## 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.
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](image)

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](image)

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
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.

---

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:**

<table>
<thead>
<tr>
<th>Short or group address</th>
<th>address byte:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short addresses (64)</td>
<td>0AAAAAA S</td>
</tr>
<tr>
<td>Group addresses (16)</td>
<td>100AAAAA S</td>
</tr>
<tr>
<td>Broadcast</td>
<td>1111111 S</td>
</tr>
<tr>
<td>Special command</td>
<td>101CCCCC1</td>
</tr>
<tr>
<td>Special command</td>
<td>110CCCCC1</td>
</tr>
</tbody>
</table>

- **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

<table>
<thead>
<tr>
<th>Number</th>
<th>Command Code</th>
<th>Repeat &lt; 100 ms</th>
<th>Answer Slave</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>YAAA AAA0 XXXX XXXX</td>
<td>no</td>
<td>no</td>
<td>DIRECT ARC POWER CONTROL</td>
</tr>
<tr>
<td>0</td>
<td>YAAA AAA1 0000 0000</td>
<td>no</td>
<td>no</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>YAAA AAA1 0000 0001</td>
<td>no</td>
<td>no</td>
<td>UP</td>
</tr>
<tr>
<td>2</td>
<td>YAAA AAA1 0000 0010</td>
<td>no</td>
<td>no</td>
<td>DOWN</td>
</tr>
<tr>
<td>3</td>
<td>YAAA AAA1 0000 0011</td>
<td>no</td>
<td>no</td>
<td>STEP UP</td>
</tr>
<tr>
<td>4</td>
<td>YAAA AAA1 0000 0100</td>
<td>no</td>
<td>no</td>
<td>STEP DOWN</td>
</tr>
<tr>
<td>5</td>
<td>YAAA AAA1 0000 0101</td>
<td>no</td>
<td>no</td>
<td>RECALL MAX LEVEL</td>
</tr>
<tr>
<td>6</td>
<td>YAAA AAA1 0000 0110</td>
<td>no</td>
<td>no</td>
<td>RECALL MIN LEVEL</td>
</tr>
<tr>
<td>7</td>
<td>YAAA AAA1 0000 0111</td>
<td>no</td>
<td>no</td>
<td>STEP DOWN AND OFF</td>
</tr>
<tr>
<td>8</td>
<td>YAAA AAA1 0000 1000</td>
<td>no</td>
<td>no</td>
<td>ON AND STEP UP</td>
</tr>
<tr>
<td>9-15</td>
<td>YAAA AAA1 0000 1XXX</td>
<td>no</td>
<td>no</td>
<td>RESERVED</td>
</tr>
<tr>
<td>16-31</td>
<td>YAAA AAA1 0001 XXXX</td>
<td>no</td>
<td>no</td>
<td>GO TO SCENE</td>
</tr>
<tr>
<td>32</td>
<td>YAAA AAA1 0010 0000</td>
<td>yes</td>
<td>no</td>
<td>RESET</td>
</tr>
<tr>
<td>33</td>
<td>YAAA AAA1 0010 0001</td>
<td>yes</td>
<td>no</td>
<td>STORE ACTUAL LEVEL IN THE DTR</td>
</tr>
<tr>
<td>34-41</td>
<td>YAAA AAA1 0010 XXXX</td>
<td>yes</td>
<td>no</td>
<td>RESERVED</td>
</tr>
<tr>
<td>42</td>
<td>YAAA AAA1 0010 1010</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS MAX LEVEL</td>
</tr>
<tr>
<td>43</td>
<td>YAAA AAA1 0010 1011</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS MIN LEVEL</td>
</tr>
<tr>
<td>44</td>
<td>YAAA AAA1 0010 1100</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS SYSTEM FAILURE LEVEL</td>
</tr>
<tr>
<td>45</td>
<td>YAAA AAA1 0010 1101</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS POWER ON LEVEL</td>
</tr>
<tr>
<td>46</td>
<td>YAAA AAA1 0010 1110</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS FADE TIME</td>
