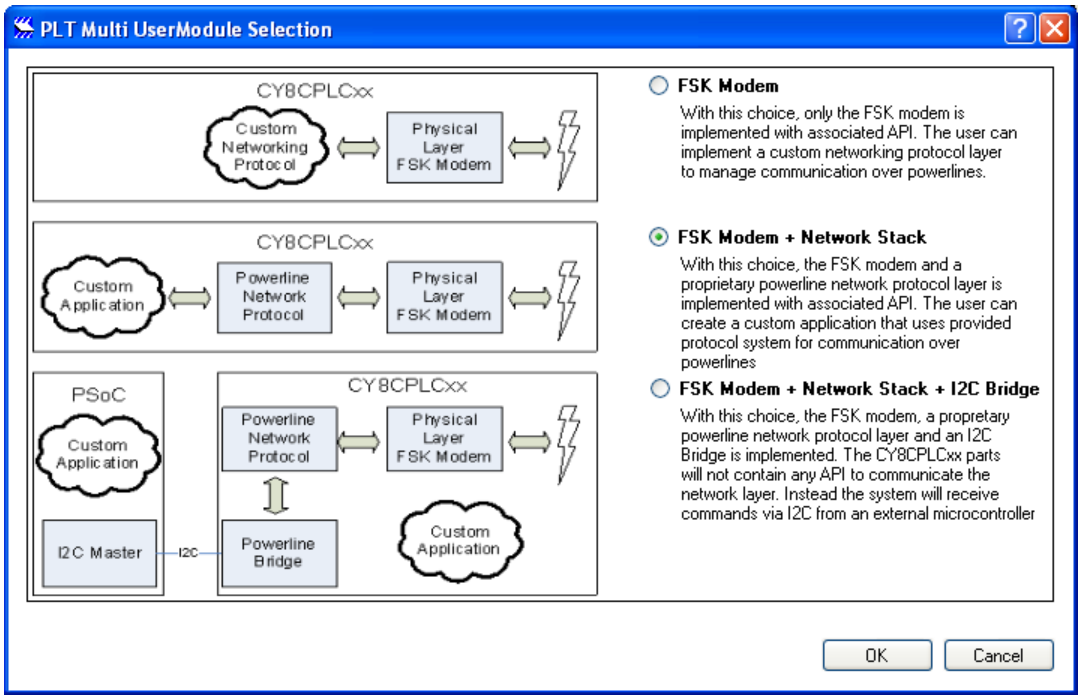
   With this option, the FSK modem and a proprietary network protocol layer is implemented. The user needs to create a custom application that uses the provided protocol to communicate over the powerline.

   "**FSK Modem + Network Stack + I2C Bridge**":

   With this option, the FSK modem a proprietary network protocol layer and an I^2C bridge is implemented. There are no APIs to communicate with the network layer; instead the system will receive commands from the external I^2C host. For detailed steps on how to implement this, see "I2C Interface for CY8CPLC20."

   Select the "FSK Modem + Network Stack"option. With this option, the FSK modem and the proprietary Cypress network protocol is implemented. This allows you to focus on your custom application, while the network protocol manages the CSMA, transmission, reception, and error detection. For details regarding the other two options, refer to the PLT UM datasheet.

Figure 7. PLT Multi-User Module Selection Window



5. Open the PLT Configuration Wizard and set the properties shown in Figure 8 and Table 1. Note that these properties can also be set in the application code using the PLT memory array.

Figure 8. PLT Config Wizard

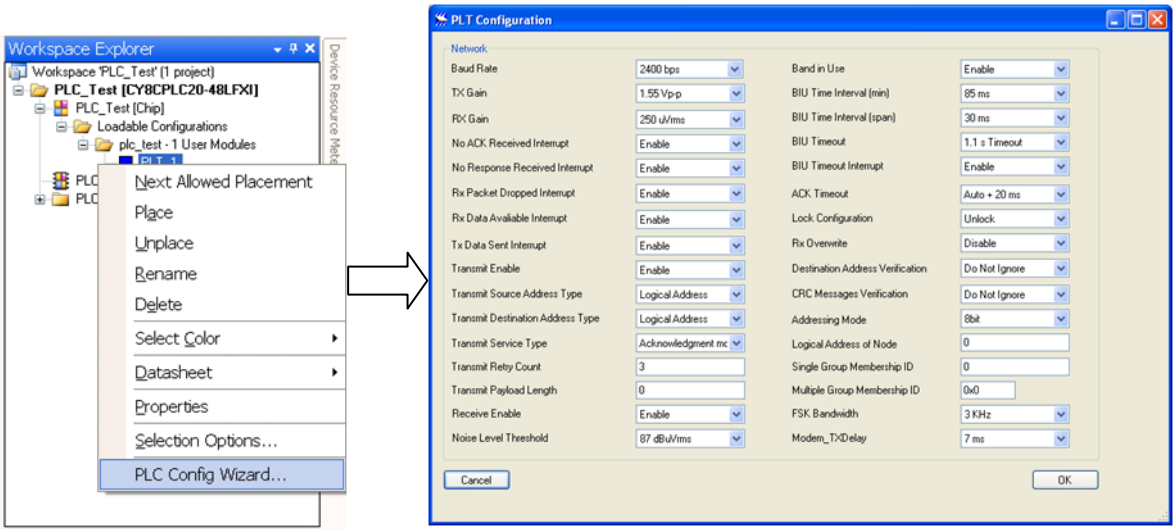


Table 1. PLT UM Configuration

No.	Property	Register	Value	Details
1	Baud Rate	Modem_Config	2400bps (default)	Sets the baud rate for the PLC PHY
2	Tx Gain	TX_Gain	1.55V_{p-p} (default)	Set to 1.55 V_{p-p}. Set to 125 mV_{p-p} (for CENELEC).
3	Rx Gain	Rx_Gain	250uVrms	Sets the minimum input sensitivity for the receiver
4	No ACK Received Interrupt	INT_Enable	Enable	Enable Interrupt for no acknowledgment received after if Service Type = 1 (ACK Mode)
5	No Response Received Interrupt	INT_Enable	Enable	Enable Interrupt for No Response Received
6	Rx Packet Dropped Interrupt	INT_Enable	Enable	Enable Interrupt when RX Packet is dropped because RX Buffer is full
7	Rx Data Available Interrupt	INT_Enable	Enable	Enable Interrupt when RX buffer has new data
8	Tx Data Sent Interrupt	INT_Enable	Enable	Enable Interrupt when TX data is sent successfully
9	Transmit Enable	PLC_Mode	Enable	Enables Transmit Mode
10	Transmit Source Address Type	TX_Config	Logical Address (default)	Logical Address / Physical Address
11	Transmit Destination Address Type	TX_Config	Logical Address (default)	Logical Address / Group Address / Physical Address
12	Transmit Service Type	TX_Config	Acknowledgement Mode	Transmissions are acknowledged
13	Transmit Retry Count	TX_Config	3	Number of times a transmitter should retry before sending a new packet
14	Transmit Payload Length	TX_Message_Length	0	Length of the payload
15	Receive Enable	PLC_Mode	Enable	Enable Receiver Mode
16	Noise Threshold Value	Threshold_Noise	87dBuV (default)	Threshold for BIU detection (set to 87 dBuV for CENELEC)
17	Band In Use	PLC_Mode	Enable (default)	Should be enabled for CENELEC
18	BIU Time Interval (min)	Timing_Config	85ms (default)	85 ms for CENELEC
19	BIU Time Interval (span)	Timing_Config	30ms (default)	30 ms for CENELEC
20	BIU Timeout	Timing_Config	1.1s Timeout (default)	1.1 s for CENELEC

No.	Property	Register	Value	Details
21	BIU Timeout Interrupt	INT_Enable	Enable (default)	Enable Interrupt for BIU Timeout and the Modem is unable to Transmit, if Disable BIU = 0
22	ACK Timeout	Timing_Config	Auto +20ms	Time for which Transmitter waits for ACK
23	Lock Configuration	PLC_Mode	Unlock (default)	Allow device to be configured remotely
24	Rx Overwrite	PLC_Mode	Disable (default)	If RX Buffer is full, new RX Message isdropped.
25	Destination Address Verification	PLC_Mode	Do not Ignore	Check if the Destination Address is a match before processing the packet
26	CRC Address Verification	PLC_Mode	Do not Ignore	Drop the packet if CRC fails
27	Addressing Mode	PLC_Mode	8bit	Select between 8bit and 16bit
28	Logical Address of Node	Local_LA_LSB Local_LA_MSB	0	In this project, Logical Address is set in the code.
29	Single Group Membership ID	Local_Group	0	For this example project Group Membership is not used
30	Multiple Group Membership ID	Local_Group_Hot	0x0	For this example project Group Membership is not used
31	FSK Bandwidth	Modem_ Config	3kHz (default)	Separation of FSK signals for logic ‘1’ and ‘0’
32	Modem Tx Delay	Modem_ Config	7ms (default)	This value is dependent on the BAUD rate

