Understanding the *AVR* ICEPRO I/O Registers

Introduction

This document describes the I/O Register views seen in AVR Studio when using the ICEPRO emulator. Most I/O registers behave as expected, but in the UART, SPI and Timer/Counters, some registers have dual meaning. This document will be useful when modifying these registers, both from software and from the PC through AVR Studio.

The table on the following pages describes:

- How the register is modified by software.
- The value shown in AVR Studio when the register is read.
- The effect of writing a new value into the register from AVR Studio.

AVR® ICEPRO I/O Registers

Application Note

Rev. 1015A–04/98

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$3F	SREG	read same/write same(2)	*current register contents*	*update register contents(1)*
$3E	SPH	read same/write same(2)	*current register contents*	*update register contents(1)*
$3D	SPL	read same/write same(2)	*current register contents*	*update register contents(1)*
$3B	GIMSK	read same/write same(2)	*current register contents*	*update register contents(1)*
$3A	GIFR	• Writing a one to any flag will clear this flag • Writing a zero to any flag will always have no effect • A flag can't be set by AVR software	*current register contents*	• Flags are cleared by writing a zero • Flags are set by writing a one
$39	TIMSK	read same/write same(2)	*current register contents*	update register contents(1)
$38	TIFR	• Writing a one to any flag will clear this flag • Writing a zero to any flag will always have no effect • A flag can't be set by AVR software	*current register contents*	• Flags are cleared by writing a zero • Flags are set by writing a one
$35	MCUCR	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$33	TCCR0	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$32	TCNT0	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$2F	TCCR1A	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$2E	TCCR1B	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$2D	TCNT1H	• Reading TCNT1H returns the value stored in the common TEMP1 register • A write to TCNT1H stores the new value in the common TEMP1 register • The value stored in TCNT1H is transferred into the common TEMP1 register when is TCNT1L is read • The value stored in the common TEMP1 register is transferred into TCNT1H when TCNT1L is written	TEMP1 The common TEMP1 register is NOT updated when AVR Studio reads TCNT1L See separate description for PWM mode	• A write to TCNT1H updates the common TEMP1 register • It is not possible to update TCNT1H directly from AVR Studio • TCNT1H will be updated only when writing a new value to TCNT1L from the AVR software

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$2C	TCNT1L	*read same/write same*[2] • When reading TCNT1L, the value in TCNT1H is transferred into the common TEMP1 register • When writing TCNT1L, the value in the common TEMP1 register is transferred into TCNT1H	*current register contents* The common TEMP1 register is NOT updated when AVR Studio reads TCNT1L See separate description for PWM mode	*update register contents*[1] • When TCNT1L is written from AVR Studio, this will NOT update TCNT1H
$2B	OCR1AH	• Reading OCR1AH returns the value stored in the register • A write to OCR1AH stores the new value into the common TEMP1 register • The common TEMP1 register is not used when reading OCR1AH • The value stored in the common TEMP1 register is transferred into OCR1AH when OCR1AL is written	*current register contents* See separate description for PWM mode	• A write to OCR1AH updates the common TEMP1 register • It is not possible to update ORC1AH directly from AVR Studio • OCR1AH will be updated only when writing a new value to OCR1AL from the AVR software
$2A	OCR1AL	*read same/write same*[2] • The value stored in the common TEMP1 register is transferred into OCR1AH when OCR1AL is written	*current register contents* See separate description for PWM mode	*update register contents*[1] • When OCR1AL is written from AVR Studio, this will NOT update OCR1AH
$29	OCR1BH	• Reading OCR1BH returns the value stored in the register • A write to OCR1BH stores the new value into the common TEMP1 register • The common TEMP1 register is not used when reading OCR1BH • The value stored in the common TEMP1 register is transferred into OCR1BH when OCR1BL is written	*current register contents* See separate description for PWM mode	• A write to OCR1BH updates the common TEMP1 register • It is not possible to update ORC1BH directly from AVR Studio • OCR1BH will be updated only when writing a new value to OCR1BL from the AVR software
$28	OCR1BL	*read same/write same*[2] • The value stored in the common TEMP1 register is transferred into OCR1AH when OCR1AL is written	*current register contents* See separate description for PWM mode	*update register contents*[1] • When OCR1AL is written from AVR Studio, this will NOT update OCR1AH