</tr>
<tr>
<td>47</td>
<td>YAAA AAA1 0010 1111</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS FADE RATE</td>
</tr>
<tr>
<td>48-63</td>
<td>YAAA AAA1 0011 XXXX</td>
<td>yes</td>
<td>no</td>
<td>RESERVED</td>
</tr>
<tr>
<td>64-79</td>
<td>YAAA AAA1 0100 XXXX</td>
<td>yes</td>
<td>no</td>
<td>STORE THE DTR AS SCENE</td>
</tr>
<tr>
<td>80-95</td>
<td>YAAA AAA1 0101 XXXX</td>
<td>yes</td>
<td>no</td>
<td>REMOVE FROM SCENE</td>
</tr>
<tr>
<td>96-111</td>
<td>YAAA AAA1 0110 XXXX</td>
<td>yes</td>
<td>no</td>
<td>ADD TO GROUP</td>
</tr>
<tr>
<td>112-127</td>
<td>YAAA AAA1 0111 XXXX</td>
<td>yes</td>
<td>no</td>
<td>REMOVE FROM GROUP</td>
</tr>
<tr>
<td>128</td>
<td>YAAA AAA1 1000 0000</td>
<td>yes</td>
<td>no</td>
<td>STORE DTR AS SHORT ADDRESS</td>
</tr>
<tr>
<td>129-143</td>
<td>YAAA AAA1 1000 XXXX</td>
<td>yes</td>
<td>no</td>
<td>RESERVED</td>
</tr>
<tr>
<td>144</td>
<td>YAAA AAA1 1001 0000</td>
<td>no</td>
<td>yes</td>
<td>QUERY STATUS</td>
</tr>
<tr>
<td>145</td>
<td>YAAA AAA1 1001 0001</td>
<td>no</td>
<td>yes</td>
<td>QUERY BALLAST</td>
</tr>
<tr>
<td>146</td>
<td>YAAA AAA1 1001 0010</td>
<td>no</td>
<td>yes</td>
<td>QUERY LAMP FAILURE</td>
</tr>
<tr>
<td>147</td>
<td>YAAA AAA1 1001 0011</td>
<td>no</td>
<td>yes</td>
<td>QUERY LAMP POWER ON</td>
</tr>
<tr>
<td>148</td>
<td>YAAA AAA1 1001 0100</td>
<td>no</td>
<td>yes</td>
<td>QUERY LIMIT ERROR</td>
</tr>
<tr>
<td>149</td>
<td>YAAA AAA1 1001 0101</td>
<td>no</td>
<td>yes</td>
<td>QUERY RESET STATE</td>
</tr>
<tr>
<td>150</td>
<td>YAAA AAA1 1001 0110</td>
<td>no</td>
<td>yes</td>
<td>QUERY MISSING SHORT ADDRESS</td>
</tr>
<tr>
<td>151</td>
<td>YAAA AAA1 1001 0111</td>
<td>no</td>
<td>yes</td>
<td>QUERY VERSION NUMBER</td>
</tr>
<tr>
<td>152</td>
<td>YAAA AAA1 1001 1000</td>
<td>no</td>
<td>yes</td>
<td>QUERY CONTENT DTR</td>
</tr>
<tr>
<td>Number</td>
<td>Command Code</td>
<td>Repeat &lt; 100 ms</td>
<td>Answer Slave</td>
<td>Command Name</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>----------------</td>
<td>--------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>153</td>
<td>YAAA AAA1 1001 1001</td>
<td>no</td>
<td>yes</td>
<td>QUERY DEVICE TYPE</td>
</tr>
<tr>
<td>154</td>
<td>YAAA AAA1 1001 1010</td>
<td>no</td>
<td>yes</td>
<td>QUERY PHYSICAL MINIMUM LEVEL</td>
</tr>
<tr>
<td>155</td>
<td>YAAA AAA1 1001 1011</td>
<td>no</td>
<td>yes</td>
<td>QUERY POWER FAILURE</td>
</tr>
<tr>
<td>156 - 159</td>
<td>YAAA AAA1 1001 11XX</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
<tr>
<td>160</td>
<td>YAAA AAA1 1010 0000</td>
<td>no</td>
<td>yes</td>
<td>QUERY ACTUAL LEVEL</td>
</tr>
<tr>
<td>161</td>
<td>YAAA AAA1 1010 0001</td>
<td>no</td>
<td>yes</td>
<td>QUERY MAX LEVEL</td>
</tr>
<tr>
<td>162</td>
<td>YAAA AAA1 1010 0010</td>
<td>no</td>
<td>yes</td>
<td>QUERY MIN LEVEL</td>
</tr>
<tr>
<td>163</td>
<td>YAAA AAA1 1010 0011</td>
<td>no</td>
<td>yes</td>
<td>QUERY POWER ON LEVEL</td>
</tr>
<tr>
<td>164</td>
<td>YAAA AAA1 1010 0100</td>
<td>no</td>
<td>yes</td>
<td>QUERY SYSTEM FAILURE LEVEL</td>
