DMX512 Communication on the AC LED Lighting & Communications Developer's Kit

Version 1.0 – November 2011 *C2000 Systems and Applications Team*

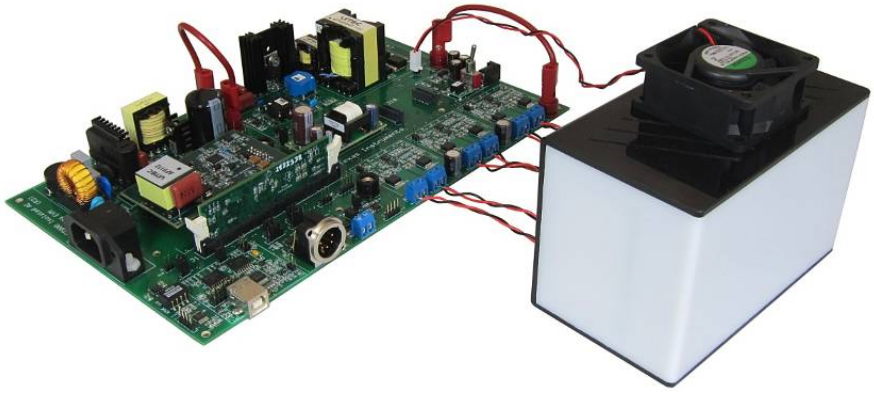


Figure 1: TMDSIACLEDCOMKIT

1 Introduction

The AC LED Lighting & Communications Developer's Kit provides a great way to learn and experiment with using a single MCU to accurately control a series of LED strings and efficiently control the power stages needed to make the LEDs work. This document explains how to use the TMDSIACLEDCOMKIT to communicate via DMX512. The accompanying Code Composer Studio project enables the MCU to control the kit's power stages, LED strings, and serve as a DMX512 slave. The MCU used to perform all these tasks is a Piccolo F28027 microcontroller.

WARNING

This EVM is meant to be operated in a lab environment only and is not considered by TI to be a finished end-product fit for general consumer use.

This EVM must be used only by qualified engineers and technicians familiar with risks associated with handling high voltage electrical and mechanical components, systems and subsystems.

This equipment operates at voltages and currents that can result in electrical shock, fire hazard and/or personal injury if not properly handled or applied. Equipment must be used with necessary caution and appropriate safeguards employed to avoid personal injury or property damage.

It is the user's responsibility to confirm that the voltages and isolation requirements are identified and understood, prior to energizing the board and or simulation. When energized, the EVM or components connected to the EVM should not be touched.

2 DMX512 Physical Layer and Protocol

2.1 Summary

DMX512 (Digital MultipleX with 512 pieces of information) is a very common lighting standard that is used primarily to control stage lighting and became the primary method for controlling dimmers and many non-safety critical special effects. Because of its relative simplicity and the stability of the standard it is also very commonly seen in other lighting systems. Depending on network topology and delays an update rate of approximatly 40Hz can be expected per slave. DMX512 is a unidirectional protocol and, because of this, employs no error correction.

2.2 Physical Layer

A DMX512 network, known as a DMX Universe, consists of one master controlling up to 512 slave devices. Slaves should be seperated into parallel buses which each contain not more than 32 units. Each bus's slaves should then be daisy-chained together and each will take their respective information off the bus. The end point of any daisy chain requires a built in 120 ohm terminating resistor. Each receiver should not load the differential line more than 125pF.

The DMX512 signals are sent via the serial connection standard EIA485 (RS-485).

The standard identifies that XLR-5 connectors be used such that male connectors are receiver ports whereas female connectors should be used as transmitting ports. All cables used should be twisted pair.

Common slave devices are dimmers, intelligent lights, or special effect machines.

Figure 2: An example of a DMX512 master

Figure 3: DMX512 XLR-5 cable

2.3 Protocol

The DMX512 protocol is characterized as a asynchronous serial data stream which runs at 250kHz. As a result each "bit" will be 4us long. DMX512 has one start bit (low), eight bits of data, 2 stop bits and no parity.

