APPLICATION NOTE

AT10828: DALI Slave Stack for SAM D20/D21

ATSAM D MCU

Introduction

DALI stands for Digital Addressable Lighting Interface and targets for lighting control system. There is a specified global standard IEC 62386. The DALI stack implements the protocol set out in this standard.

This application note introduces the DALI stack of LED modules for Atmel® | SMART SAM D20/D21 devices, including:

- Software structure
- Stack functions
- Input and output of stack
- Setup the demo system

Table of Contents

1 Recommended Reading

There are several released application notes about our previous DALI solution. It is recommended to read these to get an overall idea about the DALI system. Some of these documents are also referred to later in this application note.

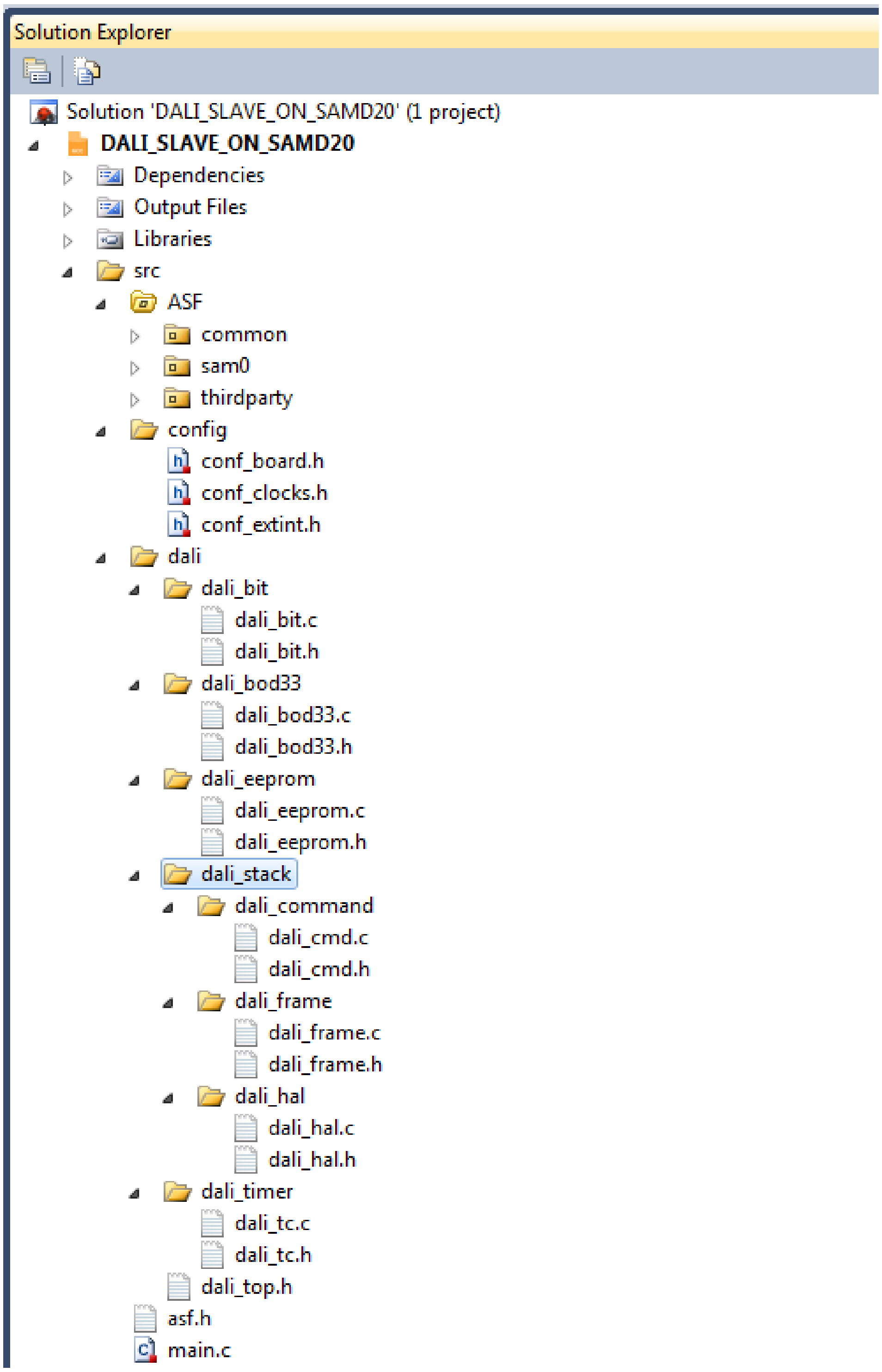
- AT03922: DALI Slave with XmegaE - Software User Guide – This application note describes the DALI slave based on ATxmega32E5. It demonstrates the software architecture and its application programming interface (API).
- AT04022: DALI Slave with XMEGA E Hardware User Guide – This application note introduces DALI slave hardware design including DALI interface, LED driver based on ATxmega32E5 device.
- AT06409: DALI Master with ATxmega32E5 User Guide – This application note demonstrates DALI master reference design based on ATxmega32E5, with the DALI bus power supply integrated. And it shows setup process of a DALI system which is operated via PC software.
- AT01244: DALI Slave Reference Design – This application note describes the DALI slave implementation based on ATmega88PA device.

