# UM0792
# User manual

## Demonstration firmware for the DMX-512 communication protocol transmitter based on the STM32F103Zx

## Introduction

This document describes how to use the demonstration firmware for the DMX-512 communication protocol transmitter. The USART (universal synchronous asynchronous receiver transmitter) module of the STM32F103Zx (ARM 32-bit Cortex™-M3) microcontroller unit transmits the data via an RS-485 transceiver. This transmitter sends DMX-512 packets with a “NULL” start code as per the DMX-512 2008 standard.

This document provides:

- an overview of the complete solution.
- a description of the STM32 demonstration firmware.
- schematics and layouts.

# Contents

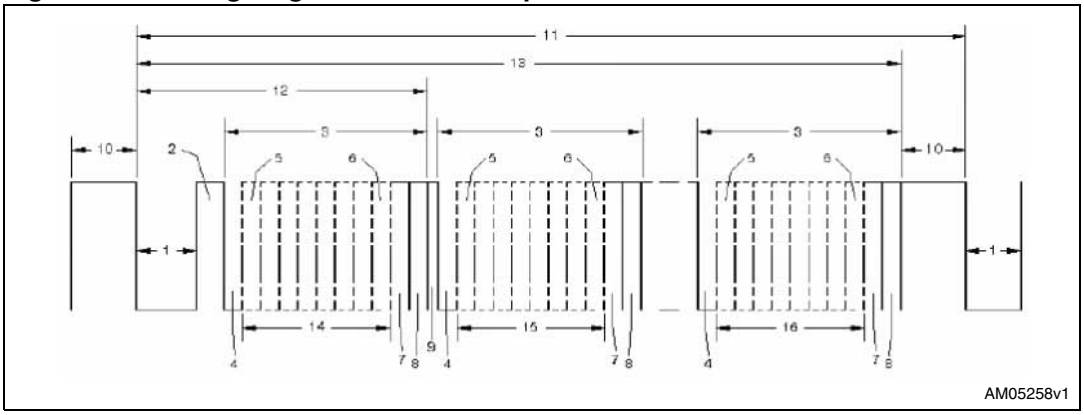
# List of figures

# 1 DMX-512 transmitter format

## 1.1 Packet format

The DMX-512 slots are transmitted sequentially in an asynchronous serial format, beginning with slot 0 and ending with the last implemented slot, up to slot 512 (a maximum total of 513 slots). Before the first data slot is transmitted, a reset sequence is transmitted, consisting of a BREAK, a MARK AFTER BREAK and a START code. Valid DMX-512 data slot values under a NULL START code are from 0 to 255 decimal.

**Figure 1. Timing diagram for DMX-512 packet**




The following points refer to the numbers shown in *Figure 1*.

1. SPACE for BREAK
2. MARK AFTER BREAK
3. Slot time
4. START time
5. LEAST SIGNIFICANT data bit
6. MOST SIGNIFICANT data bit
7. STOP bit
8. STOP bit
9. MARK TIME BETWEEN SLOTS
10. MARK BEFORE BREAK
11. BREAK to BREAK time
12. RESET sequence
13. DMX-512 packet
14. START CODE (slot 0, data)
15. SLOT 1, data
16. SLOT n, data (max 512)

## 1.2 Timing values for DMX-512 transmitter

The transmitter only accepts the data if all the timing values comply with those given in *Table 1*.

**Table 1. Timing values of DMX-512 packet transmitted**

| Description | Min | Typical | Max | Unit |
|---|---|---|---|---|
| Bit rate | 245 | 250 | 255 | Kbps |
| Bit time | 3.92 | 4 | 4.08 | µs |
| Minimum update time for 513 slots | - | 22.7 | - | ms |
| Maximum refresh rate for 513 slots | - | 44 | - | µs |
| "SPACE" for BRAEK | 92 | 176 | - | µs |
| "MARK" after BREAK (MAB) | 12 | - | <1.00 | µs<br>s |
| "MARK" time between slots | 0 | - | <1.00 | s |
| "MARK" before BREAK (MBB) | 0 | - | <1.00 | s |
| BREAK to BREAK time | 1204 | - | 1.00 | µs<br>s |
| DMX-512 packet | 1204 | - | 1.00 | µs<br>s |

# 2 DMX-512 transmitter

## 2.1 System overview

*Figure 2* represents a typical DMX-512 system.

1. The multiple receivers are connected to the DMX host in a daisy-chain manner and every packet goes through every receiver in its entirety.
2. Each controller sends the packet to the RS-485 transceiver, which transmits the corresponding differential signal to the DMX-512 receiver.
3. Each transmitter is programmed with a number of bytes to be transmitted in each packet.

