



8-bit AVR[®]
Microcontrollers

Application Note

AVR073: Accessing 10- and 16-bit registers in ATtiny261/461/861

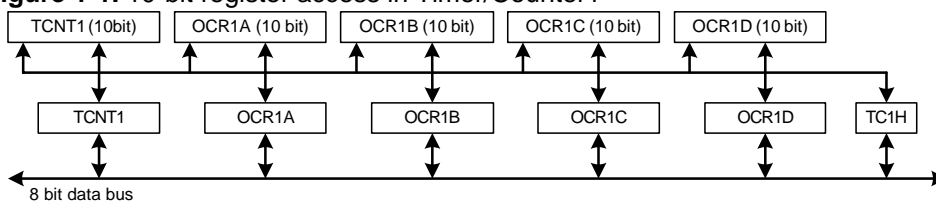
Features

- Access macros for 16-bit Timer/Counter0 registers
- Access macros for 10-bit Timer/counter1 registers
- All routines written in ANSI[®] C language

1 Introduction

The ATtiny261/461/861 microcontroller family includes Timer/Counter modules that can be used in 10- and 16-bit modes. Since the AVR[®] data bus is 8 bits wide, special considerations must be made when accessing 10- and 16-bit registers. Most AVR C compilers handle 16-bit (and 10-bit) register accesses transparently when the two 8-bit registers that makes up the 16-bit value can be accessed at consecutive addresses. In the ATtiny261/461/861 family, not all 10 and 16-bit registers are not addressed at consecutive addresses, so compiler support is limited to accessing the individual 8-bit registers. This application note explains how 10- and 16-bit accesses should be handled when using the ATtiny261/461/861 family of microcontrollers. A complete set of C macros for accessing 10- and 16-bit registers is also included with this application note.

Figure 1-1. 10-bit register access in Timer/Counter1



Rev. 8027B-AVR-01/08





2 Accessing 10- and 16-bit registers in ATtiny261/461/861

The two modules that use 10- and 16-bit registers that need special consideration in the ATtiny261/461/861:

- Timer/Counter0 (16-bit)
- Timer/Counter1 (10-bit)

2.1 Timer/Counter0

The Timer/Counter0 (TC0) module can be used in both 8- and 16-bit modes. When operated in 8-bit mode, no special considerations need to be made. When operated in 16-bit mode, special considerations must be made when accessing TCNT0L/H and OCR0A/B.

TC0 has a single 8-bit register for temporary storing of the high byte of the 16-bit access. This register is used both for access to TCNT0 in 16-bit mode and OCR0 when used in 16-bit input capture mode. Accessing the low byte (TCNT0L or OCR0A) triggers the 16-bit read/write operation: when the low byte of a 16-bit register is written by the CPU, the high byte is stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the CPU reads the low byte of a 16-bit register, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

2.1.1 16-bit write operation

A 16-bit write operation can be performed by first writing the high byte to the TCNT0H register. The high byte will be copied to the temporary register until TCNT0L is written. TCNT0L is then written, triggering the 16-bit value to be copied in one clock cycle.

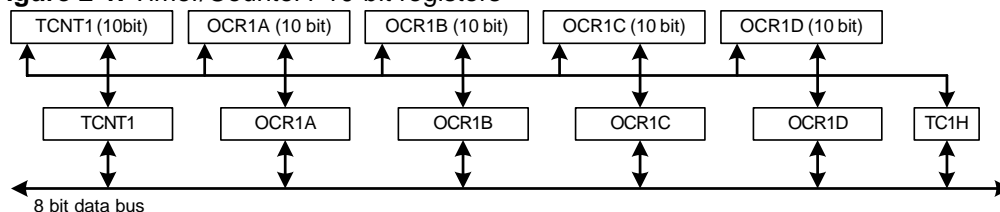
2.1.2 16-bit read operation

A 16-bit read operation is initiated by reading the low byte register (TCNT0L or OCR0A). This will trigger a copy of the high byte register to the temporary register. Reading the high byte register (TCNT0H or OCR0B) returns the previously stored high byte.

2.2 Timer/Counter1

Timer/Counter1 (TC1) is a 10-/8-bit timer/counter module. When TC1 is used in 8-bit mode, no special considerations need to be made. In 10-bit mode, access to 10-bit registers is performed using one common 2-bit register TC1H, which holds the high byte in both read and write operations. The 10-bit TC1 registers and their connections to the 8-bit bus are shown in Figure 2-1.

Figure 2-1. Timer/Counter1 10-bit registers



2.2.1 10-bit write operation

A 10-bit write operation can be performed by first writing the two most significant bits to the common TC1H register. When the low byte is written by the CPU, both the 2 bits stored in the TC1H register and the low byte are copied into the 10-bit register in the same clock cycle.

The TC1H register is **not cleared** by a write or read operation. This makes it possible to write to several 10-bit TC1 registers by writing TC1H only once. Note that if an 8-bit value is written to a 10-bit TC1 register and the TC1H register is not used, the TC1H value from a previous read or write operation might be copied to the 10-bit target register. The TC1H register should therefore be explicitly written whenever writing to the 10-bit TC1 registers unless the value of TC1H is known.

