Final Report

Irshad, Awais
Depart Of Electrical, Computer and Biomedical Engineering
Toronto Metropolitan University
Toronto, Canada
awais.irshad@torontomu.ca

***Abstract*—This document is a final report on the final project for the course COE718, offered by Dr. Gul N. Khan and labs facilitated by Yoga Suhas Kuruba Manjunath. It provides an in depth explanation of the media center implemented for the final project. It contains all code, designs, techniques, project settings and some analysis for where the project falls short of requirements.**

***Index Terms*—To be filled after report is complete**

I. Introduction

The media center project implemented for the course *COE718 - Embedded Systems Design* is one that entails the design and implementation of a simple GUI based media center that displays to the MCB1700 board's liquid crystal display, requiring students to allow for users to control and interact with the system through the MCB1700's joystick.

A high level overview of requirements for the media center and how I planned to meet each of these requirements is given in my interim report [1]. The following sections give a more in depth explanation of those requirements and how I met them. It also mentions how I differed from my planned methods for certain requirements.

The initial approach to updating the LCD based on which app the user selects was to have a global page variable, which allowed for a page-based system that continually updated the LCD based on this variable. While designing and developing such a system, a big factor that steered me away from this method was the strong coupling of code that could easily be refactored if I used threads instead. I did not mention the goal with threads during the interim report as I had to experiment with them a bit more to gauge how well they fit the use case. This resulted in me opting for a thread-based approach instead of checking this variable constantly. Further details to this approach are mentioned in the system design section.

For the photo gallery, all requirements were met. I displayed 3 different pictures converted to C source files, which I have included in the appendix.

For the MP3 player, most of the requirements were met. The audio was outputted onto the board's speakers and allowed for volume adjustment using the potentiometer. However, I could not implement a way for users to leave the MP3 player and get back to the main menu. After discussing with the TA, I theorized how I would go about implementing the above feature if I were to do this project again.

For the game, I used an implementation of snake from a past student's project [2] and made minor changes to integrate it with my media center. The changes are minor and allow for my thread based control to work.

Overall, most requirements were met and for those I did not meet, I have reflected and analyzed how I would go about meeting them were I to do this project again.

II. System Design

A. Project Setup

To begin, I created a new Keil uVision project with the NXP LPC1768 chip.

I then opened the 'Options for Target' menu, where I adjusted my target settings. Under the 'Target' section, in the 'Code Generation' pane, I set the ARM compiler to the option 'Use default compiler version 5.' I then made sure that the IRAM2 area of the memory under the 'Read/Write Memory Area' pane was unchecked. In the 'C/C++' tab, I selected the 'C99 Mode' option. In the 'Debug' tab, I selected the option to use the 'ULINK2/ME Cortex Debugger' and under the settings for this debugger, under its 'Flash Download' tab, I ensured that the 'Reset and Run' option was selected.

After creating this project, I navigated to the *Manage Run-Time Environment* menu where I checked the following options:

- Board Support
  - A/D Converter (API)
  - Joystick (API)
- CMSIS
  - CORE
  - RTOS (API)
    - Keil RTX
- Compiler
  - I/O
    - STDOUT
- Device
  - GPIO
  - PIN
  - Startup

Following that, I went to my project, navigated to the CMSIS section and opened the **RTX_Conf_CM.c** file, navigated to its configuration wizard and set the following settings:

1) Thread Configuration
   - Number of concurrent running user threads - 6
   - Default Thread stack size [bytes] - 2048
   - Main Thread stack size [bytes] - 2048

- Number of threads with user-provided stack size - 0
- Total stack size [bytes] for threads with user-provided stack size - 0
- Stack overflow checking - checked/enabled
- Stack usage watermark - unchecked/disabled
- Processor mode for thread execution - Unprivileged mode

2) RTX Kernel Timer Tick Configuration
   - Use Cortex-M SysTick timer as RTX Kernel Timer - checked/enabled
   - RTOS Kernel Timer input clock frequency [Hz] - 10000000
   - RTX Timer tick interval value [us] - 10000
3) System Configuration
   - Round-Robin Thread switching - unchecked/disabled
   - User Timers - checked/enabled
     - Timer Thread Priority - High
     - Timer Thread stack size [bytes] - 200
     - Timer Callback Queue size - 4
   - ISR FIFO Queue size - 16 entries

Following the above settings, I added the following files from the previous labs to my source folder, which are all present in the Appendix section of this report. All files modified by me are marked as such. The following is a list of files and which lab they are from:

1) Lab 1
   - IRQ.c
   - Font_6x8_h.h
   - Font_16x24_h.h
   - GLCD_SPI_LPC1700.c
   - GLCD.h
   - KBD.c
   - KBD.h
   - LED.c
   - LED.h
2) Lab 3
   - Thread.c (from part 1) - modified
   - main.c - modified
3) Lab 4
   - osObjects.h (from part 1)
4) USBAudio Example
   - adcuser.c
   - adcuser.h
   - audio.h
   - type.h
   - usb.h
   - usbaudio.h
   - usbcfg.h
   - usbcore.c
   - usbcore.h
   - usbdesc.c
   - usbdesc.h
   - usbdmain.c - modified

The remaining project files I have written on my own or are from sources not within the course material. A brief description of each file is given as follows, where external sources are cited:

1) Created
   - mainMenu.c
   - mainMenu.h
   - gallery.c
   - gallery.h
   - game.h
   - IMG_6156.c
   - IMG_6157.c
   - IMG_6158.c
   - usbdmain.h
2) External source
   - game.c [2] - modified

As for files that are generated by Keil, I have modified a single file replaced a single one, as follows:

- system_LPC17xx.c (Startup) - replaced
- LPC17xx.h - modified

The system_LPC17xx.c (startup) file given by Keil upon generating the project, Keil gives us the version of this file from 2016, with the following file description in the code:

```
/*----------------------------------------------------------------------------//**
 * @file  system_LPC17xx.c
 * @brief CMSIS Device System Source File for
 *        NXP LPC17xx Device Series
 * @version V1.14
 * @date  05. April 2016
 *------------------------------------------------------------------------------/
/* Copyright (c) 2012 - 2016 ARM LIMITED
```

This file's APIs for setting the system core clock and accessing the variable containing the core clock frequency differ from ones used in our USB audio example. Therefore, replacing this file with the system_LPC17xx.c (startup) file from the USBAudio example project, included in the appendix and with the following file description in the code, allows us to solve this difference in API names and implementations. The description is as follows:

```
/*----------------------------------------------------------------------------//**
 * @file  system_LPC17xx.c
 * @brief CMSIS Cortex-M3 Device System
    Source File for
 *        NXP LPC17xx Device Series
 * @version V1.13
 * @date  18. April 2012
 *
 * @note
 * Copyright (C) 2009-2012 ARM Limited. All
    rights reserved.
 *
 * @par
 * ARM Limited (ARM) is supplying this
    software for use with Cortex-M
```

```
 * processor based microcontrollers. This
    file can be freely distributed
 * within development tools that are
    supporting such ARM based processors.
 *
 * @par
 * THIS SOFTWARE IS PROVIDED "AS IS". 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.
 * ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE
    LIABLE FOR SPECIAL, INCIDENTAL, OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON
    WHATSOEVER.
 *
 ****************************************/
```

When programming using the NXP LPC1768 chip, most files will use the header file LPC17xx.h, a file that gives information on registers of the LPC1768 chip, and access to these registers through nicely formatted C data structures, allowing for operations to be performed on these registers quickly and conveniently. One such data structure is the LPC_ADC_TypeDef, which is a general data structure to access registers of the analog to digital converter. Originally during project setup, that data structure gives access to a few control registers, as follows:

```
/*------------- Analog-to-Digital Converter
    (ADC) -----------------------------*/
/** @brief Analog-to-Digital Converter (ADC)
    register structure definition */
typedef struct
{
  __IO uint32_t ADCR;
  __IO uint32_t ADGDR;
  __IO uint32_t ADINTEN;
  __I uint32_t ADDR0;
  __I uint32_t ADDR1;
  __I uint32_t ADDR2;
  __I uint32_t ADDR3;
  __I uint32_t ADDR4;
  __I uint32_t ADDR5;
  __I uint32_t ADDR6;
  __I uint32_t ADDR7;
  __I uint32_t ADSTAT;
} LPC_ADC_TypeDef;
```

When setting the ADCR register, the audio feature on the project from the provided example file did not work in a plug and play manner when integrating with the rest of my project. As a result, I navigated into this same header file on the example project, noticing the difference between this data structure between both header files. The media center project's provided header file lacked a few extra registers that the example project had. Adding the following missing registers resulted in the files being easily integratable with the media center:

```
/*------------- Analog-to-Digital Converter
    (ADC) -----------------------------*/
/** @brief Analog-to-Digital Converter (ADC)
    register structure definition */
typedef struct
{
  __IO uint32_t CR;               /*!< Offset:
     0x000 (R/W) A/D Control Register */
  __IO uint32_t GDR;              /*!< Offset:
     0x004 (R/W) A/D Global Data Register */
     uint32_t RESERVED0;
  __IO uint32_t INTEN;            /*!< Offset:
     0x00C (R/W) A/D Interrupt Enable
     Register */
  __I uint32_t DR[8];             /*!< Offset:
     0x010 (R/ ) A/D Channel # Data Register
     */
  __I uint32_t STAT;              /*!< Offset:
     0x030 (R/ ) A/D Status Register */
  __IO uint32_t ADTRM;            /*!< Offset:
     0x034 (R/W) ADC trim Register */
   __IO uint32_t ADCR;
  __IO uint32_t ADGDR;
  __IO uint32_t ADINTEN;
  __I uint32_t ADDR0;
  __I uint32_t ADDR1;
  __I uint32_t ADDR2;
  __I uint32_t ADDR3;
  __I uint32_t ADDR4;
  __I uint32_t ADDR5;
  __I uint32_t ADDR6;
  __I uint32_t ADDR7;
  __I uint32_t ADSTAT;
} LPC_ADC_TypeDef;
```

To be specific, the usb audio files worked directly with the CR register in many instances, and some of the other missing registers on a few occasions. To first avoid as many changes to system files as possible, I tried modifying the the usb audio code to use the ADCR register instead, which failed to work, after which I resorted to only modifying this data structure as to minimally affect other pieces of code using the header file.

B. Page Based Display and Limitations

Page based display, in the context of this project, refers to the project running in a single thread, where the current output to the LCD is dependent on a single variable, named page. When implementing the logic to manage the output, the first thing one would need to test for would be the value of this variable. From there, we could execute actions based on what its value is. The limitations with this project are that logic responsible for running and displaying different apps are present all under the same scope, increasing complexity and decreasing the readability of the project. Code such as checking for joystick inputs depending on the value of the page number and accounting for edge cases would be all present under the same method. If I decided to externalize those checks to functions or files of their own, it would bloat the project, decreasing efficiency and increasing complexity for readers (including myself). If one needed to follow a flow of control, they'd have to span multiple different places in the same file

or over multiple files. Figure 1 visualizes this approach at a high level.

