AN1220: DALI Communication Using the EFR32

This application note uses Series 1 and Series 2 devices to implement the Digital Addressable Lighting Interface (DALI) protocol. DALI uses a wired bus structure to create a communication path between a control device (main) and a control gear (secondary).

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

For DALI example projects, see: https://github.com/SiliconLabs/platform_applications/tree/master/platform_dali

KEY POINTS

- Supports DALI main and secondary.
- Bit-banged and hardware implementation.
- Manchester encoding and decoding.
- Option to use DMADRV.
- Software examples.

DALI

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:

- EFM32JG1/EFM32JG12
- EFM32PG1/EFM32PG12
- EFM32GG11/EFM32GG12
- EFM32TG11

Wireless SoC Series 1 consists of:

- EFR32BG1/EFR32BG12/EFR32BG13
- EFR32FG1/EFR32FG12/EFR32FG13/EFR32FG14
- EFR32MG1/EFR32MG12/EFR32MG13/EFR32MG14

MCU Series 2 consists of:

- EFM32PG22
- EFM32PG23
- EFM32PG28

Wireless SoC Series 2 consists of:

- EFR32BG21/EFR32MG21
- EFR32BG22/EFR32FG22/EFR32MG22
- EFR32FG23/EFR32SG23/EFR32ZG23
- EFR32BG24/EFR32MG24
- EFR32FG25
- EFR32BG27/EFR32MG27
- EFR32FG28/EFR32SG28/EFR32ZG28

2. DALI Overview

2.1 Terminology

The following terms are generally used in a DALI system:

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

2.2 Introduction

DALI is an international standard (IEC 62386) lighting control system that provides a single interface for electronic control devices (mains) and gears (secondaries). Up to 64 different secondaries (e.g., ballasts) can be controlled within the same 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 main to the secondaries), and a “backward” frame (sent by a secondary to the main).

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

2.3 Frame Structure

Major characteristics of the DALI frame structure are:

- Standard asynchronous serial protocol
- Communication speed fixed at 1200 baud ± 10%, half-duplex
- Manchester encoding used for better resynchronisation
- Most significant bit (MSB) sent out first
- Bus is in idle (high) state between frames
- Main unit controls the communication
- Main unit sends 1 start bit, 16-bit data, and 2 stop bits
- Secondary unit sends 1 start bit, 8-bit data, and 2 stop bits

2.3.1 Manchester Encoding

DALI uses Manchester (also called bi-phase) encoding to send the start bit and data bits, meaning the data is transmitted using the edges of the signal. A falling edge indicates a '0', and a rising edge indicates a '1' as shown in Figure 2.1 DALI Manchester Encoding on page 4. TE is the half-bit time, and this is where the signal changes phase. The defined bit rate of DALI is 1200 bps, so one bit period (2TE) is about 833 µs.

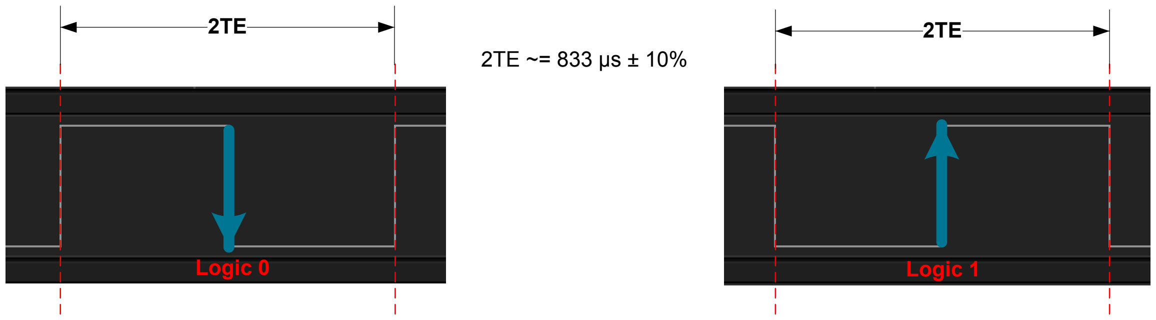


Figure 2.1. DALI Manchester Encoding

2.3.2 Forward Frame

A forward frame is the packet sent by the control device (main) to the control gear (secondary). It consists of a Manchester encoded 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.

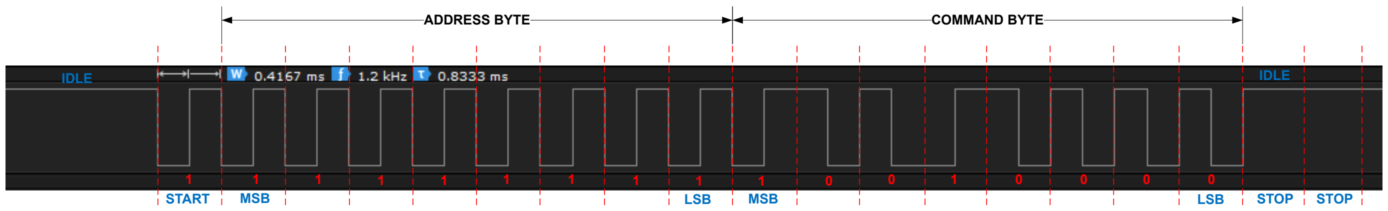


Figure 2.2. DALI Forward Frame

2.3.3 Backward Frame

A backward frame is the response packet sent by the control gear (secondary) back to the control device (main). It consists of a Manchester encoded 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.

