Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

AN4291
Application note

STEVAL385LED4CH: four independent high brightness LED channels using the STLUX385A digital controller

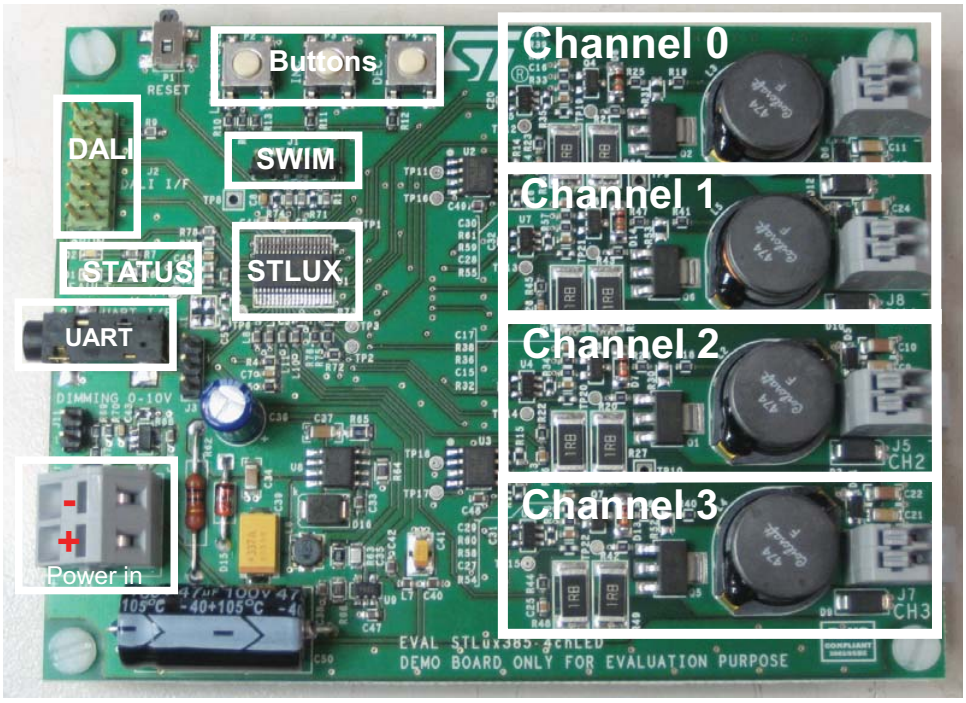
Introduction

The STEVAL385LED4CH demonstration board is a complete and configurable solution to manage four independent high brightness LED channels using the STLUX385A digital controller, which embeds advanced peripherals tailored to generate high resolution PWM signals. The LED current is independently regulated by the fixed-off-time (FOT) principle on each channel.

This document describes how to use the STEVAL385LED4CH board, the hardware and the firmware implemented on the board as well as the relevant measurements. The STLUX385A shipped with this board has already been programmed with firmware tailored for the STEVAL385LED4CH application and the source code is available.

The following picture gives an overview of the demonstration board diagram.

Figure 1. Demonstration board diagram



AM16703v1

Contents

List of tables

List of figures

1 Demonstration board features

The STEVAL385LED4CH demonstration board implements the following features:

- Four independent LED channels (external LEDs)
- 3 to 10 LEDs per channel, connected in series
- Average LED current independently adjustable from 245 mA to 1065 mA with ~82 mA/step resolution
- LED brightness adjustable via PWM dimming (256 steps independent for each channel)
- Board power supply voltage (V_{LED_PW}) from 12 V to 48 V based on the longest LED string (external AC-DC power supply not included)
- Real-time fault detection and protection, such as open or short-circuit, independent for each channel
- Board configuration (chain current, LED numbers, …) accessible via USB-UART interfaces
- Three buttons to adjust the LED brightness for each channel
- Two status LEDs: green = ready-run, red = fault

Warning: The user is responsible for configuring correctly the board before connecting any LED string.

The user configures the right values according to the following parameters:

- Number of LEDs connected on each channel (from 3 to 10)
- Average LED current desired on each LED channel (from 245 mA to 1065 mA)
- 200 Hz PWM dimming duty cycle for each LED channel (from 0% to 100% in 256 steps)
- Board power supply voltage (from 12 V to 48 V)

The number of LEDs, the current and the PWM dimming working point can be set through the command interpreter accessible via the serial interface (*Section 7.3*). Alternatively, PWM dimming can also be manually adjusted via two on-board buttons.

The firmware installed on the STLUX385A automatically computes, for each LED string:

- Primary frequency upper and lower limits (400 kHz to 15 kHz with frequency step generated by PLL @ 96 MHz - auto-setup)
- Primary frequency duty cycle depending on the selected current, number of LEDs and power voltage

To guarantee the maximum reliability of the demonstration board against the most common LED failures, the STLUX385A implements the following software and hardware protections:

- For each LED string:
  - overcurrent
  - short-circuit
  - LED voltage drop falls below the 2.9 V limit
  - LED voltage drop higher than the 4.2 V limit
  - disconnected string (open circuit)
- Power voltage outside configured lower / upper limits

The STLUX385A device drives the 4 LED channels (CH0, CH1, CH2, CH3) seen on the right side of the board.

The board uses a DC-DC converter to supply the MOS drivers (U3, U2) with 10 V, while the STLUX385A (3.3 V or 5 V) is powered by a linear regulator.

There are three buttons to manually change the dimming period. The DALI and UART TTL interfaces are supported.

Two LEDs indicate the board status: the green one shows that the system is “ready” while the red one indicates a system fault or the intervention of a protection mechanism.

2 Getting started with the STEVAL385LED4CH demonstration board

This section aims to help designers to correctly power up and control the demonstration board in a short time. It describes how the board is connected to the load and how the serial command line and the board buttons control the light/color of the LED strings.

The STEVAL385LED4CH board provides 4 independent LED channels and the user can connect up to 4 different LED chains (not included in the STEVAL385LED4CH package). The board is configured to control standard LEDs with a forward voltage (VF) between 2.9 ("`V_LMIN`") and 4.2 ("`V_LMAX`") on each LED. The limits are defined in the firmware and they can be changed by modifying the source code (see [`#define V_LMIN`] and [`#define V_LMAX`] into "led.h"). Alternatively, the user can simulate the required voltage drop by configuring, via the (ln command) an apparent incorrect number of LEDs.

