UM1763
User manual

Description of STLUX385A examples kit

Introduction

This user manual provides complete information for SW developers about a set of guide examples useful to get familiar developing applications for the STLUX385A digital controller and its peripherals.

The STLUX385A is an STMicroelectronics® digital device tailored for lighting applications. The heart of the STLUX is the SMED (“State Machine Event Driven”) technology which allows the device to operate several independently configurable PWM clocks with 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 examples provided by this document will help you to understand the SMEDs and how to program them in your applications. All the examples have been developed and tested on an STEVAL-ILL068V1 evaluation board.

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

Reference documents

- For hardware information on the STLUX385A controller and product specific SMED configuration, please refer to the STLUX385A product datasheet.
- For information on programming, erasing and protection of the internal Flash memory please refer to the STM8 Flash programming manual (PM0047).
- 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 UM1760 “STLUX™ SMED configurator 1.0" user manual.
- For information on the STLUX385A peripheral library please refer to the STLUX™ peripheral library user manual (UM1753).
- For information on the STEVAL-ILL068V1 evaluation board please refer to “Design resources” tab of the STEVAL-ILL068V1 product folder on www.st.com.

Contents

List of figures

1 Acronyms

A list of acronyms used in this document:

Table 1. List of acronyms

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

Table 1. List of acronyms (continued)

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

2 Examples kit

This examples kit is composed of five guide examples. 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 guide examples can be tested on the STEVAL-ILL068V1 evaluation board.

