USB to DMX512 Converter

Authors: Petro Koblyuk, Anastasiya Yablokova

Associated Code Example: Yes
Associated Part Family: CY8C24x94, CY8CLED0x,CY8CLED0xD0x
Software Version: PSoC® Designer™ 5.4
Related Application Notes: None

AN45022 describes a USB to DMX512 converter, which uses Cypress's CY8C24x94 family of devices. The included GUI application allows you to transfer data through the DMX512 interface as the network master. This application note also explains the major aspects of the USB device and the DMX512 operation. In addition, an example of communications between the GUI application and the USB to DMX512 converter is provided. Design files for a DMX512 interface board are also attached.

Introduction

Computer-controlled equipment, consoles, and control panels are responsible for controlling stage lighting in theaters, discos, concert halls, and similar locations. These devices use the DMX512 interface for interconnection and communication. The maximum length of the interface cable is 1000 m, allowing flexibility in placing the required control devices.

Networked systems require a method of sending user controls to remote devices. However, special, dedicated hardware controllers that implement this functionality can be expensive. The USB to DMX512 converter discussed in this application note utilizes Cypress's PSoC® platform to offer a cost-effective, easy-to-use, and serviceable method for implementing this conversion.

Table 1 lists the characteristics of the USB to DMX512 converter.

Table 1. USB to DMX512 Converter Characteristics

Characteristics	Value
Data refresh rate through DMX interface	40 Hz (at 512 channels)
Maximum delay of output signal relative to the USB input stream	(1/40 Hz) = 25 ms
Number of DMX channels	24–512
Current consumption (active mode)	71.3 mA
Current consumption (suspend mode)	224 μA

DMX Protocol

The DMX protocol was defined by the United States Institute for Theatre Technology (USITT) in 1986. This protocol was developed to be the standard interface for data communication between devices used for lighting. In 1990, it was revised and is now known as DMX512. This application note treats the terms "DMX" and "DMX512" as interchangeable.

The DMX protocol uses the RS-485 standard on the physical layer, which defines the electrical interface levels, voltage, and currents on a bus. This standard specifies a receiver and transmitter "ground" connection to the cable shielding to lower the influence of external noise and improve electrical safety. However, the voltage of the transmitter and receiver "ground" can be different, which causes current to flow through the cable wire shield. If too much current flows through the shielding, the DMX cable and electronic circuit may be destroyed. To avoid this possibility and improve the safety of the devices used, it is necessary to implement galvanic isolation of DMX devices.

The DMX data stream passes in the form of a continually repeated burst. This data burst consists of the preamble, which informs the receiver of the start of the data packet. These bits are followed by the stream of serial data, which contains the values for each channel, ranging from 1 to 512. Figure 1 shows the structure of the DMX512 packet and Table 2 lists the duration for each bit. The bit rate of the DMX protocol is 250 kbaud. Therefore, the duration of every bit is 4 μs. The quantity of used channels is not fixed but is limited to 24–512 by the standard. If information is sent on all 512 channels, then the information is updated at the rate of 44 Hz. The maximum frequency of transmitted information is 836 Hz, corresponding to 24 used channels.

Figure 1. Structure of DMX512 Signal

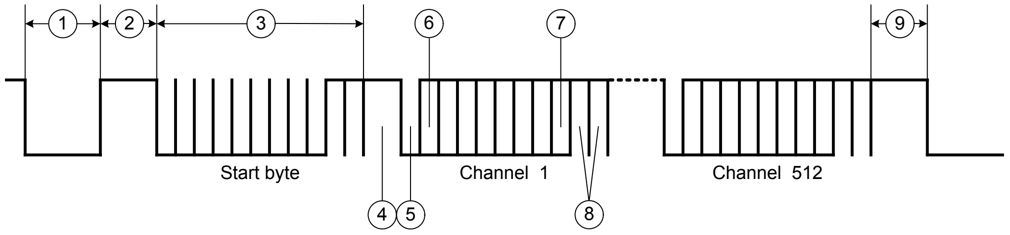


Table 2. DMX Package Field

Reference	Description	Duration
1	Space for Break (Reset)	Minimum 88 µs
2	Mark After Break (MAB)	8 µs–1 s
3	Slot Time	44 µs
4	Mark Time Between Slots	0 µs–1 s
5	Start Bit	4 µs
6	Least Significant Data Bit	4 µs
7	Most Significant Data Bit	4 µs
8	Stop Bit	4 µs
9	Mark Before Break (MBB)	0 µs–1 s