The LED numbers and the current associated are configured via the serial command defined in *Section 7.3*.

The board accepts a DC input power in the range 12 V - 48 V. The user chooses the correct input voltage depending on the LED numbers and the voltage drop. The firmware accepts a minimum voltage of 2.8 V ("`V_COM_MIN`") across the inductances when the MOSFET is ON. If an incorrect power voltage value is set up, the LED current is not regulated correctly but the STLUX385A prevents the external LED module from damage. The only way to damage an external LED module is to configure a current over the maximum accepted by the LED module or define a number of LEDs different from those used. When a new LED module is applied to the board, it is recommended the user to start with the minimum current ("lc x 0") and with the dimming OFF ("ll x 0). Note that the firmware stores the last serial commands (LED number, LED current, LED dimming and adaptive voltage compensation enable/disable) on the internal E^2PROM and applies those values at power-on or reset.

2.1 First power-on procedure

The following section describes a step-by-step procedure, which guarantees a correct power-on during the first configuration of the board.

The guide assumes that the following elements are available:

- STEVAL358LED4CH board
- USB TTL serial cable shipped with the board
- Power supply
- Computer running Windows® operating system
- LED string

It is recommended the LED string to be disconnected before the first startup.

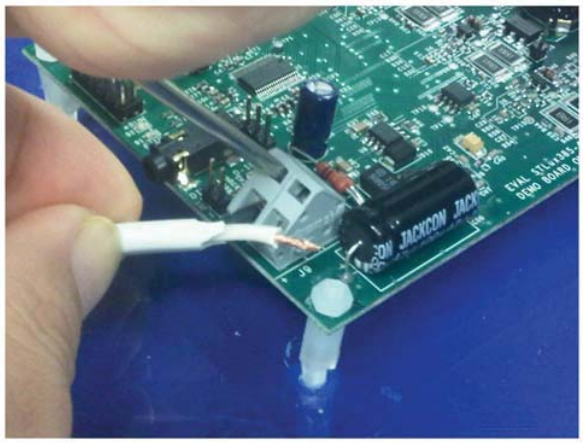
The first power-on instructions as follows:

1. Connect the USB TTL serial cable to the USB port of the computer. Windows recognizes the cable as a TTL and launches the driver installation.
2. Install the appropriate USB TTL serial cable drivers.The drivers are the "virtual COM port (VCP)" and they are provided by FTDI (http://www.ftdichip.com). Should you need any help install the drivers, please refer to your IT administrator. Once the drivers are

installed, the Windows device manager panel shows a new USB serial port (COMx) device and the COM number associated.

3. Verify that the power supply output voltage is within the board specification.
4. Switch OFF the power supply.
5. Connect the power supply to the board and check the input polarity. The following picture illustrates how to connect the power cables:

Figure 2. Power supply connection



6. Power on the board and check that the power supply current level is within the appropriate levels.
7. Monitor the red status LED on the STEVAL358LED4CH demonstration board. During the startup, the red status LED goes ON for few seconds before switching OFF. Should the red status LED stay ON for more than 3 seconds, the STLUX385A device is indicating a problem. The status LED should be checked after every step of the power-up procedure. The STLUX385A indicates any issue encountered: short-circuit, overcurrent, input power under the limit, etc. Note that, once the status LED is red, it can only be switched OFF via the "co" console command.
8. Connect the USB TTL serial cable to the STEVAL358LED4CH board.
9. Run Hyperterminal or an equivalent program from the computer connected to the USB TTL serial cable. The connection configuration is as follows:
   - baud rate 115200
   - data bits: 8
   - stop bits: 1
   - parity: no
   - flow control: no handshake

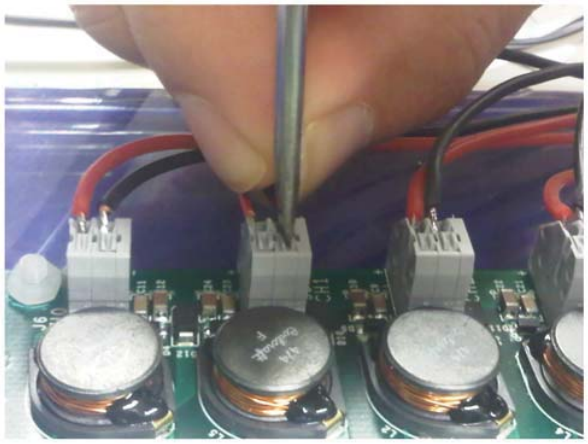
In order to verify that the connection is valid, type "?" and press ENTER. If the link is ok, the STLUX385A list of commands is displayed.

10. Should the red status LED list a problem, use the "st" command and look at *Table 3* to find the root of the problem.
11. Configure the LED string parameter via the "lc" (LED current) and "ln" (LED number) commands. See *Section 7.3* for the command reference. Note that any new setting is stored in Flash and is automatically used during the next startup.

Warning: **The current level has to be configured according to the LED capability. A current level higher than the one supported by the LED may damage the LED itself.**

12. Switch OFF the dimming using the "ll x 0" command where "x" is the channel number selected to drive the LED string. Once the dimming is disabled, channel x doesn't generate current.
13. Connect the LED string to the appropriate channel "x" connector. Please observe the LED polarity. The following picture illustrates how to connect a LED string.

Figure 3. LED string connection



14. Increase the dimming level using the "ll" command and observe the LED string switching ON.
15. Enable the adaptive voltage compensation using the "au x 1" command.

3 Design concept

The STEVAL385LED4CH demonstration board is based on a modified buck topology (reversed buck) and the fixed-off-time (FOT) principle is used to regulate the current into each LED string. The advantage of the FOT principle is that the high frequency current ripple and, therefore, the average LED current depend only on the off-time and the LED output voltage. With a fixed-off-time, the output voltage depends on the temperature and the number of LEDs. To compensate this effect, the STLUX385A implements the adaptive voltage compensation algorithm, which constantly monitors the LED voltage drop and adjusts the off-time. The input voltage variations do not affect the average current.The average LED current is directly controlled by regulating the LED peak current (configurable), which is sensed during the MOSFET on-time.

