Engineer-To-Engineer's Note EE-33

Technical Notes on using Analog Devices' DSP components and development tools
Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

Programming The ADSP-21xx Timer In C

Last modified 01.25.99

The contents of this Tech-note are intended to complement documentation that is already available on the topic of C programming on the ADSP-21xx family DSPs, in the form of EE Notes 14, 31, and 32.

The programmable timer on the ADSP-21xx is a very important and widely used feature of the DSP. The timer can generate periodic interrupts based on multiples of the processor's clock. For detailed information on the timer, please refer Chapter 6 of the *ADSP-2100 Family User's manual*.

The timer can be programmed in either assembly language, or in C. In order to simplify the procedure for C language programmers, one of the standard C header files provided by Analog Devices, *misc.h*, defines C-callable function prototypes *timer_set*, *timer_on*, and *timer_off*, that can be used to setup, start, and stop the timer, respectively.

Interrupt Handling Within C

The procedure of timer interrupt handling happens as follows. The idea can be extended to other interrupts.

The occurrence of the timer interrupt (if the interrupt is not masked) causes the DSP to vector off to address 0x0028 in internal Program Memory, which is the timer location in the interrupt vector table *(listing 3)*. The code at this memory location in turn causes a jump to the timer interrupt dispatcher, `___lib_tmri_ctrl`. The timer interrupt dispatcher (the source for which can be found in *\Program Files\Analog Devices\ VisualDSP\ 21xx\lib\Src\dis_tmr.dsp* and *\Program Files\ Analog Devices\VisualDSP\ 21xx\lib\ Src\ int_ctrl.h*) saves a number of registers and sets some status bits before calling the user function (*myfunction* in the attached example). The registers and contexts are restored at the end, before program flow returns to the main program. It is important to remember that an interrupt call within C incurs an overhead of approximately 50 cycles on entry and 50 cycles on return.

More information on the operation of the timer_set and timer_on functions are provided in the *ADSP-2100 Family C Runtime Library Manual*. The sources for these functions are provided in *\Program Files\Analog Devices\ VisualDSP\21xx\lib\Src\tmr_ctrl.dsp*.

The Example Itself

The example shows the procedure to set up the timer to blink the LED on the ADSP-2181 EZ-KIT LITE at a rate of approximately 0.5 second (assuming a 33.33 MHz CLKOUT). It was built using the latest release 6.1 of the ADSP-21xx software, and downloaded to the EZ-KIT LITE via the ADDS-218x EZ-ICE.

ANALOG DEVICES

```
/* MAIN.C */
/* Example to illustrate the use of C-callable timer functions.
   RC, 1/22/99  */

#include <signal.h>       /* Header file that contains signal handling
                             and function prototypes information */
#include <misc.h>         /* Contains the timer functions, such as
                             timer_set() and timer_on()  */
#include "cdef2181.h"      /* header file to define memory-mapped registers */


void myfunction();         /* prototyping the used functions */


void main(void )
{

timer_set(16384,16384,255);      /* configure the timer */

interrupt(SIGTIMER,myfunction);

timer_on();                     /* enable timer */

while(1) {
   asm("idle;");              /* wait for an interrupt */
   }

}                           /* end of main */


void myfunction()
{
   asm("toggle fl1;");
}
```

listing 1. main.c

```
$ARCH
$ADSP2181
$MMAP0
$0000 3FFF paxINT_PM t
$0000 3FDF dadINT_DM t
$
```

listing 2. arch.ach

```
.MODULE/ABS=0           ADSP2181_Runtime_Header;

.ENTRY            ___lib_prog_term;
.EXTERNAL   ___lib_setup_everything;
.EXTERNAL         main_;

.EXTERNAL   ___lib_int2_ctrl, ___lib_sp0x_ctrl, ___lib_sp0r_ctrl;
.EXTERNAL   ___lib_int1_ctrl, ___lib_int0_ctrl, ___lib_tmri_ctrl;
.EXTERNAL   ___lib_intl1_ctrl, ___lib_intl0_ctrl, ___lib_inte_ctrl,___lib_bdma_ctrl;
.EXTERNAL   ___lib_pwdi_ctrl;

__Reset_vector:         CALL ___lib_setup_everything;
                  CALL main_;             {Begin C program}
___lib_prog_term: JUMP ___lib_prog_term;
                  NOP;
__Interrupt2:           JUMP ___lib_int2_ctrl;NOP;NOP;NOP;
__InterruptL1:          JUMP ___lib_intl1_ctrl;NOP;NOP;NOP;
__InterruptL0:          JUMP ___lib_intl0_ctrl;NOP;NOP;NOP;
__Sport0_trans:         JUMP ___lib_sp0x_ctrl;NOP;NOP;NOP;
__Sport0_recv:          JUMP ___lib_sp0r_ctrl;NOP;NOP;NOP;
__InterruptE:           JUMP ___lib_inte_ctrl;NOP;NOP;NOP;
__BDMA_interrupt:       JUMP ___lib_bdma_ctrl;NOP;NOP;NOP;
__Interrupt1:           JUMP ___lib_int1_ctrl;NOP;NOP;NOP;
__Interrupt0:           JUMP ___lib_int0_ctrl;NOP;NOP;NOP;
__Timer_interrupt:      JUMP ___lib_tmri_ctrl;NOP;NOP;NOP;
__Powerdown_interrupt:  JUMP ___lib_pwdi_ctrl;NOP;NOP;NOP;
.ENDMOD;
```

listing 3. 2181_hdr.dsp

```
g21 main.c -a arch -runhdr 2181_hdr.dsp -g -save-temps -o main
```

listing 4. run.bat
