

AN10717

DMX512 communication using the LPC2000

Rev. 01 — 1 July 2008

Application note

Document information

Info	Content
Keywords	LPC2148, ARM7, DMX512, USB to DMX512, DMX512 slave
Abstract	This application note demonstrates the use of a low cost ARM7 based NXP microcontroller as a DMX512 transmitter and receiver.

Revision history

Rev	Date	Description
01	20080701	Initial version.

Contact information

For additional information, please visit: <http://www.nxp.com>

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

1. Introduction

The DMX512 standard describes a method of digital data transmission between controllers and controlled equipment. It is designed to carry repetitive control data from a single controller to one or more receivers.

DMX512 is a unidirectional asynchronous serial communication protocol. There's no error checking or correction mechanism and there's no handshake between all receivers and the transmitter. This makes the protocol extremely simple, but also unsuitable for safety critical applications. The transmission rate is 250k Baud (11 bits data: 1 start bit, 8 data bits and 2 stop bits) over an RS-485 interface. The physical interface (like cables and connectors) is not discussed in this application note.

The transmitter is sending data in packets of up to 513 slots (see [Fig 1](#)). Each slot contains an 8-bit value, between 0 and 255. The first slot is a START Code, which defines the meaning of the information in the subsequent slots in the packet. The NULL Start Code is reserved for sending dimming data, where 0 means light off and 255 represents a maximum light intensity.

All receiver devices connected to the link choose one of the 512 slots (address selection) to extract the data for processing from each transmitted packet. The DMX512-A transmitter continuously repeats (at least once per second) the transmission of a packet as shown in [Table 1](#).

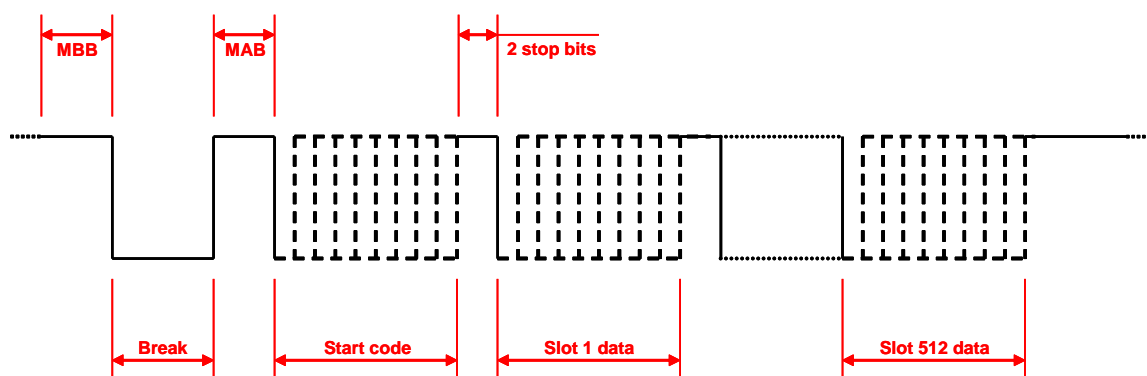


Fig 1. DMX512 Packet

Table 1. DMX512 Timing Values

Description	Min	Typical	Max	Unit
MBB – mark before break	0	-	< 1.00	µsec / s
Break	92	176	-	µsec
MAB – mark after break	12	-	< 1.00	µsec / s
Bit Time	3.92	4	4.08	µsec
DMX512 Packet	1204	-	< 1.00	µsec / s

2. DMX512 transmitter

The DMX transmitter described in this application note is in fact a USB to DMX512 protocol converter. It's a small board connected to the USB port of a PC running a simple GUI that can send a dimming value to one of 512 DMX slaves.

2.1 Hardware

For the design an LPC2141 microcontroller is used (see [Fig 2](#)) because of its on-chip USB interface (used to communicate with a PC GUI). UART1 of the LPC2141 is used for the DMX512 interface.

