LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.
**XC82x/XC83x**

**Revision History: V1.1 2012-10**

Previous Version(s):

<table>
<thead>
<tr>
<th>Page</th>
<th>Subjects (major changes since last revision)</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>Figure 26. R5 changed from 1k to 560R</td>
</tr>
<tr>
<td>28</td>
<td>Figure 28. Updated schematic version number from 1.4 to 1.5</td>
</tr>
<tr>
<td>-</td>
<td>Updated all Figures with DALI-DMX512 Board version 1.5</td>
</tr>
</tbody>
</table>

---

**We Listen to Your Comments**  
Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:  
**mcdocu.comments@infineon.com**
# Table of Contents

1 Introduction .................................................................................................................. 5

2 Overview of DMX512 ...................................................................................................... 5
  2.1 System Overview ........................................................................................................ 5
  2.2 Physical Layer ............................................................................................................ 6
    2.2.1 XLR-5 Connector ................................................................................................. 6
    2.2.2 Isolated Topology ................................................................................................. 7
  2.3 DMX512 Protocol ........................................................................................................ 7

3 DMX512 Implementation with XC836 ............................................................................ 9
  3.1 Hardware .................................................................................................................... 9
    3.1.1 DMX512 Connectors ......................................................................................... 10
    3.1.2 DIP Switch ......................................................................................................... 10
    3.1.3 Power Supply Connector and SPD Connector ................................................... 11
  3.2 Software .................................................................................................................... 12
    3.2.1 Software Abstraction Layers ............................................................................. 12
    3.2.2 Interrupt Timing Diagram .................................................................................. 13
    3.2.3 List of Source Code Files ................................................................................... 13
    3.2.4 Recommended DMX512 Signal Characteristics ................................................. 14

4 DMX512 Software Stack Configuration .......................................................................... 15
  4.1 Configuring the Required DMX512 Slots ................................................................. 15
  4.2 Alternative Pinouts .................................................................................................... 15
  4.3 DMX512 Address Setting by Software ..................................................................... 16

5 Evaluating DALI - DMX512 Board for LED Color Control Application ....................... 17
  5.1 Connecting the Boards in a Daisy-Chain .................................................................... 17
  5.2 Setting the DMX512 Address with DIP switches ..................................................... 17
  5.3 Connecting the Transmitting Device to the Daisy Chain .......................................... 18
  5.4 Powering Up the Receiving Devices ....................................................................... 18
  5.5 Powering Up the Transmitting Device from USB ..................................................... 19
  5.6 Controlling LED Color with the Transmitting Device .............................................. 19

6 Enhancement Features .................................................................................................. 22
  6.1 Packet Length Check ............................................................................................... 22
  6.2 Valid Address Check ............................................................................................... 23
  6.3 Packet Loss Handling ............................................................................................. 24

7 Summary ....................................................................................................................... 24

8 References ...................................................................................................................... 25

Appendix - DALI-DMX512 Board Schematic .................................................................. 26
Appendix - DMX512 Software Stack Flowchart ................................................................ 29
Appendix - Suggested Circuit to Fulfill Isolated Topology Requirement .......................... 31
1 Introduction

DMX512 is a communication protocol commonly used in stage lighting applications. It describes the digital data transmission between the controller and the stage equipment, such as a washlight, moving head, or fog machine for example. The E1.11-2008 USITT DMX512-A protocol is maintained by ESTA (Entertainment Service and Technology Association).

This application note describes the implementation of a DMX512 software stack to act as a receiving device, using the XC836 microcontroller from Infineon. We start with an overview of the DMX512, where the system architecture and protocol structure are explained, then discuss the implementation using the XC836 microcontroller, and finally describe the integrated DALI-DMX512 board which can be used to control the on-board RGB LED via the DMX512 protocol.

2 Overview of DMX512

DMX512 is a packet-based, asynchronous, serial and unidirectional communication protocol. Because there is no error checking or correction mechanism specified in the standard, this makes it relatively simple, but also means that it is unsuitable for safety-critical applications.

