

AN1220: DALI Communication Using the EFR32



This application note uses the Wireless SoC Series 1 and Series 2 devices to implement the Digital Addressable Lighting Interface (DALI). The DALI uses a wired bus structure to create a communication path between control device (master) and control gear (slave).

This application note describes how to implement DALI timing, packet formats, and Manchester encoding/decoding with minimum overhead on the Wireless SoC Series 1 and Series 2 core.

KEY POINTS

- Supports DALI master and slave.
- Manchester encoding and decoding.
- Option to use DMADRV.
- Software examples.



Digital Addressable Lighting Interface

1. Device Compatibility

This application note supports multiple device families, and some functionality is different depending on the device.

MCU Series 1 consists of:

- EFM32 Jade Gecko (EFM32JG1/EFM32JG12)
- EFM32 Pearl Gecko (EFM32PG1/EFM32PG12)
- EFM32 Giant Gecko (EFM32GG11/EFM32GG12)
- EFM32 Tiny Gecko (EFM32TG11)

Wireless SoC Series 1 consists of:

- EFR32 Blue Gecko (EFR32BG1/EFR32BG12/EFR32BG13)
- EFR32 Flex Gecko (EFR32FG1/EFR32FG12/EFR32FG13/EFR32FG14)
- EFR32 Mighty Gecko (EFR32MG1/EFR32MG12/EFR32MG13/EFR32MG14)

Wireless SoC Series 2 consists of:

- EFR32 Blue Gecko (EFR32BG21)
- EFR32 Mighty Gecko (EFR32MG21)

2. DALI Overview

2.1 Terminology

The following terms are generally used in DALI system:

- **Control Device (Master):** Controller or Transmitter
- **Control Gear (Slave):** Ballast or Receiver
- **Forward Frame:** Packet sent from the master to the slave
- **Backward Frame:** Response packet sent from the slave to the master
- **Address:**
 - **Short Address:** Up to 64 slaves can be connected to the same network and each slave has an individual short address
 - **Group Address:** Up to 16 groups can exist and a slave unit can belong to several groups
 - **Broadcast:** Address used to address all the slaves

2.2 Introduction

DALI is an international standard (IEC 62386) lighting control system that provides a single interface for electronic control devices (master) and gears (slave). It can control up to 64 different slaves (e.g., ballasts) within the same control system.

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 (sent by the master to the slaves), and a “backward” frame (sent by a slave to the master).

The following sections briefly describe the basic principles of the DALI system. More information about the DALI standard can be found in <http://www.dali-ag.org>.

2.3 Frame Structure

Major characteristics of the DALI frame structure are:

- Standard asynchronous serial protocol
- The communication speed is fixed at 1200 baud $\pm 10\%$, half-duplex
- Manchester encoding is used for better resynchronisation
- The Most Significant Bit (MSB) is sent out first
- Bus is in idle (high) state between frames
- The master unit controls the communication
- The master unit sends 1 start bit, 16 bit data, and 2 stop bits (no change of phase)
- The slave unit sends 1 start bit, 8 bit data, and 2 stop bits (no change of phase)

2.3.1 Manchester Encoding

DALI uses Manchester (also called bi-phase) encoding to send the start bit and data bits, which means that the data is transmitted using the edges of the signal. A falling edge indicates a “0”, and a rising edge indicates a “1” (see figure below). TE is the half-bit time, and this is where the signal changes the phase. The defined bit rate of DALI is 1200 bps so one bit period (2TE) is about 833 μs .

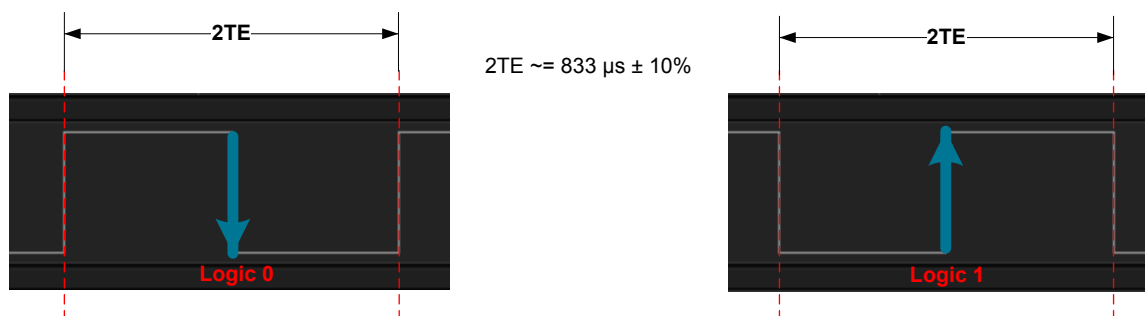


Figure 2.1. DALI Manchester Encoding

2.3.2 Forward Frame

Forward frame is the packet sent by the control device (master) to the control gear (slave). It consists of Manchester encoded bits: one start bit (logical '1'), one address byte, and one command byte. The frame is terminated by two stop bits (idle). The stop bits (4TE) do not contain any change of phase.