While the off-time parameter is fixed, the on-time is a function of the time that the current takes to reach the selected level: this time depends on the LED current, power voltage and LED voltage. The firmware implements the following two functions to compute the OFF and ON-time:

Equation 1

$$T_{OFF} = \frac{(2 \times (I_{MAX} - I_{AVR})) \times L}{V_{LED}}$$

Equation 2

$$T_{OFF} = \frac{(2 \times (I_{MAX} - I_{AVR})) \times L}{V_{LED\_PW} - V_{LED}}$$

where:

- V_{LED}: output voltage of the LED string (equal to V_{LED_PW} -V_{COMx} signals on the schematics)
- I_{MAX}: maximum (peak) current imposed by user and defined by *Table 1* (near 92 mA/step)
- I_{AVR}: current imposed by design (I_{MAX} - 10%)
- L: inductance imposed by design (470 µH)
- V_{LED_PW}: input voltage present at the main power pin

The on-time calculated in equation 2 has to be considered as an ideal value: the real one is actually a consequence of the triggering of a comparator monitoring the peak value of the LED current. For this reason, a percentage of the calculated on-time is added (T_x). This is empirically imposed by adding 20% to the on-time and doubling the resulting value.

Equation 3

$$T_{ON(MAX)} = (T_{ON} + T_X) \times 2$$

Equation 4

$$T_{ON(MAX)} = (T_{ON} + (T_{ON} \times 0.2))x2 = 2.4xT_{ON}$$

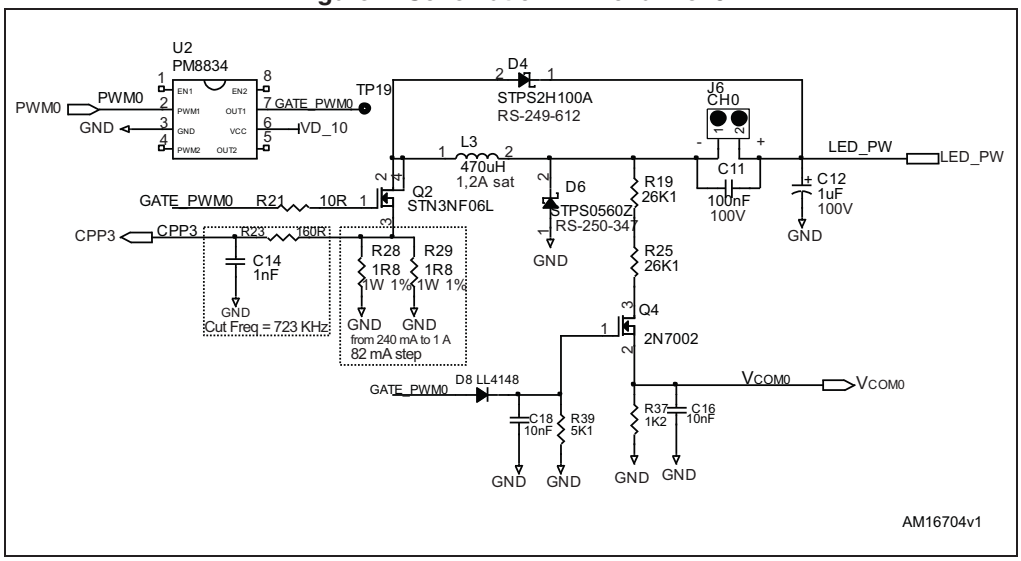
Where T_x is the time to search current and it is equal to 20% of T_{ON}. The extra time has two effects: the former, during the dimming operation, allows a quicker response from OFF to ON transition; the latter prevents a frequency lower than 15 kHz (in the audible spectrum). On the board the circuit for the transition mode principle is also present but it is not described in this manual.

3.1 LED channel schematic

To implement the FOT principle and to use the STLUX385A microcontroller, the following schematic (single LED chain) is used. It represents a standard reverse buck structure. The STLUX385A applies the PWM frequency via the P_{WM0} signal and monitors the voltage of the V_{COM0} signal. It also monitors the LED current during the "on-time" by sensing the voltage present at CPP3 line. The V_{LED} signal is calculated by the subtraction from V_{LED_PW} and V_{COM0}. The V_{COM0} signal is acquired 100 µs after the PWM dimming starts.

The schematic for a single LED channel is reported in *Figure 4*:

Figure 4. Schematic - LED channel 0



Explanation:

- The LED chain is connected to the J6 connector.
- The MOSFET Q2 drives the current from LED_PW to the LED channel and the inductor L3, charging it. The Inductor L3 is defined for all selectable current ranges. Should the user change the inductor values, some parameters, described in this manual, such as the K constant, have to be recalculated.
- The resistors R28 and R29 act as shunt resistor to sense the current through the LEDs. The current signal is then fed to CPP3 pin through an RC filter represented by the resistor R23 and the capacitor C14. The filter removes the spike caused by the turn-on current generated by MOSFET Q2. The STLUX385A detects the peak current by comparing CPP3 with an internal, DAC generated, voltage.
- The R19, R25, Q4 and R37 are used to feed the voltage of the LED chain cathode to the STLUX385A's ADC. The MOSFET Q4 is used to prevent the small biasing current

of the voltage divider from flowing through the LED during the dimming off-time. The ADC maximum voltage is 1.25 V (the ADC gain is configured to x1).

- The D4 diode is active during the OFF period, when the voltage across the inductor goes high.
- The U2 circuit (PM8834 MOSFET driver) is used to generate the correct gate voltage in case the STLUX385A is powered from 3.3 V.
- The D6 diode is used to protect the circuit if the LED chain opens or is accidentally removed during the normal operation.