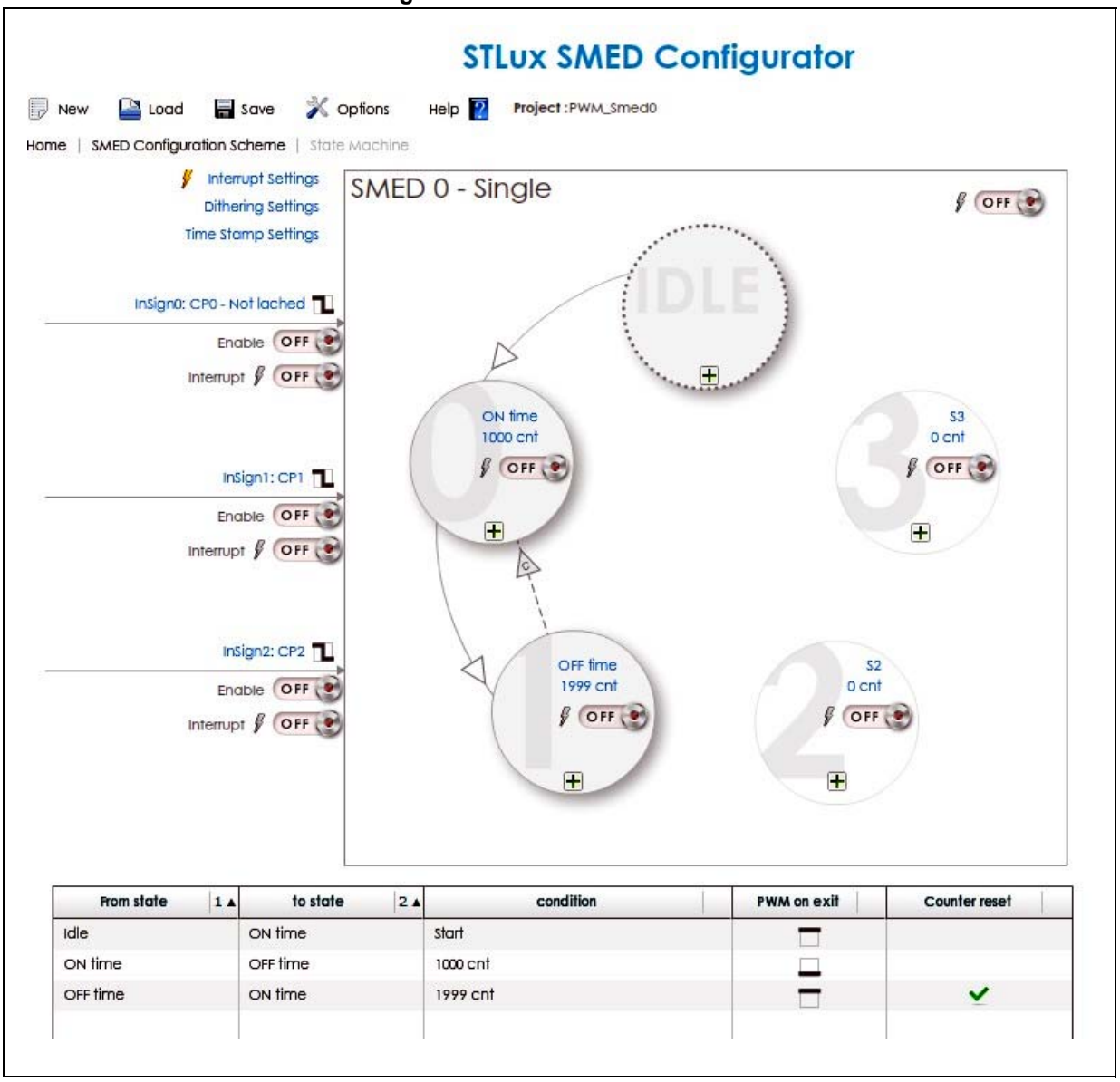
3 Example I

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 PWM_Smed0.prj file.

Figure 1. SMED 0 state machine



From state 1	to state 2	condition	PWM on exit	Counter reset
Idle	ON time	Start		
ON time	OFF time	1000 cnt		
OFF time	ON time	1999 cnt		✓

4 Example II

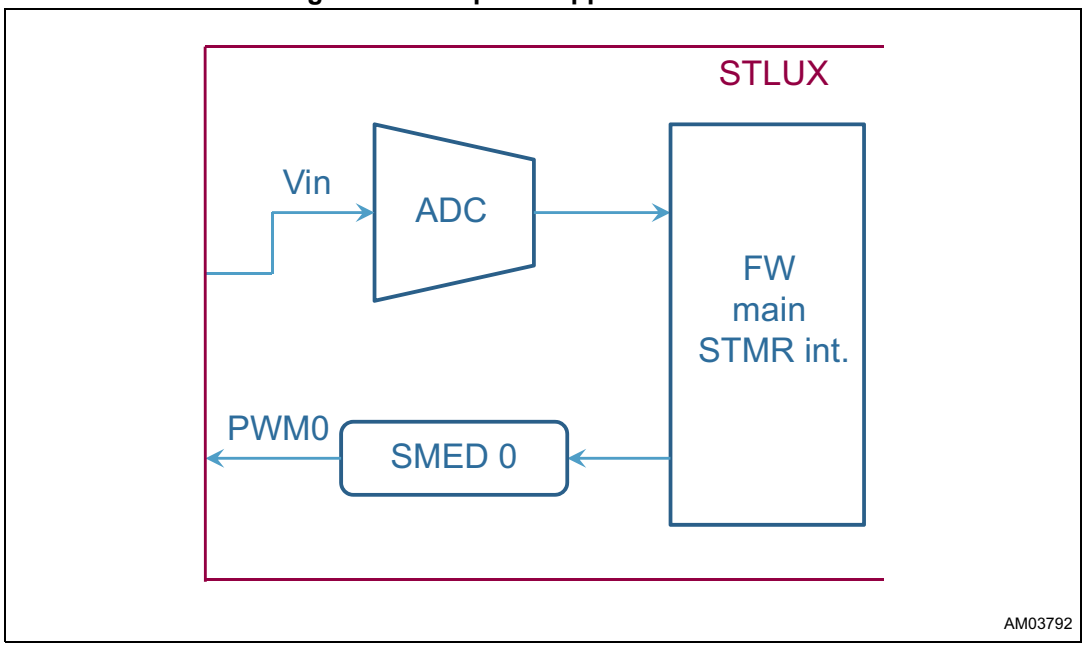
In order to guide you step by step to a more realistic application, we extended the first example by introducing the use of the system timer and the ADC peripherals in the second example.

