Introduction

This user manual provides complete information for SW developers about a set of guide examples useful to get familiar developing applications for the STLUX and STNRG digital controllers and their peripherals.

The STLUX family of controllers is a part of the STMicroelectronics® digital devices tailored for lighting applications. The STLUX controllers have been successfully integrated in a wide range of architectures and applications, starting from simple buck converters for driving multiple LED strings, boost for power factor corrections, half-bridge resonant converters for high power dimmable LED strings and up to full-bridge controllers for HID lamp ballasts. The STLUX natively supports the DALI via the internal DALI communication module (DCM). The DALI is a serial communication standard used in the lighting industry.

The STNRG devices are a part of the STNRG family of the STMicroelectronics digital devices designed for advanced power conversion applications. The STNRG improves the design of the STLUX family to support industrial power conversion applications such as the PFC + LLC, interleaved LC DC-DC, interleaved PFC for smart power supplies as well as the full-bridge for pilot line drivers for electric vehicles.

The heart of the STLUX (and consequently the STNRG where not differently specified) is the SMED (“State Machine, Event Driven”) technology which allows the device to operate several independently configurable PWM clocks with an up to 1.3 ns resolution. An SMED is a powerful autonomous state machine which is programmed to react to both external and internal events and may evolve without any software intervention. The SMED even reaction time can be as low as 10 ns, giving the STLUX the ability of operating in time critical applications.

SMEDs are configured and programmed via the STLUX internal low-power microcontroller (STM8). This user manual describes the whole set of examples provided in this kit.
Contents

1 Reference documents ............................................................... 4
2 Acronyms ................................................................................. 5
3 Examples kit ............................................................................. 7
4 Example I .................................................................................. 8
5 Example II ................................................................................. 10
6 Example III ............................................................................... 11
7 Example IV ............................................................................... 12
8 Example V .................................................................................. 14
9 Example VI ................................................................................ 17
10 Example VII ............................................................................. 21
11 Example VIII ............................................................................ 26
12 Revision history ....................................................................... 29
List of figures

Figure 1. SMED 0 state machine ................................................. 9
Figure 2. Example II application scheme .................................. 10
Figure 3. Example IV application scheme ................................. 12
Figure 4. PWM0 constant frequency with variable duty cycle .......... 12
Figure 5. Example IV - SMED 0 state machine ......................... 13
Figure 6. Example V application scheme ................................. 14
Figure 7. PWM5 and output voltage regulation ......................... 14
Figure 8. SMED 5 input and CP2 and CP3 comparators configuration . 15
Figure 9. Example V - SMED 5 state machine .......................... 16
Figure 10. Example VI application scheme ...................... 17
Figure 11. Example VI - PWM5 and output voltage regulation ...... 18
Figure 12. Example VI - SMED 5 input and CP2 and CP3 comparators configuration . 18
Figure 13. Example VI - SMED 5 state machine ....................... 19
Figure 14. PWM5 and output voltage regulation - oscilloscope plot - short charge time 20
Figure 15. PWM5 and output voltage regulation - oscilloscope plot - long charge time 20
Figure 16. Example VII - PWM5 and output voltage regulation ...... 21
Figure 17. Example VII - SMED 5 state machine ..................... 22
Figure 18. SMED 2 state machine (same applies to SMED 3) ........ 23
Figure 19. SMED configuration .............................................. 24
Figure 20. PWM2, PWM3, PWM5 and output voltage regulation - oscillator plot 25
Figure 21. Example VIII - PWM5 and output voltage regulation ....... 26
Figure 22. Example VIII - SMED 5 state machine ..................... 27
Figure 23. PWM5 and output voltage regulation - oscillator plot .... 28
1 Reference documents