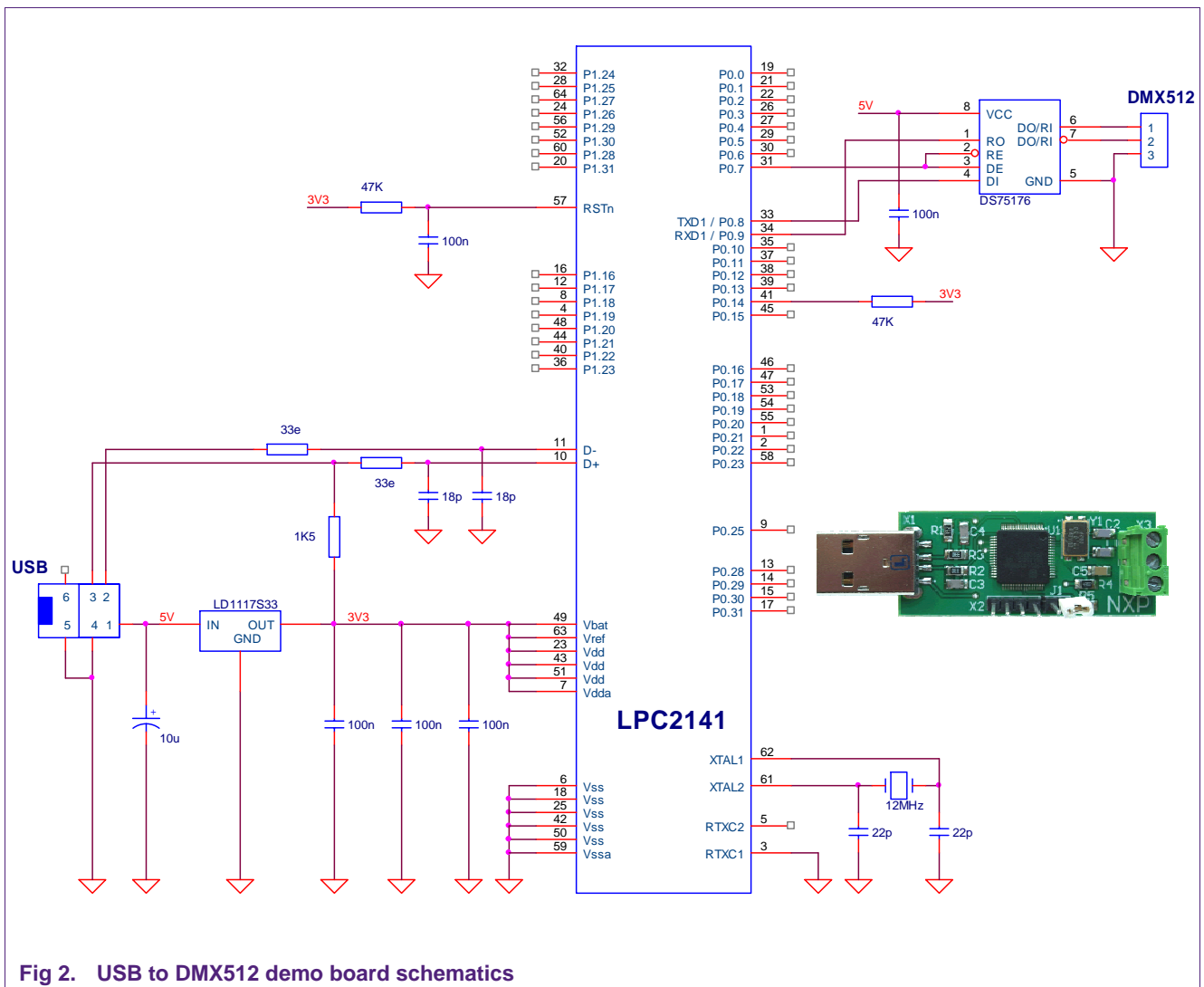


Fig 2. USB to DMX512 demo board schematics

2.2 Software

The example software is written in C language and compiled using Keil's uVision (ARM7 RealView, V3.1) free demo compiler. It performs following main tasks:

- Initialization: for LPC2141 configuration the standard startup code from Keil was used and set as CCLK = PCLK = 60 MHz
- USB (HID class) interface for receiving slave number and dimming data. The USB modules from Keil's HID example were used (not listed in this application note)
- Use Timer 1 to generate a system-interrupt every 10 milliseconds and a timing event every 200 milliseconds (see timer1.c module listed below)
- Sending of a DMX512 packet every 200 milliseconds. This part consists of three modules (main.c – dmx.c – uart1.c), all listed below

