**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.

# THIS SPEC IS OBSOLETE

Spec No: 0001-60934

Spec Title: AN60934 - REMOTE, HIGH BRIGHTNESS LED CONTROL USING POWERPSOC(R) AND POWERLINE COMMUNICATION (PLC)

Replaced by: NONE

AN60934

# Remote, High Brightness LED Control Using PowerPSoC® and Powerline Communication (PLC)

**Associated Project: Yes**
**Associated Part Family: CY8CPLC20**
**Software Version: PSoC® Designer™ 5.4**
**Associated Application Notes**: AN52478, AN51012

AN60934 describes how to implement a lighting solution with Cypress's PowerPSoC® High Brightness LED Controller and Cypress's Powerline Communication (PLC) solution. The attached code example controls up to four High Brightness (HB) LEDs based on the color information received from a PLC device (for example, CY8CPLC20). This code example uses the Master/Transmitter and Slave/Receiver auto-node discovery code example from the Application Note "PLC - LED Lighting Control using Powerline Communication - AN58717."

## Introduction

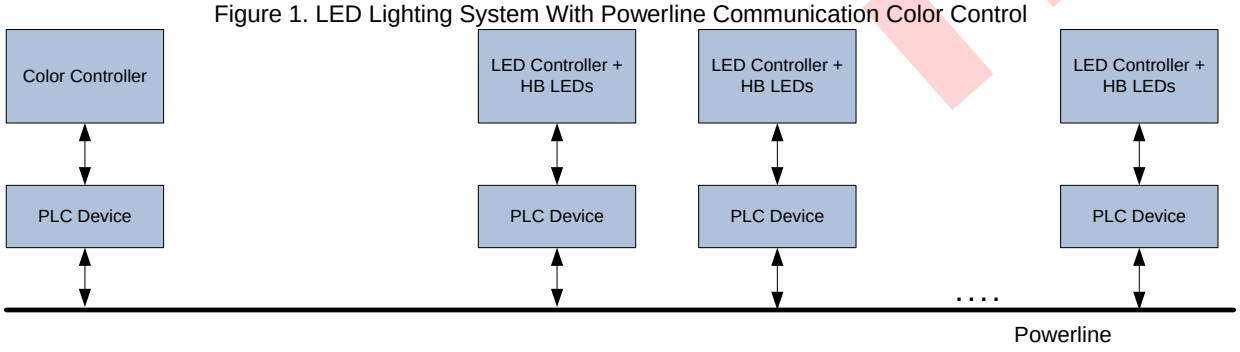
In recent years, lighting technology has been evolving at a much faster pace than ever before. Consumers are more conscious about energy efficiency and have been shifting from incandescent light bulbs to compact fluorescent light bulbs (CFLs), and recently to High Brightness (HB) LED light bulbs. While the main benefit is energy efficiency, LED lighting has an additional feature of being able to emit different colors from the same bulb. The traditional method of implementing HB-LED lighting systems involves using a microcontroller or a system-on-chip to interface with high-power discrete devices such as constant current drivers and MOSFET switches.

PowerPSoC® combines the classic PSoC core with high-performance power electronics. The result is an integrated, intelligent power electronics solution in a single QFN package. PowerPSoC significantly reduces the cost, part count, and board space while retaining performance, which makes it ideal to use in lighting systems to control multiple light fixtures.

Typically, lighting designs require a color controller, which sets the dimming values for the LEDs. The control mechanism can be on the PowerPSoC itself or at a remote location. When the controller is at a remote location, the system requires a communication interface between the controller and the PowerPSoC. The interface can be wired or wireless (for example, LAN, UART, DALI, and so on). However, wired interfaces require additional wiring to connect the different light fixtures to the color controller and this adds to the installation cost. For a light fixture to be controlled by one of these protocols, additional wires need to be installed behind the walls and junction boxes need to be rewired. Connecting a new light fixture afterwards also requires additional effort.

A technology that addresses the problem of installation cost is PLC, which uses the existing powerlines to send the lighting control information. With Cypress's easy-to-use PLC solution, an LED lighting system with color control (shown in Figure 1) can be developed without any modifications to the infrastructure.

Figure 1. LED Lighting System With Powerline Communication Color Control



## Cypress PowerPSoC Overview

The PowerPSoC family of devices combines up to four independent channels of constant current drivers. These drivers feature hysteretic controllers with PSoC, which contains an 8-bit microcontroller, configurable digital and analog peripherals, and embedded flash memory. The LED drivers operate from 7 V to 32 V and drive up to 1 A of current per channel using internal MOSFET switches. It also drives more than 1 A of current by using external switches. The device supports common power topologies such as buck and boost.

PowerPSoC features three options of hardware modulators, including the Cypress's patented Precise Illumination Signal Modulation (PrISM™) scheme, which interfaces with hysteretic controllers and modulates the signal to provide dimming.

For more information on PowerPSoC, refer to the datasheets and application notes available at http://www.cypress.com/powerpsoc.

