```c
/*
 * @brief NXP LPC1769 LPCXpresso board file
 *
 * @note
 * Copyright(C) NXP Semiconductors, 2012
 * All rights reserved.
 *
 * @par
 * Software that is described herein is for illustrative purposes only
 * which provides customers with programming information regarding the
 * LPC products.  This software is supplied "AS IS" without any warranties of
 * any kind, and NXP Semiconductors and its licensor disclaim any and
 * all warranties, express or implied, including all implied warranties of
 * merchantability, fitness for a particular purpose and non-infringement of
 * intellectual property rights.  NXP Semiconductors assumes no responsibility
 * or liability for the use of the software, conveys no license or rights under any
 * patent, copyright, mask work right, or any other intellectual property rights in
 * or to any products. NXP Semiconductors reserves the right to make changes
 * in the software without notification. NXP Semiconductors also makes no
 * representation or warranty that such application will be suitable for the
 * specified use without further testing or modification.
 *
 * @par
 * Permission to use, copy, modify, and distribute this software and its
 * documentation is hereby granted, under NXP Semiconductors' and its
 * licensor's relevant copyrights in the software, without fee, provided that it
 * is used in conjunction with NXP Semiconductors microcontrollers.  This
 * copyright, permission, and disclaimer notice must appear in all copies of
 * this code.
 */

#include "board.h"
#include "string.h"

#include "retarget.h"

/*****************************************************************************
 * Private types/enumerations/variables
 ****************************************************************************/
#define BUTTONS_BUTTON1_GPIO_PORT_NUM           2
#define BUTTONS_BUTTON1_GPIO_BIT_NUM            10
#define JOYSTICK_UP_GPIO_PORT_NUM               2
#define JOYSTICK_UP_GPIO_BIT_NUM                3
#define JOYSTICK_DOWN_GPIO_PORT_NUM             0
#define JOYSTICK_DOWN_GPIO_BIT_NUM              15
#define JOYSTICK_LEFT_GPIO_PORT_NUM             2
#define JOYSTICK_LEFT_GPIO_BIT_NUM              4
#define JOYSTICK_RIGHT_GPIO_PORT_NUM            0
#define JOYSTICK_RIGHT_GPIO_BIT_NUM             16
#define JOYSTICK_PRESS_GPIO_PORT_NUM            0
#define JOYSTICK_PRESS_GPIO_BIT_NUM             17
#define LED_RED_GPIO_PORT_NUM                   0
#define LED_RED_GPIO_BIT_NUM                    22
#define LED_GREEN_GPIO_PORT_NUM                 3
```

```
55 #define LED_GREEN_GPIO_BIT_NUM                  25
56 #define LED_BLUE_GPIO_PORT_NUM                  3
57 #define LED_BLUE_GPIO_BIT_NUM                   26
58
59 /*****************************************************************************
60  * Public types/enumerations/variables
61  ****************************************************************************/
62
63 /* System oscillator rate and RTC oscillator rate */
64 const uint32_t OscRateIn = 12000000;
65 const uint32_t RTCOscRateIn = 32768;
66
67 /*****************************************************************************
68  * Private functions
69  ****************************************************************************/
70
71 /* Initializes board LED(s) */
72 static void Board_LED_Init(void)
73 {
74     /* Pin PIO0_22 is configured as GPIO pin during SystemInit */
75     /* Set the PIO_22 as output */
76     Chip_GPIO_WriteDirBit(LPC_GPIO, LED_RED_GPIO_PORT_NUM, LED_RED_GPIO_BIT_NUM,