**Figure 2. Typical DMX-512 transmitter system**

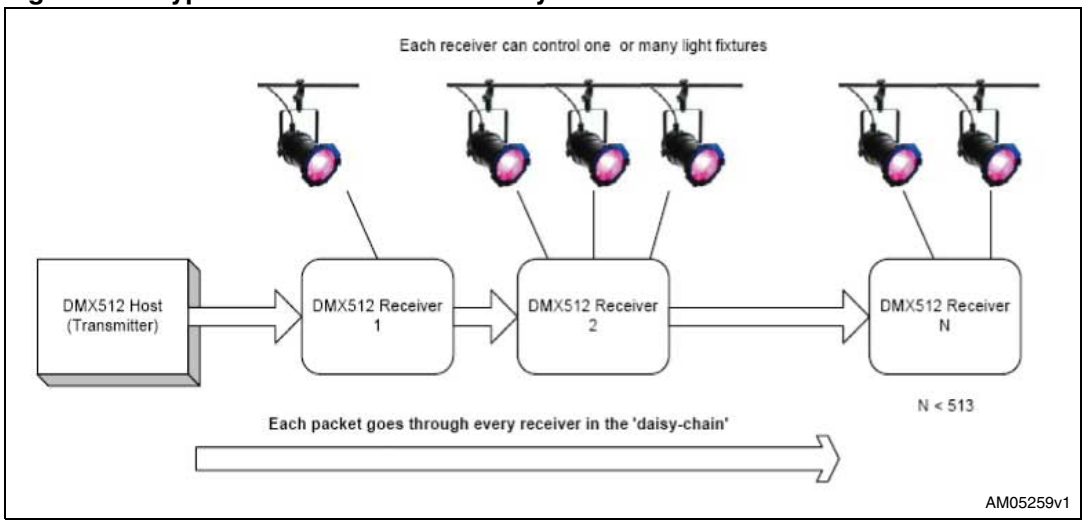
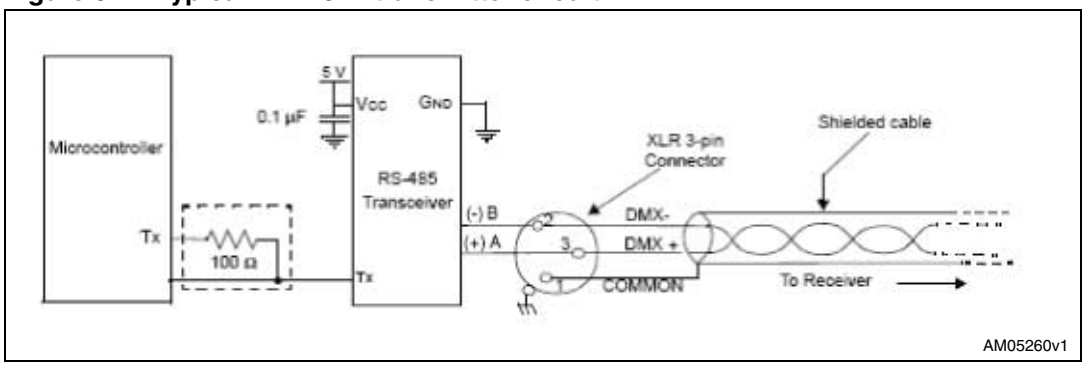



**Figure 3. Typical DMX-512 transmitter circuit**



## 2.2 DMX-512 transmitter block diagram

*Figure 4* shows the block diagram of the DMX-512 controller transmitter. The signals are transmitted through the USART_Tx pin and an IO pin. A BREAK signal must be sent at the beginning of each new packet of data. The break signal allows receivers to synchronize with the DMX transmitter. The USART module available on the STM32F103Zx microcontrollers has the ability to automatically generate a 12-bit long break signal, corresponding to 48 µs at 250 k baud. Unfortunately, this is too short for use in a DMX-512 application as the protocol requires a minimum length of 92 µs.

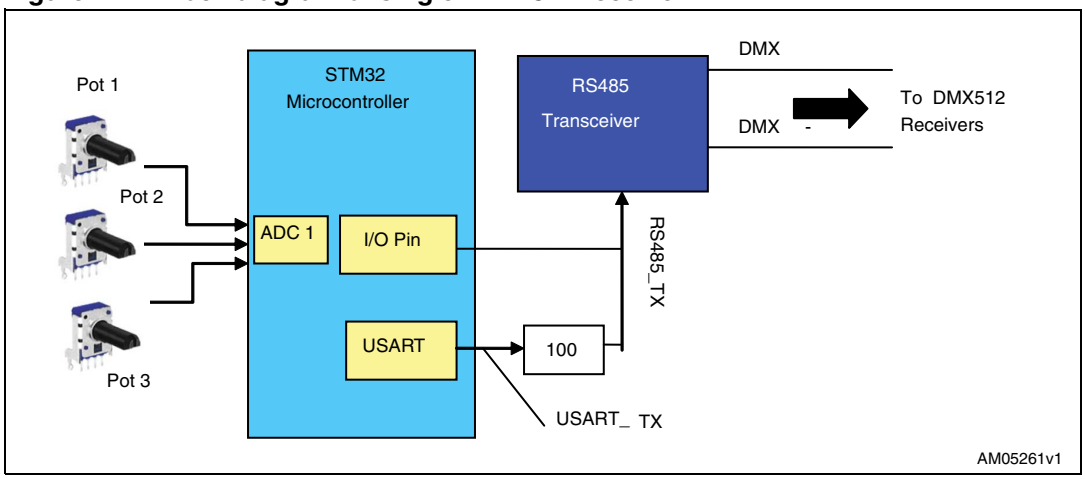
*Figure 4* shows the alternative hardware method chosen to generate the longer break signal. A 100 Ω resistor is connected in series with the microcontroller's USART transmit pin and the other end of the resistor to an I/O pin. With this solution, the break time can be varied in the software from 92 µs to 176 µs to meet the DMX protocol break time specification. When a Break signal is sent, the I/O pin is driven low. At a later stage, the I/O pin is tri-stated to allow transmission from the USART to resume.

