AN11154

DMX512/RDM master using LPC11U1x

Rev. 1 — 1 February 2012 Application note

Document information

Info	Content
Keywords	LPC11U12FHN33; LPC11U12FBD48; LPC11U13FBD48; LPC11U14FHN33; LPC11U14FHI33; LPC11U14FBD48; LPC11U14FET48, LPC11U1x, ARM, Cortex M0, DMX, DMX512, RDM, USB, Architectural Lighting, Entertainment Lighting, USB - DMX interface
Abstract	This application note describes the use of the NXP LPC11U1x Cortex M0 microcontroller to create a RDM enabled DMX512 Master (USB - DMX interface).

Revision history

Rev	Date	Description
1	20120201	Initial version.

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Document purpose

The purpose of this document is to explain the NXP DMX512/RDM demoboard design. Both hardware and software are described in detail.

This document is intended for technical persons, such as system architects and hardware/software engineers, interested in designing/developing a DMX512 master using an NXP microcontroller.

2. Introduction

The DMX512 master is an example implementation for an RDM enabled DMX512 controller and monitoring device, built around the NXP LPC11U1x microcontroller. The DMX512 master features a USB interface, a red heartbeat LED, and a green traffic LED. The USB and UART of the LPC11U1x MCU are the main hardware blocks needed to implement the DMX512 master. The DMX512 master enables Remote Device Management, but the host has to assemble/parse the RDM packets to be sent to/received from the RDM devices connected to the DMX transmission line.

References in this document to DMX512 refer to DMX512-A, since both hardware and software are designed using the latest standard (see [1]).

The software can be built with

- LPCXpresso v4.1.0_190, and
- IAR Embedded Workbench for ARM v6.20.4.

Fig 1. DMX512 master

3. DMX512 master hardware

3.1 Physical layer

The schematic of the physical DMX layer is depicted in Fig 2.

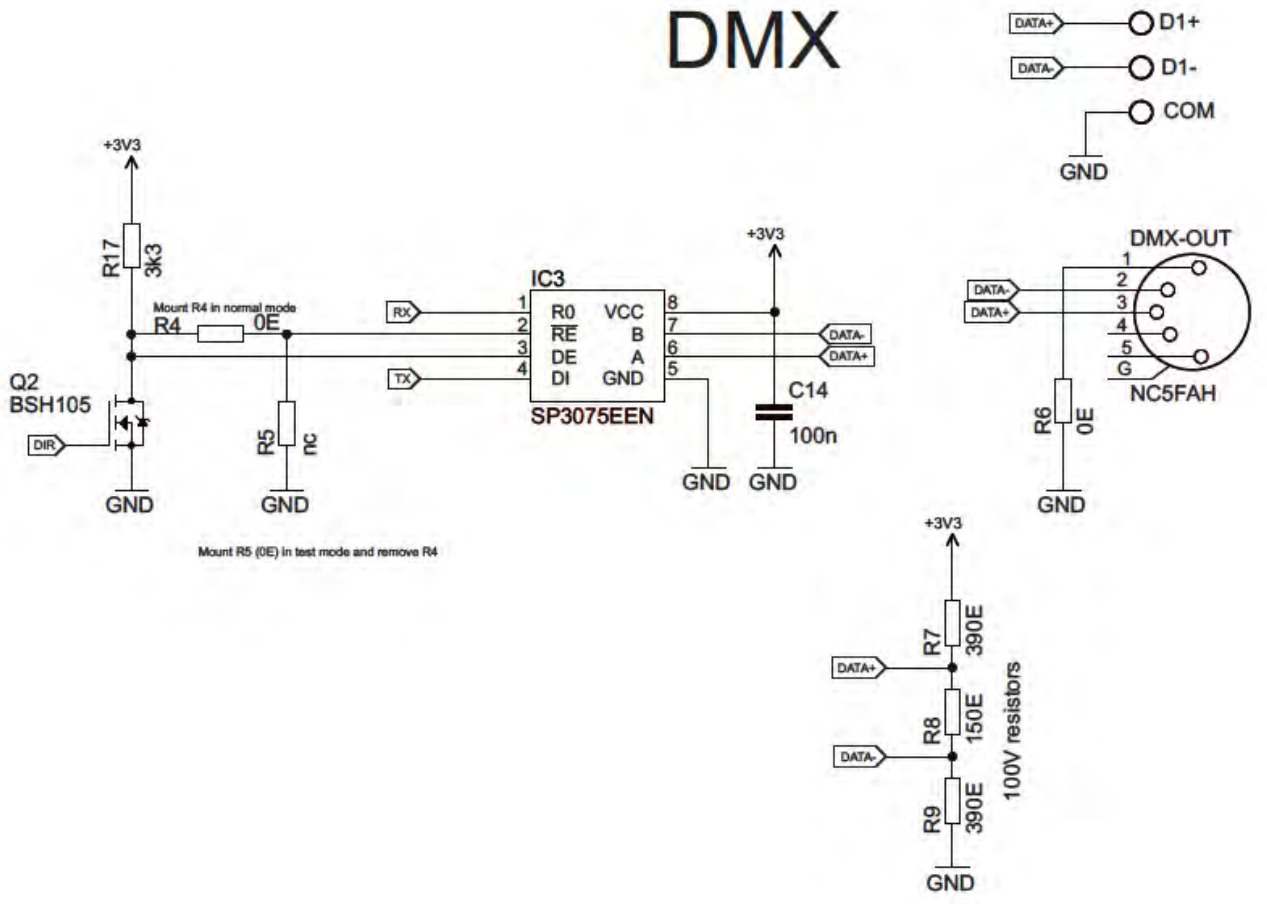

Fig 2. DMX512 physical layer

The DMX bus connects to DATA+ and DATA- of the DMX-OUT connector. This is a 5-pole female XLR receptacle. A 3-pin spare connector (not mounted) is available for other form factors. IC3 is an RS-485/RS422 transceiver, providing 15 kV ESD protection. Resistors R7-R9 (100 V types) determine the impedance on the DMX bus to avoid reflections and set the DC levels on DATA+ and DATA- necessary for correct RDM communication. During startup the IO pins of the micro are high, and FET Q2 is used as an inverter to switch the DIR of IC3 to receive mode to avoid unwanted driving of the DMX bus.

The physical layer can be set in test mode (loopback) by removing resistor R4 and placing resistor R5 (0 Ohm).

3.2 Microcontroller

The schematic of the microcontroller is depicted in Fig 3.