77             true);
78     Chip_GPIO_WriteDirBit(LPC_GPIO, LED_GREEN_GPIO_PORT_NUM,
79             LED_GREEN_GPIO_BIT_NUM, true);
80     Chip_GPIO_WriteDirBit(LPC_GPIO, LED_BLUE_GPIO_PORT_NUM,
81             LED_BLUE_GPIO_BIT_NUM, true);
82 }
83
84 /*****************************************************************************
85  * Public functions
86  ****************************************************************************/
87
88 /* Initialize UART pins */
89 void Board_UART_Init(LPC_USART_T *pUART)
90 {
91     /* Pin Muxing has already been done during SystemInit */
92 }
93
94 /* Initialize debug output via UART for board */
95 void Board_Debug_Init(void)
96 {
97 #if defined(DEBUG_ENABLE)
98     Board_UART_Init(DEBUG_UART);
99
100     Chip_UART_Init(DEBUG_UART);
101     Chip_UART_SetBaud(DEBUG_UART, 115200);
102     Chip_UART_ConfigData(DEBUG_UART,
103     UART_LCR_WLEN8 | UART_LCR_SBS_1BIT | UART_LCR_PARITY_DIS);
104
105     /* Enable UART Transmit */
106     Chip_UART_TXEnable(DEBUG_UART);
107 #endif
108 }
```

```
109
110 /* Sends a character on the UART */
111 void Board_UARTPutChar(char ch)
112 {
113 #if defined(DEBUG_ENABLE)
114     while ((Chip_UART_ReadLineStatus(DEBUG_UART) & UART_LSR_THRE) == 0)
115     {
116     }
117     Chip_UART_SendByte(DEBUG_UART, (uint8_t) ch);
118 #endif
119 }
120
121 /* Gets a character from the UART, returns EOF if no character is ready */
122 int Board_UARTGetChar(void)
123 {
124 #if defined(DEBUG_ENABLE)
125     if (Chip_UART_ReadLineStatus(DEBUG_UART) & UART_LSR_RDR)
126     {
127         return (int) Chip_UART_ReadByte(DEBUG_UART);
128     }
129 #endif
130     return EOF;
131 }
132
133 /* Outputs a string on the debug UART */
134 void Board_UARTPutSTR(char *str)
135 {
136 #if defined(DEBUG_ENABLE)
137     while (*str != '\0')
138     {
139         Board_UARTPutChar(*str++);
140     }
141 #endif
142 }
143
144 /* Sets the state of a board LED to on or off */
145 void Board_LED_Set(uint8_t LEDNumber, bool On)
146 {
147     /* There is only one LED */
148     if (LEDNumber == 0) // red LED
149     {
150         Chip_GPIO_WritePortBit(LPC_GPIO, LED_RED_GPIO_PORT_NUM,
151         LED_RED_GPIO_BIT_NUM, On);
152     }
153     else if (LEDNumber == 1) // green LED
154     {
155         Chip_GPIO_WritePortBit(LPC_GPIO, LED_GREEN_GPIO_PORT_NUM,
156         LED_GREEN_GPIO_BIT_NUM, On);
157     }
158     else if (LEDNumber == 2) // blue LED
159     {
160         Chip_GPIO_WritePortBit(LPC_GPIO, LED_BLUE_GPIO_PORT_NUM,
161         LED_BLUE_GPIO_BIT_NUM, On);
162     }
```

```
163
164 }
165
166 /* Returns the current state of a board LED */
167 bool Board_LED_Test(uint8_t LEDNumber)
168 {
169     bool state = false;
170
171     if (LEDNumber == 0)
172     {
173         state = Chip_GPIO_ReadPortBit(LPC_GPIO, LED_RED_GPIO_PORT_NUM,
174         LED_RED_GPIO_BIT_NUM);
175     }
176     else if (LEDNumber == 1)
177     {
178         state = Chip_GPIO_ReadPortBit(LPC_GPIO, LED_GREEN_GPIO_PORT_NUM,
179         LED_GREEN_GPIO_BIT_NUM);
180     }
181     else if (LEDNumber == 2)
182     {
183         state = Chip_GPIO_ReadPortBit(LPC_GPIO, LED_BLUE_GPIO_PORT_NUM,
184         LED_BLUE_GPIO_BIT_NUM);
185     }
186
187     return state;
188 }
189
190 void Board_LED_Toggle(uint8_t LEDNumber)
191 {
192     if (LEDNumber == 0)
193     {
194         Board_LED_Set(LEDNumber, !Board_LED_Test(LEDNumber));
195     }
196 }
197
198 /* Set up and initialize all required blocks and functions related to the