DMX512 uses differential signals for communication specified by the RS-485 standard. Therefore, it has good immunity against noise and is able to communicate at relatively long distance (up to 1200 meters). However, it also inherits the limitation of RS-485 which allows only a maximum of 32 devices to be connected to the same communication line.

2.1 System Overview

A DMX512 system consists of one master device (transmitting device) connected to multiple slave devices (receiving devices) in ‘daisy-chain’ manner as illustrated in Figure 1 below.

Figure 1 DMX512 System connected in Daisy Chain

A unique address must be assigned to each slave device by configuring the DIP switch embedded on each device. The address may range from 1 to 512, depending on how many devices are connected and how many DMX512 slots/channels are consumed by each device. For example, an RGB-LED wallwasher may consume one DMX512 slot for each color. If its address is set at 10, it will consume slot 10, 11 and 12. The address for the next device must be set to 13.

A termination resistance typically of 120Ω is connected at the furthest slave device to prevent signal reflection. In the case where more than 32 devices are required in a system, an in-line device, such as optosplitter, can be used.
2.2 Physical Layer

The standard specifies the physical layer of DMX512 to consist of the connector configuration and the circuit topology. DMX512 system uses XLR-5 connectors, as shown in Figure 2. Transmitting device uses Ground Referenced Topology while receiving device uses Isolated Topology. In this document, only the physical layer of the slave/receiving device is discussed.

2.2.1 XLR-5 Connector

Each receiving device has both a male and a female XLR-5 connector that is used for receiving and transmitting the DMX512 signal, respectively. This is to create the daisy-chain topology that ensures the continuity of communication from the transmitting device to the last receiving device.

The following Table 1 specifies the connections for the receiving device:

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Signal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Common Reference</td>
<td>Data Link Common</td>
</tr>
<tr>
<td>2</td>
<td>Data 1+</td>
<td>Primary Data Link</td>
</tr>
<tr>
<td>3</td>
<td>Data 1-</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Data 2+</td>
<td>Secondary Data Link (Optional)</td>
</tr>
<tr>
<td>5</td>
<td>Data 2-</td>
<td></td>
</tr>
</tbody>
</table>

In a typical DMX512 application, only Primary Data Link (DATA1+ and DATA1-) and Common Reference are used.

The Secondary Data Link is not used and is reserved for future use. It is therefore common to find some lighting fixtures that use XLR-3 connectors.
2.2.2 Isolated Topology

The standard specifies the use of isolated topology in the receiving device circuit, as illustrated in Figure 3 below. This topology is not implemented in our solution, however users are encouraged to follow this topology strictly when implementing their own solution. Refer to Figure 32 in the APPENDIX section for suggested circuit that fulfill isolated topology.

Figure 3 Isolated Topology as specified in the standard

2.3 DMX512 Protocol

As the name suggests, there are 512 “pieces of information” carried in a DMX512 packet. Each “piece of information” is also known as a slot or channel, which consists of 1 start bit, 8 data bits and 2 stop bits. A Reset Sequence consisting BREAK, MAB and NULL Start Code must be transmitted before the slots. Figure 4 illustrates a DMX512 packet.

Figure 4 DMX512 Protocol
The timing requirements for the receiving device are shown in Table 2.

