AN10760

USB - DALI master using the LPC2141

Rev. 01 — 1 November 2008 Application note

Document information

Info	Content
Keywords	LPC2148, ARM7, DALI, USB to DALI, DALI master
Abstract	This application note describes the design of a DALI (Digitally Addressable Lighting Interface) master unit, based on the LPC2141 microcontroller from NXP Semiconductors with a USB connection to a Host (PC) running a Graphical User Interface.

Revision history

Rev	Date	Description
01	20081101	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 international standard (IEC929) DALI-bus communication protocol is intended for use in digital TL-ballast intelligent lighting systems. In a typical application, a DALI-bus consists of one controller (master), and multiple slaves (normally TL-ballasts). It can control up to 64 different slaves (ballasts) within the same control system. It's possible to transmit commands to single ballasts or to a group of ballasts.

The DALI bus consists of two wires, providing a differential signal. Data is transmitted in frames. There are two different frame types: a "forward" frame (2 bytes, sent by the master to the slaves), and a "backward" frame (1 byte, sent by a slave to the master, possibly containing status info). DALI uses a bi-phase (also called Manchester) encoding, which means that the data is transmitted using the edges of the signal. A rising edge indicates a '1'; a falling edge indicates a '0' (see Fig 1).

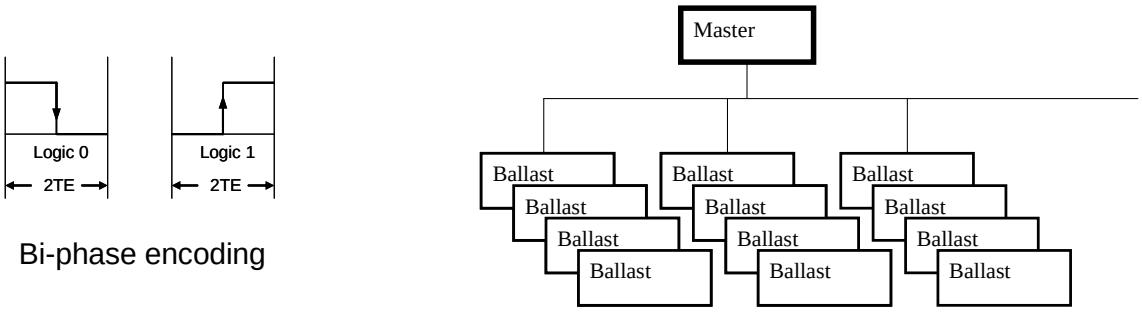

Fig 1. DALI bi-phase encoding and typical DALI bus network structure

Every bit takes two periods TE. The defined bit rate of DALI is 1200 bps. So, 1 bit period (2TE) is ~834 µsec. A frame is started by a start bit, and ends with two high-level stop bits (no change of phase). Data is transmitted with the MSB first. Between frames, the bus is in idle (high) state (see Fig 2).

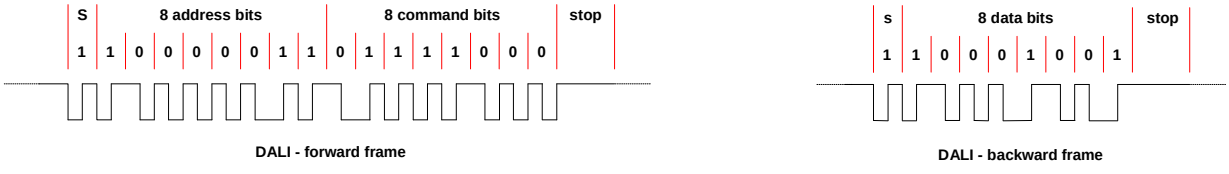

Fig 2. DALI forward and backward frame format

Additional protocol timing requirements for transmission are (see Fig 3):

- The settling time between two subsequent forward frames shall be at least 9.17 ms. This means that 4 forward frames with accompanying periods of 9.17 ms shall fit exactly in 100 ms.
- The settling time between forward and backward frames (transition from forward to backward) shall be between 2.92 and 9.17 ms. After sending the forward frame, the

master unit will wait for 9.17 ms. If no backward frame has been started after 9.17 ms this is interpreted as “no answer” from slave.