The dimming data is 8 bits wide, where '0' represents a light off and '255' represents full intensity. *Figure 1 on page 5* shows the digital representation of the dimming data. To generate the two stop bits required by the DMX-512 protocol, the STM32F103Zx USART is configured for 8-bit mode.

The dimming data is stored in a 512-byte buffer (TX Buffer), located in the RAM memory. The data is written to or read from the buffer using the indirect addressing registers available in the microcontroller for linear memory access.

A counter keeps track of the number of bytes transmitted from the buffer. There are three data slots in this particular example, but there could be a maximum of 512. The dimming data value is taken from the potentiometer that is connected to channel_15, channel_5 and channel_7 of ADC1. Since the ADC is a 12-bit one, the data read from the potentiometer is 12-bit data, so to get it converted into 8-bit data the data is shifted 4 bits to the right.

**Figure 4. Block diagram of single DMX-512 receiver**

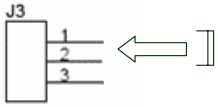


## 2.3 RS-485 transceiver

*Figure 5* shows the front view of the RS-485 transceiver board.

The jumpers on the RS-485 transceiver board should be set as shown below to receive the DMX differential signal from the transmitter and transmit the signal to the microcontroller.

- Jumper J3 on pin 1 and pin 2




- Jumper J4 on pin 1 and pin 2

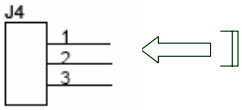



**Figure 5. Front view of RS-485 transceiver board**

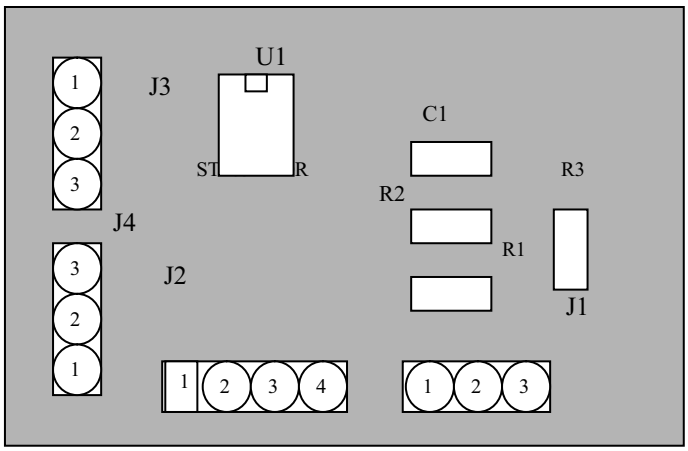
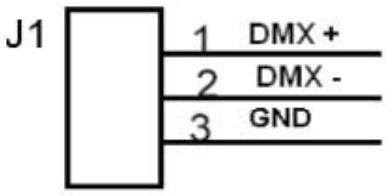


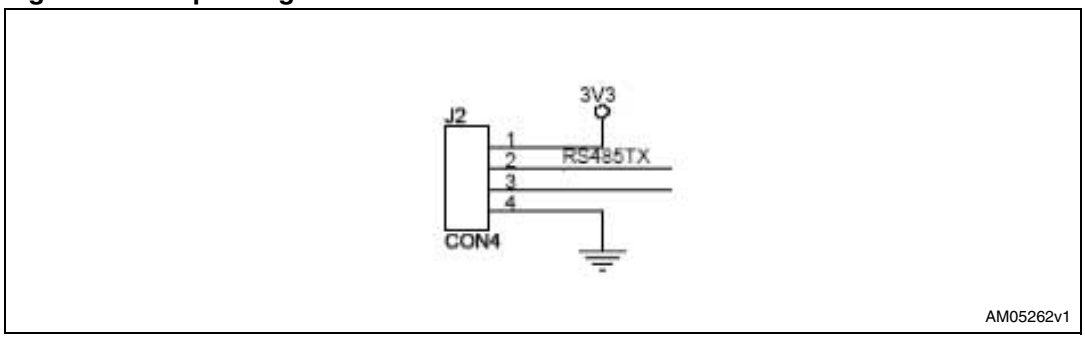

*Figure 6* and *Figure 7* show the connections of J1 and J2 on the RS-485 transceiver board.

**Figure 6. J1 pin diagram**




DMX+ and DMX- are the differential signals from the DMX-512 transmitter. NC means *not connected*.