### Table 2: Timing Requirements for Receiving Device

<table>
<thead>
<tr>
<th>Signal</th>
<th>Min. Value</th>
<th>Typ. Value</th>
<th>Max. Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Rate</td>
<td>245 kbps</td>
<td>250 kbps</td>
<td>255 kbps</td>
<td>Transmission rate for DMX512 protocol.</td>
</tr>
<tr>
<td>Bit Time</td>
<td>3.92 µs</td>
<td>4 µs</td>
<td>4.08 µs</td>
<td></td>
</tr>
<tr>
<td>BREAK</td>
<td>88 µs</td>
<td>176 µs</td>
<td>-</td>
<td>A falling edge transition followed by a low of at least 88 µs followed by a rising edge.</td>
</tr>
<tr>
<td>MAB</td>
<td>8 µs</td>
<td>-</td>
<td>&lt; 1 s</td>
<td>Mark After Break - The period of time measured from the rising edge at the end of BREAK to the falling edge of the start bit of the START Code.</td>
</tr>
<tr>
<td>MTBS</td>
<td>0</td>
<td>-</td>
<td>&lt; 1 s</td>
<td>Mark Time Between Slot - The period measured from the end of the second stop bit (bit 9) of the previous slot to the falling edge of the start bit of the current slot.</td>
</tr>
<tr>
<td>MBB</td>
<td>0</td>
<td>-</td>
<td>&lt; 1 s</td>
<td>Mark Before Break - The period measured from the end of the second stop-bit of the last slot to the falling edge of the next BREAK.</td>
</tr>
<tr>
<td>BREAK-TO-BREAK</td>
<td>1196 µs</td>
<td>-</td>
<td>1.25 s</td>
<td>The period between two BREAKs</td>
</tr>
</tbody>
</table>
3 DMX512 Implementation with XC836

Current solutions in the market use either a 16-bit or 32-bit microcontroller with a large memory size to implement DMX512. The microcontroller receives and stores all the slots before processing them. This method is inefficient as it requires 512 bytes of data memory while only a few are relevant to the receiving device. In addition, some implementations use two or more pins because each IO pin can only support one function.

The implemented software stack described in this Application Note will selectively receive and store relevant slots. This method reduces the required memory for the DMX512 software stack and gives more space for application-specific code. In addition, it will also be implemented on a single pin. This is possible because XC800 devices are able to map multiple functions into a single IO pin.

The software stack will be implemented using Timer 2, UART and Timer 0. It will occupy 1.2KB of Flash and a few bytes of RAM, depending on the number of required slots in the application.

Timer 2 and Timer 0 are 16-bit general purpose timers which are functionally compatible with the C501 product family. Timer 2 has Capture Mode for pulse width measurement, which is useful to measure the width of BREAK and MAB, while Timer 0 can be used to measure the BREAK-to-BREAK signal.

UART is an integrated communication peripheral that can be set as a 9-bit serial port to receive and validate the START Code and the following slots.

3.1 Hardware

Figure 5 shows the DMX512 solution from Infineon. Aside from DMX512, the DALI (Digitally Addressable Lighting Protocol) is also implemented on the board. This enables the evaluation of both lighting protocols on a single platform by simply downloading a different protocol stack via the programming connector. The previous protocol stack will be overwritten by the new one and the relevant circuitry will be activated after downloading.

The connectors found on the board are the DMX512, DIP switches, power supply connector and SPD (programming) connector.
3.1.1 DMX512 Connectors

The DMX512 connector is shown in Figure 6 below. The differential DMX512 signal is received and converted to TTL level by RS485 transceiver, where its Receive Output (RO) pin and Receive Enable (RE) pin are connected to P2.7 and P0.6 of XC836, respectively.

In addition, there is a 120Ω termination resistance that must be enabled at the end of the daisy-chain connection to avoid signal reflection. Users can enable this termination resistance by shorting the on-board jumper.

![Termination Resistance](image)

**Figure 6** DMX512 Connector

3.1.2 DIP Switch

Each receiving device has embedded DIP switches to allow the user to assign a DMX512 address. In a typical DMX512 application, 9-bit DIP switches are used. This allows an address range from 1 to 512. The user can set the DIP switches to assign DMX Address 1 to 511, while address 512 is assigned by setting the DIP switches to zero.

As a demonstration only board, our solution only uses a 4-bit DIP switch, where the supported address will range from 1 to 16. Address 1 to 15 is achieved by setting the DIP switch and Address 16 is achieved by setting the DIP switch to zero.

When an address higher than 16 is required, the user is able to set it by software, which will be explained in Chapter 4.3.