## Cypress PLC Solution Overview

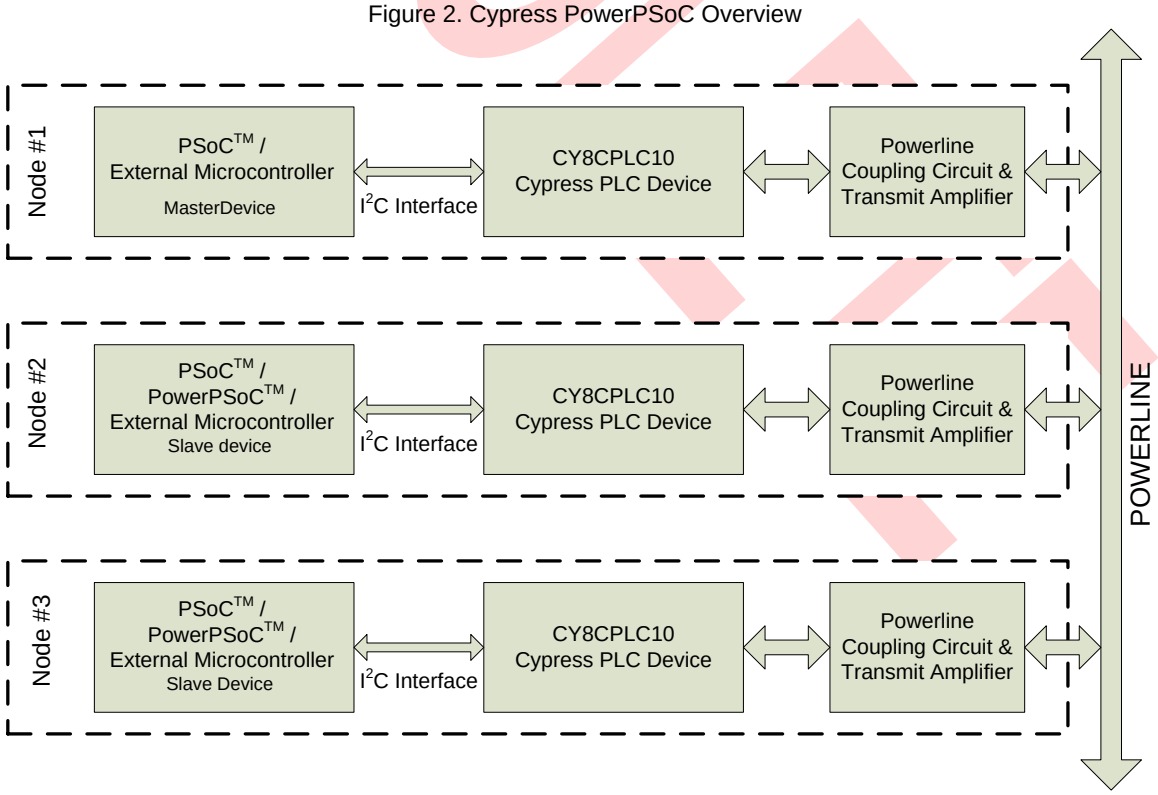
Cypress provides a robust solution to implement PLC for low-bandwidth applications. CY8CPLC10 and CY8CPLC20 (Cypress's PLC chips) integrate an FSK modem and powerline network protocol into a single-chip solution. CY8CPLC10 is a fixed-function device with $I^2C$ interface, whereas CY8CPLC20 is programmable device with multiple interface options (UART, I2C, SPI, etc). Solutions using either of these devices complies with key standards governing the use of powerline for communication in Europe and North America.

CY8CPLC10 and CY8CPLC20 are designed for systems that require a communication interface over commercial high voltage (HV) or low voltage (LV) powerlines. Typically, these systems consist of a microcontroller or processor along with other electronic components that implement the host application functionality. The PLC interface is provided by integrating the CY8CPLC10 orCY8CPLC20 with a powerline coupling circuit. In this system, the host application is running on the PowerPSoC device and connects to the CY8CPLC20 via an $I^2C$ interface.

For more information on CY8CPLC10, refer to the data sheet available at http://www.cypress.com/?rID=38236.