- The settling time between backward and forward frames (transition from backward to forward) shall be at least 9.17 ms.

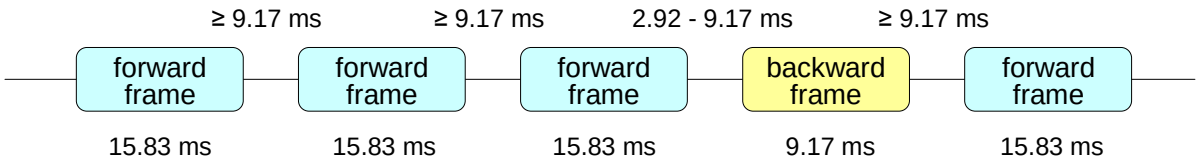

Fig 3. Example of frame repetition time

Every DALI slave is able to react to a short address, 16 group addresses and broadcast. The following addressing scheme is used.

Type of addresses:	**address byte:**
Short or group address	**Y**AAAAAA**S**
Short addresses (64)	0**AAAAAA**S
Group addresses (16)	100**AAAA**S
Broadcast	**1111111**S
Special command	**101**CCCC1
Special command	**110**CCCC1

S: selector bit: S = ‘0’ direct arc power level following
S = ‘1’ command following

Y: short- or group address: Y = ‘0’ short address
Y = ‘1’ group address or broadcast

A: significant address bit

C: significant command bit

Table 1 contains a complete summary of the DALI command set. Basically there are four types of commands (forward frames):

1. Direct / Indirect arc power control commands – used to set ballast power level.
2. Configuration commands – configures the ballast (for example: add to a group or store level). Command must be repeated within 100 ms, otherwise it’s ignored.
3. Query commands – ask slave (ballast) for status information (for example: power level or version number). The slave can send a backward frame.
4. Special commands – used to initialize and setup the ballast, some must be repeated within 100 ms, and some require an answer from the slave. Most commands are only processed within 15 minutes after an “INITIALIZE” command is received.

Table 1. DALI Command Set Summary

Number	Command Code	Repeat < 100 ms	Answer Slave	Command Name
-	YAAA AAA0 XXXX XXXX	no	no	DIRECT ARC POWER CONTROL
0	YAAA AAA1 0000 0000	no	no	OFF
1	YAAA AAA1 0000 0001	no	no	UP
2	YAAA AAA1 0000 0010	no	no	DOWN
3	YAAA AAA1 0000 0011	no	no	STEP UP
4	YAAA AAA1 0000 0100	no	no	STEP DOWN
5	YAAA AAA1 0000 0101	no	no	RECALL MAX LEVEL
6	YAAA AAA1 0000 0110	no	no	RECALL MIN LEVEL
7	YAAA AAA1 0000 0111	no	no	STEP DOWN AND OFF
8	YAAA AAA1 0000 1000	no	no	ON AND STEP UP
9-15	YAAA AAA1 0000 1XXX			RESERVED
16 - 31	YAAA AAA1 0001 XXXX	no	no	GO TO SCENE
32	YAAA AAA1 0010 0000	yes	no	RESET
33	YAAA AAA1 0010 0001	yes	no	STORE ACTUAL LEVEL IN THE DTR
34 - 41	YAAA AAA1 0010 XXXX			RESERVED
42	YAAA AAA1 0010 1010	yes	no	STORE THE DTR AS MAX LEVEL
43	YAAA AAA1 0010 1011	yes	no	STORE THE DTR AS MIN LEVEL
44	YAAA AAA1 0010 1100	yes	no	STORE THE DTR AS SYSTEM FAILURE LEVEL
45	YAAA AAA1 0010 1101	yes	no	STORE THE DTR AS POWER ON LEVEL
46	YAAA AAA1 0010 1110	yes	no	STORE THE DTR AS FADE TIME
47	YAAA AAA1 0010 1111	yes	no	STORE THE DTR AS FADE RATE
48 - 63	YAAA AAA1 0011 XXXX			RESERVED
64 - 79	YAAA AAA1 0100 XXXX	yes	no	STORE THE DTR AS SCENE
80 - 95	YAAA AAA1 0101 XXXX	yes	no	REMOVE FROM SCENE
96 - 111	YAAA AAA1 0110 XXXX	yes	no	ADD TO GROUP
112 -127	YAAA AAA1 0111 XXXX	yes	no	REMOVE FROM GROUP
128	YAAA AAA1 1000 0000	yes	no	STORE DTR AS SHORT ADDRESS
129 -143	YAAA AAA1 1000 XXXX			RESERVED
144	YAAA AAA1 1001 0000	no	yes	QUERY STATUS
145	YAAA AAA1 1001 0001	no	yes	QUERY BALLAST
146	YAAA AAA1 1001 0010	no	yes	QUERY LAMP FAILURE
147	YAAA AAA1 1001 0011	no	yes	QUERY LAMP POWER ON
148	YAAA AAA1 1001 0100	no	yes	QUERY LIMIT ERROR
149	YAAA AAA1 1001 0101	no	yes	QUERY RESET STATE
150	YAAA AAA1 1001 0110	no	yes	QUERY MISSING SHORT ADDRESS
151	YAAA AAA1 1001 0111	no	yes	QUERY VERSION NUMBER
152	YAAA AAA1 1001 1000	no	yes	QUERY CONTENT DTR