3.2 Solving the equations

The T_{OFF} value introduced in the previous equation has to be converted into a number that could be used by the STLUX385A. In particular, T_{OFF} is implemented in the SMED as a timer. This timer is a 16-bit register and is clocked according to the SMED clock (CLK = 96 Mhz). The timer register value is equal to:

Equation 5

$$T_{OFF(SMED)} = T_{OFF} \times CLK$$

where:

- $T_{OFF(SMED)}$: timer value
- CLK: SMED operating frequency

The STLUX385A monitors the voltage drop caused by the LED and it is calculated as the difference between the power voltage (V_{LED_PW}) and the voltage measured after the LED string (V_{COMx}). Both V_{LED_PW} and V_{COMx} are read by the ADC unit.

Equation 6

$$V_{LED} = \frac{ADC_{LED\_VAL} \times ADC_{FS} \times P}{ADC_{TOP}}$$

where:

ADC_{LED_VAL}: the sensed LED drop voltage and calculated as V_{LED_PW} - V_{COMx}

V_{COMx}: ADC value read from middle point of each channel (LED-inductor)

V_{LED_PW}: ADC value read from the main power

P: is the resistor partition gain $(R_{19}+R_{25}+R_{37})/R_{37}$ = (53400)/1200= 44.5 Ω

ADC_{FS}: ADC top voltage = 1.25 V

ADC_{TOP}: is the step value of ADC (2^{10}) = 1024

Equation 1 can be rewritten as follows:

Equation 7

$$T_{OFF(SMED)} = 2 \times (I_{MAX} - I_{AVR}) \times L \times \left(\frac{ADC_{TOP}}{ADC_{LED\_VAL} \times ADC_{FS} \times P}\right) \times CLK$$

for example:

Equation 8

$$T_{OFF(SMED)} = \frac{2 \times (I_{MAX} - I_{AVR}) \times L \times ADC_{TOP} \times CLK}{ADC_{LED\_VAL} \times ADC_{FS} \times P}$$

where:

- I_{MAX}: max.current imposed through the DAC setup (1/16)
- I_{AVG}: average current - imposed as 90% of I_{MAX} (by design)
- (I_{MAX} - I_{AVG}): current difference: (I_{MAX}*0.1)
- I: is the inductance value = 470 10^{-6} H (by design)
- CLK: Is the SMED CLK = 96 10^{6} Hz

The average current I_{AVG} is set by design to be equal to 90% of I_{MAX}.The algorithm controls the current to be centered in I_{AVG}, with a ripple equal to +/- 10% of I_{MAX}. In equation 8, the current and the voltage across LED are unknown. The current (I_{MAX}) is an input value defined by the user. The voltage on the LED string is the difference between the main power voltage and the V_{COMx} voltage sensed by two ADC channels. Considering 1.184 A I_{MAX} current and I_{AVG} approximated to 1.066 (the real value is 1.0654), the equation 4 is expanded into:

Equation 9

$$T_{OFF(CLOCK)} = \frac{2 \times (1.184 - 1.066) \times 470 \times 10^{-6} \times 1024 \times 96x10^{6}}{ADC_{LED\_VAL} \times 1.25 \times 44.5} =$$

$$= \frac{196763}{ADC_{LED\_VAL}} = \frac{K}{ADC_{LED\_VAL}}$$

where K (196763 in this case) is a constant for the given LED current selected by the user. The I_{MAX} values accepted by the STEVAL385LED4CH board are bounded by the DAC limits as shown in the following table:

Table 1. Current, DAC, K relation table

Index	DAC value	Medium current(1)	Maximum current	K (dec)
10	13 (0x0D)	1065 mA	1184 mA	196.763E+03
9	12 (0x0C)	984 mA	1093 mA	181.628E+03
8	11 (0x0B)	901 mA	1002 mA	166.492E+03
7	10 (0x0A)	819 mA	911 mA	151.356E+03
6	9 (0x09)	738 mA	820 mA	136.221E+03
5	8 (0x08)	648 mA	729 mA	121.085E+03
4	7 (0x07)	574 mA	638 mA	105.949E+03
3	6 (0x06)	492 mA	547 mA	90.814E+03
2	5 (0x05)	410 mA	456 mA	75.678E+03

Table 1. Current, DAC, K relation table (continued)

Index	DAC value	Medium current(1)	Maximum current	K (dec)
1	4 (0x04)	329 mA	364 mA	60.543E+03
0	3 (0x03)	245 mA	273 mA	45.407E+03

1. Medium current: the value has been approximated to the nearest unit.

The index level is used by command "Ic"(see *Section 7.4.1*) to select a current value.

The output of equation 9 is the value to be stored in the SMED register that represents T_{OFF}.

Since K is constant, as long as the user doesn't change the current index level, the STLUX385A updates the T_{OFF} value only when the ADC_{LED_VAL} changes, when the LED voltage drop changes. The only operation required to update T_{OFF} is a 32-bit division. Equation 4 can be developed as follows:

Equation 10

$$T_{ON(MAX)} = T_{ON} \times 2.4 = \frac{2 \times (I_{MAX} - I_{AVG}) \times L \times ADC_{TOP} \times CLK}{ADC_{VCOM\_VAL} \times ADC_{FS} \times P} \times 2.4 =$$

$$= \frac{2.4 \times K}{ADC_{VCOM\_VAL}}$$

Since "K" is constant during operations, $T_{ON(MAX)}$ can be updated with a single 32-bit division. SMED implements the $T_{ON(MAX)}$ timer as two consecutive timers operating in different states (S1, S2).The overall update of T_{ON} and T_{MAX} timers requires 3 divisions plus the time to update the relevant SMED shadow registers.The STLUX385A needs just 200 µs to complete the operation.The "K" parameters are calculated to satisfy a board operating with a 12 V - 48 V input power range and support from 3 to 10 LEDs. Should the user use either different inductances or different parameters such as the ΔI current, only the "K" constant has to be recalculated and the FW has to be updated with the new K indexes.

4 SMED implementation

This section describes how the FOT (fixed off-time) principle is implemented in the SMED (state machine event driven). Each SMED generates the correct PWM signals and the SMED state evolution depends on both external events and internal timers.

Note: Please refer to the STLUX385A product documentation for a detailed explanation of the SMED technology.

