```
//*****************************************************************************
//
// startup_ccs.c - Startup code for use with TI's Code Composer Studio.
//
// Copyright (c) 2012-2013 Texas Instruments Incorporated.  All rights reserved.
// Software License Agreement
//
// Texas Instruments (TI) is supplying this software for use solely and
// exclusively on TI's microcontroller products. The software is owned by
// TI and/or its suppliers, and is protected under applicable copyright
// laws. You may not combine this software with "viral" open-source
// software in order to form a larger program.
//
// THIS SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS.
// NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT
// NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY
// CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
// DAMAGES, FOR ANY REASON WHATSOEVER.
//
// This is part of revision 1.1 of the EK-TM4C123GXL Firmware Package.
//
//*****************************************************************************

#include <stdint.h>
#include "hw_nvic.h"
#include "hw_types.h"

//*****************************************************************************
//
// Forward declaration of the default fault handlers.
//
//*****************************************************************************
void ResetISR(void);
static void NmiSR(void);
static void FaultISR(void);
static void IntDefaultHandler(void);

//*****************************************************************************
//
// External declaration for the reset handler that is to be called when the
// processor is started
//
//*****************************************************************************
extern void _c_int00(void);

//*****************************************************************************
//
// Linker variable that marks the top of the stack.
//
//*****************************************************************************
extern uint32_t __STACK_TOP;

//*****************************************************************************
//
// The vector table.  Note that the proper constructs must be placed on this to
```

```
57// ensure that it ends up at physical address 0x0000.0000 or at the start of
58// the program if located at a start address other than 0.
59//
60//*****************************************************************************
61
62extern void Uart0Isr(void);                 // Refer to UART0 handler in keyboard.c
63extern void keyboardIsr(void);              // Refer to TIMER1 handler in keyboard.c
64
65#pragma DATA_SECTION(g_pfnVectors, ".intvecs")
66void (* const g_pfnVectors[])(void) =
67{
68    (void (*)(void))((uint32_t)&__STACK_TOP),
69                                            // The initial stack pointer
70    ResetISR,                               // The reset handler
71    NmiSR,                                  // The NMI handler
72    FaultISR,                               // The hard fault handler
73    IntDefaultHandler,                      // The MPU fault handler
74    IntDefaultHandler,                      // The bus fault handler
75    IntDefaultHandler,                      // The usage fault handler
76    0,                                      // Reserved
77    0,                                      // Reserved
78    0,                                      // Reserved
79    0,                                      // Reserved
80    IntDefaultHandler,                      // SVCall handler
81    IntDefaultHandler,                      // Debug monitor handler
82    0,                                      // Reserved
83    IntDefaultHandler,                      // The PendSV handler
84    IntDefaultHandler,                      // The SysTick handler
85    IntDefaultHandler,                      // GPIO Port A
86    IntDefaultHandler,                      // GPIO Port B
87    IntDefaultHandler,                      // GPIO Port C
88    IntDefaultHandler,                      // GPIO Port D
89    IntDefaultHandler,                      // GPIO Port E
90    Uart0Isr,                               // UART0 Rx and Tx (modified)
91    IntDefaultHandler,                      // UART1 Rx and Tx
92    IntDefaultHandler,                      // SSI0 Rx and Tx
93    IntDefaultHandler,                      // I2C0 Master and Slave
94    IntDefaultHandler,                      // PWM Fault
95    IntDefaultHandler,                      // PWM Generator 0
96    IntDefaultHandler,                      // PWM Generator 1
97    IntDefaultHandler,                      // PWM Generator 2
98    IntDefaultHandler,                      // Quadrature Encoder 0
99    IntDefaultHandler,                      // ADC Sequence 0
100   IntDefaultHandler,                      // ADC Sequence 1
101   IntDefaultHandler,                      // ADC Sequence 2
102   IntDefaultHandler,                      // ADC Sequence 3
103   IntDefaultHandler,                      // Watchdog timer
104   IntDefaultHandler,                      // Timer 0 subtimer A