USB to DMX Converter

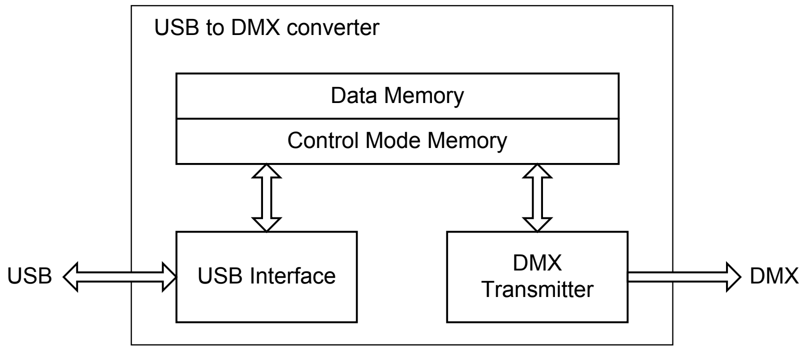
The USB to DMX converter block diagram is shown in Figure 2. The device transmits data through the DMX bus. USB and DMX buses have different protocols and work at different rates. As a result, the DMX transmitter cannot work with the USB interface directly.

To implement this protocol conversion, some additional memory buffers must be used. The USB receiver and DMX transmitter operate with these buffers directly, but at different times. This allows the device to operate effectively, and the previous data is transmitted when the new data is absent.

The USB Interface receives data from the USB device and saves it in additional buffers (the data memory buffer or control mode memory buffer, depending on the received command; see Figure 2). The data memory buffer contains data for all 512 DMX channels. The control mode memory buffer contains the operating modes (settings) of the transmitter. These blocks are described in detail in the following sections.

The DMX transmitter transmits data from the data memory to the DMX bus, depending on the settings in control mode memory. This method allows the previous data to be transmitted if the new data is not received through the USB interface (refer to the DMX Protocol section).

Figure 2. USB to DMX Converter Block Diagram

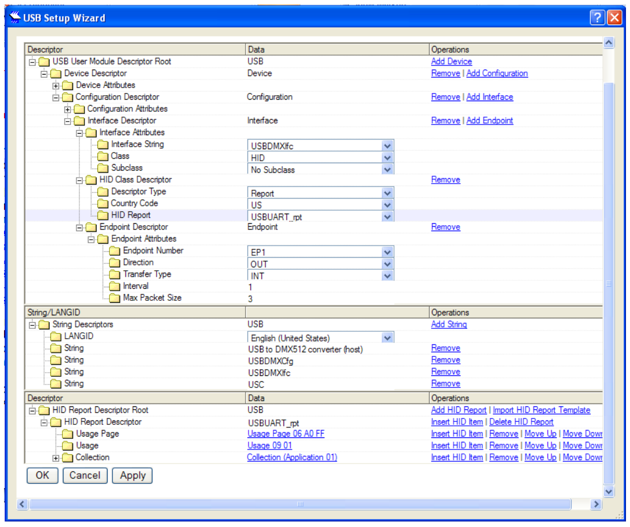


USB Interface

The USB interface is realized using the USB User Module. The USB device class used is Human Interface Device (HID). This enables the standard HID driver functions of the operating system, which is used by the GUI application for device communication. The data rate of HID-class devices is 64 KB/s, which is sufficient for the current task. Figure 3 gives the USB Full Speed (USBFS) configuration.