2.2.1 main.c

```

1  #include <LPC214x.H>                                // LPC214x definitions
2
3  void SetOutReport(unsigned char *rep)                 // OutReport received from USB host
4  {
5      unsigned short i;
6
7      i = (rep[0] * 100) + rep[1];                      // First 2 bytes are slave nr: 1-512
8      DMX_buf[i] = rep[2];                             // Third byte is dim value
9  }
10
11 int main (void)
12 {
13     USB_Init();                                       // USB Initialization
14     USB_Connect(TRUE);                               // USB Connect
15     DMX_Init();
16     T1_Init();
17
18     while(1)
19     {
20         if (f_200ms)                                // every 200 msec . . .
21         {
22             f_200ms = 0;
23             DMX_SendPacket();                         // DMX512 send data to slaves
24         }
25     }
26 }
```

2.2.2 timer1.c

```

1  #include <LPC214x.h>
2
3  char f_10ms = 0;
4  char f_200ms = 0;
5
6  __irq void T1_Isr(void)                             // Timer 1 ISR every 10 msec
7  {
8      static unsigned char cnt = 0;
9
10     f_10ms = 1;                                       // toggles every 10 mseconds
11
12     if (++cnt > 20)
```

```

13     {
14         cnt = 0;
15         f_200ms = 1;           // toggles every 200 mseconds
16     }
17     T1IR = 0x01;               // reset interrupt flag
18     VICVectAddr = 0;           // reset VIC
19 }
20
21 void T1_Init(void)
22 {
23     VICVectAddr2 = (unsigned int) &T1_Isr;
24     VICVectCntl2 = 0x25;       // Channel2 on Source#5 ... enabled
25     VICIntEnable |= 0x20;      // Channel#5 is the Timer 1
26
27     T1MR0 = 600000;            // = 10 msec / 16,67 nsec
28     T1MCR = 3;                 // Interrupt on MR0, reset TC on match
29     T1TC = 0;                  // reset Timer counter
30     T1TCR = 1;                 // enable Timer
31 }

```

2.2.3 dmx.c

```

1  #include <LPC214x.h>
2
3  unsigned char DMX_buf[513];
4
5  void DMX_SendPacket(void)
6  {
7      TOTC = 0;                 // reset Timer counter
8      TOIR = 0x01;              // reset interrupt flag
9      TOMR0 = 92;               // set match to 92 us
10     ULLCR = 0x47;              // 'break'
11     TOTCR = 1;                 // start timer 0
12     while ((TOIR & 0x01) == 0); // wait for timer match
13
14     TOTC = 0;                 // reset Timer counter
15     TOIR = 0x01;              // reset interrupt flag
16     TOMR0 = 12;               // set match to 12 us
17     ULLCR = 7;                // 'mark'
18     TOTCR = 1;                 // start timer 0
19     while ((TOIR & 0x01) == 0); // wait for timer match
20
21     UART1_Send(DMX_buf, 513); // send data packet to slaves
22 }
23
24 void DMX_Init(void)
25 {
26     int i;
27
28     for (i = 0; i < 513; i++) DMX_buf[i] = 0;
29
30     IODIR0 |= 0x00000080;      // P0.7 = DS75176 enable
31     IOSET0 |= 0x00000080;      // Tx enable high active
32     UART1_Init(250000);
33     TOPR = 60;                 // 60, timer runs at 60 MHz / 60 = 1 MHz
34     TOMCR = 7;                 // Int on MR0, reset and stop the timer
35 }

```

2.2.4 uart1.c

```

1  #include <LPC214x.H>
2
3  #define Fosc      12000000
4  #define Fpclk     60000000
5
6  unsigned int  txin;           // Next In Index
7  unsigned int  txout;          // Next Out Index
8  unsigned char *txbuf;         // pointer to Tx buffer
9
10 __irq void U1_Isr(void)
11 {
12     char i = 16;
13
14     if ((U1IIR & 0x0F) == 2)    // THRE Interrupt ?
15     {
16         while (i && txout)
17         {
18             U1THR = txbuf[txin++];
19             txout --;
20             i --;
21         }
22     }
23     VICVectAddr = 0;           // Acknowledge Interrupt
24 }
25
26 void UART1_Send(unsigned char *buf, unsigned int len)
27 {
28     char i = 16;
29
30     if (txout == 0)            // previous message send ?
31     {
32         txbuf = buf;           // copy buffer pointer
33         txout = len;
34         txin = 0;
35         while (i && txout)
36         {
37             U1THR = txbuf[txin++];
38             txout --;
39             i --;
40         }
41     }
42 }
43
44 void UART1_Init(unsigned int baudrate)
45 {
46     volatile char dummy;
47     unsigned int brd = (Fpclk / (baudrate << 4));
48
49     txin = 0;
50     txout = 0;
51     PINSEL0 |= 0x00050000;    // Select U1 RXD/TXD
52
53     U1FCR = 7;                // Enable and clear FIFO's
54     U1LCR = 0x87;              // Set DLAB and set word format to 8-N-2
55     U1DLL = (brd & 0xFF);      // Set baud rate dividers
56     U1DLM = (brd >> 8);        //
57     U1LCR = 7;                // Disable Divisor latch bit
58
59     VICVectAddr3 = (unsigned int) &U1_Isr;
60     VICVectCntl3 = 0x27;      // Channel2 on Source #7 ... enabled

