



... the world's most energy friendly microcontrollers

TIMER

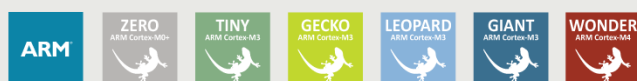
AN0014 - Application Note

Introduction

This application note gives an overview of the EFM32 TIMER module, followed by explanations on how to configure and use its primary functions, including up/down count, input capture, output compare and PWM.

This application note includes:

- This PDF document
- Source files (zip)
 - Example C-code
 - Multiple IDE projects

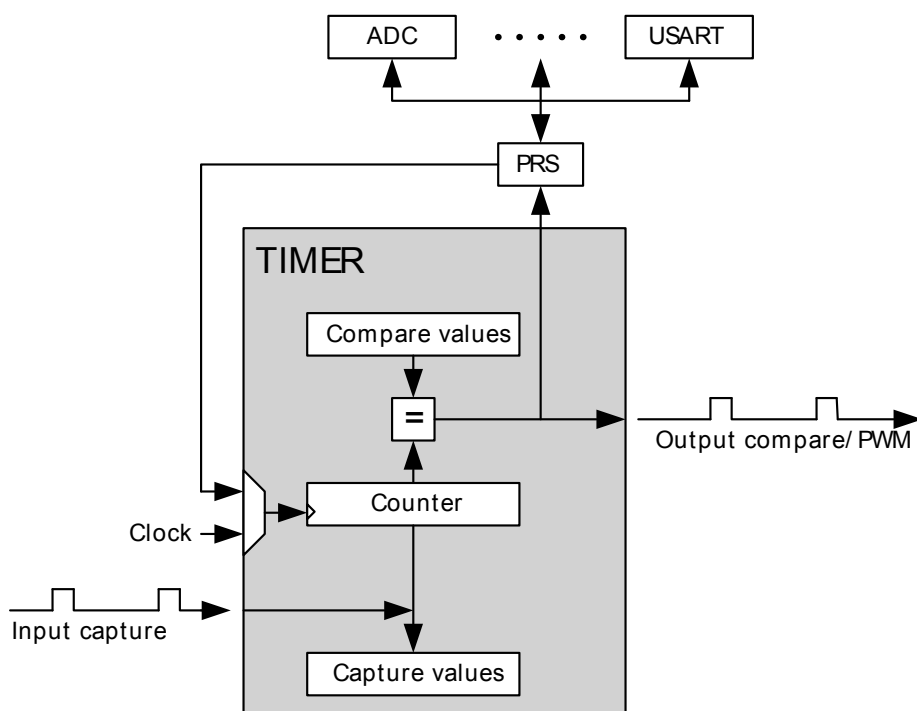


1 Introduction

The EFM32 TIMER module can be used for a number of different functions which include up/down count, input capture or output compare. Figure 1.1 (p. 2) shows a block diagram with the TIMER module overview.

The TIMER can be clocked from several sources, both internal and external. It includes a 16-bit counter with multiple modes, input capture, output compare and PWM. If more bits are needed, the 3 timers can be connected, resulting in a counter with 32 or 48 bits. Also, it can generate Reflex Signals to trigger other peripherals such as the ADC, USART or DAC and interrupts to wake up the processor.