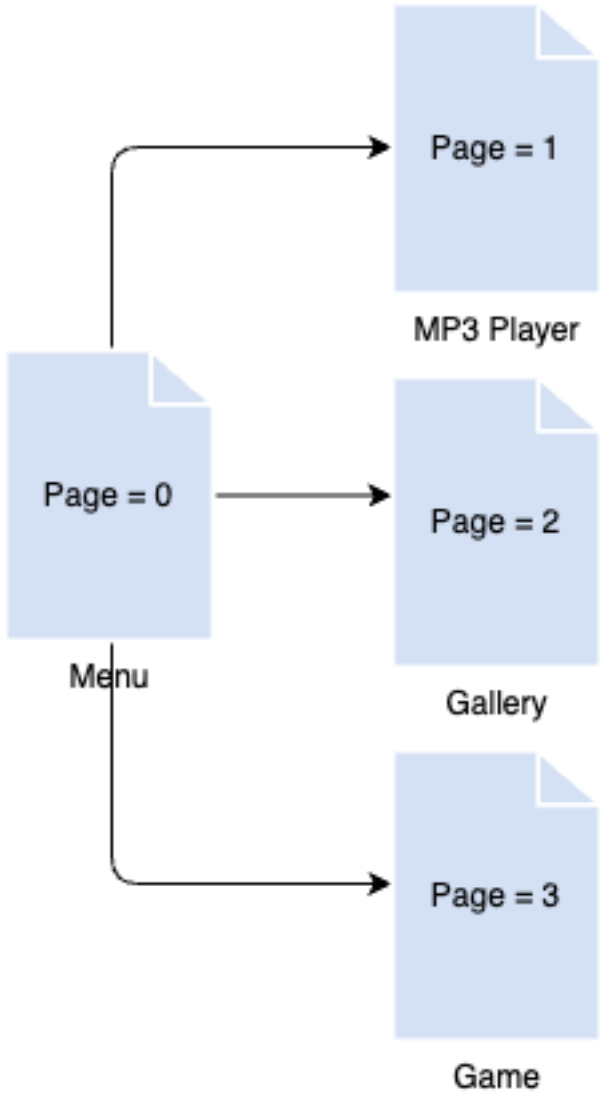

Fig. 1. The global variable, 'Page', determines the application displayed to the LCD

C. Thread Based Display

A thread based display would operate similar to the page based display, but instead of a variable controlling the LCD output and handling joystick inputs from the same thread, each app would have their own thread and handle the joystick and LCD from either within the app's respective file or from within the file that manages each thread. This approach allowed for better modularization of code within my project as well as much better readability.

The basics of thread based display operate similar to the page based display showed in figure 1, except the values of the page variable would correspond to the value of the flag to signal each thread with, apart from the main menu, for which we signaled it with the value 4.

The method to implement a thread based approach would be to initialize all four threads. Of these four threads, the three apps would with be initialized with normal priority and the main menu with high priority. Upon running the project, the first thread to execute would be the main menu, from which users could navigate to apps using the joystick. Upon selecting another app, the priority of that app would also be elevated to high, similar to the priority inheritance protocol that prevents priority inversion, without the mutex [3]. The main menu's thread would then be told to wait indefinitely for a signal. This allows for us to stop execution of the main menu, then once we are ready to leave our app (upon a certain user input), we would signal the main thread and wait indefinitely for the corresponding app's flag, which would then allow us to return to the main menu's thread and subsequently lower the priority of that app's thread.

By separating the code for each app into its own thread, the code's structure turned out to be much cleaner and modifiable in the event of adding more features.

III. Implementation Details

A. Main File

Starting with the main.c file located in Appendix A, it acts as an entry point into the media center program. The main thing this file does is include the appropriate header files to allow for RTX, initialize all peripherals aside from audio, and initializes all threads. Following this, it starts the operating system's thread management system and waits indefinitely, allowing for the Thread.c file to take over execution.

B. Thread Management

Appendix B contains the Thread.c file, which contains the core logic used to switch between threads as well as calls to each app's respective methods. The core idea behind the structure of this file is that the main menu, from which users can navigate to other applications, will have the highest priority among all threads so that it is always the default thread to which our program falls back to. In scenarios where it is not the highest priority thread, it's priority will be the second highest priority and at most, one other thread will higher priority. Firstly, this file defines all threads, defines the method that the main.c file calls to initialize all these threads and defines the function for each thread's execution. Note that all threads are defined with osPriorityNormal except for the MainMenuThread, which has a priority of osPriorityAboveNormal. This allows it to always be above at least 2 other threads, so that when the highest priority thread finishes executing, it always gets the CPU next. This allows us to implement the default fallback behavior to the main menu.

Going through the definition of the MainMenuThread function, we can see that we start off by calling a function to display the main menu, for which we can find the implementation in Appendix C. Since upon initialization of each thread, the main menu has the highest priority, by default our app will start there.

Once a user is in the main menu, the MainMenuThread constantly reads user joystick inputs using the get_button API. The user is able to scroll up and down as well as select an app. A few local variables to keep track of the current input read by the joystick API and the previous input as well as whether the user has selected an app help us to only conditionally call methods in the MainMenu.c file. The MainMenu.c file provides APIs for updating the LCD based on a user's input during MainMenuThread execution. Inside the MainMenu thread is conditional logic that checks the above mentioned local variables and based on their states, it knows whether we should update the display or not. This allows for us to only update 2 lines at a time, a more efficient approach than for example, reloading the LCD consistently when the user is hovering on the same app for a while.

Furthermore, once the appropriate conditions are met and an app is selected, this thread also contains the logic to elevate the appropriate app's thread to osPriorityHigh, signal it, wait indefinitely for a signal to it's flag, and then demote the thread's priority back to osPriorityNormal once it has returned. Priority elevation allows for that thread to execute by default once the MainMenuThread stops executing, and for the board's resources such as the LCD and potentiometer to easily be handled by the appropriate thread. Waiting for its own flag signal makes it possible implement the functionality for users to return back to the main menu upon certain inputs, handled by their own thread. In addition, this thread based display handling has allowed for me to separate user input handling to each thread, making it more readable in each thread's context, whereas page based display would have had most input handling all within one while loop with multiple different conditionals following for each input.

C. Gallery

Upon selecting the gallery, it's thread's priority is raised, the MainMenuThread's execution is delayed indefinitely until it receives the flag signal 0x04, and then the GalleryThread thread is executed. Upon first arriving to the gallery thread, it defines and initializes some local variables and then initializes the gallery for the first time. Most variables are to keep track of which picture is to be displayed, whether the image needs to be cleared, etc. One local variable, called exitApp, is a (software) flag that we keep track of throughout the execution of the thread in order to know whether we need to exit the app. The while loop executes infinitely and each time test whether the flag is set. If it is not, it executes the gallery program like normal. Once the flag is set, the gallery thread clears it, then signals back to the MainMenuThread that it may resume execution. However, it's priority is still higher than the main menu's thread, so it must wait indefinitely until it is called again to allow for the main menu's thread to take over the CPU. This (software) flag is present in all apps, and is the mechanism by which we return to the main menu.

Furthermore, we must also account for the ability of a user to open this app again, which once it does from the main menu, the execution of this thread will start from the point in the code where the app's respective thread is told to wait indefinitely. Therefore, we must reinitialize the app every time we return to this point, which is the last line within the conditional block of code that handles when the exitApp flag is set. This feature is simple enough to implement across all apps and a convenient way of structuring the code so that each app's thread's method is similar in structure, enhancing readability and modifiability.

The remaining code in this thread handles the user inputs so that a user may use the up and down directions of the joystick to scroll between the pictures in the gallery and using the left direction to exit (set exitApp flag to 1). The pictures are C source files, converted from JPEG format to a C source file in RGB565 (16-bit) format, with macros to define the dimensions of the photos. The actualy RGB content of the photo is stores in an array of characters that can be displayed onto the LCD by using its bitmap function with the name of the photo array. By hard coding the names of the pictures before they were converted to C source format, we are able to output the current picture's name as well, only updating the bottom line of the LCD.

D. MP3 Player

Similarly, the code for the MP3 player has the same exitApp flag. While the logic does not work in this instance because it does not trigger any interrupt, I will further discuss how I would go about getting it to work were I to redo this project in the future.

The MP3 player's implementation is exactly as given in the example project. The supporting files for interfacing with USB are included in Appendices G through T. To begin, the file usbhw.c provides hardware level configuration APIs such as initializing the USB audio. It also provides support for Direct Memory Access (DMA), interrupt handling and (USB) command handling. While we only use the initialization method directly from our usbdmain.c file, which is what allows us to stream audio to the board, other supporting files utilize the hardware level APIs this file provides. This file also has an associated header, allowed others to include it. The usbcfg.h header file provides macros for the endpoints, DMA, different kinds of events and their handlers and general configuration setting such as the maximum number of interfaces supported, etc. The hardware abstraction layer code uses these macros to execute its operations. In a way, it is similar to the core configuration file 'core_cm3.h' file we have used for Cortex-M3 specific configuration throughout our labs.

The usbuser.c file (also has an associated header file, useruser.h) manages USB events and the functions that are supposed to execute upon encountering those events, similar to how our callback functions work in multi-threaded systems. It also defines the mechanism by which we are able to 'stream' data (in this case, audio data) to the board.

The usbdesc.c file (also has an associated header file, usbdesc.h) brings all these files together, defining different interfaces for different types of data to be sent through USB. The relevance of this file in the context of this project is that it defines the interface we use for audio streaming as well

as basic functions one would reasonably assume to be present for something streaming audio such as volume adjustment, the events and handlers associated with such a feature and how they interact, etc.

The usbcore.c file (also has an associated header file, usbcore.h) is a file that handles requests for different types of USB connections, called 'classes'. The class relevant to my project is the audio class. It handles all requests in the same general flow, taking in a request, checking what the request is related to, then handles the request and processes the data, then sends the results back to the host device. It is a piece of code that allows the USB to act as different types of hardware, such as audio, keyboard, storage, etc.

Supporting header files like usb.h define standard macros for USBs to allow for easy plug and play integration between the above mentioned files. The type.h file defines standard types one would expect when programming firmware, but is implemented specifically for the NXP LPC1768 chip. audio.h defines macros to do with USB audio class and usbaudio.h defines standard macros for USB audio to do specifically with the NXP LPC1768 chip. The adcuser.c file and its associated header work with the volume APIs provided by other files in order to handle lowering and increasing the volume.