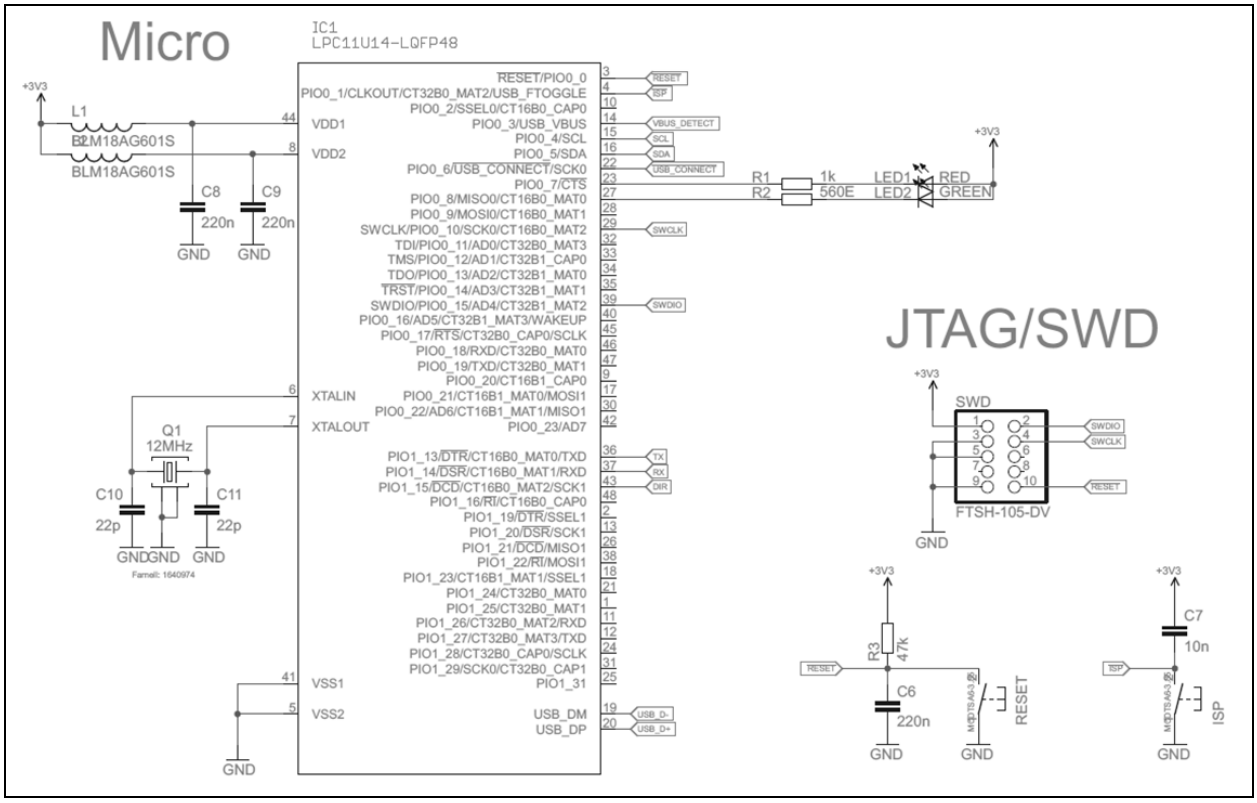

Fig 3. Microcontroller on DMX512 master

The system is built around NXP's new LPC11U14 device, which is a Cortex-M0 running at frequencies of up to 50 MHz. Included are a USB 2.0 Full-Speed device controller, 32 kB on-chip flash, 4 kB on-chip EEPROM, 6 kB SRAM (4 kB main, 2 kB USB), SSP, I^2C, UART, ADC, etc.

Debugging and flashing connection is provided by means of header SWD, which complies with the 10-pin SWD standard as supported by many flash and software tools.

RESET and ISP push buttons are available. The RESET button can be used to reset the microcontroller. The ISP button can be an input to the SW running on the microcontroller. ISP mode, entered by pressing the ISP button during reset, is not supported on this board.

Two LEDs are provided, one red and one green.

3.3 USB Interface

The schematic of the USB interface is depicted in Fig 4.

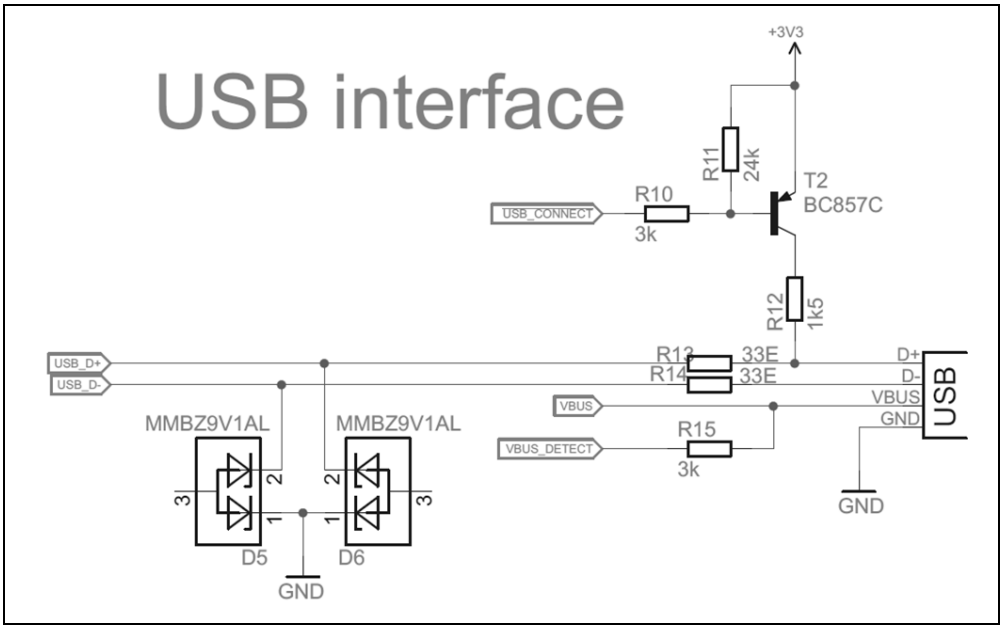

Fig 4. USB interface on DMX512 master

The USB interface uses a Type B socket.

The connection to the USB is accomplished by bringing USB_D+ (for a full-speed device) HIGH through a 1.5 kOhm pull-up resistor via T2. The USB SoftConnect feature can be used to allow software to finish its initialization sequence before deciding to establish connection to the USB. Re-initialization of the USB bus connection can also be performed without having to unplug the cable.

D5 and D6 provide extra ESD protection to the system.

3.4 System power

The schematic of the power supply from the USB interface is shown in Fig 5.

