<avr/interrupt.h>: Interrupts

Global manipulation of the interrupt flag

The global interrupt flag is maintained in the I bit of the status register (SREG).

Handling interrupts frequently requires attention regarding atomic access to objects that could be altered by code running within an interrupt context, see <**util/atomic.h**>.

Frequently, interrupts are being disabled for periods of time in order to perform certain operations without being disturbed; see **Problems with reordering code** for things to be taken into account with respect to compiler optimizations.

#define **sei**()

#define **cli**()

Macros for writing interrupt handler functions

#define **ISR**(vector, attributes)

#define **SIGNAL**(vector)

#define **EMPTY_INTERRUPT**(vector)

#define **ISR_ALIAS**(vector, target_vector)

#define **reti**()

#define **BADISR_vect**

ISR attributes

#define **ISR_BLOCK**

#define **ISR_NOBLOCK**

#define **ISR_NAKED**

#define **ISR_ALIASOF**(target_vector)

Detailed Description

Note

This discussion of interrupts was originally taken from Rich Neswold's document. See **Acknowledgments**.

Introduction to avr-libc's interrupt handling

It's nearly impossible to find compilers that agree on how to handle interrupt code. Since the C language tries to stay away from machine dependent details, each compiler writer is forced to design their method of support.

In the AVR-GCC environment, the vector table is predefined to point to interrupt routines with predetermined names. By using the appropriate name, your routine will be called when the corresponding interrupt occurs. The device library provides a set of default interrupt routines, which will get used if you don't define your own.

Patching into the vector table is only one part of the problem. The compiler uses, by convention, a set of registers when it's normally executing compiler-generated code. It's important that these registers, as well as the status register, get saved and restored. The extra code needed to do this is enabled by tagging the interrupt function with `__attribute__((signal))`.

These details seem to make interrupt routines a little messy, but all these details are handled by the Interrupt API. An interrupt routine is defined with **ISR()**. This macro register and mark the routine as an interrupt handler for the specified peripheral. The following is an example definition of a handler for the ADC interrupt.

```
#include <avr/interrupt.h>
ISR(ADC_vect)
{
// user code here
}
```

Refer to the chapter explaining **assembler programming** for an explanation about interrupt routines written solely in assembler language.

Catch-all interrupt vector

If an unexpected interrupt occurs (interrupt is enabled and no handler is installed, which usually indicates a bug), then the default action is to reset the device by jumping to the reset vector. You can override this by supplying a function named `BADISR_vect` which should be defined with **ISR()** as such. (The name BADISR_vect is actually an alias for __vector_default. The latter must be used inside assembly code in case <**avr/interrupt.h**> is not included.)

```
#include <avr/interrupt.h>
ISR(BADISR_vect)
{
// user code here
}
```

Nested interrupts

The AVR hardware clears the global interrupt flag in SREG before entering an interrupt vector. Thus, normally interrupts will remain disabled inside the handler until the handler exits, where the RETI instruction (that is emitted by the compiler as part of the normal function epilogue for an interrupt handler) will eventually re-enable further interrupts. For that reason, interrupt handlers normally do not nest. For most interrupt handlers, this is the desired behaviour, for some it is even required in order to prevent infinitely recursive interrupts (like UART interrupts, or level-triggered external interrupts). In rare circumstances though it might be desired to re-enable the global interrupt flag as early as possible in the interrupt handler, in order to not defer any other interrupt more than absolutely needed. This could be done using an **sei()** instruction right at the beginning of the interrupt handler, but this still leaves few instructions inside the compiler-generated function prologue to run with global interrupts disabled. The compiler can be instructed to insert an SEI instruction right at the beginning of an interrupt handler by declaring the handler the following way:

```
ISR(XXX_vect, ISR_NOBLOCK)
{
...
}
```

where `xxx_vect` is the name of a valid interrupt vector for the MCU type in question, as explained below.

Two vectors sharing the same code

In some circumstances, the actions to be taken upon two different interrupts might be completely identical so a single implementation for the ISR would suffice. For example, pin-change interrupts arriving from two different ports could logically signal an event that is independent from the actual port (and thus interrupt vector) where it happened. Sharing interrupt vector code can be accomplished using the **ISR_ALIASOF()** attribute to the ISR macro:

```
ISR(PCINT0_vect)
{
...
// Code to handle the event.
}
ISR(PCINT1_vect, ISR_ALIASOF(PCINT0_vect));
```

Note

There is no body to the aliased ISR.

