Text Compare
Produced: 2/20/2019 7:38:20 PM

Mode: All
Left file: Modified (Customer) = Example_2834xCpuTimer.c
Right file: Original (TI) = Example_2834xCpuTimer.c

```
// TI File $Revision: /main/4 $                                                  =  // TI File $Revision: /main/4 $
// Checkin $Date: June 28, 2010   09:18:01 $                                     =  // Checkin $Date: June 28, 2010   09:18:01 $
//###########################################################################    =  //###########################################################################
//                                                                               =  //
// FILE:    Example_2834xCpuTimer.c                                              =  // FILE:    Example_2834xCpuTimer.c
//                                                                               =  //
// TITLE:   DSP2834x Device Getting Started Program.                             =  // TITLE:   DSP2834x Device Getting Started Program.
//                                                                               =  //
// ASSUMPTIONS:                                                                  =  // ASSUMPTIONS:
//                                                                               =  //
//    This program requires the DSP2834x header files.                           =  //    This program requires the DSP2834x header files.
//                                                                               =  //
//    Other then boot mode configuration, no other hardware configuration        =  //    Other then boot mode configuration, no other hardware configuration
//    is required.                                                               =  //    is required.
//                                                                               =  //
//                                                                               =  //
//    As supplied, this project is configured for "boot to SARAM"                =  //    As supplied, this project is configured for "boot to SARAM"
//    operation.  The 2834x Boot Mode table is shown below.                      =  //    operation.  The 2834x Boot Mode table is shown below.
//    For information on configuring the boot mode of an eZdsp,                  =  //    For information on configuring the boot mode of an eZdsp,
//    please refer to the documentation included with the eZdsp,                 =  //    please refer to the documentation included with the eZdsp,
//                                                                               =  //
//       $Boot_Table:                                                            =  //       $Boot_Table:
//                                                                               =  //
//         GPIO87   GPIO86     GPIO85   GPIO84                                   =  //         GPIO87   GPIO86     GPIO85   GPIO84
//          XA15     XA14       XA13     XA12                                    =  //          XA15     XA14       XA13     XA12
//           PU       PU         PU       PU                                     =  //           PU       PU         PU       PU
//         ========================================                              =  //         ========================================
//           1        1          1        1     TI Test Only                     =  //           1        1          1        1     TI Test Only
//           1        1          1        0     SCI-A boot                       =  //           1        1          1        0     SCI-A boot
//           1        1          0        1     SPI-A boot                       =  //           1        1          0        1     SPI-A boot
//           1        1          0        0     I2C-A boot timing 1              =  //           1        1          0        0     I2C-A boot timing 1
//           1        0          1        1     eCAN-A boot timing 1             =  //           1        0          1        1     eCAN-A boot timing 1
//           1        0          1        0     McBSP-A boot                     =  //           1        0          1        0     McBSP-A boot
//           1        0          0        1     Jump to XINTF x16                =  //           1        0          0        1     Jump to XINTF x16
//           1        0          0        0     Jump to XINTF x32                =  //           1        0          0        0     Jump to XINTF x32
//           0        1          1        1     eCAN-A boot timing 2             =  //           0        1          1        1     eCAN-A boot timing 2
//           0        1          1        0     Parallel GPIO I/O boot           =  //           0        1          1        0     Parallel GPIO I/O boot
//           0        1          0        1     Parallel XINTF boot              =  //           0        1          0        1     Parallel XINTF boot
//           0        1          0        0     Jump to SARAM       <-           =  //           0        1          0        0     Jump to SARAM       <-
"boot to SARAM"                                                                   =  "boot to SARAM"
//           0        0          1        1     Branch to check boot mode        =  //           0        0          1        1     Branch to check boot mode
//           0        0          1        0     I2C-A boot timing 2              =  //           0        0          1        0     I2C-A boot timing 2
//           0        0          0        1     Reserved                         =  //           0        0          0        1     Reserved
//           0        0          0        0     TI Test Only                     =  //           0        0          0        0     TI Test Only
//                                               Boot_Table_End$$                =  //                                               Boot_Table_End$$
//                                                                               =  //
// DESCRIPTION:                                                                  =  // DESCRIPTION:
//                                                                               =  //
