Compiling Functions for Interrupt Calls

The GNU C Compiler allows us to compile a function to be called from hardware interrupts. This paper will provide a sample code in compiling function for interrupt call.

To mark a function as callable from an interrupt, include the line '`#pragma interrupt`' somewhere before the beginning of the function's definition. For readability, we might place the pragma statement immediately before the definition of the appropriate function. The pragma statement will affect only the next function defined; if we want to define more than one function with this property, include '`#pragma interrupt`' before each of them. When we define a function with '`#pragma interrupt`', the GNU Compiler alters its usual calling convention to provide the right environment when the function is called from an interrupt.

The following is a sample code on how to use the pragma statement:

In the file Test.c:

```
#pragma interrupt

int had_interrupt;

void iroutine()
{
  service_routine();
  had_interrupt = 1;
}
```

The compiler generated Assembly listing:

```
;       GCC For the Hitachi H8/300
;       By Hitachi America Ltd and Cygnus Support
;       release 2.1
; -O5


        .file   "Test.c"
        .global _service_routine
        .section .text
        .align 2
        .global _iroutine
_iroutine:
        push    r6
        push    r5
        push    r4
        push    r3
        push    r2
        push    r1
        push    r0
        jsr     @_service_routine
        mov.w   #1,r1
        mov.w   r1,@_had_interrupt
```

```
        pop     r0
        pop     r1
        pop     r2
        pop     r3
        pop     r4
        pop     r5
        pop     r6
        rte
        .comm _had_interrupt,2
        .end
```

In the compiler generated Assembly listing, the compiler uses 'rte' (return from exception) instead of 'rts' (return from subroutine) which indicates the 'iroutine' is an interrupt function.

HITACHI