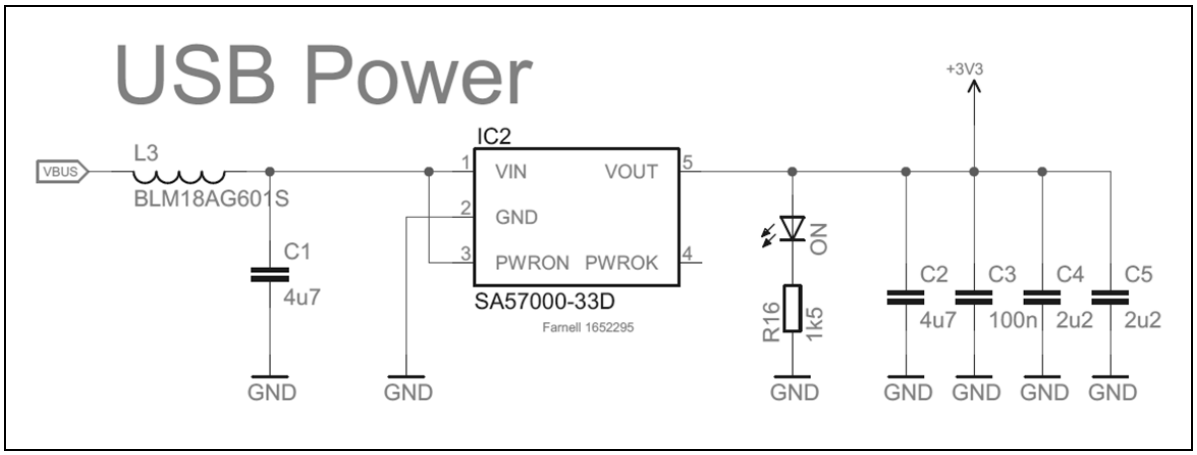

Fig 5. Power circuit of DMX512 master

When USB power is provided, the ON LED goes on.

Note: IC2 (SA57000-33D) is pin-compatible with the LP3985IM5-3.3 device.

3.5 Board layout

The layout of the board (3.2” x 2”) is shown in Fig 6.

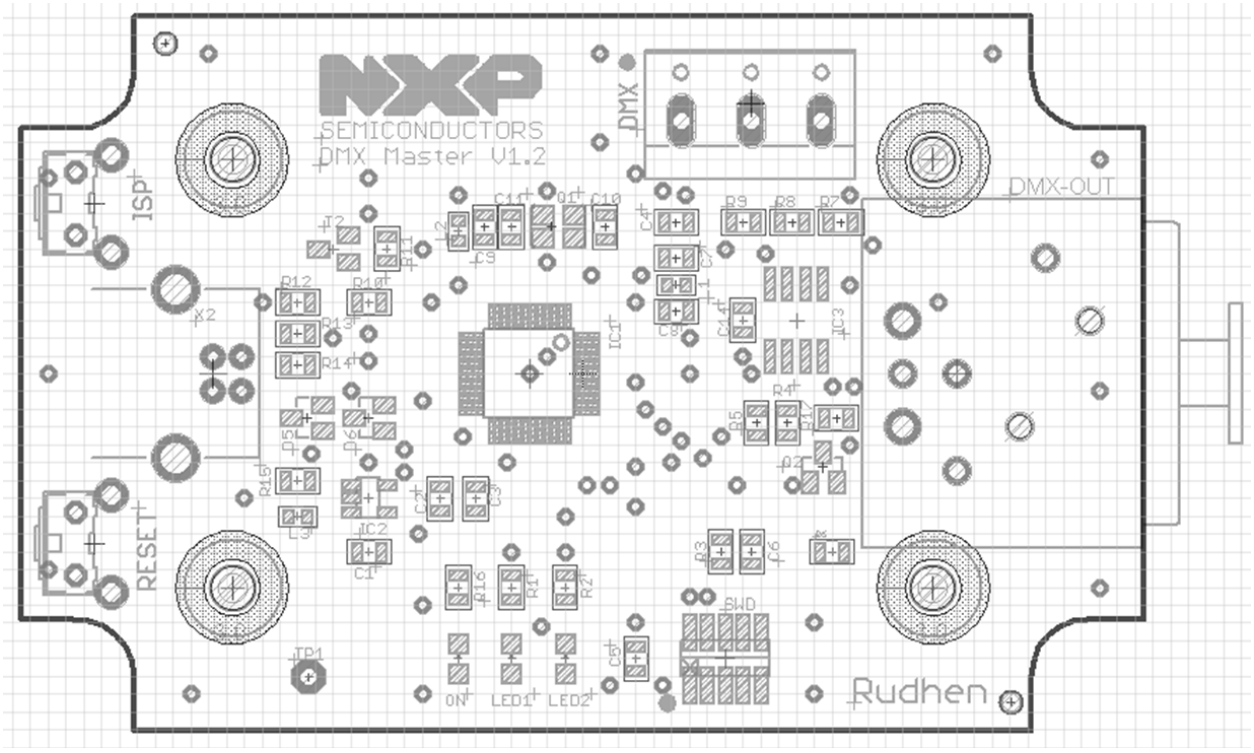

Fig 6. Board layout of DMX512 master

On the right is the DMX-OUT receptacle and on the top-right is the (not mounted) DMX extension plug.

Left are the USB Type B socket and the RESET and ISP switches.

In the heart of the system is the microcontroller, and on the bottom are the two LEDs and the SWD interface.

Table 1. DMX512 master parts list

Part	Value	Device	Package	Farnell	Description
T2	BC857C	BC857C-PNP-SOT23-BEC	SOT23-BEC	8734771	PNP Transistor
Q2	BSH105	BSH105	SOT23	1758066	N-channel enhancement mode MOS transistor
C14	100n	C0805	P0805		Capacitor
C3	100n	C0805	P0805		Capacitor
C7	10n	C0805	P0805		Capacitor
C6	220n	C0805	P0805		Capacitor
C8	220n	C0805	P0805		Capacitor
C9	220n	C0805	P0805		Capacitor
C10	22p	C0805	P0805		Capacitor
C11	22p	C0805	P0805		Capacitor
C4	2u2	C0805	P0805		Capacitor
C5	2u2	C0805	P0805		Capacitor
C1	4u7	C0805	P0805		Capacitor
C2	4u7	C0805	P0805		Capacitor
Q1	12MHz	CRYSTAL4	XTAL4	1640974	Crystal, 2.5mm x 3 mm
SWD	FTSH-105-DV	FTSH-105-DV	127_2R10_SMT	1767036	Samtec connector, 10-way
L1	BLM18AG601S	L0603	P0603		Inductor
L2	BLM18AG601S	L0603	P0603		Inductor
L3	BLM18AG601S	L0603	P0603		Inductor
IC1	LPC11U14-LQFP48	LPC11U14-LQFP48	LQFP48		NXP Cortex-M0 MCU, LPC11U14FBD48/201
ISP	MCDTSA6-2K	MCDTSA6-2K	TACT90	9471839	Tact switch, vertical, through-hole
RESET	MCDTSA6-2K	MCDTSA6-2K	TACT90	9471839	Tact switch, vertical, through-hole
D5	MMBZ9V1AL	MMBZ9V1AL	SOT23	1829223	Double ESD protection diodes
D6	MMBZ9V1AL	MMBZ9V1AL	SOT23	1829223	Double ESD protection diodes
DMX-OUT	NC5FAH	NC5FAH	NC5FAH	8020418	Neutrik Audio Connector XLR SERIES
ON	OVS-0803(B)	OVS-0803(B)	CHIPLED_0805	1716765	Chip LED 0805
LED2	OVS-0804(G)	OVS-0804(G)	CHIPLED_0805	1716766	Chip LED 0805
LED1	OVS-0808(R)	OVS-0808(R)	CHIPLED_0805	1716768	Chip LED 0805
R4	0E	R0805	P0805		Resistor
R6	0E	R0805	P0805		Resistor
R8	150E/100V	R0805	P0805		Resistor 100V
R1	1k	R0805	P0805		Resistor
R12	1k5	R0805	P0805		Resistor
R16	1k5	R0805	P0805		Resistor