</tr>
<tr>
<td>165</td>
<td>YAAA AAA1 1010 0101</td>
<td>no</td>
<td>yes</td>
<td>QUERY FADE TIME / FADE RATE</td>
</tr>
<tr>
<td>166 - 175</td>
<td>YAAA AAA1 1010 XXXX</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
<tr>
<td>176 - 191</td>
<td>YAAA AAA1 1011 XXXX</td>
<td>no</td>
<td>yes</td>
<td>QUERY SCENE LEVEL (SCENES 0-15)</td>
</tr>
<tr>
<td>192</td>
<td>YAAA AAA1 1100 0000</td>
<td>no</td>
<td>yes</td>
<td>QUERY GROUPS 0-7</td>
</tr>
<tr>
<td>193</td>
<td>YAAA AAA1 1100 0001</td>
<td>no</td>
<td>yes</td>
<td>QUERY GROUPS 8-15</td>
</tr>
<tr>
<td>194</td>
<td>YAAA AAA1 1100 0010</td>
<td>no</td>
<td>yes</td>
<td>QUERY RANDOM ADDRESS (H)</td>
</tr>
<tr>
<td>195</td>
<td>YAAA AAA1 1100 0011</td>
<td>no</td>
<td>yes</td>
<td>QUERY RANDOM ADDRESS (M)</td>
</tr>
<tr>
<td>196</td>
<td>YAAA AAA1 1100 0100</td>
<td>no</td>
<td>yes</td>
<td>QUERY RANDOM ADDRESS (L)</td>
</tr>
<tr>
<td>197 - 223</td>
<td>YAAA AAA1 110X XXXX</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
<tr>
<td>224 - 255</td>
<td>YAAA AAA1 11XX XXXX</td>
<td>no</td>
<td>yes</td>
<td>APPLICATION EXTENDED COMMANDS</td>
</tr>
<tr>
<td>256</td>
<td>1010 0001 0000 0000</td>
<td>no</td>
<td>no</td>
<td>TERMINATE</td>
</tr>
<tr>
<td>257</td>
<td>1010 0011 XXXX XXXX</td>
<td>no</td>
<td>no</td>
<td>DATA TRANSFER REGISTER (DTR)</td>
</tr>
<tr>
<td>258</td>
<td>1010 0101 XXXX XXXX</td>
<td>yes</td>
<td>no</td>
<td>INITIALISE</td>
</tr>
<tr>
<td>259</td>
<td>1010 0111 0000 0000</td>
<td>yes</td>
<td>no</td>
<td>RANDOMISE</td>
</tr>
<tr>
<td>260</td>
<td>1010 1001 0000 0000</td>
<td>no</td>
<td>yes</td>
<td>COMPARE</td>
</tr>
<tr>
<td>261</td>
<td>1010 1011 0000 0000</td>
<td>no</td>
<td>no</td>
<td>WITHDRAW</td>
</tr>
<tr>
<td>262</td>
<td>1010 1101 0000 0000</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
<tr>
<td>263</td>
<td>1010 1111 0000 0000</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
<tr>
<td>264</td>
<td>1011 0001 HHHH HHHH</td>
<td>no</td>
<td>no</td>
<td>SEARCHADDRH</td>
</tr>
<tr>
<td>265</td>
<td>1011 0011 MMMM MMMM</td>
<td>no</td>
<td>no</td>
<td>SEARCHADDRM</td>
</tr>
<tr>
<td>266</td>
<td>1011 0101 LLLL LLLL</td>
<td>no</td>
<td>no</td>
<td>SEARCHADDRL</td>
</tr>
<tr>
<td>267</td>
<td>1011 0111 0AAA AAA1</td>
<td>no</td>
<td>no</td>
<td>PROGRAM SHORT ADDRESS</td>
</tr>
<tr>
<td>268</td>
<td>1011 1001 0AAA AAA1</td>
<td>no</td>
<td>yes</td>
<td>VERIFY SHORT ADDRESS</td>
</tr>
<tr>
<td>269</td>
<td>1011 1011 0000 0000</td>
<td>no</td>
<td>yes</td>
<td>QUERY SHORT ADDRESS</td>
</tr>
<tr>
<td>270</td>
<td>1011 1101 0000 0000</td>
<td>no</td>
<td>yes</td>
<td>PHYSICAL SELECTION</td>
</tr>
<tr>
<td>271</td>
<td>1011 1111 XXXX XXXX</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
<tr>
<td>272</td>
<td>1100 0001 XXXX XXXX</td>
<td>no</td>
<td>no</td>
<td>ENABLE DEVICE TYPE X</td>
</tr>
<tr>
<td>273 - 287</td>
<td>110X XXX1 XXXX XXXX</td>
<td>no</td>
<td>yes</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>
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.