```

```
61      VICIntEnable |= 0x00000080;           // Source #7 is UART1
62
63      dummy = U1IIR;                         // Read IrqID to get interrupts started
64      U1IER = 2;                             // Enable UART1 THRE Interrupt
65  }
```

2.3 GUI

A Windows® graphical user interface is available to control the USB to DMX512 demo board (see [Fig 3](#)). The program is called “USB-DMX.EXE” and is developed in Microsoft Visual Basic 2008 Express, so it needs the Microsoft .NET framework at your PC.

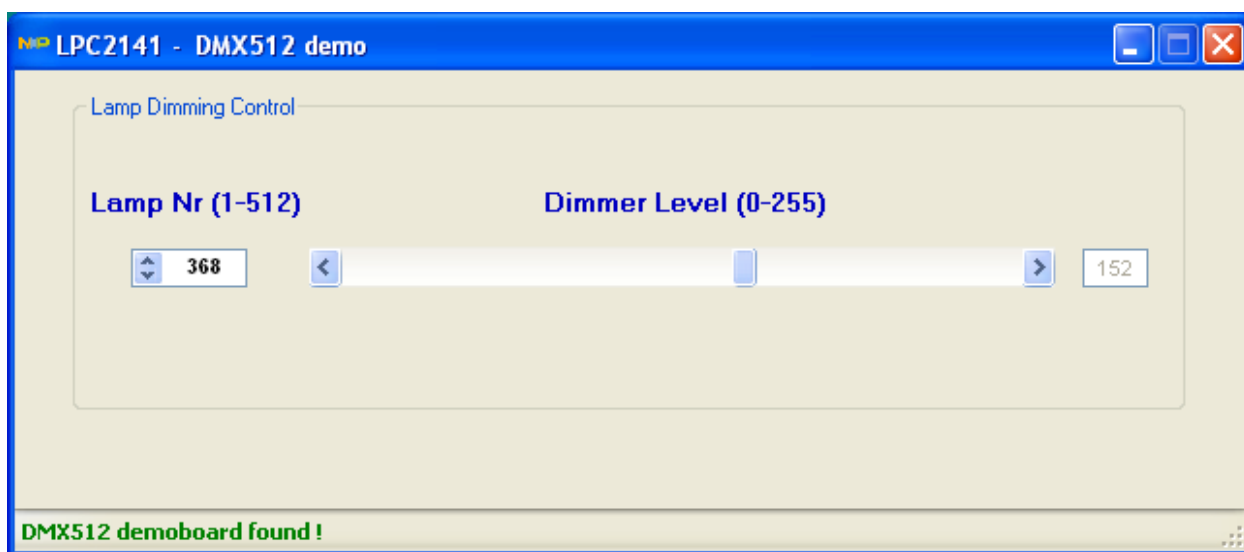


Fig 3. DMX512 - GUI

3. DMX512 receiver

3.1 Hardware

For the DMX512 receiver part an LPC2103 is used (see [Fig 4](#)). UART1 of the LPC2103 is used for the DMX512 interface. Received dimming data is output on ports P0.16 to P0.23 connected to a buffer and eight LEDs.