Note that the **ISR_ALIASOF()** feature requires GCC 4.2 or above (or a patched version of GCC 4.1.x). See the documentation of the **ISR_ALIAS()** macro for an implementation which is less elegant but could be applied to all compiler versions.

Empty interrupt service routines

In rare circumstances, in interrupt vector does not need any code to be implemented at all. The vector must be declared anyway, so when the interrupt triggers it won't execute the BADISR_vect code (which by default restarts the application).

This could for example be the case for interrupts that are solely enabled for the purpose of getting the controller out of sleep_mode().

A handler for such an interrupt vector can be declared using the **EMPTY_INTERRUPT()** macro:

```
EMPTY_INTERRUPT(ADC_vect);
```

Note

There is no body to this macro.

Manually defined ISRs

In some circumstances, the compiler-generated prologue and epilogue of the ISR might not be optimal for the job, and a manually defined ISR could be considered particularly to speedup the interrupt handling.

One solution to this could be to implement the entire ISR as manual assembly code in a separate (assembly) file. See **Combining C and assembly source files** for an example of how to implement it that way.

Another solution is to still implement the ISR in C language but take over the compiler's job of generating the prologue and epilogue. This can be done using the ISR_NAKED attribute to the **ISR()** macro. Note that the compiler does not generate *anything* as prologue or epilogue, so the final **reti()** must be provided by the actual implementation. SREG must be manually saved if the ISR code modifies it, and the compiler-implied assumption of **`zero_reg`** always being 0 could be wrong (e. g. when interrupting right after of a MUL instruction).

```
ISR(TIMER1_OVF_vect, ISR_NAKED)
{
PORTB |= _BV(0); // results in SBI which does not affect SREG
reti();
}
```

Choosing the vector: Interrupt vector names

The interrupt is chosen by supplying one of the symbols in following table.

There are currently two different styles present for naming the vectors. One form uses names starting with `SIG_`, followed by a relatively verbose but arbitrarily chosen name describing the interrupt vector. This has been the only available style in avr-libc up to version 1.2.x.

Starting with avr-libc version 1.4.0, a second style of interrupt vector names has been added, where a short phrase for the vector description is followed by `_vect`. The short phrase matches the vector name as described in the datasheet of the respective device (and in Atmel's XML files), with spaces replaced by an underscore and other non-alphanumeric characters dropped. Using the suffix `_vect` is intented to improve

portability to other C compilers available for the AVR that use a similar naming convention.

The historical naming style might become deprecated in a future release, so it is not recommended for new projects.

Note

The **ISR()** macro cannot really spell-check the argument passed to them. Thus, by misspelling one of the names below in a call to **ISR()**, a function will be created that, while possibly being usable as an interrupt function, is not actually wired into the interrupt vector table. The compiler will generate a warning if it detects a suspiciously looking name of a **ISR()** function (i.e. one that after macro replacement does not start with "__vector_").

Vector name	Old vector name	Description	Applicable for device
ADC_vect	SIG_ADC	ADC Conversion Complete	AT90S2333, AT90S4433, AT90S4434, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega48P, ATmega64, ATmega645,

			ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATtiny13, ATtiny15, ATtiny26, ATtiny43U, ATtiny48, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
ANALOG_COMP_0_vect	SIG_COMPARATOR0	Analog Comparator 0	AT90PWM3, AT90PWM2, AT90PWM1
ANALOG_COMP_1_vect	SIG_COMPARATOR1	Analog Comparator 1	AT90PWM3, AT90PWM2, AT90PWM1
ANALOG_COMP_2_vect	SIG_COMPARATOR2	Analog Comparator 2	AT90PWM3, AT90PWM2, AT90PWM1
			AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega165, ATmega165P, ATmega168P,

ANALOG_COMP_vect	SIG_COMPARATOR	Analog Comparator	ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
ANA_COMP_vect	SIG_COMPARATOR	Analog Comparator	AT90S1200, AT90S2313, AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, ATmega16, ATmega161, ATmega162, ATmega163, ATmega32, ATmega323, ATmega8, ATmega8515,