//    This example configures CPU Timer0, 1, and 2 and increments                =  //    This example configures CPU Timer0, 1, and 2 and increments
//    a counter each time the timers assert an interrupt.                        =  //    a counter each time the timers assert an interrupt.
//                                                                               =  //
//       Watch Variables:                                                        =  //       Watch Variables:
//          CpuTimer0.InterruptCount                                             =  //          CpuTimer0.InterruptCount
//          CpuTimer1.InterruptCount                                             =  //          CpuTimer1.InterruptCount
//          CpuTimer2.InterruptCount                                             =  //          CpuTimer2.InterruptCount
//                                                                               =  //
//###########################################################################    =  //###########################################################################
// $TI Release: 2834x Header Files V1.12 $                                       =  // $TI Release: 2834x Header Files V1.12 $
// $Release Date: March 2011 $                                                   =  // $Release Date: March 2011 $
//###########################################################################    =  //###########################################################################

                                                                                  +-
#include "DSP28x_Project.h"     // Device Headerfile and Examples Include      =  #include "DSP28x_Project.h"     // Device Headerfile and Examples Include
File                                                                              =  File

//#define  MS_INT_ENABLE(key)                  if(key) CpuTimer2.RegsAddr-       +-
>TCR.bit.TIE = 1;
//#define  MS_INT_DISABLE()                    CpuTimer2.RegsAddr-
>TCR.bit.TIE; CpuTimer2.RegsAddr->TCR.bit.TIE = 0; //This variant of
interrupt disable should be called when only the millisecond interrupt must
be blocked
//#define  MS_INT_ENABLE(key)                  if(key) CpuTimer2.RegsAddr-
```

```
>TCR.all = ((CpuTimer2.RegsAddr->TCR.all | 0x4000) & (~0x8000));
//#define  MS_INT_DISABLE()                    CpuTimer2.RegsAddr-
>TCR.bit.TIE; CpuTimer2.RegsAddr->TCR.all = ((CpuTimer2.RegsAddr->TCR.all &
(~0x4000)) & (~0x8000)); //This variant of interrupt disable should be
called when only the millisecond interrupt must be blocked

#define  MS_INT_ENABLE()                     CpuTimer2.RegsAddr->TCR.all =
0x4001;
#define  MS_INT_DISABLE()                    CpuTimer2.RegsAddr->TCR.all =
0x0001;
```

```
// Prototype statements for functions found within this file.
```

```
interrupt void cpu_timer0_isr(void);
```

```
interrupt void cpu_timer1_isr(void);
interrupt void cpu_timer2_isr(void);
```

```
void ConfigMicrosecondTimer(struct CPUTIMER_VARS *Timer, unsigned long
Freq);


unsigned long lastTime;
unsigned long maxDifference = 0;
```

```
void main(void)
{

// Step 1. Initialize System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the DSP2834x_SysCtrl.c file.
   InitSysCtrl();

// Step 2. Initalize GPIO:
// This example function is found in the DSP2834x_Gpio.c file and
// illustrates how to set the GPIO to it's default state.
// InitGpio();  // Skipped for this example


// Step 3. Clear all interrupts and initialize PIE vector table:
// Disable CPU interrupts
   DINT;

// Initialize the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP2834x_PieCtrl.c file.
   InitPieCtrl();

// Disable CPU interrupts and clear all CPU interrupt flags:
   IER = 0x0000;
   IFR = 0x0000;

// Initialize the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP2834x_DefaultIsr.c.
// This function is found in DSP2834x_PieVect.c.
   InitPieVectTable();

// Interrupts that are used in this example are re-mapped to
// ISR functions found within this file.
   EALLOW;  // This is needed to write to EALLOW protected registers
```

```
   PieVectTable.TINT2 = &cpu_timer2_isr;
   EDIS;    // This is needed to disable write to EALLOW protected
registers

// Step 4. Initialize the Device Peripheral. This function can be
//         found in DSP2834x_CpuTimers.c
   InitCpuTimers();   // For this example, only initialize the Cpu Timers
```

```
   // Configure CPU-Timer 2 to interrupt every ms (1kHz)




   ConfigCpuTimer(&CpuTimer2, 300, 1000L);   // XXXMHz CPU Freq, 1 milli
second Period (in uSeconds)
```

```
void main(void)
{

// Step 1. Initialize System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the DSP2834x_SysCtrl.c file.
   InitSysCtrl();

// Step 2. Initalize GPIO:
// This example function is found in the DSP2834x_Gpio.c file and
// illustrates how to set the GPIO to it's default state.
// InitGpio();  // Skipped for this example


// Step 3. Clear all interrupts and initialize PIE vector table:
// Disable CPU interrupts
   DINT;

// Initialize the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP2834x_PieCtrl.c file.
   InitPieCtrl();

// Disable CPU interrupts and clear all CPU interrupt flags:
   IER = 0x0000;
   IFR = 0x0000;

// Initialize the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP2834x_DefaultIsr.c.
// This function is found in DSP2834x_PieVect.c.
   InitPieVectTable();

// Interrupts that are used in this example are re-mapped to
// ISR functions found within this file.
   EALLOW;  // This is needed to write to EALLOW protected registers
```

```
   PieVectTable.TINT0 = &cpu_timer0_isr;
   PieVectTable.XINT13 = &cpu_timer1_isr;
```

```
   PieVectTable.TINT2 = &cpu_timer2_isr;
   EDIS;    // This is needed to disable write to EALLOW protected
registers

// Step 4. Initialize the Device Peripheral. This function can be
//         found in DSP2834x_CpuTimers.c
   InitCpuTimers();   // For this example, only initialize the Cpu Timers
```

```
#if (CPU_FRQ_300MHZ)
// Configure CPU-Timer 0, 1, and 2 to interrupt every second:
// 300MHz CPU Freq, 1 second Period (in uSeconds)

   ConfigCpuTimer(&CpuTimer0, 300, 1000000);
   ConfigCpuTimer(&CpuTimer1, 300, 1000000);
   ConfigCpuTimer(&CpuTimer2, 300, 1000000);
```