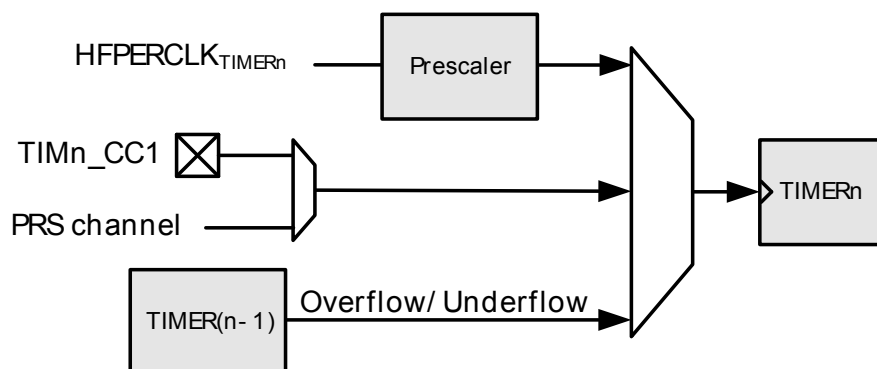
Figure 1.1. TIMER overview



2 Clock Sources

The clock input for the counter can come from 3 different sources: peripheral high frequency clock (HFPERCLK), compare/capture channel 1 input (TIMn_CC1 pin or PRS channel) or underflow/overflow from the lower numbered timer. If the peripheral high frequency clock is chosen, it can be prescaled up to a factor of 1024.

Figure 2.1. Clock Sources



To select the TIMER clock source, the bit fields CLKSEL and PRESC from TIMERN_CTRL register are used. The first bit field selects the clock source for the timer and the second sets the prescaler for the HFPERCLK (if selected). If using compare/capture channel 1 for clock input using TIMn_CC1 pin then it should be configured as an input on the GPIO module. AN0012 GPIO contains more details on pin configuration.

3 Interrupts, PRS and DMA

The timer has 5 output events that can trigger an interrupt. All except the buffer overflow also trigger a PRS Signal and a DMA request:

- Counter Underflow
- Counter Overflow
- Compare match, input capture or buffer overflow (one per Compare/Capture channel)

To enable one or more interrupts, TIMERN_IEN register must be written with the corresponding interrupt enable bits. Also, the NVIC vector for the corresponding interrupt line must be enabled.

There is also a function available on the emlib to activate timer interrupts: `void TIMER_IntEnable(TIMER_TypeDef *timer, uint32_t flags);`

The TIMER always generates Reflex outputs (one HFPERCLK cycle pulse), on the above output events regardless of the TIMERN_IEN register or if it is selected by a PRS channel.

If an interrupt is generated the interrupt flag can be read from the TIMERN_IF register and cleared by writing TIMERN_IFC.

4 Counting Modes and Timer Setup

4.1 Timer Setup

Prior to start using the timer some configurations must be performed to ensure that it works as intended. In order to facilitate this configuration there is an emlib function available: `void TIMER_Init(TIMER_TypeDef *timer, const TIMER_Init_TypeDef *init);`

By using this function the user will be able to configure the following parameters:

- Start counting when configuration is complete
- Counter running during debug
- Prescaler if HFPER clock selected
- Clock selection
- Action on falling edge input
- Action on rising edge input
- Counting mode
- DMA request clear on active
- X2 or X4 quadrature decode mode (if used)
- One shot or continuous counting
- Timer start/stop/reload by other timers

4.2 Up, Down and Up/Down

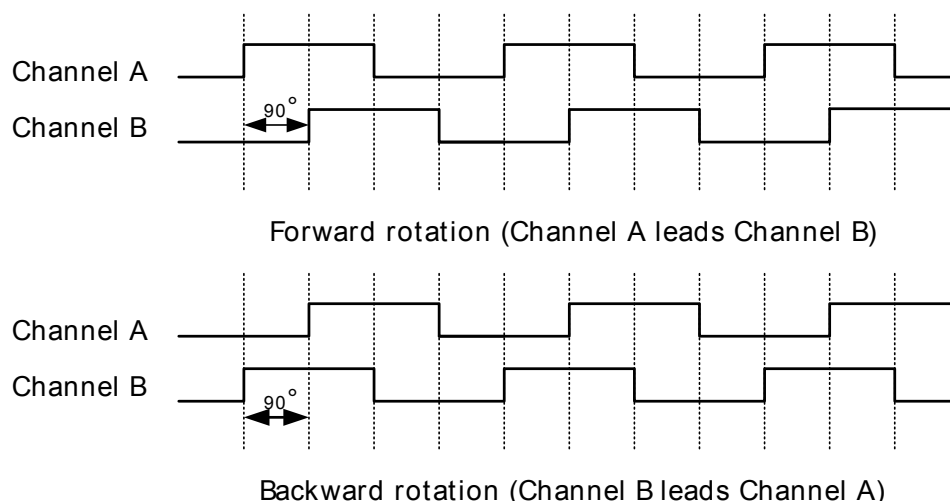
The counter can be used for up, down or up/down counting with different behaviors in each mode:

- Up-count: counts up until it reaches the value in `TIMERn_TOP`, then it is reset to 0 before counting up again (if in continuous counting). When the counter goes from `TIMERn_TOP` to 0 an overflow event occurs.
- Down-count: counts from `TIMERn_TOP` down to 0, then it is reloaded with the value of `TIMERn_TOP`. When the counter goes from 0 to `TIMERn_TOP` an underflow event occurs.
- Up/Down-count: counter starts at 0 and counts up until it reaches `TIMERn_TOP`. Then it counts down to 0 and starts counting up again. Overflow occurs when the counter goes from `TIMERn_TOP` to `TIMERn_TOP-1` and underflow when it goes from 0 to 1.

The counting mode is configured by using the Counting Mode parameter in the `TIMER_Init` function. It can also be configured by writing `MODE` bit field in the `TIMERn_CTRL` register:

4.3 Quadrature Decoding

Quadrature Decoding is the 4th counting mode available on the `TIMER` module. This mode will increment or decrement the counter based on two signals which are 90 degrees out of phase. These signals are tapped from `CC0` (Channel A) and `CC1` (Channel B).

Figure 4.1. Input signals for quadrature decoding

If Channel A is leading Channel B the counter will be incremented. If the opposite happens, it will be decremented. The Quadrature Decoder can be set to X2 or X4 modes. The difference between these two modes is the number of counter increments/decrements for each signal period. In X2 mode the counter will be incremented/decremented 2 times per signal period, and on X4 mode it will be incremented/decremented 4 times.

Like the other counting modes, the quadrature mode can be configured using the Counting Mode parameter plus the X2 or X4 quadrature decode mode parameter from the `Timer_Init()` function. Alternatively the user can write directly to MODE bit field in the `TIMERn_CTRL` register.

4.4 Hardware Count Control

The TIMER can be configured to start/stop/reload autonomously when there is a rising and/or falling edge on the input from the Compare/Capture channel 0. This is done using the parameters Action on falling edge input and Action on rising edge input from the `Timer_Init()` function. The options available for each parameter are:

- No action
- Start counter without reload
- Stop counter without reload
- Reload and start counter

4.5 Counter Software Examples

To demonstrate basic counter functionality of the TIMERS, several examples for various device families are included with this application note. The project names are postfixed with the device family they are made for (e.g. `timer_up_count_gg`).

4.5.1 Up Count

The `timer_up_count` projects demonstrate how to configure the timer for up counting operation together with interrupt handling. The EFM32 wakes up from a TIMER overflow every 2 seconds and toggles LED 0 on the STK, or PC0 (P4.3 on the protoboard) on the DVK.

4.5.2 Quadrature Decoding

For quadrature decoding the `timer_quad_decode` examples can be used for the EFM32_Gxxx_STK. This demonstration uses PB0 and PB1 from the STK to simulate the 2 signals that will be decoded. To

visualize the decoding the buttons should be pressed in two different order resulting in different LED movements.

Executing the following cycle will light up the LEDs from right to left.

- Press and hold PB0
- Press and hold PB1 while holding PB0
- Release PB0
- Release PB1

Executing the following cycle will light up the LEDs from left to right.