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 02 application scheme



5 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).

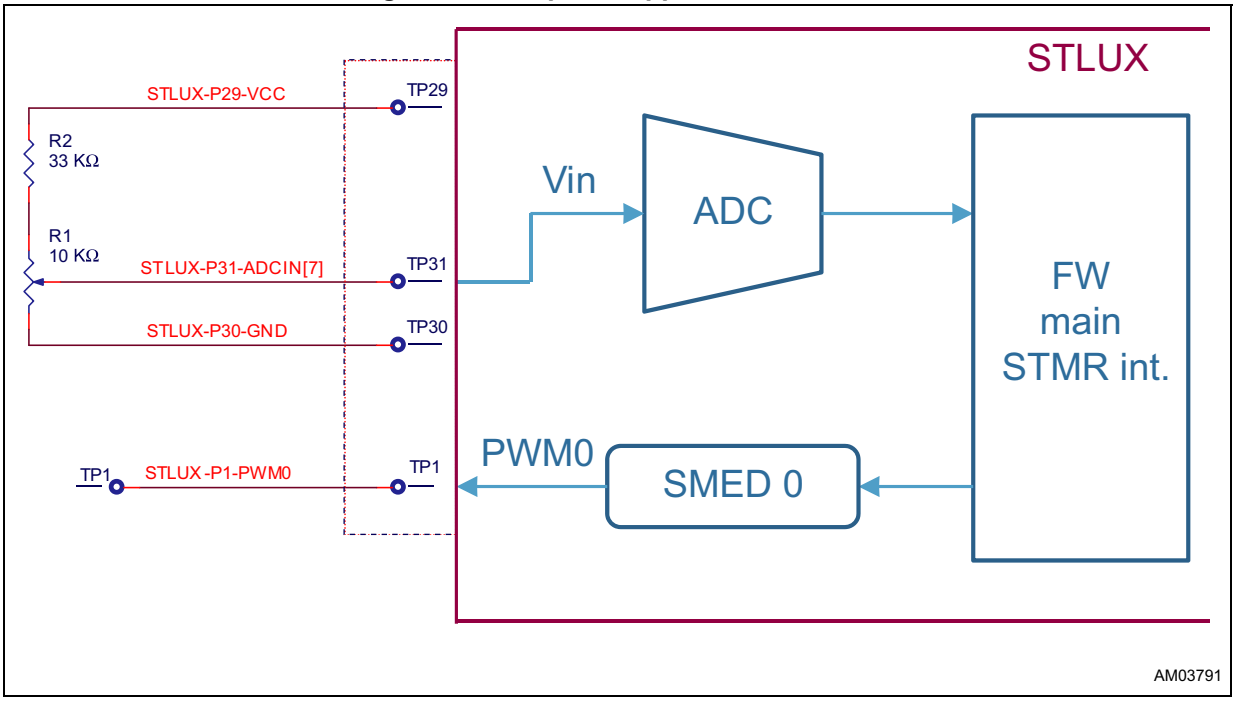
The auxiliary timer is not a real peripheral but can be considered as a virtual peripheral that's physically build grouping some functionality already existing in the silicon device and spread on different IPs to optimize the silicon cost.

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.

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 [16/(8 + 1)] = 1.78 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.