**Figure 7. J2 pin diagram**

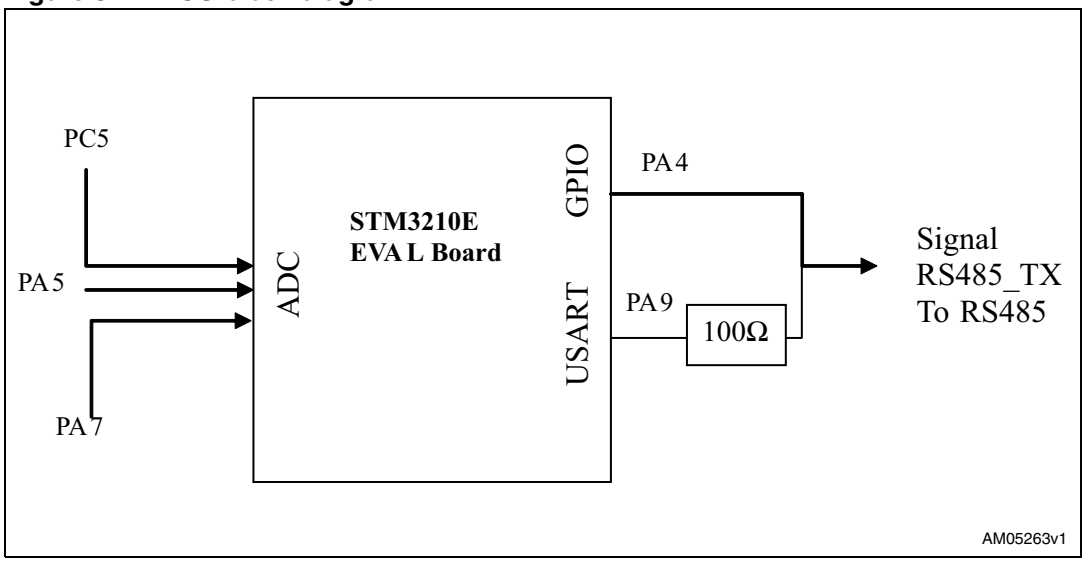



RS485_RX is the signal given to the STM32. NC means *not connected*.

## 2.4 MCU block diagram

*Figure 8* shows the connections to the STM3210E-EVAL board.

**Figure 8. MCU block diagram**




The USART_Tx pin (PA9) is connected to PA4 through 100 Ω resistors. PA4 is connected to RS485_Tx of the RS-485 transceiver board.

- PA4 is configured as push-pull low to send the BREAK signal with a specified duration, and the BREAK signal in turn is configured in tri-stated mode.
- The time of the BREAK signal is defined by TIM2.
- The USART_Tx pin (PA9) is used for sending the start code and data.
- The MARK time between slots is defined by TIM3.
- PC5, PA5 and PA7 are configured as analog inputs and ADC1 is configured as scan with continuous conversion mode. The DMA pushes the converted data of these three analog inputs directly into the internal RAM.

# 3 Demonstration firmware

This section describes the software developed for the DMX-512 transmitter. A reset sequence (“BREAK” and “MARK after BREAK”) is sent. A null start code and data slots are sent thereafter. The software is described as it appears in each of the files, starting with the DMX initialization file followed by the main file and then the interrupt file. The software is developed using IAR EWARM5.3. The project uses the STM32 firmware library version 3.0.0.

The demonstration firmware consists of four main parts.

1. A *dmx_init.c* file for configuring all the peripherals used in this application.
2. A main routine.
3. A Timer2 interrupt routine.
4. A Timer3 interrupt routine.

## 3.1 *dmx_init.c* file description

The *dmx_init* file is used for configuring different peripherals used in the application. Below is the description of all the functions:

### 3.1.1 USART1_Intialise

This function configures USART1 in transmitter mode with a baud rate of 250 kbps.

### 3.1.2 Timer2_Initialise

This function configures channel 1 to generate the reset sequence.

### 3.1.3 Timer3_Intialise

This function configures Timer3 to send a MARK time between slots.

### 3.1.4 Send_ResetSequence

This function is called in the main routine each time a data slot is sent. Timer2 is used to send the reset sequence.

## 3.2 *main.c* file description

The *main.c* file contains all the data required to initialize the peripherals used for this application. The effective packet time – including the reset sequence – is calculated. The break-to-break time is defined in the firmware, and the time required after the last slot is calculated based on this time.

The following sections describe each of the functions defined in the *main.c* file.

### 3.2.1 RCC_Configuration

This function enables the high-speed external crystal, including the PLL. It also configures various system clocks.

### 3.2.2 GPIO_Configuration

This function sets up various IO ports used for the application.

### 3.2.3 USART1_Intialise

This function initializes USART1 in transmitter mode.

### 3.2.4 DMA1_Channel1_Configuration

This function configures DMA1.

### 3.2.5 ADC_Configuration

This function configures ADC1.

### 3.2.6 Timer2_Intialise

This function initializes Timer2 to send the reset sequence.

### 3.2.7 Timer3_Intialise

This function initializes Timer3 to send the MARK time between slots.

### 3.2.8 NVIC_Configuration

This function sets up the interrupts used in the STM32F103Zx.

### 3.2.9 Send_ResetSequence

This function sends the reset sequence (BREAK and MARK AFTER BREAK).