![DIP Switch](image)

**Figure 7** 4-bit on-board DIP switch
3.1.3 Power Supply Connector and SPD Connector

5V DC is required at the power supply connector to power up the board. The programming connector (SPD) allows the user to download the application code and the software stack using miniWiggler.

![Power Supply Connector Diagram]

**Figure 8** Power Supply Connector
3.2 Software
The protocol stack utilizes three peripherals from XC836, namely Timer 2, UART and Timer 0 to receive and process the DMX512 signal. It will validate and pass only the relevant slots to the application code.
This implementation demonstrates an RGB LED color control application where the protocol stack processes and passes three slots of information to the application code. Another peripheral called CCU6 (Capture/Compare Unit 6), a PWM generator module, is also used to control the intensity of each LED color channel.

3.2.1 Software Abstraction Layers
Figure 9 shows the software abstraction layers which illustrate the processing of the DMX512 signal.

---

**Figure 9** Software Abstraction Layer

When the DMX512 signal is received, its BREAK and MAB will be detected and verified by Timer 2. If the BREAK and MAB fulfills the timing requirements, UART will then be activated to receive the START code and the slots. The slots with an index number that matches the assigned DMX512 address will be stored temporarily.
BREAK-to-BREAK time is measured and validated using Timer 0. The temporary slot values are passed on to the application code only if the packet's BREAK-to-BREAK time fulfills the required timing.
### 3.2.2 Interrupt Timing Diagram

This process is summarized in Figure 10, the interrupt timing diagram.

![Interrupt Timing Diagram](image)

The verified slots will be passed on to the application code, which updates the CCU6 duty cycle registers with the verified slots at a regular interval.

### 3.2.3 List of Source Code Files

The source code files used in the software stack are summarized in the following Table 3.

<table>
<thead>
<tr>
<th>Code Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMX512_CONFIG.H</td>
<td>Contains Software Stack Defines and Configuration, e.g. DMX512 pin, RS485 Receive Enable pin, number of required slots, etc.</td>
</tr>
<tr>
<td>MAIN.C</td>
<td>Contains hardware peripherals initialization, Valid Address Check code and sample of application code.</td>
</tr>
<tr>
<td>T2.C</td>
<td>Timer 2 initialization and Timer 2 External Interrupt subroutine (EXF2 flag) to detect BREAK and MAB.</td>
</tr>
<tr>
<td>UART.C</td>
<td>UART Initialization, the interrupt subroutine (RI flag) to detect START code and slots, reset sequence and Packet Length Check code.</td>
</tr>
<tr>
<td>T01.C</td>
<td>Timer 0 initialization and ISR to check for valid BREAK to BREAK time and Packet Loss Handling code.</td>
</tr>
<tr>
<td>IO.C</td>
<td>GPIO initialization for the software stack. It follows the assigned DMX512 pin in the DMX512_CONFIG.H</td>
</tr>
<tr>
<td>CC6.C</td>
<td>CCU6 peripheral initialization.</td>
</tr>
</tbody>
</table>
3.2.4 **Recommended DMX512 Signal Characteristics**

The following Table 4 describes the recommended signal characteristics of the incoming DMX512 packet for the implemented software stack.

**Table 4  Recommended DMX512 Signal Characteristics**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Recommended Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Rate</td>
<td>250 kbps</td>
</tr>
<tr>
<td>Bit Time</td>
<td>4 $\mu$s</td>
</tr>
<tr>
<td>BREAK</td>
<td>&gt; 92 $\mu$s</td>
</tr>
<tr>
<td>MAB</td>
<td>&gt; 12 $\mu$s</td>
</tr>
<tr>
<td>MTBS</td>
<td>&gt; 8 $\mu$s</td>
</tr>
<tr>
<td>MBB</td>
<td>&gt; 13 $\mu$s</td>
</tr>
<tr>
<td>BREAK-to-BREAK</td>
<td>&gt; 1204 $\mu$s</td>
</tr>
</tbody>
</table>
4 DMX512 Software Stack Configuration

This chapter describes the configuration of the implemented software stack. All the settings can be found in DMX512_Config.h

4.1 Configuring the Required DMX512 Slots

For an RGB color control application, there are at least three required DMX512 slots. Some other applications may require a different number of slots. When this software stack is used for such applications, it can be configured to receive a different number of slots, as shown in Figure 11 below.