Part	Value	Device	Package	Farnell	Description
R11	24k	R0805	P0805		Resistor
R13	33E	R0805	P0805		Resistor
R14	33E	R0805	P0805		Resistor
R7	390E/100V	R0805	P0805		Resistor 100V
R9	390E/100V	R0805	P0805		Resistor 100V
R10	3k	R0805	P0805		Resistor
R15	3k	R0805	P0805		Resistor
R17	3k3	R0805	P0805		Resistor
R3	47k	R0805	P0805		Resistor
R2	560E	R0805	P0805		Resistor
R5	nc	R0805	P0805		Resistor
IC2	SA57000-33D	SA57000-33D	SOT23-5	1826832	CapFREE 150 mA, low-noise, low-drop LDO
IC3	SP3075EEN-L	SP3075EEN-L	SO08	9386688	RS-485/RS422 Transceivers
TP1	TPSPAD1-13	TPSPAD1-13	P1-13		TEST PIN
X2	USBPTH	USBPTH	USB-B-PTH	1308876	USB Connectors
DMX	DMX	W237-103	W237-103		WAGO SCREW CLAMP

4. DMX512 master software

4.1 Architecture

The software of the DMX512 master has a modular architecture, meaning that each hardware block has its own software driver. The software running on the DMX512 master does not incorporate a Real Time Operating System, in order to keep the memory footprint of the firmware as small as possible. The software is written in the “C” programming language.

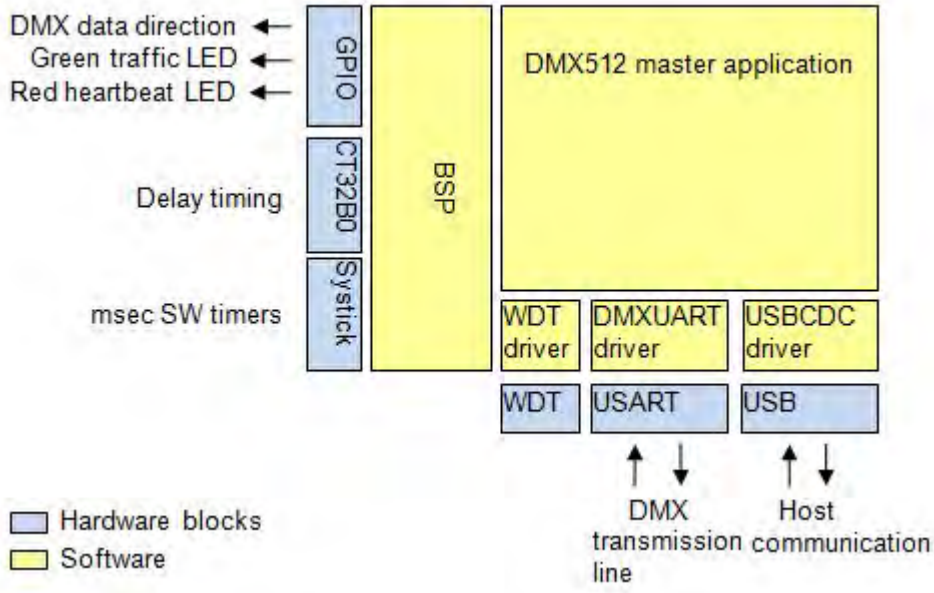

Fig 7. Software architecture of DMX512 master

The software of the DMX512 master has been developed for two purposes: to let the DMX512 master function as a controller of the DMX transmission line, and to let it function as a monitoring device of communication on the DMX transmission line. The protocol used for communication between the host (PC) and the DMX512 master (and vice versa) is based on the “DMX USB Pro Widget API Specification 1.44” from ENTTEC Pty/Ltd[4].

4.2 Main software execution flow

Fig 8 shows the flowchart of the DMX512 master software. The DMX master init is described in Section 4.3. The following described actions are repeated until the DMX512 master is powered down or reset.

The main loop starts in DMX input mode (left branch in flowchart), trying to read a DMX512 packet from the DMX transmission line. This is done by calling the function Uart_RecvDmxSlotValues(). When data is received (only possible when another DMX controller is connected to the DMX transmission line) the GREEN traffic LED is toggled and the DMX packet is sent to the host.

The next action is checking whether there is any data received from the host via USB. The host can put the DMX512 master in output mode, allowing the main loop to execute the right branch of the flowchart. The GREEN traffic LED is toggled and the DMX packet, as received from the host, will be put on the DMX transmission line. If the DMX packet is instead an RDM packet, the DMX512 master will go back to DMX input mode (executing the left branch of the flowchart) to be able to receive a RDM response packet from the addressed RDM device. If it is not an RDM packet, then the DMX512 master will wait until it is time (given the current DMX refresh rate) to resend the DMX packet, or send a new DMX packet (received while it was waiting), or switch back to DMX input mode on request of the host.

The DMX512 master uses double buffering for receiving DMX packet data from the host; this is to ensure that a DMX packet is not corrupted by (partially) received new data from the host while being output on the DMX transmission line. The watchdog is fed every four seconds, and at the same time the actual DMX refresh rate is calculated which is available for the host to be used (displayed).