The file that brings together all of the above files is usbdmain.c. It is the file orchestrating volume control by using the potentiometer's APIs to read its level and control the volume using the volume APIs mentioned above. In adding it to my project, I renamed the main method of this file to 'usbAudio', also creating an associated header file, usbdmain.h, allowing it to be called from the MP3 player's thread. In addition to this refactoring, I implented a method to display a picture on the LCD after initializing.

This file also has several more important responsibilities to manage usb audio, such as adjusting volume based on the potentiometer reading read by the 'Analog to Digital (A/D)' converter and defining an interrupt handler for audio interrupts. This interrupt handler allows for continuous audio streaming by constantly updating the audio data coming in.

The current way the code is written accounts for user inputs during the execution time of the 'usbAudio' function, which is too short for a user to be able to exit from the MP3 player successfully. If I were to redo this code, I would test for KBD interrupts within the audio interrupt handler so that while audio interrupts are constantly sent by the system, our code is also constantly checking for user inputs while handling the audio streaming, which uses audio interrupts. In the file usbdmain.c (Appendix V), the section of the interrupt handler where the keyboard input testing would be done is marked by comments.

E. Game

For the game, a simple C implementation of the popular game snake was found online by me from a past student's project [2]. However, the flow of control of the game did not allow for users to exit the game upon my desired input. As a result, some modifications I made were to add another conditional statement when the user was actively playing the game, that if they pressed the select input of the joystick, it would end the current game. From the ended game screen or the initial menu of the game, if the user pushed the left input of the joystick, it would exit the game back to the main menu. This allowed for easy integration with the rest of my project. The modified code is present in Appendix W.

IV. Conclusion

In summary, the media center I implemented used threads to manage each application as well as the main menu, used most files from the example project for audio as well as external sources for the game, with slight modifications to some of the files.It also modified some files provided to us by Keil's run time environment manager, in order to accommodate older code for the USB audio. An emphasis was placed on the readability and modifiability of the project and its structure in order to make continuous integration and development easier.

References

[1] A. Irshad, "Interim Report," 2024, pp 1-3

[2] M. F, "snake.c," github.com, https://github.com/mfoee/MCB1700/blob/master/snake.c

[3] G. N. Khan, "RT Scheduling", p. 47 Toronto Metropolitan University, 2024, https://www.ecb.torontomu.ca/%7Ecourses/coe718/lectures/RT-Scheduling.pdf

Appendix

A. main.c

```
/*------------------------------
 * Name: main.c
 * Purpose: Thread initialization upon system
     startup

 *------------------------------
 * Name: Awais Irshad
 *------------------------------*/

#include <stdio.h>
#include "main.h"
#include "LPC17xx.h"
#include "GLCD.h"
#include "LED.h"
#include "Board_ADC.h"
#include "KBD.h"
#include "mainMenu.h"
#define osObjectsPublic           // define
    objects in main module
#include "osObjects.h"            // RTOS
    object definitions

/* Import external variables from IRQ.c file
                */
extern uint8_t clock_ms;

//void displayMainMenu(void);
extern int Init_Thread (void);

/*----------------------------------
  Main Program
 *----------------------------------*/
```

```
int main (void) {
// initialize peripherals
  LED_Init();                          /* LED
      Initialization    */
  ADC_Initialize();                        /* ADC
      Initialization    */
   KBD_Init();

#ifdef __USE_LCD
  GLCD_Init();                         //
      Initialize graphical LCD (if enabled
#endif
   osKernelInitialize();
   // initialize our threads, the media center
       will run from here

   Init_Thread();
   osKernelStart();
   osDelay(osWaitForever);

}
```

B. *Thread.c*

```
#include "cmsis_os.h"
                              // CMSIS RTOS
    header file
#include <stdio.h>
#include "KBD.h"
#include "mainMenu.h"
#include "gallery.h"
#include "game.h"
#include "usbdmain.h"


void MainMenuThread( void const *argument);
void GalleryThread( void const *argument);
void MP3PlayerThread( void const *argument);
void GameThread( void const *argument);

osThreadId mainMenuId;
osThreadDef(MainMenuThread,
    osPriorityAboveNormal, 1, 0);

osThreadId galleryThreadId;
osThreadDef(GalleryThread, osPriorityNormal,
    1, 0);

osThreadId mp3PlayerThreadId;
osThreadDef(MP3PlayerThread,
    osPriorityNormal, 1, 0);

osThreadId gameThreadId;
osThreadDef(GameThread, osPriorityNormal, 1,
    0);

// global variable for current picture in
    gallery
uint8_t currentPic;

int Init_Thread (void) {
  // create all our threads
  mainMenuId = osThreadCreate
      (osThread(MainMenuThread), NULL);
   galleryThreadId = osThreadCreate
       (osThread(GalleryThread), NULL);
   mp3PlayerThreadId = osThreadCreate
       (osThread(MP3PlayerThread), NULL);
   gameThreadId = osThreadCreate
       (osThread(GameThread), NULL);
   // if any of the threads not created,
       return -1 (failure)
  if((!mainMenuId) || (!galleryThreadId) ||
      (!mp3PlayerThreadId) || (!gameThreadId))
      return(-1);
  return(0);
}

void delay(void);

void MainMenuThread( void const *argument){
   displayMainMenu();
   uint32_t joyStick;
   //uint32_t joyStickPrev = 0U;
   unsigned int currentSelection = 1;
   unsigned int previousSelection = 1;
   unsigned int appSelectedBoolean = 0;
   selectCursor(currentSelection);

   while (1) {                       /* Loop
       forever          */
     if(appSelectedBoolean==0){
       joyStick = get_button();

       // these cases handle navigating the
           main menu
       if(joyStick == KBD_DOWN){
         if(previousSelection!=3){
           if(previousSelection!=currentSelection){
             previousSelection =
                 currentSelection;
           }
           currentSelection+=1;
           clearCursor(previousSelection);
           selectCursor(currentSelection);
         }
         else{
           selectCursor(3);
         }
       }
       else if(joyStick == KBD_UP){
         if(previousSelection!=1){
           if(previousSelection!=currentSelection){
             previousSelection =
                 currentSelection;
           }
           currentSelection-=1;
           clearCursor(previousSelection);
           selectCursor(currentSelection);
         }
         else{
           selectCursor(1);
         }

       }
       // this case handles selecting an app
       else if(joyStick == KBD_SELECT){
         appSelectedBoolean = 1;
       }
     }
     else{
       // switch to the given app based on
           currentSelection, raise that
```

```
            thread's priority to high, set
            mainMenu thread to normal
        // and wait to receive a signal back
            from that thread. Once that app is
            exited, it will set its own
            priority back to normal,
        // set the mainMenu thread to high and
            signal to it.
        switch(currentSelection){
          // Gallery, switch to thread with
              flag 0x01
          case 1:
            osThreadSetPriority(galleryThreadId,
                osPriorityHigh);
            osSignalSet(galleryThreadId, 0x01);
            osSignalWait(0x04, osWaitForever);
            // once we've returned from the
                given app, set its priority
                back to normal
            osThreadSetPriority(galleryThreadId,
                osPriorityNormal);
            break;
          // MP3 Player, switch to thread with
              flag 0x02
          case 2:
            osThreadSetPriority(mp3PlayerThreadId,
                osPriorityHigh);
            osSignalSet(mp3PlayerThreadId,
                0x02);
            osSignalWait(0x04, osWaitForever);
            // once we've returned from the
                given app, set its priority
                back to normal
            osThreadSetPriority(mp3PlayerThreadId,
                osPriorityNormal);
            break;
          // Game, switch to thread with flag
              0x03
          case 3:
            osThreadSetPriority(gameThreadId,
                osPriorityHigh);
            osSignalSet(gameThreadId, 0x03);
            osSignalWait(0x04, osWaitForever);
            // once we've returned from the
                given app, set its priority
                back to normal
            osThreadSetPriority(gameThreadId,
                osPriorityNormal);
            break;
        }

        appSelectedBoolean = 0;
        displayMainMenu();
      }
    }
}

void GalleryThread( void const *argument){
  uint8_t exitApp = 0;
  uint32_t joyStick;
  uint8_t clearImg = 1;
  uint8_t selectedImg = 1;
  currentPic = 1;
  initializeGallery();
  while(1){
    if(exitApp==1){
      // then exit this app, reinitialize it
          if we return
      exitApp = 0; // set this (software)
          flag back to 0,
      //if we return, we don't want the
          infinite loop to go back to the
          main menu right away, which is what
          will happen if it remains 1
      osSignalSet(mainMenuId, 0x04);
      osSignalWait(0x01, osWaitForever);
      initializeGallery();
    }
    else{
      joyStick = get_button();
      // if the clearImg flag is set to 1,
          it means in the previous iteration
          of the while loop
      // the joystick was used to navigate
          to either the next or previous
          picture, therefore
      // set the flag back to 0 and display
          the picture that's value
          corresponds to currentPic
      // which is adjusted by the joystick
          input handling below
      if(clearImg==1){
        clearImg = 0;
        displayPicture(currentPic);
      }
      // exit the gallery
      if(joyStick==KBD_LEFT) {
        exitApp = 1;
      }
      // JOYSTICK INPUT HANDLING AS
          MENTIONED ABOVE
      // next picture
      else if(joyStick == KBD_DOWN){
        // the if block here is to handle
            the value of currentPic staying
            between 1 and 3
        if(currentPic == 3){
          currentPic = 1;
        }
        else{
          currentPic += 1;
        }
        clearImg = 1;
      }
      // previous picture
      else if(joyStick == KBD_UP){
        // the if block here is to handle
            the value of currentPic staying
            between 1 and 3
        if(currentPic == 1){
          currentPic = 3;
        }
        else{
          currentPic -= 1;
        }
        clearImg = 1;
      }
      // call this function so that the
          while loop detecting joystick
          inputs doesn't quickly
      // scroll through the pictures
      delay();
    }
```

```
  }
}

void MP3PlayerThread( void const *argument){
  uint8_t exitApp = 0;
  uint8_t audioRunning = 0;
  uint32_t joyStick;
  while(1){
    if(exitApp==1){
      // then exit this app, reinitialize it
          if we return
      exitApp = 0; // set this (software)
          flag back to 0,
      //if we return, we don't want the
          infinite loop to go back to the
          main menu right away, which is what
          will happen if it remains 1
      osSignalSet(mainMenuId, 0x04);
      osSignalWait(0x02, osWaitForever);

    }
    else{
      joyStick = get_button();
      if(joyStick==KBD_LEFT){
        exitApp = 1;
        //suspendUsbAudio();
      }
      if(exitApp == 0 && audioRunning==0) {
        audioRunning = usbAudio();
      }
    }
  }
}

void GameThread( void const *argument){
  uint8_t exitApp = 0;
  uint32_t joyStick;
  while(1){
    if(exitApp==1){
      // then exit this app, reinitialize it
          if we return
      exitApp = 0; // set this (software)
          flag back to 0,
      //if we return, we don't want the
          infinite loop to go back to the
          main menu right away, which is what
          will happen if it remains 1
      osSignalSet(mainMenuId, 0x04);
      osSignalWait(0x03, osWaitForever);
    }
    else{
      joyStick = get_button();
      if(joyStick == KBD_SELECT) exitApp = 1;
      //int gameReturn = game();
      if(game()==0) exitApp = 1;
    }
  }
}


void delay(void){
  int i,j;
  for(i=0; i<4000; i++){
    for(j=0; j<500; j++){}
  }
}
```