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

Figure 2. Cypress PowerPSoC Overview

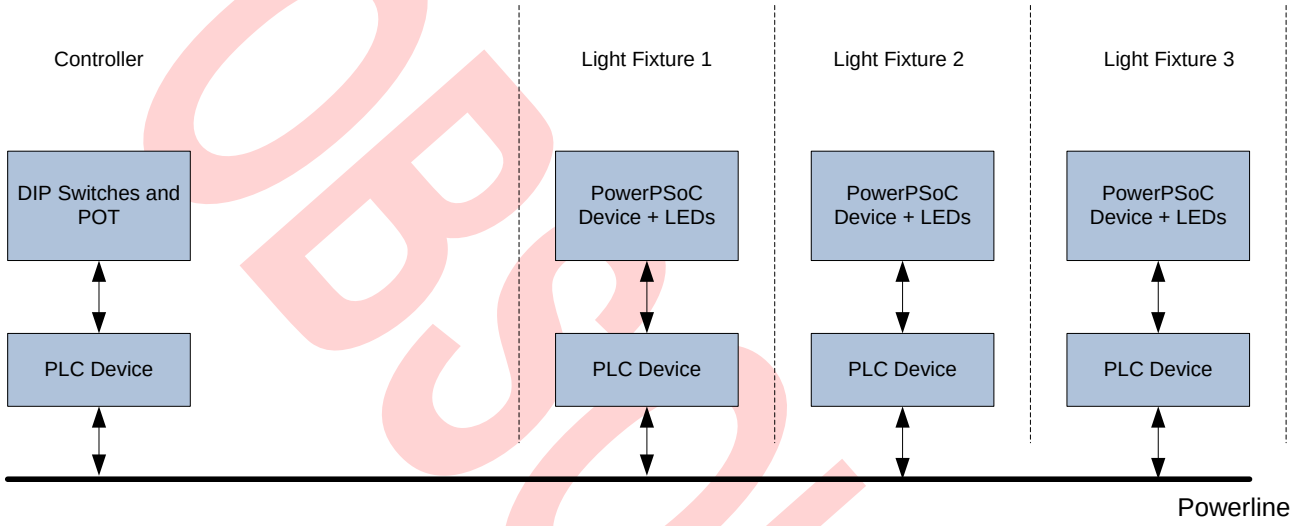


## LED Control System Implementation

A typical LED system is composed of a controller and multiple light fixtures. A system diagram of this is shown in Figure 3.

There are multiple ways to implement a PLC-enabled light fixture using Cypress's PLC solution.This application note describes how to implement this solution usingPowerPSoC with CY8CPLC20: The PowerPSoC directly drives the high-brightness LEDs and reads the data from the CY8CPLC20 device via an $I^2C$ interface

Figure 3. Block Diagram of LED Controller and Light Fixtures




## Addressing Light Fixtures

An addressing method is employed to control each light fixture individually on the line. Because all the fixtures share the same communication line (powerline), they will receive all messages from the controller. If each fixture has a unique address, then it will only process messages that have a destination address that matches its own address.

The Cypress PLC solution has several options for addressing. They are logical (8-bit), extended logical (16-bit), and physical (64-bit). Each Cypress PLC device comes with a unique physical address, which simplifies the initialization process for mapping the devices in the network. The most commonly used mode is the logical addressing because it reduces the packet size and most systems have less than 256 devices.

CY8CPLC20 can have the address set in the firmware itself or using external dual inline package (DIP) switches, depending on firmware implementation. The Cypress CY3274 PLC evaluation kits have on-board DIP switches that can be used as logical address select pins.

## Sending Color Information

The controller changes the light fixture's color by sending data in the PLC packet. There are two common ways to send the color information (using the standards published by the International Commission on Illumination, or CIE):

1) CIE Coordinates: The color is represented by x and y coordinates of the color on the CIE color chart and the intensity of that color.

2) Direct LED Control: The intensity of each LED is represented by a byte of data.

The PLC packet (see Figure 4) contains a command ID, which is used to indicate the type of message and data payload, which contains the color information. For more information on all the fields in the PLC packet, refer to the CY8CPLC20 datasheet.

Figure 4. PLC Packet Structure

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

To send the CIE coordinates and intensity for individual LEDs, the application uses a Command ID of 0x31. The payload in this case contains the 16-bit x-coordinate, 16-bit y-coordinate, and 8-bit intensity. This method is preferred because each light fixture may have a different number or type of LEDs. These differences can be managed by the light fixture when performing the color mixing algorithm to determine the brightness level of each LED.

To send the direct LED dimming values for individual LEDs, the application uses a Command ID of 0x30. The payload in this case contains the dimming values for individual channels. To control three LED channels (Red, Green, and Blue) the payload contains three bytes. To control four LED channels (Red, Green, Blue, and Amber) the payload contains four bytes.

The Master example project in this application note will send Direct LED control data to slave.

## Interfacing PowerPSoC to PLC

The PLC family of devices uses a memory array structure that sets the configuration of the powerline communication interface, contains the transmit data and receive data, and reports the status of operation. The memory array is exposed to the host microcontroller via an I²C interface.

When the PLC device receives a packet, it extracts the payload length, source address, command ID, and the data from the packet. Then, it copies the values to the memory array at the registers shown in Table 1. The PowerPSoC can either access this array directly for LED data or the CY8CPLC20 can extract the actual LED color data values, which is then read by PowerPSoC over I²C.

In the implementation of project in this application note, the CY8CPLC20 device extracts the LED color data and places it into separate buffer for the PowerPSoC to read. The PowerPSoC device polls the PLC device for new packets. When it reads that the data ready byte is set, it reads the packet for the color information to control the individual LED channels.

In case you need to design a system where the PLC memory array has to be directly accessed using the I²C interface, refer to the application note "Designing an External Host Application for Cypress's Powerline Communication IC CY8CPLC10 – AN52478".

Table 1. PLC Application level Memory Array for RGB color information only

| Offset | Register Name | Access | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | Data Ready | RW | Non-Zero value when a new PLT packet from Master is received | | | | | | | |
| 0x01 | RED | RW | 8-bit Red color intensity value | | | | | | | |
| 0x02 | GREEN | RW | 8-bit Green color intensity value | | | | | | | |
| 0x03 | BLUE | RW | 8-bit Blue color intensity value | | | | | | | |

The following tables provide information on direct PLC memory array access, if needed.

The PLC memory array settings for receiving the CIE color coordinates are shown in Table 2. The local logical address (Local_LA_LSB) must be set to 0x02, 0x03, or 0x04 if it is to be compatible with the CapSense application note. The RX_Override bit is set to '1' so that any new message overwrites the existing message in the memory array. Therefore, the most recent data is available.