			ATmega8535, ATtiny11, ATtiny12, ATtiny13, ATtiny15, ATtiny2313, ATtiny26, ATtiny28, ATtiny43U, ATtiny48, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861
CANIT_vect	SIG_CAN_INTERRUPT1	CAN Transfer Complete or Error	AT90CAN128, AT90CAN32, AT90CAN64
EEPROM_READY_vect	SIG_EEPROM_READY, SIG_EE_READY		ATtiny2313
EE_RDY_vect	SIG_EEPROM_READY	EEPROM Ready	AT90S2333, AT90S4433, AT90S4434, AT90S8535, ATmega16, ATmega161, ATmega162, ATmega163, ATmega32, ATmega323, ATmega8, ATmega8515, ATmega8535, ATtiny12, ATtiny13, ATtiny15, ATtiny26, ATtiny43U, ATtiny48, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861
			AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega165,

EE_READY_vect	SIG_EEPROM_READY	EEPROM Ready	ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega406, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
EXT_INT0_vect	SIG_INTERRUPT0	External Interrupt Request 0	ATtiny24, ATtiny44, ATtiny84
			AT90S1200, AT90S2313, AT90S2323, AT90S2333, AT90S2343, AT90S4414, AT90S4433, AT90S4434,

			AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega406, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281,
INT0_vect	SIG_INTERRUPT0	External Interrupt 0	

			ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny11, ATtiny12, ATtiny13, ATtiny15, ATtiny22, ATtiny2313, ATtiny26, ATtiny28, ATtiny43U, ATtiny48, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
			AT90S2313, AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega168P, ATmega32,

INT1_vect	SIG_INTERRUPT1	External Interrupt Request 1	ATmega323, ATmega328P, ATmega32HVB, ATmega406, ATmega48P, ATmega64, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny2313, ATtiny28, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
			AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega32, ATmega323, ATmega32HVB, ATmega406,

INT2_vect	SIG_INTERRUPT2	External Interrupt Request 2	ATmega64, ATmega8515, ATmega8535, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
INT3_vect	SIG_INTERRUPT3	External Interrupt Request 3	AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega32HVB, ATmega406, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
INT4_vect	SIG_INTERRUPT4	External Interrupt	AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega64, ATmega640, ATmega1280, ATmega1281,

		Request 4	ATmega2560, ATmega2561, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
INT5_vect	SIG_INTERRUPT5	External Interrupt Request 5	AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
INT6_vect	SIG_INTERRUPT6	External Interrupt Request 6	AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
		External	AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega64, ATmega640, ATmega1280,

INT7_vect	SIG_INTERRUPT7	Interrupt Request 7	ATmega1281, ATmega2560, ATmega2561, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
IO_PINS_vect	SIG_PIN, SIG_PIN_CHANGE	External Interrupt Request 0	ATtiny11, ATtiny12, ATtiny15, ATtiny26
LCD_vect	SIG_LCD	LCD Start of Frame	ATmega169, ATmega169P, ATmega329, ATmega3290, ATmega3290P, ATmega649, ATmega6490
LOWLEVEL_IO_PINS_vect	SIG_PIN	Low-level Input on Port B	ATtiny28
OVRIT_vect	SIG_CAN_OVERFLOW1	CAN Timer Overrun	AT90CAN128, AT90CAN32, AT90CAN64
PCINT0_vect	SIG_PIN_CHANGE0	Pin Change Interrupt Request 0	ATmega162, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega406, ATmega48P, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640,

			ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATtiny13, ATtiny43U, ATtiny48, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
PCINT1_vect	SIG_PIN_CHANGE1	Pin Change Interrupt Request 1	ATmega162, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega406, ATmega48P, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P,