Number	Command Code	Repeat < 100 ms	Answer Slave	Command Name
153	YAAA AAA1 1001 1001	no	yes	QUERY DEVICE TYPE
154	YAAA AAA1 1001 1010	no	yes	QUERY PHYSICAL MINIMUM LEVEL
155	YAAA AAA1 1001 1011	no	yes	QUERY POWER FAILURE
156 - 159	YAAA AAA1 1001 11XX			RESERVED
160	YAAA AAA1 1010 0000	no	yes	QUERY ACTUAL LEVEL
161	YAAA AAA1 1010 0001	no	yes	QUERY MAX LEVEL
162	YAAA AAA1 1010 0010	no	yes	QUERY MIN LEVEL
163	YAAA AAA1 1010 0011	no	yes	QUERY POWER ON LEVEL
164	YAAA AAA1 1010 0100	no	yes	QUERY SYSTEM FAILURE LEVEL
165	YAAA AAA1 1010 0101	no	yes	QUERY FADE TIME / FADE RATE
166 - 175	YAAA AAA1 1010 XXXX			RESERVED
176 - 191	YAAA AAA1 1011 XXXX	no	yes	QUERY SCENE LEVEL (SCENES 0-15)
192	YAAA AAA1 1100 0000	no	yes	QUERY GROUPS 0-7
193	YAAA AAA1 1100 0001	no	yes	QUERY GROUPS 8-15
194	YAAA AAA1 1100 0010	no	yes	QUERY RANDOM ADDRESS (H)
195	YAAA AAA1 1100 0011	no	yes	QUERY RANDOM ADDRESS (M)
196	YAAA AAA1 1100 0100	no	yes	QUERY RANDOM ADDRESS (L)
197 - 223	YAAA AAA1 110X XXXX			RESERVED
224 - 255	YAAA AAA1 11XX XXXX			APPLICATION EXTENDED COMMANDS
256	1010 0001 0000 0000	no	no	TERMINATE
257	1010 0011 XXXX XXXX	no	no	DATA TRANSFER REGISTER (DTR)
258	1010 0101 XXXX XXXX	yes	no	INITIALISE
259	1010 0111 0000 0000	yes	no	RANDOMISE
260	1010 1001 0000 0000	no	yes	COMPARE
261	1010 1011 0000 0000	no	no	WITHDRAW
262	1010 1101 0000 0000			RESERVED
263	1010 1111 0000 0000			RESERVED
264	1011 0001 HHHH HHHH	no	no	SEARCHADDRH
265	1011 0011 MMMM MMMM	no	no	SEARCHADDRM
266	1011 0101 LLLL LLLL	no	no	SEARCHADDRL
267	1011 0111 0AAA AAA1	no	no	PROGRAM SHORT ADDRESS
268	1011 1001 0AAA AAA1	no	yes	VERIFY SHORT ADDRESS
269	1011 1011 0000 0000	no	yes	QUERY SHORT ADDRESS
270	1011 1101 0000 0000	no	no	PHYSICAL SELECTION
271	1011 1111 XXXX XXXX			RESERVED
272	1100 0001 XXXX XXXX	no	no	ENABLE DEVICE TYPE X
273 - 287	110X XXX1 XXXX XXXX			RESERVED

2. USB - DALI master

The DALI master described in this application note is in fact a USB to DALI protocol converter. It's a simple design that connects to the USB port of a PC running a simple GUI that can send DALI commands and receive slave answers.