Table 2. Receiver Memory Array Settings for CIE Coordinates

| Offset | Register Name | Access | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x01 | Local_LA_LSB | RW | **0x02 – 0x04** (depending on the address select pins) | | | | | | | |
| 0x05 | PLC_Mode | RW | TX_Enable = '**1**' | RX_Enable = '**1**' | | | RX_Override = '**1**' | | | |
| 0x40 | RX_Message_INFO | RW | New_RX_Msg = '**1**' | RX_DA_Type ='**0**' (Direct) or '**1**' (Broadcast) | RX_SA_Type = '**0**' (Logical) | RX_Msg_Length = '**00101**' | | | | |
| 0x41 | RX_SA | R | Remote Node Source Address(8 Bytes) = **0x01** | | | | | | | |
| 0x49 | RX_CommandID | R | **0x31** | | | | | | | |
| 0x4a | RX_Data[0] | R | MSB of CIE x-coordinate | | | | | | | |
| 0x4b | RX_Data[1] | R | LSB of CIE x-coordinate | | | | | | | |
| 0x4c | RX_Data[2] | R | MSB of CIE y-coordinate | | | | | | | |
| 0x4d | RX_Data[3] | R | LSB of CIE y-coordinate | | | | | | | |
| 0x4e | RX_Data[4] | R | Brightness value | | | | | | | |

The PLC memory array settings for receiving the RGB Direct Control message are shown in Table 3. Note the different message length, command ID, and data payload.

Table 3. Receiver Memory Array Settings for Direct RBG Control

| Offset | Register Name | Access | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x01 | Local_LA_LSB | RW | **0x02 – 0x04** (depending on the address select pins) | | | | | | | |
| 0x05 | PLC_Mode | RW | TX_Enable = '**1**' | RX_Enable = '**1**' | | | RX_Override = '**1**' | | | |
| 0x40 | RX_Message_INFO | RW | New_RX_Msg = '**1**' | RX_DA_Type ='**0**' (Direct) or '**1**' (Broadcast) | RX_SA_Type = '**0**' (Logical) | RX_Msg_Length = '**00011' or '00100**' | | | | |
| 0x41 | RX_SA | R | Remote Node Source Address(8 Bytes) = **0x01** | | | | | | | |
| 0x49 | RX_CommandID | R | **0x30** | | | | | | | |
| 0x4a | RX_Data[0] | R | Red Dimming Value | | | | | | | |
| 0x4b | RX_Data[1] | R | Green Dimming Value | | | | | | | |
| 0x4c | RX_Data[2] | R | Blue Dimming Value | | | | | | | |
| 0x4d | RX_Data[3] | R | Amber Dimming Value | | | | | | | |

## Code Example

The code example is developed using Cypress's CY3267 – PowerPSoC Lighting Evaluation Kit and CY3274 – Programmable High Voltage Powerline Communication Development Kit.

The PowerPSoC device in this application performs two tasks:

1. **LED Color Control:** The PowerPSoC device controls three LEDs independently using the PowerPSoC resources.
2. **Interface to the CY8CPLC20:** The PowerPSoC device interfaces to the powerline using the CY8CPLC20. It receives the color information from this PLC device over I²C.

There are three sections to the complete project implementation:

Master/Transmitter: This project works on the CY3274 kit and is responsible for auto node discovery, binding and sending the individual intensity values for RGB LED over powerline.

Slave/Transmitter: This project works on the CY3274 kit and binds with the Master PLC kit. Also, it receives the PLT packets with LED values, extracts the content and updates the application-level memory array for PowerPSoC to read. It acts as an I²C slave.

PowerPSoC: This project works on the CY3267 kit and drives the three LEDs after reading updated values from the Slave PLC kit over an I²C interface. This project serves as an I²C Master.

All the firmware is developed using Cypress's PSoC Designer™ IDE. The PowerPSoC device used is the CY8CLED04OCD, whereas the PLC device used is the CY8CPLC20.

The following user modules are used in this application.

1. **PRISM16HW (Three Instances)**

   This user module controls the brightness of the individual LEDs by modulating the pulse width of the signal that controls the LED driver. Precision Illumination Signal Modulation (PrISM) technology is implemented using a high-resolution Stochastic Signal Density modulation (SSDM) technique. This is accomplished by comparing the output of a pseudo-random counter with a signal density value. PrISM spreads the energy at all higher frequencies to reduce the EMI. Three instances of this user module are required to control three LED channels. The project is capable of driving one more LED in case 4-LED control is supported by the Master.

2. **CURSENSEHW (Three Instances)**

   The high-side Current Sense Amplifiers (CSA) provides a differential sense capability to measure the voltage across the current sense resistors in this application. This is needed for the constant current control of the LEDs. Again, three instances of this user module are required to control three LED channels, but can be easily extended to use four LED channels.

3. **HYSTCTRL (Three Instances)**

   This user module is configured to drive the internal PowerPSoC FET with default gate drive strength and get its feedback from the CSAs.