			ATmega164P, ATmega644P, ATmega644, ATtiny43U, ATtiny48, ATtiny24, ATtiny44, ATtiny84, AT90USB162, AT90USB82
PCINT2_vect	SIG_PIN_CHANGE2	Pin Change Interrupt Request 2	ATmega3250, ATmega3250P, ATmega328P, ATmega3290, ATmega3290P, ATmega48P, ATmega6450, ATmega6490, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATtiny48
PCINT3_vect	SIG_PIN_CHANGE3	Pin Change Interrupt Request 3	ATmega3250, ATmega3250P, ATmega3290, ATmega3290P, ATmega6450, ATmega6490, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATtiny48
PCINT_vect	SIG_PIN_CHANGE, SIG_PCINT		ATtiny2313, ATtiny261, ATtiny461, ATtiny861
PSC0_CAPT_vect	SIG_PSC0_CAPTURE	PSC0 Capture Event	AT90PWM3, AT90PWM2, AT90PWM1
			AT90PWM3,

PSC0_EC_vect	SIG_PSC0_END_CYCLE	PSC0 End Cycle	AT90PWM2, AT90PWM1
PSC1_CAPT_vect	SIG_PSC1_CAPTURE	PSC1 Capture Event	AT90PWM3, AT90PWM2, AT90PWM1
PSC1_EC_vect	SIG_PSC1_END_CYCLE	PSC1 End Cycle	AT90PWM3, AT90PWM2, AT90PWM1
PSC2_CAPT_vect	SIG_PSC2_CAPTURE	PSC2 Capture Event	AT90PWM3, AT90PWM2, AT90PWM1
PSC2_EC_vect	SIG_PSC2_END_CYCLE	PSC2 End Cycle	AT90PWM3, AT90PWM2, AT90PWM1
SPI_STC_vect	SIG_SPI	Serial Transfer Complete	AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329,

			ATmega3290, ATmega3290P, ATmega32HVB, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny48, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
SPM_RDY_vect	SIG_SPM_READY	Store Program Memory Ready	ATmega16, ATmega162, ATmega32, ATmega323, ATmega8, ATmega8515, ATmega8535
			AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega165, ATmega165P,

SPM_READY_vect	SIG_SPM_READY	Store Program Memory Read	ATmega168P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega406, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIM0_COMPA_vect	SIG_OUTPUT_COMPARE0A	Timer/Counter Compare Match A	ATtiny13, ATtiny43U, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85
TIM0_COMPB_vect	SIG_OUTPUT_COMPARE0B	Timer/Counter Compare Match B	ATtiny13, ATtiny43U, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85
TIM0_OVF_vect	SIG_OVERFLOW0	Timer/Counter0 Overflow	ATtiny13, ATtiny43U, ATtiny24, ATtiny44, ATtiny84, ATtiny45,

			ATtiny25, ATtiny85
TIM1_CAPT_vect	SIG_INPUT_CAPTURE1	Timer/Counter1 Capture Event	ATtiny24, ATtiny44, ATtiny84
TIM1_COMPA_vect	SIG_OUTPUT_COMPARE1A	Timer/Counter1 Compare Match A	ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85
TIM1_COMPB_vect	SIG_OUTPUT_COMPARE1B	Timer/Counter1 Compare Match B	ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85
TIM1_OVF_vect	SIG_OVERFLOW1	Timer/Counter1 Overflow	ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85
TIMER0_CAPT_vect	SIG_INPUT_CAPTURE0	ADC Conversion Complete	ATtiny261, ATtiny461, ATtiny861
TIMER0_COMPA_vect	SIG_OUTPUT_COMPARE0A	TimerCounter0 Compare Match A	ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny2313, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
			AT90PWM3, AT90PWM2, AT90PWM1, ATmega1284P, ATmega168P, ATmega328P, ATmega32HVB, ATmega48P, ATmega88P, ATmega168,

TIMER0_COMPB_vect	SIG_OUTPUT_COMPARE0B, SIG_OUTPUT_COMPARE0_B	Timer Counter 0 Compare Match B	ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny2313, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER0_COMP_A_vect	SIG_OUTPUT_COMPARE0A, SIG_OUTPUT_COMPARE0_A	Timer/Counter0 Compare Match A	AT90PWM3, AT90PWM2, AT90PWM1
TIMER0_COMP_vect	SIG_OUTPUT_COMPARE0	Timer/Counter0 Compare Match	AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega16, ATmega161, ATmega162, ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega329, ATmega3290, ATmega3290P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490,