2.1 Hardware

For the design an LPC2141 microcontroller is selected (see Fig 4) because of its on-chip USB interface (used to communicate with a PC - GUI). To send DALI commands a general purpose output pin (P0.28) is used. For the reception of slave backward frames Timer 0, capture 0 (P0.30), together with a general purpose input pin (P0.29) are used.

The boost converter is needed to generate 12V / 250 mA out of the USB 5V bus supply. This part of the design, as well as the DALI hardware bus driver logic, is not handled in this application note.

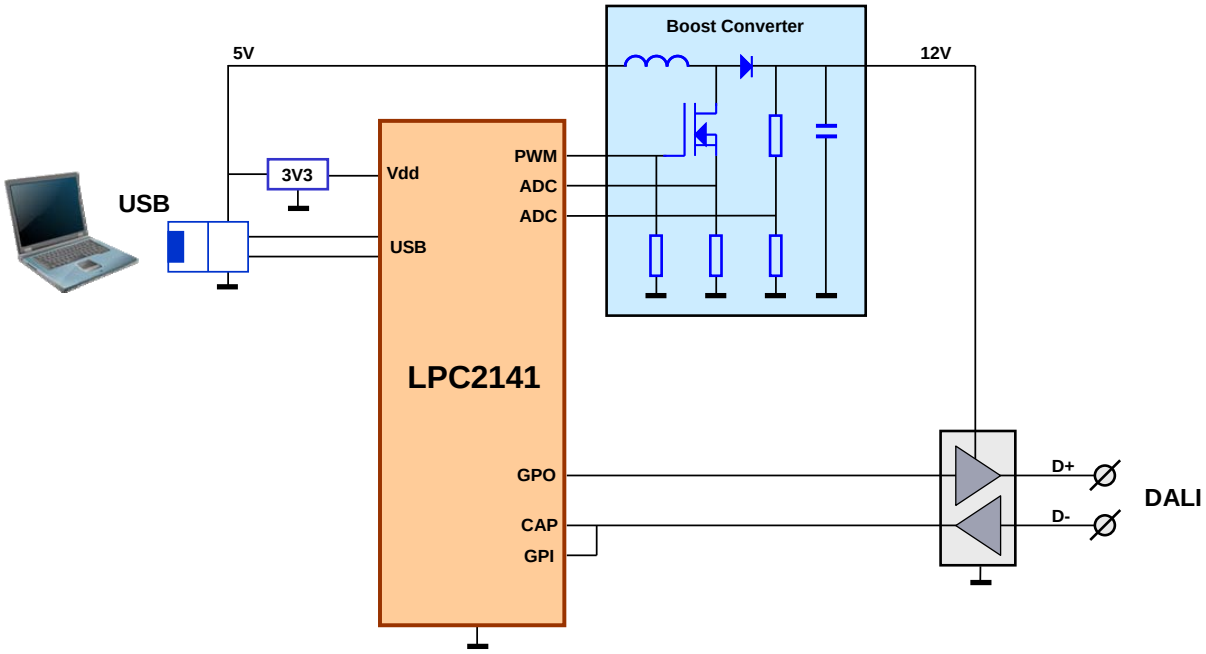

Fig 4. USB to DALI master simplified block diagram

2.2 Software

TRANSMITTING A DALI - MESSAGE

Sending a frame is relatively easy. The implementation uses Timer 0 interrupt every period 'TE' to generate the DALI message. Sending a single bit via bi-phase encoding requires two interrupts, in order to produce a good transition. A '1' is sent by pulling down the line for one period, followed by releasing it for one period. Sending a '0' is exactly the opposite. A position counter is used to keep track of which bit is being transmitted. The counter runs at twice the bit frequency (just like the interrupt), so bit 0 can be used to detect whether the first or the second period of this bit is to be transmitted.

