AVR1001: Getting Started With the XMEGA Event System

Features

- **Flexible routing of peripheral events**
  - **8 configurable event channels**
  - **Signal filtering**
- **Ability to control peripherals independent of CPU**
- **Quadrature decoding**

1 Introduction

The XMEGA™ event system is a set of features that allows peripherals to interact without intervention from the CPU. Several peripheral modules can generate events, often on the same conditions as interrupt requests. These events are routed through the event routing system to the event users, where certain actions can be triggered by the event. The CPU is not involved in this process, except in the setup phase. As an example, it is possible to trigger a Timer/Counter input capture when a user presses a button, or start an analog to digital conversion at the overflow of a timer/counter. When fully utilizing the power of the event system, it is possible to configure the chip to do complex operations, with very little intervention from the CPU, saving both valuable program memory and execution time.

8-bit AVR® Microcontrollers

Application Note

Rev. 8071A-AVR-02/08

2 Event System Overview

The event system can be divided into three distinct parts:

- Event generators, with one or more event sources
- The event routing network
- Event users

This chapter will explain the details of how these parts interact.

2.1 What is an Event?

An event, as used in this document, is an indication that a change of state within a peripheral has occurred. A peripheral capable of generating events is called an event generator. One event generator may be able to generate events on several changes within the peripheral. Each of these is an individual event source. As an example, a Timer/Counter module is an event generator with several event sources, since it can generate events on overflow, error and compare/capture.

2.2 Event Types

Two types of events exist in the XMEGA event system, signaling events and data events. A signaling event does not contain any information except the fact that a change has occurred. A data event contains additional information about the change of state. The encoding of a data event is determined by the event source.

Since data events are only used in a few peripherals, they are not covered in detail until chapter 3. In the rest of this document, the word “event” will be used when referring to signaling events, except in situations where there is a need to distinguish between the two.

2.3 Event Generators

An event generator is a peripheral module having one or more event sources. There is generally a strong correlation between the available event sources and the available interrupt and DMA trigger sources belonging to a peripheral module. An event generator is feeding all its event sources to the event routing system, and is not aware of which event sources is being used by other modules.

2.4 The Event Routing Network

The Event Routing Network handles the routing of events from the event generator to the event user. The Event Routing Network consists of 8 equal event channels. Each channel consists of a multiplexer (controlled by the CH*n*MUX register) and a control and filtering logic (controlled by the CH*n*CTRL register), where *n* is the channel number.

Every event source from every event generator is connected to the inputs of each of the eight multiplexers. This means that each event channel can be connected to any event source. Several event channels can also choose to relay the same event source.

The filtering and special functions of the event channels are covered in chapter 3.

2.5 Event Users

An event user is a peripheral module that can make use of an event to trigger an action, referred to as an event action. An event user selects the event source to react to by selecting an event channel. The actual event source is determined by the multiplexer setting in the selected event channel.

Event users can also be event generators. For example Timer/Counter modules have several event sources, and can also use an event from another peripheral module to trigger an input capture.

2.6 Event Timing

An event usually lasts for one clock cycle. Some event sources are able to generate events continuously on some conditions, but this is not covered in detail here.

When an event is generated, it takes up to two clock cycles before the event action is triggered. It takes up to one clock cycle from the event has occurred until it is registered by the event routing network, and then another clock cycle to propagate the event to the event user.

Since it will never take more than two clock cycles for an event to trigger an event action, the event system provides more deterministic timing than using interrupts.

2.7 Manual Event Generation

It is possible to generate events either from software or using the on-chip debugging system. The generated events are injected directly in the event channels. The event channel does not need to have an event source associated with it to use the manual event generation possibilities. If an event source is associated with the event channel, the manually generated event has priority and will override the peripheral event.

Two registers are used for manual event generation, STROBE and DATA. The event generation is triggered by a write to the STROBE register. When generating signaling events, only the STROBE register is needed. When generating data events, both STROBE and DATA must be used and STROBE must be written after DATA.

The STROBE and DATA registers contain one bit for each event channel. Bit *n* corresponds to event channel *n*. It is possible to generate events on several channels at the same time by writing to several channels at once.

Table 2-1 shows the mapping between the value written to the STROBE and DATA register, and the generated event. Notice that only Data Event 02 and Data Event 03 will result in an event that is recognized by an event user that expects a signaling event. Data events are covered more in detail in chapter 3.

Table 2-1. Event encoding

STROBE	DATA	Data Event User	Signaling Event User
0	0	No Event	No Event
0	1	Data Event 01	No Event
1	0	Data Event 02	Signaling Event
1	1	Data Event 03	Signaling Event

2.7.1 Manually Generating Signaling Events

When generating signaling events, only the STROBE register needs to be written. For example, writing 0x05 to the STROBE register will generate a signaling event on channels 0 and 2 simultaneously. When the events have been issued, the STROBE register as well as the DATA register will be automatically cleared.

2.7.2 Manually Generating Data Events

When generating data events, the DATA register must be written first, followed by the STROBE register. The events will not be generated before the STROBE register is written. When the event(s) have been issued, the DATA and STROBE registers are automatically cleared. For example, writing 0x05 to the DATA register, then writing 0x01 to the STROBE register will cause a Data Event 03 to be generated on channel 0, while Data Event 01 will be generated on channel 2. Notice that an event user listening for signaling events will recognize an event on channel 0, but not on channel 2.