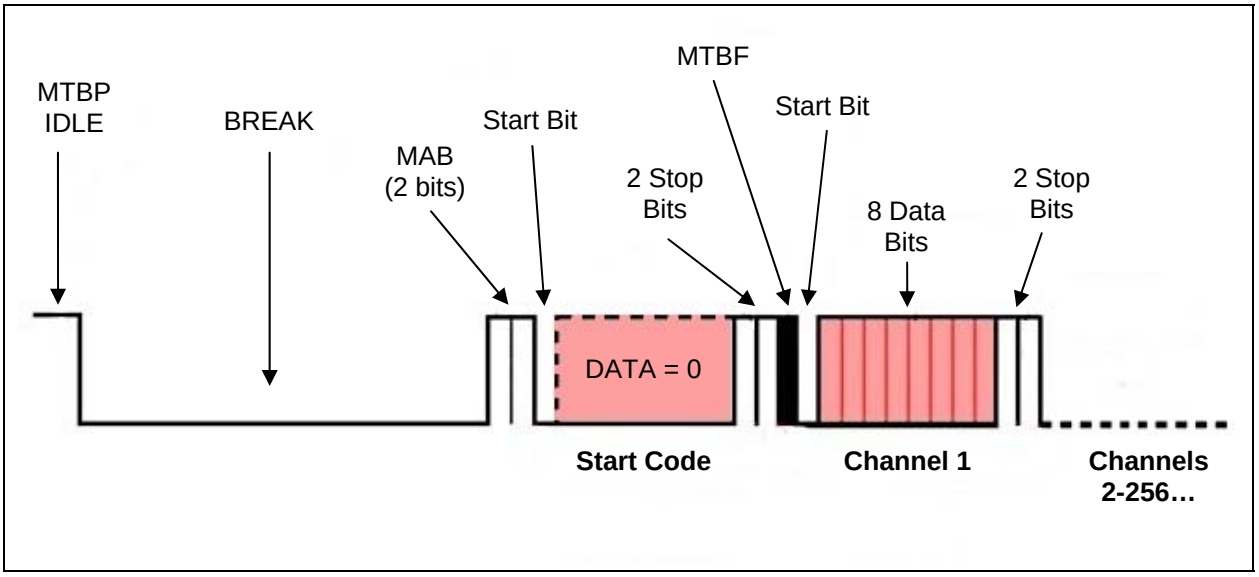


Figure 4: DMX512 Timing Diagram

	Min	Typ	Max	Unit
Break	88	**88**	1000000	us
Mark After Break (MAB)		**8**		us
Frame Width		**44**		us
Start/Data/Stop bits		**4**		us
Mark Time Between Frames (MTBF)	0	**N/A**	1000000	us
Mark Time Between Packets (MTBP)	0	**N/A**	1000000	us

Table 1: DMX512 Timing Chart

Each packet starts with a break in which the software is held low for at least 88us. Following the break, the Mark After Break (MAB) occurs in which the communications line will be held high for two bits. The protocol will then cycle through 513 8-bit data frames. The first frame is reserved as the start code in which the data will be 0. Following the start code frame, a frame is sent which contains the data of channel 1. The next frame is channel 2 and so on. On the receiver end, this means that a slave will receive the data for all channels, but only needs to react on the channel which corresponds to its address.

3 Implementation

A hardware interface and a Code Composer Studio project were developed to control the LLC resonant stage, control the LED stages, and receive DMX512 commands on a F28027 device. The driver software for DMX512 communications is included with this project and is found largely in the DMX512Comms.c file.

This project can be found at:
C:\TI\controlSUITE\development_kits\IsoACLighting_vX.X\IsoACLighting-F28027-DMX512

3.1 Hardware interface

A DMX512 hardware interface has been built into the TMDSIACLEDCOMKIT development kit. See Figure 5 for the schematic of this interface. Net terminals DMX-TX and DMX-RX will attach to the SCI peripheral of the F28027 MCU. Note that below TX and RX are relative to the fixture controlled by the Piccolo F28027.