- For hardware information on the STLUX and STNRG controllers and product specific SMED configuration, please refer to the STLUX and STNRG product datasheets and reference manual (RM0380).
- For information about the debug and SWIM (single-wire interface module) refer to the “STM8 SWIM communication protocol and debug module” user manual (UM0470).
- For information on the STM8 core and assembler instruction please refer to the “STM8 CPU programming manual” (PM0044).
- For information on the SMED configurator please refer to the “SMED configurator v2.0 for STLUX™ and STNRG digital controllers” user manual (UM1981).
- For information on the STLUX peripheral library please refer to the “Standard peripheral library for STLUX™ and STNRG digital controllers” user manual (UM2001).
- For information on the STEVAL-ILL075V1 or STEVAL-ISA164V1 evaluation boards please refer to the product datasheets.
## Acronyms

A list of acronyms used in this document:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACU</td>
<td>Analog comparator unit</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog-to-digital converter</td>
</tr>
<tr>
<td>ATM</td>
<td>Auxiliary timer</td>
</tr>
<tr>
<td>AWU</td>
<td>Auto wake-up unit</td>
</tr>
<tr>
<td>BL</td>
<td>Bootloader - used to load the user program without the emulator</td>
</tr>
<tr>
<td>CCO</td>
<td>Configurable clock output</td>
</tr>
<tr>
<td>CKC</td>
<td>Clock control unit</td>
</tr>
<tr>
<td>CKM</td>
<td>Clock master</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit</td>
</tr>
<tr>
<td>CSS</td>
<td>Clock security system</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital-to-analog converter</td>
</tr>
<tr>
<td>DALI</td>
<td>Digital addressable lighting interface</td>
</tr>
<tr>
<td>ECC</td>
<td>Error Correction Code</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite state machine</td>
</tr>
<tr>
<td>FW</td>
<td>Firmware loaded and running on the CPU</td>
</tr>
<tr>
<td>GPIO</td>
<td>General purpose input/output</td>
</tr>
<tr>
<td>HSE</td>
<td>High-speed external crystal - ceramic resonator</td>
</tr>
<tr>
<td>HSI</td>
<td>High-speed internal RC oscillator</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-integrated circuit interface</td>
</tr>
<tr>
<td>IAP</td>
<td>In-application programming</td>
</tr>
<tr>
<td>ICP</td>
<td>In-circuit programming</td>
</tr>
<tr>
<td>ITC</td>
<td>Interrupt controller</td>
</tr>
<tr>
<td>IWDG</td>
<td>Independent watchdog</td>
</tr>
<tr>
<td>LSI</td>
<td>Low-speed Internal RC oscillator</td>
</tr>
<tr>
<td>MCU</td>
<td>Microprocessor central unit</td>
</tr>
<tr>
<td>MSC</td>
<td>Miscellaneous</td>
</tr>
<tr>
<td>PM</td>
<td>Power management</td>
</tr>
<tr>
<td>RFU</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>ROP</td>
<td>Read-out protection</td>
</tr>
<tr>
<td>RST</td>
<td>Reset control unit</td>
</tr>
<tr>
<td>RTC</td>
<td>Real-time clock</td>
</tr>
</tbody>
</table>
Table 1. List of acronyms (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMED</td>
<td>State machine event driven</td>
</tr>
<tr>
<td>STMR</td>
<td>System timer</td>
</tr>
<tr>
<td>SW</td>
<td>Software, is the firmware loaded and running on the CPU (synonymous of FW)</td>
</tr>
<tr>
<td>SWI</td>
<td>Clock switch interrupt</td>
</tr>
<tr>
<td>SWIM</td>
<td>Single-wire interface module</td>
</tr>
<tr>
<td>UART</td>
<td>Universal asynchronous receiver/transmitter</td>
</tr>
<tr>
<td>WWDG</td>
<td>Window watchdog</td>
</tr>
</tbody>
</table>
3 Examples kit

This examples kit is composed of five guide examples targeting the STLUX and three more examples targeting the STNRG. These examples are incrementally built and are thought to be a starting point to get in touch with the STLUX toolset and libraries for handling the STM8 core, peripherals and SMEDs. All the STLUX guide examples can be tested on the STEVAL-ILL075V1 evaluation board. All the STNRG guide examples can be tested on the STEVAL-ISA164V1evaluation board.
4 Example I

This is the first and the simplest example. The aim of the code is to build an equivalent “Hello World!” application for the STLUX but trying to use also its strength point, the SMEDs. In a few words the goal of the application is to configure the SMED 0 in order to drive the output pin toggling at a fixed frequency and fixed duty cycle. As a results on the pin 1, there will be an output signal toggling with a 50% duty cycle at a 48 KHz frequency.