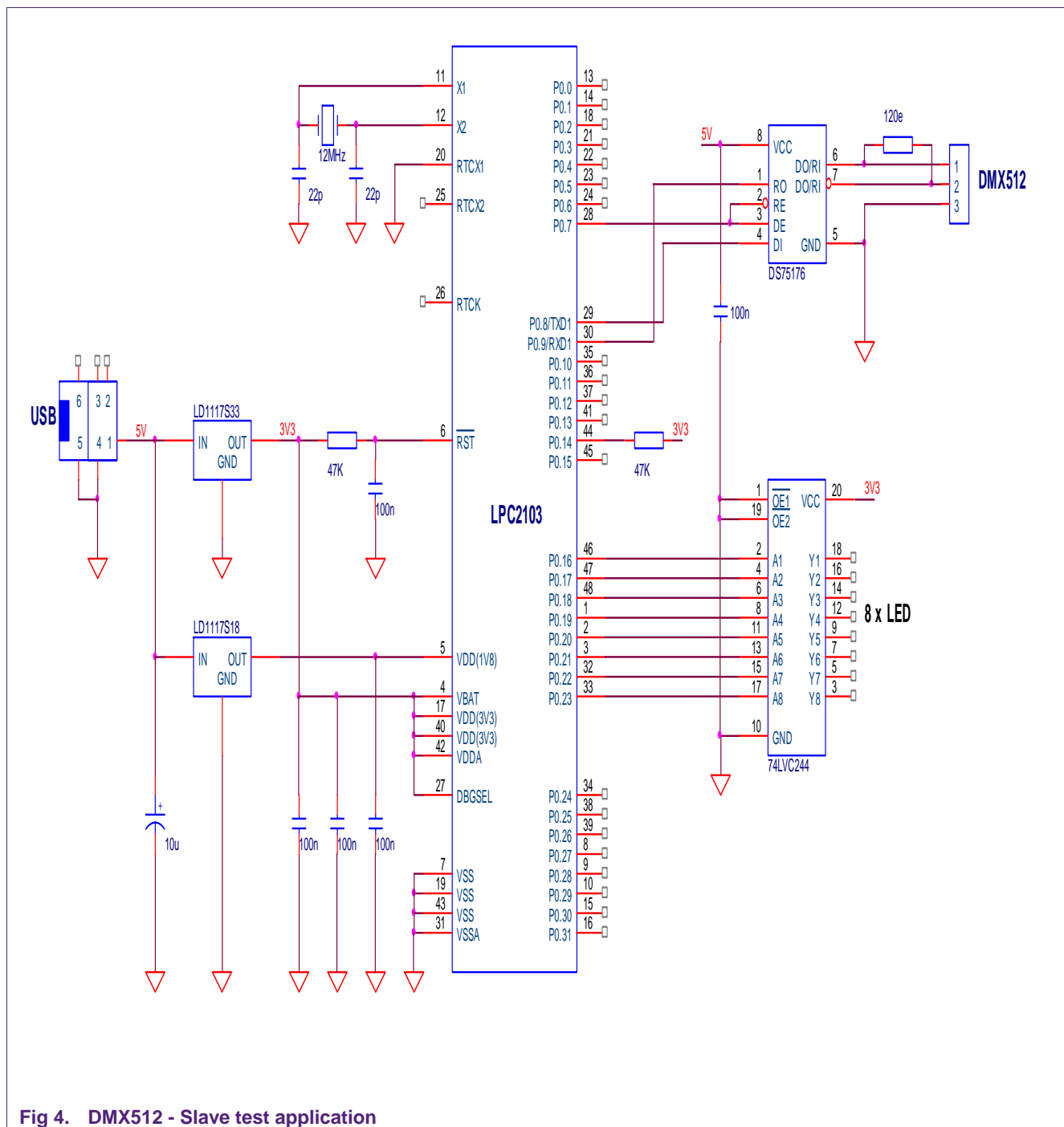


Fig 4. DMX512 - Slave test application

3.2 Software

The receiver example software is written in C language and compiled using Keil's uVision (ARM7 RealView, V3.1) free demo compiler. It performs following main tasks:

- Initialization: for LPC2103 configuration the standard startup code from Keil was used and set as CCLK = PCLK = 60 MHz
- Use Timer 1 to generate a system-interrupt every 10 milliseconds and a timing event every 200 milliseconds (see timer1.c module listed below), used to output the received slave data to 8 LEDs (see main.c module listed below)
- Receiving of DMX512 packets (uart1.c module listed below)