## 3.3 Timer2 interrupt routine

In the timer2 interrupt subroutine, when the CC1 interrupt is initiated (that is, after completion of the break time), the PA4 port is converted into a floating IO. When the update interrupt is initiated, timer2 is disabled and reset. A null start code is then sent with timer3 enabled.

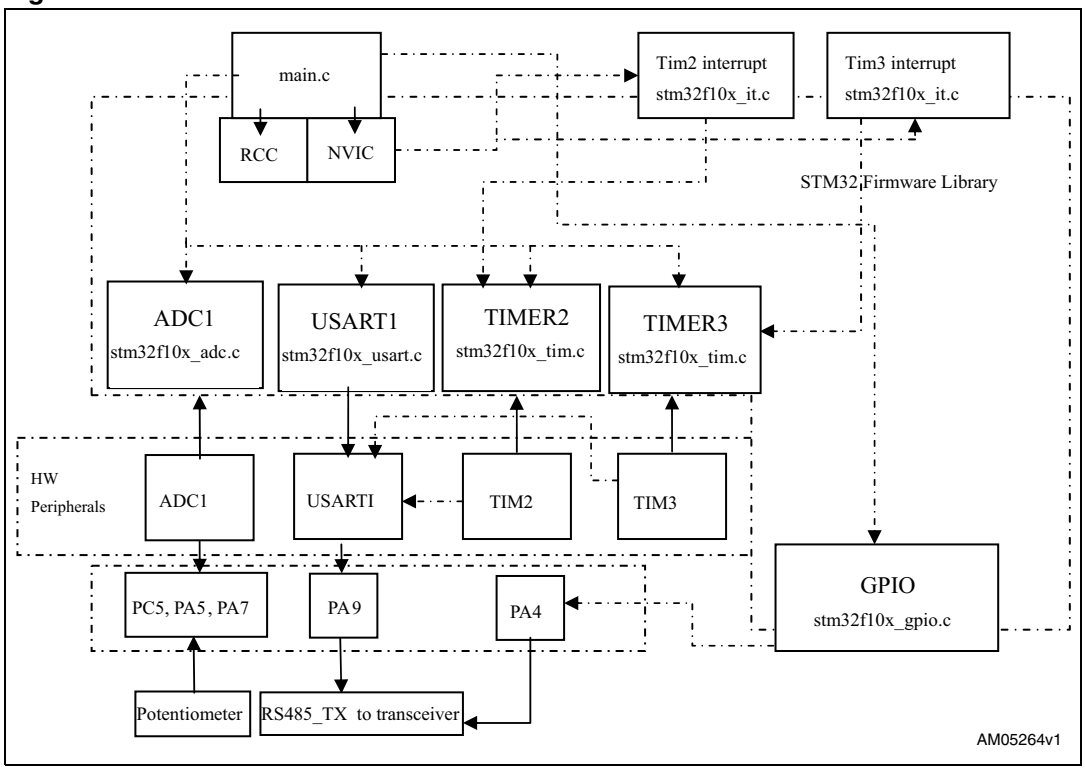
## 3.4 Timer3 interrupt routine

The timer3 interrupt subroutine is used to send a MARK time between slots.

# 4 Firmware architecture overview

*Figure 9* shows the architecture of the demonstration firmware.