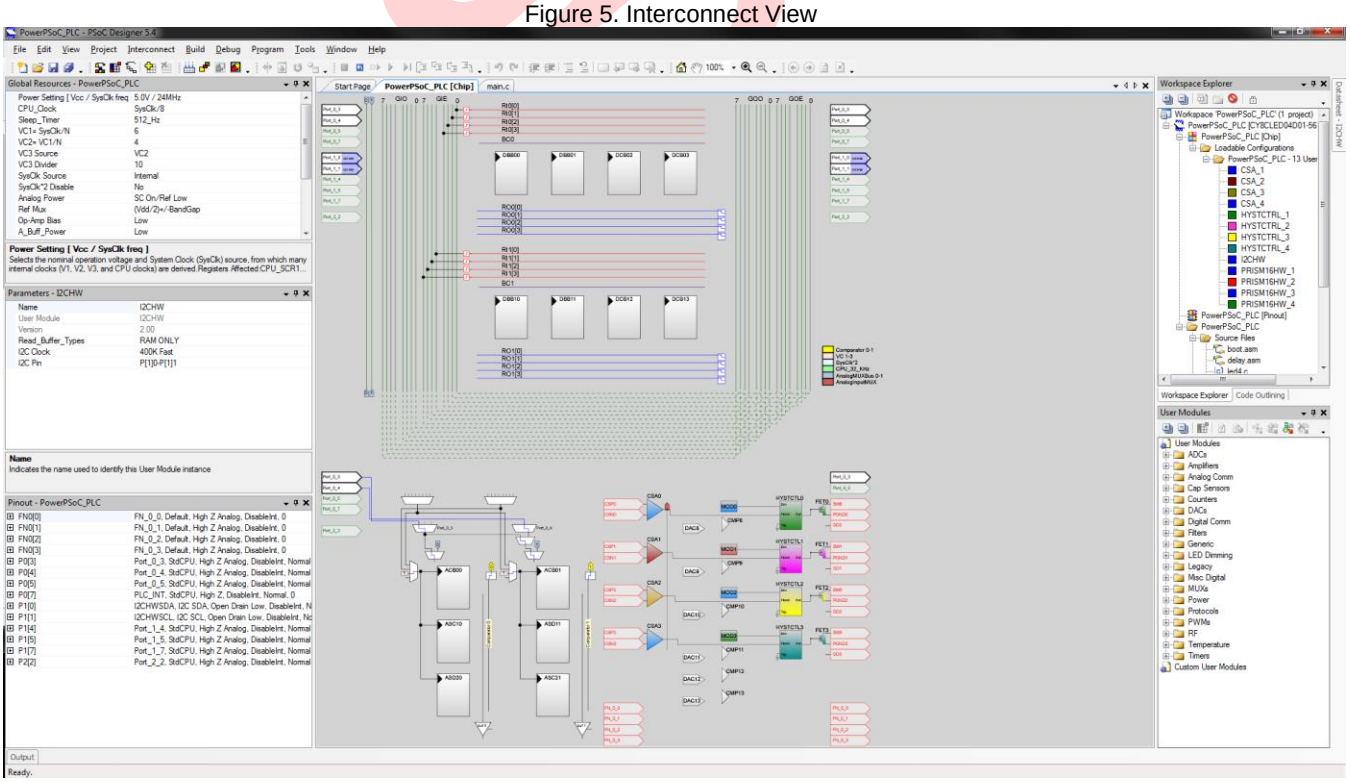
   Refer to application note, PowerPSoC® Firmware Design Guidelines – AN51012 for more details on PWM16HW, CURSENSEHW and HYSTCTRL user modules.

4. **I2CHW (One Instance)**

   The CY8CPLC20 device is interfaced through $I^2C$. This user module enables the PowerPSoC device to interface to the PLC device.

The following figure shows the interconnect view for this application. Refer to the section User Module Properties for the user module parameters and color control using PowerPSoC.

Figure 5. Interconnect View

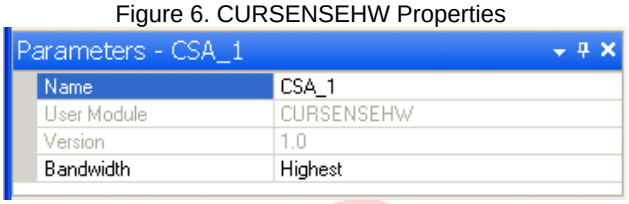


## User Module Properties

### Properties for CURSENSEHW

The Bandwidth is set to the highest setting.

Figure 6. CURSENSEHW Properties



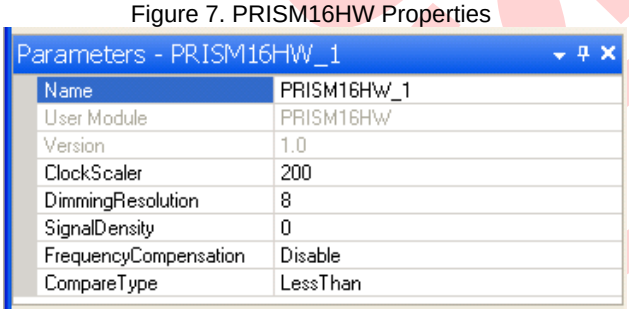

### Properties for PRISM16HW

The Clock Scalar is set to 200 and the dimming resolution is set to 8 bits, which means the signal density (dimming value) range is 0-255. The input to the PRISM16HW block is SysClk (24 MHz). With these settings, the frequency of the PWM output is:

$$PWMout = \frac{SysClk}{(ClockScalar * Period)}$$

This results in a frequency of 469 Hz. The signal density (average pulse width) is controlled through the application code.