3.2.1 main.c

```

1  #include <LPC2103.H>                                // LPC2103 definitions
2
3  #define DMX_NR  368                                  // My own DMX slave number
4
5  int main(void)
6  {
7      UART1_Init(250000);
8      T1_Init();
9
10     IODIR |= 0x00FF0000;                             // LEDs at P0.16 - 23
11     IOCLR |= 0x00FF0000;                             // turn LEDs off
12
13     while(1)
14     {
15         if (f_200ms)
16         {
17             f_200ms = 0;
18             IOPIN = (IOPIN & 0xFF00FFFF) | (DMX_buf[DMX_NR] << 16);
19         }
20     }
21 }
```

3.2.2 timer1.c

```

1  #include <LPC2103.H>                                // LPC2103 definitions
2
3  char f_10ms = 0;
4  char f_200ms = 0;
5
6  __irq void T1_Isr(void)                             // Timer 1 ISR every 10 msec
7  {
8      static unsigned char cnt = 0;
9
10     f_10ms = 1;                                       // toggles every 10 mseconds
11
12     if (++cnt > 20)
13     {
14         cnt = 0;
15         f_200ms = 1;                                 // toggles every 200 mseconds
16     }
17     T1IR = 0x01;                                     // reset interrupt flag
18     VICVectAddr = 0;                                 // reset VIC
19 }
20
```

```

21 void Tl_Init(void)
22 {
23     VICVectAddr1 = (unsigned int) &Tl_Isr;
24     VICVectCntl1 = 0x25; // Channel1 on Source#5 ... enabled
25     VICIntEnable |= 0x20; // Channel#5 is the Timer 1
26
27     TlMR0 = 60000-1; // every 10 msec
28     TlMCR = 3; // Interrupt on MR0, reset timer
29     TlTTC = 0; // reset Timer counter
30     TlTCR = 1; // enable Timer
31 }

```

3.2.3 uart1.c

```

1  #define Fpclk 6000000
2
3  unsigned int rxin = 0; // buffer index
4  unsigned char DMX_buf[513], dummy;
5
6  __irq void U1_Isr(void)
7  {
8      static int iid;
9
10     while (((iid = U1IIR) & 1) == 0)
11     {
12         if ((iid & 0x0E) == 6) // Receive Line Status
13         {
14             U1LSR; U1RBR; // read LSR to clear the interrupt
15             rxin = 0;
16         }
17         Else // Receive Data Available
18         do
19         {
20             DMX_buf[rxin] = U1RBR;
21             if (rxin < 513)
22                 rxin ++;
23             } while (U1LSR & 1); // receive data ready
24         }
25         VICVectAddr = 0; // Acknowledge Interrupt
26     }
27
28 void UART1_Init(unsigned int baudrate)
29 {
30     unsigned int brd = (Fpclk / (baudrate << 4));
31
32     IODIR |= 0x00000080; // P0.7 = DS75176 enable
33     IOCLR |= 0x00000080; // Rx enable, low active
34     PINSEL0 |= 0x00050000; // Select U1 RXD/TXD
35
36     U1FCR = 0x87; // En and clear FIFO's, trigger level 2
37     U1LCR = 0x87; // Set DLAB and set word format to 8-N-2
38     U1DLL = (brd & 0xFF); // Set baud rate dividers
39     U1DLM = (brd >> 8); //
40     U1LCR = 7; // Disable Divisor latch bit
41
42     VICVectAddr0 = (unsigned int) &U1_Isr;
43     VICVectCntl0 = 0x27; // Channel0 on Source #7 ... enabled
44     VICIntEnable |= 0x00000080; // Source #7 is UART1
45     dummy = U1IIR; // Read IrqID to get interrupts started
46     U1IER = 5; // Enable U1 RBR + Rx line status Int
47 }

```

4. Legal information

4.1 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.

4.2 Disclaimers

General — 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.

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 medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for 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.

4.3 Trademarks

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

5. Contents

1. Introduction3

2. DMX512 transmitter.....4

2.1 Hardware.....4

2.2 Software5

2.2.1 main.c.....5

2.2.2 timer1.c5

2.2.3 dmx.c.....6

2.2.4 uart1.c7

2.3 GUI8

3. DMX512 receiver.....9

3.1 Hardware.....9

3.2 Software10

3.2.1 main.c.....10

3.2.2 timer1.c10

3.2.3 uart1.c11

4. Legal information12

4.1 Definitions.....12

4.2 Disclaimers.....12

4.3 Trademarks12

5. Contents.....13

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.



© NXP B.V. 2008. All rights reserved.

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

Date of release: 1 July 2008
Document identifier: AN10717_1