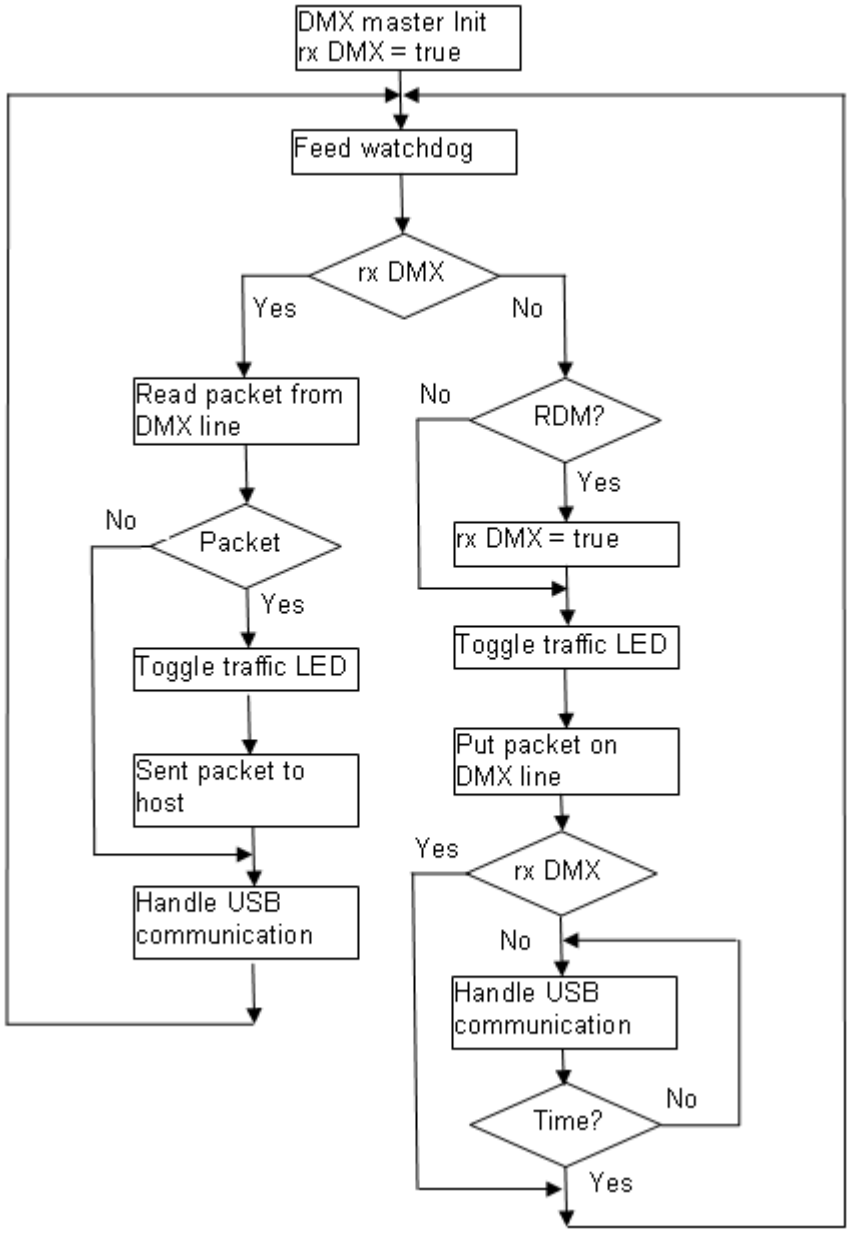

Fig 8. Flowchart of DMX512 master main loop

4.3 Initialization sequence

When the LPC11U1x microcontroller starts up, as a result of the DMX512 master being powered, the stack pointer is loaded with the value from address 0:3, and the program counter is loaded with the value from address 4:7 being the address of the reset handler. This will result in the calling of main().

The function main() implements the infinite loop which handles the USB communication and transmits/receives data to/from the DMX transmission line. Before the endless loop is entered the function SystemInit() is called, which sets up the main PLL that is used to set the proper CPU clock, as configured by LPC_CORE_CLOCKSPEED_HZ in 'app_config.h'. Function SystemInit() comes with CMSIS.

After calling SystemInit() the function bsp_init() is called. This function sets up the system tick counter to 1000 ticks per second, it enables the HW blocks GPIO / CT32B0 / USART / USB / WWDT, it configures the LPC pins for controlling the LEDs / USART pins / etc, and it initializes the DMX UART driver, USB CDC driver and WWDT driver. Then init_globals() is called which initializes the variables used within main.c, and puts the DMX512 master in monitoring (receive) mode so that it won't disturb the communication on the DMX transmission line. The last thing that is done before the main loop is entered is claiming a milliseconds counter to be used for timing measurement via the BSP.

4.4 DMX UART driver

The UART driver in the DMX512 master is based on the UART driver of the DMX512 slave, but has some functionality only needed by the DMX512 master. For example, the DMX512 slave only has to support communication via the DMX transmission line, so all read and write actions to the UART driver can be blocking actions until the UART driver has finished its work. The DMX512 master doesn't allow any blocking actions when the USB communication and DMX transmission line communications are simultaneously active. The design features in this driver needed for the DMX512 master are explained in Section 4.4 of AN11153[5].

4.4.1 Function Uart_XmitDmxDone

Since the DMX512 master must also handle USB communication when outputting a DMX packet on the DMX transmission line, the function Uart_XmitDmxSlotValues() is called in a non-blocking way (parameter 'blocking_write' = false). The application can now use the function Uart_XmitDmxDone() to check if the DMX packet has been completely sent, while at the same time handling USB communication. The following source code lines from the application show how this is used:

```
1    Uart_XmitDmxSlotValues(dmx_packet, packet_len,
2                           break_time, mab_time,
3                           false, xmit_disc_request);
4    while(!Uart_XmitDmxDone())
5    {
6        handle_host_data(true);
7    }
```

4.4.2 Function Uart_XmitDmxSlotValues

The DMX512 master can be in monitoring mode or in controlling mode. When in controlling mode the DMX512 master will be busy outputting DMX packets most of the time. However, when it is busy with Remote Device Management actions (on command of the host), it will output an RDM packet and immediately switch back to DMX input mode to receive an RDM response packet from the addressed RDM device. Only when a DISC_UNIQUE_BRANCH RDM packet is output the parameter 'xmit_disc_request' must be set to true; the reason for this is that the response on this packet doesn't start with a "Break" for which the driver must take special actions (detect line activity) to be able to capture this response.

4.4.3 Function Uart_RecvDmxSlotValues

Since the DMX512 master must also handle USB communication when receiving a DMX packet from the DMX transmission line, the function Uart_RecvDmxSlotValues () is normally called in a non blocking way (parameter 'blocking_read' = false). Again the only exception is when receiving the response on the DISC_UNIQUE_BRANCH RDM packet, then the parameter 'blocking_read' must be set to true. Since multiple RDM devices can send a response at the same time (even giving collisions on the DMX transmission line), a timeout mechanism must be used to pass back the received data to the application. For the discovery process to work properly it must know whether there were multiple, one or none RDM devices responding. If there was no line activity during this timeout period, this function will return 0. If there were responding RDM devices, line activity is noticed and the host is informed about this. This line activity detection will only take place in the DMX512 master when it waits on the response on the DISC_UNIQUE_BRANCH RDM packet.

4.4.4 Function Uart_DmxBusMonitoring

The DMX512 master can also be in monitoring mode (default state when it is powered). Then all packets monitored on the DMX transmission line will be captured and sent to the host. This mode is a little bit different from a DMX512 master being in controlling mode while waiting for DMX input.

A DMX512 slave can receive an RDM packet from the DMX512 master, and a DMX512 master can receive an RDM response packet from a DMX512 slave. In both cases such an RDM packet is not followed by a "Break" of a next packet. So to be able to pass such an RDM packet to the application, the DMX UART driver has knowledge of the structure of an RDM packet. In monitoring mode, however, the DMX UART driver will detect the DISC_UNIQUE_BRANCH RDM request packet and wait for the response. This is necessary; otherwise, the DISC_UNIQUE_BRANCH RDM response packet data is never passed back to the host when in monitoring mode.