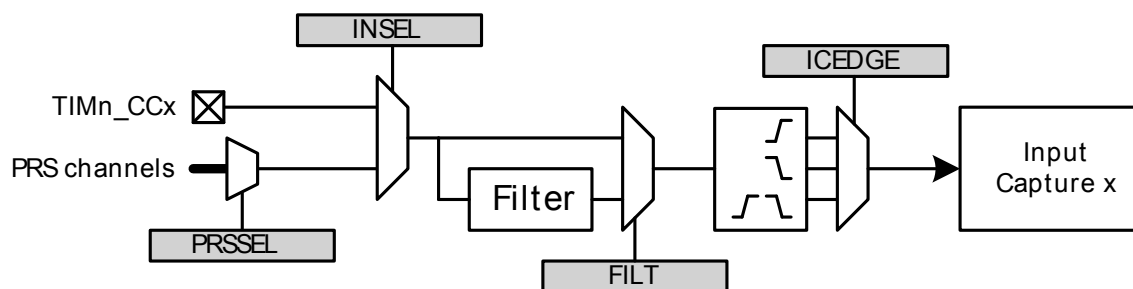
- Press and hold PB1
- Press and hold PB0 while holding PB1
- Release PB1
- Release PB0

5 Capture/Compare

5.1 Input Capture

This mode will capture the value of the counter based on a triggering event such as a signal rising edge on the CCx input pin or a Reflex Signal. This feature can be used for instance to measure pulse width or period. Figure 5.1 (p. 8) shows the input path for the TIMER Input Capture Channel.

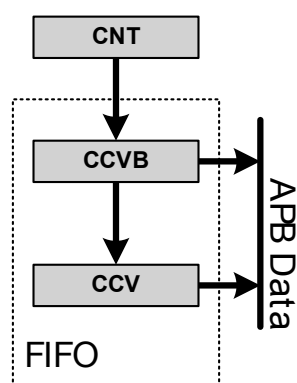
Figure 5.1. Input Capture Path



The capture signal can come from either a TIMn_CCx pin or a PRS channel. It can also be filtered before it is used, which requires the input to remain stable for 5 cycles in a row before it is propagated. The signal edge that triggers a capture can be rising, falling or both

There are two capture registers associated with the input capture mode, TIMERNn_CCx_CCV and TIMERNn_CCx_CCVB. Together they form a FIFO buffer.

Figure 5.2. Input Capture Buffer

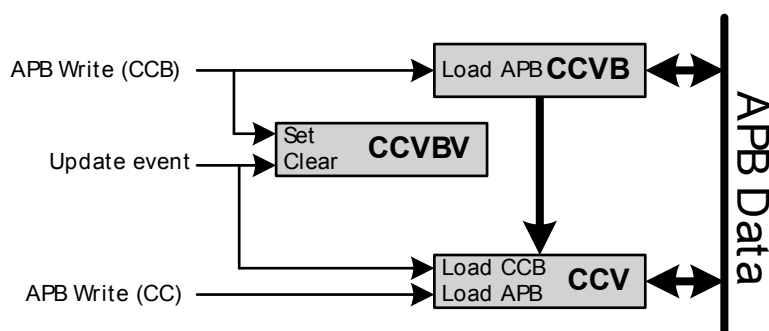


The first capture will go directly to TIMERNn_CCx_CCV and reading it will load the next capture value from TIMERNn_CCx_CCVB if existing. If the capture value is to be read without changing the FIFO content, it can be done by reading TIMERNn_CCx_CCV. Also, reading from TIMERNn_CCx_CCVB does not change the FIFO content. If a capture occurs while both TIMERNn_CCx_CCV and TIMERNn_CCx_CCVB contain unread data, a buffer overflow interrupt will occur and the new capture value will overwrite the value in TIMERNn_CCx_CCVB. In capture mode TIMERNn_CCx_CCV is read only. The ICVx flag in TIMERNn_STATUS indicates if there is a valid unread capture. CCPOLx bits indicate the polarity edge that triggered the capture contained in TIMERNn_CCx_CCV.

5.2 Output Compare

Each capture/compare channel contains a comparator which outputs a compare match if the contents of TIMERNn_CCx_CCv matches the counter value. On an event (compare match, overflow or underflow) each channel can be configured to either set, clear or toggle the output. The TIMERNn_CCx_CCv register is writable and its value will be compared against the count value. It can be written either directly or by writing the TIMERNn_CCx_CCvB register, which will load TIMERNn_CCx_CCv on the next update event. An update event is set on overflow in up-count mode and on underflow in down-count or up/down count mode. It is recommended to always write on TIMERNn_CCx_CCvB both on Output Compare and PWM modes. This way TIMERNn_CCx_CCv is buffered and that avoids glitches on the output.