The host will be connected into J8 and will then be translated into a standard serial signal by a Texas Instruments SN65HVD3082ED transceiver.

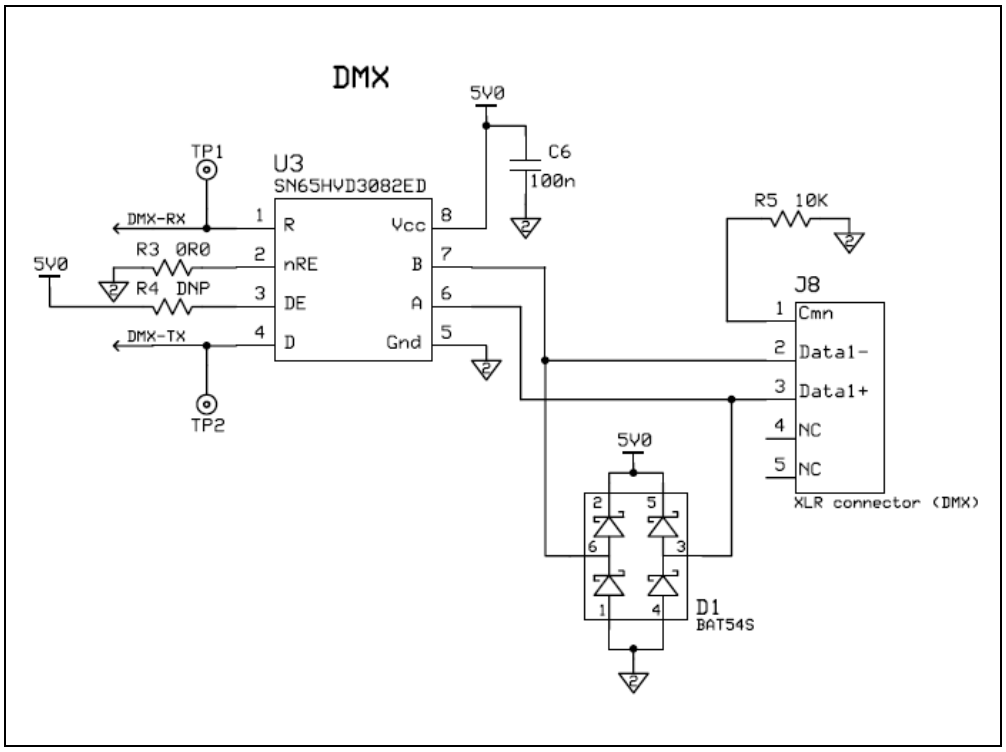


Figure 5: DMX512 interface schematic

3.2 DMX512 Software Driver

The DMX512 software driver relies on the C2000's Serial Communication Interface (SCI) peripheral to gather the data and store it in a buffer. This communications peripheral is configurable and can be edited to resemble the DMX512 protocol. In the project, the SCI is configured to expect 1 stop bit, no parity, and 8 character bits.

The data rate in the DMX512 protocol pushes the SCI to, for it, a relatively high 512kbaud. In order to effectively receive the DMX512 signal, the speed of the low speed clock (LOSPCP), which controls the SCI clock speed, was set to 30MHz and the SCILBAUD bits were set to 14.

The SCI translates the DMX512 data into words with no problem most of the time, however the BREAK time specified in the DMX512 protocol does cause some issues. This is because the SCI peripheral does not expect the protocol to go high for a significant amount of time and therefore sees the BREAK as an error.

In order to workaround this issue, the software driver switches the pins of the C2000 device between GPIO functionality and SCI functionality.