The SMED is responsible for the generation of the PWM necessary to maintain the I_{AVG} current through the LEDs. When the SMED is triggering the PWM output, the LEDs are ON.

Digital dimming is achieved by applying a duty cycle to the LEDs. The software controls the duty cycle (referred as dimming in this document) by suspending and restoring the SMED's PWM generation. When the SMED stops triggering the PWM line, the LEDs are OFF.

Note that the dimming frequency is much lower (~200 Hz) than the PWM frequency used for current control (70 Khz - 400 kHz).

4.1 SMED input matrix

Each SMED has three associated signals. Only two inputs are used in the FOT principle described in this manual. The inputs have been logically mapped to represent:

- SMED_in(0) - digital input: senses the inductor null current (currently not used).
- SMED_in(1) - comparator input: detects when the current has reached the DAC value.
- SMED_in(2) - software input: starts and stops the SMED. Used for dimming control.

Table 2 summarizes the input and event relationship:

Table 2. SMED input association

	LED CH	SMED_in(0)	SMED_in(1)	SMED_in(2)
SMED 0	0	DIGIN 2	DAC-CPP 3	SW 0
SMED 1	1	DIGIN 1	DAC-CPP 2	SW 1
SMED 2	2	DIGIN 4	DAC-CPP 0	SW 2
SMED 3	3	DIGIN 3	DAC-CPP 1	SW 3
SMED 4	Not used	DIGIN 0		SW 4
SMED 5	Not used			SW 5

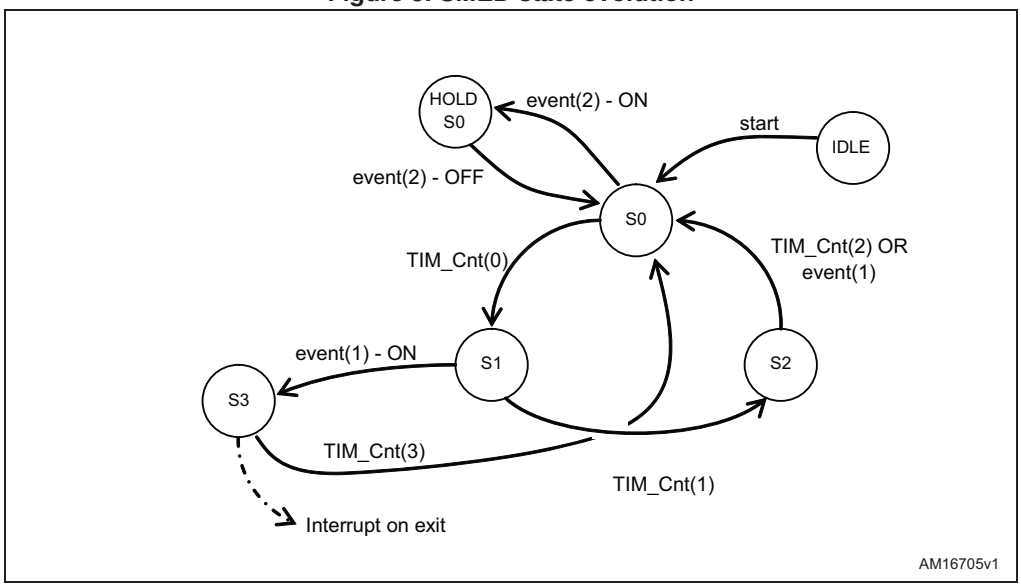
Option:

- SMED 4 PWM output is optional.
- SMED 5 PWM output is filtered and connected to CPM3. SMED5 may be optionally used to generate a reference signal for the CPM(3) input. CPM3 can be used to sense and detect the top-current event for LED channel 0 and increase the string average current granularity.The possible current steps are equal to 5 mA (typical). This option is not used in the current firmware version.

4.2 State description

The STEVAL385LED4CH demonstration board uses four independent SMEDs, one for each LED string. All the SMEDs use the same configuration and share the same control algorithm. This section describes a single SMED but the same principles are applied to all the SMEDs. The following diagram shows the SMED state evolution:

Figure 5. SMED state evolution



A state is defined by the Sx symbol while a transaction between two states is represented by a line. A transaction is generated by an event, displayed as text on the line. The event TIM_Cnt (x) is generated by the timer when it reaches the value configured in State x. An event(line) represents an event generated by the input line. The inputs associated to a SMED are defined in *Section 4.1*.

Each SMED has 3 associated events, marked as event(line) in *Figure 5*. Each event can be: type Event_Seq or Event_Jmp. While Event_Seq events push the state machine to the next sequential state, Event_Jmp allows the state machine to reach any state.

Each state is here described:

- IDLE: initial state

  This is the SMED initial state. The PWM line is set to 0 to keep the MOSFET OFF. The next state is S(0) and it is automatically entered when the SMED starts. On exit: clear timer AND set PWM to ON.

- S(0): off-time

  This state implements the FOT time and the output PWM line is set at 0 V. S(0) enters the HOLD state when the SWx bit (event 2) is set to 0 or it goes to S(1) if the timer reaches the FOT time. The SMED timer is cleared on exit.

  Event_Jmp = Hold jump ON = event(2).

  Event_Seq = TIM_Cnt.

  On exit: clear timer AND set PWM to ON.

- S(1): fault zone

  This state is used to prevent damages due to an external malfunction (either a hardware problem or an extreme fluctuation of the input voltage). The PWM output line is set high during the transaction between S(0) and S(1) and turns the MOSFET ON, leaving the current flow through the LEDs and the sense resistor. In normal conditions, the current shouldn't reach the peak during S(1) and the S(2) timer overflows, pushing the SMED to S(2). The S(1) time is set to be equal 33% of the total T_{ON} state (S(1) + S(2)) maximum time. In exceptional conditions, the CPPx signal arrives earlier, the S(1) timer expires and the SMED evolves to S(3) (OCP). CPPx indicates the overcurrent in the circuit. The SMED timer is cleared on exit.

  Event_Jmp = event(1).

  Event_Seq = TIM_Cnt.

  On exit to S(3): clear SMED timer; set PWM to OFF.