Note These settings can also be done with the following application code (not included in the example project as it is already done in the PLC Config Wizard).

```
/* Set the baud  rate to 2400bps with a 3kHz deviation */
PLT_Memory_Array[Modem_Config] = (Modem_FSKBW_3M | Modem_BPS_2400);
/* Enable the PLC Transmitter and Receiver*/
PLT_Memory_Array[PLC_Mode] = (TX_Enable | RX_Enable);
/* Enable Acknowledgement and 3 retries*/
PLT_Memory_Array[TX_Config] = (TX_Service_Type | 0x03);
/* Set the transmitter gain to 1.55Vp-p.*/
PLT_Memory_Array[TX_Gain] = 0x0b;
/* Set the receiver gain to 250uVrms */
PLT_Memory_Array[RX_Gain] = 0x06;
/* Set the length of the transmit buffer to 0x00 */
PLT_Memory_Array[TX_Message_Length] = 0x00;
/* Enable Interrupt reporting for all events */
PLT_Memory_Array[INT_Enable]   =   (INT_UnableToTX   |   INT_TX_NO_ACK   |INT_TX_NO_RESP   |
INT_RX_Packet_Dropped | INT_RX_Data_Available |INT_TX_Data_Sent);
```

6. Select **Build > Generate Configuration** (or press Ctrl + F6) to generate the configuration that includes the user module assembly and C code needed for the application.

7. Place an LCD UM and three LED UMs with the following parameters:
   a. LCD UM:
      i. Name = “LCD” LCD_Port = “Port_4”. A LCD can be directly connected to the available LCD port.
   b. 3 x LED UM (The corresponding LEDs are already hardwired to the respective pins on the Cypress PLC Development Kits CY3274)
      i. Name = “BIU_LED”, Port = “Port_2”, Pin = “Port_2_1”, Drive = “Active High”
      ii. Name = “RX_LED”, Port = “Port_2”, Pin = “Port_2_3”, Drive = “Active High”
      iii. Name = “TX_LED”, Port = “Port_2”, Pin = “Port_2_5”, Drive = “Active High”
   c. Update the BIU, TX, and RX ISR code snippets in the *PLT_1INT.asm*file from Appendix A - BIU, TX, RX Interrupt Service Routines (Section of PLT_1INT.asm). Each ISR enables the appropriate LED when the status is active (for example, turn on TX_LED when transmitting) or disables the appropriate LED when the status is complete.
      i. PLT_BIU_Active_ISR: This ISR is called when Band-In-Use is active
      ii. PLT_BIU_Complete_ISR: This ISR is called when Band-In-Use is no longer set
      iii. PLT_TX_Active_ISR: This ISR is called when the Transmitter is actively sending a message
      iv. PLT_TX_Complete_ISR: This ISR is called when the Transmitter has completed sending the message
      v. PLT_RX_Active_ISR: This ISR is called when the Receiver is in process of receiving a packet
      vi. PLT_RX_Complete_ISR: This ISR is called when the Receiver is no longer receiving a packet
   d. Buttons
      i. Set P0[1] according to the settings shown below. This pin P01 should be manually connected to the button with a jumper wire, which will be used to switch the transmit state on and off.

Name	Tx_Trigger
Port	P0[1]
Port	StdCPU
Drive	Pull Down
Interrupt	Falling Edge
Initial Value	0

      ii. Set P0[4] according to the settings shown below. This ping is used to select the Local Logical Address of device.

Name	ADD_Select
Port	P0[4]
Port	StdCPU
Drive	Pull Up
Interrupt	DisableInt
Initial Value	1

8. Copy the *main.c* code from the attached code example into this project’s *main.c* file. A flowchart of the application code and the block diagram are in Figure 9 and Figure 10 respectively. A high-level description of the code is as follows:
   a. The PLT and LCD User Modules are initialized. The global interrupts are enabled. The PLT UM is configured.
   b. Check if the Pin ADD_Select is HIGH. If yes, set the Local Logical Address to 0x01 and Destination Address to 0x02. Otherwise, set the Local Logical Address to 0x02 and Destination Address to 0x01.

c. 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. If the transmitter mode is enabled, then the Tx_Count is also updated on the LCD.

d. When a GPIO interrupt occurs (for example, a button press) on pin P01, the transmitter state is inverted. If the transmitter is enabled, then a packet is transmitted to the destination address in the TX_DA register. TX_Count is incremented and the LCD is updated. If an acknowledgement is received, then the TX_Success_Count is incremented and the LCD is updated. The PSoC Host will continue to initiate data transmission until a GPIO interrupt occurs again.

Figure 9. Block Diagram of Example Project

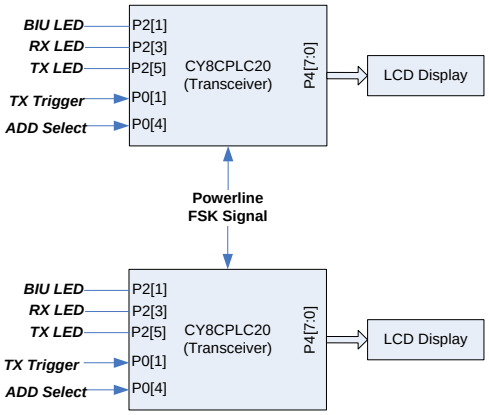
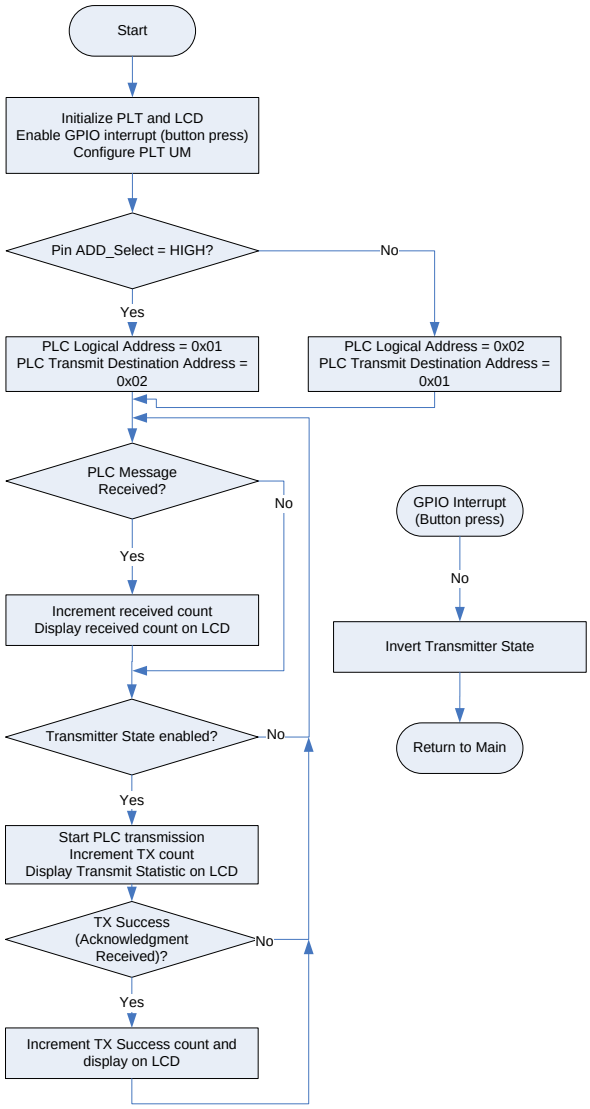


Figure 10. Code Example Flow Chart



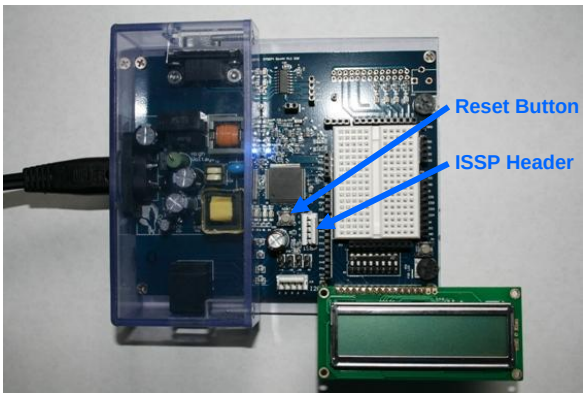
Evaluating the Example Project on Hardware

This example is designed to run on the CY3274 high-voltage PLC DVK. You can see the latest kit documentation for this board (CY3274). The BIU, RX, and TX LEDs on the board are hardwired to P2[1], P2[3], and P2[5], respectively.