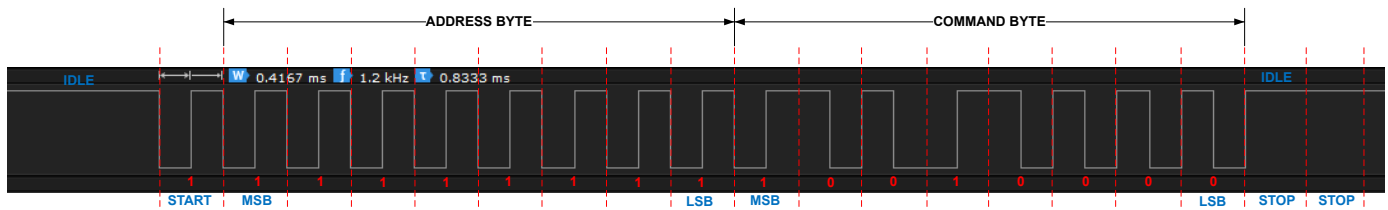


Figure 2.2. DALI Forward Frame

2.3.3 Backward Frame

Backward frame is the response packet sent by the control gear (slave) back to the control device (master). It consists of Manchester encoded bits: one start bit (logical '1'), and one response byte. The frame is terminated by two stop bits (idle). The stop bits (4TE) do not contain any change of phase.

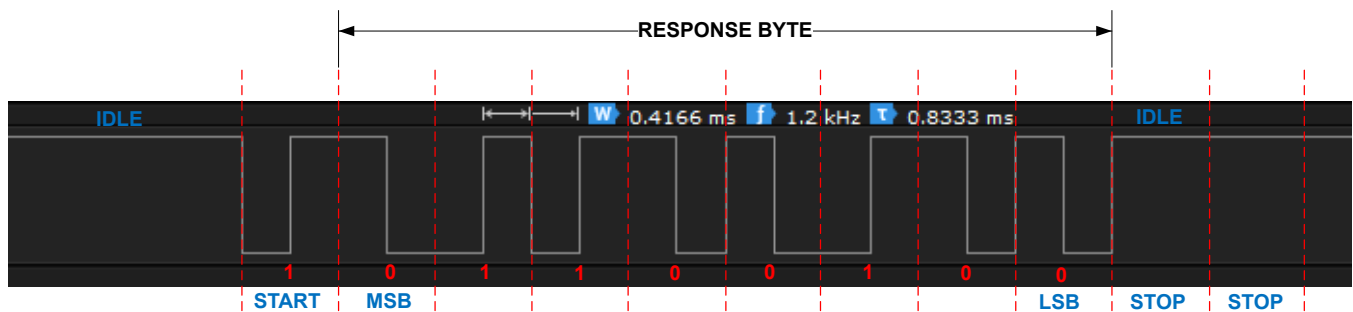


Figure 2.3. DALI Backward Frame

2.4 Timing

As described in [2.3.1 Manchester Encoding](#), TE is used to indicate half-bit time, which is about 417 μ s. The timing requirements based on TE for transmission are (see figure below):

- A forward frame takes 38TE or 15.83 ms.
- A backward frame takes 22TE or 9.17 ms.
- The settling time between two consecutive forward frames is at least 22TE or 9.17 ms.
- Four 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) is greater than or equal to 7TE or 2.92 ms, and less than or equal to 22TE or 9.17 ms.
- After sending the forward frame, the master unit will wait for 22TE or 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) is at least 22TE or 9.17 ms.