Figure 5.3. Timer Output Compare Buffer



If more than one event occurs in the same cycle a matching event will have priority over overflow or underflow. Output compare mode can for example be used to output a time reference or to generate a frequency for instance.

For frequency generation the TIMER must be configured in up-count mode and the CC channel must be setup to toggle the output on overflow. When the counter reaches TOP value the output will toggle, generating a frequency given by Equation 5.1 (p. 9)

Up-count Frequency Generation Equation

$$f_{\text{FRG}} = f_{\text{HFPERCLK}} / (2^{\text{PRESC}} + 1) \times (\text{TOP} + 1) \quad (5.1)$$

5.3 Pulse-Width Modulation (PWM)

The TIMER module has a separate mode for PWM generation, and is only supported for up-count and up/down-count modes. The same way as on compare mode, TIMERNn_CCx_CCv is writable but TIMERNn_CCx_CCB should be used to prevent glitches.

On up-count, the TIMER will generate a single slope PWM output. The period is equal to TOP+1 cycles and the output will be high if the counter is between 0 and TIMERNn_CCx_CCv, and low between TIMERNn_CCx_CCv+1 and TOP. Up/Down-count mode will generate dual-slope PWM mode with the same behavior described above. The PWM frequency is given by Equation 5.2 (p. 9) for single slope and Equation 5.3 (p. 9) for dual slope.

Up-count PWM Frequency Equation

$$f_{\text{PWM}_{\text{up/down}}} = f_{\text{HFPERCLK}} / (2^{\text{PRESC}} \times (\text{TOP} + 1)) \quad (5.2)$$

Up/Down-count PWM Frequency Equation

$$f_{\text{PWM}_{\text{up/down}}} = f_{\text{HFPERCLK}} / (2^{\text{PRESC}+1} \times \text{TOP}) \quad (5.3)$$

5.4 Configuration

The emlib includes a function for configuring the CC channels into any of the previously mentioned modes which has the following prototype:

`void TIMER_InitCC(TIMER_TypeDef *timer, unsigned int ch, const TIMER_InitCC_TypeDef *init).` The input path for the capture/compare channels is shown on Figure 5.1 (p. 8).

By using this function, the user will configure one of the CC channels with the following parameters:

- Input capture event
- Input capture edge
- PRS channel trigger selection (if PRS input selected)
- Counter underflow output action
- Counter overflow output action
- Counter match output action
- CC Channel mode
- Digital filter
- TIMn_CCx or PRS input

5.5 Capture/Compare/PWM Software Examples

To demonstrate the capture, compare and PWM functionality of the TIMERS, several examples for various device families are included with this application note. The project names are postfixed with the device family they are made for (e.g. `timer_input_capture_gg` for Giant Gecko devices).

5.5.1 Input Capture

The `timer_input_capture` projects measures the time that the user presses PB0 (active low) on the STK or connecting P3.12 to ground on the DVK protoboard. When the pin is released the time is displayed on the LCD, and if overflow occurs it is also displayed. The counter control is entirely done by hardware. When the pin is grounded there will be a falling edge transition that causes the TIMER to reload and start counting. When the pin is released there will be a rising edge transition and TIMER stops counting and the counter value is captured. This example only works on Compare/Capture channel 0 as this is the only channel that can be used to automatically start and stop the TIMER. To run this project on the DVK it is necessary an MCU board with LCD.

5.5.2 Input Capture with DMA

The `timer_input_capture_dma` projects measures the time that the user presses PB0 (active low) on the STK or connecting P3.12 to ground on the DVK protoboard. The counter control is entirely done by hardware. When the pin is grounded there will be a falling edge transition that causes the TIMER to reload and start counting. When the pin is released there will be a rising edge transition and TIMER stops counting and the counter value is captured. The DMA is set up to transfer the captured counter values to two buffers in RAM. Ping-pong mode is used for the DMA to allow the CPU to handle one result buffer while the other is being written, while still being able to handle all incoming capture values. For more information on the DMA and ping-pong mode, please refer to application note an0013. This example only works on Compare/Capture channel 0 as this is the only channel that can be used to automatically start and stop the TIMER.