Programming the Boards

1. Connect the LCD daughter card to the LCD connector.
2. Connect the power cable to one of the boards. The Blue Power LED should turn on.

   Figure 11. Hardware Setup for CY3274

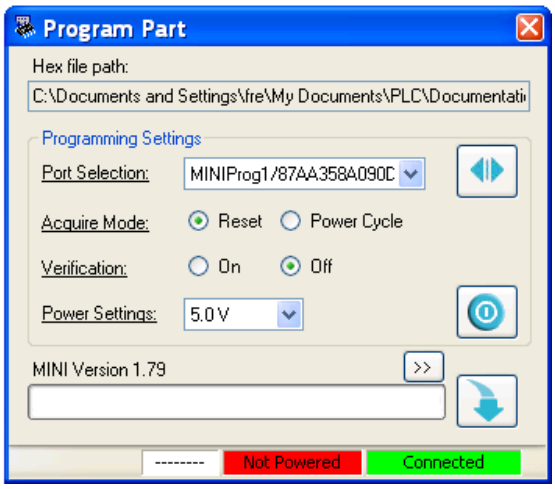


3. Attach one end of the MiniProg programmer to the ISSP header and the other end to the PC through the USB cable.

   Figure 12. Connecting the MiniProg Programmer

4. Open PSoC Designer and open the project AN54416_PLC_Transceiver.
5. Click **Program > Program Part**.
6. On the Program Part window, set “Acquire Mode” to Reset. This is because the board is already powered.

Figure 13. Programming Settings

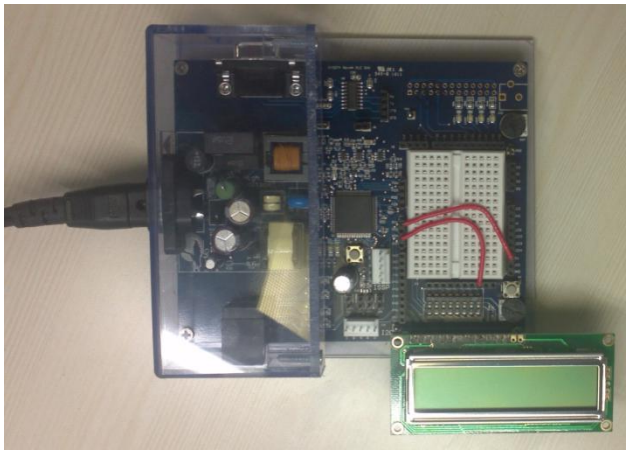


7. Click the arrow at the bottom right corner to start programming. After the progress bar is complete, the status should say ‘Programming Successful’.
8. Remove MiniProg from the ISSP header and press the RESET pushbutton to reset the device.
9. Repeat the above steps to program another board.

Hardware Setup

Follow the steps below to set up the board as shown in Figure 14.

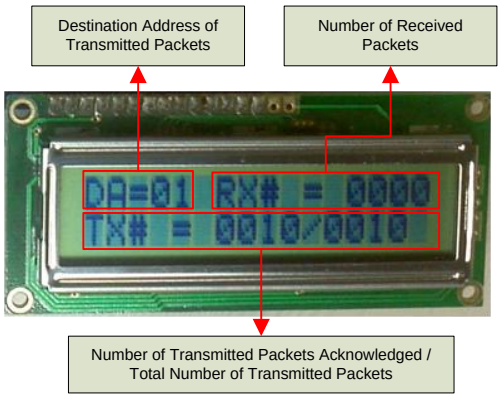
Figure 14. Hardware Setup for CY3274



1. Connect P0[1] to Switch SW (this is the Switch S4 for toggling the Tx state).
2. Connect P0[4] to one of the 8 DIP switches.
3. Power up the board and make sure the blue power LED is turned ON.
4. Repeat Steps 1 to 3 to set up the second board.

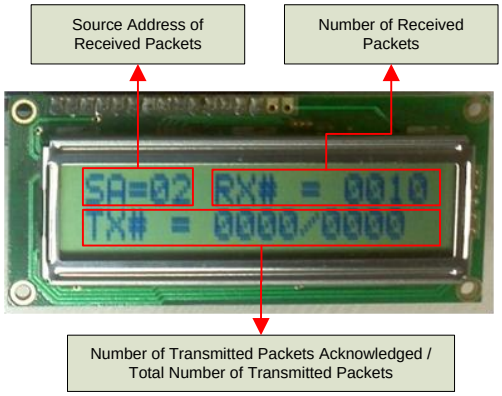
5. Turn on the DIP switch on the first board and reset it. Make sure the other board the DIP switches is different. The board should display local logical address, LA=02. Press the reset button on both boards to load the addresses.
6. Press the switch S4. The first board will continuously transmit 1000 packets to the second board with logical address 0x01.

Figure 15. Transmitter Board After Sending 10 Packets



When the tranceiver is transmitting, the LCD updates the destination address and the transmitter statistics, that is., total number of succesfully transmitted data packets upon the total number of transmitted packets.

Figure 16. Receiver LCD after Receiving 10 Packets



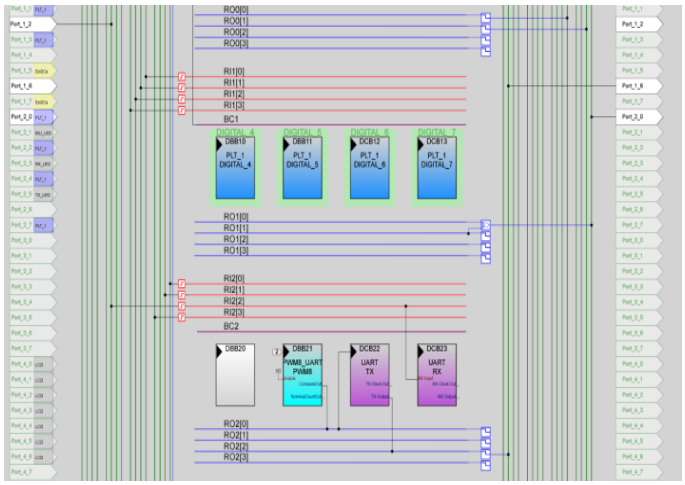
When the tranceiver is receiving the LCD updates the source address and the number of received data packets.

UART Interface for CY8CPLC20

UART Configuration

The UART in this code example is designed to run at 9600 bps, which means a 76.8 kHz clock needs to be generated (8x the UART baud rate). Since the VC1, VC2, and VC3 clock dividers are used by the PLT user module, a PWM8 user module is needed.

Figure 17. UART Connections



In this example, Pin P1[2] will be the input (RX) and P1[6] will be the output (TX).

1. Place a PWM8 user module (PWMs folder) and set the following parameters (including Name) shown in Figure 18. The clock frequency is determined by knowing (from the Global Resources window) that VC2 = (VC1 / 6) = ((SysClk / 2) / 6) = 2 MHz. The actual divider for this clock is (Period + 1) = 26. Therefore, the generated clock from the PWM is 2 MHz / 26 = 76.9 kHz.

Figure 18. PWM8_UART Parameters

Parameters - PWM8_UART

Name	PWM8_UART
User Module	PWM8
Version	2.60
Clock	VC2
Enable	High
CompareOut	Row_2_Output_0
TerminalCountOut	None
Period	25
PulseWidth	12
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

2. Place a UART UM (Digital Comm folder) and set the following parameters (including Name) shown in Figure 19. The key parameters are Clock, RX Input, TX Output, ClockSync, and RxCmdBuffer. Note that the Clock input is from the bus that is connected to the output of the PWM8_UART.

Figure 19. UART Parameters

Parameters - UART

Name	UART
User Module	UART
Version	5.3
Clock	Row_2_Output_0
RX Input	Row_2_Input_2
TX Output	Row_2_Output_2
TX Interrupt Mode	TXComplete
ClockSync	Sync to SysClk
RxCmdBuffer	Disable
RxBufferSize	16
CommandTerminator	13
Param_Delimiter	32
IgnoreCharsBelow	32
Enable_BackSpace	Disable
RX Output	None
RX Clock Out	None
TX Clock Out	None
InvertRX Input	Normal

3. UART Pinouts

a. UART Rx should be connected to P1[2].

Set the P1[2] properties as shown in Figure 20.