According to the USB specification, all devices that are connected to the USB bus must pass into the suspend mode at a command. In the suspend mode, the device consumption must not be more than 0.5 mA.

Figure 3. USB Setup Wizard



Descriptor	Data	Operations
USB User Module Descriptor Root	USB	Add Device
Device Descriptor	Device	Remove | Add Configuration
Device Attributes		
Configuration Descriptor	Configuration	Remove | Add Interface
Configuration Attributes		
Interface Descriptor	Interface	Remove | Add Endpoint
Interface Attributes		
Interface String	USBDMXIfc	
Class	HID	
Subclass	No Subclass	
HID Class Descriptor		Remove
Descriptor Type	Report	
Country Code	US	
HID Report	USBUART_rpt	
Endpoint Descriptor	Endpoint	Remove
Endpoint Attributes		
Endpoint Number	EP1	
Direction	OUT	
Transfer Type	INT	
Interval	1	
Max Packet Size	3	

String/LANGID		Operations
String Descriptors	USB	Add String
LANGID	English (United States)	
String	USB to DMX512 converter (host)	Remove
String	USBDMXCfg	Remove
String	USBDMXIfc	Remove
String	USC	Remove

Descriptor	Data	Operations
HID Report Descriptor Root	USB	Add HID Report | Import HID Report Template
HID Report Descriptor	USBUART_rpt	Insert HID Item | Delete HID Report
Usage Page	Usage Page 06 A0 FF	Insert HID Item | Remove | Move Up | Move Dowr
Usage	Usage 09 01	Insert HID Item | Remove | Move Up | Move Dowr
Collection	Collection (Application 01)	Insert HID Item | Remove | Move Up | Move Dowr

DMX Transmitter

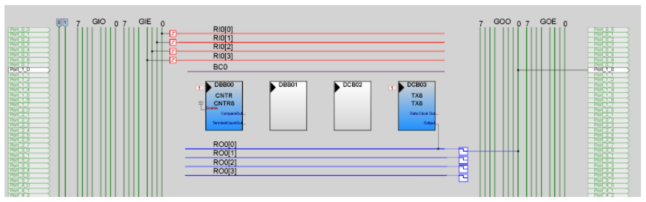
The DMX transmitter is constructed in PSoC Designer™ by using the TX8 User Module.

An 8-Bit Counter (Counter8) User Module in PSoC Designer is used to time each slot in the entire packet transmission process. Break and MAB signals (refer to the DMX Protocol section) are generated using the digital LUTs. Figure 4 shows the user module placements in PSoC Designer. The output of TX8 is routed to P1_0, which is the DMX transmitter output pin.

The Counter8 User Module parameters are set up to generate the TerminalCount interrupt every 48 μs. This corresponds to the duration of one channel transmission.

The break signal is formed when FALSE is on the LUT0 output during two Counter8 periods. To do so, 0x00 must be written in the RDI0LT0 register, which results in disconnection of the P1 [0] pin from theTX8 User Module output. RDIOLT0 is the Row Digital Interconnect Logic Table Register 0. For more details, refer to the "Register Reference" section in the PSoC TRM.

Figure 4. User Module Placement



The MAB signal is formed when the P1 [0] pin connects to the TX8 User Module output during one Counter8 period. This connection is made by writing '0x03' to the RDI0LT0 register (LUT0 output = A). However, no value is written into the TX8 User Module to transmit it over the DMX line. In this case, the logical '1' is on the TX8 User Module output.

To form other signals (see Table 2), the P1 [0] pin must be connected to the TX8 User Module output continually. The value stored in data memory for each channel is consecutively written to the TX8 User Module. As a result, the required DMX signal is formed. The following sections explain the firmware device performance.