2.8 Events and Sleep Modes

The event system is operative in Active and Idle mode. In all other sleep modes, peripheral modules will not be able to communicate using the event system.

3 Advanced Usage

In addition to relaying information about changes in peripherals, the event system has built-in functionality to handle advanced tasks such as filtering and quadrature decoding. These functions need special handling in the event channels, controlled by the CH*n*CTRL register, where *n* is the channel number.

3.1 Filtering

Each event channel has an associated digital filter, controlled by the DIGFILT[2:0] field in the CH*n*CTRL register. The input on the multiplexer for the event channel must be logic high for (DIGFILT + 1) clock cycles before an event is signaled through the event channel.

The digital filter is typically used when an I/O port pin is used to produce events, to prevent multiple consecutive events caused by switch bouncing or electric noise.

3.2 Quadrature Decoding

The event system has extensions that make it possible to use a Timer/Counter as a quadrature decoder. Quadrature encoders are commonly used as position sensors in motor application, but are also found in other rotary sensors, such as the ball tracker in computer mice. The application note AVR1600 covers the setup and use of the quadrature decoder in greater detail.

4 Examples

This chapter lists a range of useful ways to use the event system. Each example has instructions on how to set up each part of the system. Additional and more detailed examples can be found in application notes for the specific event users.

4.1 Input Capture

Any event source can be used to trigger an input capture on one of the Timer/Counter modules. This can be used to timestamp events.

4.1.1 Configuration

This example shows how to configure TCC0 for input capture, triggered by a change on the input of the I/O port pin PD0.

1. Configure TCC0 with the desired frequency and period.
2. Configure event channel 0 to use the PD0 event as event channel multiplexer input.
3. Select event channel 0 as event source for TCC0.
4. Set event action for the TCC0 to “Input capture”.
5. Enable Compare or Capture Channel A.
6. Configure PD0 as input, and sense both edges.

TCC0 will now perform an input capture every time there is a logic change on the input of PD0.

4.2 Sweep of 4 ADC Channels on Timer/Counter Overflow

The ADC can be configured to do a sweep of four channels on any event. In this example, a Timer/Counter overflow event is used. This can be very useful when the Timer/Counter is used for PWM generation, as the ADC sampling can be synchronized to the PWM.

4.2.1 Configuration

This example shows how to configure a sweep of the four virtual channels of ADCA on an overflow of TCC0, using event channel 0.

1. Configure TCC0 with the desired frequency and period.
2. Select the TCC0 overflow event as an event source for event channel 0.
3. Configure ADCA for a four-channel sweep.
4. Configure ADCA to start sweep on event, using event channel 0.

4.3 32-bit Timer/Counter with 32-bit Input Capture

Sometimes, it is beneficial to have a Timer/Counter with a resolution of more than 16 bits. The event system makes it possible to cascade two 16-bit Timer/Counters and use them as one 32-bit Timer/Counter, with input capture support.

For more detailed information about using the Timer/Counters in 32-bit mode, please see the application note AVR1306.

4.3.1 Configuration

This example shows how to configure TCC0 and TCC1 as one 32-bit Timer/Counter with input capture channel A triggered by a logic change on PD0, routed through event channel 1. Event channel 0 is used for overflow propagation.

1. Configure PD0 as input, sense on both edges.
2. Select TCC0 overflow event as event multiplexer input for event channel 0.

3. Select PD0 as event multiplexer input for event channel 1.
4. Select event channel 0 as clock source for TCC1.
5. Set EVACT in CTRLD to input capture as event action for both TCC0 and TCC1.
6. Set EVSEL bits in CTRLD to event channel 1 for both TCC0 and TCC1.
7. Set the EVDLY bit in TCC1.CTRLD to delay the event to the high word TC.
8. Enable input capture channel A for both TCC0 and TCC1.
9. Select system clock as clock source for TCC0.

A change in logic level on PD0 will now trigger a 32-bit input capture on channel A of TCC0 and TCC1.

4.4 Event Counting

It is possible to use an event channel as a clock source for a Timer/Counter. This can be used to count the number of events on an event channel.

4.4.1 Configuration

In this example, TCC0 will be used to count the number of times a button connected to PD0 has been pressed.

1. Configure PD0 to trigger on rising or falling edge.
2. Select PD0 as event source for event channel 0.
3. Set the digital filter for event channel 0 to the highest possible value.
4. Configure event channel 0 as the clock source for TCC0.

TCC0 will now count the number of times the button connected to PD0 has been pressed.

4.5 Sample Rate Distribution

An event channel can be set up to function as a sample rate distribution channel. Many control systems need to sample and output data at regular intervals, called the sample rate. Using one Timer/Counter as a sample rate generator, it is possible to distribute the overflow and/or compare match events from the Timer/Counter to all ADCs, DACs, PWMs and other peripheral modules that need to perform actions at regular intervals.

5 Driver/Example Implementation

The included driver has functions that can be used to configure the Event system. The driver is written in ANSI® C, and should compile on all compilers with Xmega support.

Note that this driver is **not** written with high performance in mind. It is designed as a library to get started with the Xmega Event system and an easy-to-use framework for rapid prototyping. For time and code space critical application development, consider replacing function calls with macros or direct access to registers.

5.1 Files

The source code package consists of the following files:

- event_system_driver.c – Event system driver source file
- event_system_driver.h – Event system driver header file
- event_system_example.c – Examples of using the event system

5.2 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit http://www.doxygen.org. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature