#if defined (\_\_USE\_LPCOPEN)

#if defined(NO\_BOARD\_LIB)

#include "chip.h"

#else

#include "board.h"

#endif

#endif

#include <cr\_section\_macros.h>

void delay(unsigned long int del)

{

unsigned long int i=0;

while((i++)<del);

return;

}

int main(void)

{

#if defined (\_\_USE\_LPCOPEN)

// Read clock settings and update SystemCoreClock variable

SystemCoreClockUpdate();

#if !defined(NO\_BOARD\_LIB)

// Set up and initialize all required blocks and

// functions related to the board hardware

Board\_Init();

Board\_LED\_Set(0, true);

Board\_LED\_Set(1, true);

Board\_LED\_Set(2, true);

while(1) {

Board\_LED\_Set(0,false);

delay(12000000);

Board\_LED\_Toggle(0);

Board\_LED\_Set(1,false);

delay(12000000);

Board\_LED\_Toggle(1);

Board\_LED\_Set(2,true);

delay(12000000);

Board\_LED\_Toggle(2);

}

return 0 ;

}

#endif

#endif

/\*

\* @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 LED0\_GPIO\_PORT\_NUM 0

#define LED0\_GPIO\_BIT\_NUM 22/\* RED LED\*/

#define LED1\_GPIO\_PORT\_NUM 3

#define LED1\_GPIO\_BIT\_NUM 26/\* BLUE LED\*/

#define LED2\_GPIO\_PORT\_NUM 3

#define LED2\_GPIO\_BIT\_NUM 25/\* GREEN LED\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Public types/enumerations/variables

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* System oscillator rate and RTC oscillator rate \*/

const uint32\_t OscRateIn = 12000000;

const uint32\_t RTCOscRateIn = 32768;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Private functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initializes board LED(s) \*/

static void Board\_LED\_Init(void)

{

/\* Pin PIO0\_22 is configured as GPIO pin during SystemInit \*/

/\* Set the PIO\_22 as output \*/

Chip\_GPIO\_WriteDirBit(LPC\_GPIO, LED0\_GPIO\_PORT\_NUM, LED0\_GPIO\_BIT\_NUM,

true);

Chip\_GPIO\_WriteDirBit(LPC\_GPIO, LED1\_GPIO\_PORT\_NUM, LED1\_GPIO\_BIT\_NUM,

true);

Chip\_GPIO\_WriteDirBit(LPC\_GPIO, LED2\_GPIO\_PORT\_NUM, LED2\_GPIO\_BIT\_NUM,

true);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Public functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initialize UART pins \*/

void Board\_UART\_Init(LPC\_USART\_T \*pUART)

{

/\* Pin Muxing has already been done during SystemInit \*/

}

/\* Initialize debug output via UART for board \*/

void Board\_Debug\_Init(void)

{

#if defined(DEBUG\_ENABLE)

Board\_UART\_Init(DEBUG\_UART);

Chip\_UART\_Init(DEBUG\_UART);

Chip\_UART\_SetBaud(DEBUG\_UART, 115200);

Chip\_UART\_ConfigData(DEBUG\_UART, UART\_LCR\_WLEN8 | UART\_LCR\_SBS\_1BIT |

UART\_LCR\_PARITY\_DIS);

/\* Enable UART Transmit \*/

Chip\_UART\_TXEnable(DEBUG\_UART);

#endif

}

/\* Sends a character on the UART \*/

void Board\_UARTPutChar(char ch)

{

#if defined(DEBUG\_ENABLE)

while ((Chip\_UART\_ReadLineStatus(DEBUG\_UART) & UART\_LSR\_THRE) == 0) {}

Chip\_UART\_SendByte(DEBUG\_UART, (uint8\_t) ch);

#endif

}

/\* Gets a character from the UART, returns EOF if no character is ready \*/

int Board\_UARTGetChar(void)

{

#if defined(DEBUG\_ENABLE)

if (Chip\_UART\_ReadLineStatus(DEBUG\_UART) & UART\_LSR\_RDR) {

return (int) Chip\_UART\_ReadByte(DEBUG\_UART);

}

#endif

return EOF;

}

/\* Outputs a string on the debug UART \*/

void Board\_UARTPutSTR(char \*str)

{

#if defined(DEBUG\_ENABLE)

while (\*str != '\0') {

Board\_UARTPutChar(\*str++);

}

#endif

}

/\* Sets the state of a board LED to on or off \*/

void Board\_LED\_Set(uint8\_t LEDNumber, bool On)

{

/\* RED LED \*/

if (LEDNumber == 0) {

Chip\_GPIO\_WritePortBit(LPC\_GPIO, LED0\_GPIO\_PORT\_NUM,

LED0\_GPIO\_BIT\_NUM, On);