The PSoC device incorporates flexible internal clock generators, including a 24-MHz internal main oscillator (IMO) accurate to ±4 percent over temperature and voltage. When the PSoC device is communicating on the USB bus, the IMO self-tunes to ±0.25 percent accuracy for USB communication. The DMX512 specification requires that the transmit bit period be within ±2 percent. Therefore, if this USB to DMX converter is used without connecting to the USB device, a more accurate external clock generator (for example, an external 32.768-kHz crystal) must be used to clock the PSoC device. For more details, refer to Cypress application note, AN2027 – PSoC 1 - 32.768-kHz External Crystal Oscillator.

Data Memory

The data memory buffer contains a RAM memory array of 512 bytes. Each byte in the array contains the value of an appropriate DMX channel.

Control Mode Memory

The Control Mode Memory contains two variables. One of them is bStopDMX. If a logical '1' is in this variable, the device stops transmitting data over the DMX bus. By default, this variable holds '0', which enables DMX transmission. The second variable is wDMXPacketSize, which contains the value of the transmitted channel quantity (refer to the DMX Protocol section). The value of this variable is 24 by default.

Firmware Operation

The data received from the USB bus and the data transmitted over the DMX bus run in real-time mode. Therefore, device control is achieved using interrupts. Four sources of interrupts are used (see Table 3).

Table 3. Interrupt Sources

Number	Source	Type
1	USB EP1	Data endpoint interrupt
2	Counter8	Terminal count
3	Sleep Timer	Terminal count
4	USB Wakeup	Wakeup interrupt

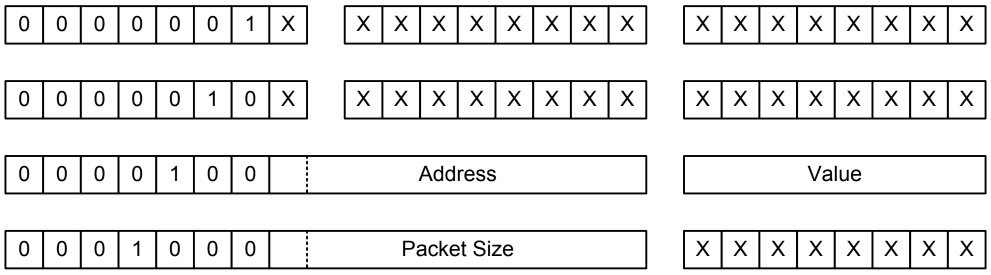
USB EP1 Interrupt

The USB EP1 interrupt occurs when the data from the PC is received. The flag for received data (set by the interrupt's ISR) is checked in the program main loop. When the flag is set, the command from the USB buffer is read. There are four types of commands (see Table 4), each having 3 bytes. The most significant 7 bits of the first byte define the operation code (see Figure 5).

Table 4. Command Types

Operation Code	Command Description	Command Byte Description
0x02 (0x01 – 7 bit)	Disables DMX transmission	1 (only opcode)
0x04 (0x02 – 7 bit)	Enables DMX transmission	1 (only opcode)
0x08 (0x04 – 7 bit)	Sends data	All 3 bytes (opcode, address and value)
0x10 (0x08 – 7 bit)	Sets DMX packet size	First 2 bytes (opcode and packet size)

Figure 5. Data Format of USB Packets



Counter8 Interrupt

The Counter8 interrupt occurs every 48 μs. Its ISR performs the action defined by the value of the wStageCntr counter variable. Figure 6 shows the counter wStageCntr values relative to the DMX package steps. The Counter8 ISR performs five actions (see Table 5).

Figure 6. Data Format Packet

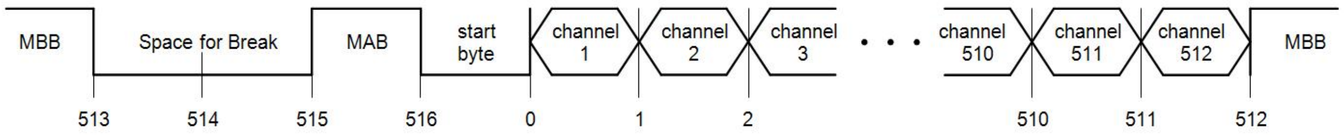


Table 5. Action Types

wStageCntr	DMX Packet Steps	Action
513, 514	Space for break	LUT0 output is set on FALSE, and wStageCntr is incremented.[1]
515	MAB	LUT0 output is set on TX8 output without sending any byte, and wStageCntr is incremented.
516	Start byte	TX8 sends 0 byte, and wStageCntr is reset to 0.
0–511	Channels from 1 to 512	TX8 sends data that corresponds to the current channel, and wStageCntr is incremented.[2]
512	MBB	wStageCntr is incremented.