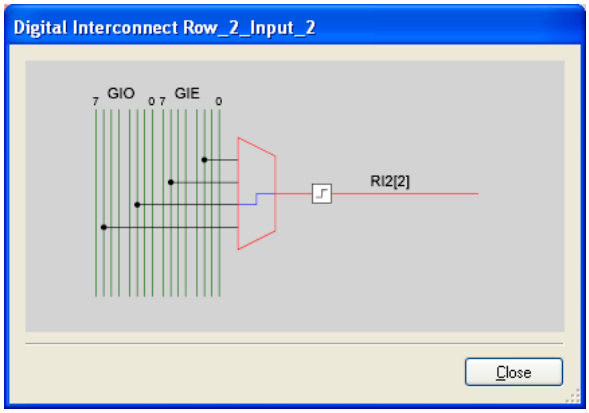
Connect column bus GlobalInOdd_2 to Row_2Input_2, as shown in Figure 21.

Figure 20. Pin P1[2] Parameters

Name	Port_1_2
Pin	Port_1_2
Select	GlobalInOdd_2
Drive	High Z
Interrupt	DisableInt
InitialValue	0

OK Cancel

Figure 21. Row 2 Input 2 Connection



b. UARTTx should be connected to P1[6].

Set the P1[6] properties as shown in Figure 22.

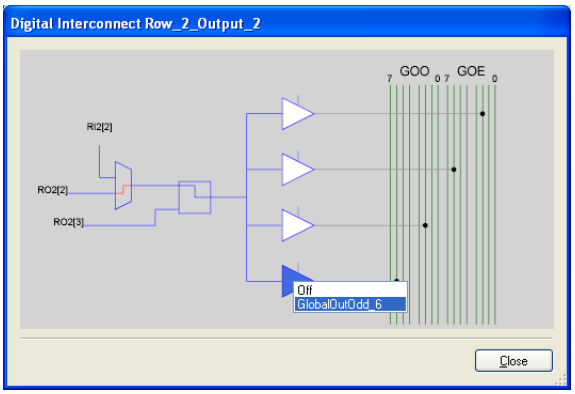
Connect Row_2_Ouput_2 to the column bus GlobalOutOdd_6, as shown in Figure 23.

Figure 22. Pin P1[6] Parameters

Name	Port_1_6
Pin	Port_1_6
Select	GlobalOutOdd_6
Drive	Strong
Interrupt	DisableInt
InitialValue	0

OK Cancel

Figure 23. Row 2 Output 2 Connection



UART Packet Structure

To run the UART in the application, it is important to understand how and when the packets are sent to and from the host. The host always initiates communication and configures the PLT user module, transmits, and receives data by writing to and reading from the PLT_Memory_Array table. To write data, the host must send a write packet. To read data, the host must send a read packet and then wait for the CY8CPLC20 device to reply with a read response packet. Host write packets are processed immediately by the CY8CPLC20 device (handled in an ISR `Host_UART_ISR`). However, the response packets are transmitted only when the device is not busy.

UART Interface Host Write Packet Structure

When the host wants to write data to the PLT_Memory_Array, it sets the MSb of the first byte to '0' to indicate a write. The remaining seven bits indicate how many bytes will be written to the memory array. The next byte indicates the starting offset of the PLT_Memory_Array that will be written to. The subsequent bytes contain the data that will be stored.

	7	6	5	4	3	2	1	0
Byte 0	0	Length						
Byte 1	Offset							
Byte 2+	Data							

UART Interface Host Read Packet Structure

When the host wants to read data from the PLT_Memory_Array, it sets the MSb of the first byte to '1' to indicate a read. The remaining seven bits indicate how many bytes should be read from the memory array. The next byte indicates the starting offset of the PLT_Memory_Array that will be read from.

	7	6	5	4	3	2	1	0
Byte 0	1	Length						
Byte 1	Offset							

UART Interface CY8CPLC20 Read Response Packet Structure

After receiving the read command, the device responds by sending the requested data from the PLT_Memory_Array. The number of bytes to send was set by the Length parameter in the Read packet from the host.

	7	6	5	4	3	2	1	0
Byte 0+	Data							

UART Application

When a new byte of data is received, the ISR Host_UART_ISR is triggered. This executes a state machine based on the byte position of the packet that is received. Initially, the position is 0, so the length and read/write bit will be stored. Then, based on the read/write bit, the subsequent received bytes are processed accordingly. To make sure that the byte position is aligned correctly, a timeout resets the positions if a byte is not received within ~2.5 ms. A byte should be received every ~1.25 ms (9600 bps including start and stop bits), so there is a buffer of ~100%. The 2.5 ms timeout is achieved by incrementing the value `bUART_Cycles` in the PWM8_UART_ISR, which occurs for every cycle (76.9 kHz). After the value reaches 200, the timeout occurs.

If a read packet is successfully received, the requested data is transferred to the TX buffer as a response packet. Also, a flag `bHost_Reply` is set to notify the application that there is a message to be sent. When the application code reaches the point to check for this flag, it transmits the response.

UART Host Example

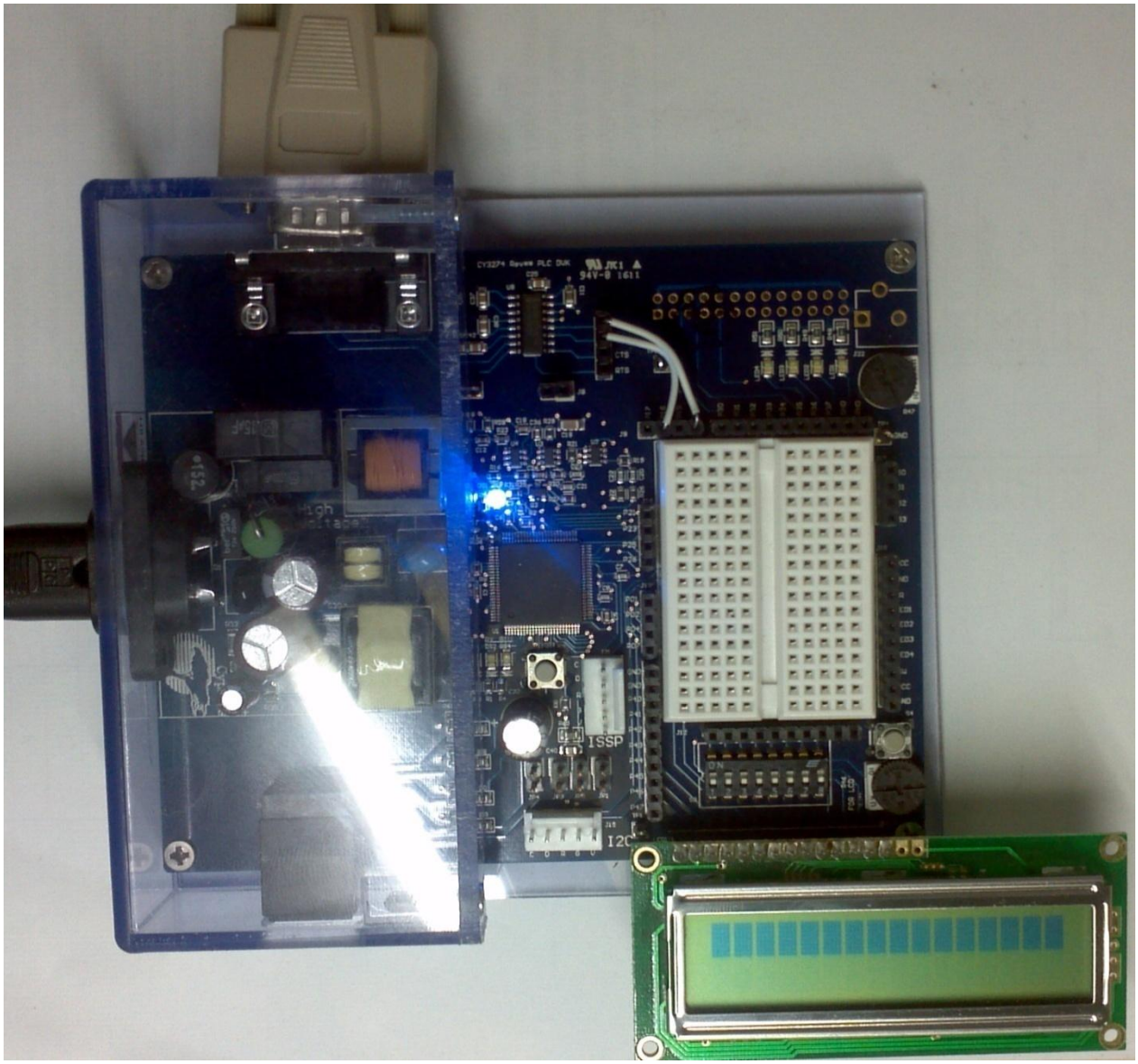
An example algorithm of a host trying to send a PLC message is:

1. Send {0x01, 0x06, 0x81} which writes the value 0x81 to PLT_Memory_Array[0x06] to send a one-byte message
2. Send {0x81, 0x69} which reads from PLT_Memory_Array[0x69] to read the status of transmission
3. Wait until receiving one byte of data
4. If the received byte contains an event update of the transmission, it is done. Otherwise, repeat step 2.