C. MainMenu.c

```
#include "LPC17xx.h"
#include "GLCD.h"

#define __FI    1


// 1 - Gallery, 2 - MP3 Player, 3 - Game
uint8_t currentSelection;
uint8_t previousSelection;

void displayMainMenu(void){
  currentSelection = 3;

   GLCD_Clear(White);
  GLCD_SetBackColor(Red);
  GLCD_SetTextColor(White);
  GLCD_DisplayString(0, 0, __FI, ( unsigned
      char *)"COE718 Media Centre ");
  GLCD_DisplayString(1, 0, __FI, ( unsigned
      char *)" Navigate Menu: ");

  GLCD_SetBackColor(White);
  GLCD_SetTextColor(Black);
   GLCD_DisplayString(2, 0, __FI, ( unsigned
       char *)"--------------------");
  GLCD_DisplayString(3, 0, __FI, ( unsigned
      char *)"| Gallery |");
   GLCD_DisplayString(4, 0, __FI, ( unsigned
       char *)"--------------------");
   GLCD_DisplayString(5, 0, __FI, ( unsigned
       char *)"| MP3 Player |");
   GLCD_DisplayString(6, 0, __FI, ( unsigned
       char *)"--------------------");
   GLCD_DisplayString(7, 0, __FI, ( unsigned
       char *)"| Game |");
   GLCD_DisplayString(8, 0, __FI, ( unsigned
       char *)"--------------------");
   GLCD_SetBackColor(Red);
  GLCD_SetTextColor(White);
   GLCD_DisplayString(9, 0, __FI, ( unsigned
       char *)"    ");
}


// this select cursor only updates the 2
    lines of the main menu that we need to,
    minimizing LCD operations
void selectCursor(unsigned int num){
  switch(num){
    case 1:
      // Gallery
      GLCD_ClearLn(3,__FI);
      GLCD_SetBackColor(Blue);
      GLCD_SetTextColor(White);
      GLCD_DisplayString(3, 0, __FI, (
          unsigned char *)"| Gallery |");
      break;
    case 2:
      // MP3 player
      GLCD_ClearLn(5,__FI);
      GLCD_SetBackColor(Blue);
      GLCD_SetTextColor(White);
```

```
      GLCD_DisplayString(5, 0, __FI, (
          unsigned char *)"| MP3 Player |");
      break;
    case 3:
      // Game
      GLCD_ClearLn(7,__FI);
      GLCD_SetBackColor(Blue);
      GLCD_SetTextColor(White);
      GLCD_DisplayString(7, 0, __FI, (
          unsigned char *)"| Game |");
      break;
  }
}

void clearCursor(unsigned int num){
  switch(num){
    case 1:
      // Take cursor off Gallery
      GLCD_ClearLn(3,__FI);
      GLCD_SetBackColor(White);
      GLCD_SetTextColor(Black);
      GLCD_DisplayString(3, 0, __FI, (
          unsigned char *)"| Gallery |");
      break;
    case 2:
      // Take cursor off MP3 Player
      GLCD_ClearLn(5,__FI);
      GLCD_SetBackColor(White);
      GLCD_SetTextColor(Black);
      GLCD_DisplayString(5, 0, __FI, (
          unsigned char *)"| MP3 Player |");
      break;
    case 3:
      // Take cursor off Game
      GLCD_ClearLn(7,__FI);
      GLCD_SetBackColor(White);
      GLCD_SetTextColor(Black);
      GLCD_DisplayString(7, 0, __FI, (
          unsigned char *)"| Game |");
      break;
  }
}
```

D. *MainMenu.h*

```
#include "LPC17xx.h"
#include "GLCD.h"

#define __FI    1

void displayMainMenu(void);
void selectCursor(unsigned int num);
void clearCursor(unsigned int num);
```

E. *gallery.c*

```
#include "GLCD.h"
#include "LPC17xx.h"
#include "stdint.h"
#include "IMG_6156.c" // picNum 1
#include "IMG_6157.c" // picNum 2
#include "IMG_6158.c" // picNum 3

// an array that holds the different numbers
    that represent the pictures in our gallery
uint8_t gallery[3] = {1,2,3};
unsigned char *picNames [] = {
  (unsigned char *) "| Awais.jpg |",
  (unsigned char *) "| MCB.jpg |",
  (unsigned char *) "| Code.jpg |"
};


extern uint8_t currentPic;
// this function clears the current picture
    and re displays the header and scroll
    options for our gallery
void clearLCDForGallery(void){
  GLCD_Clear(White);
  GLCD_SetBackColor(Blue);
  GLCD_SetTextColor(White);
  GLCD_DisplayString(0, 0, 1, ( unsigned char
      *)"| Gallery |");
  GLCD_SetBackColor(Red);
  GLCD_SetTextColor(White);
  GLCD_DisplayString(4, 0, 0, ( unsigned char
      *)" down = next picture, up = previous,
      left = exit ");
  //GLCD_DisplayString(26, 0, 0,
      picNames[currentPic-1]);
  GLCD_DisplayString(9, 0, 1, ( unsigned char
      *)"|  |");
}

// this function displays a picture in our
    gallery
void displayPicture(uint8_t picNum){
  // clear the display between each display,
      so reinitialize the gallery
  clearLCDForGallery();
  switch(picNum){
    case 1:
      GLCD_Bitmap( 90, 58, 150, 147,
          (unsigned char*) SELFIE_pixel_data);
      break;
    case 2:
      GLCD_Bitmap( 90, 58, 150, 150,
          (unsigned char*)
          BOARDPIC_pixel_data);
      break;
    case 3:
      GLCD_Bitmap( 90, 58, 150, 144,
          (unsigned char*)
          MYCODEPIC_pixel_data);
      break;
  }
  // display the name of the picture
  GLCD_ClearLn(9, 1);
  GLCD_DisplayString(9, 0, 1,
      picNames[picNum-1]);
}

// to be called whenever the gallery thread
    is signalled to execute
void initializeGallery(void){
  currentPic = 1;
  clearLCDForGallery();
  // start the gallery off by displaying the
      very first
```

```
  displayPicture(currentPic);
}
```

F. gallery.h

```
#include "stdint.h"
void initializeGallery(void);
void displayPicture(uint8_t picNum);
extern uint8_t currentPic;
```