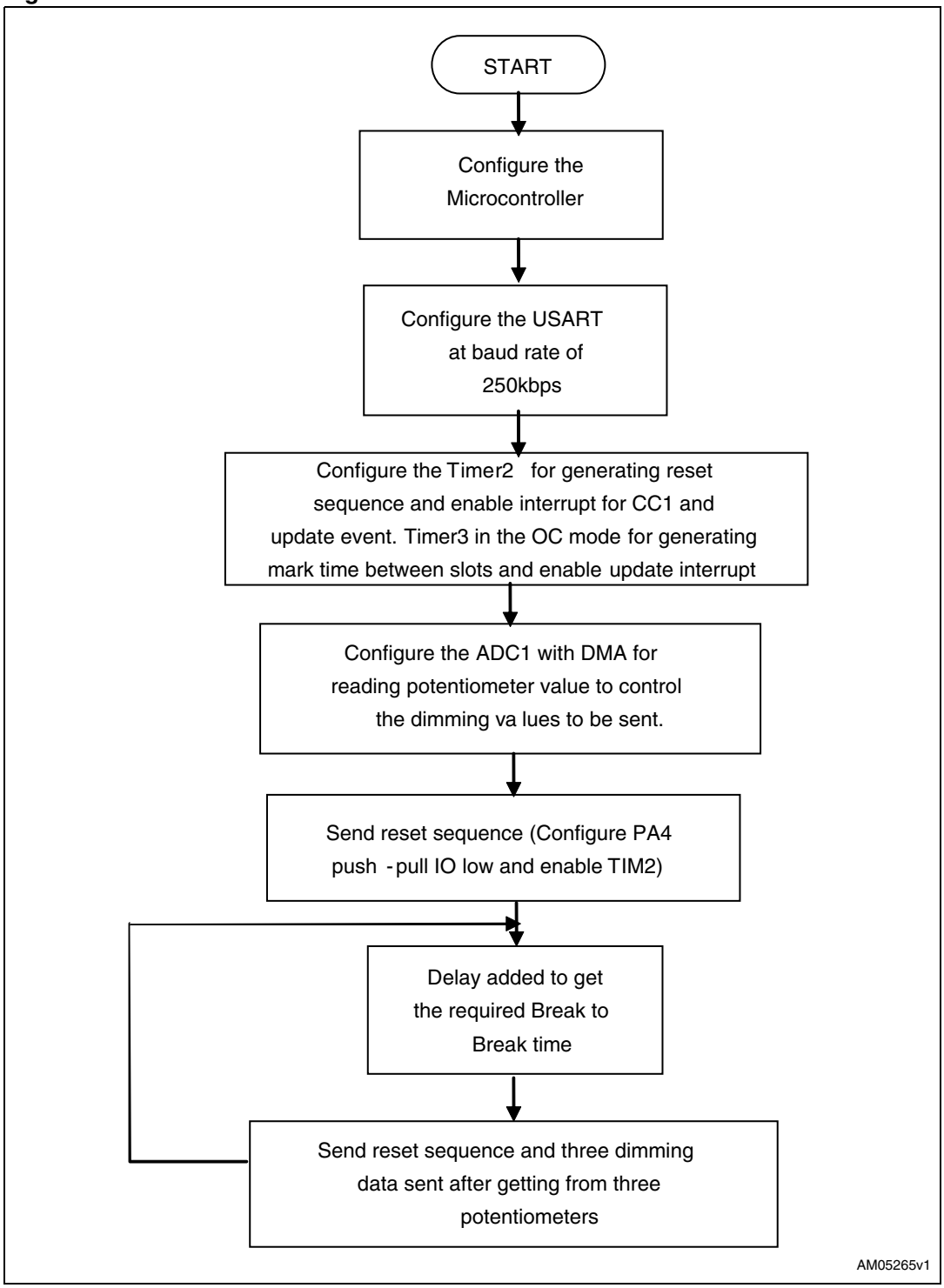
**Figure 9. Demonstration firmware architecture overview**