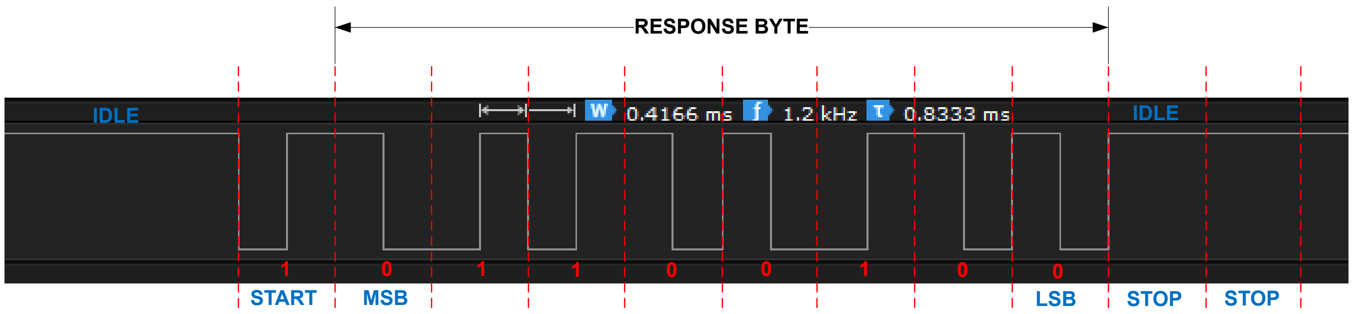


Figure 2.3. DALI Backward Frame

2.4 Timing

As described in 2.3.1 Manchester Encoding, TE is used to indicate a half-bit time, which is about 417 µs. The timing requirements based on TE for transmission are listed below as shown in Figure 2.4 DALI Frame Timing on page 5:

- A forward frame takes 38TE (15.83 ms).
- A backward frame takes 22TE (9.17 ms).
- The settling time between two consecutive forward frames is at least 22TE (9.17 ms).
- Four forward frames with accompanying periods of 22TE shall fit exactly in 100 ms.
- The settling time between a forward and backward frame (transition from forward to backward) is 7TE (2.92 ms) to 22TE (9.17 ms).
- After sending the forward frame, the main unit will wait for 22TE (9.17 ms). If no backward frame has been started within 22TE, this is interpreted as “no answer” from the secondary.
- The settling time between a backward and forward frame (transition from backward to forward) is at least 22TE (9.17 ms).

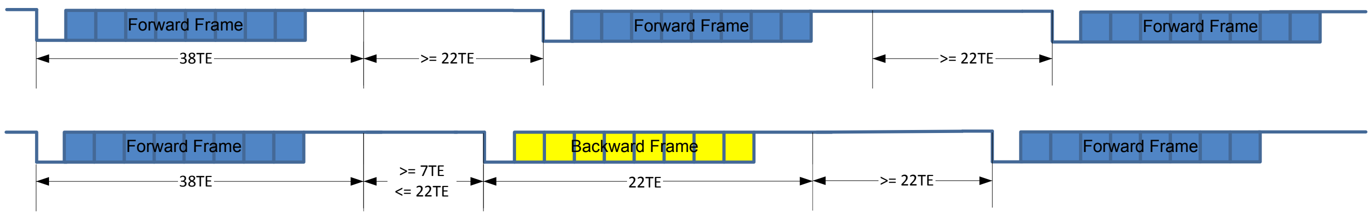


Figure 2.4. DALI Frame Timing

2.5 Physical Layer

In order to achieve better noise immunity from interference associated with nearby power installation cables, the physical DALI bus does not use TTL voltage levels. In a typical system, the low voltage is 0 V and the high voltage is 16 V. The maximum and minimum bus voltages 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 digital logic used for DALI communication. Implementation of the analog interface that complies with the electrical specifications discussed in 2.5 Physical Layer is outside the scope of this application note.

3.1 Hardware Resources

Depending on the functionality available, DALI can be implemented either with bit-banging and assistance from the USART/EUSART module or with full hardware support using the EUSART module.

The hardware resources used for DALI transmission in the bit-banged examples can be found below:

Table 3.1. Hardware Resources for Bit-Banged 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

The hardware resources used for DALI reception in the bit-banged examples can be found below:

Table 3.2. Hardware Resources for Bit-Banged 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

The hardware resources used for DALI transmission and reception in the hardware-supported examples can be found below:

Table 3.3. Hardware Resources for Hardware-Supported DALI Transmission and Reception

Resource	Quantity	Usage
EUSARTn	1	1. EUSARTn_TX is configured as UART TX to transmit DALI frame 2. EUSARTn_RX is configured as UART RX to receive DALI frame
DMA channel	2	1. Transfers data from memory to EUSARTn_TXDATA 2. Transfers data from EUARTn_RXDATA to memory
SYSRTC	1	Provides timing for the timeout and settling time between DALI frames by using the sleeptimer service component
GPIO	1	Detects start bit on RX pin