  On exit to S(2): do not clear SMED timer; turn ON the PWM output.

- S(2): current limit

  This state is used to find the current peak in normal conditions. The current peak is imposed by the voltage configured on the DAC. When the CPPx event is detected, the SMED advances to the S(0) state. During this state, the output PWM line is high and the MOSFET is ON. A timer overflow indicates that the current has not reached the selected peak, for example because the LED string is not attached. This time is computed by the (66% + 20%)*2 equation 3. On exit, the PWM output is set to 0 V, switching OFF the MOSFET. The next state is the non-sequential state S(0).

  Event_Jmp = N/A.

  Event_Seq = Event(1) OR TIM_Cnt ("and/or" bit guarantees a non-sequential jump).

  On exit to S(0): reset timer, set PWM to 0.

- S(3): overcurrent protection

  When Event_Jmp is received during state S(1), the SMED has found an OCP (overcurrent protection) event and goes to S(3). During the transaction, the PWM output line is set to 0 V in order to turn the MOSFET OFF. The S(3) timer is set to its maximum value (0x1F0 = 5 µs) to protect the hardware. When the SMED timer overflows, the SMED advances to sequential S(0) state. The transaction also

generates an internal interrupt and the software can monitor the OCP and stop the SMED evolution.

Event_Jmp = N/A.

Event_Seq = TIM_Cnt.

On exit to S(0): reset timer, generate SW interrupt.

- HOLD S(0)

  This state starts from S(1) when the software needs to stop the SMED evolution, due to the dimming duty cycle OFF time. HOLD is also used when OCP is detected during S(1). The PWM output line on HOLD state is 0 and the MOSFET is OFF. When the FW sets the SWx bit to 1, the SMED goes back to S(1).

Note: The register associated to the SMED is updated when either the SMED is on HOLD state (during the dimming off-time) or the output PWM line is zero (S0 state).This approach guarantees that the PWM output is always set to 0 when the new parameters are applied.

4.3 Input/output signals

The SMED uses the following list of input/output signals and pins:

- PWMx: PWM output signal

  The signal is fed into the MOSFET's driver input to control the MOSFET's switching state. The PWM operates according to the T_{ON} and T_{OFF} time faced in *Section 2*. When the PWM is high, the MOSFET is ON and vice versa. When the supply voltage of the STLUX385A is 5 V, the output signal can be directly connected to the MOSFET. Instead, when the supply voltage is 3,3 V, the PM8834 buffer is required.

- CPPx: comparator input signal

  CPPx receives the current sense signal and compares it to the internal DAC value. When the voltage on CPPx pin is higher than the DAC value, the output of the internal analog comparator goes high, and vice versa. The comparator output is the event(1) internal signal. Event(1) is programmed at level mode so that the SMED can detect immediately the eventual anomaly during S(1). If edge mode was used and the anomaly happened outside S(1) then S(1) wouldn't receive the edge event. There are only two SMED states that check for this event: fault zone (S1) and current limit (S2).

- SWx: software event

  Used to start and pause the SMED. When the SMED is paused, the PWM output has to be set to 0. The SWx event is used during S(0) to pause the SMED and during HOLD to restart. The event is programmed at level mode and is used by the software to define the dimming duty cycle period. Thanks to the event mode, the SW trigger is detected by S(0) even if the SW edge happens while the SMED is in a different state then S(0). In this case, the SMED completes the cycle, reaches S(0), captures the SW level and goes on HOLD.

- DIGINx: zero current detection

  The pin is used for zero current detection while it is in transition mode. At the moment the transition mode is a proposal and the pin is not managed.

- V_{COM0}

  The voltage applied to the pin is read by the STLUX385A ADC and it is used to compute the T_{ON} time and the T_{OFF} time. The voltage is captured 100 ns after the

beginning of the ON portion of the dimming cycle as the current is expected to reach the correct value.

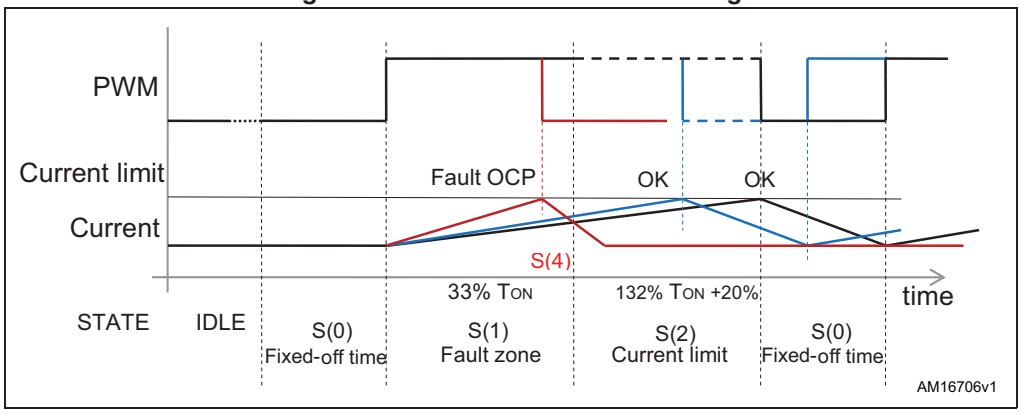
- V_{LED_PW}

  The voltage applied to this pin is read by the STLUX385A ADC and is used to compute the T_{ON} time and the T_{OFF} time. It is acquired at the same time of the V_{COM0} voltage using the 8 values, four for V_{COM0} and four for V_{LED_PW} - interleaved, stored by the ADC in registers ADC_DATH< 7-0 > and ADC_DATL< 7-0>.

4.4 SMED state time evolution

The following diagram describes the SMED state time evolution. Only one PWM period is illustrated.

Figure 6. SMED state evolution. Timing



The colors identify the case of a correct or incorrect situation:

- The black trace is the output current observed during a correct setup where the SMED evolution is triggered by timer events only. In this case, no event_Seq or event_Jmp are received; note that S2 may not have reached the peak level during S(2).
- The blue trace represents the situation where the current reaches the limit point (event_Seq) during S(2).
- The red line shows the SMED evolution when an OCP event (event_Jmp) is detected during S1. The SMED is programmed to generate an interrupt when an OCV condition happens. The main program receives the interrupt and stops the SMED.