![Figure 11 Configuring Required DMX512 Slots](image)

4.2 Alternative Pinouts

The implemented DMX512 software stack is designed to be portable across the XC800 microcontroller family and offers alternative pinouts, as shown in Table 5 below.

<table>
<thead>
<tr>
<th>Device</th>
<th>Pin #</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC82x</td>
<td>P1.0</td>
</tr>
<tr>
<td>XC835</td>
<td>P1.0, P3.2</td>
</tr>
<tr>
<td>XC836</td>
<td>P1.0, P3.2, P2.7</td>
</tr>
</tbody>
</table>

The DMX512 pinout can be changed in DMX512_Config.h by typing the desired pinout after #define DMX_AT_P2_7 to another pinout.
The following Figure 12 shows an example where the user changes the DMX512 pin to P3.2.

Figure 12 DMX512 Software Stack Configuration in DMX512_Config.h

When using P3.2 as the DMX512 pin, the SPD pin must be assigned to another pin. Refer to AP08108 for programming the BMI value in the XC82x and XC83x devices.

4.3 DMX512 Address Setting by Software

The implemented DALI-DMX512 board supports an address range from 1 to 16. The address of the receiving device can be set by software, by commenting the #define DIPSWITCH_ENABLE in DMX512_Config.h, then uncommenting the #define SW_DMX_ADDRESS and adding the desired address, as shown in the Figure 13:

Figure 13 DMX512 Address Setting by Software
5 Evaluating DALI - DMX512 Board for LED Color Control Application

The DALI-DMX512 Board from Infineon is a receiving or slave device that contains the two common lighting protocols, DALI and DMX512. The board is designed to demonstrate an LED Color Control application on both protocols. This chapter is intended as a step-by-step guide for users to evaluate this board.

5.1 Connecting the Boards in a Daisy-Chain

A simple twisted pair of shielded wires can be used to connect the boards. The diagram inserted in the top-left of the following Figure 14 shows the wiring connection.

![Figure 14](image)

**Figure 14** Two Boards connected in Daisy Chain

5.2 Setting the DMX512 Address with DIP switches

In a typical DMX512 device, the user can set the DMX512 address using DIP switch, where DIP switch #4 refers to the Least Significant Bit (LSB) of the address. The following Figure 15 shows an example where the board is set to Address 11.

![Figure 15](image)

**Figure 15** Setting DMX512 Address to 4
5.3 Connecting the Transmitting Device to the Daisy Chain

Once the address of each board is set, the transmitting device can be connected as shown in Figure 16.

5.4 Powering Up the Receiving Devices

Figure 17 shows the receiving devices being powered up from 5V power supply with on-board LED turned on.
5.5  Powering Up the Transmitting Device from USB

The transmitting device requires 5V DC and is powered from USB, as shown in Figure 18.

![Figure 18](image)

Figure 18  Powering the Transmitting Device via USB

5.6  Controlling LED Color with the Transmitting Device

Figure 19 and Figure 20 show the touch pads and the 7-segment LED display functions on the transmitting device.

![Figure 19](image)

Figure 19  XC836 Easy Kit Touch Pads
The LED to indicate transmission will only light up when there is a DMX512 signal transmission. The following Table 6 describes the functionality of each touch pad.