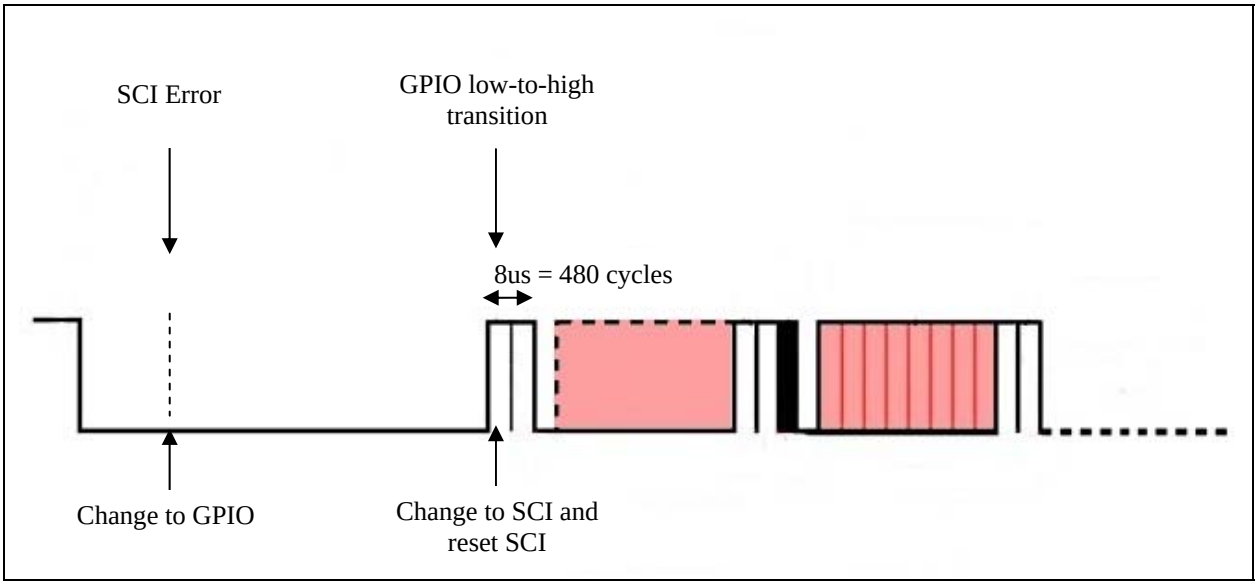


Figure 6: DMX512 Software Timing

After a successful transmission, the DMX512 line goes to IDLE for some amount of time with no issue. After this the DMX512 line enters into the BREAK. Once the SCI has been held low for at least ten bits, the SCI's break detect (BRKDT) error flag is set and this will halt the SCI communication. The automatic setting of the flag will generate a receive interrupt. After determining that the interrupt was caused by an error, the driver configures the GPIO28 pin to become a GPIO and configures an external GPIO interrupt to occur the next time that the GPIO senses a low-to-high transition.

Once the BREAK finishes, the MAB occurs for 2 bits. At the low-to-high transition the external GPIO interrupt will fire. Inside this interrupt, GPIO28 is reconfigured to be an SCI pin and the SCI peripheral is reset. By the time the start bit for the Start Code arrives, the SCI will be reset and the SCI will receive all the new data.

Because the C2000 CPU may be doing other interrupts, as is the case in the IsoACLighting-F28027-DMX512 project and the way that the DMX512 driver works, the cycle usage of the CPU around the MAB is somewhat critical. The MAB is 2 DMX512 bits, which corresponds to 8us or 480 F28027 clock cycles.

Therefore we must guarantee that the external interrupt and any other high priority interrupts (which may already be running when the external interrupt occurs) can be finished in 480 cycles.

In the IsoACLighting-F28027-DMX512 project, the non-DMX512 ISRs take up a total of 350 cycles. The external interrupt takes about 8 cycles. The ISR-to-entry point delays must also be added in which for the C2000 device are 14-18 cycles. A chart below summarizes the worst case cycle usage during the MAB.

	ISR cycles	ISR plus ISR trigger to entry point delays
Control Loop ISR	230	248
Resonant PWM Update ISR x 3	120	174
GPIO28 L-to-H External Interrupt	8	26
TOTAL	**358**	**448**
Spare ISR cycles	122	32

Table 2: Worst case ISR Timings during DMX512 MAB

In this chart, the Resonant PWM Update interrupt is set to happen three times which is the worst case. The amount of cycles going toward the Resonant PWM update will depend on the load on the resonant power stage.