			ATmega8515, ATmega8535
TIMER0_OVF0_vect	SIG_OVERFLOW0	Timer/Counter0 Overflow	AT90S2313, AT90S2323, AT90S2343, ATtiny22, ATtiny26
TIMER0_OVF_vect	SIG_OVERFLOW0	Timer/Counter0 Overflow	AT90S1200, AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649,

			ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny11, ATtiny12, ATtiny15, ATtiny2313, ATtiny28, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER1_CAPT1_vect	SIG_INPUT_CAPTURE1	Timer/Counter1 Capture Event	AT90S2313
			AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103,

TIMER1_CAPT_vect	SIG_INPUT_CAPTURE1	Timer/Counter Capture Event	ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATtiny2313, ATtiny48, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646

TIMER1_CMPA_vect	SIG_OUTPUT_COMPARE1A	Timer/Counter1 Compare Match 1A	ATtiny26
TIMER1_CMPB_vect	SIG_OUTPUT_COMPARE1B	Timer/Counter1 Compare Match 1B	ATtiny26
TIMER1_COMP1_vect	SIG_OUTPUT_COMPARE1A	Timer/Counter1 Compare Match	AT90S2313
TIMER1_COMPA_vect	SIG_OUTPUT_COMPARE1A	Timer/Counter1 Compare Match A	AT90S4414, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490,

			ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny2313, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
			AT90S4414, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P,

TIMER1_COMPB_vect	SIG_OUTPUT_COMPARE1B	Timer/Counter1 Compare MatchB	ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny2313, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
			AT90CAN128, AT90CAN32, AT90CAN64, ATmega128,

TIMER1_COMPC_vect	SIG_OUTPUT_COMPARE1C	Timer/Counter1 Compare Match C	ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER1_COMPD_vect	SIG_OUTPUT_COMPARE0D	Timer/Counter1 Compare Match D	ATtiny261, ATtiny461, ATtiny861
TIMER1_COMP_vect	SIG_OUTPUT_COMPARE1A	Timer/Counter1 Compare Match	AT90S2333, AT90S4433, ATtiny15
TIMER1_OVF1_vect	SIG_OVERFLOW1	Timer/Counter1 Overflow	AT90S2313, ATtiny26
			AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM3, AT90PWM2, AT90PWM1, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32,

TIMER1_OVF_vect	SIG_OVERFLOW1	Timer/Counter1 Overflow	ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega32HVB, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny15, ATtiny2313, ATtiny48, ATtiny261, ATtiny461, ATtiny861, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
			ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560,

TIMER2_COMPA_vect	SIG_OUTPUT_COMPARE2A	Timer/Counter2 Compare Match A	ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER2_COMPB_vect	SIG_OUTPUT_COMPARE2B	Timer/Counter2 Compare Match A	ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER2_COMP_vect	SIG_OUTPUT_COMPARE2	Timer/Counter2 Compare Match	AT90S4434, AT90S8535, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega329, ATmega3290, ATmega3290P, ATmega64,

			ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8535
TIMER2_OVF_vect	SIG_OVERFLOW2	Timer/Counter2 Overflow	AT90S4434, AT90S8535, AT90CAN128, AT90CAN32, AT90CAN64, ATmega103, ATmega128, ATmega1284P, ATmega16, ATmega161, ATmega162, ATmega163, ATmega165, ATmega165P, ATmega168P, ATmega169, ATmega169P, ATmega32, ATmega323, ATmega325, ATmega3250, ATmega3250P, ATmega328P, ATmega329, ATmega3290, ATmega3290P, ATmega48P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega8, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P,

			ATmega164P, ATmega644P, ATmega644, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER3_CAPT_vect	SIG_INPUT_CAPTURE3	Timer/Counter3 Capture Event	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega162, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER3_COMPA_vect	SIG_OUTPUT_COMPARE3A	Timer/Counter3 Compare Match A	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega162, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER3_COMPB_vect	SIG_OUTPUT_COMPARE3B	Timer/Counter3 Compare Match B	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega162, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560,