DECODING A DALI - MESSAGE

The easiest method to decode DALI messages is to detect the edges of the signal and measure the time between these edges. Using a timer capture input of the LPC2141 this is easy to accomplish, because the input can capture and generate an interrupt at both rising and falling edge. At the falling edge the pulse ‘high time’ is captured and stored. At a rising edge the pulse ‘low time’ is captured, and the received bit(s) is decoded.

The example software is written in C language and compiled using Keil’s uVision (ARM7 RealView, V3.2) 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 the DALI command and return of the slave’s answer. The USB modules from Keil’s HID example were used (not listed in this application note)
- DALI driver: use Timer 0 match 0 for sending DALI forward frames. Before a frame is send the driver determines whether or not the command should be repeated (resend) within 100 msec). Use timer 0 capture 0 to receive DALI backward frames (see **dali_drv.c** module listed below)
- Main: check and clear event flags and take action (see module **main.c** listed below)

2.2.1 main.c

```
1     #include <LPC214x.H>                       // LPC214x definitions
2     #include "global.h"
3
4     WORD forward  = 0;                        // DALI forward frame
5     BYTE answer   = 0;                        // DALI slave answer
6     BYTE f_dalitx = 0;
7     BYTE f_dalirx = 0;
8
9     void GetInReport(BYTE *rep)               // USB host is asking for an InReport
10    {
11        if (f_dalirx)                         // answer from slave received ?
12        {
13            rep[0] = 1;                       // send answer
14            rep[1] = answer;
15        }
16        else
17            rep[0] = 0;                       // no answer
18    }
19
20    void SetOutReport(BYTE *rep)              // OutReport received from USB host
21    {
22        forward  = (rep[0] << 8) | rep[1];
23        f_dalitx = 1;                         // set DALI send flag
24    }
25
26    int main(void)
27    {
28        USB_Init();                           // USB Initialization
29        USB_Connect(TRUE);                    // USB Connect
30        DALI_Init();
31
```

```
32      while (1)
33      {
34          if (f_dalitx)                           // flag set from USB or the DALI module
35          {
36              f_dalitx = 0;                       // clear DALI send flag
37              f_dalirx = 0;                       // clear DALI receive (answer) flag
38              DALI_Send();                        // DALI send data to slave(s)
39          }
40      }
41  }
```