The aim of the example is also to try to get familiar with the firmware development for the STLUX and STM8 and learning how to drive peripherals through the APIs included in the STLUX libraries. For this example in particular, the stlux_clk library APIs are used to properly configure the SMED clock to the maximum frequency of 96 MHz. The stlux_smed library APIs are used to start the SMED 0.

Another key component of the STLUX toolset, the STLUX SMED configurator GUI is used here to set the SMEDs configuration. The designed SMED configuration can also be saved in a *.prj format file. In particular in this example only the SMED 0 is used and it is configured according to the scheme in Figure 1 generated using the tool and saved in the Example_I.prj file.
Figure 1. SMED 0 state machine
5 Example II

In order to guide you step by step to a more realistic application, we extended the first example by introducing the use of the system timer and the ADC peripherals in the second example.

The system timer (STMR) consists of a 16-bit autoreload upcounter driven by a programmable prescaler. Specific APIs have been created and integrated in the stlux_stmr library to configure and pilot this timer. Here it is used as a time base to generate data sampling at a given frequency. The timer clock is prescaled by 4 and an interrupt is raised each time 200 tics the period event and therefore input data are sampled through the ADC at a 20 KHz frequency.

The ADC is a 10 bit successive approximation analog-to-digital converter. In this example four of the possible eight channels (channel 0 to channel 3) are used to sample input data amplified by a 1.0 gain factor (the other possible value is 4.0). The ADC clock is set to 6 MHz frequency.

The main code properly initializes all the peripherals and clocks and then starts the system timer and the ADC sampling. It tests the x0 input variable and when it becomes greater then a given threshold, it starts the SMED 0.

Figure 2. Example II application scheme
6 Example III

The third example furtherly extends the small application described in the second example adding another timer among the STLUX specific features, the auxiliary timer (AUXTIM).

The auxiliary timer is not a real peripheral but can be considered as a virtual peripheral that's physically build grouping some functionality already existing in the silicon device and spread on different IPs to optimize the silicon cost.

Basically it works as a light timer that can be configured to generate interrupts at a set frequency derived by the system clock. In order to make this functionality easily usable and configurable, a set of APIs has been built and collected in the stlux_atm library. For further details on the AUXTIM APIs please refer to the “Standard peripheral library for STLUX™ and STNRG digital controllers” user manual (UM2001).

The main code properly configures the auxiliary timer to work at a frequency that's a fraction of the HSI system clock (16 MHz). In particular a frequency of \( \frac{16}{8 + 1} = 1.78 \text{ MHz} \) has been chosen to generate interrupts. The interrupt handler only clears the interrupt status flag and could be a starting point to develop applications where a specific routine must be performed at a given frequency.
The example IV aims to create an application generating a PWM signal as a function of the analog input voltage. As shown in Figure 4, the PWM0 will have a fixed frequency of 50 KHz but a variable duty cycle according to the sensed input signal.

In order to dime the input voltage, the analog circuit shown in Figure 3 is implemented by default on the example IV area of the STEVAL-ILL075V1 and STEVAL-ISA164V1 evaluation boards.

When the input voltage is below the lower threshold set to 200 mV, the PWM0 is OFF. If the input voltage $V_{\text{in}}$ ranges from 200 mV to 1 V, the PWM0 toggles at a fixed frequency of 50 KHz, but its duty cycle ranges from 2.5% to 97.5% with a 10 ns step leaving, so a minimum off-time equals to 500 ns. When $V_{\text{in}}$ overcomes the upper threshold set to 1 V, the PWM0 is OFF. The implemented FSM is shown in Figure 5.
The input voltage measure is performed by the ADC at a given frequency set by the system timer. The STMR interrupt handler performs the regulation loop reading the Vin sampled value and computing the PWM0 according to the behavior described above.