Note that even though there are only 32 spare ISR cycles, this is not equivalent to the number of spare CPU cycles. To perhaps better state things, there are 32 spare cycles during the MAB until it is possible that occasionally the driver may miss part of a DMX512 packet and therefore won't update during that DMX512 period. Outside of the MAB time there will be a significantly larger amount of spare cycles to do host control, etc.

The DMX512 standard only describes the packet and limitations on the hardware interface. An implementation of commands is not strictly given, however, often a packet of data relates to an address's brightness. For example, a RGB light with addresses 56-58 may set address 56 to be the brightness of red, address 57 to be the brightness of green and address 58 to be the brightness of blue. The driver grabs the DMX512 data from all addresses and places it in one large buffer of memory. Once one buffer is filled a second buffer is filled and this continues in a ping-pong fashion so that there is always a known good buffer. The API function shown below returns a passed address's most recent data.

```
Uint16 getValueOfDmxAddress(Uint16 address)
```

3.3 DMX512 Project

Before working with this project it will be useful to have already run and experimented with the IsoACLighting-F28027 project found at:

C:\TI\controlSUITE\development_kits\TMDSIACLEDCOMKIT_v1.0\IsoACLighting-F28027

That project and its documentation thoroughly explain the TMDSIACLEDCOMKIT and how the F28027 controls the kit's power stages. The IsoACLighting-F28027-DMX512 project builds on the knowledge found there.

Hardware-wise the only change that needs to be done to make the DMX512 project work is that the jumpers at [Main]-J6 and [Main]-J7 both need to be moved and placed to connect positions 2 and 3.

The key software difference between the IsoACLighting-F28027 project and the IsoACLighting-F28027-DMX512 project is that the IsoACLighting-Comms.c file has changed and now interfaces and calls

functions in the DMX512Comms.c file. The Comms.c file also initializes and turns on the Resonant stage so that the DMX512 interface can directly change the brightness of the LED strings without needing to do anything with the LLC resonant stage. This DMX512Comms.c file contains the DMX512 driver and the API into the driver. Figure 7 gives some details on the major files used in the DMX512 project.