4.4.4.1 Example

The next data comes from two DMX512 masters (and two GUIs on the host) connected to the same DMX transmission lines (universe). The first DMX512 master should be in controlling mode, and the second DMX512 master should be in monitoring mode.

Received data by the host when the (first) DMX512 master is in controlling mode and has initiated the discovery process:

```
18h4m1s427ms : Startcode=CC

01 18 FF FF FF FF FF FF 3B 10 47 27 1D EE 00 01 00 00 00 10 00 03 00 08 B7

18h4m1s431ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 01 01 00 00 00 10 00 01 0C 00 00 00 00 00
00 FF FF FF

FF FF FF 0E C8

16s835ms : Startcode=FE

FE FE FE FE FE FE AA BB 7F BA 55 FF FD AB D5 FF 7D BE FF AA 5D AA 5F

18h4m1s445ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 02 01 00 00 00 10 00 01 0C 80 00 00 00 00
00 FF FF FF

FF FF FF 0F 49

18h4m1s469ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 03 01 00 00 00 10 00 01 0C 00 00 00 00 00
00 7F FF FF

FF FF FF 0E 4A

16s874ms : Startcode=FE

FE FE FE FE FE FE AA BB 7F BA 55 FF FD AB D5 FE FD FE FF AA 5D BE 5F
```

Notice that the host generated the RDM packets (Startcode=CC) by the different timestamp (h:m:s:ms), where the DISC_UNIQUE_BRANCH RDM responses have a timestamp (with s:ms only) that comes from the DMX512 master.

Received data by host when the (second) DMX512 master is in monitoring mode:

```
169s295ms : Startcode=CC

01 18 FF FF FF FF FF FF 3B 10 47 27 1D EE 00 01 00 00 00 10 00 03 00 08 B7

169s307ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 01 01 00 00 00 10 00 01 0C 00 00 00 00 00
00 FF FF FF

FF FF FF 0E C8 FE FE FE FE FE FE FE AA BB 7F BA 55 FF FD AB D5 FE 7F FA DF AA 5D AA
5F

169s339ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 02 01 00 00 00 10 00 01 0C 80 00 00 00 00
00 FF FF FF FF FF FF 0F 49

169s351ms : Startcode=CC
```

```
01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 03 01 00 00 00 10 00 01 0C 00 00 00 00 00
00 7F FF FF

FF FF FF 0E 4A FE FE FE FE FE FE FE AA BB 7F BA 55 FF FD EB D5 FF 7F FF FF AA 5D EE
DF
```

Notice that the DMX512 master in monitoring mode doesn't separate the RDM packets and the DISC_UNIQUE_BRANCH RDM response (shaded bytes), one packet with one timestamp, because there is no "Break" before the DISC_UNIQUE_BRANCH RDM response.

Received data by host when the (second) DMX512 master is in controlling mode (while capturing DMX input), but not initiated the discovery process (in fact not a correct mode since two controllers on the same DMX transmission lines / universe are not allowed):

```
391s289ms : Startcode=CC

01 18 FF FF FF FF FF FF 3B 10 47 27 1D EE 00 01 00 00 00 10 00 03 00 08 B7

391s293ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 01 01 00 00 00 10 00 01 0C 00 00 00 00 00
00 FF FF FF FF FF FF 0E C8

391s302ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 02 01 00 00 00 10 00 01 0C 80 00 00 00 00
00 FF FF FF FF FF FF 0F 49

391s335ms : Startcode=CC

01 24 FF FF FF FF FF FF 3B 10 47 27 1D EE 03 01 00 00 00 10 00 01 0C 00 00 00 00 00
00 7F FF FF FF FF FF 0E 4A
```

Notice that in the last data the DISC_UNIQUE_BRANCH RDM response data is not available.

4.5 USB CDC driver

The USB CDC driver is developed in another project and is not described in detail in this document. This driver comes from the file "code_bundle_lpc11uxx_keil.zip". Since this project was developed for Keil, the USB CDC software was adapted for both the GNU and the IAR "C" compiler. Other changes were made to the file "cdcuser.c" for reading and writing data via the USB interface from and to the host.

4.6 Interrupts

The LPC11U1x microcontroller has four interrupt priority levels, where 0 is the highest and 3 the lowest priority level. Table 2 shows the interrupts that are handled by the software (the actual interrupt priority level in brackets).

Table 2. Used interrupts and assigned priorities in DMX512 master software

Interrupt	Priority	Description
USB_IRQn	Highest (0)	Generated by the USB HW block. Used for transmitting/receiving data to/from the host.
UART_IRQn	Middle (2)	Generated by the UART HW block. Used for transmitting/receiving data to/from the DMX transmission line.
TickHandler	Lowest (3)	Generated by the systick timer. Used for implementing software timers.

The TickHandler is called every millisecond; millisecond counters, to be used by the application, are incremented. After one second has elapsed, this TickHandler toggles the RED heartbeat LED. The millisecond counters overflow after 2^32 milliseconds (1193 hours = 49 days).

4.7 Software source code files

The software tree holding the source code files of the DMX512 master is shown in Fig 9. All DMX512 master source code is contained in the directory LPC11U1xMaster. The files ".cproject" and ".project" are LPCXpresso project files, the files "LPC11U1xDMXmaster.*" are IAR EWARM project files.