}

/\* BLUE LED \*/

if (LEDNumber == 1) {

Chip\_GPIO\_WritePortBit(LPC\_GPIO,

LED1\_GPIO\_PORT\_NUM, LED1\_GPIO\_BIT\_NUM, On);

}

/\* GREEN LED \*/

if (LEDNumber == 2) {

Chip\_GPIO\_WritePortBit(LPC\_GPIO,

LED2\_GPIO\_PORT\_NUM, LED2\_GPIO\_BIT\_NUM, On);

}

}

/\* Returns the current state of a board LED \*/

bool Board\_LED\_Test(uint8\_t LEDNumber)

{

bool state = false;

if (LEDNumber == 0) {

state = Chip\_GPIO\_ReadPortBit(LPC\_GPIO, LED0\_GPIO\_PORT\_NUM,

LED0\_GPIO\_BIT\_NUM);

}

if (LEDNumber == 1) {

state = Chip\_GPIO\_ReadPortBit(LPC\_GPIO,

LED1\_GPIO\_PORT\_NUM, LED1\_GPIO\_BIT\_NUM);

}

if (LEDNumber == 2) {

state = Chip\_GPIO\_ReadPortBit(LPC\_GPIO,

LED2\_GPIO\_PORT\_NUM, LED2\_GPIO\_BIT\_NUM);

}

return state;

}

void Board\_LED\_Toggle(uint8\_t LEDNumber)

{

if (LEDNumber == 0) {

Board\_LED\_Set(LEDNumber, !Board\_LED\_Test(LEDNumber));

}

if (LEDNumber == 1) {

Board\_LED\_Set(LEDNumber,

!Board\_LED\_Test(LEDNumber));

}

if (LEDNumber == 2) {

Board\_LED\_Set(LEDNumber,

!Board\_LED\_Test(LEDNumber));

}

}

/\* Set up and initialize all required blocks and functions related to the

board hardware \*/

void Board\_Init(void)

{

/\* Sets up DEBUG UART \*/

DEBUGINIT();

/\* Initializes GPIO \*/

Chip\_GPIO\_Init(LPC\_GPIO);

Chip\_IOCON\_Init(LPC\_IOCON);

/\* Initialize LEDs \*/

Board\_LED\_Init();

}

/\* Returns the MAC address assigned to this board \*/

void Board\_ENET\_GetMacADDR(uint8\_t \*mcaddr)

{

const uint8\_t boardmac[] = {0x00, 0x60, 0x37, 0x12, 0x34, 0x56};

memcpy(mcaddr, boardmac, 6);

}

/\* Initialize pin muxing for SSP interface \*/

void Board\_SSP\_Init(LPC\_SSP\_T \*pSSP)

{

if (pSSP == LPC\_SSP1) {

/\* Set up clock and muxing for SSP1 interface \*/

/\*

\* Initialize SSP0 pins connect

\* P0.7: SCK

\* P0.6: SSEL

\* P0.8: MISO

\* P0.9: MOSI

\*/

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 7, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 6, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 8, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 9, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

}

else {

/\* Set up clock and muxing for SSP0 interface \*/

/\*

\* Initialize SSP0 pins connect

\* P0.15: SCK

\* P0.16: SSEL

\* P0.17: MISO

\* P0.18: MOSI

\*/

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 15, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 16, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 17, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 18, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

}

}

/\* Initialize pin muxing for SPI interface \*/

void Board\_SPI\_Init(bool isMaster)

{

/\* Set up clock and muxing for SSP0 interface \*/

/\*

\* Initialize SSP0 pins connect

\* P0.15: SCK

\* P0.16: SSEL

\* P0.17: MISO

\* P0.18: MOSI

\*/

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 15, IOCON\_MODE\_PULLDOWN, IOCON\_FUNC3);

if (isMaster) {

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 16, IOCON\_MODE\_PULLUP,

IOCON\_FUNC0);

Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 0, 16, true);

Board\_SPI\_DeassertSSEL();

}

else {

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 16, IOCON\_MODE\_PULLUP,

IOCON\_FUNC3);

}

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 17, IOCON\_MODE\_INACT, IOCON\_FUNC3);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 18, IOCON\_MODE\_INACT, IOCON\_FUNC3);

}

/\* Assert SSEL pin \*/

void Board\_SPI\_AssertSSEL(void)

{

Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 16, false);

}

/\* De-Assert SSEL pin \*/

void Board\_SPI\_DeassertSSEL(void)

{

Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 16, true);

}

void Board\_Audio\_Init(LPC\_I2S\_T \*pI2S, int micIn)

{

I2S\_AUDIO\_FORMAT\_T I2S\_Config;

/\* Chip\_Clock\_EnablePeripheralClock(SYSCTL\_CLOCK\_I2S); \*/

I2S\_Config.SampleRate = 48000;