G. usbhw.c

```
/*----------------------------------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------------------------------
 * Name: usbhw.c
 * Purpose: USB Hardware Layer Module for
    NXP's LPC17xx MCU
 * Version: V1.20
 *----------------------------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------------------------------
 * History:
 *       V1.20 Added USB_ClearEPBuf
 *       V1.00 Initial Version
 *----------------------------------------------------------------------------*/
#include "LPC17xx.h"                    /* LPC17xx
    definitions */
#include "type.h"

#include "usb.h"
#include "usbcfg.h"
#include "usbreg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbuser.h"


#pragma diag_suppress 1441


#define EP_MSK_CTRL 0x0001 /* Control
    Endpoint Logical Address Mask */
#define EP_MSK_BULK 0xC924 /* Bulk Endpoint
    Logical Address Mask */
#define EP_MSK_INT 0x4492 /* Interrupt
    Endpoint Logical Address Mask */
#define EP_MSK_ISO 0x1248 /* Isochronous
    Endpoint Logical Address Mask */


#if USB_DMA

#pragma arm section zidata = "USB_RAM"
uint32_t UDCA[USB_EP_NUM];          /* UDCA in
    USB RAM */

uint32_t DD_NISO_Mem[4*DD_NISO_CNT]; /*
    Non-Iso DMA Descriptor Memory */
uint32_t DD_ISO_Mem [5*DD_ISO_CNT]; /* Iso
    DMA Descriptor Memory */
#pragma arm section zidata
uint32_t udca[USB_EP_NUM];          /* UDCA
    saved values */
uint32_t DDMemMap[2];               /* DMA
    Descriptor Memory Usage */

#endif

/*
 * Get Endpoint Physical Address
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: Endpoint Physical Address
 */

uint32_t EPAdr (uint32_t EPNum) {
  uint32_t val;

  val = (EPNum & 0x0F) << 1;
  if (EPNum & 0x80) {
    val += 1;
  }
  return (val);
}

/*
 * Write Command
 *   Parameters:  cmd: Command
 *   Return Value: None
 */

void WrCmd (uint32_t cmd) {

  LPC_USB->DevIntClr = CCEMTY_INT;
  LPC_USB->CmdCode = cmd;
  while ((LPC_USB->DevIntSt & CCEMTY_INT) ==
      0);
}


/*
 * Write Command Data
 *   Parameters:  cmd: Command
 *                val: Data
 *   Return Value: None
 */

void WrCmdDat (uint32_t cmd, uint32_t val) {
```

```
  LPC_USB->DevIntClr = CCEMTY_INT;
  LPC_USB->CmdCode = cmd;
  while ((LPC_USB->DevIntSt & CCEMTY_INT) ==
      0);
  LPC_USB->DevIntClr = CCEMTY_INT;
  LPC_USB->CmdCode = val;
  while ((LPC_USB->DevIntSt & CCEMTY_INT) ==
      0);
}


/*
 * Write Command to Endpoint
 *   Parameters:  cmd: Command
 *                val: Data
 *   Return Value: None
 */

void WrCmdEP (uint32_t EPNum, uint32_t cmd){

  LPC_USB->DevIntClr = CCEMTY_INT;
  LPC_USB->CmdCode = CMD_SEL_EP(EPAdr(EPNum));
  while ((LPC_USB->DevIntSt & CCEMTY_INT) ==
      0);
  LPC_USB->DevIntClr = CCEMTY_INT;
  LPC_USB->CmdCode = cmd;
  while ((LPC_USB->DevIntSt & CCEMTY_INT) ==
      0);
}


/*
 * Read Command Data
 *   Parameters:  cmd: Command
 *   Return Value: Data Value
 */

uint32_t RdCmdDat (uint32_t cmd) {

  LPC_USB->DevIntClr = CCEMTY_INT | CDFULL_INT;
  LPC_USB->CmdCode = cmd;
  while ((LPC_USB->DevIntSt & CDFULL_INT) ==
      0);
  return (LPC_USB->CmdData);
}


/*
 * USB Initialize Function
 *  Called by the User to initialize USB
 *   Return Value: None
 */

void USB_Init (void) {

  LPC_PINCON->PINSEL1 &= ~((3<<26)|(3<<28));
      /* P0.29 D+, P0.30 D- */
  LPC_PINCON->PINSEL1 |= ((1<<26)|(1<<28)); /*
      PINSEL1 26.27, 28.29 = 01 */

  LPC_PINCON->PINSEL3 &= ~((3<< 4)|(3<<28));
      /* P1.18 GoodLink, P1.30 VBUS */
  LPC_PINCON->PINSEL3 |= ((1<< 4)|(2<<28)); /*
      PINSEL3 4.5 = 01, 28.29 = 10 */

  LPC_PINCON->PINSEL4 &= ~((3<<18) ); /* P2.9
      SoftConnect */
  LPC_PINCON->PINSEL4 |= ((1<<18) ); /*
      PINSEL4 18.19 = 01 */

  LPC_SC->PCONP |= (1UL<<31);      /* USB PCLK
      -> enable USB Per. */

  LPC_USB->USBClkCtrl = 0x12;      /* Dev, AHB
      clock enable */
  while ((LPC_USB->USBClkSt & 0x12) != 0x12);

  NVIC_EnableIRQ(USB_IRQn);      /* enable USB
      interrupt */

  USB_Reset();
  USB_SetAddress(0);
}


/*
 * USB Connect Function
 *  Called by the User to Connect/Disconnect
     USB
 *   Parameters:  con: Connect/Disconnect
 *   Return Value: None
 */

void USB_Connect (uint32_t con) {
  WrCmdDat(CMD_SET_DEV_STAT, DAT_WR_BYTE(con ?
      DEV_CON : 0));
}


/*
 * USB Reset Function
 *  Called automatically on USB Reset
 *   Return Value: None
 */

void USB_Reset (void) {
#if USB_DMA
  uint32_t n;
#endif

  LPC_USB->EpInd = 0;
  LPC_USB->MaxPSize = USB_MAX_PACKET0;
  LPC_USB->EpInd = 1;
  LPC_USB->MaxPSize = USB_MAX_PACKET0;
  while ((LPC_USB->DevIntSt & EP_RLZED_INT) ==
      0);

  LPC_USB->EpIntClr = 0xFFFFFFFF;
  LPC_USB->EpIntEn = 0xFFFFFFFF ^ USB_DMA_EP;
  LPC_USB->DevIntClr = 0xFFFFFFFF;
  LPC_USB->DevIntEn = DEV_STAT_INT |
      EP_SLOW_INT |
            (USB_SOF_EVENT ? FRAME_INT : 0) |
            (USB_ERROR_EVENT ? ERR_INT : 0);

#if USB_DMA
  LPC_USB->UDCAH = USB_RAM_ADR;
  LPC_USB->DMARClr = 0xFFFFFFFF;
  LPC_USB->EpDMADis = 0xFFFFFFFF;
  LPC_USB->EpDMAEn = USB_DMA_EP;
  LPC_USB->EoTIntClr = 0xFFFFFFFF;
  LPC_USB->NDDRIntClr = 0xFFFFFFFF;
  LPC_USB->SysErrIntClr = 0xFFFFFFFF;
  LPC_USB->DMAIntEn = 0x00000007;
```

```
  DDMemMap[0] = 0x00000000;
  DDMemMap[1] = 0x00000000;
  for (n = 0; n < USB_EP_NUM; n++) {
    udca[n] = 0;
    UDCA[n] = 0;
  }
#endif
}


/*
 * USB Suspend Function
 *  Called automatically on USB Suspend
 *   Return Value: None
 */

void USB_Suspend (void) {
  /* Performed by Hardware */
}


/*
 * USB Resume Function
 *  Called automatically on USB Resume
 *   Return Value: None
 */

void USB_Resume (void) {
  /* Performed by Hardware */
}


/*
 * USB Remote Wakeup Function
 *  Called automatically on USB Remote Wakeup
 *   Return Value: None
 */

void USB_WakeUp (void) {

  if (USB_DeviceStatus &
      USB_GETSTATUS_REMOTE_WAKEUP) {
    WrCmdDat(CMD_SET_DEV_STAT,
        DAT_WR_BYTE(DEV_CON));
  }
}


/*
 * USB Remote Wakeup Configuration Function
 *   Parameters:  cfg: Enable/Disable
 *   Return Value: None
 */

void USB_WakeUpCfg (uint32_t cfg) {
  /* Not needed */
}


/*
 * USB Set Address Function
 *   Parameters:  adr: USB Address
 *   Return Value: None
 */

void USB_SetAddress (uint32_t adr) {
  WrCmdDat(CMD_SET_ADDR, DAT_WR_BYTE(DEV_EN |
      adr)); /* Don't wait for next */
  WrCmdDat(CMD_SET_ADDR, DAT_WR_BYTE(DEV_EN |
      adr)); /* Setup Status Phase */
}


/*
 * USB Configure Function
 *   Parameters:  cfg: Configure/Deconfigure
 *   Return Value: None
 */

void USB_Configure (uint32_t cfg) {

  WrCmdDat(CMD_CFG_DEV, DAT_WR_BYTE(cfg ?
      CONF_DVICE : 0));

  LPC_USB->ReEp = 0x00000003;
  while ((LPC_USB->DevIntSt & EP_RLZED_INT) ==
      0);
  LPC_USB->DevIntClr = EP_RLZED_INT;
}


/*
 * Configure USB Endpoint according to
     Descriptor
 *   Parameters:  pEPD: Pointer to Endpoint
     Descriptor
 *   Return Value: None
 */

void USB_ConfigEP (USB_ENDPOINT_DESCRIPTOR
    *pEPD) {
  uint32_t num;

  num = EPAdr(pEPD->bEndpointAddress);
  LPC_USB->ReEp |= (1 << num);
  LPC_USB->EpInd = num;
  LPC_USB->MaxPSize = pEPD->wMaxPacketSize;
  while ((LPC_USB->DevIntSt & EP_RLZED_INT) ==
      0);
  LPC_USB->DevIntClr = EP_RLZED_INT;
}


/*
 * Set Direction for USB Control Endpoint
 *   Parameters:  dir: Out (dir == 0), In (dir
     <> 0)
 *   Return Value: None
 */

void USB_DirCtrlEP (uint32_t dir) {
  /* Not needed */
}


/*
 * Enable USB Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                EPNum.0..3: Address
 *                EPNum.7: Dir
 *   Return Value: None
 */
```

```
void USB_EnableEP (uint32_t EPNum) {
  WrCmdDat(CMD_SET_EP_STAT(EPAdr(EPNum)),
      DAT_WR_BYTE(0));
}


/*
 * Disable USB Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_DisableEP (uint32_t EPNum) {
  WrCmdDat(CMD_SET_EP_STAT(EPAdr(EPNum)),
      DAT_WR_BYTE(EP_STAT_DA));
}


/*
 * Reset USB Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_ResetEP (uint32_t EPNum) {
  WrCmdDat(CMD_SET_EP_STAT(EPAdr(EPNum)),
      DAT_WR_BYTE(0));
}


/*
 * Set Stall for USB Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_SetStallEP (uint32_t EPNum) {
  WrCmdDat(CMD_SET_EP_STAT(EPAdr(EPNum)),
      DAT_WR_BYTE(EP_STAT_ST));
}


/*
 * Clear Stall for USB Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_ClrStallEP (uint32_t EPNum) {
  WrCmdDat(CMD_SET_EP_STAT(EPAdr(EPNum)),
      DAT_WR_BYTE(0));
}


/*
 * Clear USB Endpoint Buffer
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_ClearEPBuf (uint32_t EPNum) {
  WrCmdEP(EPNum, CMD_CLR_BUF);
}


/*
 * Read USB Endpoint Data
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *                pData: Pointer to Data Buffer
 *   Return Value: Number of bytes read
 */

uint32_t USB_ReadEP (uint32_t EPNum, uint8_t
    *pData) {
  uint32_t cnt, n;

  LPC_USB->Ctrl = ((EPNum & 0x0F) << 2) |
      CTRL_RD_EN;

  do {
    cnt = LPC_USB->RxPLen;
  } while ((cnt & PKT_RDY) == 0);
  cnt &= PKT_LNGTH_MASK;

  for (n = 0; n < (cnt + 3) / 4; n++) {
    *((__packed uint32_t *)pData) =
        LPC_USB->RxData;
    pData += 4;
  }
  LPC_USB->Ctrl = 0;

  if (((EP_MSK_ISO >> EPNum) & 1) == 0) { /*
      Non-Isochronous Endpoint */
    WrCmdEP(EPNum, CMD_CLR_BUF);
  }
  return (cnt);
}


/*
 * Write USB Endpoint Data
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *                pData: Pointer to Data Buffer
 *                cnt: Number of bytes to write
 *   Return Value: Number of bytes written
 */

uint32_t USB_WriteEP (uint32_t EPNum, uint8_t
    *pData, uint32_t cnt) {
  uint32_t n;

  LPC_USB->Ctrl = ((EPNum & 0x0F) << 2) |
      CTRL_WR_EN;

  LPC_USB->TxPLen = cnt;

  for (n = 0; n < (cnt + 3) / 4; n++) {
    LPC_USB->TxData = *((__packed uint32_t
        *)pData);