105   IntDefaultHandler,                      // Timer 0 subtimer B
106   keyboardIsr,                            // Timer 1 subtimer A
107   IntDefaultHandler,                      // Timer 1 subtimer B
108   IntDefaultHandler,                      // Timer 2 subtimer A
109   IntDefaultHandler,                      // Timer 2 subtimer B
110   IntDefaultHandler,                      // Analog Comparator 0
111   IntDefaultHandler,                      // Analog Comparator 1
112   IntDefaultHandler,                      // Analog Comparator 2
```

```
113     IntDefaultHandler,                      // System Control (PLL, OSC, BO)
114     IntDefaultHandler,                      // FLASH Control
115     IntDefaultHandler,                      // GPIO Port F
116     IntDefaultHandler,                      // GPIO Port G
117     IntDefaultHandler,                      // GPIO Port H
118     IntDefaultHandler,                      // UART2 Rx and Tx
119     IntDefaultHandler,                      // SSI1 Rx and Tx
120     IntDefaultHandler,                      // Timer 3 subtimer A
121     IntDefaultHandler,                      // Timer 3 subtimer B
122     IntDefaultHandler,                      // I2C1 Master and Slave
123     IntDefaultHandler,                      // Quadrature Encoder 1
124     IntDefaultHandler,                      // CAN0
125     IntDefaultHandler,                      // CAN1
126     IntDefaultHandler,                      // CAN2
127     0,                                      // Reserved
128     IntDefaultHandler,                      // Hibernate
129     IntDefaultHandler,                      // USB0
130     IntDefaultHandler,                      // PWM Generator 3
131     IntDefaultHandler,                      // uDMA Software Transfer
132     IntDefaultHandler,                      // uDMA Error
133     IntDefaultHandler,                      // ADC1 Sequence 0
134     IntDefaultHandler,                      // ADC1 Sequence 1
135     IntDefaultHandler,                      // ADC1 Sequence 2
136     IntDefaultHandler,                      // ADC1 Sequence 3
137     0,                                      // Reserved
138     0,                                      // Reserved
139     IntDefaultHandler,                      // GPIO Port J
140     IntDefaultHandler,                      // GPIO Port K
141     IntDefaultHandler,                      // GPIO Port L
142     IntDefaultHandler,                      // SSI2 Rx and Tx
143     IntDefaultHandler,                      // SSI3 Rx and Tx
144     IntDefaultHandler,                      // UART3 Rx and Tx
145     IntDefaultHandler,                      // UART4 Rx and Tx
146     IntDefaultHandler,                      // UART5 Rx and Tx
147     IntDefaultHandler,                      // UART6 Rx and Tx
148     IntDefaultHandler,                      // UART7 Rx and Tx
149     0,                                      // Reserved
150     0,                                      // Reserved
151     0,                                      // Reserved
152     0,                                      // Reserved
153     IntDefaultHandler,                      // I2C2 Master and Slave
154     IntDefaultHandler,                      // I2C3 Master and Slave
155     IntDefaultHandler,                      // Timer 4 subtimer A
156     IntDefaultHandler,                      // Timer 4 subtimer B
157     0,                                      // Reserved
158     0,                                      // Reserved
159     0,                                      // Reserved
160     0,                                      // Reserved
161     0,                                      // Reserved
162     0,                                      // Reserved
163     0,                                      // Reserved
164     0,                                      // Reserved
165     0,                                      // Reserved
166     0,                                      // Reserved
167     0,                                      // Reserved
168     0,                                      // Reserved
```

```
169    0,                                      // Reserved
170    0,                                      // Reserved
171    0,                                      // Reserved
172    0,                                      // Reserved
173    0,                                      // Reserved
174    0,                                      // Reserved
175    0,                                      // Reserved
176    0,                                      // Reserved
177    IntDefaultHandler,                      // Timer 5 subtimer A
178    IntDefaultHandler,                      // Timer 5 subtimer B
179    IntDefaultHandler,                      // Wide Timer 0 subtimer A
180    IntDefaultHandler,                      // Wide Timer 0 subtimer B
181    IntDefaultHandler,                      // Wide Timer 1 subtimer A
182    IntDefaultHandler,                      // Wide Timer 1 subtimer B