4. Software Description

4.1 EFR32MG12 and EFR32xG21 Software Examples

4.1.1 Build Options

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

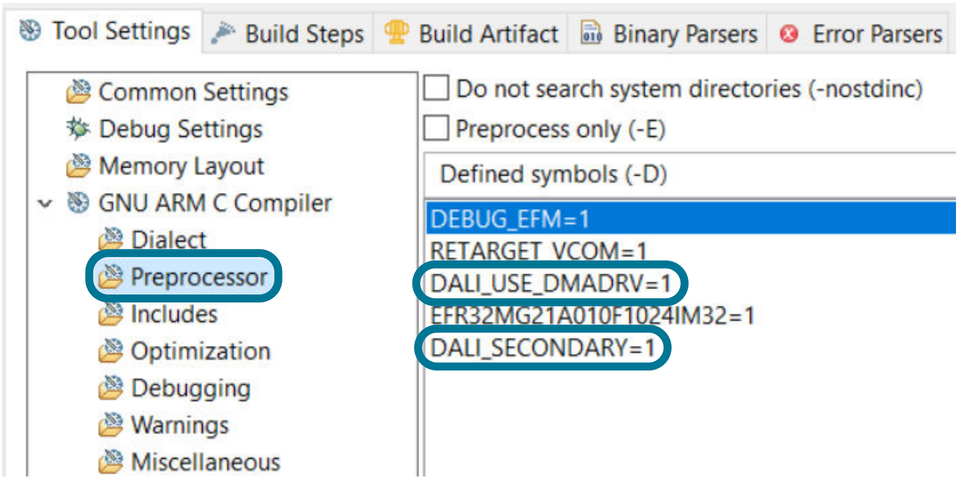
Table 4.1. Hardware Configuration for EFR32MG12

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 main
`TX_BWARD_WAIT`	`7000`[5]	Settling time (7TE) between forward and backward frames for DALI secondary

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 the `DALI_USE_DMADRV` compile time option is set, the DMA channels will be allocated by the DMADRV.
3. This define is only for Series 1 MCUs.
4. This value is based on a 38.4 MHz HFXO and a TIMER prescaling factor 1. One TE is equal to 38400000/2400 = 16000
5. These values are based on a 38.4 MHz HFXO and a TIMER prescaling factor 16. One TE is equal to 38400000/(16 x 2400) = 1000

By default, the EFR32MG12 example project is built as a DALI main. The project can be built as a DALI secondary by defining the `DALI_SECONDARY` symbol in the project settings or the `dali_config.h` file. DMADRV can be intergrated to the project by defining the

DALI_USE_DMADRV symbol in the project settings or the dali_config.h file. Figure 4.1 DALI_SECONDARY and DALI_USE_DMADRV Symbols in Simplicity Studio IDE on page 8 shows how to define these symbols in the Simplicity Studio IDE project settings.



- [ ] Do not search system directories (-nostdinc)
- [ ] Preprocess only (-E)

Defined symbols (-D)

```
DEBUG_EFM=1
RETARGET_VCOM=1
DALI_USE_DMADRV=1
EFR32MG21A010F1024IM32=1
DALI_SECONDARY=1
```

Figure 4.1. DALI_SECONDARY and DALI_USE_DMADRV Symbols in Simplicity Studio IDE

4.1.2 PRS Producers and Consumers

The PRS producers and consumers used in the example code 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)	• DMA_CH_RX_PIN SYNCTRIG[1] (DALI main) • DMA_CH_SPI_TX SYNCTRIG[1] (DALI secondary)
PIN_PRS_CH	DALI_RX_PIN	PIN (Rising and falling edges)	• DMA_CH_RX_PIN SYNCTRIG[1] • 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 AN1029: Linked Direct Memory Access (LDMA) Controller

4.1.3 DALI Frame Transmission

On devices without DALI hardware support, it is traditional to use bit-banging for frame transmission at a hardware timer rate corresponding to the 417 µs TE interval. A '1' is sent by driving the transmit line low for one TE, followed by driving the line high for another TE (see Figure 2.1 DALI Manchester Encoding on page 4). A ‘0’ is sent by driving the transmit line high for one TE, followed by driving the line low for another TE. In both cases, sending a single Manchester-encoded bit requires two interrupts.

To eliminate the periodic 417 µs timer interrupt overhead, the USART is configured for SPI mode to encode the DALI frame in the example code. The SPI runs at twice the bit frequency (2 x 1200 = 2400 Hz), so the phase is changed in the middle of every single bit.

2 or 26 idle level padding bits are inserted at the beginning of the SPI data so the number of data bits is a multiple of 8 (24 bits for a backward frame and 64 bits for a forward frame). The additonal 24 padding bits on forward frames are used to meet the settling time of at least 22TE between two consecutive forward frames and the settling time for transition from a backward frame to a 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 so the DMA can move the frame to the selected hardware serial port (`SPI_USART`) for transmission. Figure 4.2 SPI for Manchester Encoding on Backward Frame on page 9 shows how the DALI backward frame (`0x64`) is encoded into three bytes (`0xD9`, `0x69`, and `0xAF`) for `SPI_USART` transfer.

To minimize DMA servicing delays between bytes, assign the `DMA_CH_SPI_TX` DMA channel the highest priority that can be managed to minimize latency if there is the possibility that multiple DMA channels can contend for arbitration in the application code.

