APPLICATION NOTE

AT12031: Advanced Features of SAM D21 Timer/Counter for Control Application (TCC)

ATSAMD21E16L

Introduction

This application note describes the below mentioned Advanced features of the Timer/Counter for Control Applications available on the Atmel® | SMART SAM D21.

1. RAMP2C mode.
2. DMA Operation with circular buffer in RAMP2, RAMP2A modes.
3. Fault Blanking prescaler.

This application note details the configurations for above features of the Timer/Counter for Control Applications.

The software example mentioned in this document is provided in latest ASF (Atmel® Software Framework).

For more details on TCC module refer SAMD21E16L complete datasheet.

Features

- Up to four compare/capture channels (CC) with:
  - Double buffered period setting
  - Double buffered compare or capture channel
  - Circular buffer on period and compare channel registers
- Waveform generation:
  - Frequency generation
  - Single-slope pulse-width modulation (PWM)
  - Dual-slope pulse-width modulation with half-cycle reload capability
- Input capture:
  - Event capture
  - Frequency capture
  - Pulse-width capture
- Waveform extensions:
  - Configurable distribution of compare channel outputs across port pins
  - Low- and high-side output with programmable dead-time insertion
  - Waveform swap option with double buffer support
  - Pattern generation with double buffer support
  - Dithering support

- Fault protection for safe driver disabling:
  - Two recoverable fault sources
  - Two non-recoverable fault sources
  - Debugger can be source of non-recoverable fault
- Input event:
  - Two input events for counter

Table of Contents

1 Glossary

TCC	Timer/Counter for Control Applications
PER	Period Register
GCLK	Generic clock
CC	Compare/capture
SWD	Serial Wire Debugger
DMAC	Direct Memory Access Controller
ASF	Atmel Software Framework
Atmel Studio	Integrated Development Environment (IDE) for Atmel microcontrollers
PWM	Pulse Width Modulation
IDE	Integrated Development Environment
EVSYS	Event System

2 Pre-requisites

The solutions discussed in this document require the following:

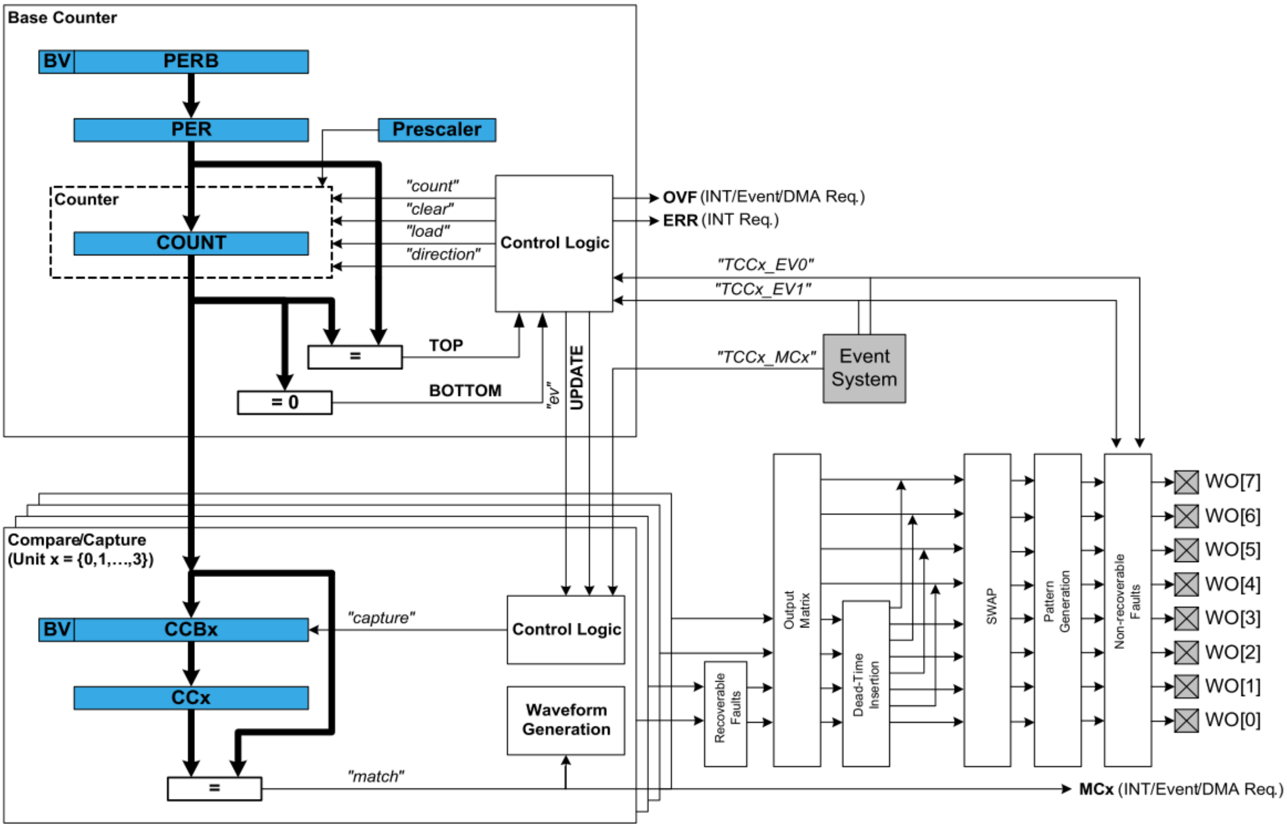
- Atmel Studio 6.2 or above
- ASF version 3.24 or above
- Programming/debugging tool for Atmel SAM devices with SWD interface support (Ex: - Atmel-ICE)
- ATSAMD21E16L board with SWD interface header

3 TCC

3.1 Module Overview

The Timer/Counter for Control Applications (TCC) module provides a set of timing and counting related functionalities, such as the generation of periodic waveforms, the capture of a periodic waveform's frequency/duty cycle, software timekeeping for periodic operations, waveform extension control, fault detection etc. It enables low- and high-side output with optional dead-time insertion. It can also generate a synchronized bit pattern across the waveform output pins. The fault options enable fault protection for safe and deterministic handling, disabling, and/or shut down of external drivers. Waveform extensions are intended for use in different types of motor control, ballast, LED, H-bridge, power converter, and other types of power control applications. The counter size of this TCC module is maximum 24-bit.