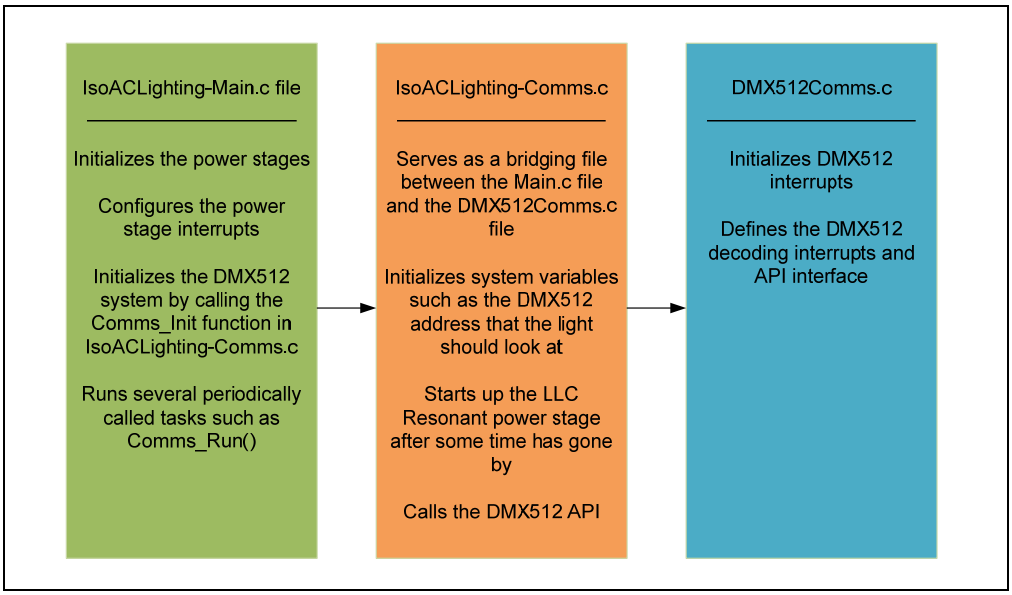


Figure 7: Major Files in the DMX512 Project

To run the project and have DMX512 communications work you will need a DMX512 master such as the Enttec USB Pro. For a computer to interface with this master, you will require some software such as Light Factory (http://www.lifact.com/). You will also need a DMX compatible cable such as Newark part number 18J1683.

Steps:

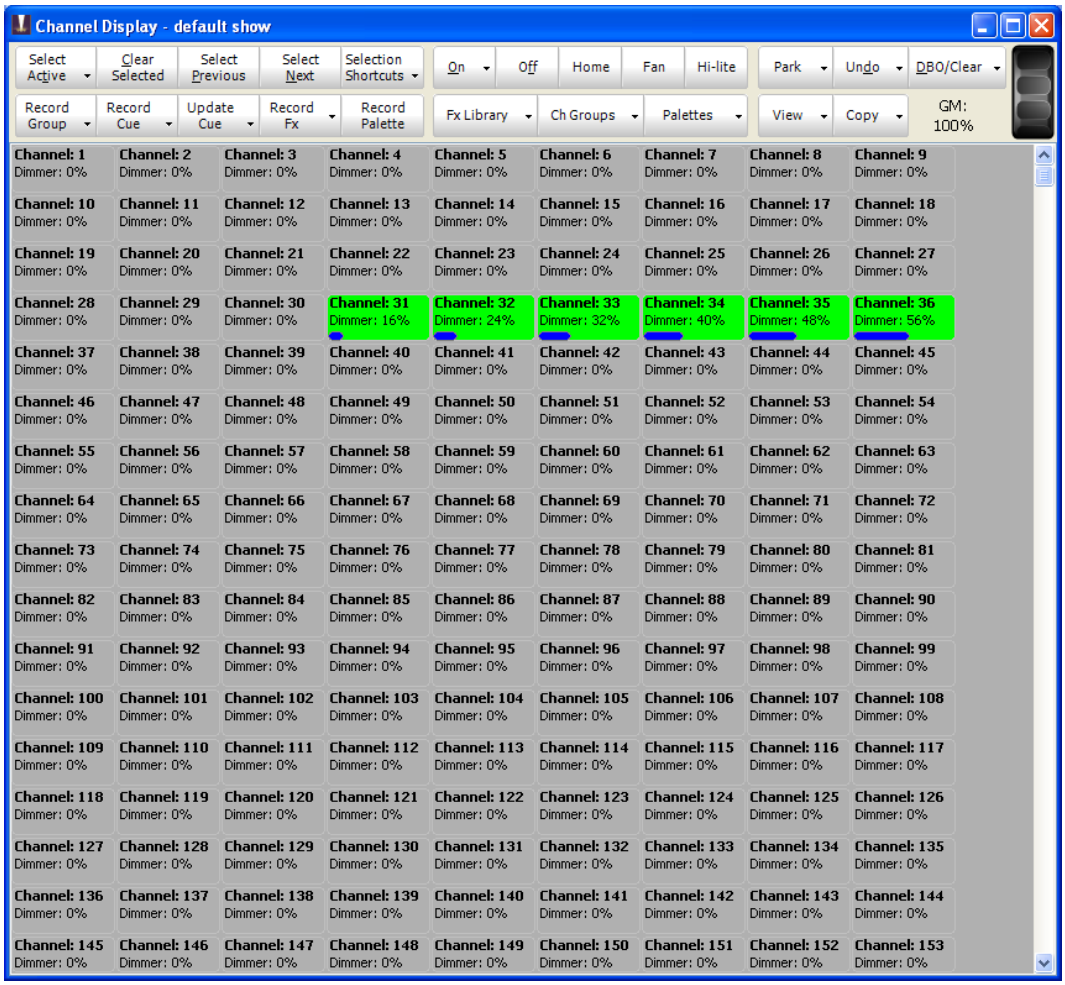
1. Run the software and configure the hardware as instructed in build 1 & 2 of the TMDSIACLEDCOMKIT-CCS guide.
2. Attach the USB cable to [M8]-JP1
3. Ensure that [Main]-J3 is populated
4. Put jumpers across positions 2 & 3 of [Main]-J6 and [Main]-J7
5. Attach a DMX512 cable to [Main]-J8
6. Setup the DMX512 master and software as instructed in the master's user manuals and guides.
7. Connect a 12V DC supply to [M7]-JP1
8. Switch [M7]-SW1 to the "Ext" position.
9. Open CCS
10. Import the IsoACLighting-F28027-DMX512 project found at:
    C:\TI\controlSUITE\development_kits\TMDSIACLEDCOMKIT_vX.X\
    IsoACLighting-F28027-DMX512
11. Compile the software
12. Connect and flash the code by clicking the "bug" button.
13. In CCS, reset the processor and then click restart (so that the program starts at the beginning of main)
14. In CCS, enable real-time mode and enable continuous refresh of the watch window.
15. Click the green play button to run the software. (Step 16 should be done shortly after this step is complete.)