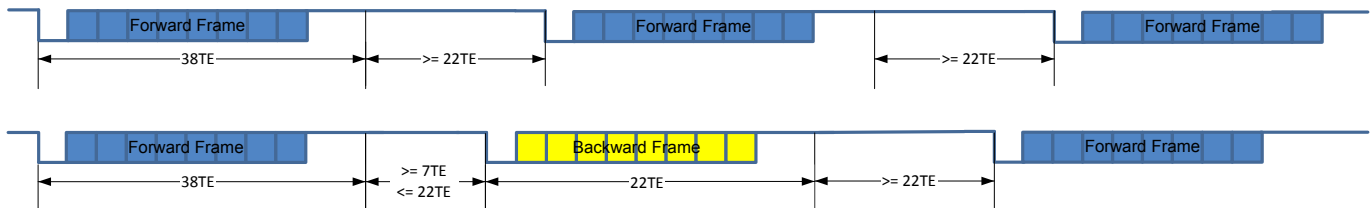


Figure 2.4. DALI Frame Timing

2.5 Physical Layer

Voltage levels present on DALI communication wires are higher than the Transistor-Transistor Logic (TTL) levels that are usually used. This is due to better noise immunity because of higher interference present on nearby power installation cables. Typically, the low voltage is 0 V and the high voltage is 16 V. The maximum and minimum bus voltage at both the transmitting unit and the receiving unit are defined as follows:

- Low level state
 - -4.5 to 4.5 V (transmitter)
 - -6.5 to 6.5 V (receiver)
- High level state
 - 11.5 to 20.5 V (transmitter)
 - 9.5 to 22.5 V (receiver)

3. Hardware Description

This section describes the hardware used for DALI communication. The electrical specifications mentioned in [2.5 Physical Layer](#) are not handled in this application note.

3.1 Hardware Resources

There are [compile options](#) to set up the hardware resources for DALI master and slave.

Table 3.1. Hardware Resources for DALI Transmission

Resource	Quantity	Usage
USARTn	1	USARTn_TX pin is configured as SPI MOSI to transmit DALI frame
PRS channel	1	USARTn_TXC is used as signal producer
DMA channel	1	Transfers data from memory buffer to USARTn_TXDATA register
Data Flash	512 bytes	Lookup table for Manchester encoding

Table 3.2. Hardware Resources for DALI Reception

Resource	Quantity	Usage
GPIO	1	GPIO is configured as input to receive DALI frame
TIMERN CC0	2	1. Provides timing to sample the GPIO for reception 2. Provides timing for the timeout and settling time between DALI frames
PRS channel	2	1. GPIO for reception (rising and falling edges) is used as signal producer 2. Sampling TIMER overflow is used as signal producer
DMA channel	2	1. Updates the TIMERN_TOP and TIMERN_TOPB registers of two TIMERS 2. Transfers data from GPIO_Px_DIN register to memory buffer
Data Flash	256 bytes	Lookup table for Manchester decoding

3.2 User Interface

The board controller on the WSTK provides a virtual COM port (CDC) interface when connected to a computer. The device on radio board can connect to this serial port interface and communicate directly (baud rate 115200-8-N-1) with the host computer terminal program (e.g., Tera Term).

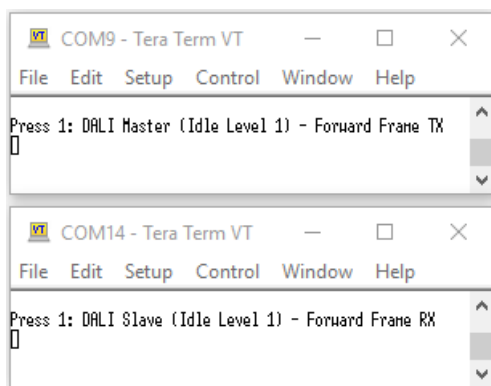


Figure 3.1. User Interface of DALI Master and Slave

4. Software Description

4.1 Compile Options

The user application must provide a header file named `dali_config.h` to configure the hardware for DALI communication. An example of EFR32MG12 hardware configuration is shown in the table below.

Table 4.1. Hardware Configuration for EFR32MG12

Define	Parameter	Description
IDLE_LEVEL ¹	1	DALI idle level is LOW if 0 and HIGH if 1
DALI_TIMER_NUM	0	TIMERN (n = 0) for DALI_TIMER to sample the GPIO for reception
TO_TIMER_NUM	1	TIMERN (n = 1) for TO_TIMER to set up timeout and settling time between DALI frames
SPI_USART_NUM	3	USARTn (n = 3) for SPI_USART to transmit DALI frame
PIN_PRS_CH	0	PRS channel for DALI_RX_PIN
TX_PRS_CH	1	PRS channel for end of SPI_USART transmission
TIMER_PRS_CH	2	PRS channel for DALI_TIMER overflow
DMA_CH_SPI_TX ²	0	DMA channel for SPI_USART transmission
DMA_CH_RX_PIN ²	1	DMA channel to update registers of DALI_TIMER and TO_TIMER
DMA_CH_RX_TMR ²	2	DMA channel to capture DALI_RX_PIN
SPI_MOSI_PIN	11	GPIO pin for SPI_USART MOSI
SPI_MOSI_PORT	gpioPortD	GPIO port for SPI_USART MOSI
SPI_TX_LOC ³	USART_ROUTELOC0_TXLOC_LOC3	SPI_USART TX location of selected GPIO
DALI_RX_PIN	12	GPIO pin to receive DALI frame
DALI_RX_PORT	gpioPortD	GPIO port to receive DALI frame
DMAREQ_NUM	0	DMA request 0 or 1 from the TIMER_PRS_CH
DALI_HALF_T	7999 ⁴	Timing for ½TE on Manchester encoding
RX_EDGE_TO	5000 ⁵	Timeout (5TE) for no edge toggles on DALI_RX_PIN
RX_BWARD_TO	22000 ⁵	Receive backward frame timeout (22TE) for DALI master
TX_BWARD_WAIT	7000 ⁵	Settling time (7TE) between forward and backward frames for DALI slave