5 Firmware implementation

The following section offers a view of the firmware implementation. For more details about the firmware, refer to the source code. The compiled portion of the firmware core is 3.5 Kbyte and the code size is ~12 Kbyte. Most of the code size is allocated to the debugging machine and general user interface (see *Section 7.3*).

The PWM algorithm and the OCP controls are entirely implemented in the SMED block, leaving the CPU free for control tasks such as: slow varying algorithmic compensations, the command line, and interrupt management.

5.1 Dimming algorithm

The dimming algorithm controls the brightness of the LED string by modulating the amount of time, the LED is switched ON during a ~5 ms period. This period defines the dimming cycle, which operates with a ~200 Hz frequency. The longer the LED is ON during a single cycle, the brighter the LED results to human eyes. To turn ON a LED, the algorithm simply activates the SMED associated by setting the SMED's SWx event to active state.

Once the desired on-time period is passed, the LED is turned OFF and the algorithm waits for the beginning of the next dimming cycle to turn ON the LED again.

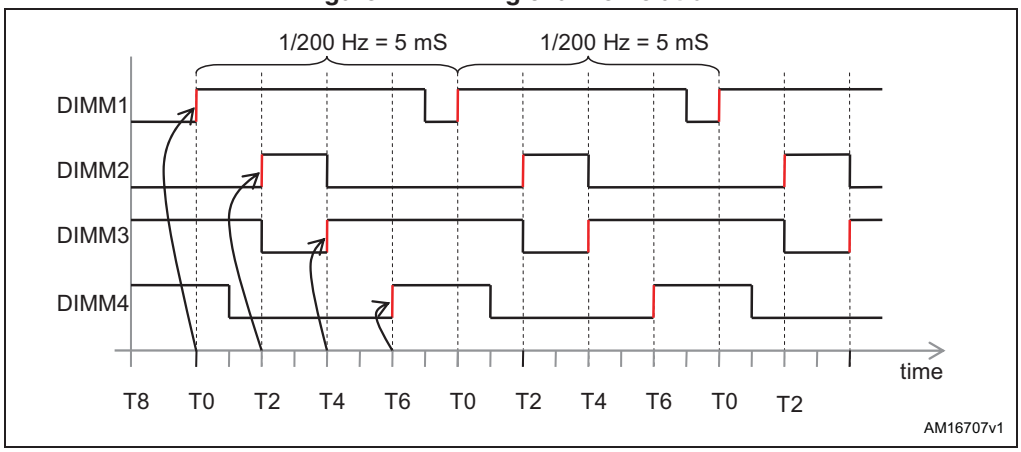
To optimize power, the dimming cycle in channel x is 90 degrees shifted respect to channel x-1. LED chains can be configured with different on-time length; however the dimming start period is fixed by design. In this way, the LEDs are always turned ON at different times, reducing current peaks on the power supply.

The LED on-time is instead a multiple of 20 µs and is based on the STMR timer, which itself has 1 µs granularity.

The software stores the time-on length in the `onoff[8][2]` structure, where the first byte indicates the channel and the PWM action (turn ON or OFF the LED) and the second byte specifies the time (up to 256 time units = 256 x 20 µs = 5120 µs) before the next event.

The next example shows the activity of LED channels with different dimming on-times: DIMM1 is active at 87,5%, DIMM2 is active at 25%, DIMM3 is active at 75% and DIMM4 is active at 37.5%. When the dimming is ON the SMED output PWM is active and the LED light is ON.

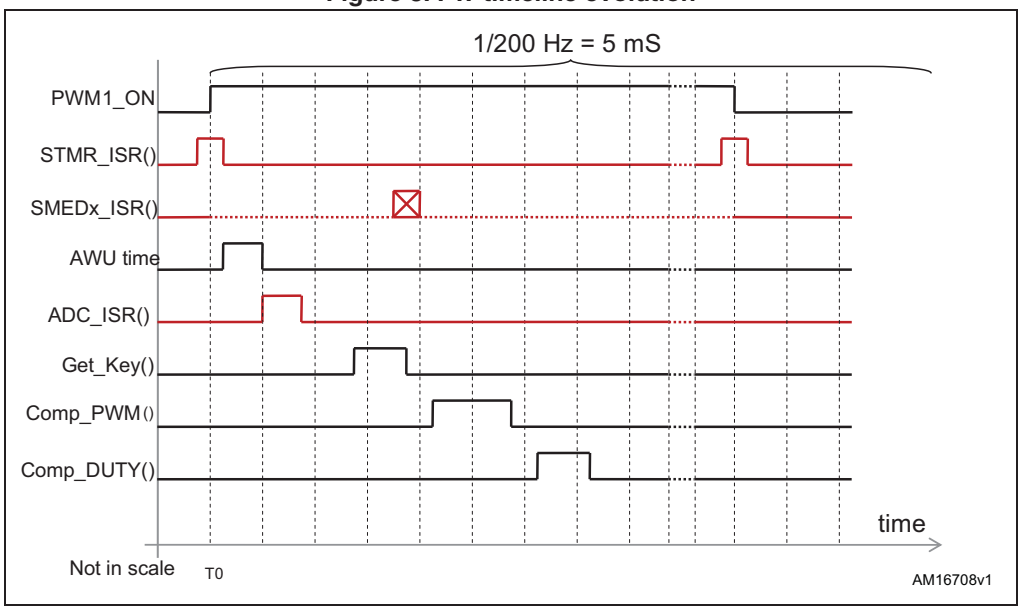
Figure 7. Dimming channel relation



5.2 Firmware evolution

The following section describes the software timing characteristics as shown in the following picture. The picture shows the relationship between two consecutive `STMR_ISR()` routines and the other principal subroutines. The colors define if the routines run in interrupt mode (colored) or are called from main (black). The PMW1_ON line is the dimming period: when the line is high, the PWM signal is high too.