Figure 7. PRISM16HW Properties




### Properties for HYSTCTRL

The Gate Driver is set to Internal (using internal FETs)
The Feedback Input is set to the corresponding current sense amplifier (CSA), so that there is a closed loop for current control. The CSA gain is fixed at 20.
The DAC Voltage range is set to 2.6 VStep10 mV.

The following calculations determine the $Ref_{High}$ and $Ref_{Low}$ values for this user module. The system is designed for a maximum system current of 1 A. Assuming the total current required by the PowerPSoC low voltage circuitry is 100 mA, this gives 900 mA current available for the four LED channels. That is, 225 mA for each LED channel.

The LED current oscillates between a minimum (known as valley) and a maximum (known as peak) with an average of 225 mA. For ripple less than 25%, the current values should be:

$$I_{peak} = 250mA$$
$$I_{valley} = 200mA$$
$$I_{avg} = 225mA$$

On the CY3267 board, sense resisto is 0.1 ohms.

When using the CY3267 board with $R_{sense} = 0.10\Omega$, the peak and valley voltages are.

$$V_{(I=Ipeak)} = I_{peak} * R_{sense} * CSA_{gain}$$
$$V_{(I=Ivalley)} = I_{valley} * R_{sense} * CSA_{gain}$$

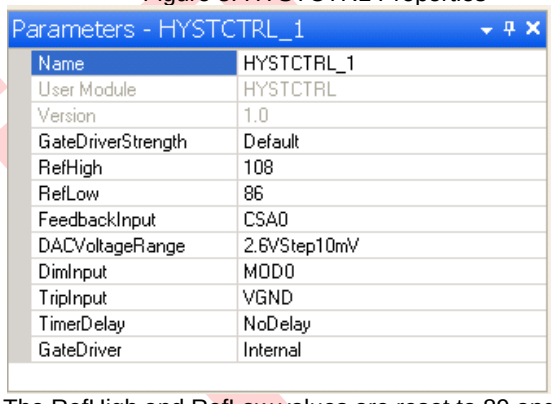
$$V_{(I=Ipeak)} = 0.5V$$
$$V_{(I=Ivalley)} = 0.4V$$

The DAC reference values are:

$$REF_{high} = V_{(I=Ipeak)} / DACrange * 255$$
$$REF_{valley} = V_{(I=Ivalley)} / DACrange * 255$$

$$REF_{high} = 0.5 / 2.6 * 255 = 49$$
$$REF_{valley} = 0.4 / 2.6 * 255 = 39$$

Figure 8. HYSTCTRL Properties

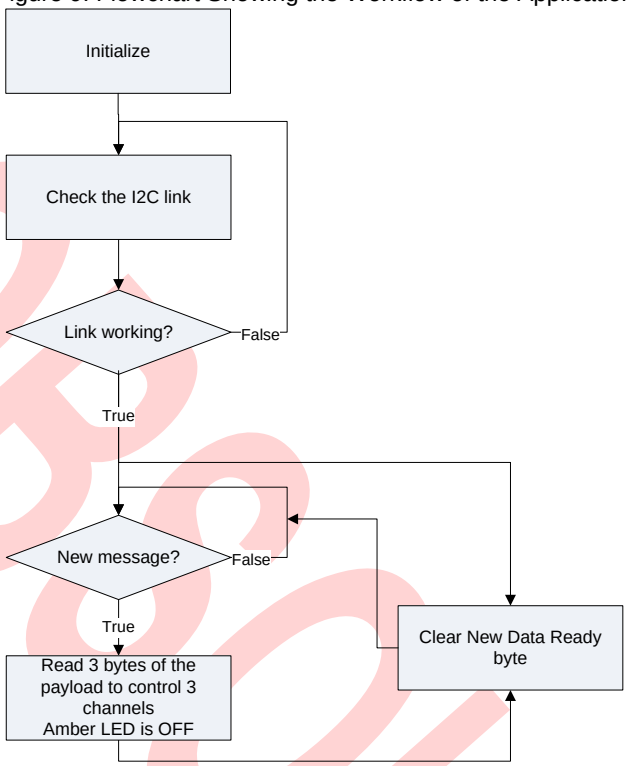



The RefHigh and RefLow values are reset to 89 and 39 in firmware.

## Application Code Flow

Figure 9 explains the workflow of this application.

Figure 9. Flowchart Showing the Workflow of the Application






## Description

When PowerPSoC starts up, it initializes all the user modules.

After initializing the user modules, the PowerPSoC writes a message over $I^2C$ to the memory array in the PLC device. It then reads back from the same memory location and compares with the written value. During this time, the PowerPSoC drives the Red LED to indicate that the $I^2C$ link has not yet been verified. If the write and read values match, it means that the $I^2C$ link between the PowerPSoC and the PLC is working. At this point, the red LED will be turned off to indicate that the $I^2C$ link has been verified.

Once the link has been established, the PowerPSoC device keeps checking for a new message by continuously polling the “Data ready” byte in the application leverlmemory array of the Slave PLC. If the byte is set, the PowerPSoC reads the next 3 bytes of data, which is interpreted as R,G and B color values.

The amber LED in this case is turned OFF.