Note:

1. The IDLE_LEVEL should set to 0 when the signal from the DALI bus is inverted by the isolator (e.g., opto-coupler).
2. If DALI_USE_DMADRV is defined in IDE setting, the DMA channels will be allocated by the DMADRV.
3. This define is only for Wireless SoC Series 1.
4. This figure is based on 38.4 MHz HFXO and TIMER prescaling factor 1. One TE is equal to $38400000/2400 = 16000$
5. These figures are based on 38.4 MHz HFXO and TIMER prescaling factor 16. One TE is equal to $38400000/(16 \times 2400) = 1000$

The following sections will refer to the items of Define column in the table.

The project can be built as DALI slave (default is DALI master) by defining DALI_SLAVE symbol in IDE setting or `dali_config.h`. If DMADRV had already been used in existing project, the DALI function can integrate into this project by defining DALI_USE_DMADRV symbol in IDE setting or `dali_config.h`.

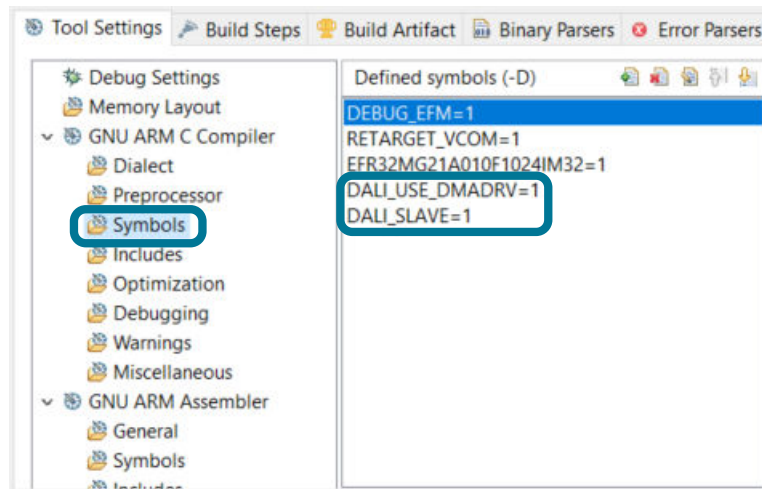


Figure 4.1. DALI_SLAVE and DALI_USE_DMADRV Symbols in Simplicity IDE

4.2 PRS Producers and Consumers

The PRS producers and consumers used in this application note are shown in the table below.

Table 4.2. PRS Producers and Consumers

PRS Channel	Producer	Signal	Consumer
TX_PRS_CH	SPI_USART	TXC (Transmission has completed)	<ul style="list-style-type: none"> DMA_CH_RX_PIN SYNCTRIG¹ (DALI master) DMA_CH_SPI_TX SYNCTRIG¹ (DALI slave)
PIN_PRS_CH	DALI_RX_PIN	PIN (Rising and falling edges)	<ul style="list-style-type: none"> DMA_CH_RX_PIN SYNCTRIG¹ DALI_TIMER reload and start TO_TIMER reload and start
TIMER_PRS_CH	DALI_TIMER	OF (Overflow)	DMAREQ_NUM (DMAREQ0 or 1)
Note: 1. For more information on using SYNCTRIG with PRS, see the <i>AN1029: Linked Direct Memory Access (LDMA) Controller</i> .			

4.3 Manchester Encoding

The traditional method to encode the DALI message is to bit-bang a GPIO pin. An interrupt is generated using TIMER every TE, which is about 417 μ s. A '1' is sent by pulling low the line for one TE, followed by pulling it high for one TE (see [Figure 2.1 DALI Manchester Encoding on page 3](#)). Sending a '0' is exactly the opposite and sending a single bit via Manchester encoding requires two interrupts.