The above application notes include hardware and software packages which can be downloaded from www.atmel.com. They provide hardware design files, software source code, and PC tools.

2 Software Structure

After downloading the software package of this application note and opening it with Atmel Studio 6, the SAM D20 project tree view should be as shown in Figure 2-1 below.

Figure 2-1. SAM D20 DALI Slave Project Tree



```
Solution Explorer
Solution 'DALI_SLAVE_ON_SAMD20' (1 project)
◢ DALI_SLAVE_ON_SAMD20
    ▷ Dependencies
    ▷ Output Files
    ▷ Libraries
    ◢ src
        ◢ ASF
            ▷ common
            ▷ sam0
            ▷ thirdparty
        ◢ config
              conf_board.h
              conf_clocks.h
              conf_extint.h
        ◢ dali
            ◢ dali_bit
                  dali_bit.c
                  dali_bit.h
            ◢ dali_bod33
                  dali_bod33.c
                  dali_bod33.h
            ◢ dali_eeprom
                  dali_eeprom.c
                  dali_eeprom.h
            ◢ dali_stack
                ◢ dali_command
                      dali_cmd.c
                      dali_cmd.h
                ◢ dali_frame
                      dali_frame.c
                      dali_frame.h
                ◢ dali_hal
                      dali_hal.c
                      dali_hal.h
            ◢ dali_timer
                  dali_tc.c
                  dali_tc.h
              dali_top.h
          asf.h
          main.c
```

The SAM D21 DALI slave project tree is as same as the SAM D20 project tree.

The contents of the folders are:

- ASF:
  ASF (Atmel Software Framework) provides the low level drivers/services of microcontroller modules.
- config:
  Provide board, system clock, and external interrupt controller configuration files.
- dali:
  Provide the DALI application and stack files.
  - dali_bit, dali_bod33, dali_eeprom, and dali_timer
    These folders contain the application files for the DALI stack.
    - dali_bit
      DALI bits are decoded and encoded here. EIC (External Interrupt Controller) peripheral is used for decoding.
    - dali_bod33
      When power off is detected, DALI slave need store the parameters into EEPROM if there is an update. BOD interrupt deals with this detection.
    - dali_eeprom
      Provides read and write EERPOM functions. Additional BCC (Block Check Character) code is used for error detection.
    - dali_timer
      System timers are provided here. They are used for DALI bit, frame and fade timing, PWM dimming, and random address seeds.
  - dali_stack
    The DALI stack is located in this folder.
    - dali frame
      Provide the DALI frame process files.
    - dali_cmd
      Provide DALI command implementation files.
    - dali_hal
      Hardware abstraction layer, including a complete set of APIs for using hardware resources by DALI stack that is convenient for rapid design-in and smooth integration with varied peripherals.

3 Stack Functions

For the DALI stack, the process and flowchart are same as that in application note AT03922. Refer to AT03922 “2.2 Service” and “3.2 Service Layer API Introduction” sections for details.

4 Input and Output of the Stack

The input and output of the stack act through functions defined in dali_hal.h.

- dali_hal_forward_disable_backward_enable( )
  This function is used before the DALI slave starts to send a backward frame. This will disable DALI input detection and enable slave sending. It is necessary to disable input detection because the backward frame signal can route back to DALI input.
- dali_hal_forward_enable_backward_disable( )

This function is used after DALI slave finishes sending a backward frame. This will re-enable DALI external input detection again.

- dali_hal_disable_fading_interrupt( ), dali_hal_enable_fading_interrupt( )

  When operating the variables both active in stack and the fading interrupt, it must disable the interrupt inside stack functions to avoid conflict. After operation, re-enable the interrupt again.
- dali_hal_get_dali_input_level( )

  This function detects DALI interface input voltage level used to check the interface failure state.
- dali_hal_disable_frame_timer_interrupt( ), dali_hal_enable_frame_timer_interrupt( )

  Enable and disable the frame timer interrupt, after receiving a new frame, it is necessary to start a timer interrupt for frame and frame sequence timing.
- dali_hal_update_pwm_output( )

  This function is used to update the PWM output for LED light dimming.
- dali_hal_get_seed0_value( ), dali_hal_get_seed1_value( )

  These two functions generate random address seeds for auto address allocation.
- dali_hal_read_check_data( )

  In system initialization, this function reads out the DALI parameters stored in EEPROM.