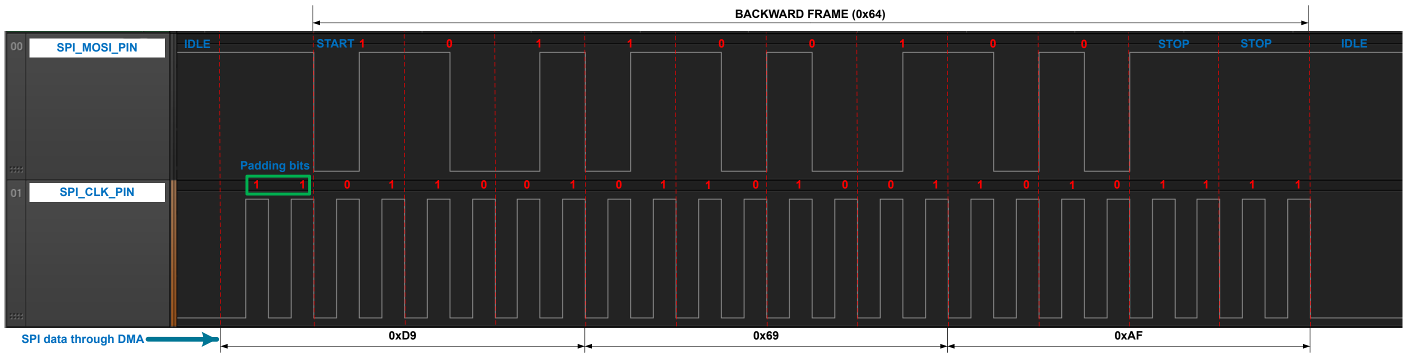


Figure 4.2. SPI for Manchester Encoding on Backward Frame

Note: The SPI_CLK_PIN signal is shown in Figure 4.2 SPI for Manchester Encoding on Backward Frame on page 9 to show how the SPI is used to manage edge transitions in the transmitted data. This pin is not required for the DALI physical interface, and the associated GPIO pin is available to the firmware for other purposes.

4.1.4 DALI Frame Reception

The traditional way to receive a DALI message is to detect the edges of the RX signal and measure the time between these edges. This can be done by using a TIMER input capture on Series 1 and Series 2 devices because the input capture can generate interrupts on both rising and falling edges. At a 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) are decoded.

To eliminate the TIMER input capture interrupt overhead, the TIMER, PRS, and DMA are used to receive DALI frames in the example code. Figure 4.3 PRS for Manchester Decoding on Wireless SoC Series 1 on page 10 shows how the bit stream `01 1001011010011010 1111` is detected for the Manchester-encoded backward frame (`0x64`).

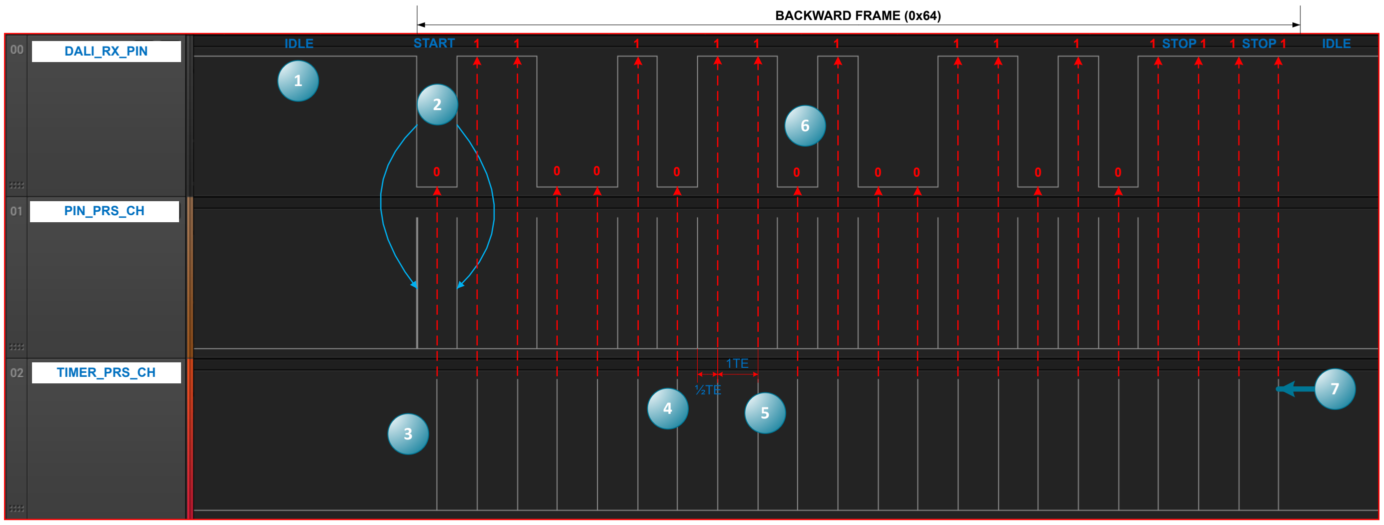


Figure 4.3. PRS for Manchester Decoding on Wireless SoC Series 1

1. The PRS, TIMER, and DMA are initialized at the end of the forward frame transmission (DALI main) or the idle state (DALI secondary)
   - For the selected `DALI_TIMER`, sets the TOP register to ½TE and the TOPB register to 1TE
   - For the selected `TO_TIMER`, sets the TOP register to the timeout interval
   - Sets the `DALI_TIMER` and `TO_TIMER` to reload and start when the PRS channel `PIN_PRS_CH` outputs a pulse
   - Enables the DMA channels `DMA_CH_RX_PIN` and `DMA_CH_RX_TMR`
   - Waits for a pulse from `PIN_PRS_CH` to trigger `DMA_CH_RX_PIN`
2. The `PIN_PRS_CH` pulses are generated by the falling and rising edges on the `DALI_RX_PIN`
   - The `DALI_TIMER` is started, and the TOP register is reloaded with ½TE
   - The `TO_TIMER` is started, and the TOP register is reloaded with the timeout interval
   - The `DMA_CH_RX_PIN` writes ½TE to the TOP register and 1TE to the TOPB register of `DALI_TIMER` for the next PRS trigger
   - The `DMA_CH_RX_PIN` writes `RX_EDGE_TO` to the TOP register of `TO_TIMER`
   - Waits for the next `PIN_PRS_CH` pulse to repeat the above processes
3. The `TIMER_PRS_CH` pulses are generated by `DALI_TIMER` overflows
   - The `DMA_CH_RX_TMR` captures the logic level (0 or 1) on the `DALI_RX_PIN` and stores the value to a memory buffer
   - The TOP register of `DALI_TIMER` is updated by the TOPB register (1TE) for the next capture
4. The `DALI_RX_PIN` capture timing is ½TE 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) from the `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 be decoded by the `decodeDaliRx()` function in `dali_rx.c`