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

```c
#include <LPC214x.H> // LPC214x definitions
#include "global.h"

WORD forward = 0; // DALI forward frame
BYTE answer = 0;  // DALI slave answer
BYTE f_dalitx = 0;
BYTE f_dalirx = 0;

void GetInReport(BYTE *rep) // USB host is asking for an InReport
{
    if (f_dalirx)  // answer from slave received ?
    {
        rep[0] = 1;  // send answer
        rep[1] = answer;
    }
    else
    {  
        rep[0] = 0;  // no answer
    }
}

void SetOutReport(BYTE *rep) // OutReport received from USB host
{
    forward = (rep[0] << 8) | rep[1];
    f_dalitx = 1;  // set DALI send flag
}

int main(void)
{
    USB_Init();  // USB Initialization
    USB_Connect(TRUE);  // USB Connect
    DALI_Init();
```
while (1) {
    if (f_dalitx) // flag set from USB or the DALI module
        f_dalitx = 0; // clear DALI send flag
    f_dalirx = 0; // clear DALI receive (answer) flag
    DALI_Send(); // DALI send data to slave(s)
}

2.2.2 dali_drv.c

/********************************************************************
 *        DALI master
 *  For transmit, this module uses GPIO - P0.28 (DALI send pin)
 *  DALI forward frame format:
 *      | S |        8 address bits         |        8 command bits         | stop  |
 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |
 ---+ +-+ +---+ +-+ +-+ +-+ +-+   +-+ +---+   +-+ +-+ +-+ +---+ +-+ +-+ +------------
 | | | |   | | | | | | | | |   | | |   |   | | | | | | |   | | | | | |
 +-+ +-+   +-+ +-+ +-+ +-+ +---+ +-+   +---+ +-+ +-+ +-+   +-+ +-+ +-+     
 |2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|4TE | 4TE |
 /
 For receive, this module uses T0-CAP0 input (capture and interrupt on both edges)
 CAP0.0 (P0.30) is connected to P0.29 (to check high / low level by software)
 DALI slave backward frame format:
 | S |         8 data bits           | stop  |
 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |   |   |
 +---------------+ +-+ +---+ +-+ +-+ +-+ +-+   +-+ +-------------
 |               | | | |   | | | | | | | | |   | | |
 -+               +-+ +-+   +-+ +-+ +-+ +-+ +---+ +-+     
 |4 + 7 to 22 TE |2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|2TE|4TE | 4TE |
 /
 2TE = 834 usec (1200 bps)