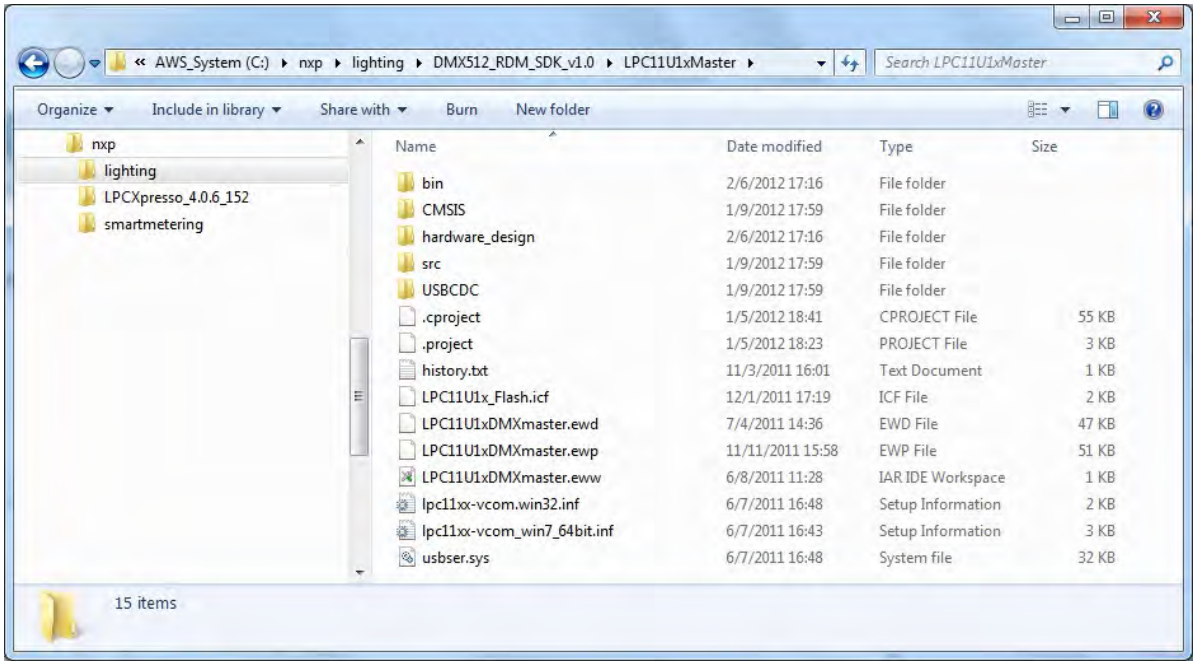

Fig 9. Software tree of DMX512 master

This directory contains a “CMSIS” directory containing the CMSIS source code, all the source files of Fig 10 in the directory “src”, which is the source code that forms the heart of the DMX512 master, and a “USBCDC” directory containing the USB software stack.

A brief description of these source code files is shown in Table 3.

Table 3. Source code file description

File name	Description
app_config.h	Configures the DMX512 master application
bsp.c	This file implements the “board support package”, software functions specific for this MCU and the electronics (LEDs) on the PCB
bsp.h	Header file describing the exported functions by bsp.c
cr_startup_lpc11U1x.c	This file contains the Reset vector and exception vector table (needed for the LPCXpresso build only)
main.c	This file contains the application entry point, main loop and host message handling
my_printf.c	This file contains a lightweight printf function (only needed for the LPCXpresso build)
my_printf.h	Header file describing the exported functions by my_printf.h
startup_LPC11Uxx.s	Assembly file containing the Reset vector and exception vector table (needed for IAR build only)
uart.c	Software driver for the UART hardware block especially for DMX512 packet RTX
uart.h	Header file describing the exported functions by uart.c
wwdt.c	Software driver for the WDT hardware block
wwdt.h	Header file describing the exported functions by wwdt.c

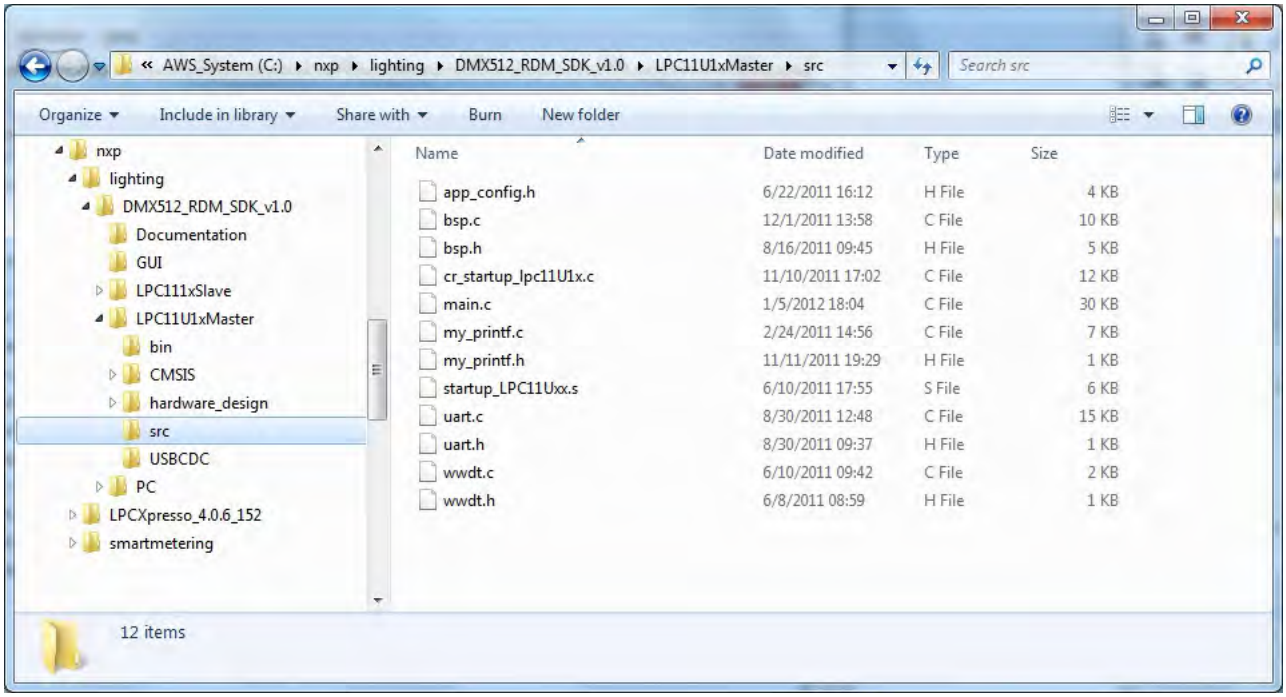

Fig 10. DMX512 master source code files

4.8 Building the software

The software tree includes project files for LPCXpresso v4.1.0_190. When using LPCXpresso for building the DMX512 master, use workspace location <your install path>\DMX512 and import existing project <your install path>\DMX512\LPC11U1xMaster and make sure to uncheck the checkbox "Copy projects into workspace".

The software tree also includes project files for the IAR Embedded Workbench for ARM v6.20.4. When the IAR workbench is installed, the project can be opened by double clicking the file ‘LPC11U1xDMXmaster.eww’.

The software can be configured via the source file ‘app_config.h’, which holds one configuration option as listed in Table 4.

Table 4. DMX512 master configuration options

Define	Description
LPC_CORE_CLOCKSPEED_HZ	Sets the CPU clock.

The CPU clock is configured at 48 MHz (a multiple of 12 MHz), which gives the performance needed for the DMX512 master.

Table 5 shows the firmware sizes (in bytes) of a RELEASE build of the DMX512 master.