The main code properly configures the I/O pins, the ADC, the SMED 0. It also sets the STMR to work at a frequency that's a fraction of the HSI (16 MHz). In particular a frequency of 10 KHz has been chosen to generate interrupts. Nothing more is needed as the SMED 0 will autonomously regulate the PWM0 to follow the behavior specified above.

Now when the input voltage falls off of the [200 mV; 1 V] range, the SMED 0 stops. Possible evolution of this example could be modifying the firmware so that outcoming 1 V, the SMED 0 keeps going on keeping the PWM duty cycle constant. This “protection” is handled using the same loop regulation or use the input event (CP0 and DAC0 for example) directly on the SMED.
The goal of the fifth example is to learn how to generate a PWM signal as a function of two comparators. In this case we want to control the PWM5 variable frequency so to keep an analog output voltage between a high threshold and a low threshold respectively set to 410 mV and 246 mV by the CPP2 and CPP3 references. In order to dim the Vout fall and hence the capacitor discharge, the analog circuit shown in Figure 6 is implemented by default on the example V area of the STEVAL-ILL075V1 and STEVAL-ISA164V1 evaluation boards.

As shown in Figure 7, in order to keep Vout over 246 mV, the PWM5 is turned on when voltage reaches the low threshold reference sensed by the DAC3 and CPP3. Also keeping Vout under 410 mV requires the PWM5 to be turned off when voltage hits the high threshold reference sensed by the DAC2 and CPP2. The SMED 5 input and analog comparators...
configuration can be easily handled also via the SMED Configurator as shown in Figure 8. No more firmware interaction is needed as the output control is automatically handled by the SMED 5 FSM described in Figure 9.

Figure 8. SMED 5 input and CP2 and CP3 comparators configuration
The main code simply takes care of initializing the I/O pins, it also enables the PLL generating the 96 MHz clock for the SMEDs. Then it properly sets the DAC, the CPP2 and CPP3 analog comparators so to configure them according to the high and low threshold values. Finally it enables the SMED 5. As everything is set and since now everything will be controlled by the SMED 5, the STM8 processor can be halted to reduce power consumption.
9  Example VI

As for the previous example, the goal of the sixth example is to learn how to generate a PWM signal as a function of two comparators. In this case we want to control the PWM5 variable frequency so to keep an analog output voltage between a high threshold and a low threshold respectively set to 410 mV and 246 mV by the CPP2 and CPP3 references. We also want to exploit a specific STNRG feature setting on the two comparators a hysteresis level to add a certain tolerance on the upper and lower voltage threshold. The hardware involved is set as in Figure 10 and it is implemented by default on the example V area of the STEVAL-ISA164V1 evaluation board.

Figure 10. Example VI application scheme

As shown in Figure 11, in order to prevent Vout falling below 246 mV minus hysteresis, the PWM5 is turned on when voltage reaches the low threshold reference sensed by the DAC3 and CPP3. Also preventing Vout to rise above 410 mV plus hysteresis requires the PWM5 to be turned off when voltage hits the high threshold reference sensed by the DAC2 and CPP2. No more firmware interaction is needed as the output control is automatically handled by the SMED 5 FSM described in Figure 13 and by the comparators set as shown in Figure 12.
The main code simply takes care of initializing the I/O pins, it also enables the PLL generating the 96 MHz clock for the SMEDs. Then it properly sets the DAC, the CPP2 and CPP3 analog comparators so to configure them according to the high and low threshold values. Finally it enables the SMED 5. As everything is set and since now everything will be
controlled by the SMED 5, the STM8 processor can be halted to reduce power consumption.