		#endif
// Configure CPU-Timer 1 as microsecond timer		#if (CPU_FRQ_250MHZ) // Configure CPU-Timer 0, 1, and 2 to interrupt every second: // 250MHz CPU Freq, 1 second Period (in uSeconds)
ConfigMicrosecondTimer(&CpuTimer1, 300);		ConfigCpuTimer(&CpuTimer0, 250, 1000000); ConfigCpuTimer(&CpuTimer1, 250, 1000000); ConfigCpuTimer(&CpuTimer2, 250, 1000000); #endif
	=	
	-+	#if (CPU_FRQ_200MHZ) // Configure CPU-Timer 0, 1, and 2 to interrupt every second: // 200MHz CPU Freq, 1 second Period (in uSeconds)
	=	
	-+	ConfigCpuTimer(&CpuTimer0, 200, 1000000); ConfigCpuTimer(&CpuTimer1, 200, 1000000); ConfigCpuTimer(&CpuTimer2, 200, 1000000); #endif

// To ensure precise timing, use write-only instructions to write to the entire register. Therefore, if any
// of the configuration bits are changed in ConfigCpuTimer and InitCpuTimers (in DSP2834x_CpuTimers.h), the
// below settings must also be updated.
	=	// To ensure precise timing, use write-only instructions to write to the entire register. Therefore, if any
		// of the configuration bits are changed in ConfigCpuTimer and InitCpuTimers (in DSP2834x_CpuTimers.h), the
		// below settings must also be updated.