199  board hardware */
200 void Board_Init(void)
201 {
202     /* Sets up DEBUG UART */
203     DEBUGINIT();
204
205     /* Initializes GPIO */
206     Chip_GPIO_Init(LPC_GPIO);
207     Chip_IOCON_Init(LPC_IOCON);
208
209     /* Initialize LEDs */
210     Board_LED_Init();
211 }
212
213 /* Returns the MAC address assigned to this board */
214 void Board_ENET_GetMacADDR(uint8_t *mcaddr)
215 {
216     const uint8_t boardmac[] =
```

```
217     { 0x00, 0x60, 0x37, 0x12, 0x34, 0x56 };
218
219     memcpy(mcaddr, boardmac, 6);
220 }
221
222 /* Initialize pin muxing for SSP interface */
223 void Board_SSP_Init(LPC_SSP_T *pSSP)
224 {
225     if (pSSP == LPC_SSP1)
226     {
227         /* Set up clock and muxing for SSP1 interface */
228         /*
229          * Initialize SSP0 pins connect
230          * P0.7: SCK
231          * P0.6: SSEL
232          * P0.8: MISO
233          * P0.9: MOSI
234          */
235         Chip_IOCON_PinMux(LPC_IOCON, 0, 7, IOCON_MODE_INACT, IOCON_FUNC2);
236         Chip_IOCON_PinMux(LPC_IOCON, 0, 6, IOCON_MODE_INACT, IOCON_FUNC2);
237         Chip_IOCON_PinMux(LPC_IOCON, 0, 8, IOCON_MODE_INACT, IOCON_FUNC2);
238         Chip_IOCON_PinMux(LPC_IOCON, 0, 9, IOCON_MODE_INACT, IOCON_FUNC2);
239     }
240     else
241     {
242         /* Set up clock and muxing for SSP0 interface */
243         /*
244          * Initialize SSP0 pins connect
245          * P0.15: SCK
246          * P0.16: SSEL
247          * P0.17: MISO
248          * P0.18: MOSI
249          */
250         Chip_IOCON_PinMux(LPC_IOCON, 0, 15, IOCON_MODE_INACT, IOCON_FUNC2);
251         Chip_IOCON_PinMux(LPC_IOCON, 0, 16, IOCON_MODE_INACT, IOCON_FUNC2);
252         Chip_IOCON_PinMux(LPC_IOCON, 0, 17, IOCON_MODE_INACT, IOCON_FUNC2);
253         Chip_IOCON_PinMux(LPC_IOCON, 0, 18, IOCON_MODE_INACT, IOCON_FUNC2);
254     }
255 }
256
257 /* Initialize pin muxing for SPI interface */
258 void Board_SPI_Init(bool isMaster)
259 {
260     /* Set up clock and muxing for SSP0 interface */
261     /*
262      * Initialize SSP0 pins connect
263      * P0.15: SCK
264      * P0.16: SSEL
265      * P0.17: MISO
266      * P0.18: MOSI
267      */
268     Chip_IOCON_PinMux(LPC_IOCON, 0, 15, IOCON_MODE_PULLDOWN, IOCON_FUNC3);
269     if (isMaster)
270     {
```

```
271         Chip_IOCON_PinMux(LPC_IOCON, 0, 16, IOCON_MODE_PULLUP, IOCON_FUNC0);
272         Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 16, true);
273         Board_SPI_DeassertSSEL();
274
275     }
276     else
277     {
278         Chip_IOCON_PinMux(LPC_IOCON, 0, 16, IOCON_MODE_PULLUP, IOCON_FUNC3);
279     }
280     Chip_IOCON_PinMux(LPC_IOCON, 0, 17, IOCON_MODE_INACT, IOCON_FUNC3);
281     Chip_IOCON_PinMux(LPC_IOCON, 0, 18, IOCON_MODE_INACT, IOCON_FUNC3);
282 }
283
284 /* Assert SSEL pin */
285 void Board_SPI_AssertSSEL(void)
286 {
287     Chip_GPIO_WritePortBit(LPC_GPIO, 0, 16, false);
288 }
289
290 /* De-Assert SSEL pin */
291 void Board_SPI_DeassertSSEL(void)
292 {
293     Chip_GPIO_WritePortBit(LPC_GPIO, 0, 16, true);
294 }
295
296 void Board_Audio_Init(LPC_I2S_T *pI2S, int micIn)
297 {
298     I2S_AUDIO_FORMAT_T I2S_Config;
299
300     /* Chip_Clock_EnablePeripheralClock(SYSCTL_CLOCK_I2S); */
301
302     I2S_Config.SampleRate = 48000;
303     I2S_Config.ChannelNumber = 2; /* 1 is mono, 2 is stereo */
304     I2S_Config.WordWidth = 16; /* 8, 16 or 32 bits */
305     Chip_I2S_Init(pI2S);
306     Chip_I2S_TxConfig(pI2S, &I2S_Config);
307 }
308
309 /* Sets up board specific I2C interface */