6 Example IV

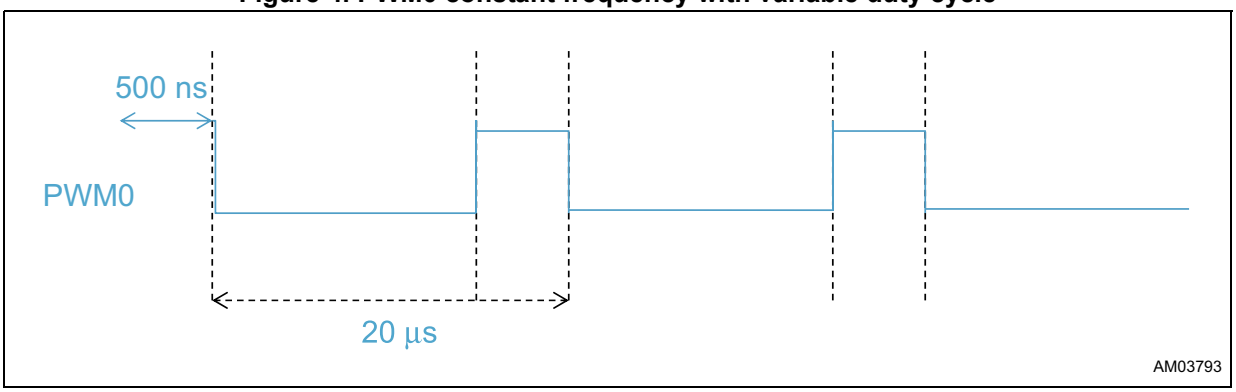
Figure 3. Example 03 application scheme



The example IV aims to create an application generating a PWM signal as a function of the analog input voltage. As shown in *Figure 3*, the PWM0 will have a fixed frequency of 50 KHz but a variable duty cycle according to the sensed input signal.

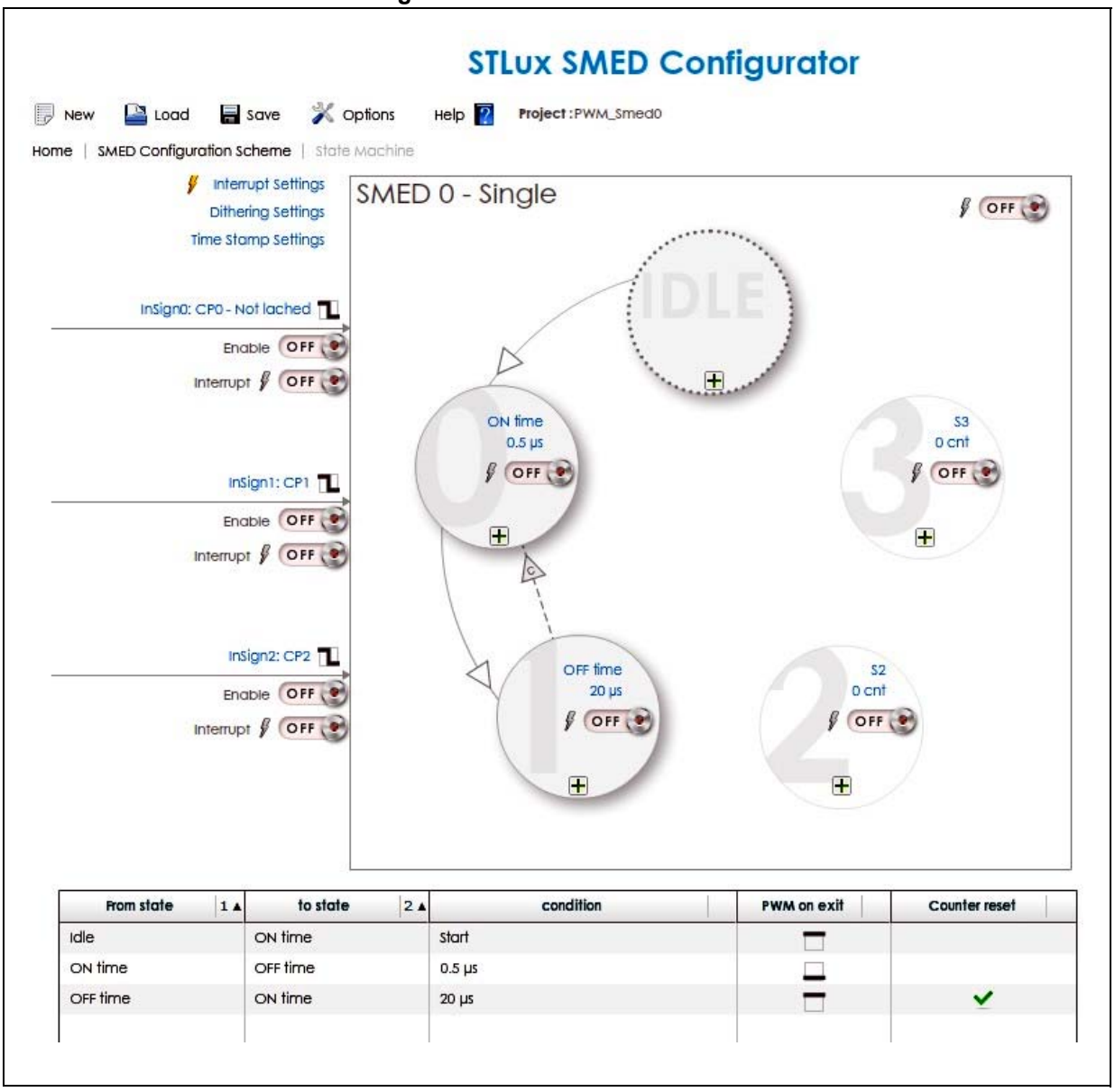
In order to dime the input voltage, the shown analog circuit must be implemented and plugged to the evaluation board.