2.2.2 dali_drv.c

```
42  /*****************************************************************************
43  ;
44  ;  DALI master
45  ;
46  ;  For transmit, this module uses GPIO - P0.28 (DALI send pin)
47  ;  DALI forward frame format:
48  ;
49  ;      | S |        8 address bits         |        8 command bits         | stop  |
50  ;      | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |   |   |
51  ;
52  ;   ---+ +-+ +---+ +-+ +-+ +-+ +-+   +-+ +---+   +-+ +-+ +-+ +---+ +-+ +-+ +------------
53  ;      | | | |   | | | | | | | | |   | | |   |   | | | | | | |   | | | | | |
54  ;      +-+ +-+   +-+ +-+ +-+ +-+ +---+ +-+   +---+ +-+ +-+ +-+   +-+ +-+ +-+
55  ;
56  ;      |2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|  4TE  |
57  ;
58  ;
59  ;  For receive, this module uses T0-CAP0 input (capture and interrupt on both edges)
60  ;  CAP0.0 (P0.30) is connected to P0.29 (to check high / low level by software)
61  ;  DALI slave backward frame format:
62  ;
63  ;                    | S |         8 data bits           | stop  |
64  ;                    | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |   |   |
65  ;
66  ;    +---------------+ +-+ +---+ +-+ +-+ +-+ +-+   +-+ +------------
67  ;    |               | | | |   | | | | | | | | |   | | |
68  ;   -+               +-+ +-+   +-+ +-+ +-+ +-+ +---+ +-+
69  ;
70  ;    |4 + 7 to 22 TE |2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|  4TE  |
71  ;
72  ;  2TE = 834 usec (1200 bps)
73  ;
74  *****************************************************************************/
75  #include <LPC214x.h>                            // LPC21xx definitions
76  #include "global.h"
77
78  #define INITIALISE     0xA500                   // command starting initialization mode
79  #define RANDOMISE      0xA700                   // command generating a random address
80
81  #define TE             834/2                    // half bit time = 417 usec
82  #define MIN_TE         TE - 60                  // minimum half bit time
83  #define MAX_TE         TE + 60                  // maximum half bit time
84  #define MIN_2TE        2*TE - 60                // minimum full bit time
85  #define MAX_2TE        2*TE + 60                // maximum full bit time
86
87  static int  low_time;                           // captured puls low time
88  static int  high_time;                          // captured puls high time
89
```

```
90    static BYTE value;                              // used for dali send bit
91    static BYTE position;                           // keeps track of sending bit position
92    static BYTE previous;                           // previous received bit
93    static WORD frame;                              // holds received slave backward frame
94    static BYTE f_repeat;                           // flag command shall be repeated
95    static BYTE f_busy;                             // flag DALI transfer busy
96
97
98    static void DALI_Shift_Bit(BYTE val)
99    {
100       if (frame & 0x100)                          // frame full ?
101           frame = 0;                              // yes, ERROR
102       else
103           frame = (frame << 1) | val;             // shift bit
104   }
105
106   /*************************************************************************
107   ; DALI_Decode (we only take action at a rising edge)
108   ;
109   ; Half(prev) Bit   Low Time        High Time      Action     New Half Bit
110   ; ---------------------------------------------------------------------
111   ;     0               0               0          Shift 0         0
112   ;     0               0               1          -ERROR-         *
113   ;     0               1               0          Shift 0,1       1
114   ;     0               1               1          -ERROR-         *
115   ;     1               0               0          Shift 1         1
116   ;     1               0               1          Shift 0         0
117   ;     1               1               0          -ERROR-         *
118   ;     1               1               1          Shift 0,1       1
119   ;
120   *************************************************************************/
121   static void DALI_Decode(void)
122   {
123     BYTE action;
124
125       action = previous << 2;
126
127       if ((high_time > MIN_2TE) && (high_time < MAX_2TE))
128           action = action | 1;                    // high_time = long
129       else if (!((high_time > MIN_TE) && (high_time < MAX_TE)))
130       {
131           frame = 0;                              // DALI ERROR
132           return;
133       }
134
135       if ((low_time > MIN_2TE) && (low_time < MAX_2TE))
136           action = action | 2;                    // low_time = long
137       else if (!((low_time > MIN_TE) && (low_time < MAX_TE)))
138       {
139           frame = 0;                              // DALI ERROR
140           return;
141       }
142
143       switch (action)
144       {
145         case 0:  DALI_Shift_Bit(0);               // short low, short high, shift 0
146                  break;
147         case 1:  frame = 0;                       // short low, long high, ERROR
148                  break;
149         case 2:  DALI_Shift_Bit(0);               // long low, short high, shift 0,1
150                  DALI_Shift_Bit(1);
151                  previous = 1;                    // new half bit is 1
152                  break;
```

```
153         case 3:  frame = 0;                         // long low, long high, ERROR
154                  break;
155         case 4:  DALI_Shift_Bit(1);                 // short low, short high, shift 1
156                  break;
157         case 5:  DALI_Shift_Bit(0);                 // short low, long high, shift 0
158                  previous = 0;                      // new half bit is 0
159                  break;
160         case 6:  frame = 0;                         // long low, short high, ERROR
161                  break;
162         case 7:  DALI_Shift_Bit(0);                 // long low, long high, shift 0,1
163                  DALI_Shift_Bit(1);