Notes

1. If bStopDMX equals 1, then wStageCntr is not incremented.
2. If wStageCntr is greater than or equal to wDMXPacketSize, then wStageCntr is set to 512.

Sleep Timer Interrupt

To determine whether the suspend mode is necessary, the USB_bCheckActivity () function must be performed at least once every 3 ms. The Sleep Timer interrupt is used to do so. If the function returns 0, it is necessary to pass into suspend mode within 1 second.

The procedure to pass into suspend mode is as follows:

1. Disable the DMX transmission.
2. Wait for the transmission of the DMX package to end.
3. Disable Counter8 and Sleep interrupts.
4. Enable USB Wakeup interrupt.
5. Enter sleep mode using M8C_Sleep.

While the processor prepares to go to sleep, it may execute the commands after M8C_Sleep. Therefore, after the M8C_Sleep command, an empty loop must be used to ensure the device goes into sleep mode.

USB Wakeup Interrupt

The USB Wakeup interrupt is used to wake up the processor. The wakeup procedure is as follows:

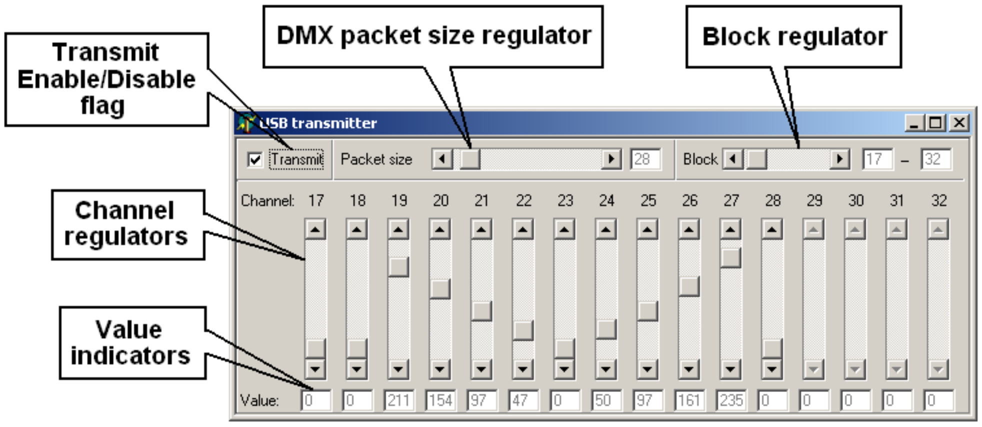
1. Disable the USB Wakeup interrupt.
2. Enable Counter8 and Sleep interrupts.
3. Restore the state of the DMX transmission.

GUI Application

The graphical user interface (GUI) application verifies the device's attributes (vendor ID, product ID, and version number) of all the present HID devices. When the device with the required attributes is found, it is opened for the next operation. Note that the GUI has been tested on a 32-bit Windows XP machine. Figure 7 shows a screen shot of the GUI application.

The program operation is set up so that user actions take place when the device opens. If any channel value changes, the new value is transmitted over the USB interface. If no changes are made to the GUI, nothing is transmitted. The program is used to change the value for each active channel, toggle DMX transmission, and set the quantity of active channels. The **Packet size** regulator is used to change the size of the DMX packet from 24 up to 512 channels. Because 16 channels are displayed at one time, the **Block** regulator is used to navigate to the desired block of channels (for example, channels 33–48).