4.1.5 API

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

Table 4.3. API for DALI Communication

Function	Parameter	Return	Usage
`void initDali(void)`	—	—	Initialize USART, DMA, PRS, TIMER for DALI communication
`void startDaliTxDma(uint8_t addr, uint8_t data)`	• Forward frame — Address and data • Backward frame — Data only	—	• DALI main — Initialize forward frame transmission and backward frame reception • DALI secondary — Initialize backward frame transmission
`void startDaliRxDma(void)`	—	—	• DALI main — Not applicable • DALI secondary — Initialize forward frame reception
`bool decodeDaliRx(uint8_t *addr, uint8_t *data)`	• Forward frame — Pointers of address and data • Backward frame — Pointer of data only	• True if succeed • False if framing error	• DALI main — Decode received bit stream into address and data • DALI secondary — Decode received bit stream into data
`DaliStatus_t getDaliStatus(void)`	—	DALI status	Get current DALI status
`void setDaliStatus( DaliStatus_t status)`	DALI status to be set	—	Set DALI status

4.1.6 Events and Interrupts

4.1.6.1 DALI Main

Figure 4.4 DALI Main Transmission and Reception on page 12 shows the transmission of a forward frame followed by reception of a backward frame on a device operating as the main. Each number in the figure corresponds to an event in the firmware and associated hardware activity or interrupt code that occurs in response to each of these events. The correspondence between these events, associated interrupts, and software actions is shown in Table 4.4 DALI Main Events and Interrupts on page 12.

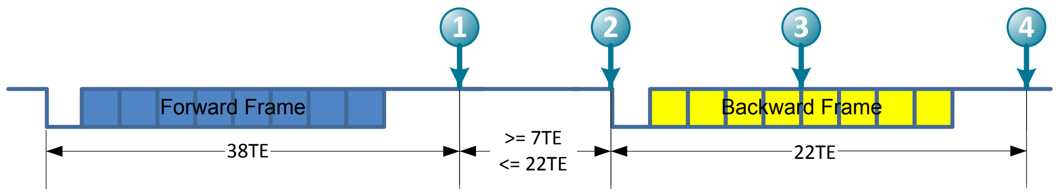


Figure 4.4. DALI Main Transmission and Reception

Table 4.4. DALI Main Events and Interrupts

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`	Inform user application no backward frame has been started after 22TE (`RX_BWARD_TO`)
3 — Data reception timeout	`TO_TIMER`	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

4.1.6.2 DALI Secondary

Figure 4.5 DALI Secondary Reception and Transmission on page 13 shows the transmission of a forward frame followed by reception of a backward frame on a device operating as the secondary. Each number in the figure corresponds to an event in the firmware and associated hardware activity or interrupt code that occurs in response to each of these events. The correspondence between these events, associated interrupts, and software actions is shown in Table 4.5 DALI Secondary Events and Interrupts on page 13.

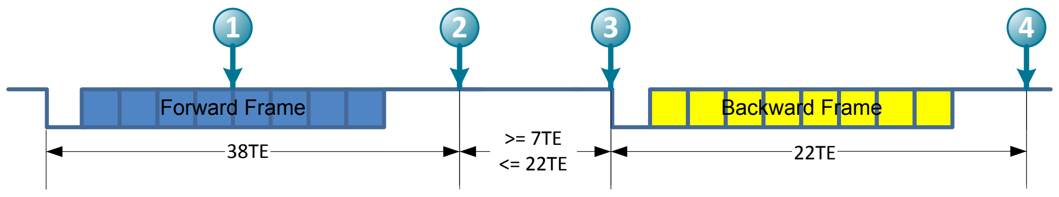


Figure 4.5. DALI Secondary Reception and Transmission

Table 4.5. DALI Secondary Events and Interrupts

Event	Interrupt	Action
1 — Data reception timeout	`TO_TIMER`	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`	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`	Start the backward frame transmission in user application
4 — End of backward frame transmission	`DMA_CH_SPI_TX`	Wake up core from Energy Mode 1 (EM1) if necessary

4.1.6.3 Comparison

A comparison between the traditional bit-bang method and the hardware-assisted bit-bang method used in this application note is shown below.

Table 4.6. Traditional Bit-Bang Method Versus Hardware-Assisted Bit-Bang Method

	Traditional Method	Hardware-Assisted
Advantages	• Straightforward GPIO implementation for both transmission and reception • Requires a single hardware timer in addition to the RX and TX GPIO pins	• Only one interrupt from DMA during DALI main transmission and reception • Three interrupts from DMA and TIMER during DALI secondary reception and transmission • Hardware-assistance reduces energy use by allowing the CPU to stay in EM1 energy mode • Minimized overhead within interrupt service routines • Efficient Manchester encoding and decoding with software look-up tables
Disadvantages	• Regular periodic interrupts can interfere with wireless stack operation • Manchester encoding and decoding introduces overhead within interrupt service routines	• Hardware assistance requires the firmware developer to understand the complex interaction between peripherals • Some applications may not be able to forgo the extra hardware resources (USART, PRS, DMA, extra TIMER) used to offload the CPU

4.2 EFR32xG24 Software Example

The EFR32xG24 software example uses the EUSART peripheral to send and receive DALI frames. Manchester encoding and decoding is handled by the peripheral, so no software encoding or decoding is needed. The sleeptimer service component and GPIO interrupts are used to manage the timeouts between the forward and backward frames.