On completion of this operation, the PowerPSoC device clears the ”Data Ready” byte.

## Hardware Implementation

This application is developed using Cypress’s CY3267 kit, which has the CY8CLED04DOCD part and LED Daughter board. The PLC interface is provided by using Cypress’s CY3274 kit, which has the CY8CPLC20 part. The following figure shows the connections for this application on the light fixture side. Follow these steps to evaluate the code example.

1. Connect the MiniProg programmer (included in the PowerPSoC kit) from the USB port on the PC to the 5-pin programming header on the PowerPSoC board.
2. Open the PowerPSoC_PLC project from the attached zip file. Ensure you have PSoC Designer 5.4. If not, then download and intstall the IDE from here.
3. Go to Build and click on “Generate/Build”. After that, click on Program -> Program Part. Slect ‘Power Cycle’ as acquire mode and click the Program button.
4. After the programming step is successful, remove the MiniProg from the board.
5. Take one CY3274 kit (which is to act as master) and connect jumper wires between following pins on the kit:

| | |
|---|---|
| J12-1 | J9-P30 |
| J12-2 | J9-P31 |

| | |
|---|---|
| J12-3 | J9-P32 |
| J12-4 | J9-P33 |
| J12-5 | J9-P34 |
| J12-6 | J9-P35 |
| J12-7 | J9-P36 |
| J18-VR | J13-P07 |
| J18-SW | J9-P37 |
| J18-LED1 | J17-P50 |
| J18-LED2 | J17-P51 |
| J18-LED3 | J17-P52 |
| J18-LED4 | J17-P53 |

J12-1 to J12-4 are DIP switches to set the logical address. J12-5 to J12-7 are on/off control for three LEDs; RGB.

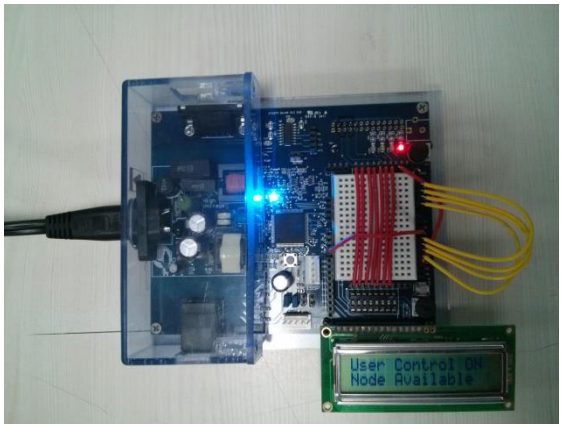
J18-VR is the onboard Potentiometer connection, which will control the intensity of the RGB LEDs.

J18-SW is the onboard user switch (S4) that allows you to look for a node, bind to it and start the user control for RGB LED.

J18-LED1 to J18-LED4 are status LEDs.

6. Connect the MiniProg to ISSP connector (J21) on the CY3274 kit.
7. Open the project 'PLC20_Auto_Node_Discovery_TX' in the attached ZIP with PSoC Designer 5.4 or greater. Click on Build -> Generate/Build.
8. After a successful build, click on Program -> Program Part. Select acquire mode as Power Cycle and click on Program Icon.
9. After the programming is completed, remove the MiniProg from J21 connector and power the CY3274 kit using the power cable. The onboard Character LCD will show User message.

Figure 10: Master/Transmitter PLC Kit setup



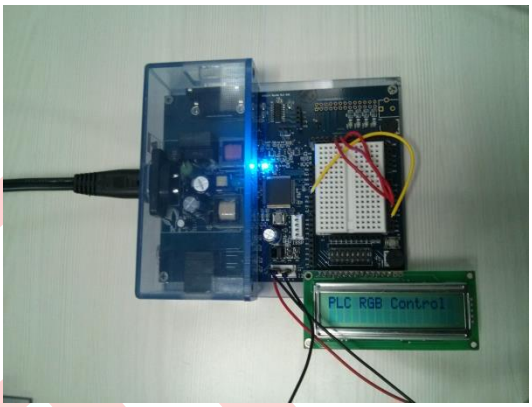

10. Take the other CY3274 kit (which is to act as PLC slave) and connect jumper wires between the following pins on the kit:

| | |
|---|---|
| J18-LED1 | J9-P30 |
| J18-LED2 | J9-P31 |
| J18-LED3 | J9-P32 |

J18-LED1 to J18-LED3 are LED control status LEDs.

11. Connect the MiniProg to ISSP connector (J21) on the Cy3274 kit.
12. Open the "LED16_Auto_Node_Discovery_RX" project from the attached ZIP with PSoC Designer 5.4 or greater. Click on Build -> Generate/Build.
13. After a successful build, click on Program -> Program part. Select acquire mode as power Cycle and click on Program icon.
14. After the programming is completed, remove the MiniProg from J21 connector and power the CY3274 kit using the power cable. The onboard Character LCD will show User message "PLC RGB Control".