			ATmega2561, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER3_COMPC_vect	SIG_OUTPUT_COMPARE3C	Timer/Counter3 Compare Match C	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER3_OVF_vect	SIG_OVERFLOW3	Timer/Counter3 Overflow	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega162, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TIMER4_CAPT_vect	SIG_INPUT_CAPTURE4	Timer/Counter4 Capture Event	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER4_COMPA_vect	SIG_OUTPUT_COMPARE4A	Timer/Counter4 Compare Match A	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER4_COMPB_vect	SIG_OUTPUT_COMPARE4B	Timer/Counter4 Compare Match B	ATmega640, ATmega1280, ATmega1281, ATmega2560,

			ATmega2561
TIMER4_COMPC_vect	SIG_OUTPUT_COMPARE4C	Timer/Counter4 Compare Match C	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER4_OVF_vect	SIG_OVERFLOW4	Timer/Counter4 Overflow	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER5_CAPT_vect	SIG_INPUT_CAPTURE5	Timer/Counter5 Capture Event	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER5_COMPA_vect	SIG_OUTPUT_COMPARE5A	Timer/Counter5 Compare Match A	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER5_COMPB_vect	SIG_OUTPUT_COMPARE5B	Timer/Counter5 Compare Match B	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER5_COMPC_vect	SIG_OUTPUT_COMPARE5C	Timer/Counter5 Compare Match C	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
TIMER5_OVF_vect	SIG_OVERFLOW5	Timer/Counter5 Overflow	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
			AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega16, ATmega163, ATmega168P, ATmega32, ATmega323, ATmega328P, ATmega32HVB,

TWI_vect	SIG_2WIRE_SERIAL	2-wire Serial Interface	ATmega406, ATmega48P, ATmega64, ATmega8, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATtiny48, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
TXDONE_vect	SIG_TXDONE	Transmission Done, Bit Timer Flag 2 Interrupt	AT86RF401
TXEMPTY_vect	SIG_TXBE	Transmit Buffer Empty, Bit Itmer Flag 0 Interrupt	AT86RF401
UART0_RX_vect	SIG_UART0_RECV	UART0, Rx Complete	ATmega161
UART0_TX_vect	SIG_UART0_TRANS	UART0, Tx Complete	ATmega161
UART0_UDRE_vect	SIG_UART0_DATA	UART0 Data Register Empty	ATmega161
UART1_RX_vect	SIG_UART1_RECV	UART1, Rx Complete	ATmega161
UART1_TX_vect	SIG_UART1_TRANS	UART1, Tx Complete	ATmega161
UART1_UDRE_vect	SIG_UART1_DATA	UART1 Data Register Empty	ATmega161
UART_RX_vect	SIG_UART_RECV	UART, Rx Complete	AT90S2313, AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535,

			ATmega103, ATmega163, ATmega8515
UART_TX_vect	SIG_UART_TRANS	UART, Tx Complete	AT90S2313, AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, ATmega103, ATmega163, ATmega8515
UART_UDRE_vect	SIG_UART_DATA	UART Data Register Empty	AT90S2313, AT90S2333, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535, ATmega103, ATmega163, ATmega8515
USART0_RXC_vect	SIG_USART0_RECV	USART0, Rx Complete	ATmega162
USART0_RX_vect	SIG_UART0_RECV	USART0, Rx Complete	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega325, ATmega329, ATmega64, ATmega645, ATmega649, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644
		USART0, Tx	

USART0_TXC_vect	SIG_USART0_TRANS	Complete	ATmega162
USART0_TX_vect	SIG_UART0_TRANS	USART0, Tx Complete	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega329, ATmega3290, ATmega3290P, ATmega64, ATmega645, ATmega6450, ATmega649, ATmega6490, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644
USART0_UDRE_vect	SIG_UART0_DATA	USART0 Data Register Empty	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega162, ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega325, ATmega329, ATmega64, ATmega645, ATmega649, ATmega640, ATmega1280, ATmega1281, ATmega2560,

			ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644
USART1_RXC_vect	SIG_USART1_RECV	USART1, Rx Complete	ATmega162
USART1_RX_vect	SIG_UART1_RECV	USART1, Rx Complete	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
USART1_TXC_vect	SIG_USART1_TRANS	USART1, Tx Complete	ATmega162
USART1_TX_vect	SIG_UART1_TRANS	USART1, Tx Complete	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286,

			AT90USB647, AT90USB646
USART1_UDRE_vect	SIG_UART1_DATA	USART1, Data Register Empty	AT90CAN128, AT90CAN32, AT90CAN64, ATmega128, ATmega1284P, ATmega162, ATmega64, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, AT90USB162, AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646
USART2_RX_vect	SIG_USART2_RECV	USART2, Rx Complete	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
USART2_TX_vect	SIG_USART2_TRANS	USART2, Tx Complete	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
USART2_UDRE_vect	SIG_USART2_DATA	USART2 Data register Empty	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
USART3_RX_vect	SIG_USART3_RECV	USART3, Rx Complete	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
USART3_TX_vect	SIG_USART3_TRANS	USART3, Tx Complete	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561

USART3_UDRE_vect	SIG_USART3_DATA	USART3 Data register Empty	ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561
USART_RXC_vect	SIG_USART_RECV, SIG_UART_RECV	USART, Rx Complete	ATmega16, ATmega32, ATmega323, ATmega8
USART_RX_vect	SIG_USART_RECV, SIG_UART_RECV	USART, Rx Complete	AT90PWM3, AT90PWM2, AT90PWM1, ATmega168P, ATmega3250, ATmega3250P, ATmega328P, ATmega3290, ATmega3290P, ATmega48P, ATmega6450, ATmega6490, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATtiny2313
USART_TXC_vect	SIG_USART_TRANS, SIG_UART_TRANS	USART, Tx Complete	ATmega16, ATmega32, ATmega323, ATmega8
USART_TX_vect	SIG_USART_TRANS, SIG_UART_TRANS	USART, Tx Complete	AT90PWM3, AT90PWM2, AT90PWM1, ATmega168P, ATmega328P, ATmega48P, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATtiny2313
			AT90PWM3, AT90PWM2, AT90PWM1, ATmega16, ATmega168P,

USART_UDRE_vect	SIG_USART_DATA, SIG_UART_DATA	USART Data Register Empty	ATmega32, ATmega323, ATmega3250, ATmega3250P, ATmega328P, ATmega3290, ATmega3290P, ATmega48P, ATmega6450, ATmega6490, ATmega8, ATmega8535, ATmega88P, ATmega168, ATmega48, ATmega88, ATtiny2313
USI_OVERFLOW_vect	SIG_USI_OVERFLOW	USI Overflow	ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega329, ATmega3290, ATmega3290P, ATmega645, ATmega6450, ATmega649, ATmega6490, ATtiny2313
USI_OVF_vect	SIG_USI_OVERFLOW	USI Overflow	ATtiny26, ATtiny43U, ATtiny24, ATtiny44, ATtiny84, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861
		USI Start	ATmega165, ATmega165P, ATmega169, ATmega169P, ATmega325, ATmega3250, ATmega3250P, ATmega329, ATmega3290, ATmega3290P,

USI_START_vect	SIG_USI_START	Condition	ATmega645, ATmega6450, ATmega649, ATmega6490, ATtiny2313, ATtiny43U, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861
USI_STRT_vect	SIG_USI_START	USI Start	ATtiny26
USI_STR_vect	SIG_USI_START	USI START	ATtiny24, ATtiny44, ATtiny84
WATCHDOG_vect	SIG_WATCHDOG_TIMEOUT	Watchdog Time-out	ATtiny24, ATtiny44, ATtiny84
WDT_OVERFLOW_vect	SIG_WATCHDOG_TIMEOUT, SIG_WDT_OVERFLOW	Watchdog Timer Overflow	ATtiny2313
WDT_vect	SIG_WDT, SIG_WATCHDOG_TIMEOUT	Watchdog Timeout Interrupt	AT90PWM3, AT90PWM2, AT90PWM1, ATmega1284P, ATmega168P, ATmega328P, ATmega32HVB, ATmega406, ATmega48P, ATmega88P, ATmega168, ATmega48, ATmega88, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, ATmega324P, ATmega164P, ATmega644P, ATmega644, ATmega16HVA, ATtiny13, ATtiny43U, ATtiny48, ATtiny45, ATtiny25, ATtiny85, ATtiny261, ATtiny461, ATtiny861, AT90USB162,

			AT90USB82, AT90USB1287, AT90USB1286, AT90USB647, AT90USB646

Macro Definition Documentation

#define BADISR_vect

```
#include <avr/interrupt.h>
```

This is a vector which is aliased to __vector_default, the vector executed when an ISR fires with no accompanying ISR handler. This may be used along with the **ISR()** macro to create a catch-all for undefined but used ISRs for debugging purposes.