UART Hardware Connection

On the PLC DVKs, to connect the CY8CPLC20 device to the RS232 port, connect a jumper wire from P12 to RX on header J20 and another wire from P16 to TX on header J20, as shown with the white wires in Figure 24.

Figure 24. UART Hardware Connection



I²C Interface for CY8CPLC20

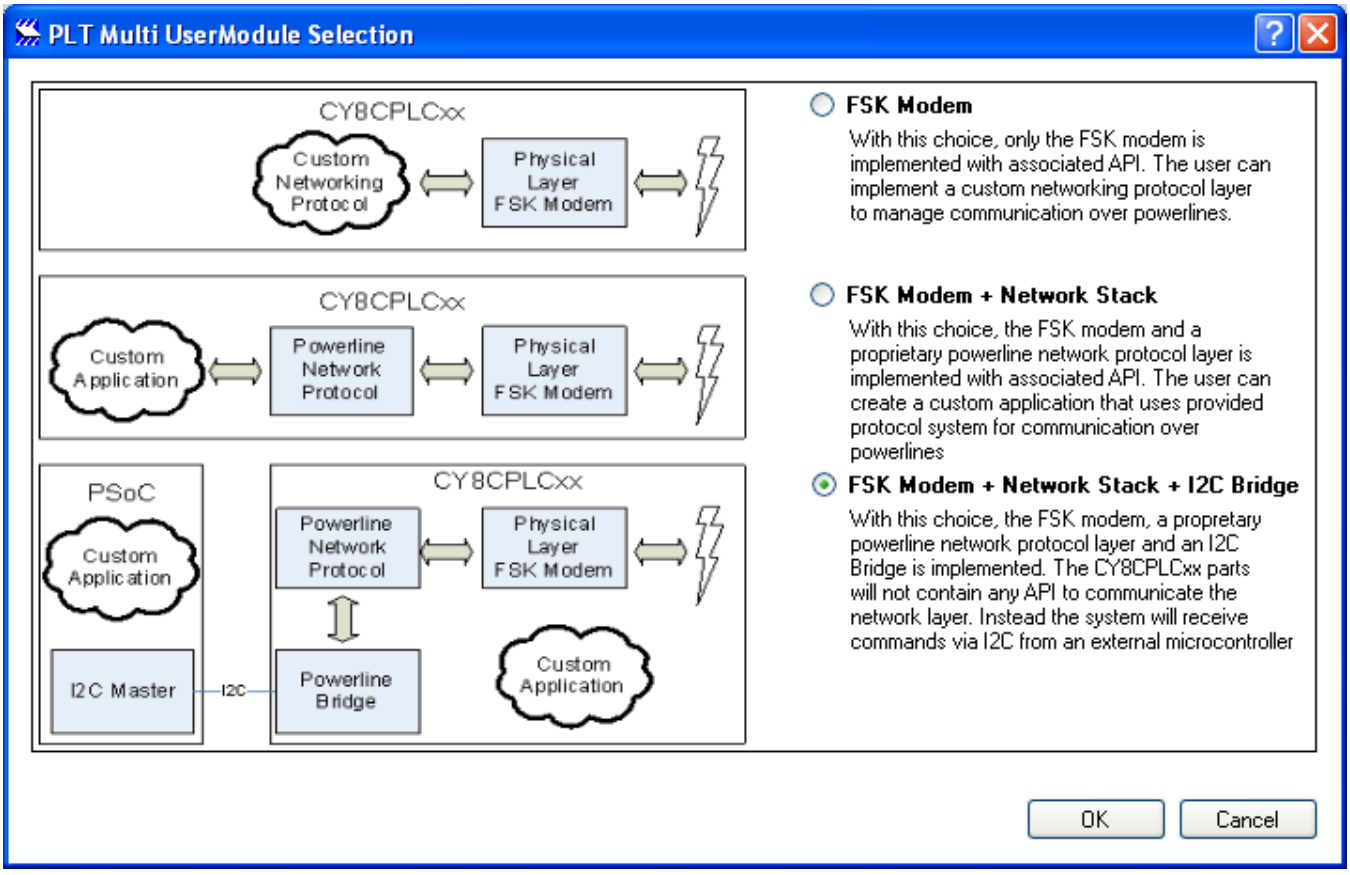
This section describes how the I²C interface is developed. The I²C interface is compatible with the Cypress PLC control panel GUI, which can be downloaded here. This section explains the I²C configuration, packet structure, application, and provides an example algorithm for how the host communicates with the CY8CPLC20 device.

PLT Configuration

Follow these steps to set up the PLT UM with I²C interface:

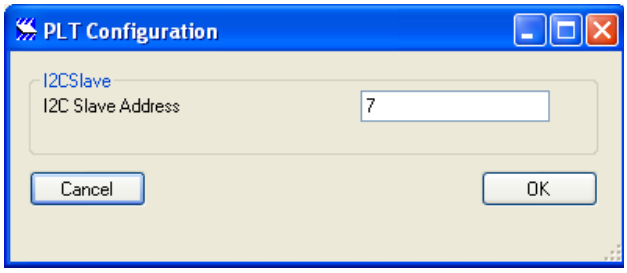
1. Place the PLT UM and Select the “FSK Modem + Network Stack + I²C Bridge” option as shown below.

Figure 25. Select Option "FSK Modem + Network Stack + I²C Bridge"



2. Set the I²C address in the PLT Configuration window. Note that all the other parameters are set by the Host.

Figure 26. PLT Configuration



3. Copy the sample code below for “FSK Modem + Network Stack + I²C Bridge”.

```
#include <m8c.h>
// part specific constants and macros
#include"PSoCAPI.h"     //     PSoC     API
definitions for all user modules
void main(void)
{
PLT_Start();
//Initialize the PLT module
do
     {
    PLT_Check_I2C_Activity();
    //Act on any I2C activity
PLT_Poll();
    // Run the network protocol
}while(1);
}
```

4. 3 x LED UM (The corresponding LEDs are already hardwired to the respective pins on the Cypress PLC Development Kit CY3274)

a. Name = “BIU_LED”, Port = “Port_2”, Pin = “Port_2_1”, Drive = “Active High”

b. Name = “RX_LED”, Port = “Port_2”, Pin = “Port_2_3”, Drive = “Active High”

c. Name = “TX_LED”, Port = “Port_2”, Pin = “Port_2_5”, Drive = “Active High”

5. Update the BIU, TX and RX ISR code snippets in PLT_1INT.asm from Appendix A - BIU, TX, RX Interrupt Service Routines (Section of PLT_1INT.asm). Each ISR enables the appropriate LED when the status is active (for example, turn on TX_LED when transmitting) or disable the appropriate LED when the status is complete.

a. PLT_BIU_Active_ISR: This ISR is called when the Band In Use is active

b. PLT_BIU_Complete_ISR: This ISR is called when the Band In Use is no longer set

c. PLT_TX_Active_ISR: This ISR is called when the Transmitter is actively sending a message

d. PLT_TX_Complete_ISR: This ISR is called when the Transmitter has completed sending the message

e. PLT_RX_Active_ISR: This ISR is called when the Receiver is in process of receiving a packet

f. PLT_RX_Complete_ISR: This ISR is called when the Receiver is no longer receiving a packet

I²C Interface Write Packet Structure

The I²C interface follows the packet structure defined by the I²C specification. The write packet is as follows. The master always sends the entire packet.

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Byte 0	I2C Address (0b0000001)							0
Byte 1	Offset							
Byte 2+	Data from host to CY8CPLC20 (Optional)							

I²C Interface Read Packet Structure

The I²C interface follows the packet structure defined by the I²C specification. The read packet is as follows. The master sends the first byte. The offset is set by sending a write packet for that offset (sending data is not necessary).

	7	6	5	4	3	2	1	0
Byte 0	I2C Address (0b0000001)							1
Byte 1+	Data from CY8CPLC20 to host							

I²C Application

The I²C write and read messages are processed automatically by the EzI2Cs user module by the ISRs associated with the user module. It is important to note that I²C interrupts are disabled by the PLT user module when it is transmitting. During this time, if the host initiated an I²C message, the CY8CPLC20 device holds the SCL bus low until it finishes transmitting the PLC packet. It then resumes processing the I²C message.

I²C Host Example

An example algorithm of a host trying to tell the CY8CPLC20 message (I²C address 0x01) to send a PLC message would be (Note that the I²C address is shifted left by one bit):

1. Send {0x02, 0x06, 0x81} which writes the value 0x81 to the PLT_Memory_Array[0x06] to send a one-byte message
2. Send {0x03, 0x69} which reads from the PLT_Memory_Array[0x69] to read the status of transmission
3. Read one byte of data
4. If the received byte contains an event update of the transmission, it is done. Otherwise, repeat step 2.