Table 5. DMX512 slave firmware sizes

Firmware sizes for DMX512 master	IAR EWARM v6.20.4		LPCXpresso v4.1.0_190	
	Flash [Bytes]	RAM [Bytes]	Flash [Bytes]	RAM [Bytes]
Release build	10004 + 360	3860	10364	40 + 3308

5. Document management

5.1 Abbreviations

Table 6. Abbreviations

Acronym	Description
BSP	Board Support Package
CDC	Communications Device Class
CMSIS	Cortex Microcontroller Software Interface Standard
CPU	Central Processing Unit
CT	Counter Timer
GPIO	General Purpose Input/Output
HW	Hardware
IRQ	Interrupt Request
LED	Light Emitting Diode
MCU	Micro Controller Unit
NVM	Non Volatile Memory
PC	Personal Computer
PCB	Printed Circuit Board
PLASA	The lead international membership body for those who supply technologies and services to the event, entertainment and installation industries
PLL	Phase Locked Loop
RDM	Remote Device Management
RGB	Red Green Blue
RTX	Receive and Transmit
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
UID	Unique device ID
USB	Universal Serial Bus
WDT	Watchdog Timer
WWDT	Windowed Watchdog Timer

5.2 Referenced documents

Table 7. Referenced documents

Doc Title		Version	Author	Issue Date
[1]	ANSI E1.11 - Asynchronous Serial Data Transmission Standard for Controlling Lighting Equipment and Accessories	2008	ESTA	20081204
[2]	UM10462 - LPC11U1x User manual	Rev. 1	NXP	20110414
[3]	ANSI E1.20 - Remote Device Management Over DMX512 Networks	2010	PLASA	20110104
[4]	DMX USB Pro Widget API Specification http://www.enttec.com/?main_menu=Products&pn=70304&show=downloads	1.44	ENTTEC	20071016
[5]	AN11153 - DMX512/RDM slave using LPC111x	1	NXP	20120201

6. Appendix A: Testing the DMX512 master

6.1 USB power

Action	Power the DMX512 master (disconnected from DMX512 bus) via the USB connector.
Result	The BLUE ON LED must light up continuously. The LPC11U14 must start to run the firmware; the RED LED1 and the GREEN LED2 turn on. After the USB connection with the PC is established, the RED LED1 starts to blink with a 1 Hz frequency.

6.2 Reset button

Action	Press the reset button while the board is powered (and keep it pressed).
Result	The BLUE ON LED stays on. The RED LED1 stops blinking and doesn't light up. The GREEN LED2 turns off.

Action	Release the reset button.
Result	The LPC11U14 must start to run the firmware; the RED LED1 and the GREEN LED2 turn on. After the USB connection with the PC is established the RED LED1 starts to blink with a 1 Hz frequency.

6.3 DMX512 output

Action	Connect DMX512 master to DMX512 bus and start controlling the DMX bus with the DMX512 control program.
Result	The GREEN LED2 blinks with the refresh rate of the DMX512 packets put on the bus.

6.4 DMX512 input

Action	Connect a second DMX512 master to PC and DMX512 bus which carries DMX512 packets.
Result	The BLUE ON LED turns on. After the USB connection with the PC is established the RED LED1 starts to blink with a 1 Hz frequency. The GREEN LED2 blinks with the refresh rate of the DMX512 packets captured from the bus.

7. Appendix B: Flashing the DMX512 master

When using IAR Embedded Workbench for ARM v6.20.4 and a j-link JTAG / SWD +SWO probe, the firmware can be easily flashed in the LPC11U14 of the DMX512 master.

The IAR project also generates a binary file that can be flashed in the DMX512 master using an LPC-Link and LPCXpresso v4.1.0_190.

The steps for Windows7 are:

Step 1) Booting LPC-Link 'boot_LPC-link.bat'

Step 2) Running the flash programming utility 'flash_DMXmaster.bat'

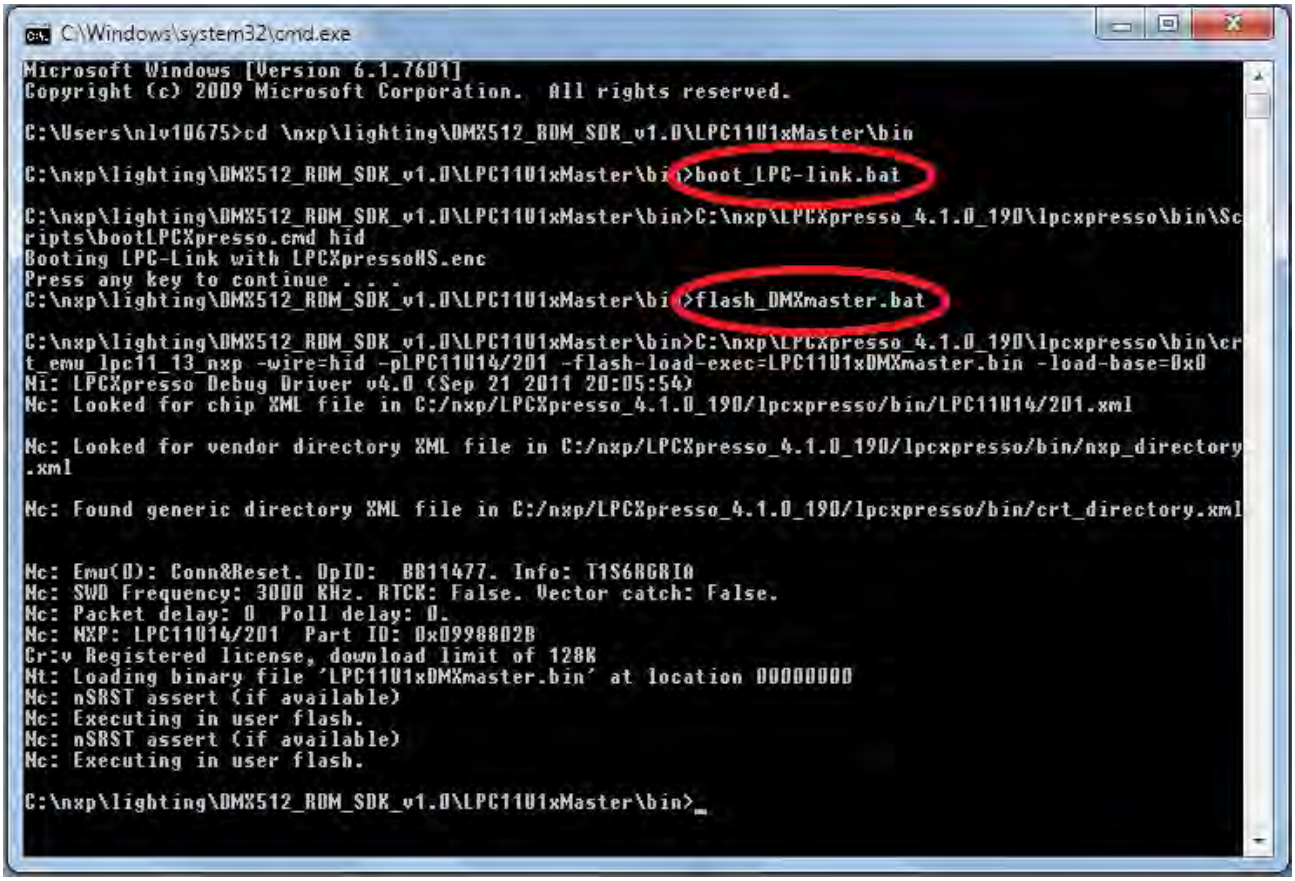

Fig 11. Flashing the DMX512 master

For help with the command line flash programming tool using Linux, go to:

http://support.code-red-tech.com/CodeRedWiki/CommandLineFlashProgramming

8. Legal information

8.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

8.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

9. List of figures

10. List of tables

11. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 1 February 2012
Document identifier: AN11154