    pData += 4;
```

```
  }
  LPC_USB->Ctrl = 0;
  WrCmdEP(EPNum, CMD_VALID_BUF);
  return (cnt);
}

#if USB_DMA

/* DMA Descriptor Memory Layout */
const uint32_t DDAdr[2] = { DD_NISO_ADR,
    DD_ISO_ADR };
const uint32_t DDSz [2] = { 16, 20     };


/*
 * Setup USB DMA Transfer for selected
     Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                pDD: Pointer to DMA
     Descriptor
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t USB_DMA_Setup(uint32_t EPNum,
    USB_DMA_DESCRIPTOR *pDD) {
  uint32_t num, ptr, nxt, iso, n;

  iso = pDD->Cfg.Type.IsoEP;      /* Iso or
      Non-Iso Descriptor */
  num = EPAdr(EPNum);             /*
      Endpoint's Physical Address */

  ptr = 0;                        /* Current
      Descriptor */
  nxt = udca[num];                /* Initial
      Descriptor */
  while (nxt) {                   /* Go
      through Descriptor List */
    ptr = nxt;                    /* Current
        Descriptor */
    if (!pDD->Cfg.Type.Link) {    /* Check for
        Linked Descriptors */
      n = (ptr - DDAdr[iso]) / DDSz[iso]; /*
          Descriptor Index */
      DDMemMap[iso] &= ~(1 << n); /* Unmark
          Memory Usage */
    }
    nxt = *((uint32_t *)ptr);       /* Next
        Descriptor */
  }

  for (n = 0; n < 32; n++) {      /* Search
      for available Memory */
    if ((DDMemMap[iso] & (1 << n)) == 0) {
      break;                      /* Memory
          found */
    }
  }
  if (n == 32) return (FALSE);    /* Memory
      not available */

  DDMemMap[iso] |= 1 << n;        /* Mark
      Memory Usage */
  nxt = DDAdr[iso] + n * DDSz[iso]; /* Next
      Descriptor */
  if (ptr && pDD->Cfg.Type.Link) {
    *((uint32_t *)(ptr + 0)) = nxt; /* Link in
        new Descriptor */
    *((uint32_t *)(ptr + 4)) |= 0x00000004; /*
        Next DD is Valid */
  } else {
    udca[num] = nxt;              /* Save new
        Descriptor */
    UDCA[num] = nxt;              /* Update
        UDCA in USB */
  }

  /* Fill in DMA Descriptor */
  *(((uint32_t *)nxt)++) = 0;       /* Next DD
      Pointer */
  *(((uint32_t *)nxt)++) = pDD->Cfg.Type.ATLE |
                           (pDD->Cfg.Type.IsoEP << 4) |
                           (pDD->MaxSize << 5) |
                           (pDD->BufLen << 16);
  *(((uint32_t *)nxt)++) = pDD->BufAdr;
  *(((uint32_t *)nxt)++) =
      pDD->Cfg.Type.LenPos << 8;
  if (iso) {
    *((uint32_t *)nxt) = pDD->InfoAdr;
  }

  return (TRUE); /* Success */
}


/*
 * Enable USB DMA Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_DMA_Enable (uint32_t EPNum) {
  LPC_USB->EpDMAEn = 1 << EPAdr(EPNum);
}


/*
 * Disable USB DMA Endpoint
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: None
 */

void USB_DMA_Disable (uint32_t EPNum) {
  LPC_USB->EpDMADis = 1 << EPAdr(EPNum);
}


/*
 * Get USB DMA Endpoint Status
 *   Parameters:  EPNum: Endpoint Number
 *                  EPNum.0..3: Address
 *                  EPNum.7: Dir
 *   Return Value: DMA Status
 */

uint32_t USB_DMA_Status (uint32_t EPNum) {
  uint32_t ptr, val;
```

```
  ptr = UDCA[EPAdr(EPNum)];       /* Current
      Descriptor */
  if (ptr == 0)
   return (USB_DMA_INVALID);

  val = *((uint32_t *)(ptr + 3*4)); /* Status
      Information */
  switch ((val >> 1) & 0x0F) {
    case 0x00:                    /* Not
        serviced */
      return (USB_DMA_IDLE);
    case 0x01:                    /* Being
        serviced */
      return (USB_DMA_BUSY);
    case 0x02:                    /* Normal
        Completition */
      return (USB_DMA_DONE);
    case 0x03:                    /* Data
        Under Run */
      return (USB_DMA_UNDER_RUN);
    case 0x08:                    /* Data Over
        Run */
      return (USB_DMA_OVER_RUN);
    case 0x09:                    /* System
        Error */
      return (USB_DMA_ERROR);
  }

  return (USB_DMA_UNKNOWN);
}


/*
 * Get USB DMA Endpoint Current Buffer Address
 *   Parameters:  EPNum: Endpoint Number
 *                EPNum.0..3: Address
 *                EPNum.7: Dir
 *   Return Value: DMA Address (or -1 when DMA
      is Invalid)
 */

uint32_t USB_DMA_BufAdr (uint32_t EPNum) {
  uint32_t ptr, val;

  ptr = UDCA[EPAdr(EPNum)];       /* Current
      Descriptor */
  if (ptr == 0)
  {
   return ((uint32_t)(-1));      /* DMA
       Invalid */
  }

  val = *((uint32_t *)(ptr + 2*4)); /* Buffer
      Address */
  return (val);                   /* Current
      Address */
}


/*
 * Get USB DMA Endpoint Current Buffer Count
 *  Number of transfered Bytes or Iso Packets
 *   Parameters:  EPNum: Endpoint Number
 *                EPNum.0..3: Address
 *                EPNum.7: Dir
 *   Return Value: DMA Count (or -1 when DMA
      is Invalid)
 */

uint32_t USB_DMA_BufCnt (uint32_t EPNum) {
  uint32_t ptr, val;

  ptr = UDCA[EPAdr(EPNum)];       /* Current
      Descriptor */
  if (ptr == 0)
  {
   return ((uint32_t)(-1));      /* DMA
       Invalid */
  }
  val = *((uint32_t *)(ptr + 3*4)); /* Status
      Information */
  return (val >> 16);             /* Current
      Count */
}


#endif /* USB_DMA */


/*
 * Get USB Last Frame Number
 *   Parameters:  None
 *   Return Value: Frame Number
 */

uint32_t USB_GetFrame (void) {
  uint32_t val;

  WrCmd(CMD_RD_FRAME);
  val = RdCmdDat(DAT_RD_FRAME);
  val = val | (RdCmdDat(DAT_RD_FRAME) << 8);

  return (val);
}


/*
 * USB Interrupt Service Routine
 */

void USB_IRQHandler (void) {
  uint32_t disr, val, n, m;
  uint32_t episr, episrCur;

  disr = LPC_USB->DevIntSt; /* Device
      Interrupt Status */

  /* Device Status Interrupt (Reset, Connect
      change, Suspend/Resume) */
  if (disr & DEV_STAT_INT) {
    LPC_USB->DevIntClr = DEV_STAT_INT;
    WrCmd(CMD_GET_DEV_STAT);
    val = RdCmdDat(DAT_GET_DEV_STAT); /*
        Device Status */
    if (val & DEV_RST) {          /* Reset */
      USB_Reset();
#if  USB_RESET_EVENT
      USB_Reset_Event();
#endif
    }
    if (val & DEV_CON_CH) {       /* Connect
        change */
#if  USB_POWER_EVENT
      USB_Power_Event(val & DEV_CON);
```

```
#endif
   }
   if (val & DEV_SUS_CH) {        /*
       Suspend/Resume */
     if (val & DEV_SUS) {         /* Suspend */
       USB_Suspend();
#if    USB_SUSPEND_EVENT
       USB_Suspend_Event();
#endif
     } else {                     /* Resume */
       USB_Resume();
#if    USB_RESUME_EVENT
       USB_Resume_Event();
#endif
     }
   }
   goto isr_end;
  }

#if USB_SOF_EVENT
  /* Start of Frame Interrupt */
  if (disr & FRAME_INT) {
    USB_SOF_Event();
  }
#endif

#if USB_ERROR_EVENT
  /* Error Interrupt */
  if (disr & ERR_INT) {
    WrCmd(CMD_RD_ERR_STAT);
    val = RdCmdDat(DAT_RD_ERR_STAT);
    USB_Error_Event(val);
  }
#endif

  /* Endpoint's Slow Interrupt */
  if (disr & EP_SLOW_INT) {
    episrCur = 0;
    episr  = LPC_USB->EpIntSt;
    for (n = 0; n < USB_EP_NUM; n++) { /*
        Check All Endpoints */
      if (episr == episrCur) break; /* break if
          all EP interrupts handled */
      if (episr & (1 << n)) {
        episrCur |= (1 << n);
        m = n >> 1;

        LPC_USB->EpIntClr = (1 << n);
        while ((LPC_USB->DevIntSt & CDFULL_INT)
            == 0);
        val = LPC_USB->CmdData;

        if ((n & 1) == 0) {       /* OUT
            Endpoint */
          if (n == 0) {           /* Control
              OUT Endpoint */
            if (val & EP_SEL_STP) { /* Setup
                Packet */
              if (USB_P_EP[0]) {
                USB_P_EP[0](USB_EVT_SETUP);
                continue;
              }
            }
          }
          if (USB_P_EP[m]) {
            USB_P_EP[m](USB_EVT_OUT);
          }
        } else {                  /* IN
            Endpoint */
          if (USB_P_EP[m]) {
            USB_P_EP[m](USB_EVT_IN);
          }
        }
      }
    }
    LPC_USB->DevIntClr = EP_SLOW_INT;
  }

#if USB_DMA

  if (LPC_USB->DMAIntSt & 0x00000001) { /* End
      of Transfer Interrupt */
    val = LPC_USB->EoTIntSt;
    for (n = 2; n < USB_EP_NUM; n++) { /*
        Check All Endpoints */
      if (val & (1 << n)) {
        m = n >> 1;
        if ((n & 1) == 0) {       /* OUT
            Endpoint */
          if (USB_P_EP[m]) {
            USB_P_EP[m](USB_EVT_OUT_DMA_EOT);
          }
        } else {                  /* IN
            Endpoint */
          if (USB_P_EP[m]) {
            USB_P_EP[m](USB_EVT_IN_DMA_EOT);
          }
        }
      }
    }
    LPC_USB->EoTIntClr = val;
  }

  if (LPC_USB->DMAIntSt & 0x00000002) { /* New
      DD Request Interrupt */
    val = LPC_USB->NDDRIntSt;
    for (n = 2; n < USB_EP_NUM; n++) { /*
        Check All Endpoints */
      if (val & (1 << n)) {
        m = n >> 1;
        if ((n & 1) == 0) {       /* OUT
            Endpoint */
          if (USB_P_EP[m]) {
            USB_P_EP[m](USB_EVT_OUT_DMA_NDR);
          }
        } else {                  /* IN
            Endpoint */
          if (USB_P_EP[m]) {
            USB_P_EP[m](USB_EVT_IN_DMA_NDR);
          }
        }
      }
    }
    LPC_USB->NDDRIntClr = val;
  }

  if (LPC_USB->DMAIntSt & 0x00000004) { /*
      System Error Interrupt */
    val = LPC_USB->SysErrIntSt;
    for (n = 2; n < USB_EP_NUM; n++) { /*
        Check All Endpoints */
      if (val & (1 << n)) {
        m = n >> 1;
```

```
      if ((n & 1) == 0) {        /* OUT
          Endpoint */
        if (USB_P_EP[m]) {
          USB_P_EP[m](USB_EVT_OUT_DMA_ERR);
        }
      } else {                   /* IN
          Endpoint */
        if (USB_P_EP[m]) {
          USB_P_EP[m](USB_EVT_IN_DMA_ERR);
        }
      }
    }
  }
  LPC_USB->SysErrIntClr = val;
 }

#endif /* USB_DMA */

isr_end:
 return;
}
```