In order to eliminate the periodic 417 μ s TIMER interrupt overhead, the SPI is used to encode the DALI frame in this application note. The SPI runs at twice the bit frequency ($2 \times 1200 = 2400$ Hz) so the phase is changed in the middle of every single bit.

Two or 26 idle level padding bits are inserted at the beginning of SPI data so the number of data bits is a multiple of 8 (24 bits for backward frame and 64 bits for forward frame). The additional 24 padding bits on forward frame are used to meet the settling time ($\geq 22TE$) between two consecutive forward frames and settling time for transition from backward frame to forward frame (see [Figure 2.4 DALI Frame Timing on page 5](#)).

The `startDaliTxDma()` function in `dali_tx.c` encodes the forward or backward frame in Manchester format and sends it to `SPI_USART` through DMA. The figure below shows how to encode the DALI backward frame (0x64) into three bytes (0xD9, 0x69, and 0xAF) for `SPI_USART` DMA transfer.

The `DMA_CH_SPI_TX` should allocate to highest priority channel to guarantee smallest latency on serving `SPI_USART` DMA, this can avoid delay between SPI bytes DMA transfer.

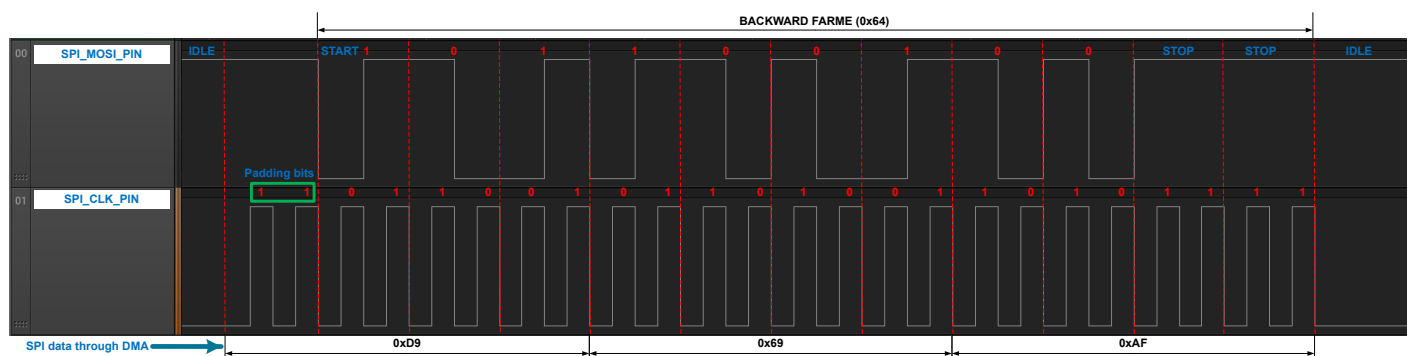


Figure 4.2. SPI for Manchester Encoding on Backward Frame

Note: The `SPI_CLK_PIN` does not need on actual implementation.

4.4 Manchester Decoding

The traditional method to decode the DALI message is to detect the edges of the signal and measure the time between these edges. This can be accomplished by using a TIMER capture input of the Wireless SoC Series 1 and Series 2 devices, because the input can capture and generate an interrupt at both rising and falling edges. 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.

In order to eliminate the periodic TIMER capture interrupt overhead, the TIMER, PRS, and DMA are used to capture the DALI frame in this application note. The figure below shows how to get a bit stream 01 1001011010011010 1111 from the DALI backward frame (0x64) for Manchester decoding.