When stack variables need executing outside, below functions offer channels for outside functions to get or set them. These functions list in file dali_top.h.

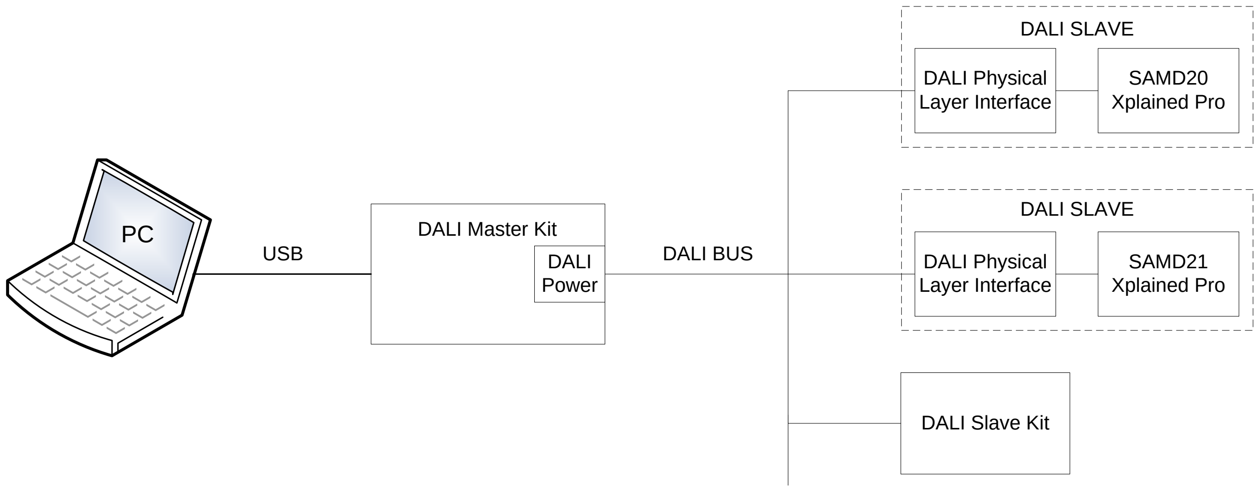
```
/**
 *  \brief Set the DALI bytes (address and data) to stack after decoding.
 */
void dali_set_addr_to_stack(uint8_t address);
void dali_set_data_to_stack(uint8_t data);

/**
 *  \brief Set the DALI bytes received flag to stack after decoding.
 */
void dali_set_received_flag_to_stack(bool flag);

/**
 *  \brief Set current DALI byte sent status to stack when encoding.
 */
void dali_set_sent_status_to_stack(uint8_t status);

/**
 *  \brief Get the DALI byte sent status from stack to start encoding.
 */
uint8_t dali_get_sent_status_from_stack(void);

/**
 *  \brief Get the DALI sent byte from stack when encoding.
 */
uint8_t dali_get_sent_data_from_stack(void);

/**
 *  \brief Get the EEPROM update flag from stack for EEPROM write.
 */
uint8_t dali_get_eeprom_flag_from_stack(void);

/**
```

```
 *  \brief Set the EEPROM update flag back to stack after EEPROM write.
 */
void dali_set_eeprom_flag_to_stack(uint8_t flag);

/**
 *  \brief Get the data address and size from stack for EEPROM write.
 */
uint8_t *dali_get_page0_data_addr_from_stack(void);
uint8_t dali_get_page0_data_size_from_stack(void);
uint8_t *dali_get_page1_data_addr_from_stack(void);

uint8_t dali_get_page1_data_size_from_stack(void);
```

5 Setup the Demo System

A typical DALI system contains DALI master, DALI slave, and DALI power. To setup this DALI demo system on SAM D20/21, see Figure 5-1 below.

Figure 5-1. DALI Demo Diagram



5.1 DALI Master Integrating DALI Power

Refer to DALI master application note AT06409 for details. PC software communicates with the DALI master to operate a DALI slave.

