TMS320 DSP
DESIGNER'S NOTEBOOK
Number 31
TEXAS INSTRUMENTS

Interrupts in C on the TMS320C3x

Contributed by Tim Grady

Design Problem How do I use interrupts from C?

Solution There are several parts to this problem: (1) writing the ISR, (2) initializing the interrupt vector table, and (3) linking the parts together in the linker command file.

A C Language ISR

The C compiler requires that each ISR be named as follows:

```
void  c_int0n(void)    /* n is the int number */
{
        /* a C function that is an ISR */
}
```

The interrupt may not return a value and has no arguments. The C compiler recognizes this naming convention and treats it as a normal ISR, which means it performs a context save where needed and returns from the routine via a RETI instruction.

A good practice is to include the interrupts in a separate file called ints.c or something similar. This makes for a more modular style, simpler maintenance, and easier to understand software.

The Interrupt Vector Table

The first 40h addresses are reserved for the interrupt and trap vectors. Address 0 (zero) holds the address of the reset routine. If using C linker options, the RTS30.lib function 'boot.asm' takes care of defining the reset function, but the vector table initialization is left to the user. You can do so with either C or assembly language.

An assembly language routine might look like this.

```
; file name is vectors.asm
;
;     .sect "vectors"    ; a new section begins here
      .word _c_int00     ; the address of the reset vector
      .word _c_int01     ; the ISR for interrupt 0
      .word _c_int02     ; the ISR for interrupt 1
;     etc.
;     end
```

Texas Instruments provides customer support in varied technical areas. Since TI does not possess full access to data concerning all of the uses and applications of customers' products, TI assumes no responsibility for customer product design or the use or application of customers' products or for any infringements of patents or rights of others which may result from TI assistance. 6/93

This routine creates a new section which is merely a list of addresses where the interrupt routines can be found. It can be written in C by encapsulating each line in an asm statement.

For example:

```
asm("   .sect \"vectors\" ");
A C function that is an ISR.
```

Linking Them Together

The linker command file provides the mechanism for including the vectors.asm object and the ints.c object.

```
/* file name == mylink.cmd */
vectors.obj
ints.obj
```

The MEMORY section needs to identify the location of the int vectors.

```
MEMORY
{
    VECTORS:  origin = 0h, length = 40h
    ...
}
```

The SECTIONS section needs to map the user-defined section called "vectors" to the memory location.

```
SECTIONS
{       vectors : > VECTORS
   ...
}
```

Summary

Writing interrupt routines in C is straight forward as long as you follow the simple rules set out in this note. You must also make sure to generate the interrupt vector table and to provide the linker with all the necessary information to link the ISRs, vector table, and section names into the correct locations.

Clearly there are variations on this theme. Some ISRs can be written in C and some in assembly so long as the naming conventions and vector tables are followed and initialized.