16. Carefully plug the AC cable into the wall. **Once this step is complete, the board will be dangerous to touch.**

Note that the LLC resonant power supply is automatically set to turn on about 10 seconds after the board is powered.

17. Use the DMX512 software to communicate brightness information to the kit. By default the DMX address each LED string is looking at is address 31-36. The DMX512 commands to these address will impact Gui_Iled[0-5] and the current reference given to each LED string.
18. From CCS, you can see look at rdata[dmxBufferNum_LastGood] to see the last good buffer of all 512 pieces of DMX512 data
19. Once done, set each applicable DMX512 address to 0.
20. Unplug the power cable from the wall.
21. Wait at least 60 seconds before touching the board.

Note: This demo can also be run without the external DC/DC power supply connected and without CCS. However, it is safer to use the external power supply first. Once the DMX512 code is flashed into the device (which is done in the procedure above), you can rerun the above with the following exceptions: remove the DC/DC supply, switch [M7]-SW1 away from "Ext", and skip steps 10-12 & 16.



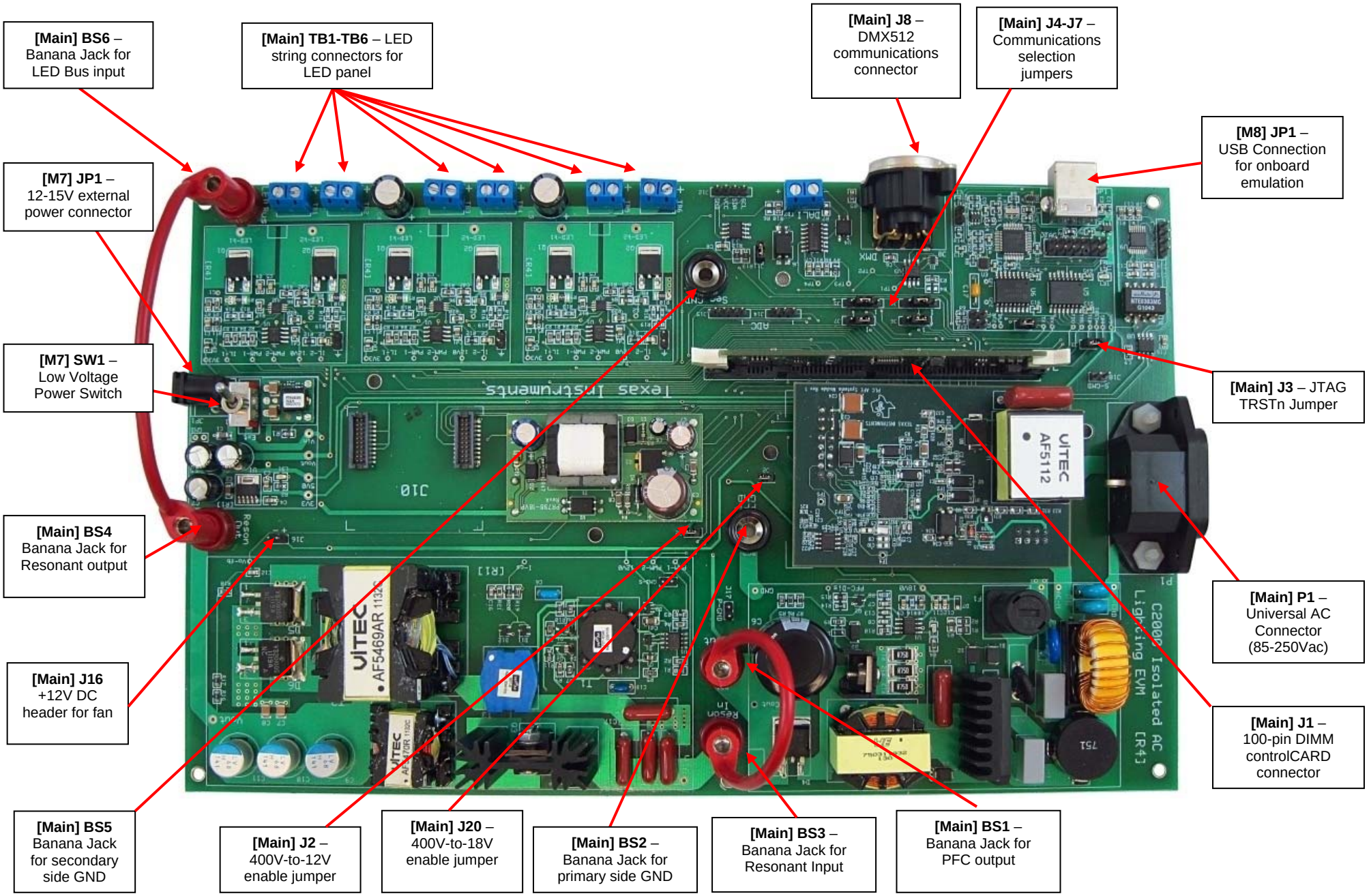


Fig5. AC LED Lighting and Communications Board Jumpers and Connectors Diagram

References

For more information please see the following guides:

- **TMDSIACLEDCOMKIT_CCS –** provides detailed information on the IsoACLighting project within Code Composer Studio. The document goes through the project in an easy to use lab-style format.

  C:\TI\controlSUITE\development_kits\TMDSIACLEDCOMKIT_vX.X\~Docs\ TMDSIACLEDCOMKIT_CCS.pdf

- **TMDSIACLEDCOMKIT-HWdevPkg** – a folder containing various files related to the hardware on the AC LED Lighting and Communications Developer's Kit board (schematics, bill of materials, Gerber files, PCB layout, etc).

  C:\TI\controlSUITE\development_kits\TMDSIACLEDCOMKIT_vX.X \ ~TMDSIACLEDCOMKIT-HwdevPkg[R4]

- **TMDSIACLEDCOMKIT-HWGuide** – presents full documentation on the hardware found on the AC LED Lighting and Communications Developer's board.

  C:\TI\controlSUITE\development_kits\TMDSIACLEDCOMKIT_vX.X\~Docs\ TMDSIACLEDCOMKIT -HWGuide.pdf

- **DMX512 Standard** – Gives links to the DMX512 and DMX512-A standard created by USITT and maintained by ESTA.

  http://www.usitt.org/Resources/Standards2/DMX512

- **OpenDMX.net** – Presents some useful information on DMX512.

  http://www.opendmx.net/index.php/DMX512-A
  http://www.opendmx.net/index.php/OpenDMX.net

- **Ujjal's DMX512 Pages** – provides a good overview of DMX512, implementation and history.

  http://www.dmx512-online.com/whats.html