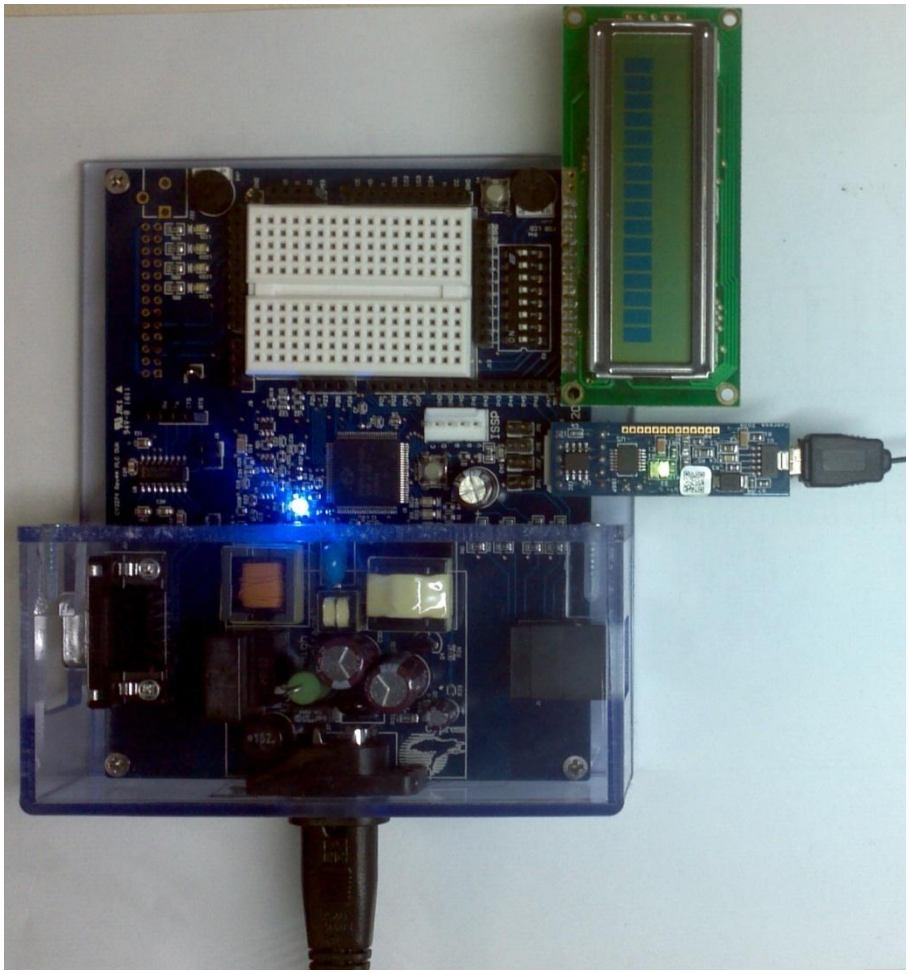
I²C Hardware Connection

On the PLC DVKs, to connect the CY8CPLC20 device to the PC, a CY3240 Cypress USB-I2C bridge can be connected to the 5-pin I²C header, as shown in Figure 27.

I²C Hardware Connection

On the PLC DVKs, to connect the CY8CPLC20 device to the PC, a CY3240 Cypress USB-I2C bridge can be connected to the 5-pin I²C header, as shown in Figure 27.

Figure 27. I²C Hardware Connection



Low-Power Firmware for CY8CPLC20

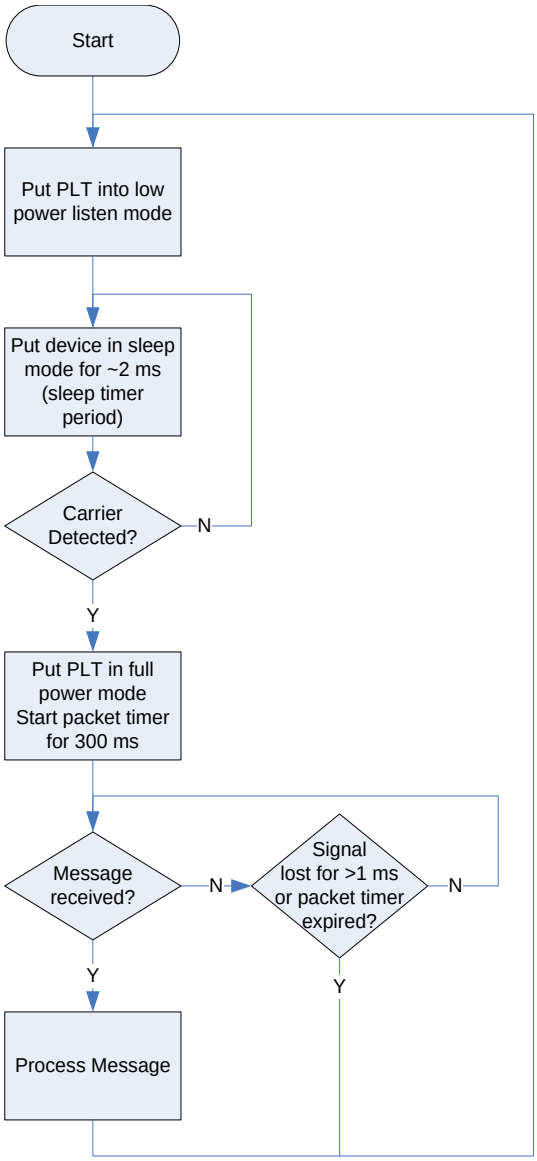
This section describes how the Cypress PLC device can perform with an average low-power consumption of < 50 mW. The code example for the CY8CPLC20 is attached.

Basic Operation

Under normal receive mode operation; CY8CPLC20 devices consume ~43 mA of current, which is equivalent to 215 mW of power at 5-V V_{CC}. For systems that have a requirement of <50 mW average power consumption, the PLC device can be configured in a low-power 'listen' mode, where it turns off most of the PLT blocks, turn on the carrier detect blocks, and wait to detect a signal on the powerline. After a signal is detected, the device turns on the remaining PLT blocks for demodulating the PLC signal and forming the PLC packet.

If the packet is intended for this node, the PLC device processes it and then returns to 'listen' mode. If the packet received is not intended for this node or if there is excessive noise that wakes the node up, the device returns to'listen' mode when the carrier detection is lost or after a 300-ms timeout period. See the following figure for the basic low-power algorithm.

Figure 28. PLC Low-Power Firmware Basic Operation



Power Consumption

In the 'listen' mode, the device's current consumption was 8.8 mA (power consumption of 44 mW). In the normal receive mode, the device's current consumption was 43 mA (power consumption of 215 mW). If the line is clean at least 97% of the time, the total average power consumed is <50 mW as seen in Table 2.

Table 2. Average Power Consumption

			Average Power (mW)
	Current (mA)	**Power (mW)**	**97% Free Line**
Listen Mode	7	35	34
Receive Mode	43.6	218	6.5
Total			40.54

Adaptive Carrier Detect Threshold

The carrier detect threshold is designed to adapt to the noise on the line so that the average power consumption stays under 50 mW. If noise causes two consecutive false triggers, the threshold is increased. On the other hand, if there is no signal on the line for >500 ms, the threshold is decreased. This is to ensure that the device has the best sensitivity possible while consuming <50 mW.

To differentiate between noise and PLC signals, the free time between carrier detections is measured. According to the CENELEC Band-In-Use (BIU) protocol, which is used by the Cypress PLC solution, there must be at least 85-ms free time before another PLC signal can be transmitted. Note that this system uses 80 ms as the threshold.

The minimum threshold is 125 μV_{rms} and the maximum threshold is 5 mV_{rms}. Therefore, if a signal that exceeds this carrier detect threshold is present for at least three sleep timer cycles (~6 ms total), the device switches to the receive mode so that it can demodulate the PLC signal. The TX Delay parameter should be set to the maximum (25 ms) in order to provide enough time for the receiver to detect the signal and switch to receive mode in time to demodulate the signal.

Code Example

The code example was created with PSoC Designer 5.4. A screenshot of the chip level view of the code example is shown in Figure 29. The following user modules are used in the code example:

- Powerline Transceiver
- Counter16: For the 300 ms timeout. The configuration is shown in Figure 30.
- OneShot: Stabilizes the filtered received signal to generate a constant high logic level when a carrier is present. It drops to a low logic level when the carrier is absent for ≥ 8 * clock period = 8 * ~50 µs = ~400 µs. The configuration of the OneShot user module is shown in Figure 31. The output of the OneShot User Module is connected to the port pin P2[6] to be monitored for the carrier detection in code.