# 5 Firmware flowcharts

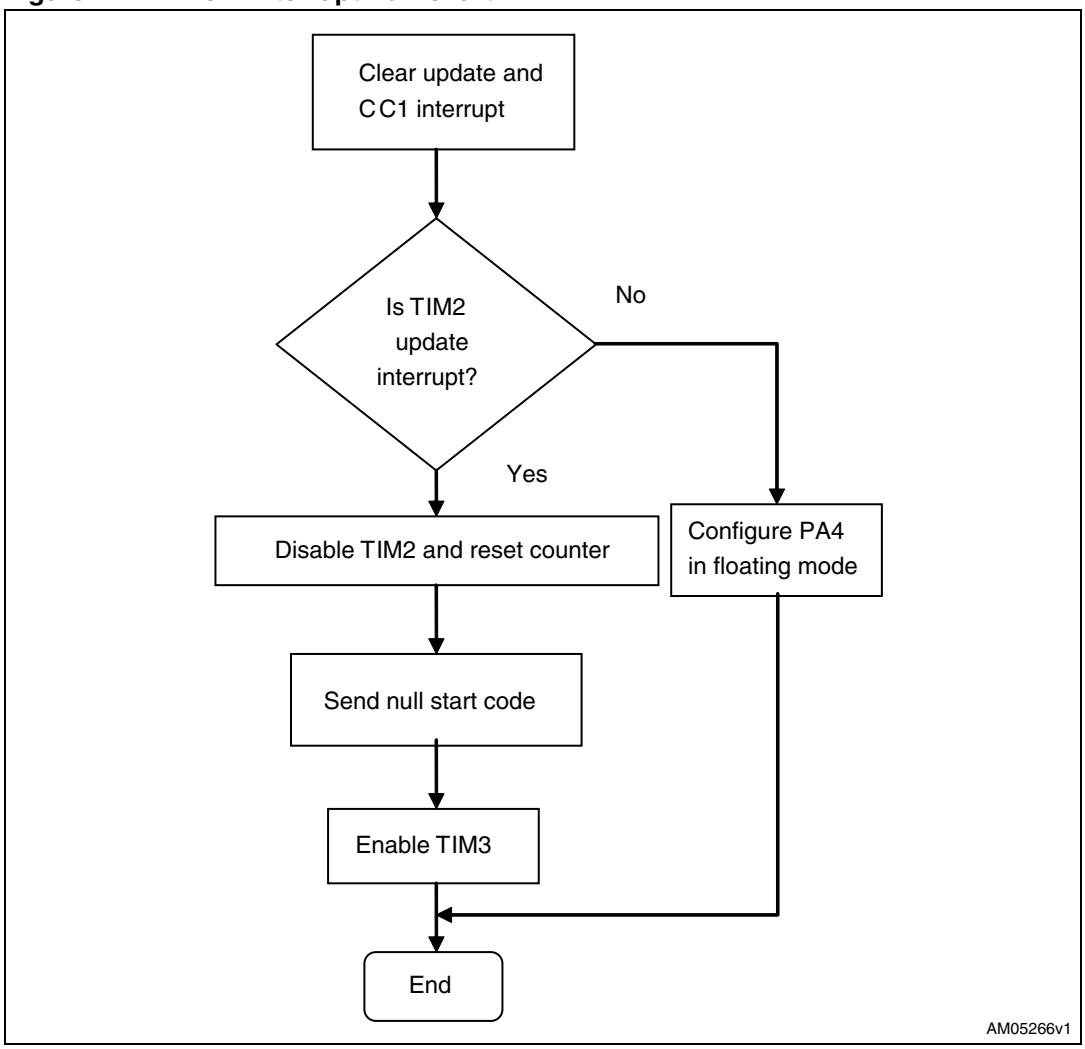
## 5.1 Main routine

**Figure 10. Main routine flow chart**



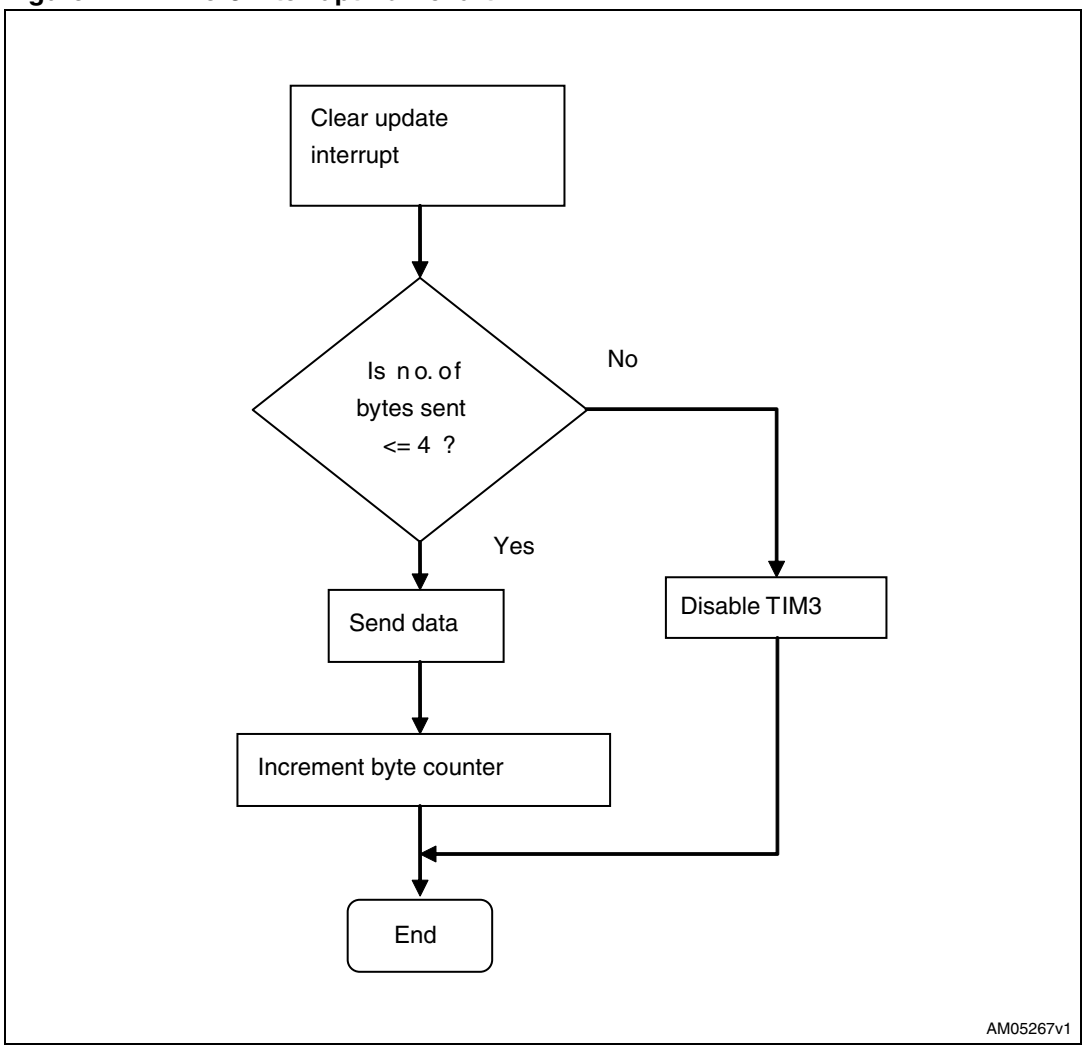
## 5.2 Timer2 interrupt routine

**Figure 11. Timer2 interrupt flow chart**



## 5.3 Timer3 interrupt routine

**Figure 12. Timer3 interrupt flow chart**



# 6 Key features/specifications

The developed solution:

- transmits data according to the DMX512 2008 standard.
- configures the number of bytes transmitted.