Figure 7. GUI Application Screen Shot



Packet size: 28
Block: 17 – 32

Channel:	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Value:	0	0	211	154	97	47	0	50	97	161	235	0	0	0	0	0

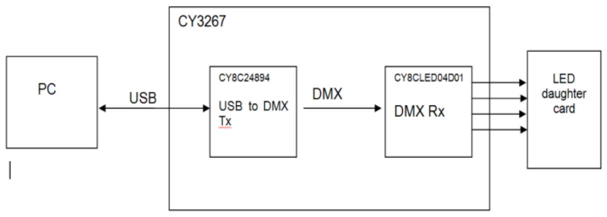
Code Example

This section describes the hardware equipment and necessary steps to evaluate the USB to DMX transmitter code example.

Hardware Equipment

- One CY3267 kit can be used for both the USB to DMX transmitter and the DMX receiver. The CY3267 has both the CY8C24894 and CY8CLED04DOCD on it. The CY8C24894 is used as the USB to DMX transmitter, and the CY8CLED04DOCD is used as the DMX receiver. The code example for the DMX receiver is included with the design guide, CY8CLED0xx0x - PowerPSoC Firmware Design Guidelines, Lighting Control Interfaces.
- The LED daughter card should be connected to the main board according to the steps in the "Main Board Wiring Scheme" section in the CY3267 – PowerPSoC Lighting Evaluation Kit Guide. The connections should be for a buck configuration. Figure 8 shows the equipment setup diagram.
- For more collateral on the CY3267 kit, visit the CY3267 – PowerPSoC Lighting Evaluation Kit web page.

Figure 8. Equipment Setup Block Diagram



Evaluating the Code Example

Follow these steps to evaluate the code example on the CY3267. You need only a simple jumper wire to connect the transmitter to the receiver. The next section explains how to interface the CY3267 board to a DMX512 cable.

1. Connect the 12-V DC supply to connector J1.
2. Set the shunt on header J3 to connect V_{DD} and V_{REG}. This uses the onboard linear regulator, which generates the 5-V rail from the 12-V supply.
3. Connect the provided USB cable from the PC to the MiniProg programmer. Then connect the MiniProg to header J16 for programming the CY8C24894.

4. Open PSoC Programmer. Set Programming Mode to **Reset** and AutoDetection to **On**.

   You can download PSoC Programmer from www.cypress.com/go/programmer.

5. Connect the MiniProg to header J16 to program the CY8C24894.

6. In PSoC Programmer, open the file *USB to DMX.hex*, which is located in the folder *\USB_to_DMX*. Press the program button to start programming. The status window should show “Programming Starting.”

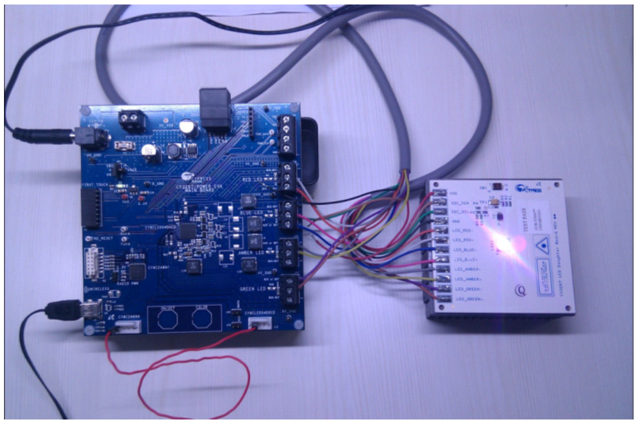
7. Connect the MiniProg to header J12 to program the CY8CLED04DOCD.

8. In PSoC Programmer, open the file *DMX_PowerPSoC.hex*, which is the associated code example for the design guide, CY8CLED0xx0x - PowerPSoC Firmware Design Guidelines, Lighting Control Interfaces. Press the program button to start programming. The status window should show “Programming Starting.”