Figure 8. FW timeline evolution



The lines are here described:

- `STMR_ISR()`: STMR timer interrupt.

  The STMR timer has a granularity of 20 µs (STMR unit). The timer is used to start and stop the dimming period for each SMED and triggers the SWx bit on the MSC SMSWEV register. On every interrupt, the timer is reprogrammed so that the next

interrupt is received when the next dimming even is required, as specified in the `onoff[][]` structure.

When the interrupt routine sets a SMED ON, the startup flag for the ADC delayed acquisition (using the AWU peripheral) is also set.

- `SMEDx_ISR()`: SMED interrupt

  The interrupt is used to stop the SMED when an OCP problem is detected. The interrupt is generated by a SMED on exit from S3. The SMED is in S3 only if event(1) is received during S1. The interrupt routines stop the SMED by changing the SWx bit and giving the fault signal to the main function, which turns on the red "fault" LED. The SMED is reactivated on its next dimming cycle and if the OCP problem is not removed then the SMED evolves into S3, triggering the `SMEDx_ISR` interrupt again.

  Auto-wakeup unit. The auto-wakeup unit is used while the CPU waits for the current startup and LED voltage settlement time before starting the ADC acquisition. For a correct FOT implementation the ADC measures the V_{COMx} voltage when the PWM is ON and when the V_{LED} has a correct value. This time is fixed to around 100 µs after the SMED start. When the AWU time expires, the interrupt starts the ADC channel acquisition.

- `ADC_ISR()`: ADC interrupt

  The routine samples, for each channel, V_{LED_PW} and V_{COMx}, 4 times respectively. The values are then averaged and stored. The ADC sampling routine occurs 100 µs after the start of the PWM dimming. Should the PWM dimming time be smaller than 100 µs, the ADC samples incorrect voltage values and therefore the samples are discarded. This protection (inconsistent voltage protection) is implemented via the `SMED_Active` bit.

  The interrupt routine uses the `adc_eoc` flag to inform the main function that the ADC operation is completed and that new data is available. As a consequence, the software discards the ADC value. The interrupt routine uses the `adc_eoc` flag to inform the main function that the ADC operation is completed and that new data is available.

- `Get_Key()`

  This routine is used to detect if the board buttons are pressed. The STLUX385A uses the ADC (ADCIN[7], pin 31) to sense the button status. Since the ADC is used also to read V_{COMx} and V_{PWR}, a protection is set to avoid conflicts.

- `Comp_PWM()`

  It computes the new SMED's operating parameters as calculated by the adaptive voltage compensation mechanism.The new values are updated on the SMED registers only during the next STOP phases (using the `upd_flags` variable with `UPD_PWMx` flags).

- `Comp_DUTY()`

  This routine is invoked when the user changes the dimming period. The routine computes a new dimming period based on the `ontime[]` structures and stores the result on `tmp_onoff[][]` structures. The update is performed at the beginning of the next dimming period (using the `upd_flags` variable with `UPD_DUTY` flags) and when the dimming is OFF. During the update, the `tmp_off[][]` structure is copied into the `onoff[][]` structure.

The tasks displayed in the previous picture refer to the PWM and dimming control operations only. The STLUX385A firmware handles other tasks such as the command line,

which uses two interrupts to read or write characters to the serial line. The serial interrupts use the lowest priority level among all the other sources (STMR, AWU, SMED and ADC).

5.3 Printf

The STLUX385A firmware implements the standard "`printf()`" subroutine, which takes several cycles to expand the printed strings and the operation delay real-time events. It is suggested the use of "`printf`" in production code is minimized or removed.

5.4 Low power modes

The STLUX385A may enter low power modes through the instruction WFI (wait for interrupt), where the internal CPU is stopped until an interrupt is received.

The "wi" console command instruction allows the user to use the WFI instruction (see *Section 7.4.10*).

The user may implement more aggressive low power schemes via new software routines. For example, the STLUX385A power consumption efficiency could be increased by disabling the SMEDs and the 96 Mhz PLL while the LED string is completely switched OFF.

6 Protection and regulation algorithm

This section describes the protection and regulation algorithms implemented in the STLUX385A. Every channel is independent from the others and the same logic applies to all channels.

The STLUX385A implements software and hardware protections aimed to preserve the external hardware in case of abnormal events or variations. The STLUX385A doesn't need external active analog circuitry on the board as the firmware replaces them. Once an error is recognized, the FW stores the error code and keeps sensing the system to verify that the problem has been fixed. The error code can be verified typing the "st" command (see *Section 7.4.6*). Protections can be summarized according to different priority levels.

6.1 Overcurrent protection

The overcurrent protection (OCP), which preserves the external circuit from damage due to a component fault, is the most important protection and requires a short detection and reaction time. The STLUX385A implements OCP directly in the SMED hardware (through states S1 and S2) and doesn't require any high priority firmware intervention.The internal SMED clock guarantees a detection time of ~10 ns (Fck_smed = 96 Mhz) however the external circuit response time pushes the overall detection time to 200 ns. When an OCP is detected, the SMED stops its PWM output and generates an interrupt. The firmware can then execute the recovery procedure by periodically restarting the PWM line.

6.2 Initial estimation

During the startup phase, the input power voltage is monitored by the Set_check_V() routine. This routine reads the power voltage (on ADC channel zero) and stores it into the global variable `Vpwr[]`. The voltage read back is stored into all channel locations (0 to 3). The routine Set_check_V() estimates the voltage drop generated by the LEDs by using the number of LEDs as specified by the user and the maximum and minimum predefined LED voltage drops. The following formula is used:

Equation 11

$$\frac{wk\_Volt[x].Min\_vl + wk\_Volt[x].Max\_vl}{2}$$

where x is the LED channel. The estimated voltage drop is used for the initial calculation of the T_{OFF} and T_{ON} values.

6.3 Adaptive voltage compensation

The adaptive compensation is a technique that monitors the system and changes the operational parameters to guarantee that the configured average current always flows through the LED. As the average current depends on the T_{ON} and T_{OFF} values, the STLUX385A recalculates the values at the beginning of every dimming ON cycles. The adaptive compensation is activated with the serial command "au x 1".