TMS320 DSP
DESIGNER'S NOTEBOOK

Setting Up and Simulating Interrupts on the TMS320C5x

APPLICATION BRIEF: SPRA277

Helga Stevenson
Digital Signal Processing Products
Semiconductor Group

Texas Instruments
November 1996

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Examples

Setting Up and Simulating Interrupts on the TMS320C5x

Abstract

This document contains an example that shows the code required to set up serial port interrupts on a ’C5x DSP. This technique can be applied to any other interrupt. Several code examples are given.

Design Problem

How do I set up an interrupt and then simulate it on a TMS320C5x DSP?

Solution

The following example shows the code required to set up serial port interrupts on a ’C5x DSP, although it can be applied to any other interrupt. The code shown in Figure 1 shows the interrupt service routine vector set-up for data received on the synchronous serial port and the initial vector “RINT” to jump to the start of the main code on reset.

Example 1. ISR Set-up

```
_____________________________________________________-
      .sect    “vectors”
      B        BEGIN    ; reset vector jumps to label
                        ;   ‘BEGIN’
      .space   16*8     ; fills the next 8 words with
            ; 0s to locate RINT at the correct location
            ; in the interrupt vector table
RINT: B RECEIVE,*,AR1
            ; Serial port receive interrupt RINT.
_____________________________________________________
```

The first section of the program code should set up the interrupt service routine. For additional information, please refer to the TMS320C5x User’s Guide (SPRU056).

Figure 2 details the steps for setting up the various registers. First, this enables the serial port receive (bit 4) of the Interrupt Mask (IMR) Register. Second, it then sets up the Serial Port Control (SPC) register by placing it in reset and then taking it out. In this example it also configures the serial port to expect frame syncs. Finally, it enables all unmasked interrupts.

Example 2. Enabling Interrupts

```
_____________________________-_____________________
.text
BEGIN:   LACC  #10h  ; Enable RINT
         SACL  IMR
         LACC  #0h   ; reset serial port
         SACL  SPC
         LACC  #0c8h ; take serial port out of reset
         SACL  SPC
         EINT        ; enable all unmasked interrupts
_____________________________-_____________________
```

The final stage on the code development side is the interrupt service routine, shown in Figure 3, which takes the data from the serial port data receive register and processes it.

Example 3. Interrupt Service Routine

```
RECEIVE:
            LAC      DRR      ; read data from DRR
              :
              :
RETURN      EINT
```

In order to simulate data appearing at the serial port receive pin, the file containing hex values must be set up with one value per line. For example:

```
0x1
0x0a
  :
  :
```

This file must then be connected to the simulator as shown in figure 4 in the ‘siminit.cmd’ file. This file defines the ’C5x memory map for the simulator.

Example 4. Section of siminit.cmd

```
       :
       :       ; simulator memory definitions
       :
ma    0x20,1,0x1,IPORT
               ; define addr 0x20 to be an input port
mc    0x20,1,data.in,read
               ; connects IPORT to file ‘data.in’
```

The final stage is to set up the serial port pseudo-registers. In this case, we will use the Receive Interrupt Period register (RIRP), which is a basic counter that counts down to zero from a user-determined value. When it reaches zero, it simulates a receive at the serial port receive pin, transferring a word from the ‘data.in’ file to the serial port receive shift register (RSR) and starting the ISR. The RIRP is set up as follows:

```
?rirp=x
```

where x is any decimal value. This can either be set up at the end of the ‘siminit.cmd’ file or manually in the simulator. The “RIRP” will automatically reset itself to x after triggering the interrupt.

More information on connecting files to the simulator and serial port pseudo-registers can be found in the TMS320C5x C Source Debugger User's Guide (SPRU055).