# 7 Schematic and layout

**Figure 13. Schematic of RS-485 transceiver board**

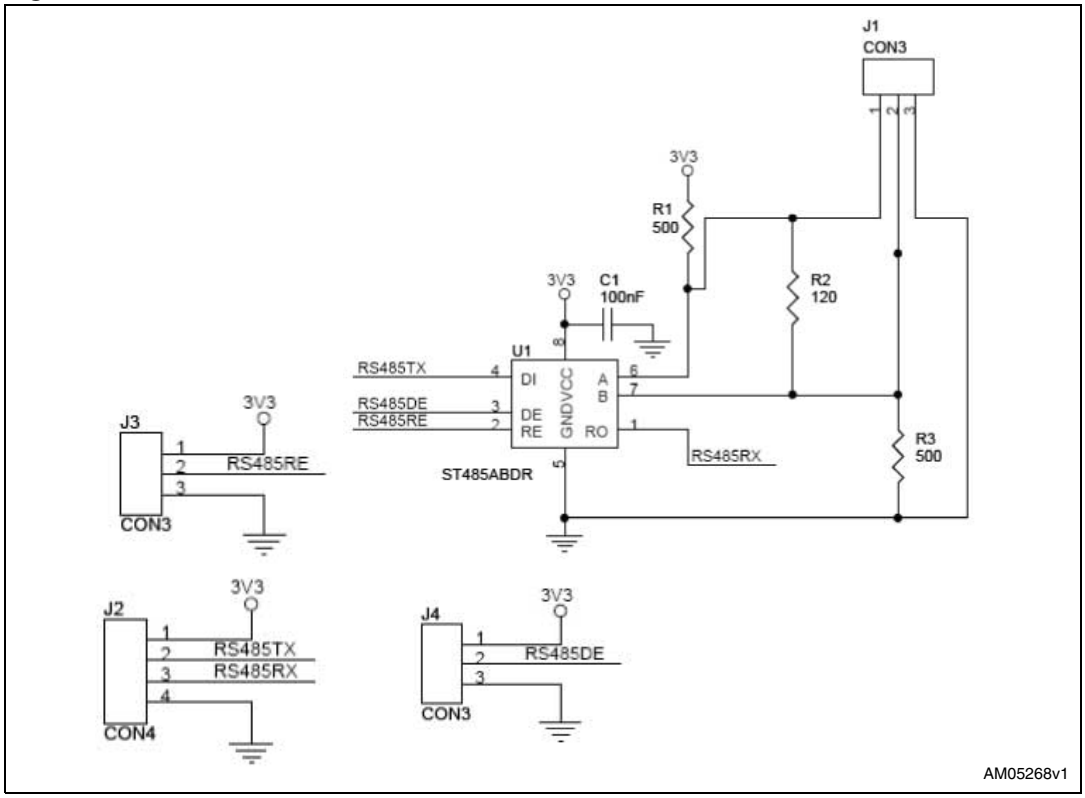
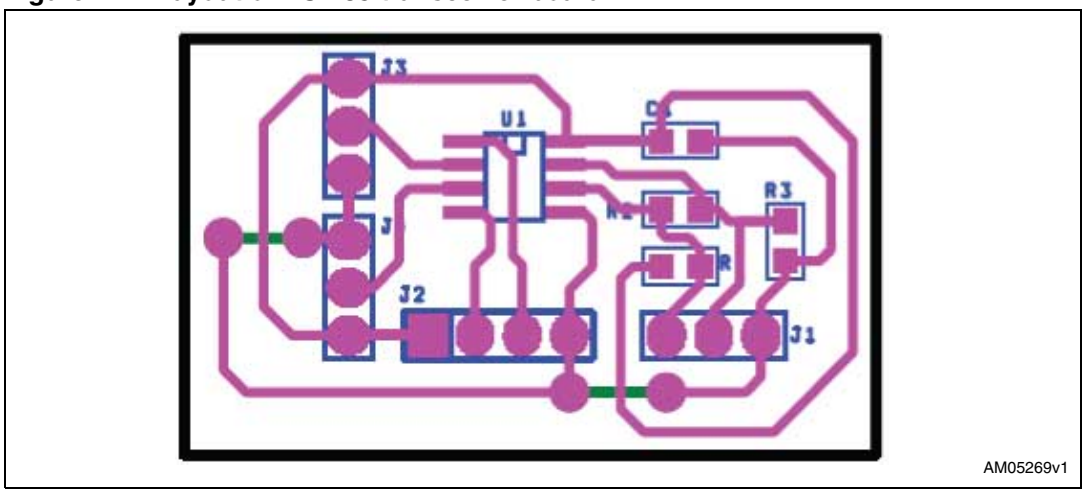



**Figure 14. Layout of RS-485 transceiver board**



# 8 Revision history

**Table 2. Document revision history**

| Date | Revision | Changes |
|---|---|---|
| 02-Feb-2010 | 1 | Initial release. |
| 17-Mar-2010 | 2 | Changed document title from “DMX-512 communications protocol algorithm for transmitter, based on the STM3210E-EVAL” to “Demonstration firmware for the DMX-512 communication protocol transmitter based on the STM32F103Zx”.<br>Replaced some references to the STM32 device with STM32F103Zx throughout the document.<br>Corrected board part number in *Section 2.4: MCU block diagram* and *Figure 8 on page 10*.<br>Minor text changes throughout the document. |

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.



STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

**www.st.com**