- LED: Three LED user modules (TX, RX, BIU) are used to represent when the PLT User Module is transmitting, receiving, or has detected a BIU timeout condition, respectively. In the debug mode (when PLC_DEBUG = 1), there are two additional LED user modules that are used to indicate the power mode of the device. ReceiveMode_LED will go low when the PLT is in 'listen' mode and go high when in full-power receive mode. In the listen mode, the SleepMode_LED user module will go HIGH when the CPU is in low-power sleep mode, go LOW when it is checking for a carrier. In full-power receive mode, the SleepMode_LED will always be LOW. The pins used are as follows:
  - P2[1] = BIU LED
  - P2[3] = RX LED
  - P2[5] = TX LED
  - P1[2] = Sleep Mode LED
  - P1[6] = Full Power Receive LED
- LCD: When a message is received, it will display the first byte of data on the second row of the LCD. The LCD also displays the number of messages received on the first row of the LCD. In the debug mode (when PLC_DEBUG = 1), it will also display the carrier detect threshold level in the bottom right corner of the LCD.

After placing the user modules, the following global resources were modified to reduce power consumption:

- CPU Clock = SysClk / 8. Running the CPU at this frequency (3 MHz) will reduce the current consumption by ~3mA.
- SysClk*2 Disable = Yes. Since the 48-MHz clock is not used, this can be disabled. This will reduce the current consumption by ~1mA.
- Analog reference power, opamp bias, and analog buffer power. These settings are dynamically modified in the firmware. In the listen mode, they are set to low power. In receive mode, they are set to high power.

Figure 29. PLC Low-Power PSoC Designer Chip View

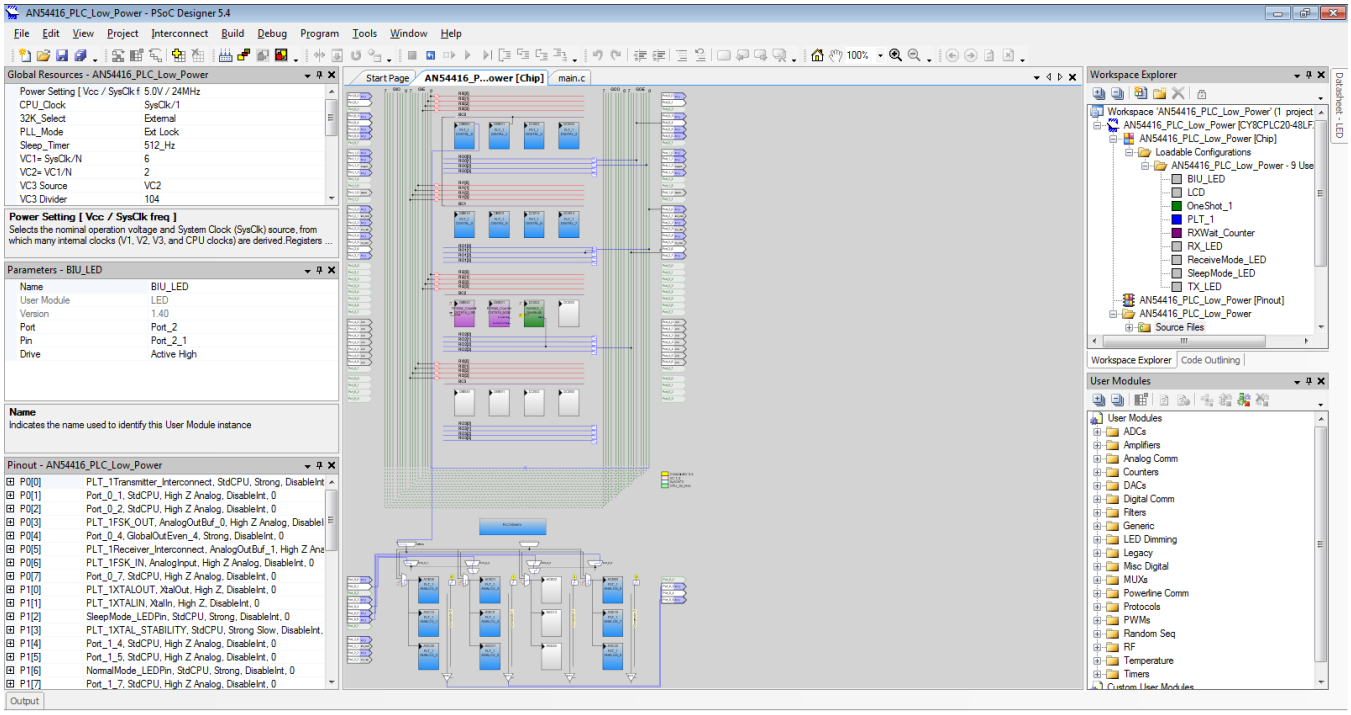
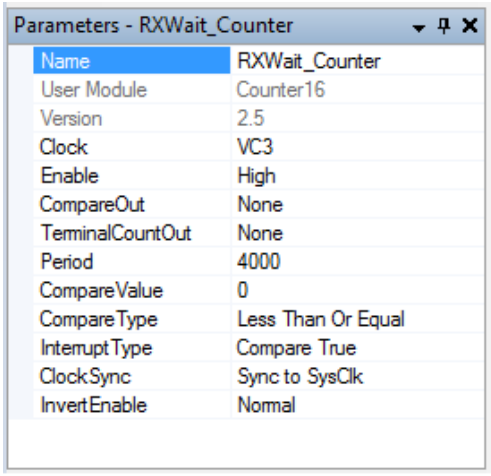


Figure 30. Counter16 Configuration



Firmware Algorithm

The firmware is written in C, with the exception of the PLT interrupt routines (in *PLT_1INT.asm*), which are modified to drive the transmit, receive, and band-in-use LEDs.

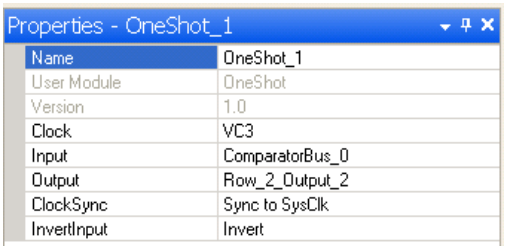
The algorithm for the code example is shown in Figure 32 and it is based on the basic operation diagram shown in Figure 27. It has additional details on detecting the carrier, as well as adapting the threshold based on the noise in the system. The received message is processed by incrementing a count and displaying it on the LCD.

Implementation on Hardware

The attached code example uses the same hardware as any of the PLC development kits (CY3274).

1. Connect the LCD User Module to the LCD header.
2. Connect the power cable from the mains to the board. The blue power LED should turn ON.
3. Connect the USB cable from the PC to the MiniProg programmer (included in the DVKs).
4. Connect the MiniProg to the ISSP connector.
5. Open the project in PSoC Programmer or in PSoC Designer. Click **Program**>**Program Part**. Load the *PLC_Low_Power.hex* file from the attached code example. Make sure programming mode is set to **Reset**. Program the device.
6. Remove the MiniProg from the ISSP connector.
7. Press the Reset button.

Figure 31. OneShot Configuration



To evaluate the system, a second PLC board with an I^2C interface can be used. The PLC Control Panel GUI can be run on a PC and interfaced to the board with the CY3240 USB-I^2C bridge board (included in the PLC kits).A high-voltage AC example using two CY3274 boards is shown in Figure 33. The receiving board displays the number of packets successfully received on an LCD display. The PLC Control Panel GUI can be downloaded at www.cypress.com/?rID=38135. The GUI user's guide contains instructions for programming the PLC DVKs to work with the GUI.