#define cli ( )

Disables all interrupts by clearing the global interrupt mask. This function actually compiles into a single line of assembly, so there is no function call overhead. However, the macro also implies a *memory barrier* which can cause additional loss of optimization.

In order to implement atomic access to multi-byte objects, consider using the macros from <**util/atomic.h**>, rather than implementing them manually with **cli()** and **sei()**.

#define EMPTY_INTERRUPT (vector)

Defines an empty interrupt handler function. This will not generate any prolog or epilog code and will only return from the ISR. Do not define a function body as this will define it for you. Example:

```
EMPTY_INTERRUPT(ADC_vect);
```

```
#define ISR ( vector,
              attributes
            )
```

Introduces an interrupt handler function (interrupt service routine) that runs with global interrupts initially disabled by default with no attributes specified.

The attributes are optional and alter the behaviour and resultant generated code of the interrupt routine. Multiple attributes may be used for a single function, with a space seperating each attribute.

Valid attributes are ISR_BLOCK, ISR_NOBLOCK, ISR_NAKED and **ISR_ALIASOF(vect)**.

`vector` must be one of the interrupt vector names that are valid for the particular MCU type.

#define ISR_ALIAS (vector,

```
target_vector
)
```

Aliases a given vector to another one in the same manner as the ISR_ALIASOF attribute for the **ISR()** macro. Unlike the ISR_ALIASOF attribute macro however, this is compatible for all versions of GCC rather than just GCC version 4.2 onwards.

Note

This macro creates a trampoline function for the aliased macro. This will result in a two cycle penalty for the aliased vector compared to the ISR the vector is aliased to, due to the JMP/RJMP opcode used.

Deprecated:

For new code, the use of ISR(..., ISR_ALIASOF(...)) is recommended.

Example:

```
ISR(INT0_vect)
{
PORTB = 42;
}
ISR_ALIAS(INT1_vect, INT0_vect);
```

#define ISR_ALIASOF (**target_vector**)

The ISR is linked to another ISR, specified by the vect parameter. This is compatible with GCC 4.2 and greater only.

Use this attribute in the attributes parameter of the ISR macro.

#define ISR_BLOCK

Identical to an ISR with no attributes specified. Global interrupts are initially disabled by the AVR hardware when entering the ISR, without the compiler modifying this state.

Use this attribute in the attributes parameter of the ISR macro.

#define ISR_NAKED

ISR is created with no prologue or epilogue code. The user code is responsible for preservation of the machine state including the SREG register, as well as placing a **reti()** at the end of the interrupt routine.

Use this attribute in the attributes parameter of the ISR macro.

#define ISR_NOBLOCK

ISR runs with global interrupts initially enabled. The interrupt enable flag is activated by the compiler as early as possible within the ISR to ensure minimal processing delay for nested interrupts.

This may be used to create nested ISRs, however care should be taken to avoid stack overflows, or to avoid infinitely entering the ISR for those cases where the AVR hardware does not clear the respective interrupt flag before entering the ISR.

Use this attribute in the attributes parameter of the ISR macro.

#define reti ( )

Returns from an interrupt routine, enabling global interrupts. This should be the last command executed before leaving an ISR defined with the ISR_NAKED attribute.

This macro actually compiles into a single line of assembly, so there is no function call overhead.

#define sei ( )

Enables interrupts by setting the global interrupt mask. This function actually compiles into a single line of assembly, so there is no function call overhead. However, the macro also implies a *memory barrier* which can cause additional loss of optimization.

In order to implement atomic access to multi-byte objects, consider using the macros from <**util/atomic.h**>, rather than implementing them manually with **cli()** and **sei()**.

#define SIGNAL (**vector**)

Introduces an interrupt handler function that runs with global interrupts initially disabled.

This is the same as the ISR macro without optional attributes.

Deprecated:

Do not use **SIGNAL()** in new code. Use **ISR()** instead.