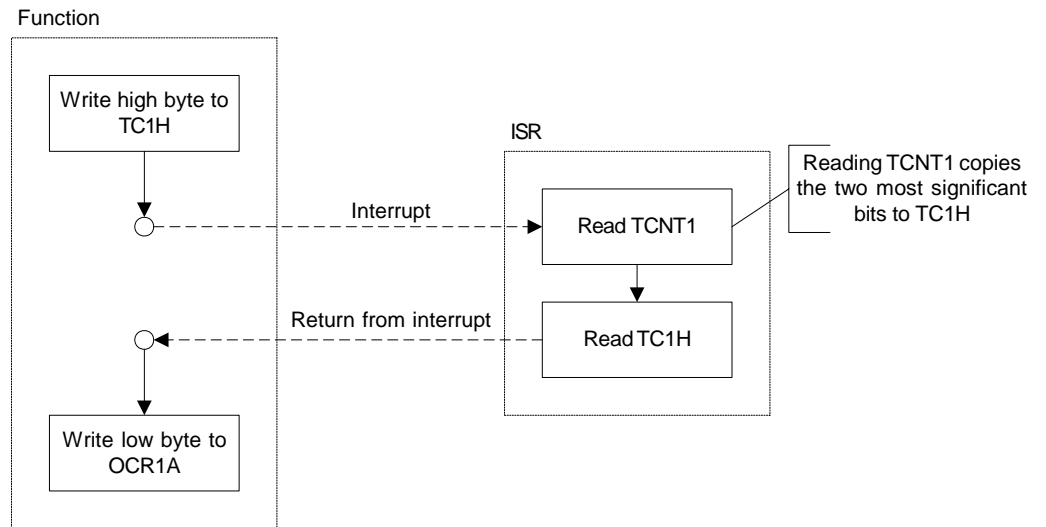
2.2.2 10-bit read operation

A 10-bit read can be performed by first reading the low byte of the 10-bit register. This will trigger the two most significant bits to be copied to the TC1H register.

2.3 Ensuring atomic access to multi-byte registers

In a system using interrupts, accessing multi-byte registers can constitute a risk. Consider the situation in Figure 2-2. A 10-bit write is about to be performed to OCR1A. The TC1H register has just been loaded with the high byte and the low byte is about to be written. However, an interrupt occurs between two write operations, executing the interrupt service routine (ISR), which reads the value of TCNT1. As TCNT1 is accessed, the high byte is copied to the TC1H register, to facilitate the 10-bit read in the ISR. When execution returns to the function that was interrupted, TC1H is corrupted and the application has no means to detect that this has happened. When the second part of the write to OCR1A is performed, the wrong high byte is copied due to the corruption of TC1H.

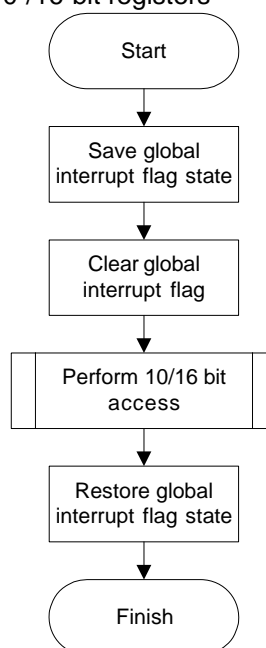
Figure 2-2. Interrupt corruption



The situation in Figure 2-2 can be prevented by making sure that 10 and 16-bit registers are always accessed in one atomic operation. This is accomplished by making sure that interrupts are disabled during all 10 and 16-bit accesses. The most general way to do this is shown in Figure 2-3. This method is always safe to use everywhere in the code, but it adds to the number of instructions needed to perform this basic operation. It is therefore important to understand when it is necessary to disable interrupts during multi-byte access.

- Most initialization is usually performed before interrupts are enabled. There is no need to disable interrupts at this stage.
- The AVR core automatically clears the global interrupt flag when an interrupt is being serviced. It is therefore safe to access multi-byte registers during the execution of ISRs, provided the global interrupt flag is not manually set within that ISR.
- If no ISRs access any of the multi-byte registers in question, it is safe to access them without disabling interrupts.

Figure 2-3. Atomic access of 10-/16-bit registers



2.4 Synchronization issues

Note that due to synchronization circuitry in TC1, several TC1 registers cannot immediately be read back after being written. Please refer to the “Synchronization” section in the ATtiny261/461/861 data sheet for more details.

2.5 Multi-byte registers not covered in this document

There are other multi-byte registers in the ATtiny261/461/861 that are not covered by this document (E.g. ADCL/H). These are addressed consecutively, allowing the multi-byte access to be transparently handled by the compiler, and are thus not covered here. Note, however, that the programmer is still responsible for ensuring atomic access to these registers as described in section 2.3 of this document

2.6 Included macros

Included with this application note is a set of macros written in ANSI C designed to perform the operations described in this application note. For each operation, two macros are available, one that is interrupt-safe and one that is not.

The source code has been documented with Doxygen comments for automatic documentation generation. Please open the ‘readme.html’ file in the source code root directory for more information.

2.7 Compiler support

All source code included with this application note is written in ANSI C for maximum portability. At the time of release of this application note, only the IAR® and AVR GCC compilers had included support for the ATtiny261/461/861 family. The source code has therefore only been tested on these compilers.





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

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 ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, 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 OF 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 product 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's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.