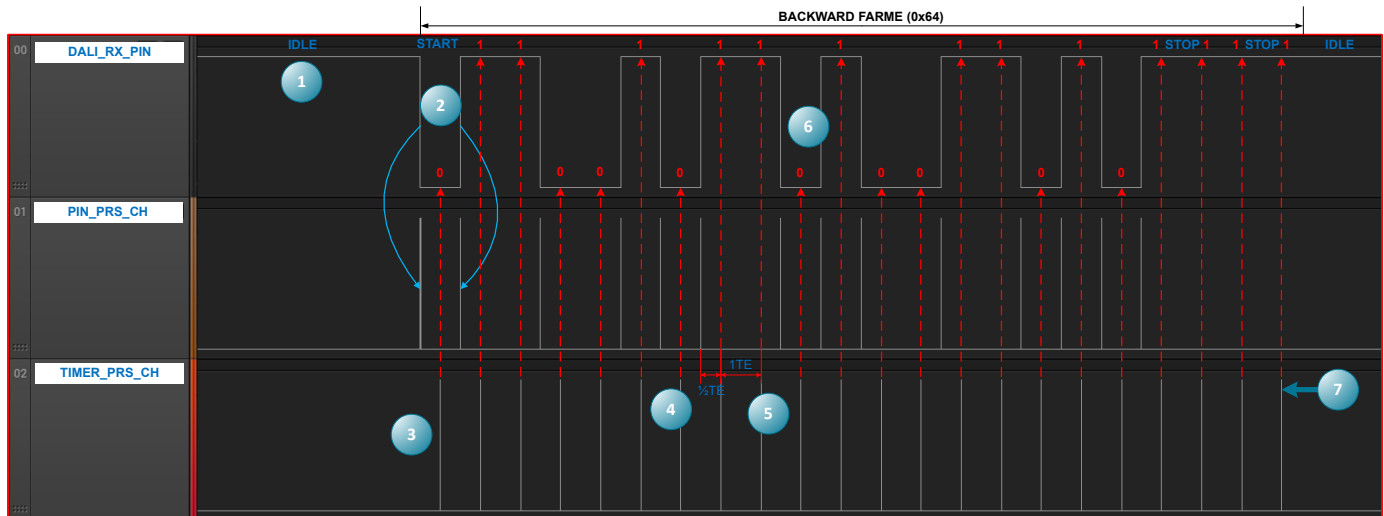


Figure 4.3. PRS for Manchester Decoding on Wireless SoC Series 1

1. Initialization of PRS, TIMER, and DMA at the end of forward frame transmission (DALI master) or idle (DALI slave)
 - Sets $\frac{1}{2}$ TE to TOP register and 1TE to TOPB register of DALI_TIMER
 - Sets timeout interval to TOP register of TO_TIMER
 - Sets PIN_PRS_CH pulse (rising edge) to reload and start DALI_TIMER and TO_TIMER
 - Starts DMA_CH_RX_PIN and DMA_CH_RX_TMR
 - Waits pulse from PIN_PRS_CH to reload and start DALI_TIMER, and TO_TIMER
 - Waits pulse from PIN_PRS_CH to trigger DMA_CH_RX_PIN SYNCTRIG
2. The PIN_PRS_CH pulses are generated by falling and rising edges on DALI_RX_PIN
 - The TOP register of DALI_TIMER is started and reloaded with $\frac{1}{2}$ TE
 - The TOP register of TO_TIMER is started and reloaded with timeout interval
 - The DMA_CH_RX_PIN writes $\frac{1}{2}$ TE to TOP register and 1TE to TOPB register of DALI_TIMER for next PRS trigger
 - The DMA_CH_RX_PIN writes RX_EDGE_TO to TOP register of TO_TIMER
 - Waits next PIN_PRS_CH pulse to repeat above processes
3. The TIMER_PRS_CH pulses are generated by overflows on DALI_TIMER
 - The DMA_CH_RX_TMR captures level (0 or 1) on DALI_RX_PIN to memory buffer
 - The TOP register of DALI_TIMER is updated by TOPB register (1TE) for next capture
4. The DALI_RX_PIN capture timing is $\frac{1}{2}$ TE when an edge toggles on DALI_RX_PIN
5. The DALI_RX_PIN capture timing is 1TE if no edge toggles on DALI_RX_PIN
6. The DALI_RX_PIN capture is always in the middle of every bit, this can eliminate the error due to drifting
7. The LDMA interrupt is triggered after capturing 22 bits (backward frame) or 38 bits (forward frame) data from DMA_CH_RX_TMR loop transfer
 - Stops DMA_CH_RX_PIN
 - Sets PIN_PRS_CH producer to none
 - Stops DALI_TIMER and TO_TIMER
 - The received bit stream is ready to decode by decodeDaliRx() function in dali_rx.c

4.5 API

The APIs for user application are described in the table below. These functions can be found in `dali_tx.c` and `dali_rx.c` source files.