4.2.1 Build Options

The user application must provide a header file named `dali_config.h` to configure the hardware for DALI communication. An example of EFR32xG24 hardware configuration is shown in the table below.

Table 4.7. Hardware Configuration for EFR32xG24

Define	Parameter	Description
`IDLE_LEVEL`[1]	`1`	DALI idle level is LOW if 0 and HIGH if 1
`DALI_TX_PIN`	`1`	GPIO pin to transmit DALI frame (EUSART TX)
`DALI_TX_PORT`	`gpioPortC`	GPIO port to transmit DALI frame (EUSART TX)
`DALI_RX_PIN`	`2`	GPIO pin to receive DALI frame (EUSART RX)
`DALI_RX_PIN`	`gpioPortC`	GPIO port to receive DALI frame (EUSART RX)

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).

By default, the EFR32xG24 example project is built as a DALI main. The project can be built as a DALI secondary by defining the `DALI_SECONDARY` symbol in the project settings or the `dali_config.h` file.

4.2.2 Events and Interrupts

4.2.2.1 DALI Main

Figure 4.6 DALI Main Transmission and Reception on page 15 shows the transmission of a forward frame followed by reception of a backward frame on a device operating as the main. Each number in the figure corresponds to an event in the firmware and associated hardware activity or interrupt code that occurs in response to each of these events. The correspondence between these events, associated interrupts, and software actions is shown in Table 4.8 DALI Main Event and Interrupt on page 15.

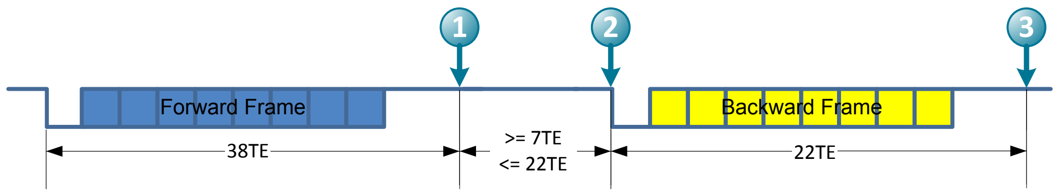


Figure 4.6. DALI Main Transmission and Reception

Table 4.8. DALI Main Event and Interrupt

Event	Interrupt	Action
1 — End of forward frame transmission	`EUSART TXC`	The timers are started at the end of the forward frame transmission
2 — Backward frame timeout	`Sleeptimer callbacks` and `GPIO`	1. Timer callback informs the user application that backward reception is ready after 7TE 2. Timer callback informs user application no backward frame has been started after 22TE 3. GPIO callback informs the user application that the start bit was detected between 7TE and 22TE
3 — End of backward frame reception	`DMA RX callback`	Inform the user application the backward frame was recieved

4.2.2.2 DALI Secondary

Figure 4.7 DALI Secondary Reception and Transmission on page 16 shows the transmission of a forward frame followed by reception of a backward frame on a device operating as the secondary. Each number in the figure corresponds to an event in the firmware and associated hardware activity or interrupt code that occurs in response to each of these events. The correspondence between these events, associated interrupts, and software actions is shown in Table 4.9 DALI Secondary Event and Interrupt on page 16.

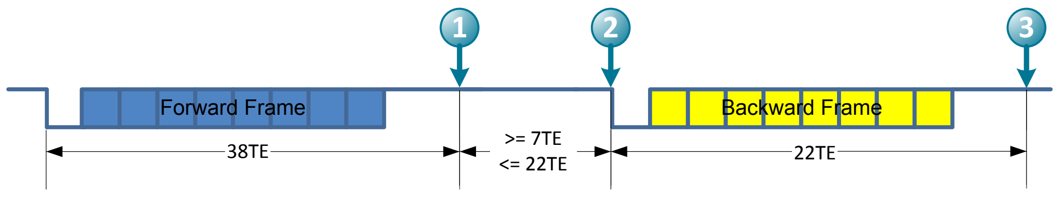


Figure 4.7. DALI Secondary Reception and Transmission

Table 4.9. DALI Secondary Event and Interrupt

Event	Interrupt	Action
1 — End of forward frame reception	`DMA RX callback`	Inform the user application the forward frame was received
2 — Settling time has expired	`Sleeptimer callback`	Start the backward frame transmission in user application
3 — End of backward frame transmission	`EUSART TXC`	Inform the user application the backward frame was transmitted

5. Testing

The DALI examples can be found in the Silicon Labs Platform Applications Example repository on Github - https://github.com/Silicon-Labs/platform_applications/tree/master.

By default, the DALI main example projects are:

- `BRD4161A_EFR32MG12P_dali.sls`
- `BRD4161A_EFR32MG12P_dali_dmadrv.sls`
- `BRD4186C_EFR32xG24_dali_dmadrv.sls`

By default, the DALI secondary example projects are:

- `BRD4181A_EFR32xG21_dali.sls`
- `BRD4181A_EFR32xG21_dali_dmadrv.sls`

Any of the main example projects can be built as a secondary by adding the `DALI_SECONDARY=1` build option. Any of the secondary example projects can be built as a main by removing the `DALI_SECONDARY=1` build option.