310 void Board_I2C_Init(I2C_ID_T id)
311 {
312     switch (id)
313     {
314     case I2C0:
315         Chip_IOCON_PinMux(LPC_IOCON, 0, 27, IOCON_MODE_INACT, IOCON_FUNC1);
316         Chip_IOCON_PinMux(LPC_IOCON, 0, 28, IOCON_MODE_INACT, IOCON_FUNC1);
317         Chip_IOCON_SetI2CPad(LPC_IOCON, I2CPADCFG_STD_MODE);
318         break;
319
320     case I2C1:
321         Chip_IOCON_PinMux(LPC_IOCON, 0, 19, IOCON_MODE_INACT, IOCON_FUNC2);
322         Chip_IOCON_PinMux(LPC_IOCON, 0, 20, IOCON_MODE_INACT, IOCON_FUNC2);
323         Chip_IOCON_EnableOD(LPC_IOCON, 0, 19);
324         Chip_IOCON_EnableOD(LPC_IOCON, 0, 20);
```

```
325         break;
326
327     case I2C2:
328         Chip_IOCON_PinMux(LPC_IOCON, 0, 10, IOCON_MODE_INACT, IOCON_FUNC2);
329         Chip_IOCON_PinMux(LPC_IOCON, 0, 11, IOCON_MODE_INACT, IOCON_FUNC2);
330         Chip_IOCON_EnableOD(LPC_IOCON, 0, 10);
331         Chip_IOCON_EnableOD(LPC_IOCON, 0, 11);
332         break;
333     }
334 }
335
336 void Board_Buttons_Init(void)
337 {
338     Chip_GPIO_WriteDirBit(LPC_GPIO, BUTTONS_BUTTON1_GPIO_PORT_NUM,
339     BUTTONS_BUTTON1_GPIO_BIT_NUM, false);
340 }
341
342 uint32_t Buttons_GetStatus(void)
343 {
344     uint8_t ret = NO_BUTTON_PRESSED;
345     if (Chip_GPIO_ReadPortBit(LPC_GPIO, BUTTONS_BUTTON1_GPIO_PORT_NUM,
346     BUTTONS_BUTTON1_GPIO_BIT_NUM) == 0x00)
347     {
348         ret |= BUTTONS_BUTTON1;
349     }
350     return ret;
351 }
352
353 /* Baseboard joystick buttons */
354 #define NUM_BUTTONS 5
355 static const uint8_t portButton[NUM_BUTTONS] =
356 {
357 JOYSTICK_UP_GPIO_PORT_NUM,
358 JOYSTICK_DOWN_GPIO_PORT_NUM,
359 JOYSTICK_LEFT_GPIO_PORT_NUM,
360 JOYSTICK_RIGHT_GPIO_PORT_NUM,
361 JOYSTICK_PRESS_GPIO_PORT_NUM };
362 static const uint8_t pinButton[NUM_BUTTONS] =
363 {
364 JOYSTICK_UP_GPIO_BIT_NUM,
365 JOYSTICK_DOWN_GPIO_BIT_NUM,
366 JOYSTICK_LEFT_GPIO_BIT_NUM,
367 JOYSTICK_RIGHT_GPIO_BIT_NUM,
368 JOYSTICK_PRESS_GPIO_BIT_NUM };
369 static const uint8_t stateButton[NUM_BUTTONS] =
370 {
371 JOY_UP,
372 JOY_DOWN,
373 JOY_LEFT,
374 JOY_RIGHT,
375 JOY_PRESS };
376
377 /* Initialize Joystick */
378 void Board_Joystick_Init(void)
```

```
379 {
380     int ix;
381 
382     /* IOCON states already selected in SystemInit(), GPIO setup only. Pullups
383      are external, so IOCON with no states */
384     for (ix = 0; ix < NUM_BUTTONS; ix++)
385     {
386         Chip_GPIO_SetPinDIRInput(LPC_GPIO, portButton[ix], pinButton[ix]);
387     }
388 }
389 
390 /* Get Joystick status */
391 uint8_t Joystick_GetStatus(void)
392 {
393     uint8_t ix, ret = 0;
394 
395     for (ix = 0; ix < NUM_BUTTONS; ix++)
396     {
397         if ((Chip_GPIO_GetPinState(LPC_GPIO, portButton[ix], pinButton[ix]))
398                 == false)
399         {
400             ret |= stateButton[ix];
401         }
402     }
403 
404     return ret;
405 }
406 
407 void Serial_CreateStream(void *Stream)
408 {
409 }
410 
411 void Board_USBD_Init(uint32_t port)
412 {
413     /* VBUS is not connected on the NXP LPCXpresso LPC1769, so leave the pin at default
    setting. */
414     /*Chip_IOCON_PinMux(LPC_IOCON, 1, 30, IOCON_MODE_INACT, IOCON_FUNC2);*//* USB VBUS
    */
415 
416     Chip_IOCON_PinMux(LPC_IOCON, 0, 29, IOCON_MODE_INACT, IOCON_FUNC1); /* P0.29 D1+,
    P0.30 D1- */
417     Chip_IOCON_PinMux(LPC_IOCON, 0, 30, IOCON_MODE_INACT, IOCON_FUNC1);
418 
419     LPC_USB->USBClkCtrl = 0x12; /* Dev, AHB clock enable */
420     while ((LPC_USB->USBClkSt & 0x12) != 0x12)
421         ;
422 }
423 
424 
```