<table>
<thead>
<tr>
<th>Touchpad Name</th>
<th>Value Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Address</td>
<td>0 - 24 [DEC]</td>
<td>Go to next/previous slot address. For demonstration purposes, only 24 slots</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are supported by the transmitting device.</td>
</tr>
<tr>
<td>Prev Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incr Slot Value</td>
<td>0 - FF [HEX]</td>
<td>Increase/Decrease current slot values. The value is represented as hexadecimal.</td>
</tr>
<tr>
<td>Decr Slot Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fine/Coarse</td>
<td>-</td>
<td>Adjust the resolution of Incr/Decr Slot Value. Selecting Coarse Mode will</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modify the 2nd digit of Slot Value while Fine Mode will modify the 1st digit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the Slot Value. Default mode is Coarse Mode.</td>
</tr>
<tr>
<td>START/STOP DMX512</td>
<td>-</td>
<td>Start/Stop DMX512 Transmission</td>
</tr>
<tr>
<td>Next Scene</td>
<td>0 - 99 [DEC]</td>
<td>Go to the next/previous predefined scene.</td>
</tr>
<tr>
<td>Prev Scene</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Scene</td>
<td>-</td>
<td>Set the selected scene to the receiving devices. Pressing this button will</td>
</tr>
<tr>
<td></td>
<td></td>
<td>automatically enable the DMX512 transmission.</td>
</tr>
</tbody>
</table>
The following Figure 21 shows the complete setup of DMX512 system for RGB-LED color control application.

Figure 21  Complete Setup of DMX512 System

The user can now control the LED color on each board from the transmitting device. For example, when the slot value of address 04 is increased, the second board will display brighter red, while increasing the slot value of address 08 will make the third board display brighter green. For more information on the transmitting device, please refer to Application Note AP08132.
6 Enhancement Features

The following features are implemented to enhance the reliability of the software stack while still complying with the DMX512 standard.

6.1 Packet Length Check

The standard never specifies the minimum number of slots that must be transmitted, therefore it is common to find some devices transmitting less than 512 slots. An error may be introduced to the receiving devices when the sum of its assigned address and the number of required slots exceed the packet length. For example, a receiving device with the address of 29 which requires 3 slots may have an error if the packet sent by the transmitting device has only 30 slots. The receiving device will not be able to obtain the data for its third slot (slot 31).

This feature is implemented to make the software stack able to recognize the packet length of incoming DMX512 and adjust itself accordingly. Users do not have to worry about the packet length sent by the transmitting device. Using the example above, the receiving device will turn itself off when it is unable to receive slot 31.

This feature works by sampling the packet length of the first few incoming packets. It will assign the maximum value from the samples as the new packet length. By default, only the first four packets will be sampled by the software stack. During this check, no data will be passed on to the application code. The subsequent packets having a shorter or longer length will be considered as an error and will be discarded.

User can increase the number of packets to be checked by changing the PACKET_TO_BE_CHECKED in DMX512_Config.h to more than four. Setting the value to 0 will make the software stack take the length of the first received packet as the default packet length.

When this feature is disabled, the default packet length will be assumed as 513, i.e. START Code + 512 Slots. Users, however, can define their own packet length by changing uwPacketLength value in MAIN.C, as shown in Figure 23.

---

![Packet Length Check Flowchart and its Code Snippet](image-url)

---

Figure 22 Packet Length Check Flowchart and its Code Snippet
6.2 Valid Address Check

This feature is implemented to complement the Packet Length Check. After the packet length is known, it will check whether the sum of DMX512 address and the required slots exceed the packet length. For example, when a receiving device address is set to 30 while the received packet length is only 24, this will be recognized as an error, and all hardware peripherals (Timer 2, UART and Timer 0) and the RS485 transceiver will be switched off.

In addition, when using 9-bit DIP switches as the mean for DMX addressing, the maximum achievable address is only 511. On the other hand, when the DIP switches are not set, the receiving address will automatically be assigned to zero, the NULL START Code, which has no practical value for the receiving device. This feature will automatically set the address to 512 when the DIP switches are not set.
6.3 Packet Loss Handling

Packet loss may occur for several reasons, such as power failure at the transmitting device or because of a disconnected communication cable. When there is no DMX512 signal received after 9.6 secs, the DMX512 software stack is reset and waits for the incoming DMX512 signal while still retaining the data before the packet loss occurred. The Packet Length Check feature will also reset to relearn the packet length.