	-+	CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0

CpuTimer1Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
CpuTimer2Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
	=	CpuTimer1Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
		CpuTimer2Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0

lastTime = -(ReadCpuTimer1Counter());	+-	
// Step 5. User specific code, enable interrupts:	=	// Step 5. User specific code, enable interrupts:

// Enable CPU int1 which is connected to CPU-Timer 0, CPU int13
// which is connected to CPU-Timer 1, and CPU int 14, which is connected
// to CPU-Timer 2:
		// Enable CPU int1 which is connected to CPU-Timer 0, CPU int13
		// which is connected to CPU-Timer 1, and CPU int 14, which is connected
		// to CPU-Timer 2:

//IER |= M_INT1; //IER |= M_INT13;	<>	IER |= M_INT1; IER |= M_INT13;
IER |= M_INT14;	=	IER |= M_INT14;

// Enable TINT0 in the PIE: Group 1 interrupt 7
PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
		// Enable TINT0 in the PIE: Group 1 interrupt 7
		PieCtrlRegs.PIEIER1.bit.INTx7 = 1;

// Enable global Interrupts and higher priority real-time debug events:
EINT; // Enable Global interrupt INTM
ERTM; // Enable Global realtime interrupt DBGM
		// Enable global Interrupts and higher priority real-time debug events:
		EINT; // Enable Global interrupt INTM
		ERTM; // Enable Global realtime interrupt DBGM

// Step 6. IDLE loop. Just sit and loop forever (optional):		// Step 6. IDLE loop. Just sit and loop forever (optional):

for(;;)
{
MS_INT_DISABLE();
DELAY_US(1);
MS_INT_ENABLE();
DELAY_US(950);
}
}
	<>	for(;;);
		
		}

	=	
interrupt void cpu_timer2_isr(void)	<>	interrupt void cpu_timer0_isr(void)
{	=	{

unsigned long newTime, difference;
EALLOW;
CpuTimer2.InterruptCount++;
// The CPU acknowledges the interrupt.
 	
EDIS;
 	
newTime = -(ReadCpuTimer1Counter());
 	
difference = newTime - lastTime;
 	
if(difference > maxDifference)
maxDifference = difference;
 	
lastTime = newTime;
	<>	EALLOW;
		CpuTimer0.InterruptCount++;
		
		// Acknowledge this interrupt to receive more interrupts from group 1
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
		EDIS;

}	=	}

```
void ConfigMicrosecondTimer(struct CPUTIMER_VARS *Timer, unsigned long Freq)
{


    // Initialize timer period:
    // Counter decrements once every TPR+1 System Clock Cycles
    //Period = Period - 1;
    Timer->CPUFreqInMHz = (float) 1.0*Freq;
    Timer->PeriodInUSec = 0xFFFFFFFF;

    Timer->RegsAddr->PRD.all = 0xFFFFFFFF;
    Timer->RegsAddr->TPR.bit.TDDR = (Freq-1) & 0xFF;
    Timer->RegsAddr->TPRH.bit.TDDRH = ((Freq-1) >> 8) & 0xFF;

    // Initialize timer control register:
    Timer->RegsAddr->TCR.bit.TSS = 1;      // 1 = Stop timer, 0 =
Start/Restart Timer
    Timer->RegsAddr->TCR.bit.TRB = 1;      // 1 = reload timer
    Timer->RegsAddr->TCR.bit.SOFT = 1;
    Timer->RegsAddr->TCR.bit.FREE = 1;     // Timer Free Run
    Timer->RegsAddr->TCR.bit.TIE = 0;      // 0 = Disable/ 1 = Enable Timer
Interrupt


   // Reset interrupt counter:

    Timer->InterruptCount = 0;


}

//===========================================================================
// No more.
//===========================================================================
```

```
interrupt void cpu_timer1_isr(void)
{
   EALLOW;
   CpuTimer1.InterruptCount++;
   // The CPU acknowledges the interrupt.




   EDIS;



}

interrupt void cpu_timer2_isr(void)
{  EALLOW;
   CpuTimer2.InterruptCount++;
   // The CPU acknowledges the interrupt.
   EDIS;
}

//===========================================================================
// No more.
//===========================================================================
```