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$25	ICR1H	• Reading ICR1H returns the value stored in the common TEMP1 register • A write to ICR1H is always ignored • The value stored in TCNT1H is transferred into the common TEMP1 register when is TCNT1L is read	TEMP1 The common TEMP1 register is NOT updated when AVR Studio reads ICR1L	• A write to ICR1H is always ignored • It is not possible to update ICR1H from AVR Studio
$24	ICR1L	• Reading ICR1L returns the value stored in the register • A write to ICR1L is always ignored • When reading ICR1L, the value in ICR1H is transferred into the common TEMP1 register • When writing ICR1L, the value in the common TEMP1 register is transferred into ICR1H	*current register contents* The common TEMP1 register is NOT updated when AVR Studio reads ICR1L	• A write to ICR1L is always ignored • It is not possible to update ICR1L from AVR Studio
$21	WDTCR	*read same/write same*[2] • This register is manipulated as described in the corresponding databook • The WDE bit can be cleared only when the WDTOE bit is set AT90S1200: The WDTOE bit is not implemented in this device. The WDE bit can be cleared at any time.	*current register contents* The Watchdog Timer will not pause when single stepping. This will cause early resets when single-stepping	*update register contents*[1]
$1F	EEARH	AT90S1200/2313/4414: No EEARH register implemented Reading EEARH returns an undefined value AT90S8515: *read same/write same*[2] • Any write to EEARH during an EEPROM Write cycle is ignored	*current register contents*	AT90S1200/2313/4414: These devices have no EEARH register. Reading EEARH always returns an undefined value AT90S8515: A write to EEARH is always ignored The Emulator EEPROM contents can be updated directly, there is no need to manipulate the EEARH for this purpose

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$1E	EEARL	*read same/write same*[2] • Any write to EEARL during an EEPROM Write cycle is ignored AT90S1200: Software should ensure that EEAR is never altered during a write cycle, as there is no mechanism to prevent the update. Updating EEAR will work in the emulator, but not in the actual device	*current register contents*	A write to EEARL is always ignored AT90S1200: EEAR can be updated at any time The Emulator EEPROM contents can be updated directly, there is no need to manipulate the EEARL for this purpose
$1D	EEDR	*read same/write same*[2] • Any write to EEDR during an EEPROM Write cycle is ignored AT90S1200: Software should ensure that EEDR is never altered during a write cycle, as there is no mechanism to prevent the update. Updating EEDR will work in the emulator, but not in the actual device	*current register contents*	*update register contents*[1] • The special EEPROM window in AVR Studio does not allow register updates. But EEDR can be updated from the general I/O Memory view • The Emulator EEPROM contents can be updated directly, there is no need to manipulate the EEDR for this purpose

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$1C	EECR	*read same/write same*[2] • This register is manipulated as described in the corresponding databook • The EERE bit never reads as one, as the read operation lasts only one cycle • The EEWE bit can be set only when the EEMWE bit is set • A write to the EEWE bit when EEMWE is zero is always ignored • Writing a zero to EEWE will not clear the flag • After the EEPROM Write cycle has ended, EEWE will clear automatically. In the emulator, EEWE clears after 16 XTAL cycles. In the chip, EEWE clears after 1.5-4.0 ms. Observe that this delay will be considerably more than 16 XTAL cycles AT90S1200: • The EEMWE bit is not implemented in this device. The EEWE can be set at any time	*current register contents*	• A write to EERE will is always ignored • A write to EEWE is always ignored • A write to EEMWE will set EEMWE. EEMWE is cleared after four cycles • The Emulator EEPROM contents can be updated directly, there is no need to manipulate the EEDR for this purpose AT90S1200: • The EEMWE bit is not implemented. • Writing a one to EEWE will enable the interface drivers and cause a contention. The I/O memory view will show this contention. No write to the EEPROM will take place
$1B	PORTA	*read same/write same*[2]	*current register contents*	*update register contents*[1]
$1A	PINA	• *Reading PINA read current voltage level on physical port* • *A write to PINA is always ignored*	*current register contents*	*A write to PINA is always ignored*
$19	DDRA	*read same/write same*[2]	*current register contents*	*update register contents*[1]
$18	PORTB	• *Reading PINB read current voltage level on physical port* • *A write to PINB is always ignored*	*current register contents*	*A write to PINB is always ignored*
$17	PINB	*read same/write same*[2]	*current register contents*	*update register contents*[1]
$16	DDRB	*read same/write same*[2]	*current register contents*	*update register contents*[1]

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$15	PORTC	• *Reading PINC read current voltage level on physical port* • *A write to PINC is always ignored*	*current register contents*	*A write to PINC is always ignored*
$14	PINC	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$13	DDRC	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$12	PORTD	• *Reading PIND read current voltage level on physical port* • *A write to PIND is always ignored*	*current register contents*	*A write to PIND is always ignored*
$11	PIND	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$10	DDRD	*read same/write same(2)*	*current register contents*	*update register contents(1)*
$0F	SPDR	• Reading SPDR will return the current contents of the Receive Data Buffer • Writing to SPDR will update the value in the 8-bit SPI shift register • When a transmission is in progress, writing to SPDR will set the WCOL flag. The SPI shift register will ignore the event and continue the transmission undisturbed	Current contents of the Receive Data Buffer	• Writing to SPDR will update the value in the 8-bit SPI shift register • When a transmission is in progress, writing to SPDR will set the WCOL flag. The SPI shift register will ignore the event and continue the transmission undisturbed Warning: AVR Studio has an SPI Module Peripheral window. Each keystroke in the SPDR register box will transmit a new value to SPDR. This usually generates a result other than the desired, like setting the WCOL flag. Instead, SPDR should be updated from the general I/O memory view
$0E	SPSR	• Any write to this register is ignored • SPIF is cleared by reading SPSR when SPIF is set, then accessing SPDR • WCOL is cleared by reading SPSR when WCOL set, then accessing SPDR	*current register contents* • Reading SPSR into AVR Studio is not recognized as an access to the SPSR register	• Any write to SPSR is ignored • SPIF and WCOL can not be set from AVR Studio A write to SPDR will qualify as an access to SPDR. If SPIF or WCOL have already been read by the AVR software, this write will clear SPIF or WCOL respectively

