Using Interrupts

Previous Top Next

The access to the AVR interrupt system is implemented with the **interrupt** keyword.
Example:

```
/* Vector numbers are for the AT90S8515 */

/* Called automatically on external interrupt */
interrupt [2] void external_int0(void) {
/* Place your code here */

}

/* Called automatically on TIMER0 overflow */
interrupt [8] void timer0_overflow(void) {
/* Place your code here */

}
```

Interrupt vector numbers start with 1.
The compiler will automatically save the affected registers when calling the interrupt functions and restore them back on exit.
A RETI assembly instruction is placed at the end of the interrupt function.
Interrupt functions can’t return a value nor have parameters.
You must also set the corresponding bits in the peripheral control registers to configure the interrupt system and enable the interrupts.

Another possibility to declare an interrupt service routine is by using the #pragma vector preprocessor directive and the **__interrupt** keyword.
#pragma vector is used for specifying that the next declared function is an interrupt service routine. Example:

```
/* Vector numbers are for the AT90S8515 */

/* Specify the vector number using the #pragma vector directive */
#pragma vector=2

/* Called automatically on external interrupt */
__interrupt void external_int0(void) {
/* Place your code here */

}

/* Specify the vector number using the #pragma vector directive */
#pragma vector=8

/* Called automatically on TIMER0 overflow */
__interrupt void timer0_overflow(void) {
/* Place your code here */

}
```

The **#pragma vector** preprocessor directive and the **__interrupt** keyword are used for compatibility with other C compilers for the Atmel AVR.

The automatic saving and restoring of registers affected by the interrupt handler, can be turned on or off using the #pragma savereg preprocessor directive.
Example:

```
/* Turn registers saving off */
#pragma savereg-

/* interrupt handler */
interrupt [1] void my_irq(void) {
/* now save only the registers that are affected by the routines in the
```

```
   interrupt handler, for example R30, R31 and SREG */
#asm
    push r30
    push r31
    in   r30,SREG
    push r30
#endasm

/* place the C code here */
/* .... */
/* now restore SREG, R31 and R30 */
#asm
    pop r30
    out SREG,r30
    pop r31
    pop r30
#endasm
}
/* re-enable register saving for the other interrupts */
#pragma savereg+
```

The default state is automatic saving of registers during interrupts.
The **#pragma savereg** directive is maintained only for compatibility with versions of the compiler prior to V1.24.1. This directive is not recommended for new projects.