5.2 DALI Slave

A SAM D20 Xplained Pro Evaluation Kit or a SAM D21 Xplained Pro Evaluation Kit plus DALI physical layer interface acts as a DALI slave. For the physical layer interface, refer to DALI slave hardware application note AT04022 section “4.1 DALI physical layer interface”. Pin PA10 is used for DALI input and PA11 is output as defined in file dali_bit.h:

```
#define DALI_INPUT_PIN          PIN_PA10
#define DALI_OUTPUT_PIN         PIN_PA11
```

LED0 on Xplained Pro is used as a lighting demo. On SAM D20 Xplained Pro, connect pin PA13 and PA14 together because PA13 is DALI dimming output and PA14 connects to LED0. On SAM D21 Xplained Pro, there is no such issue since DALI dimming output PB30 connects to LED0 directly. In the DALI stack the dimming is high level active by default. LED0 on the board is low level active, so the output must be inverted, this is done in dali_tc.c:

SAM D20:

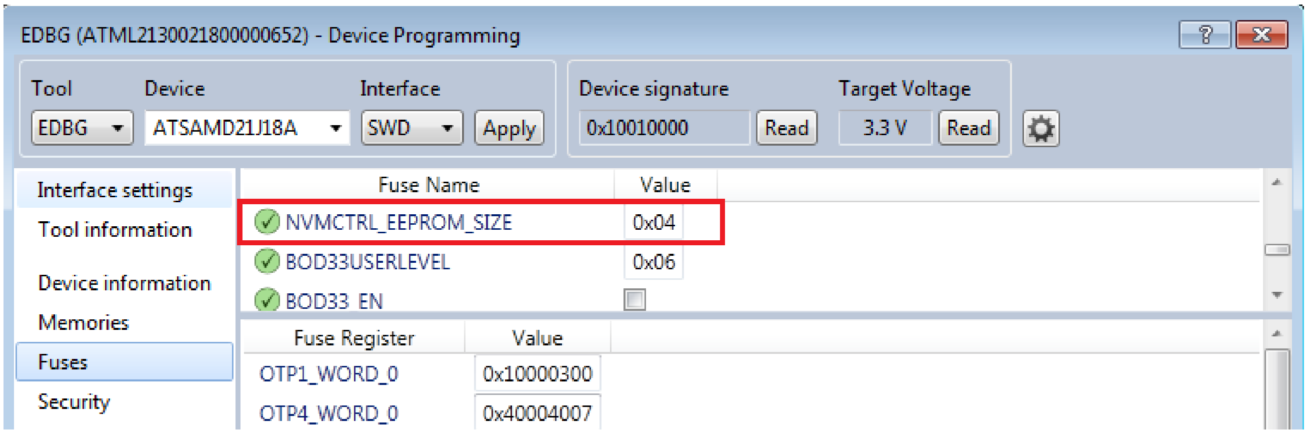
```
config_tc.waveform_invert_output  =  TC_WAVEFORM_INVERT_OUTPUT_CHANNEL_1;
```

SAM D21:

```
config_tcc.wave_ext.invert[DALI_PWM_OUTPUT]  =  true;
```

Emulated EEPROM in SAM D device is required to store DALI parameters. The device EEPROM SIZE fuse must be programmed before the demo runs. If the EEPROM section is not allocated, the application will fail inside the EEPROM initialization routine. In this application, DALI stack takes two EEPROM pages, corresponding to 120 bytes. According to emulation software described in application note AT03265: SAM D20 EEPROM Emulator Service, 0x04 or less can be set to EEPROM SIZE fuse.

Figure 5-2. EEPROM Fuse Setting



Tool: EDBG
Device: ATSAMD21J18A
Interface: SWD
Device signature: 0x10010000
Target Voltage: 3.3 V

Fuse Name	Value
NVMCTRL_EEPROM_SIZE	0x04
BOD33USERLEVEL	0x06
BOD33_EN	☐

Fuse Register	Value
OTP1_WORD_0	0x10000300
OTP4_WORD_0	0x40004007

For further development on SAM D20/21, a LED string driver should be implemented. For SAM D20, dedicated LED driver MSL20xx can be considered. Refer to DALI slave application note AT01244 section “2.3 LED driver”. For SAM D21, since there is a fault module extension in the TCC peripheral, a buck LED driver is recommended. Refer to DALI slave application note AT04022 section “4.2 Buck LED driver”. The principle is also explained in the application note “AT04204: Design a Buck Converter with XMEGA E”.

6 Reference

6.1 Device Datasheet

SAM D20/21 datasheet are available on http://www.atmel.com/ in the Documents section of Atmel SAM D product page.

6.2 Atmel Studio

Atmel Studio 6 is the integrated development platform (IDP) for developing and debugging Atmel ARM® Cortex®-M processor-based and Atmel AVR® microcontroller (MCU) applications. The latest version of Atmel Studio can be downloaded from http://www.atmel.com/tools/atmelstudio.aspx.

6.3 Xplained Pro Evaluation Kit

The SAM D20/21 Xplained Pro evaluation kits are ideal for evaluation and prototyping with the SAM D20/21 Cortex®-M0+ processor-based microcontrollers.

For SAM D20 kit: http://www.atmel.com/tools/ATSAMD20-XPRO.aspx

For SAM D21 kit: http://www.atmel.com/tools/ATSAMD21-XPRO.aspx

7 Revision History

Doc Rev.	Date	Comments
42386A	11/2014	Initial document release.