I2S\_Config.ChannelNumber = 2; /\* 1 is mono, 2 is stereo \*/

I2S\_Config.WordWidth = 16; /\* 8, 16 or 32

bits \*/

Chip\_I2S\_Init(pI2S);

Chip\_I2S\_TxConfig(pI2S, &I2S\_Config);

}

/\* Sets up board specific I2C interface \*/

void Board\_I2C\_Init(I2C\_ID\_T id)

{

switch(id)

{

case I2C0:

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 27, IOCON\_MODE\_INACT,

IOCON\_FUNC1);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 28, IOCON\_MODE\_INACT,

IOCON\_FUNC1);

Chip\_IOCON\_SetI2CPad(LPC\_IOCON, I2CPADCFG\_STD\_MODE);

break;

case I2C1:

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 19, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 20, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_EnableOD(LPC\_IOCON, 0, 19);

Chip\_IOCON\_EnableOD(LPC\_IOCON, 0, 20);

break;

case I2C2:

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 10, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 11, IOCON\_MODE\_INACT,

IOCON\_FUNC2);

Chip\_IOCON\_EnableOD(LPC\_IOCON, 0, 10);

Chip\_IOCON\_EnableOD(LPC\_IOCON, 0, 11);

break;

}

}

void Board\_Buttons\_Init(void)

{

Chip\_GPIO\_WriteDirBit(LPC\_GPIO, BUTTONS\_BUTTON1\_GPIO\_PORT\_NUM,

BUTTONS\_BUTTON1\_GPIO\_BIT\_NUM, false);

}

uint32\_t Buttons\_GetStatus(void)

{

uint8\_t ret = NO\_BUTTON\_PRESSED;

if (Chip\_GPIO\_ReadPortBit(LPC\_GPIO, BUTTONS\_BUTTON1\_GPIO\_PORT\_NUM,

BUTTONS\_BUTTON1\_GPIO\_BIT\_NUM) == 0x00) {

ret |= BUTTONS\_BUTTON1;

}

return ret;

}

/\* Baseboard joystick buttons \*/

#define NUM\_BUTTONS 5

static const uint8\_t portButton[NUM\_BUTTONS] = {

JOYSTICK\_UP\_GPIO\_PORT\_NUM,

JOYSTICK\_DOWN\_GPIO\_PORT\_NUM,

JOYSTICK\_LEFT\_GPIO\_PORT\_NUM,

JOYSTICK\_RIGHT\_GPIO\_PORT\_NUM,

JOYSTICK\_PRESS\_GPIO\_PORT\_NUM

};

static const uint8\_t pinButton[NUM\_BUTTONS] = {

JOYSTICK\_UP\_GPIO\_BIT\_NUM,

JOYSTICK\_DOWN\_GPIO\_BIT\_NUM,

JOYSTICK\_LEFT\_GPIO\_BIT\_NUM,

JOYSTICK\_RIGHT\_GPIO\_BIT\_NUM,

JOYSTICK\_PRESS\_GPIO\_BIT\_NUM

};

static const uint8\_t stateButton[NUM\_BUTTONS] = {

JOY\_UP,

JOY\_DOWN,

JOY\_LEFT,

JOY\_RIGHT,

JOY\_PRESS

};

/\* Initialize Joystick \*/

void Board\_Joystick\_Init(void)

{

int ix;

/\* IOCON states already selected in SystemInit(), GPIO setup only.

Pullups

are external, so IOCON with no states \*/

for (ix = 0; ix < NUM\_BUTTONS; ix++) {

Chip\_GPIO\_SetPinDIRInput(LPC\_GPIO, portButton[ix],

pinButton[ix]);

}

}

/\* Get Joystick status \*/

uint8\_t Joystick\_GetStatus(void)

{

uint8\_t ix, ret = 0;

for (ix = 0; ix < NUM\_BUTTONS; ix++) {

if ((Chip\_GPIO\_GetPinState(LPC\_GPIO, portButton[ix],

pinButton[ix])) == false) {

ret |= stateButton[ix];

}

}

return ret;

}

void Serial\_CreateStream(void \*Stream)

{}

void Board\_USBD\_Init(uint32\_t port)

{

/\* VBUS is not connected on the NXP LPCXpresso LPC1769, so leave the pin

at default setting. \*/

/\*Chip\_IOCON\_PinMux(LPC\_IOCON, 1, 30, IOCON\_MODE\_INACT, IOCON\_FUNC2);\*/

/\* USB VBUS \*/

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 29, IOCON\_MODE\_INACT,

IOCON\_FUNC1); /\* P0.29 D1+, P0.30 D1- \*/

Chip\_IOCON\_PinMux(LPC\_IOCON, 0, 30, IOCON\_MODE\_INACT, IOCON\_FUNC1);

LPC\_USB->USBClkCtrl = 0x12; /\* Dev, AHB clock enable \*/

while ((LPC\_USB->USBClkSt & 0x12) != 0x12);

}