Figure 3-1. Timer/Counter Block Diagram



3.2 Functional Description

The TCC module consists of following sections:

- Base Counter
- Compare/Capture channels, with waveform generation
- Waveform extension control and fault detection
- Interface to the event system, DMAC, and the interrupt system

The base counter can be configured to either count a prescaled generic clock or events from the event system. The counter value can be used by compare/capture channels which can be set up either in compare mode or capture mode. In capture mode, the counter value is stored when a configurable event occurs. This mode can be used to generate timestamps used in event capture, or it can be used for the measurement of a periodic input signal's frequency/duty cycle. In compare mode, the counter value is compared against one or more of the configured channels' compare values. When the counter value coincides with a compare value an action can be taken automatically by the module, such as generating an output event or toggling a pin when used for frequency or PWM signal generation. The connection of events between modules requires the use of the SAM Event System Driver (EVSYS) to route output event of one module to the input event of another. For more information on event routing, refer to the event driver (EVSYS) documentation. In compare mode, when output signal is generated, extended waveform controls are available, to arrange the compare outputs into specific formats. The Output matrix can change the channel output routing; Pattern generation unit can overwrite the output signal line to specific state. The Fault protection feature of the TCC supports recoverable and non-recoverable faults.

3.3 Special Considerations

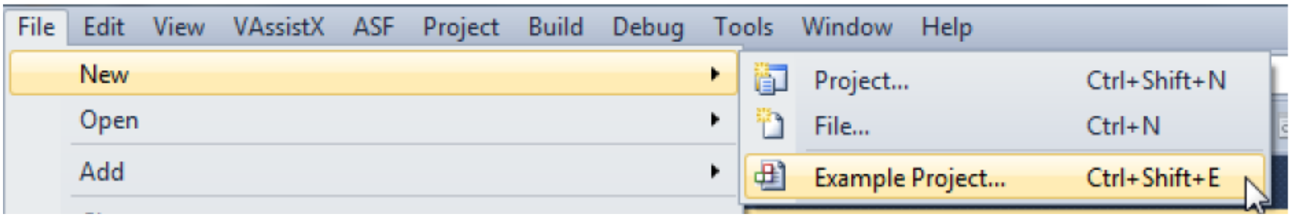
As the TCC module may have more waveform output pins than the number of compare/capture channels, the free pins (with number higher than number of channels) will reuse the waveform generated by channels subsequently. E.g., if the number of channels is four and number of wave output pins is eight, channel 0 outputs will be available on out pin 0 and 4, channel 1 output on wave out pin 1 and 5, and so on.

4 TCC Example Project

The SAMD21E16L TCC Advanced Features example code is available in the latest ASF with Atmel Studio. Follow the steps below to load example code in Atmel Studio:

A. To load the example project in Atmel Studio, go to ‘File’ -> ‘New’ and click on ‘Example Project…’.The shortcut key for to do this is (CTRL +Shift + E).

Figure 4-1. Creating Example Project in Atmel Studio

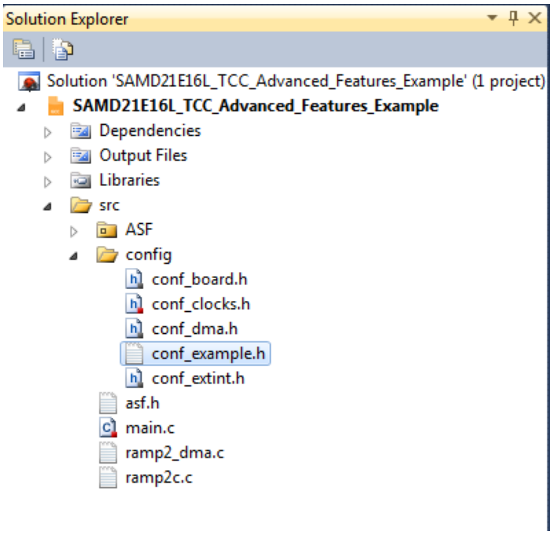


B. 'New Example Project from ASF or Extensions' dialog box will appear, in 'Search for Example Projects' text box, type 'E16L', will show 'SAM D21E16L' projects available in the ASF.

C. Select "SAMD21E16L TCC Advanced Features Examples' project and click 'OK' button.

D. After clicking "OK", the 'SAMD21E16L TCC Advanced Features Examples' project will be loaded in the Atmel Studio as shown in Figure 4-2.