9. Remove the MiniProg from J12 and remove the USB cable from the MiniProg. Then connect the USB cable from the PC directly to the onboard USB connector J8. This provides the USB traffic from the PC to the CY8C24894.

10. Connect a wire from pin 5 on connector J16 to pin 4 on connector J12 (as shown in Figure 9). This provides the DMX data from the CY8C24894 to the CY8CLED04DOCD.

Figure 9. USB to DMX Tx and DMX Rx

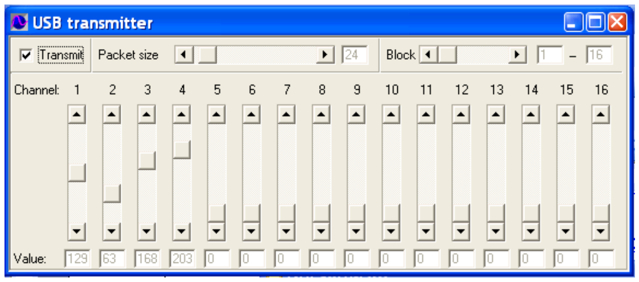


11. In the folder GUI Software, run the file *USB2DMX.exe* to start the GUI.

12. In the GUI, move the sliders of channels 1, 2, 3, and 4 to see a change in the brightness of the red, blue, amber, and green LEDs, respectively.

13. The **Packet size** regulator is used to change the size of the DMX packet from 24 up to 512 channels. Because 16 channels are displayed at one time, the **Block** regulator is used to navigate to the desired block of channels (for example, channels 33–48).

Figure 10. USB Transmitter GUI



USB transmitter

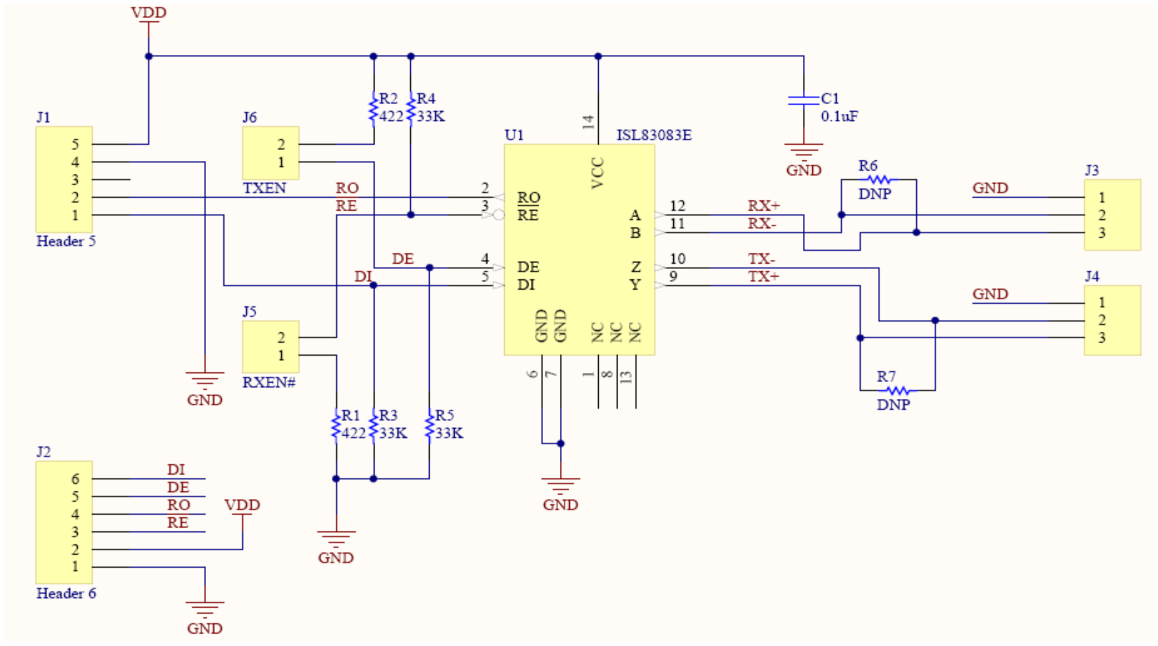
☑ Transmit Packet size 24 Block 1 – 16

Channel:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Value:	129	63	168	203	0	0	0	0	0	0	0	0	0	0	0	0