Figure 4. PWM0 constant frequency with variable duty cycle



When the input voltage is below the lower threshold set to 200 mV, the PWM0 is OFF. If the input voltage Vin 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 Vin overcomes the upper threshold set to 1 V, the PWM0 is OFF. The implemented FSM is shown in *Figure 4*.

Figure 5. SMED 0 state machine



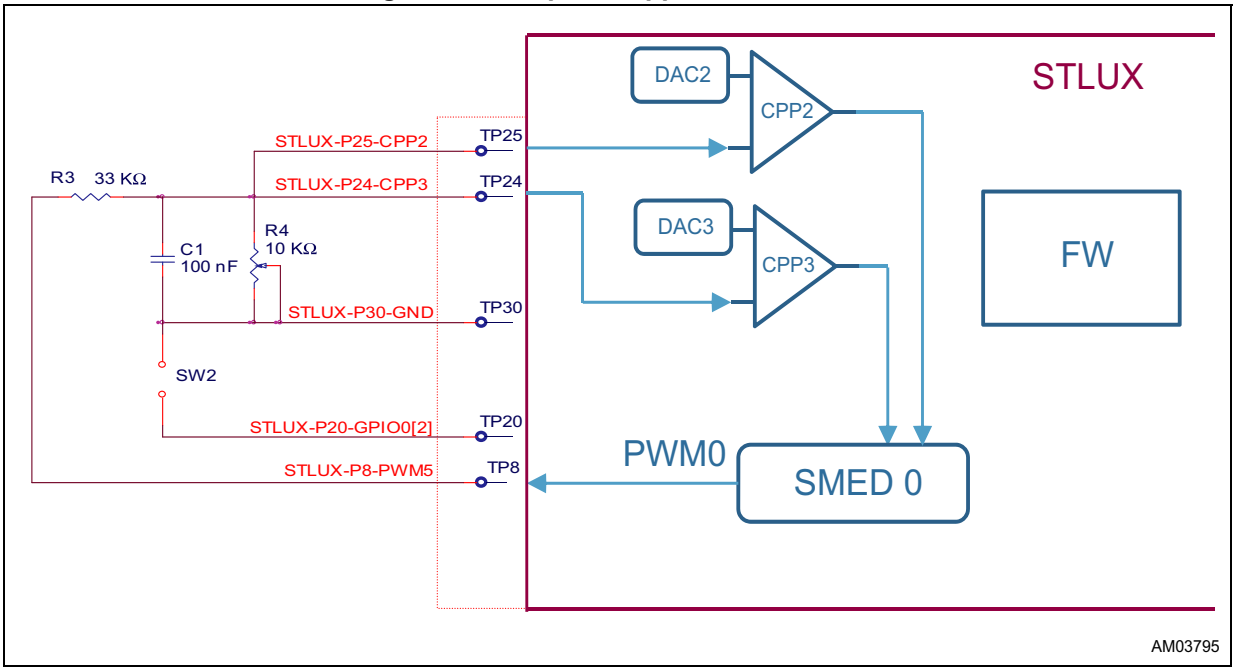
The input voltage measure is performed by the ADC at a given frequency set by the system timer. The STMR interrupt handler performs the regulation loop reading the Vin sampled value and computing the PWM0 according to the behavior described above.