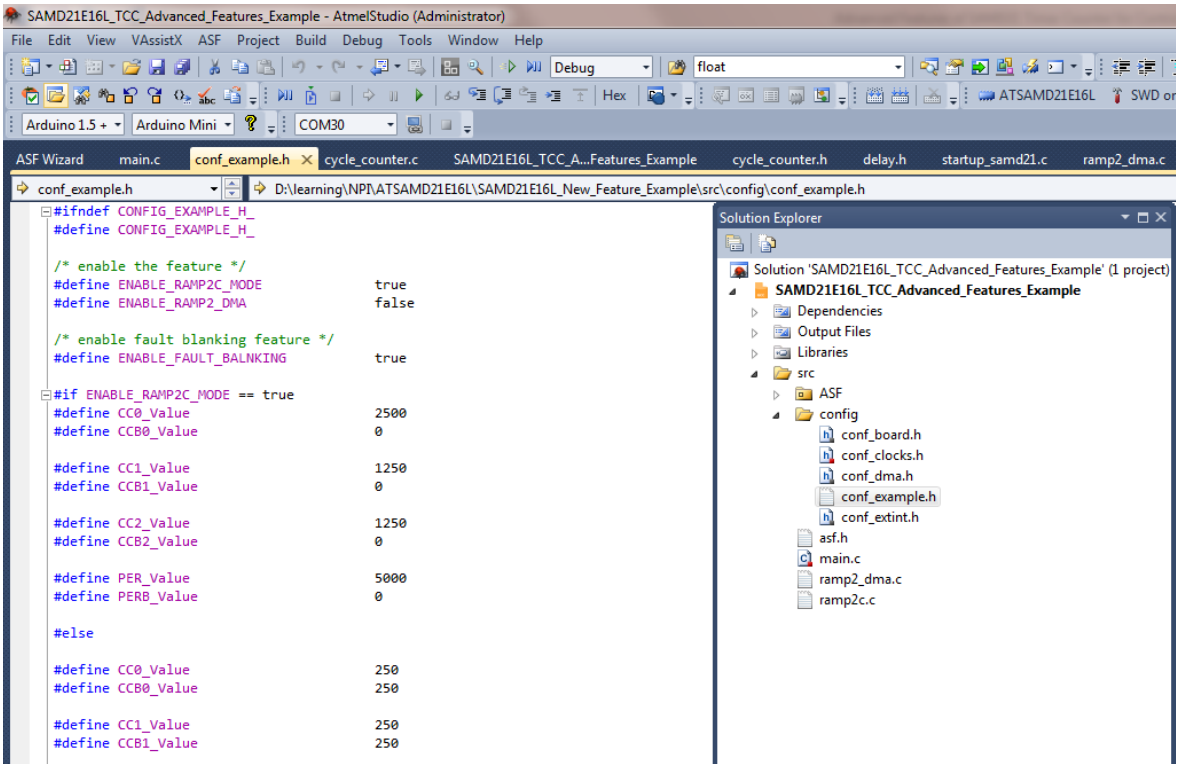
Figure 4-2. Solution Explorer View of SAMD21E16L TCC Advanced Features Example Project



```
Solution Explorer
Solution 'SAMD21E16L_TCC_Advanced_Features_Example' (1 project)
  SAMD21E16L_TCC_Advanced_Features_Example
    Dependencies
    Output Files
    Libraries
    src
      ASF
      config
        conf_board.h
        conf_clocks.h
        conf_dma.h
        conf_example.h
        conf_extint.h
      asf.h
      main.c
      ramp2_dma.c
      ramp2c.c
```

E. ‘SAMD21E16L TCC Advanced Features Example’ project has conf_example.h file, which has the macro definitions for each feature. Only one feature to be enabled at a time for the proper operation of this application code, except for fault blanking prescaler, this option can be enabled when RAMP2C feature is enabled.

Figure 4-3. Enable/Disable Feature using conf_example.h


SAMD21E16L_TCC_Advanced_Features_Example - AtmelStudio (Administrator)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
float
Hex
ATSAMD21E16L
SWD
Arduino 1.5 +
Arduino Mini
COM30
ASF Wizard
main.c
conf_example.h
cycle_counter.c
SAMD21E16L_TCC_A...Features_Example
cycle_counter.h
delay.h
startup_samd21.c
ramp2_dma.c
conf_example.h
D:\learning\NPI\ATSAMD21E16L\SAMD21E16L_New_Feature_Example\src\config\conf_example.h
Solution Explorer
Solution 'SAMD21E16L_TCC_Advanced_Features_Example' (1 project)
SAMD21E16L_TCC_Advanced_Features_Example
Dependencies
Output Files
Libraries
src
ASF
config
conf_board.h
conf_clocks.h
conf_dma.h
conf_example.h
conf_extint.h
asf.h
main.c
ramp2_dma.c
ramp2c.c

```
#ifndef CONFIG_EXAMPLE_H_
#define CONFIG_EXAMPLE_H_

/* enable the feature */
#define ENABLE_RAMP2C_MODE              true
#define ENABLE_RAMP2_DMA                false

/* enable fault blanking feature */
#define ENABLE_FAULT_BALNKING           true

#if ENABLE_RAMP2C_MODE == true
#define CC0_Value                       2500
#define CCB0_Value                      0

#define CC1_Value                       1250
#define CCB1_Value                      0

#define CC2_Value                       1250
#define CCB2_Value                      0

#define PER_Value                       5000
#define PERB_Value                      0

#else

#define CC0_Value                       250
#define CCB0_Value                      250

#define CC1_Value                       250
#define CCB1_Value                      250
```

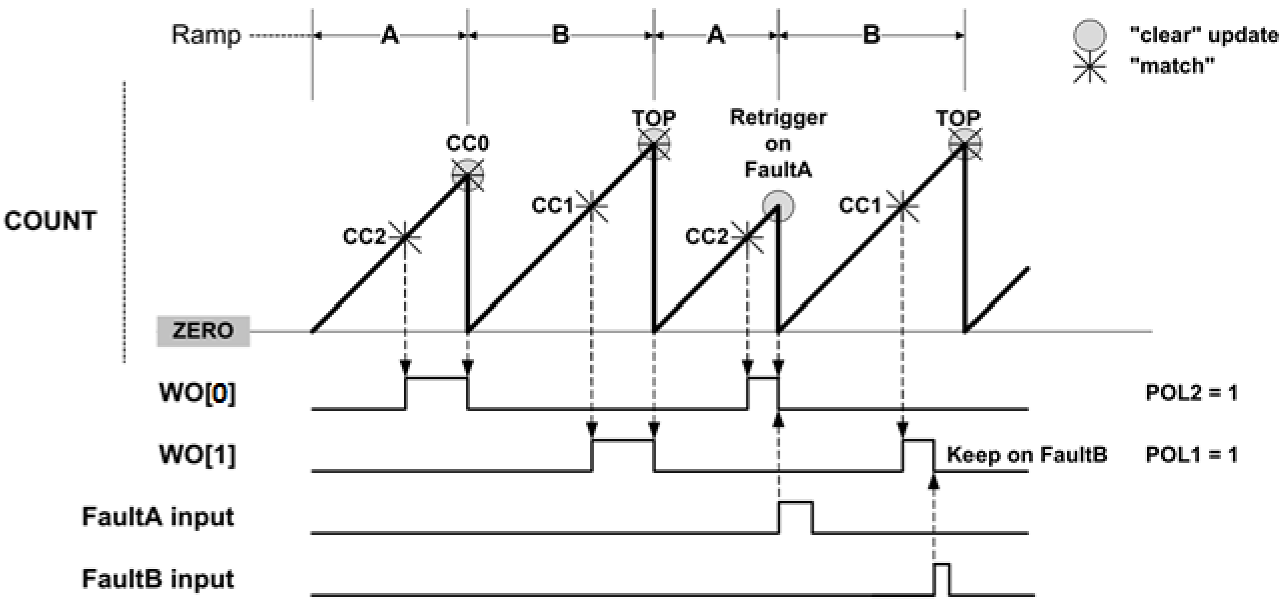
F. After enabling the feature in conf_example.h file, compile the project by selecting ‘Build’ -> ‘Build solution’.