**Figure 13. Example VI - SMED 5 state machine**

What we can see checking with the oscilloscope is the output voltage signal as shown in Figure 14 and Figure 15. Please note how PWM output frequency varies while acting on the trimmer.
Figure 14. PWM5 and output voltage regulation - oscilloscope plot - short charge time

Figure 15. PWM5 and output voltage regulation - oscilloscope plot - long charge time

Acting on the trimmer, we can make the charge time longer.
The goal of the seventh example is to learn how to save events timestamps and use them in your code, maybe to modify SMED timers. In this case we want to dump the counter at the exact time the output voltage falls below the CPP3 lower threshold set to 328 mV.

In this example the PWM5 variable frequency keeps the analog output voltage between the high threshold set to 410 mV by the CPP2 and a low threshold given by a fixed off-time. The timestamp of the Tcpp3 (here indicated as td) value is triggered by an interrupt request generated by the SMED 5 input signal as shown in Figure 16.

As shown in Figure 17, keeping Vout under 410 mV requires the PWM5 to be turned on when voltage hits the high threshold reference sensed by the CPP2. The lower threshold is set by a fixed time T1 during which the PWM5 is turned off. Since the interrupt generation is enabled on SMED 5 CP3 input, when the analog output voltage hits the threshold set by the CPP3, and an interrupt request is generated. Therefore while serving the interrupt request, a timestamp of the current timer value can be done. This time Tcpp3 will be used to set the toggle time for the PWM2 and PWM3 as shown in Figure 20 on page 25.
The SMED 2 and SMED 3 are initialized according to the scheme shown in Figure 18. A simple state machine makes each PWM toggle with a fixed time initialized to the minimum. This will be properly set by the SMED 5 right before starting these SMEDs.
The main code takes care of enabling the clock for the SMEDs. Then it properly sets the DAC, the CPP2 and CPP3 analog comparators so to configure them according to the high and low threshold values. Then it configures the SMED 2, SMED 3 and SMED 5 according to the configuration set in Figure 20. Finally it enables the SMED 5. Then it waits for the first interrupt to be served so that the SMED 2 and SMED 3 toggle times are initialized and finally it also starts them both. As everything is set and since now everything will be controlled by the SMED 5, the processor only has to serve interrupt requests with low-power consumption as a result.
Output signals can be seen below in Figure 20 for output voltage and generated PWMs.
Figure 20. PWM2, PWM3, PWM5 and output voltage regulation - oscillator plot
11 Example VIII

The goal of the example VIII is to learn how to exploit a specific STNRG feature: the ADC hardware trigger activated by SMEDs. So, starting from the output voltage regulated by the CPP2 and CPP3 analog comparators to be comprised between 410 mV and 246 mV, we are going to check when Vout falls below the lower threshold and to trigger right after an ADC conversion to sense Vout. The sensed value is going to be used as additional Toff value for the consequent status S3.

Figure 21. Example VIII - PWM5 and output voltage regulation

In order to achieve the desired configuration and to properly generate the PWM5, the SMED 5 is configured according to Figure 22.
The main code takes care of enabling the clock for the SMED 5. Then it properly sets the DAC, the CPP2 and CPP3 analog comparators so to configure them according to the high and low threshold values. Then it configures and powers up the ADC performing the proper initialization sequence. Please note that also the ADC interrupt must be enabled in order to handle the hardware trigger conversion generated by the SMED. Finally it enables the SMED 5. As everything is set and since now everything will be controlled by the SMED 5, the processor only has to serve interrupt requests with low-power consumption as a result.
Please note in Figure 23 the states sequence. As soon as you tweak the trimmer, you will see t0 and t2 changing accordingly. Time t3 though won’t change as its duration is based on the ADC acquisition of the CMP3 lower threshold set to 246 mV. You can see S3 takes about 142 us Toff. This is due to the following formula:

\[
246 \text{ mV} / \left[ \frac{1.25}{1023} \right] = x_c
\]

\[
142 \mu s = x_c \times 2^5 \times t_{\text{clk}}
\]
12 Revision history

Table 2. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-Jun-2016</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2016 STMicroelectronics – All rights reserved