Import any two of these projects into the Simplicity Studio IDE, build the projects, and program the hex file to the respective radio boards.

5.1 Test Setup

Below are the required connections between the main and secondary devices:

- Main's DALI TX pin to secondary's DALI RX pin
- Main's DALI RX pin to secondary's DALI TX pin
- Main's GND pin to secondarys GND pin

Figure 5.1 DALI Communication Connection Diagram on page 17 shows the connections between two Wireless Starter Kits (WTKs), in which the main is using the BRD4161A radio board and the secondary is using the BRD4181A radio board.

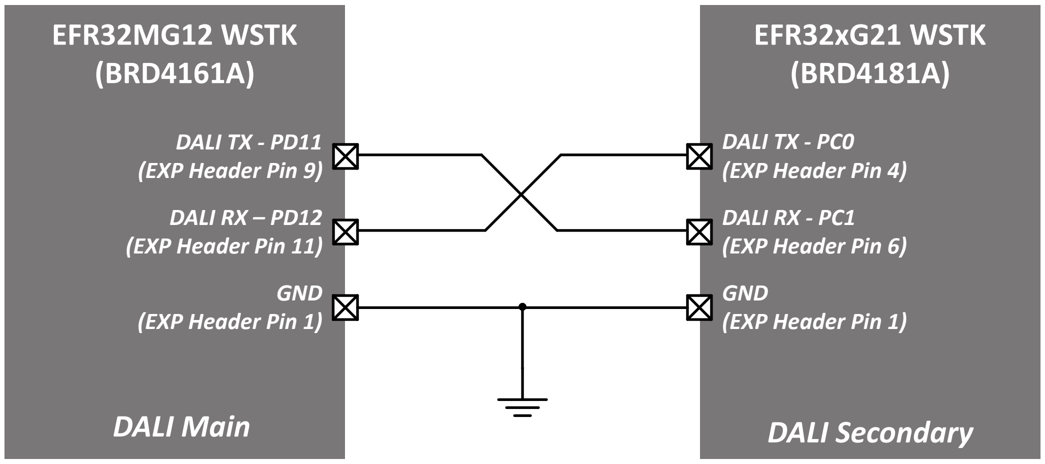
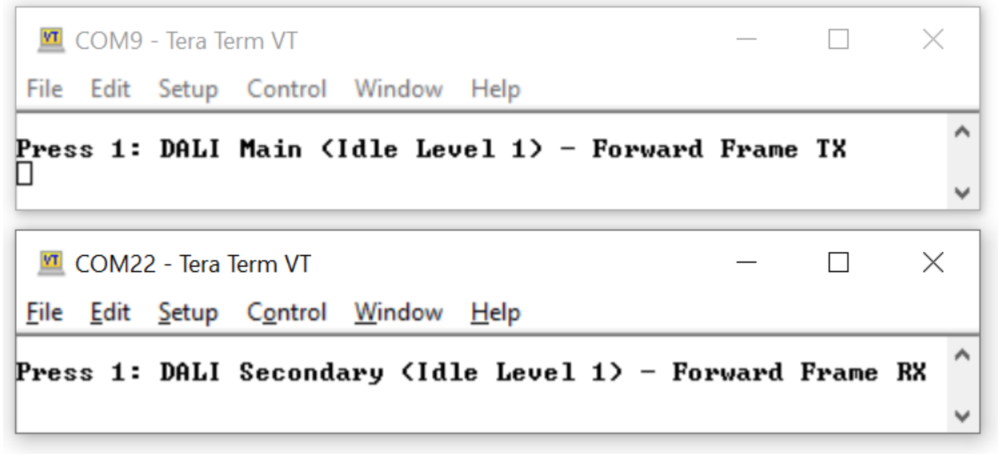


Figure 5.1. DALI Communication Connection Diagram

5.2 Example User Interface

The board controller on the WSTK provides a virtual COM port (CDC) interface when connected to a computer. This allows a host PC running a terminal program (e.g., Tera Term) to communicate with the device running the example code with the provided user interface. The virtual COM port should be set to 115200 8N1.


COM9 - Tera Term VT
File Edit Setup Control Window Help

```
Press 1: DALI Main (Idle Level 1) - Forward Frame TX
```

COM22 - Tera Term VT
File Edit Setup Control Window Help

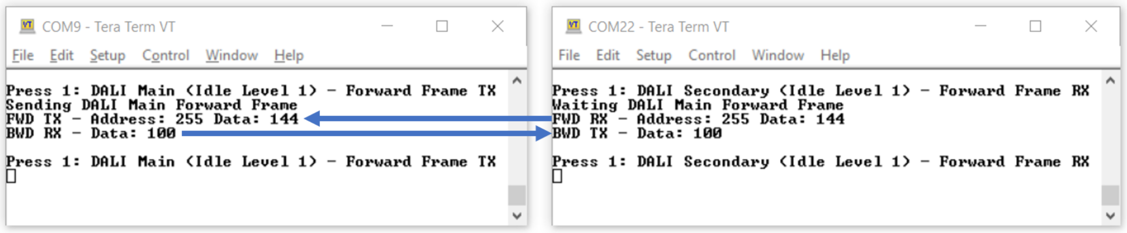
```
Press 1: DALI Secondary (Idle Level 1) - Forward Frame RX
```

Figure 5.2. User Interface of DALI Main and Secondary

5.3 Test Procedure

Follow the test procedures below to run the DALI communication example.

