Interrupt Functions

The 8051 and its derivatives provide a number of hardware interrupts that may be used for counting, timing, detecting external events, and sending and receiving data using the serial interface. The standard interrupts found on an 8051/8052 are listed in the following table:

Interrupt Number	Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h
5	TIMER/COUNTER 2 (8052)	002Bh

As 8051 vendors create new parts, more interrupts are added. The Cx51 Compiler supports interrupt functions for 32 interrupts (0-31). Use the interrupt vector address in the following table to determine the interrupt number.

Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

The **interrupt** function attribute specifies that the associated function is an interrupt service routine. For example:

```
unsigned int  interruptcnt;
unsigned char second;

void timer0 (void) interrupt 1 using 2  {
  if (++interruptcnt == 4000)  {    /* count to 4000 */
    second++;                       /* second counter    */
    interruptcnt = 0;               /* clear int counter */
  }
}
```

The **interrupt** attribute takes as an argument an integer constant in the 0 to 31 value range. Expressions with operators and the **interrupt** attribute are not allowed in function prototypes. The **interrupt** attribute affects the object code of the function as follows:

- When required, the contents of **ACC**, **B**, **DPH**, **DPL**, and **PSW** are saved on the stack at function invocation time.
- All working registers used in the interrupt function are stored on the stack if a register bank is not specified with the **using** attribute.
- The working registers and special registers that were saved on the stack are restored before exiting the function.
- The function is terminated by the 8051 **RETI** instruction.

In addition, the Cx51 Compiler generates the interrupt vector automatically.

The following sample program demonstrates how to use the **interrupt** attribute. The program also shows you what the code generated to enter and exit the interrupt function looks like. The **using** function attribute is used to select a register bank different from that of the non-interrupt program code. However, because no working registers are needed in this function, the code generated to switch the register bank is eliminated by the optimizer.

```
stmt level  source

   1        extern bit alarm;
   2        int alarm_count;
   3
   4
   5        void falarm (void) interrupt 1 using 3  {
   6   1      alarm_count *= 2;
   7   1      alarm = 1;
   8   1      }


ASSEMBLY LISTING OF GENERATED OBJECT CODE

      ; FUNCTION falarm (BEGIN)
0000 C0E0       PUSH  ACC
0002 C0D0       PUSH  PSW
                      ; SOURCE LINE # 5
                      ; SOURCE LINE # 6
0004 E500   R   MOV   A,alarm_count+01H
0006 25E0       ADD   A,ACC
0008 F500   R   MOV   alarm_count+01H,A
000A E500   R   MOV   A,alarm_count
000C 33         RLC   A
000D F500   R   MOV   alarm_count,A
                      ; SOURCE LINE # 7
000F D200   E   SETB   alarm
                      ; SOURCE LINE # 8
0011 D0D0       POP   PSW
0013 D0E0       POP   ACC
0015 32         RETI
      ; FUNCTION falarm (END)
```

In the example above, note that the **ACC** and **PSW** registers are saved at offset 0000h and restored at offset 0011h. Note also the **RETI** instruction generated to exit the interrupt.

The following rules apply to interrupt functions.

- No function arguments may be specified for an interrupt function. The compiler emits an error message

if an interrupt function is declared with any arguments.

- Interrupt function declarations may not include a return value. They must be declared as void (see the above examples). The compiler emits an error message if any attempt is made to define a return value for the interrupt function. The implicit **int** return value, however, is ignored by the compiler.
- The compiler recognizes direct calls to interrupt functions and rejects them. It is pointless to call interrupt procedures directly, because exiting the procedure causes execution of the **RETI** instruction which affects the hardware interrupt system of the 8051 chip. Because no interrupt request on the part of the hardware existed, the effect of this instruction is indeterminate and usually fatal. Do not call an interrupt function indirectly through a function pointer.
- The compiler generates an interrupt vector for each interrupt function. The code generated for the vector is a jump to the beginning of the interrupt function. Generation of interrupt vectors can be suppressed by including the **NOINTVECTOR** control directive in the Cx51 command line. In this case, you must provide interrupt vectors from separate assembly modules. Refer to the **INTVECTOR** and **INTERVAL** control directives for more information about the interrupt vector table.
- The Cx51 Compiler allows **interrupt** numbers within the 0-31 range. Refer to your 8051 derivative document to determine which interrupts are available.
- Functions called from an interrupt procedure must function with the same register bank as the interrupt procedure. When the **NOAREGS** directive is not explicitly specified, the compiler may generate absolute register accesses using the register bank selected by the **using** attribute or by the **REGISTERBANK** control for that function. Unpredictable results may occur when a function assumes a register bank other than the one currently selected. Refer to Register Bank Access for more information.