5.5.3 Output Compare

The `timer_output_compare` projects generates a frequency by toggling pin PD1 (P5.3 on the DVK protoboard) on counter overflow using the compare mode feature. The example can be used on both the STKs and the DVK

5.5.4 PWM

The `timer_pwm` projects generates PWM signals with varying duty cycle over time. Similarly to the Output Compare example, the output PWM signal can be probed on pin PD1 (P5.4 on the DVK protoboard). The example can be used on both the STKs and the DVK.

5.5.5 PWM with DMA

The `timer_pwm_dma` projects generates PWM signals with varying duty cycle over time. The output PWM signal can be probed on pin PD1 (P5.4 on the DVK protoboard). The example can be used on both the STKs and the DVK. The DMA is used to fetch new compare values for the PWM duty cycle for every PWM period. The compare values are stored in two RAM buffers, which are accessed using DMA ping-pong mode. This allows the CPU to modify one set of compare values while the other set is being used, without interrupting the PWM sequence. For more information on the DMA, please refer to application note an0013.

6 Revision History

6.1 Revision 1.08

2013-09-03

New cover layout

6.2 Revision 1.07

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

Added DMA examples for PWM and Input Capture.

Changed PWM examples to set the PWM frequency as a define instead of the TOP value.

Fixed an error in the format string for the input capture example.

6.3 Revision 1.06

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

6.4 Revision 1.05

2012-08-13

Added software support for the Tiny and Giant Gecko STKs.

6.5 Revision 1.04

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

6.6 Revision 1.03

2011-10-21

Updated IDE project paths with new kits directory.

6.7 Revision 1.02

2011-05-18

Updated projects to align with new bsp version.

6.8 Revision 1.01

2010-11-16

Changed example folder structure, removed build and src folders.

Updated chip init function to newest efm32lib version.

Updated register defines in code to match newest efm32lib release.

6.9 Revision 1.00

2010-09-24

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

A.2 Trademark Information

Silicon Laboratories Inc., Silicon Laboratories, the Silicon Labs logo, Energy Micro, EFM, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

B Contact Information

Silicon Laboratories Inc.

400 West Cesar Chavez

Austin, TX 78701

Please visit the Silicon Labs Technical Support web page:

<http://www.silabs.com/support/pages/contacttechnicalsupport.aspx>

and register to submit a technical support request.

Table of Contents

1. Introduction	2
2. Clock Sources	3
3. Interrupts, PRS and DMA	4
4. Counting Modes and Timer Setup	5
4.1. Timer Setup	5
4.2. Up, Down and Up/Down	5
4.3. Quadrature Decoding	5
4.4. Hardware Count Control	6
4.5. Counter Software Examples	6
5. Capture/Compare	8
5.1. Input Capture	8
5.2. Output Compare	9
5.3. Pulse-Width Modulation (PWM)	9
5.4. Configuration	10
5.5. Capture/Compare/PWM Software Examples	10
6. Revision History	12
6.1. Revision 1.08	12
6.2. Revision 1.07	12
6.3. Revision 1.06	12
6.4. Revision 1.05	12
6.5. Revision 1.04	12
6.6. Revision 1.03	12
6.7. Revision 1.02	12
6.8. Revision 1.01	12
6.9. Revision 1.00	13
A. Disclaimer and Trademarks	14
A.1. Disclaimer	14
A.2. Trademark Information	14
B. Contact Information	15
B.1.	15

List of Figures

1.1. TIMER overview 2

2.1. Clock Sources 3

4.1. Input signals for quadrature decoding 6

5.1. Input Capture Path 8

5.2. Input Capture Buffer 8

5.3. Timer Output Compare Buffer 9

List of Equations

5.1. Up-count Frequency Generation Equation 9

5.2. Up-count PWM Frequency Equation 9

5.3. Up/Down-count PWM Frequency Equation 9

silabs.com