183    IntDefaultHandler,                      // Wide Timer 2 subtimer A
184    IntDefaultHandler,                      // Wide Timer 2 subtimer B
185    IntDefaultHandler,                      // Wide Timer 3 subtimer A
186    IntDefaultHandler,                      // Wide Timer 3 subtimer B
187    IntDefaultHandler,                      // Wide Timer 4 subtimer A
188    IntDefaultHandler,                      // Wide Timer 4 subtimer B
189    IntDefaultHandler,                      // Wide Timer 5 subtimer A
190    IntDefaultHandler,                      // Wide Timer 5 subtimer B
191    IntDefaultHandler,                      // FPU
192    0,                                      // Reserved
193    0,                                      // Reserved
194    IntDefaultHandler,                      // I2C4 Master and Slave
195    IntDefaultHandler,                      // I2C5 Master and Slave
196    IntDefaultHandler,                      // GPIO Port M
197    IntDefaultHandler,                      // GPIO Port N
198    IntDefaultHandler,                      // Quadrature Encoder 2
199    0,                                      // Reserved
200    0,                                      // Reserved
201    IntDefaultHandler,                      // GPIO Port P (Summary or P0)
202    IntDefaultHandler,                      // GPIO Port P1
203    IntDefaultHandler,                      // GPIO Port P2
204    IntDefaultHandler,                      // GPIO Port P3
205    IntDefaultHandler,                      // GPIO Port P4
206    IntDefaultHandler,                      // GPIO Port P5
207    IntDefaultHandler,                      // GPIO Port P6
208    IntDefaultHandler,                      // GPIO Port P7
209    IntDefaultHandler,                      // GPIO Port Q (Summary or Q0)
210    IntDefaultHandler,                      // GPIO Port Q1
211    IntDefaultHandler,                      // GPIO Port Q2
212    IntDefaultHandler,                      // GPIO Port Q3
213    IntDefaultHandler,                      // GPIO Port Q4
214    IntDefaultHandler,                      // GPIO Port Q5
215    IntDefaultHandler,                      // GPIO Port Q6
216    IntDefaultHandler,                      // GPIO Port Q7
217    IntDefaultHandler,                      // GPIO Port R
218    IntDefaultHandler,                      // GPIO Port S
219    IntDefaultHandler,                      // PWM 1 Generator 0
220    IntDefaultHandler,                      // PWM 1 Generator 1
221    IntDefaultHandler,                      // PWM 1 Generator 2
222    IntDefaultHandler,                      // PWM 1 Generator 3
223    IntDefaultHandler                       // PWM 1 Fault
224};
```

```
225
226//*****************************************************************************
227//
228// This is the code that gets called when the processor first starts execution
229// following a reset event.  Only the absolutely necessary set is performed,
230// after which the application supplied entry() routine is called.  Any fancy
231// actions (such as making decisions based on the reset cause register, and
232// resetting the bits in that register) are left solely in the hands of the
233// application.
234//
235//*****************************************************************************
236void
237ResetISR(void)
238{
239    //
240    // Jump to the CCS C initialization routine.  This will enable the
241    // floating-point unit as well, so that does not need to be done here.
242    //
243    __asm("    .global _c_int00\n"
244          "    b.w     _c_int00");
245}
246
247//*****************************************************************************
248//
249// This is the code that gets called when the processor receives a NMI.  This
250// simply enters an infinite loop, preserving the system state for examination
251// by a debugger.
252//
253//*****************************************************************************
254static void
255NmiSR(void)
256{
257    //
258    // Enter an infinite loop.
259    //
260    while(1)
261    {
262    }
263}
264
265//*****************************************************************************
266//
267// This is the code that gets called when the processor receives a fault
268// interrupt.  This simply enters an infinite loop, preserving the system state
269// for examination by a debugger.
270//
271//*****************************************************************************
272static void
273FaultISR(void)
274{
275    //
276    // Enter an infinite loop.
277    //
278    while(1)
279    {
280    }
```

```
}

//*****************************************************************************
//
// This is the code that gets called when the processor receives an unexpected
// interrupt.  This simply enters an infinite loop, preserving the system state
// for examination by a debugger.
//
//*****************************************************************************
static void
IntDefaultHandler(void)
{
    //
    // Go into an infinite loop.
    //
    while(1)
    {
    }
}

```