#include <LPC214x.h>     // LPC21xx definitions
#include "global.h"
#define INITIALISE     0xA500 // command starting initialization mode
#define RANDOMISE      0xA700 // command generating a random address
#define TE             834/2   // half bit time = 417 usec
#define MIN_TE        TE - 60 // minimum half bit time
#define MAX_TE        TE + 60 // maximum half bit time
#define MIN_2TE       2*TE - 60 // minimum full bit time
#define MAX_2TE       2*TE + 60 // maximum full bit time
static int low_time; // captured pulse low time
static int high_time; // captured pulse high time
static BYTE value; // used for dali send bit
static BYTE position; // keeps track of sending bit position
static BYTE previous; // previous received bit
static WORD frame; // holds received slave backward frame
static BYTE f_repeat; // flag command shall be repeated
static BYTE f_busy; // flag DALI transfer busy

static void DALI_Shift_Bit(BYTE val)
{
    if (frame & 0x100) // frame full ?
        frame = 0; // yes, ERROR
    else
        frame = (frame << 1) | val; // shift bit
}

/************************************************************************
; DALI_Decode (we only take action at a rising edge)
;
; Half(prev) Bit   Low Time        High Time      Action     New Half Bit
; -------------------------------------------------------------------
;     0               0               0          Shift 0         0
;     0               0               1          -ERROR-         *
;     0               1               0          Shift 0,1       1
;     0               1               1          -ERROR-         *
;     1               0               0          Shift 1         1
;     1               0               1          Shift 0         0
;     1               1               0          -ERROR-         *
;     1               1               1          Shift 0,1       1
;************************************************************************

static void DALI_Decode(void)
{
    BYTE action;
    action = previous << 2;
    if ((high_time > MIN_2TE) && (high_time < MAX_2TE))
        action = action | 1; // high_time = long
    else if (!((high_time > MIN_TE) && (high_time < MAX_TE)))
        frame = 0; // DALI ERROR
        return;

    if ((low_time > MIN_2TE) && (low_time < MAX_2TE))
        action = action | 2; // low_time = long
    else if (!((low_time > MIN_TE) && (low_time < MAX_TE)))
        frame = 0; // DALI ERROR
        return;

    switch (action)
    {
    case 0: DALI_Shift_Bit(0); // short low, short high, shift 0
        break;
    case 1: previous = 1; // new half bit is 1
        break;
    case 2: DALI_Shift_Bit(0); // long low, short high, shift 0,1
        break;
    case 3: DALI_Shift_Bit(1); // long low, short high, shift 0,1

case 3: frame = 0; // long low, long high, ERROR
break;
case 4: DALI_Shift_Bit(1); // short low, short high, shift 1
break;
case 5: DALI_Shift_Bit(0);
    if (previous == 0)
        // new half bit is 0
        break;
case 6: frame = 0; // long low, short high, ERROR
break;
case 7: DALI_Shift_Bit(1);
    if (frame & 0x100)
        // shift 0,1
        break;
default: break; // invalid