Figure 11: Slave/Receiver PLC Kit setup

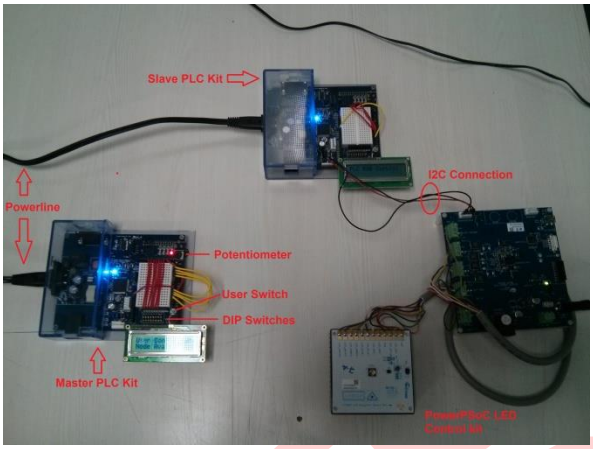



15. Connect 3 jumper wires from CY3274 kit to CY3267 kit between the following headers to complete the I2C connection:

| | |
|---|---|
| J15-4 (Ground on CY3274) | J12-4 (Ground on CY3267) |
| J15-2 ($I^2C$ Data on CY3274) | J12-2 ($I^2C$ Data on Cy3267) |
| J15-1 ($I^2C$ Clock on CY3274) | J12-1 ($I^2C$ Clock on CY3267) |

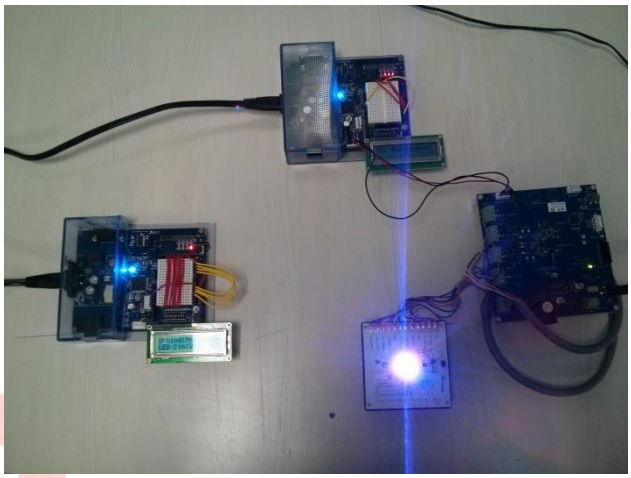
16. Connect the two jumpers on JP3 and JP4 on CY3274 kit. This connects the $I^2C$ Clock and Data lines to pull-up resistor onboard, required for I2C Communication.
17. Connect the 12 V wall wart adapter to the mains and the other end to the 12 V connector on the PowerPSoC board.

Figure 12: Board setup




18. Press the reset button on either the PowerPSoC or PLC board. The red LED should briefly flash on, and then turn off.
19. Press the S4 switch on Master PLC kit. The same user button will initiate binding with Slave PLC kit as well as entering the User control mode. Only after entering the user control mode will the DIP switches 5-7 on S3 as well as the R47 Potentiometer on Master CY3274 kit control the RGB LEDs on the Power PSoC kit.
20. If no action is taken on the CY3274 Master kit for 10 seconds, then then system returns to User Control ON Message on Character LCD. To control LEDs further, press S4 switch again to unter the user control mode.

Figure 10. LED Control in action



## Summary

The design presented here is intended to be a ready-to-use PLC controlled lighting solution. The implementation is simple and covers direct LED control, using color control through CIE coordinates. The application uses Cypress's state of the art PLC solution to reliably communicate the color information from the LED controller to the LED light fixtures with Cypress's PowerPSoC device performing high brightness LED Control. The system in this application is developed in a modular way to enable easy changes of the architecture and reuse of the relevant code.

For more information (data sheets, application notes, evaluation GUIs) on PowerPSoC, please visit www.cypress.com/go/powerpsoc. Similarly, for more information on PLC, please visit www.cypress.com/go/plc .

## Document History

Document Title: AN60934 - Remote, High Brightness LED Control Using PowerPSoC® and Powerline Communication (PLC)

Document Number: 001-60934

| Revision | ECN | Submission Date | Description of Change |
|---|---|---|---|
| ** | 2912098 | 04/13/10 | New Application Note. |
| *A | 3247568 | 05/03/11 | Updated to PSoC Designer 5.1 |
| *B | 4400077 | 6/3/2014 | Updated to PSoC Designer 5.4<br>Obsolete kit CY3272 and CY3268 references replaced by CY3274 and CY3267 kits<br>Example project modified completely to showcase LED control using CY8CPLC20 device.<br>Images of setup added |
| *C | 5726360 | 05/08/2017 | Updated logo and Copyright.<br>Completing Sunset Review. |
| *D | 7081279 | 02/02/2021 | Obsolete this application notes as it uses obsolete Power PSoC part. |

## 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

| | |
|---|---|
| ARM® Cortex® Microcontrollers | cypress.com/arm |
| Automotive | cypress.com/automotive |
| Clocks & Buffers | cypress.com/clocks |
| Interface | cypress.com/interface |
| Internet of Things | cypress.com/iot |
| Memory | cypress.com/memory |
| Microcontrollers | cypress.com/mcu |
| PSoC | cypress.com/psoc |
| Power Management ICs | cypress.com/pmic |
| Touch Sensing | cypress.com/touch |
| USB Controllers | cypress.com/usb |
| Wireless Connectivity | cypress.com/wireless |

### PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

### Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

### Technical Support

cypress.com/support



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