- Type `1` in the secondary's user interface to wait for a forward frame from DALI main
- Type `1` in the main's user interface to send a forward frame (address 255 and data 144) to the secondary and to wait for a backward frame from the secondary
- The secondary will display the forward and backward frames on the user interface and send the backward frame (data 100) to the main
- The secondary will display an error message onto the user interface in the case of a communication failure
- The main will display the forward and backward frames or error message on the user interface


COM9 - Tera Term VT
File Edit Setup Control Window Help

```
Press 1: DALI Main (Idle Level 1) - Forward Frame TX
Sending DALI Main Forward Frame
FWD TX - Address: 255 Data: 144
BWD RX - Data: 100

Press 1: DALI Main (Idle Level 1) - Forward Frame TX
```

COM22 - Tera Term VT
File Edit Setup Control Window Help

```
Press 1: DALI Secondary (Idle Level 1) - Forward Frame RX
Waiting DALI Main Forward Frame
FWD RX - Address: 255 Data: 144
BWD TX - Data: 100

Press 1: DALI Secondary (Idle Level 1) - Forward Frame RX
```

Figure 5.3. DALI Communication Example

6. Revision History

Revision 0.2

September, 2023

- Added EFR32xG22, EFx32xG23, EFR32xG24, EFR32FG25, EFR32xG27, and EFx32xG28 to 1. Device Compatibility
- Added section 4.2 EFR32xG24 Software Example
- Updated sections 3.1 Hardware Resources and 5. Testing
- Moved software examples to Silicon Labs Platform Applications Examples Github repository
- Updated terminology according to Silicon Labs' Inclusive Lexicon Project

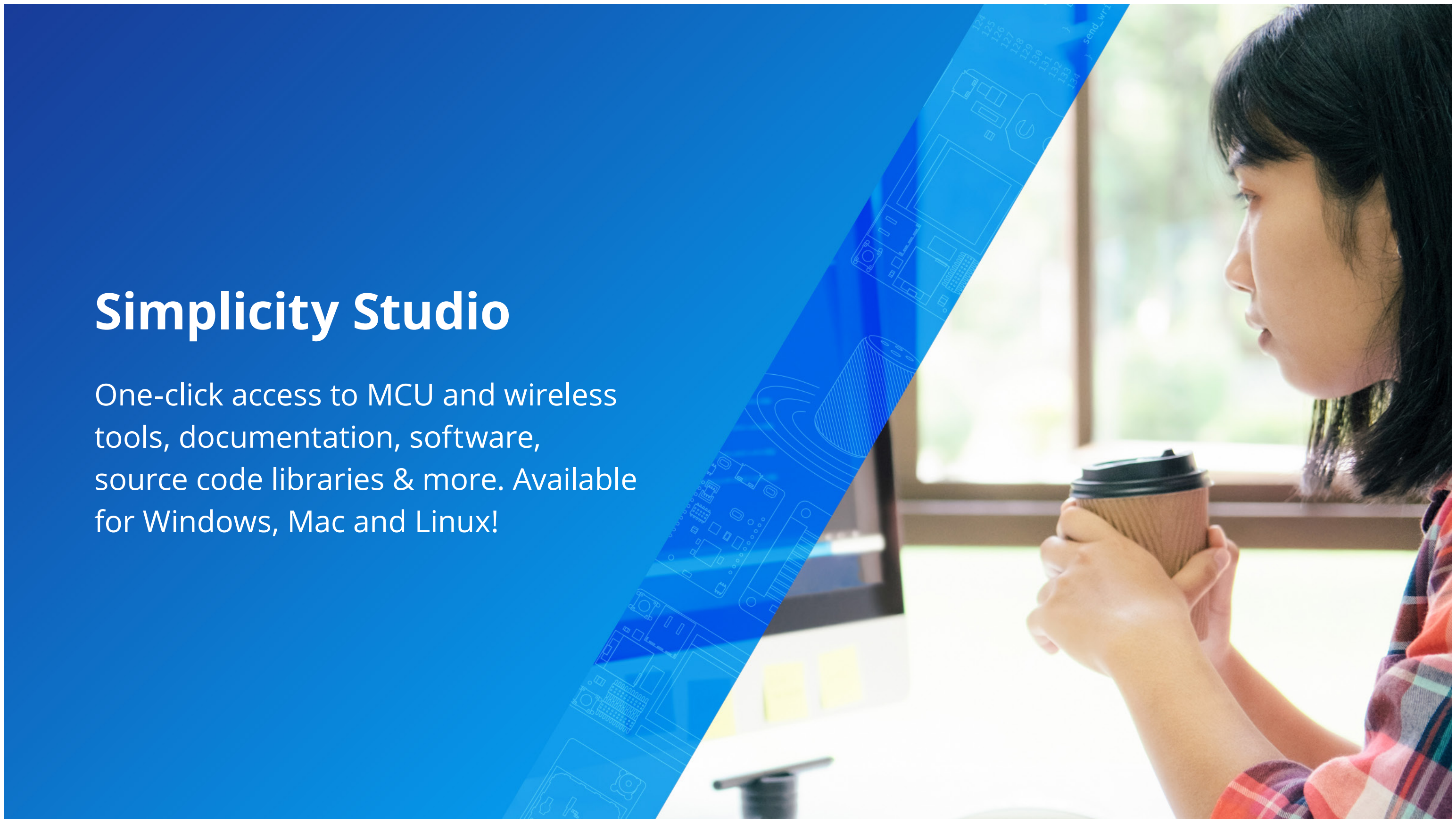
Revision 0.1

September, 2019

- Initial Revision

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 & Community
www.silabs.com/community

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

www.silabs.com