Adr	Register	AVR software read/write	Value shown in AVR Studio	AVR Studio write
$0D	SPCR	*read same/write same*[2] The MSTR bit is immediately cleared when a master contention is detected. The event occurs when: • The SPI is enabled • The SS line is input • The SS input reads low	*current register contents*	*update register contents*[1] If the emulator seemingly refuses to set the MSTR bit, this is caused by the master contention. This contention is avoided by externally driving the SS line high, or forcing the SS pin to an output
$0C	UDR	• Reading UDR will return the current contents of the Receive Data Buffer • Writing to UDR will update the value in the Transmit Data Buffer	Receive Buffer Contents • Reading UDR into AVR Studio is not recognized as an access to the UDR register	• Writing to UDR will update the value in the 8-bit Transmit Buffer Warning: AVR Studio has a UART Module Peripheral window. Each keystroke in the UDR register box will transmit a new value to UDR. This usually generates a result other than the desired. Instead, UDR should be manipulated from the general I/O memory view
$0B	USR	• RXC is cleared by reading UDR • A write to RXC is always ignored • TXC is cleared by writing a one to the flag • Writing a zero to TXC is always ignored • UDRE is always set if the Transmit Buffer is empty. • A write to UDRE is always ignored • FE is altered only when a new byte is transferred to the Receive Data Buffer • A write to FE is always ignored • OR is altered only when a byte is read from UDR • A write to OR is always ignored	*current register contents*	• A write to RXC is always ignored • TXC is cleared by writing a one to the flag • Writing a zero to TXC is always ignored • A write to UDRE is always ignored • A write to FE is always ignored • A write to OR is always ignored
$0A	UCR	*read same/write same*[2] RXB8 is a read only bit. Any write to this bit is ignored	*current register contents*	*update register contents*[1] Any write to the RXB8 bit is always ignored
$09	UBRR	*read same/write same*[2]	*current register contents*	*update register contents*[1]
$08	ACSR	*read same/write same*[2]	*current register contents*	*update register contents*[1]

Notes:
1. Most registers allow new values to be written from AVR Studio. Some of the devices emulated by the ICEPRO will not support all the I/O Registers. A write to an I/O Register not supported by the emulated device is always ignored. The value returned by a read operation is undefined, but normally the data value from the last successful read or write operation.
2. A write to this I/O register will store the new data value in the register. A read will return the current register contents. Some I/O registers will contain unused bits. These bits always return zero when the bit is read. A write to these bits is always ignored. Some of the devices emulated by the ICEPRO will not support all the I/O registers. A write to an I/O register not supported by the emulated device is always ignored. The value returned by a read operation is undefined, but normally the data value from the last successful read or write operation.

Timer/Counter 1 in PWM Mode

TCNT1L/TCNT1H: In PWM mode, these registers are read and written as usual. In this mode, the timer will operate as a 16-bit counter and change counting direction at TOP and zero. When writing a new value to TCNT1H, no bits will be truncated by hardware before the value is stored in the counter register. This truncation should be handled by software whenever necessary.

OCR1AL/OCR1AH: In PWM mode, the value written to these registers is stored in a separate 10-bit register OCR1A_TEMP. This register is not the same register as the TEMP1 register used when reading high bytes. In the ICEPRO, reading OCR1AL/OCR1AH will return the contents of OCR1AL/OCR1H, and not the contents of OCR1A_TEMP. This is also the case with 8515 revision A. Starting from revision B, however, the returned value will be the contents of OCR1A_TEMP. The two registers usually read the same value. They only differ when a value has been stored in OCR1A_TEMP, before the timer reaches TOP and latches this value into OCR1AL/OCR1AH. The change will allow the software to perform Read-Modify-Write changes to OCR1A more than once per PWM cycle.

OCR1BL/OCR1BH: These registers operate exactly like OCR1AL/OCR1AH. The 10-bit OCR1B_TEMP register is separate for OCR1BL/OCR1BH, and not shared with output compare A.

ICR1L/ICR1H: These registers have no special function when the timer is running in PWM mode. The values the software could expect to be returned from this register is naturally limited to the number of bits used for the PWM operation.