5 TCC New Features Demonstration

5.1 RAMP2C Mode

This mode provides a way to cover RAMP2 operation requirements without the update constraint associated to the use of circular buffers. When this mode is enabled, a 2-channel TCC instance can generate one output signal. A 4-channels TCC instance can generate up to two output signals, and one channel can be enabled in capture mode. In Figure 5-1 Ramp A and B periods are controlled through CC0 and PER register value, while CC2 and CC1 control the duty cycle of WO[0] and WO[1] respectively.

Figure 5-1. RAMP2 Critical Operation



5.1.2 Code Snippet

5.1.2.1 Mode Configuration

```
/* Generator 1 as clock source */
config_tcc.counter.clock_source = GLCK_SOURCE;
config_tcc.counter.clock_prescaler = TCC_CLOCK_DIVIDER;

/* Set up initial period */
config_tcc.counter.period = PER_Value;

/* set CC0 */
/* CC0 will act as ramp A top in RAMP2C mode */
config_tcc.compare.match[0] = CC0_Value;

/* set compare reg CC1 & CC2 in RAMP2C mode */
config_tcc.compare.match[1] = CC1_Value;
config_tcc.compare.match[2] = CC2_Value;

/* Set up polarity output ramp */
config_tcc.compare.wave_polarity[1] = 1;
config_tcc.compare.wave_polarity[2] = 1;

/* Normal PWM in RAMP2 mode */
config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
config_tcc.compare.wave_ramp = TCC_RAMP_RAMP2;
```

5.1.2.2 Output Configuration

```
/* Set up output pins */
/* CC2 will be included to generate ramp output WO[0] in RAMP2C mode*/
config_tcc.pins.enable_wave_out_pin[0] = true;
config_tcc.pins.wave_out_pin[0]        = PIN_PA04E_TCC0_WO0;
config_tcc.pins.wave_out_pin_mux[0]    = MUX_PA04E_TCC0_WO0;

/* CC1 will be included to generate ramp output WO[1] RAMP2C mode */
config_tcc.pins.enable_wave_out_pin[1] = true;
config_tcc.pins.wave_out_pin[1]        = PIN_PA09E_TCC0_WO1;
config_tcc.pins.wave_out_pin_mux[1]    = MUX_PA09E_TCC0_WO1;
```

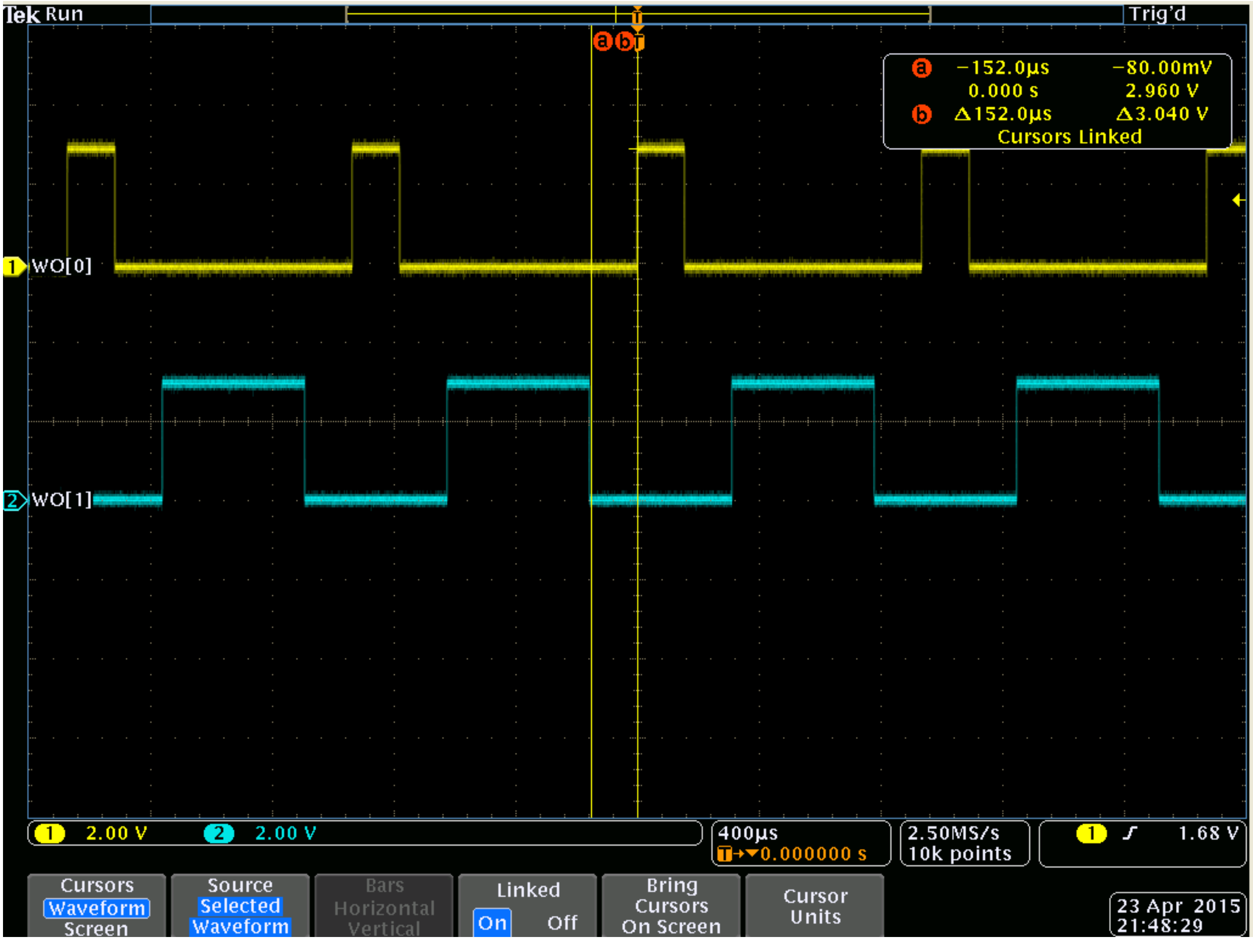
5.1.3 Waveform Output

Output is captured from pins:

PA04 – Waveform Output 0

PA09 – Waveform Output 1

Figure 5-2. Ramp2C Operation



TCC clock frequency	= 8MHz
TCC clock divider	= 1

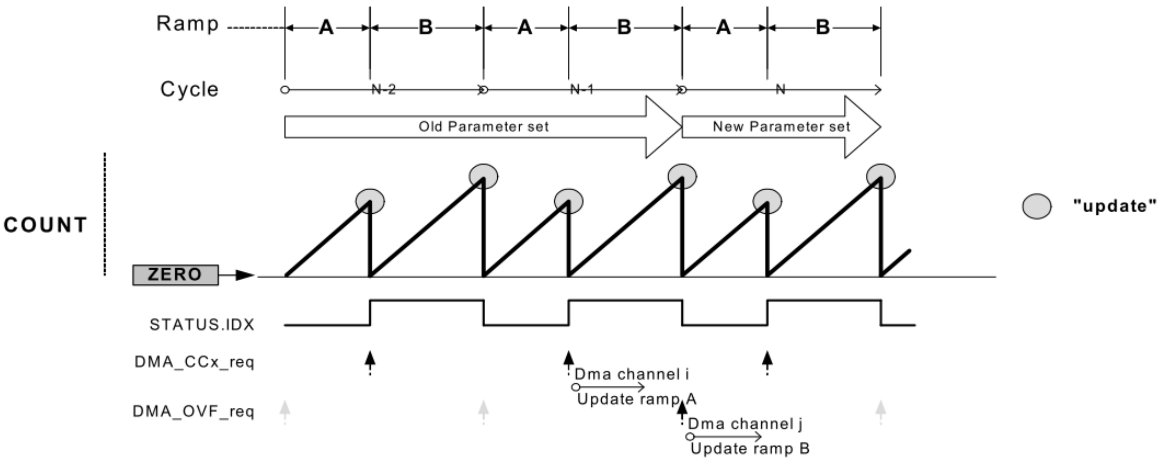
Time period for 1 count	= 1/8000000 = 125ns
CC0 value 2500	2500 x 125ns = 312.5µs
PER value 5000	5000 x 125ns = 625µs
CC1 value 1250	1250 x 125ns = 156µs
CC2 value 1250	1250 x 125ns = 156µs

5.2 DMA Operation with Circular Buffer in RAMP2, RAMP2A Mode

When circular buffering feature is enabled for compare channel, the related DMA request is not set on a compare match detection, but on start of ramp B. If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of ramp A with an effective DMA transfer on previous ramp B. The update of all circular buffer values for ramp A, can be done through a DMA channel triggered on a MC trigger. The update of all circular buffer values for ramp B, can be done through a second DMA channel triggered by the overflow DMA request.

Note: When an update of TCC value is required, the channel triggered by DMA overflow trigger must be enabled first.

Figure 5-3. DMA Triggers in RAMP and RAMP2A Operation Mode and Circular Buffer Enabled



5.2.2 Code Snippet

```
dma_descriptor_get_config_defaults(&descriptor_config);

descriptor_config.block_transfer_count = 1;
descriptor_config.beat_size = DMA_BEAT_SIZE_WORD;

/* Disable auto increment for source and destination */
descriptor_config.src_increment_enable = false;
descriptor_config.dst_increment_enable = false;

/* define source and destination address */
descriptor_config.source_address = (uint32_t) tcc_cycle_a_val;
descriptor_config.destination_address =  (uint32_t) &TCC0->CCB[0];

/* fill the descriptor with descriptor_config parameters */
dma_descriptor_create(&tcc_ramp_a_dma_descriptor, &descriptor_config);

/* add to descriptor chain */
dma_add_descriptor(&tcc_ramp_a_dma_resource, &tcc_ramp_a_dma_descriptor);

dma_start_transfer_job(&tcc_ramp_a_dma_resource);
```

5.2.3 Waveform Output

Output is captured from pins:

PA04 – Waveform Output 0

PA05 – Waveform Output 1

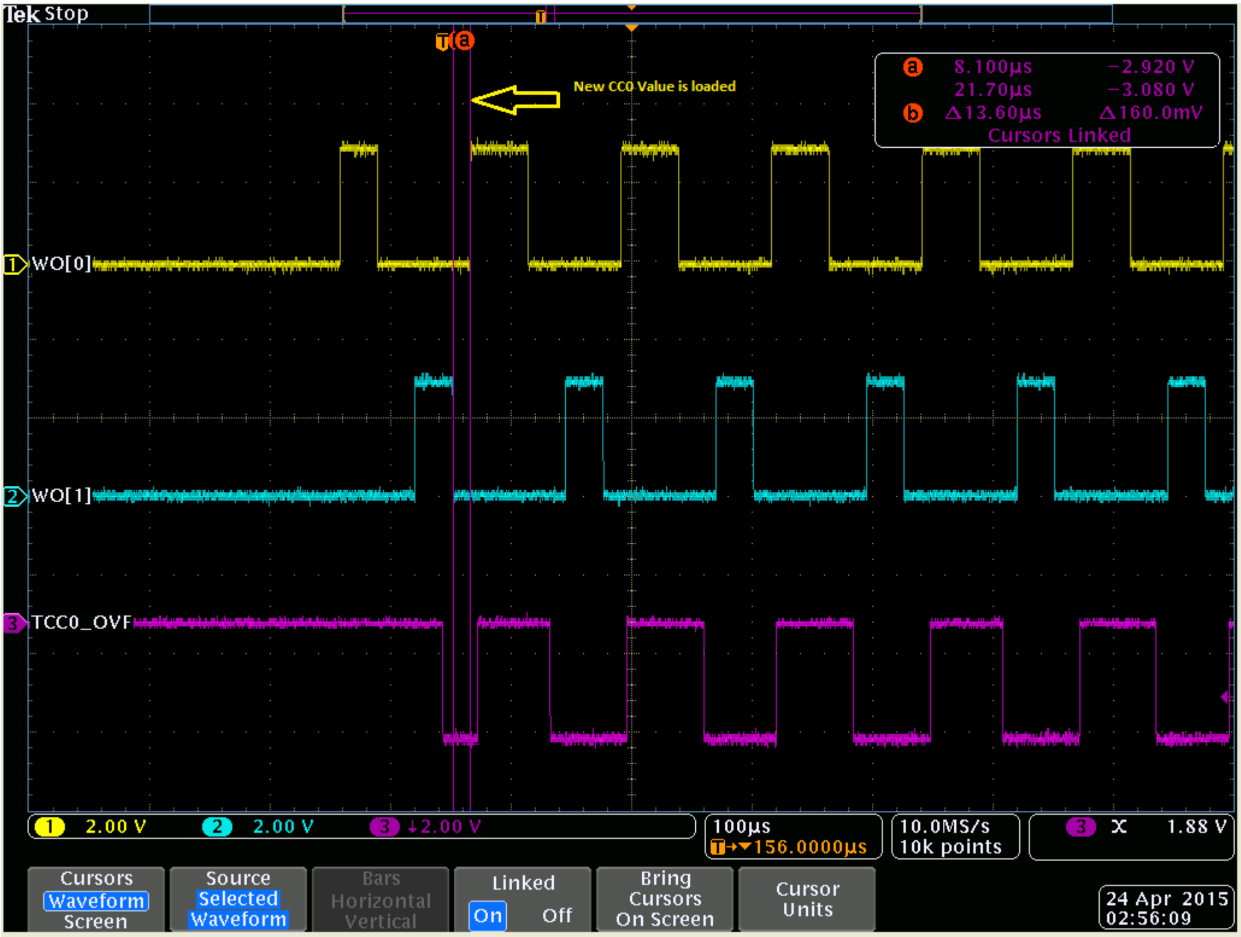
PA10 – TCC_OVF output

TCC Clock – 8MHz

CC0(A) initial = 250

CC0(A) after DMA = 120

Figure 5-4. DMA Operation with RAMP2A Mode



5.3 Fault Blanking Prescaler

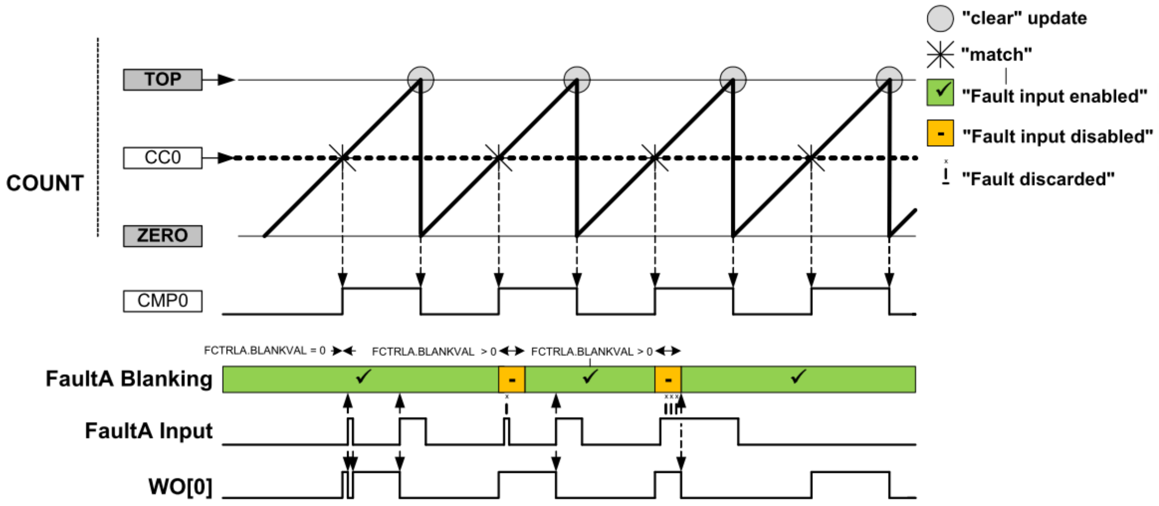
Provides a way to disable fault input just after a selected waveform output edge to prevent false fault triggering because of signal bouncing on fault signal, as shown in Figure 5-5. Blanking can be enabled by writing the edge triggering configuration to the Faultn Blanking Mode bits in the Recoverable FaultnConfiguration register (FCTRLn.BLANK), and the number of clock cycles to blank is written to the Faultn Blanking Time bits in the Recoverable Faultn Configuration register (FCTRLn.BLANKVAL). The blank counter can be prescaled by x64 factor. The maximum blanking time is:

256 / (96 x 10^6) = 2.66µs for 96MHz peripheral clock frequency (GCLK_TCCx)

256 x 64/ (96 x 10^6) = 170µs for prescaled and 96MHz peripheral clock frequency

256 / (1 x 10^6) = 256µs for 1MHz peripheral clock frequency (GCLK_TCCx)

Figure 5-5. Fault Blanking Operation with Inverted Polarity



5.3.2 Code Snippet

```
#if (ENABLE_FAULT_BALNKING == true)

       /* set fault blanking prescaler*/
       tcc_inst.hw->FCTRLA.reg |= 1 << BLANKPRESC;
       tcc_inst.hw->FCTRLB.reg |= 1 << BLANKPRESC;

       /* Enable recoverable fault event */
       struct tcc_events events;
       memset(&events, 0, sizeof(struct tcc_events));
       events.on_event_perform_channel_action[0] = true;
       events.on_event_perform_channel_action[1] = true;
       tcc_enable_events(&tcc_inst, &events);

#endif
```