Table 4.3. API for DALI Communication

Function	Parameter	Return	Usage
<code>void initDali(void)</code>	—	—	To initialize USART, DMA, PRS, TIMER for DALI communication
<code>void startDaliTxDma(uint8_t addr, uint8_t data)</code>	<ul style="list-style-type: none"> Forward frame — Address and data Backward frame — Data only 	—	<ul style="list-style-type: none"> DALI master — To initialize forward frame transmission and backward frame reception DALI slave — To initialize backward frame transmission
<code>void startDaliRxDma(void)</code>	—	—	<ul style="list-style-type: none"> DALI master — Not applicable DALI slave — To initialize forward frame reception
<code>bool decodeDaliRx(uint8_t *addr, uint8_t *data)</code>	<ul style="list-style-type: none"> Forward frame — Pointers of address and data Backward frame — Pointer of data only 	<ul style="list-style-type: none"> True if succeed False if framing error 	<ul style="list-style-type: none"> DALI master — To decode received bit stream into address and data DALI slave — To decode received bit stream into data
<code>DaliStatus_t getDaliStatus(void)</code>	—	DALI status	To get current DALI status
<code>void setDaliStatus(DaliStatus_t status)</code>	DALI status to be set	—	To set DALI status

4.6 Event and Interrupt

4.6.1 DALI Master

The event and interrupt on DALI master are described in the figure and table below.

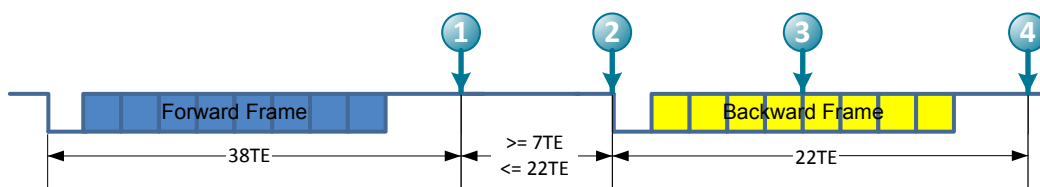


Figure 4.4. DALI Master Transmission and Reception

Table 4.4. DALI Master Event and Interrupt

Event	Interrupt	Action
1 — End of forward frame transmission	—	The backward frame reception (DMA_CH_RX_PIN SYNCTRIG) is triggered by TX_PRS_CH pulse (SPI_USART TXC)
2 — Backward frame timeout	TO_TIMER	To inform user application no backward frame has been started after 22TE (RX_BWARD_TO)
3 — Data reception timeout	TO_TIMER	To inform user application no edge toggles in 5TE (RX_EDGE_TO) after receiving start bit
4 — End of backward frame reception	DMA_CH_RX_TMR	The backward frame bit stream is ready for decode in user application

4.6.2 DALI Slave

The event and interrupt on DALI slave are described in the figure and table below.

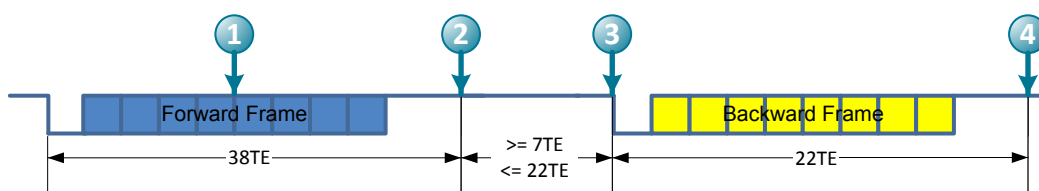


Figure 4.5. DALI Slave Reception and Transmission

Table 4.5. DALI Slave Event and Interrupt

Event	Interrupt	Action
1 — Data reception timeout	TO_TIMER	To inform user application no edge toggles in 5TE (RX_EDGE_TO) after receiving start bit
2 — End of forward frame reception	DMA_CH_RX_TMR	To start the TO_TIMER for settling time 7TE (TX_BWARD_WAIT) between forward and backward frame and the forward frame bit stream is ready for decode in user application.
3 — Settling time has expired	TO_TIMER	To start the backward frame transmission in user application
4 — End of backward frame transmission	DMA_CH_SPI_TX	To wake up core from Energy Mode 1 (EM1) if necessary

4.6.3 Comparison

The comparison between traditional method and method used in this application note is shown in the table below.

Table 4.6. Traditional Method Versus Method Used in this Application Note

	Traditional Method	Method Used in this Application Note
Advantages	<ul style="list-style-type: none"> • Straight forward • Simple hardware (TIMER) is enough 	<ul style="list-style-type: none"> • Only one interrupt from DMA during DALI master transmission and reception • Three interrupts from DMA and TIMER during DALI slave reception and transmission • Most of the time can stay in Energy Mode 1 (EM1) to reduce power consumption • Small overhead within interrupt service routines • Efficient Manchester encoding and decoding
Disadvantages	<ul style="list-style-type: none"> • Periodic interrupts may cause problem with the wireless stack • Overhead for Manchester encoding and decoding within interrupt service routines 	<ul style="list-style-type: none"> • Needs to understand the inter-communications between peripherals • Requires extra hardware resources (USART, PRS, and DMA)

5. Testing

The DALI master example is in `BRD4161A_EFR32MG12P_dali.slsproj` and `BRD4161A_EFR32MG12P_dali_dmadrv.slsproj` project files under `SimplicityStudio` folder. Imports one of these projects into Simplicity IDE, builds the project and programs the hex file to EFR32MG12 radio board (BRD4161A).