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.

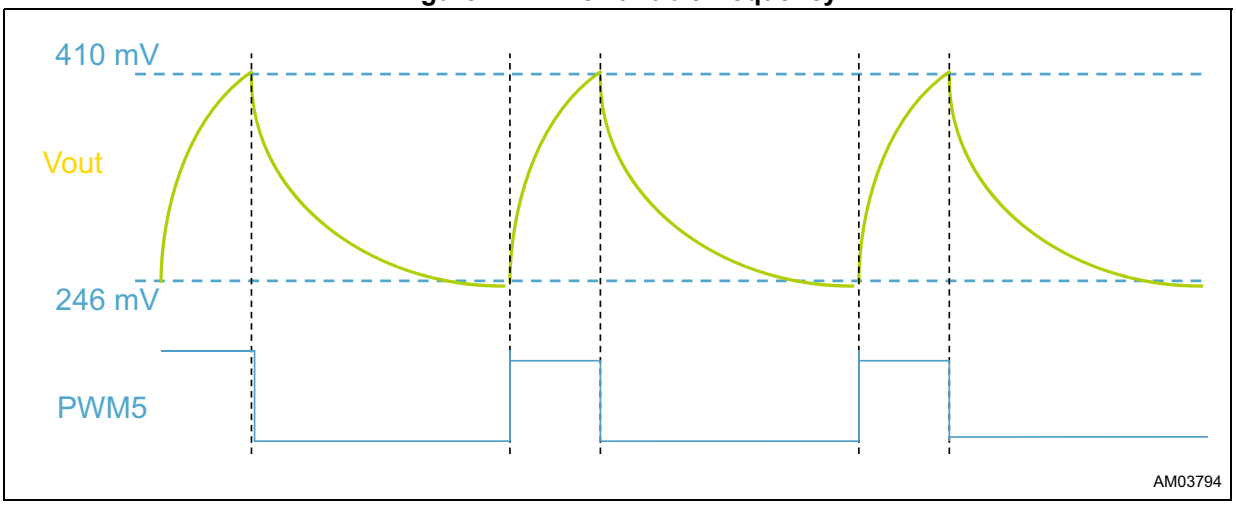
7 Example V

Figure 6. Example 04 application scheme



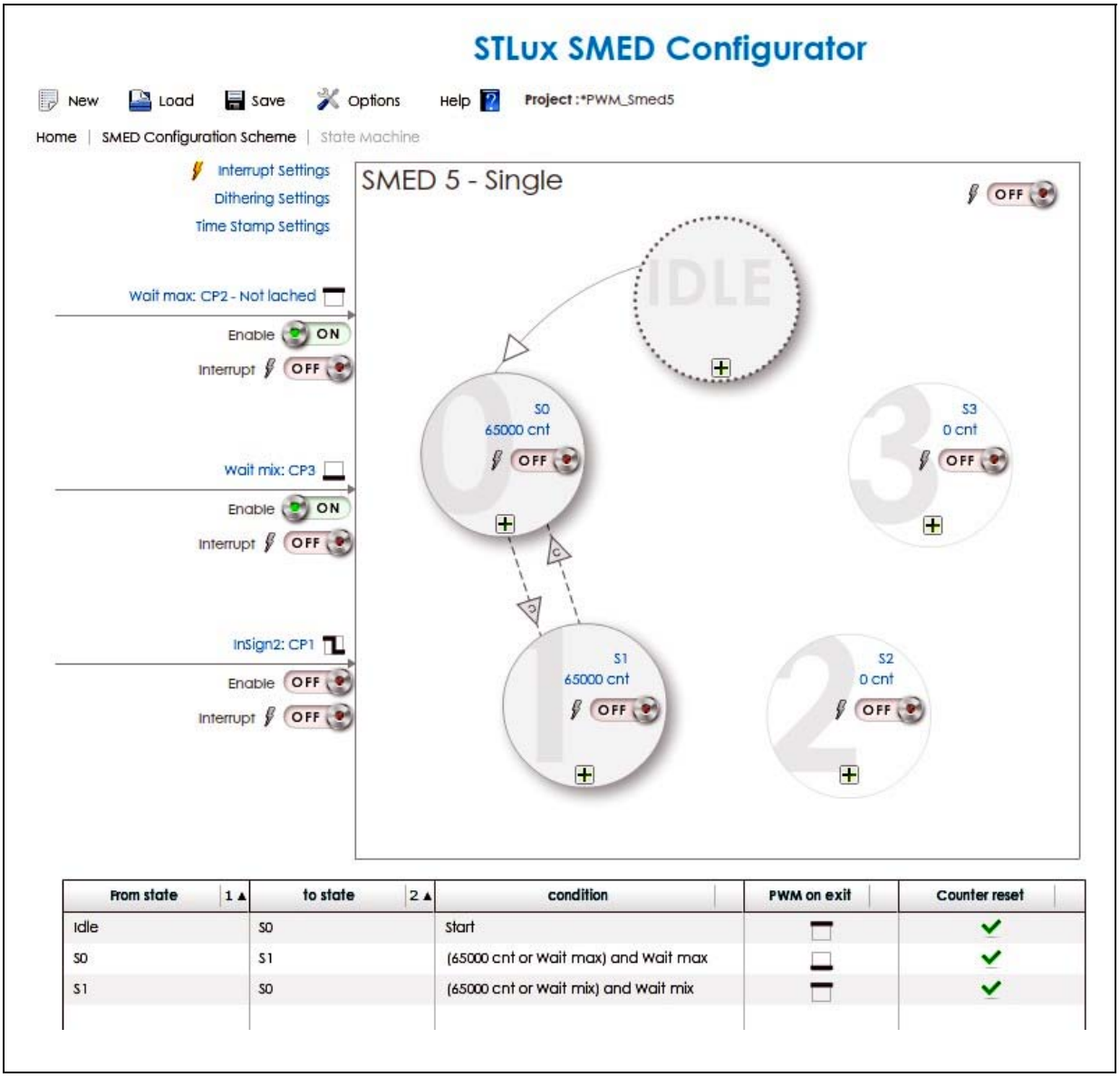
The goal of the fifth example is to learn how to generate a PWM signal as a function of two comparators. In this case we want to control the PWM5 variable frequency so to keep an analog output voltage between a high threshold and a low threshold respectively set to 410 mV and 246 mV by CPP2 and CPP3 references.

Figure 7. PWM5 variable frequency



As shown in *Figure 6*, in order to keep Vout over 246 mV, 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. No more firmware interaction is needed as the output control is automatically handled by the SMED 5 FSM described in *Figure 7*.

Figure 8. SMED 5 state machine



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 is halted to reduce power consumption.

Let's note that actually the SMED 5 is configured in a way that feedback from comparators makes the PWM5 toggle. If the comparator feedback is missing, it gets stuck. Possible evolution of this exercise can be to try modifying the finite state machine, so that the PWM5 restarts toggling after a max_time the SMED 5 gets stuck in a state.

8 Revision history

Table 2. Document revision history

Date	Revision	Changes
13-Jun-2014	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2014 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com