5.3.3 Waveform Output

Figure 5-6. Without Fault Blanking Enabled

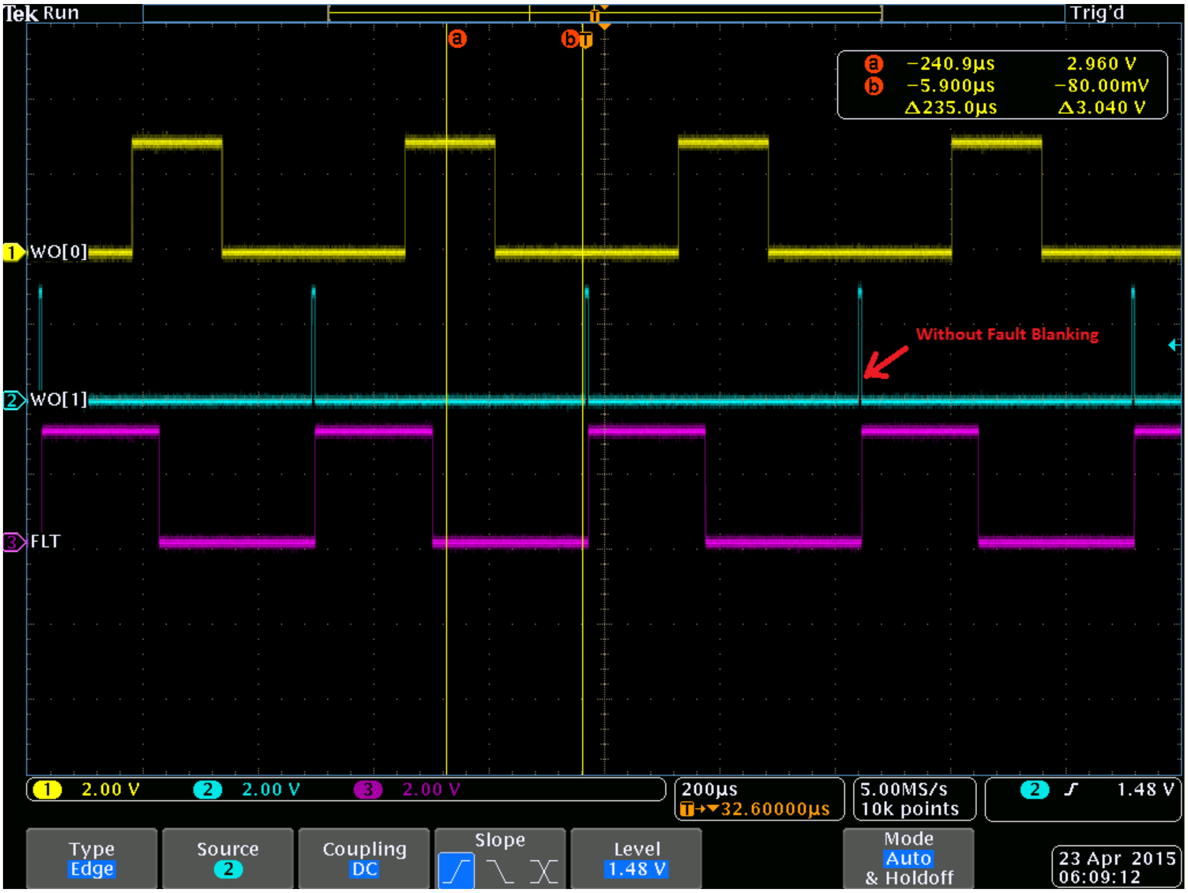
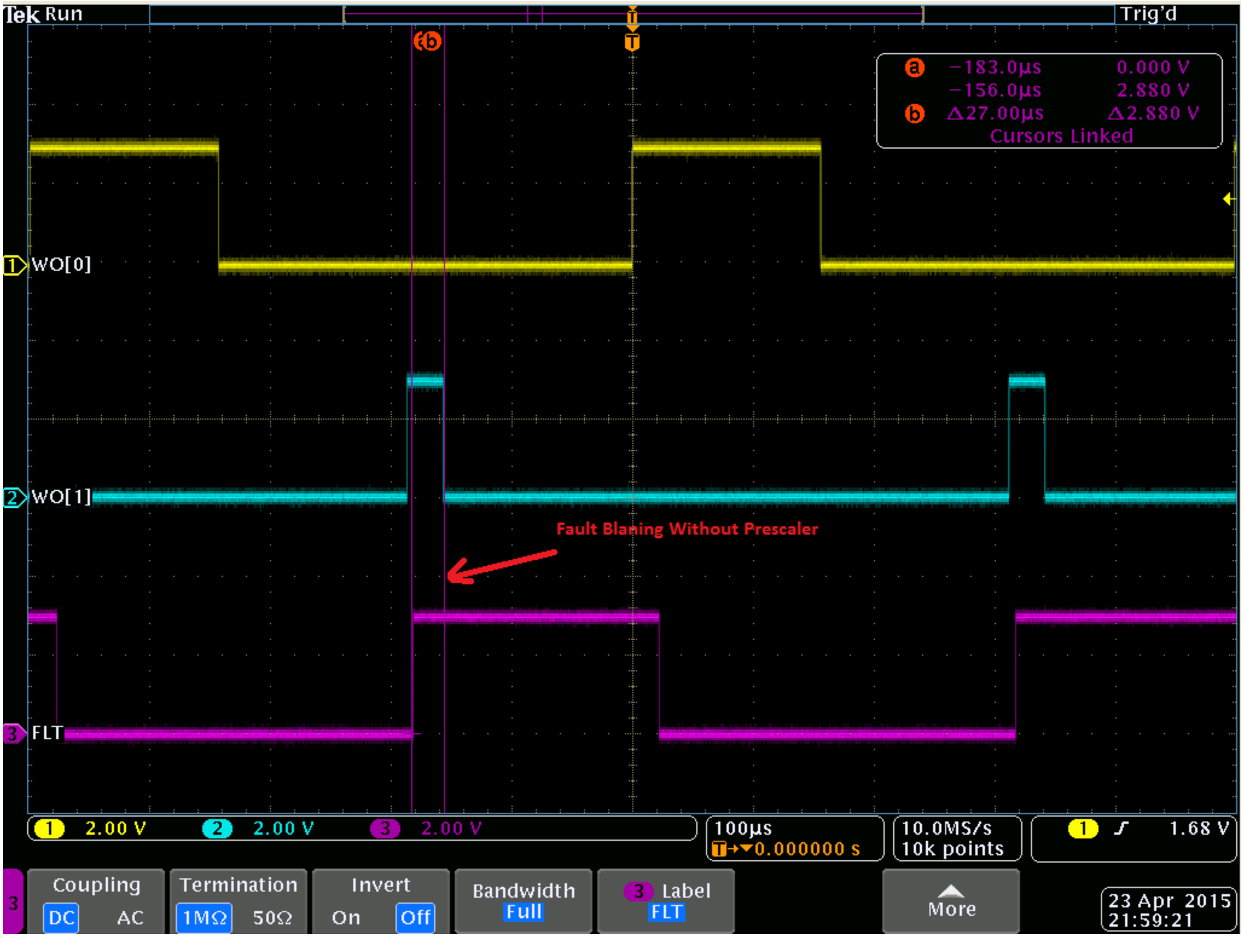