The DALI slave example is in `BRD4181A_EFR32xG21_dali.slsproj` and `BRD4181A_EFR32xG21_dali_dmadrv.slsproj` project files under `SimplicityStudio` folder. Imports one of these projects into Simplicity IDE, builds the project and programs the hex file to EFR32xG21 radio board (BRD4181A).

5.1 Test Setup

The interconnection diagram used for DALI communication is shown in the figure below. The DALI master is the EFR32MG12 WSTK (radio board BRD4161A) and the DALI slave is the EFR32xG21 WSTK (radio board BRD4181A).

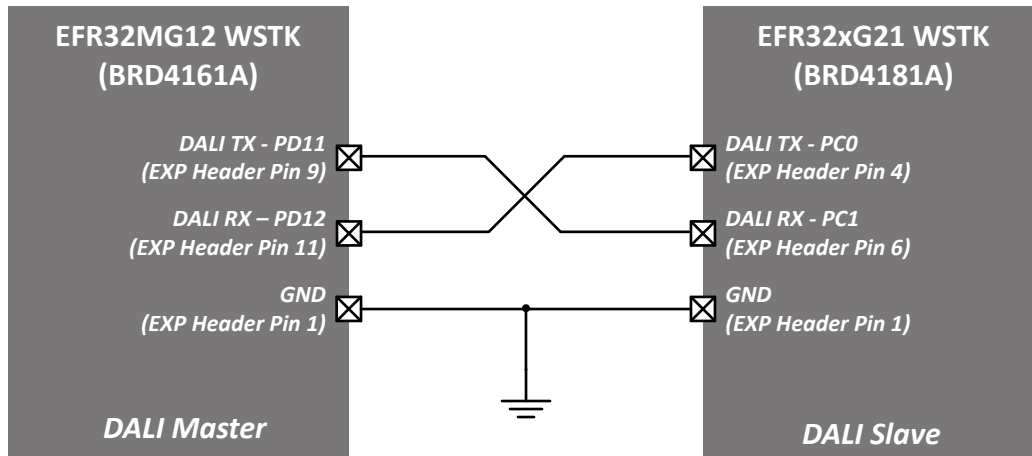


Figure 5.1. DALI Communication Connection Diagram

5.2 Test Procedure

Follow test procedures below to run the DALI communication example.

- Press 1 in the DALI slave user interface to wait for the forward frame from DALI master
- Press 1 in the DALI master user interface to send forward frame (address 255 and data 144) to DALI slave and wait for the backward frame from DALI slave
- DALI slave will display the forward frame and backward frame (if succeed) on user interface and send the backward frame (data 100) to DALI master
- DALI slave will display the error message (if fail) on user interface
- DALI master will display the forward frame and backward frame (if succeed) or error message (if fail) on user interface

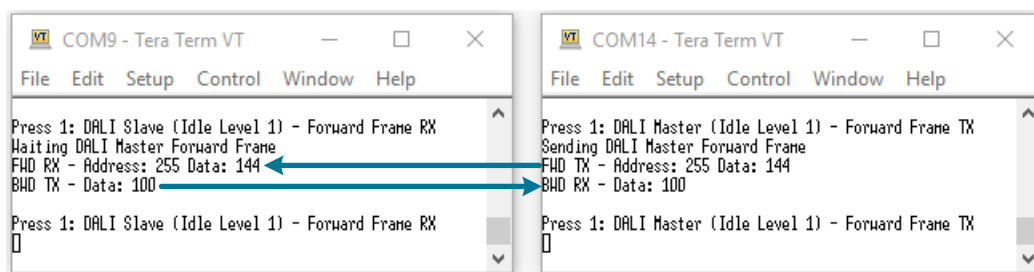


Figure 5.2. DALI Communication Example

6. Revision History

Revision 0.1

September, 2019

- Initial Revision

Silicon Labs

Simplicity Studio™4



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



IoT Portfolio
www.silabs.com/IoT



SW/HW
www.silabs.com/simplicity



Quality
www.silabs.com/quality



Support and Community
community.silabs.com

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, ClockBuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, ISOmodem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

<http://www.silabs.com>