H. usbhw.h

```
/*----------------------------------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------------------------------
 * Name: usbhw.h
 * Purpose: USB Hardware Layer Definitions
 * Version: V1.20
 *----------------------------------------------------------------------------
 *    This software is supplied "AS IS" without any warranties, express,
 *    implied or statutory, including but not limited to the implied
 *    warranties of fitness for purpose, satisfactory quality and
 *    noninfringement. Keil extends you a royalty-free right to reproduce
 *    and distribute executable files created using this software for use
 *    on NXP Semiconductors LPC family microcontroller devices only. Nothing
 *    else gives you the right to use this software.
 *
 * Copyright (c) 2009 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------
 * History:
 *       V1.20 Added USB_ClearEPBuf
 *       V1.00 Initial Version
 *----------------------------------------------------------------------------*/

#ifndef __USBHW_H__
#define __USBHW_H__


/* USB RAM Definitions */
#define USB_RAM_ADR 0x20080000 /* USB RAM Start Address */
#define USB_RAM_SZ 0x00004000 /* USB RAM Size (4kB) */

/* DMA Endpoint Descriptors */
#define DD_NISO_CNT     16 /* Non-Iso EP DMA Descr. Count (max. 32) */
#define DD_ISO_CNT       8 /* Iso EP DMA Descriptor Count (max. 32) */
#define DD_NISO_SZ (DD_NISO_CNT * 16) /* Non-Iso DMA Descr. Size */
#define DD_ISO_SZ (DD_ISO_CNT * 20) /* Iso DMA Descriptor Size */
#define DD_NISO_ADR (USB_RAM_ADR + 128) /* Non-Iso DMA Descr. Address */
#define DD_ISO_ADR (DD_NISO_ADR + DD_NISO_SZ) /* Iso DMA Descr. Address */
#define DD_SZ          (128 + DD_NISO_SZ + DD_ISO_SZ) /* Descr. Size */

/* DMA Buffer Memory Definitions */
#define DMA_BUF_ADR (USB_RAM_ADR + DD_SZ) /* DMA Buffer Start Address */
#define DMA_BUF_SZ (USB_RAM_SZ - DD_SZ) /* DMA Buffer Size */

/* USB Error Codes */
#define USB_ERR_PID  0x0001 /* PID Error */
#define USB_ERR_UEPKT 0x0002 /* Unexpected Packet */
#define USB_ERR_DCRC 0x0004 /* Data CRC Error */
#define USB_ERR_TIMOUT 0x0008 /* Bus Time-out Error */
#define USB_ERR_EOP  0x0010 /* End of Packet Error */
#define USB_ERR_B_OVRN 0x0020 /* Buffer Overrun */
#define USB_ERR_BTSTF 0x0040 /* Bit Stuff Error */
#define USB_ERR_TGL  0x0080 /* Toggle Bit Error */

/* USB DMA Status Codes */
#define USB_DMA_INVALID 0x0000 /* DMA Invalid - Not Configured */
#define USB_DMA_IDLE 0x0001 /* DMA Idle - Waiting for Trigger */
#define USB_DMA_BUSY 0x0002 /* DMA Busy - Transfer in progress */
#define USB_DMA_DONE 0x0003 /* DMA Transfer Done (no Errors)*/
#define USB_DMA_OVER_RUN 0x0004 /* Data Over Run */
#define USB_DMA_UNDER_RUN 0x0005 /* Data Under Run (Short Packet) */
#define USB_DMA_ERROR 0x0006 /* Error */
#define USB_DMA_UNKNOWN 0xFFFF /* Unknown State */

/* USB DMA Descriptor */
typedef struct _USB_DMA_DESCRIPTOR {
  uint32_t BufAdr;            /* DMA Buffer Address */
  uint16_t BufLen;             /* DMA Buffer Length */
  uint16_t MaxSize;            /* Maximum Packet Size */
  uint32_t InfoAdr;           /* Packet Info Memory Address */
  union {                    /* DMA Configuration */
```

```
    struct {
      uint32_t Link : 1;       /* Link to
          existing Descriptors */
      uint32_t IsoEP : 1;      /* Isonchronous
          Endpoint */
      uint32_t ATLE : 1;       /* ATLE (Auto
          Transfer Length Extract) */
      uint32_t Rsrvd : 5;      /* Reserved */
      uint32_t LenPos : 8;     /* Length
          Position (ATLE) */
    } Type;
    uint32_t Val;
  } Cfg;
} USB_DMA_DESCRIPTOR;

/* USB Hardware Functions */
extern void USB_Init (void);
extern void USB_Connect (uint32_t con);
extern void USB_Reset (void);
extern void USB_Suspend (void);
extern void USB_Resume (void);
extern void USB_WakeUp (void);
extern void USB_WakeUpCfg (uint32_t cfg);
extern void USB_SetAddress (uint32_t adr);
extern void USB_Configure (uint32_t cfg);
extern void USB_ConfigEP
    (USB_ENDPOINT_DESCRIPTOR *pEPD);
extern void USB_DirCtrlEP (uint32_t dir);
extern void USB_EnableEP (uint32_t EPNum);
extern void USB_DisableEP (uint32_t EPNum);
extern void USB_ResetEP (uint32_t EPNum);
extern void USB_SetStallEP (uint32_t EPNum);
extern void USB_ClrStallEP (uint32_t EPNum);
extern void USB_ClearEPBuf (uint32_t EPNum);
extern uint32_t USB_ReadEP (uint32_t EPNum,
    uint8_t *pData);
extern uint32_t USB_WriteEP (uint32_t EPNum,
    uint8_t *pData, uint32_t cnt);
extern uint32_t USB_DMA_Setup (uint32_t
    EPNum, USB_DMA_DESCRIPTOR *pDD);
extern void USB_DMA_Enable (uint32_t EPNum);
extern void USB_DMA_Disable(uint32_t EPNum);
extern uint32_t USB_DMA_Status (uint32_t
    EPNum);
extern uint32_t USB_DMA_BufAdr (uint32_t
    EPNum);
extern uint32_t USB_DMA_BufCnt (uint32_t
    EPNum);
extern uint32_t USB_GetFrame (void);
extern void USB_IRQHandler (void);


#endif /* __USBHW_H__ */
```