__irq void DALI_Isr(void)
{
    T0TC = 0; // reset timer
    if (TOIR & 1)
    {
        if (value)
            IOSET0 |= 0x10000000; // DALI output pin high
        else
            IOCLR0 |= 0x10000000; // DALI output pin low
    } // 0TE second half of start bit = 1
    else if (position == 0)
    {
        value = 1;
    }
    else if (position < 33) // 1TE - 32TE, so address + command
    {
        value = (forward >> ((32 - position)/2)) & 1;
        if (position & 1)
            value = !value; // invert if first half of data bit
    }
    else if (position == 33) // 33TE start of stop bit (4TE)
    {
        value = 1;
    }
    else if (position == 44) // 44TE, end stop bits + settling time
    {
        T0MR0 = 9174; // receive slave answer, timeout
        T0CCR = 0x0007; // enable rx, capture on both edges
    }
    else if (position == 45) // end of transfer
    {
        T0TCR = 2; // stop and reset timer
        if (frame & 0x100)
        {  // backward frame (answer) completed ?
            answer = (BYTE)frame;
            f_dalirx = 1; // OK ! save answer
            f_busy = 0; // end of transmission
        }
        frame = 0;
        if (f_repeat)
        {  // repeat forward frame ?
            answer = (BYTE)frame;
            f_dalitx = 1; // OK ! save answer
            f_busy = 0; // end of transmission
        }
        f_dalirx = 1; // set flag to signal application
        position++;
    }
    else if (frame & 0x001)
    { // clear MR0 interrupt flag
        TOIR = 0x01;
    }
}
else // capture interrupt for DALI receive
  {
    if (IO0PIN & 0x20000000) // check rising or falling edge P0.29
      {
        if (frame != 0) // not first pulse ?
          {
            low_time = T0CR0; // rising, so capture low time
            DALI_Decode(); // decode received bit
          }
        else
          {
            previous = 1; // first pulse, so shift 1
            DALI_Shift_Bit(1);
          }
      }
    else
      {
        high_time = T0CR0; // falling, so capture high time
        T0IR = 0x10; // reset interrupt flag
        VICVectAddr = 0; // Ack interrupt by resetting VIC
      }
  }
void DALI_Send(void)
{
  if (f_repeat) // repeat last command ?
    {
      f_repeat = 0;
    }
  else if ((forward & 0xE100) == 0xA100 || (forward & 0xE100) == 0xC100)
    {
      if ((forward & 0xFF00) == INITIALISE || forward == RANDOMISE)
        f_repeat = 1; // special command repeat < 100 ms
    }
  else if ((forward & 0x1FF) >= 0x120 && (forward & 0x1FF) <= 0x180)
    {
      f_repeat = 1; // config. command repeat < 100 ms
    }
  while (f_busy) ; // Wait until dali port is idle
  frame = 0;
  value = 0; // first half of start bit = 0
  position = 0;
  f_busy = 1; // set transfer activate flag
  T0MR0 = TE; // ~ 2400 Hz
  T0CCR = 0x0000; // disable capture interrupt
  T0MCR = 0x0003; // intr on MR0, reset timer on match 0
  T0TC = 0; // reset timer
  T0TCR = 1; // enable timer
}
void DALI_Init(void)
{
  VICVectAddr1 = (LONG) &DALI_Isr;
  VICVectCntl1 = 0x24; // channel10 on Source#4 ... enabled
  VICIntEnable |= 0x10; // channel#4 is the Timer 0
  IODIR0 = 0x01000000; // P0.28 = DALI send pin
  IOSET0 = 0x01000000;
  PINSEL1 = 0x03000000; // P0.30 as CAP0.0 = DALI receive pin
  TOPR = 60; // timer runs at 60 MHz / 60 = 1 MHz
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](image-url)
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
</tr>
<tr>
<td>2.</td>
<td>USB - DALI master</td>
</tr>
<tr>
<td>2.1</td>
<td>Hardware</td>
</tr>
<tr>
<td>2.2</td>
<td>Software</td>
</tr>
<tr>
<td>2.2.1</td>
<td>main.c</td>
</tr>
<tr>
<td>2.2.2</td>
<td>dali_drv.c</td>
</tr>
<tr>
<td>3.</td>
<td>DALI - GUI</td>
</tr>
<tr>
<td>4.</td>
<td>Legal information</td>
</tr>
<tr>
<td>4.1</td>
<td>Definitions</td>
</tr>
<tr>
<td>4.2</td>
<td>Disclaimers</td>
</tr>
<tr>
<td>4.3</td>
<td>Trademarks</td>
</tr>
<tr>
<td>5.</td>
<td>Contents</td>
</tr>
</tbody>
</table>