The receiving board is programmed with the listen mode firmware. The transmitting board has the following configuration (set by the GUI):

- 2400-bps baud rate
- 125-mVp-p transmit amplitude (CENELEC compliant when using the external circuitry on the CY3274). If using the low-voltage PLC boards, then a 1.55-Vp-p amplitude is recommended.
- 25-ms Transmit Delay (for ensuring a clean line before signal transmission and to give the PLC device time to wake up from listen mode)
- Acknowledgment mode
- Retry Count of 1

Figure 32. PLC Low-Power Code Example

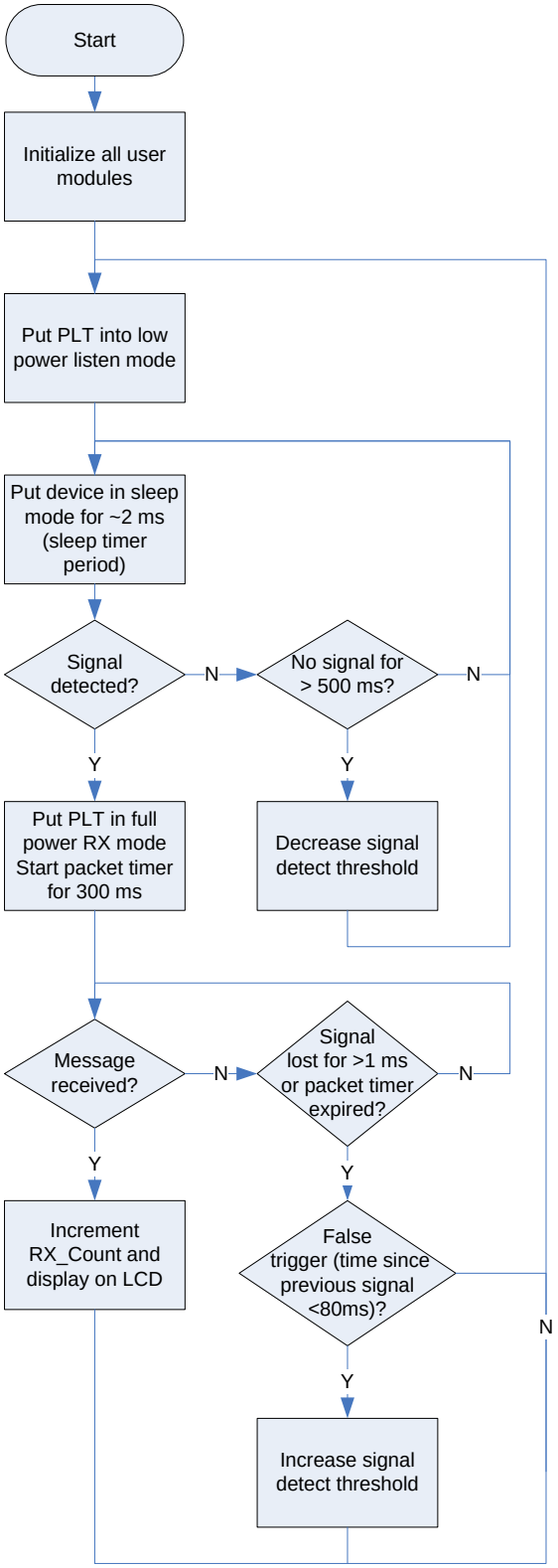
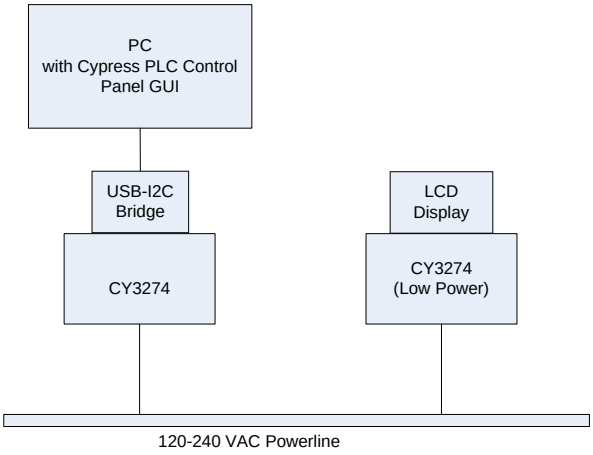


Figure 33. PLC Hardware Test Setup



Summary

The CY8CPLC20 device provides a flexible and easy-to-use solution for developing a highly integrated PLC-based system. With the flexibility of the PSoC core inside the CY8CPLC20, this code example can be modified to add additional functionality (for example, voltage measurement, current measurement, motor control, and so on) to create an intelligent end-user system.

Related Application Notes

- AN60685 - PLC - Interfacing the Cypress Powerline Communication Solution to CyFi Low-Power RF Module
- AN62792 - Updating Field Firmware with PLC
- AN62487 - Cypress Powerline Communication (PLC) Repeater Implementation
- AN62769 - Encrypted Data Communication Using Cypress PLC Solution
- AN55427 - Cypress Powerline Communication Board Design Analysis
- AN58825 - PLC - Powerline Communication Debugging Tools

About the Author

Name: Aditya Yadav
Title: Applications Engineer

Appendix A - BIU, TX, RX Interrupt Service Routines (Section of PLT_1INT.asm)

```
;-------------------------------------------------------------------------
;  FUNCTION NAME: _PLT_BIU_Active_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;      This is not a true ISR, but is called when the Band In Use is active
;
;-------------------------------------------------------------------------
;
_PLT_BIU_Active_ISR:

;@PSoC_UserCode_BIU_Active_BODY@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;   NOTE: interrupt service routines must preserve
;   the values of the A and X CPU registers.
   lcall _BIU_LED_On
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)

   ret

;-------------------------------------------------------------------------
;  FUNCTION NAME: _PLT_BIU_Complete_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;      This is not a true ISR, but is called when the Band In Use is no longer set
;
;-------------------------------------------------------------------------
;
_PLT_BIU_Complete_ISR:

;@PSoC_UserCode_BIU_Complete_BODY@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;   NOTE: interrupt service routines must preserve
;   the values of the A and X CPU registers.
   lcall _BIU_LED_Off
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)

   ret
```

```
    ;-------------------------------------------------------------------------
;  FUNCTION NAME: _PLT_TX_Active_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;      This is not a true ISR, but is called when the Transmitter is actively sending a
message
;
;-------------------------------------------------------------------------
_PLT_TX_Active_ISR:

;@PSoC_UserCode_TX_Active_BODY@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;   NOTE: interrupt service routines must preserve
;   the values of the A and X CPU registers.
      lcall _TX_LED_On
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)

   ret

;-------------------------------------------------------------------------
;  FUNCTION NAME: _PLT_TX_Complete_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;      This is not a true ISR, but is called when the Transmitter has completed sending the
message
;
;-------------------------------------------------------------------------
;
_PLT_TX_Complete_ISR:

;@PSoC_UserCode_TX_Complete_BODY@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;   NOTE: interrupt service routines must preserve
;   the values of the A and X CPU registers.
      lcall _TX_LED_Off
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)

   ret
```

```
    ;-------------------------------------------------------------------------
;  FUNCTION NAME: _PLT_RX_Active_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;      This is not a true ISR, but is called when the Receiver is in process of receiving a
packet
;
;-------------------------------------------------------------------------
;

_PLT_RX_Active_ISR:

;@PSoC_UserCode_RX_Active_BODY@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;   NOTE: interrupt service routines must preserve
;   the values of the A and X CPU registers.
      lcall _RX_LED_On

;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)
   ret

;-------------------------------------------------------------------------
;  FUNCTION NAME: _PLT_RX_Complete_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;      This is not a true ISR, but is called when the Receiver is no longer receiving a
packet
;
;-------------------------------------------------------------------------
;
_PLT_RX_Complete_ISR:

;@PSoC_UserCode_RX_Complete_BODY@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;   NOTE: interrupt service routines must preserve
;   the values of the A and X CPU registers.
      lcall _RX_LED_Off
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)

   ret
```

Document History

Document Title: Using CY8CPLC20 in Powerline Communication (PLC) Applications – AN54416

Document Number: 001-54416

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2728604	FRE	07/02/09	New Spec.
*A	2751601	FRE	08/11/09	Updated code example with TX, RX, and BIU LEDs.
*B	2828349	FRE	12/15/09	Updated for PSoC Designer 5.0 SP6 Release
*C	3143567	FRE	01/20/2011	Updated for PSoC Designer 5.1 SP1 Release Added analog voltage measurement to the transmitter code example Added optional UART and I^2C interfaces to the transmitter code example Added hardware evaluation procedure for example projects Removed LED.asm modifications
*D	3623259	ADIY	05/21/2012	Updated template. Updated for PSoC Designer 5.2 Release. Combined the section s ‘PLT User Modules’ and ‘Code Example’ section in a single section ‘Using PLT User Module in a Code Example’. Combined Transmitter and Receiver in a single project. Removed Voltage measurement. Made a separate project for Host interface. Updated images and firmware for UART and I2C Host Interfaces. Added section “Low Power Firmware”. Added section “Estimating CY8CPLC20 Power Consumption”.
*E	4090968	ADIY	08/08/2013	Minor FW update in PLC_I2C_Host project. Updated projects for PSoC Designer 5.4. Updated Worldwide Sales and Design Support.
*F	4162892	ADIY	10/16/2013	Correct error in document history
*G	4402358	ROIT	06/09/2014	Removed Reference to CY3275 kit Updated Figure 29 to reflect Designer 5.4
*H	4698464	SNVN	03/27/2015	Updated software version to PSoC Designer 5.4 CP1 Table 2. Average Power Consumption updated Reference to CY3275 kit removed in Figure 33. PLC Hardware Test Setup

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 |PSoC4|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