164         default: break;                             // invalid
165       }
166   }
167
168   __irq void DALI_Isr(void)
169   {
170       T0TC = 0;                                     // reset timer
171
172       if (T0IR & 1)                                 // match 0 interrupt for DALI send
173       {
174           if (value)
175               IOSET0 |= 0x10000000;                 // DALI output pin high
176           else
177               IOCLR0 |= 0x10000000;                 // DALI output pin low
178
179           if (position == 0)                        // 0TE second half of start bit = 1
180           {
181               value = 1;
182           }
183           else if (position < 33)                   // 1TE - 32TE, so address + command
184           {
185               value = (forward >> ((32 - position)/2)) & 1;
186               if (position & 1)
187                   value = !value;                   // invert if first half of data bit
188           }
189           else if (position == 33)                  // 33TE start of stop bit (4TE)
190           {                                         // and start of min settling time (7TE)
191               value = 1;
192           }
193           else if (position == 44)                  // 44TE, end stop bits + settling time
194           {                                         // receive slave answer, timeout
195               T0MR0 = 9174;                         // of 22TE = 9,174 msec
196               T0CCR = 0x0007;                       // enable rx, capture on both edges
197           }
198           else if (position == 45)                  // end of transfer
199           {
200               T0TCR = 2;                            // stop and reset timer
201               if (frame & 0x100)                    // backward frame (answer) completed ?
202               {
203                    answer = (BYTE)frame;            // OK ! save answer
204                    f_dalirx = 1;                    // and set flag to signal application
205               }
206               frame  = 0;                           // reset receive frame
207               f_busy = 0;                           // end of transmission
208               if (f_repeat)                         // repeat forward frame ?
209                   f_dalitx = 1;                     // yes, set flag to signal application
210           }
211           position++;
212           T0IR = 0x01;                              // clear MR0 interrupt flag
213       }
214
215
```

```
216     else                                    // capture interrupt for DALI receive
217     {
218         if (IO0PIN & 0x20000000)            // check rising or falling edge P0.29
219         {
220             if (frame != 0)                 // not first pulse ?
221             {
222                 low_time = T0CR0;           // rising, so capture low time
223                 DALI_Decode();              // decode received bit
224             }
225             else
226             {
227                 previous = 1;               // first pulse, so shift 1
228                 DALI_Shift_Bit(1);
229             }
230         }
231         else
232             high_time = T0CR0;              // falling, so capture high time
233
234         T0IR = 0x10;                        // reset interrupt flag
235     }
236     VICVectAddr = 0;                        // Ack interrupt by resetting VIC
237 }
238
239 void DALI_Send(void)
240 {
241     if (f_repeat)                           // repeat last command ?
242     {
243         f_repeat = 0;
244     }
245     else if ((forward & 0xE100) == 0xA100 || (forward & 0xE100) == 0xC100)
246     {
247         if ((forward & 0xFF00) == INITIALISE || forward == RANDOMISE)
248             f_repeat = 1;                   // special command repeat < 100 ms
249     }
250     else if ((forward & 0x1FF) >= 0x120 && (forward & 0x1FF) <= 0x180)
251     {
252         f_repeat = 1;                       // config. command repeat < 100 ms
253     }
254     while (f_busy) ;                        // Wait until dali port is idle
255
256     frame    = 0;
257     value    = 0;                           // first half of start bit = 0
258     position = 0;
259     f_busy   = 1;                           // set transfer activate flag
260     T0MR0 = TE;                             // ~ 2400 Hz
261     T0CCR = 0x0000;                         // disable capture interrupt
262     T0MCR = 0x0003;                         // intr on MR0, reset timer on match 0
263     T0TC  = 0;                              // reset timer
264     T0TCR = 1;                              // enable timer
265 }
266
267 void DALI_Init(void)
268 {
269     VICVectAddr1  = (LONG) &DALI_Isr;
270     VICVectCntl1  = 0x24;                   // channel0 on Source#4 ... enabled
271     VICIntEnable |= 0x10;                   // channel#4 is the Timer 0
272
273     IODIR0  |= 0x10000000;                  // P0.28 = DALI send pin
274     IOSET0  |= 0x10000000;
275     PINSEL1 |= 0x30000000;                  // P0.30 as CAP0.0 = DALI receive pin
276     T0PR     = 60;                          // timer runs at 60 MHz / 60 = 1 MHz
277 }
```

3. DALI - GUI

A Windows® graphical user interface is available to control the USB to DALI demo board (see Fig 5). The program is called “USB-DALI.EXE” and is developed in Microsoft Visual Basic 2008 Express, so it needs the Microsoft .NET framework installed on the PC or laptop.

The program is very easy to understand. The bottom status bar indicates whether or not the USB–DALI demo board is connected. First a DALI address - and command field are filled in (any hexadecimal value between 0 and FF). Next the address and command are sent over the DALI bus as a forward frame by pushing the SEND button. Finally the ‘slave answer’ field shows ‘no answer’ or the received slave backward frame (hex byte 0 to FF).

Fig 5. DALI master – GUI

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

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

Date of release: 1 November 2008
Document identifier: AN10760_1