DMX512 Interface Circuit

To connect the DMX512 transmitter to a DMX512 system, an interface circuit is required between the transmitter and a DMX512 cable. It translates the 5-V TTL signals to RS485 differential signals. The data in (DI) on the TTL side of the PHY is connected to the DMX512 transmitter’s output via the port pin P1[0] (or any other input pin set by the firmware). Figure 11 shows a schematic of this circuit. The design files are attached to this document. If you have any questions about the DMX512 interface board, contact your local sales representative or create a support case at www.cypress.com/MyAccount/?id=25&techSupport=1.

Figure 11. Schematic of Interface Circuit between DMX512 Controller and EZ-Color Device



Summary

The USB to DMX512 converter is simple to implement, small, and inexpensive. It supports the DMX512 interface standard and has free unused PSoC resources to implement additional features. This application note also describes a USB-DMX transmitter. To create a DMX receiver, use a device from the CY8CLEDxx or CY8CLED0xD0x family and the DMX512Rx User Module.

For a better understanding of the DMX512Rx module, power components used to drive the LED in PowerPSoC, see the design guide, CY8CLED0xx0x - PowerPSoC Firmware Design Guidelines, Lighting Control Interfaces.

About the Authors

Name: Petro Koblyuk

Title: Application engineer

Background: Petro graduated from National University “Lviv Polytechnica” (Ukraine) in specialty “computer systems” in 2001. Working in the Ukraine Solution Center since 2005. His interests involved software and hardware development, algorithm investigation, and so on.

Contact: Petro.Koblyuk@cypressua.com

Name: Anastasiya Yablokova

Title: Application engineer

Background: Anastasiya graduated from National University “Lviv Polytechnica” (Ukraine) in specialty “computer systems” in 2004. Working in the Ukraine Solution Center since 2007.

Contact: Anastasiya.Yablokova@cypressua.com

Document History

Document Title: USB to DMX512 Converter – AN45022

Document Number: 001-45022

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2522369	YJI/AESA	07/07/2008	New Application Note.
*A	3304097	MKKU	07/29/2011	Code example was updated to PSoC Designer 5.1 The USB to DMX transmitter project was modified to work on the CY3267 evaluation kit A brief explanation of building a custom board for generating differential DMX signals was added. Design files for the DMX interface board were attached. Added sections 'Evaluating the Code Example' and 'DMX512 Interface Circuit'
*A	3304097	MKKU	07/29/2011	Updated Software Version as "PSoC® Designer™ 5.1". USB to DMX Converter: Updated description. USB Interface: Updated description. DMX Transmitter: Updated description. GUI Application: Updated description. Added Code Example. Modified attached code example to PSoC Designer 5.1. Modified USB to DMX transmitter project to work on the CY3267 evaluation kit. Attached design files for the DMX interface board.
*B	4480637	PMAD	08/21/2014	Replaced references of AN51891 with "PowerPSoC Firmware Design Guidelines, Lighting Control Interfaces" because AN51891 is obsolete. Updated Software Version as "PSoC® Designer™ 5.4". Modified attached Code Example for PSoC Designer 5.4. Updated in new template. Completing Sunset Review.
*C	4634027	PMAD	01/21/2015	Changed nomenclature of Packet size and Block to be regulators instead of sliders. Added reference to AN2027 and the CY8CLED0xx0x - PowerPSoC Firmware Design Guidelines, Lighting Control Interfaces documents. Copy edit for language and style.

Worldwide Sales and Design Support

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

Products

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

PSoC® Solutions

psoc.cypress.com/solutions

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

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

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

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