Figure 5-7. Fault Blanking with Prescaler Disabled



TCC Clock = 8MHz

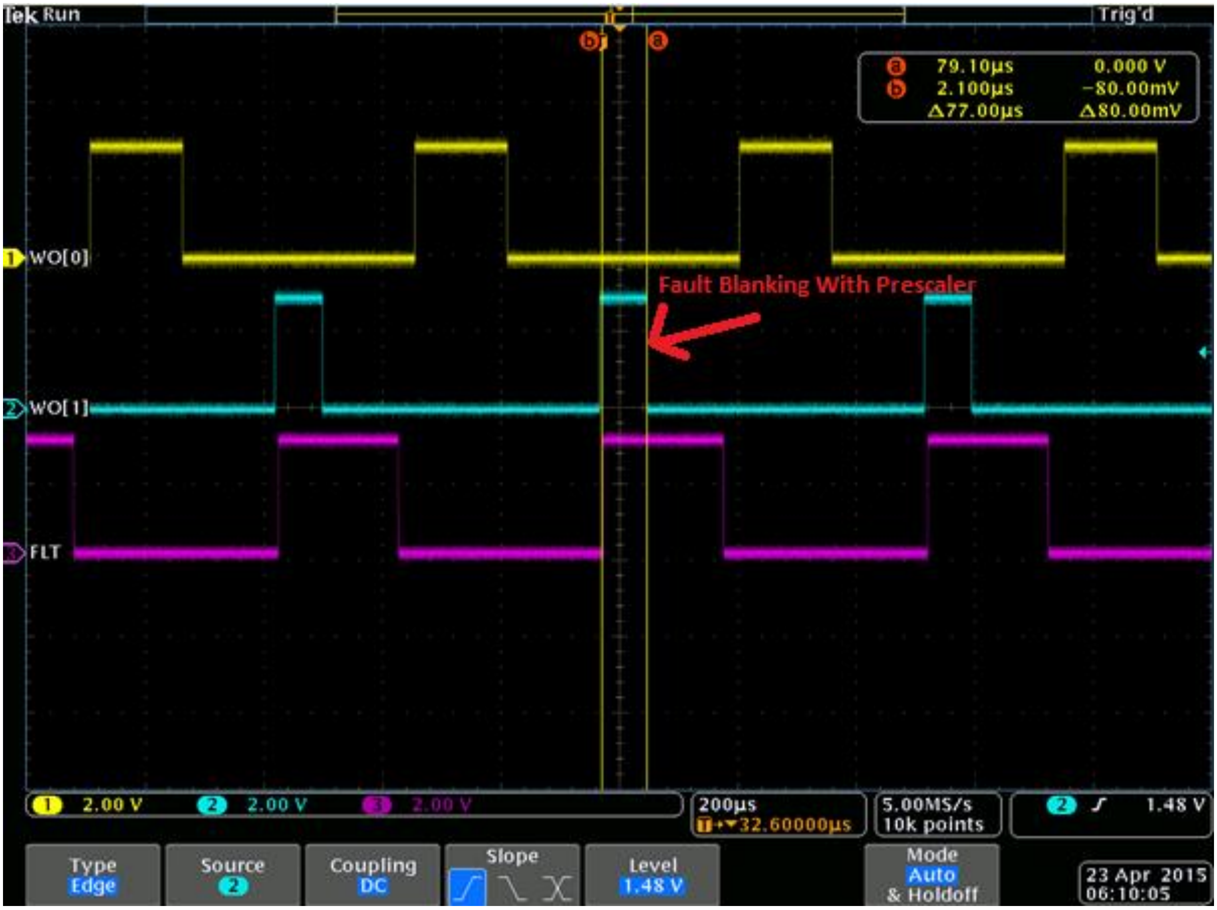
Blank Cycles = 240

Blanking Prescaler = Disabled

Blank Time = (Blank Cycles) / (TCC Clock)

Blank time = (240) / (8000000) = 30µs

Figure 5-8. Fault Blanking with Prescaler Enabled



a 79.10µs 0.000 V
b 2.100µs −80.00mV
Δ77.00µs Δ80.00mV

1 2.00 V 2 2.00 V 3 2.00 V

200µs
T→▾32.60000µs

5.00MS/s
10k points

2 ⨍ 1.48 V

Type Edge

Source 2

Coupling DC

Slope

Level 1.48 V

Mode Auto & Holdoff

23 Apr 2015
06:10:05

TCC Clock = 8MHz

Blank Cycles = 10

Blanking Prescaler = Enabled

Blank Time = (Blank Cycles x 64) / (TCC Clock)

Blank time = (10 x 64) / (8000000) = 80µs

6 Revision History

Doc Rev.	Date	Comments
42454A	06/2015	Initial document release.

Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.