![Packet Loss Handling Flowchart and its Code Snippet]

```
// Enhancement Features

// Uncomment this for Packet Length Check
#define PACKET_LENGTH_CHECK

#ifndef PACKET_LENGTH_CHECK
// Number of DMX512 packets to be checked for Packet Length Check
#define PACKETS_TO_BE_CHECKED 4
#endif

// Uncomment this for Valid Address Check
#define VALID_ADDRESS_CHECK

// Uncomment this for Packet Loss Handling
#define PACKET_LOSS_HANDLING

#endif
```

Figure 25 Packet Loss Handling Flowchart and its Code Snippet

7 Summary

This application note has described the DMX512 solution from Infineon implemented on the XC836, an 8051-based microcontroller. The DMX512 software stack is evaluated on an integrated DALI-DMX512 board, to also allow evaluation of the DALI protocol.

Unlike other implementations in the market, the software stack selectively receives and stores only relevant slots. This reduces the required memory size thus allowing for more complex application code. In addition, it is also implemented on a single pin which will reduce the pinout requirements. The software stack is portable to other XC800 devices and it also offers alternative pinouts.

Lastly, some additional features such as valid address check and packet length check are also implemented to increase the reliability of the software stack while maintaining compliance with the standard.
8 References


[4] AP08108 “Programming the BMI Value in the XC82x and XC83x Products”


[6] AP08102 “DALI Control Gear Software Stack”


[8] AP08104 “Guide to using the DALI LightNet Tool”

[9] AP08105 “DALI Demo using Touch Sense Control”
Appendix - DALI-DMX512 Board Schematic

Figure 26 and Figure 27 show the schematic of DALI-DMX512 Board.
Figure 27  DALI-DMX512 Board Schematic - XC836 and RGB LED
Figure 28  DALI-DMX512 Board Layout
Appendix - DMX512 Software Stack Flowchart

The following Figure 29, Figure 30 and Figure 31 shows the flowchart from each peripheral that is used to create the DMX512 software stack.

Figure 29  Timer 2 Flowchart

Decision Points:
- **T2 External Interrupt occurs**
  - Clear Interrupt Flag
- **Rising edge of MAB is detected**
  - Get previous packet BREAK-to-BREAK measurement
- **Falling edge of slot start bit is detected**
  - Get MAB time from T2 register
  - Valid BREAK
  - Stop Timer 2 and Reset Timer 2 register
  - Set Timer 2 to start at falling edge
  - Disable Timer 0 and reload its register
  - Get DMX data from temporary buffer
  - Clear valid temp data bit
  - Proceed to receive slot
- **Valid MAB external interrupt**
  - Reset slot counter
  - Valid BREAK-to-BREAK and known packet length & valid temp data
  - Reset BREAK-to-BREAK count
  - Get DMX data from temporary buffer
  - Clear valid temp data bit
  - Proceed to receive slot
- **Return**

Branching Points:
- **Y**
  - Valid BREAK
  - Set next interrupt to detect falling edge of start-bit
  - Disable UART
  - Reset state
- **N**
  - Stop Timer 2
  - N
  - Y
  - Proceed to detect slot

- **Y**
  - Stop Timer 2
  - N
  - Y
  - Proceed to detect slot

- **N**
  - Disable UART Receiver
  - Reset state
UART Interrupt

Figure 30 UART Flowchart

Timer 0 Interrupt

Figure 31 Timer 0 Flowchart
Appendix - Suggested Circuit to Fulfill Isolated Topology Requirement

As mentioned in Section 2.2.2, the standard requires all receiving devices to be implemented using isolated topology. The following Figure 32 shows a suggested circuit using isolated topology for DMX512 Receiving Device.

Source: AN3776 Maxim Integrated

Figure 32  Suggested Isolated Circuit for DMX512 Receiving Device