I. usbcore.c

```
/*----------------------------------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------------------------------
 * Name: usbcore.c
 * Purpose: USB Core Module
 * Version: V1.20
 *----------------------------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------------------------------
 * History:
 *       V1.20 Added vendor specific requests
 *           Changed string descriptor
    handling
 *           Reworked Endpoint0
 *       V1.00 Initial Version
 *----------------------------------------------------------------------------*/

#include "type.h"

#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbdesc.h"
#include "usbuser.h"

#if (USB_CLASS)

#if (USB_AUDIO)
#include "audio.h"
#include "adcuser.h"
#endif

#if (USB_HID)
#include "hid.h"
#include "hiduser.h"
#endif

#if (USB_MSC)
#include "msc.h"
#include "mscuser.h"
extern MSC_CSW CSW;
#endif

#if (USB_CDC)
#include "cdc.h"
#include "cdcuser.h"
#endif

#endif

#if (USB_VENDOR)
#include "vendor.h"
#endif

#pragma diag_suppress 111,177,1441

uint16_t USB_DeviceStatus;
uint8_t USB_DeviceAddress;
```

```
uint8_t USB_Configuration;
uint32_t USB_EndPointMask;
uint32_t USB_EndPointHalt;
uint32_t USB_EndPointStall;             /* EP
    must stay stalled */
uint8_t USB_NumInterfaces;
uint8_t USB_AltSetting[USB_IF_NUM];

uint8_t EP0Buf[USB_MAX_PACKET0];


USB_EP_DATA EP0Data;

USB_SETUP_PACKET SetupPacket;


/*
 * Reset USB Core
 *   Parameters:  None
 *   Return Value: None
 */

void USB_ResetCore (void) {

  USB_DeviceStatus = USB_POWER;
  USB_DeviceAddress = 0;
  USB_Configuration = 0;
  USB_EndPointMask = 0x00010001;
  USB_EndPointHalt = 0x00000000;
  USB_EndPointStall = 0x00000000;
}


/*
 * USB Request - Setup Stage
 *   Parameters:  None (global SetupPacket)
 *   Return Value: None
 */

void USB_SetupStage (void) {
  USB_ReadEP(0x00, (uint8_t *)&SetupPacket);
}


/*
 * USB Request - Data In Stage
 *   Parameters:  None (global EP0Data)
 *   Return Value: None
 */

void USB_DataInStage (void) {
  uint32_t cnt;

  if (EP0Data.Count > USB_MAX_PACKET0) {
    cnt = USB_MAX_PACKET0;
  } else {
    cnt = EP0Data.Count;
  }
  cnt = USB_WriteEP(0x80, EP0Data.pData, cnt);
  EP0Data.pData += cnt;
  EP0Data.Count -= cnt;
}


/*
 * USB Request - Data Out Stage
 *   Parameters:  None (global EP0Data)
 *   Return Value: None
 */

void USB_DataOutStage (void) {
  uint32_t cnt;

  cnt = USB_ReadEP(0x00, EP0Data.pData);
  EP0Data.pData += cnt;
  EP0Data.Count -= cnt;
}


/*
 * USB Request - Status In Stage
 *   Parameters:  None
 *   Return Value: None
 */

void USB_StatusInStage (void) {
  USB_WriteEP(0x80, NULL, 0);
}


/*
 * USB Request - Status Out Stage
 *   Parameters:  None
 *   Return Value: None
 */

void USB_StatusOutStage (void) {
  USB_ReadEP(0x00, EP0Buf);
}


/*
 * Get Status USB Request
 *   Parameters:  None (global SetupPacket)
 *   Return Value: TRUE - Success, FALSE -
    Error
 */

uint32_t USB_ReqGetStatus (void) {
  uint32_t n, m;

  switch
      (SetupPacket.bmRequestType.BM.Recipient)
      {
    case REQUEST_TO_DEVICE:
      EP0Data.pData = (uint8_t
          *)&USB_DeviceStatus;
      break;
    case REQUEST_TO_INTERFACE:
      if ((USB_Configuration != 0) &&
          (SetupPacket.wIndex.WB.L <
          USB_NumInterfaces)) {
        *((__packed uint16_t *)EP0Buf) = 0;
        EP0Data.pData = EP0Buf;
      } else {
        return (FALSE);
      }
      break;
    case REQUEST_TO_ENDPOINT:
      n = SetupPacket.wIndex.WB.L & 0x8F;
      m = (n & 0x80) ? ((1 << 16) << (n &
          0x0F)) : (1 << n);
      if (((USB_Configuration != 0) || ((n &
          0x0F) == 0)) && (USB_EndPointMask &
```

```
        m)) {
      *((__packed uint16_t *)EP0Buf) =
          (USB_EndPointHalt & m) ? 1 : 0;
      EP0Data.pData = EP0Buf;
     } else {
      return (FALSE);
     }
     break;
   default:
     return (FALSE);
  }
  return (TRUE);
}


/*
 * Set/Clear Feature USB Request
 *   Parameters:  sc:  0 - Clear, 1 - Set
 *                     (global SetupPacket)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t USB_ReqSetClrFeature (uint32_t sc) {
 uint32_t n, m;

 switch
     (SetupPacket.bmRequestType.BM.Recipient)
     {
   case REQUEST_TO_DEVICE:
     if (SetupPacket.wValue.W ==
         USB_FEATURE_REMOTE_WAKEUP) {
       if (sc) {
         USB_WakeUpCfg(TRUE);
         USB_DeviceStatus |=
             USB_GETSTATUS_REMOTE_WAKEUP;
       } else {
         USB_WakeUpCfg(FALSE);
         USB_DeviceStatus &=
             ~USB_GETSTATUS_REMOTE_WAKEUP;
       }
     } else {
       return (FALSE);
     }
     break;
   case REQUEST_TO_INTERFACE:
     return (FALSE);
   case REQUEST_TO_ENDPOINT:
     n = SetupPacket.wIndex.WB.L & 0x8F;
     m = (n & 0x80) ? ((1 << 16) << (n &
         0x0F)) : (1 << n);
     if ((USB_Configuration != 0) && ((n &
         0x0F) != 0) && (USB_EndPointMask &
         m)) {
       if (SetupPacket.wValue.W ==
           USB_FEATURE_ENDPOINT_STALL) {
         if (sc) {
           USB_SetStallEP(n);
           USB_EndPointHalt |= m;
         } else {
           if ((USB_EndPointStall & m) != 0) {
             return (TRUE);
           }
           USB_ClrStallEP(n);
#if (USB_MSC)
           if ((n == MSC_EP_IN) &&
               ((USB_EndPointHalt & m) != 0)) {
             /* Compliance Test: rewrite CSW
                after unstall */
             if (CSW.dSignature ==
                 MSC_CSW_Signature) {
               USB_WriteEP(MSC_EP_IN, (uint8_t
                   *)&CSW, sizeof(CSW));
             }
           }
#endif
           USB_EndPointHalt &= ~m;
         }
       } else {
         return (FALSE);
       }
     } else {
       return (FALSE);
     }
     break;
   default:
     return (FALSE);
  }
  return (TRUE);
}


/*
 * Set Address USB Request
 *   Parameters:  None (global SetupPacket)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t USB_ReqSetAddress (void) {

 switch
     (SetupPacket.bmRequestType.BM.Recipient)
     {
   case REQUEST_TO_DEVICE:
     USB_DeviceAddress = 0x80 |
         SetupPacket.wValue.WB.L;
     break;
   default:
     return (FALSE);
  }
  return (TRUE);
}


/*
 * Get Descriptor USB Request
 *   Parameters:  None (global SetupPacket)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t USB_ReqGetDescriptor (void) {
 uint8_t *pD;
 uint32_t len, n;

 switch
     (SetupPacket.bmRequestType.BM.Recipient)
     {
   case REQUEST_TO_DEVICE:
     switch (SetupPacket.wValue.WB.H) {
       case USB_DEVICE_DESCRIPTOR_TYPE:
         EP0Data.pData = (uint8_t
             *)USB_DeviceDescriptor;
```

```
      len = USB_DEVICE_DESC_SIZE;
      break;
    case USB_CONFIGURATION_DESCRIPTOR_TYPE:
      pD = (uint8_t *)USB_ConfigDescriptor;
      for (n = 0; n !=
          SetupPacket.wValue.WB.L; n++) {
        if (((USB_CONFIGURATION_DESCRIPTOR
            *)pD)->bLength != 0) {
          pD +=
              ((USB_CONFIGURATION_DESCRIPTOR
              *)pD)->wTotalLength;
        }
      }
      if (((USB_CONFIGURATION_DESCRIPTOR
          *)pD)->bLength == 0) {
        return (FALSE);
      }
      EP0Data.pData = pD;
      len = ((USB_CONFIGURATION_DESCRIPTOR
          *)pD)->wTotalLength;
      break;
    case USB_STRING_DESCRIPTOR_TYPE:
      pD = (uint8_t *)USB_StringDescriptor;
      for (n = 0; n !=
          SetupPacket.wValue.WB.L; n++) {
        if (((USB_STRING_DESCRIPTOR
            *)pD)->bLength != 0) {
          pD += ((USB_STRING_DESCRIPTOR
              *)pD)->bLength;
        }
      }
      if (((USB_STRING_DESCRIPTOR
          *)pD)->bLength == 0) {
        return (FALSE);
      }
      EP0Data.pData = pD;
    len = ((USB_STRING_DESCRIPTOR
        *)EP0Data.pData)->bLength;
      break;
    default:
      return (FALSE);
    }
    break;
  case REQUEST_TO_INTERFACE:
    switch (SetupPacket.wValue.WB.H) {
#if USB_HID
    case HID_HID_DESCRIPTOR_TYPE:
      if (SetupPacket.wIndex.WB.L !=
          USB_HID_IF_NUM) {
        return (FALSE); /* Only Single HID
            Interface is supported */
      }
      EP0Data.pData = (uint8_t
          *)USB_ConfigDescriptor +
          HID_DESC_OFFSET;
      len = HID_DESC_SIZE;
      break;
    case HID_REPORT_DESCRIPTOR_TYPE:
      if (SetupPacket.wIndex.WB.L !=
          USB_HID_IF_NUM) {
        return (FALSE); /* Only Single HID
            Interface is supported */
      }
      EP0Data.pData = (uint8_t
          *)HID_ReportDescriptor;
      len = HID_ReportDescSize;
      break;
    case HID_PHYSICAL_DESCRIPTOR_TYPE:
      return (FALSE); /* HID Physical
          Descriptor is not supported */
#endif
    default:
      return (FALSE);
    }
    break;
  default:
    return (FALSE);
  }

  if (EP0Data.Count > len) {
    EP0Data.Count = len;
  }

  return (TRUE);
}


/*
 * Get Configuration USB Request
 *   Parameters:  None (global SetupPacket)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t USB_ReqGetConfiguration (void) {

  switch
      (SetupPacket.bmRequestType.BM.Recipient)
      {
    case REQUEST_TO_DEVICE:
      EP0Data.pData = &USB_Configuration;
      break;
    default:
      return (FALSE);
  }
  return (TRUE);
}


/*
 * Set Configuration USB Request
 *   Parameters:  None (global SetupPacket)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t USB_ReqSetConfiguration (void) {
  USB_COMMON_DESCRIPTOR *pD;
  uint32_t alt = 0;
  uint32_t n, m;

  switch
      (SetupPacket.bmRequestType.BM.Recipient)
      {
    case REQUEST_TO_DEVICE:

      if (SetupPacket.wValue.WB.L) {
        pD = (USB_COMMON_DESCRIPTOR
            *)USB_ConfigDescriptor;
        while (pD->bLength) {
          switch (pD->bDescriptorType) {
            case
                USB_CONFIGURATION_DESCRIPTOR_TYPE:
```

```
      if (((USB_CONFIGURATION_DESCRIPTOR
          *)pD)->bConfigurationValue ==
          SetupPacket.wValue.WB.L) {
        USB_Configuration =
            SetupPacket.wValue.WB.L;
        USB_NumInterfaces =
            ((USB_CONFIGURATION_DESCRIPTOR
            *)pD)->bNumInterfaces;
        for (n = 0; n < USB_IF_NUM; n++) {
          USB_AltSetting[n] = 0;
        }
        for (n = 1; n < 16; n++) {
          if (USB_EndPointMask & (1 << n))
              {
            USB_DisableEP(n);
          }
          if (USB_EndPointMask & ((1 <<
              16) << n)) {
            USB_DisableEP(n | 0x80);
          }
        }
        USB_EndPointMask = 0x00010001;
        USB_EndPointHalt = 0x00000000;
        USB_EndPointStall= 0x00000000;
        USB_Configure(TRUE);
        if
            (((USB_CONFIGURATION_DESCRIPTOR
            *)pD)->bmAttributes &
            USB_CONFIG_POWERED_MASK) {
          USB_DeviceStatus |=
              USB_GETSTATUS_SELF_POWERED;
        } else {
          USB_DeviceStatus &=
              ~USB_GETSTATUS_SELF_POWERED;
        }
      } else {
        (uint8_t *)pD +=
            ((USB_CONFIGURATION_DESCRIPTOR
            *)pD)->wTotalLength;
        continue;
      }
      break;
    case USB_INTERFACE_DESCRIPTOR_TYPE:
      alt = ((USB_INTERFACE_DESCRIPTOR
          *)pD)->bAlternateSetting;
      break;
    case USB_ENDPOINT_DESCRIPTOR_TYPE:
      if (alt == 0) {
        n = ((USB_ENDPOINT_DESCRIPTOR
            *)pD)->bEndpointAddress &
            0x8F;
        m = (n & 0x80) ? ((1 << 16) << (n
            & 0x0F)) : (1 << n);
        USB_EndPointMask |= m;
        USB_ConfigEP((USB_ENDPOINT_DESCRIPTOR
            *)pD);
        USB_EnableEP(n);
        USB_ResetEP(n);
      }
      break;
    }
    (uint8_t *)pD += pD->bLength;
  }
}
else {
  USB_Configuration = 0;
  for (n = 1; n < 16; n++) {
    if (USB_EndPointMask & (1 << n)) {
      USB_DisableEP(n);
    }
    if (USB_EndPointMask & ((1 << 16) <<
        n)) {
      USB_DisableEP(n | 0x80);
    }
  }
  USB_EndPointMask = 0x00010001;
  USB_EndPointHalt = 0x00000000;
  USB_EndPointStall = 0x00000000;
  USB_Configure(FALSE);
}

if (USB_Configuration !=
    SetupPacket.wValue.WB.L) {
  return (FALSE);
}
break;
default:
  return (FALSE);
}
return (TRUE);
}

/*
* Get Interface USB Request
*   Parameters:  None (global SetupPacket)
*   Return Value: TRUE - Success, FALSE -
    Error
*/

uint32_t USB_ReqGetInterface (void) {

 switch
     (SetupPacket.bmRequestType.BM.Recipient)
     {
  case REQUEST_TO_INTERFACE:
    if ((USB_Configuration != 0) &&
        (SetupPacket.wIndex.WB.L <
        USB_NumInterfaces)) {
      EP0Data.pData = USB_AltSetting +
          SetupPacket.wIndex.WB.L;
    } else {
      return (FALSE);
    }
    break;
  default:
    return (FALSE);
 }
 return (TRUE);
}

/*
* Set Interface USB Request
*   Parameters:  None (global SetupPacket)
*   Return Value: TRUE - Success, FALSE -
    Error
*/

uint32_t USB_ReqSetInterface (void) {
 USB_COMMON_DESCRIPTOR *pD;
 uint32_t ifn = 0, alt = 0, old = 0, msk = 0;
 uint32_t n, m;
 uint32_t set;
```

```
switch
    (SetupPacket.bmRequestType.BM.Recipient)
    {
  case REQUEST_TO_INTERFACE:
    if (USB_Configuration == 0) return
        (FALSE);
    set = FALSE;
    pD = (USB_COMMON_DESCRIPTOR
        *)USB_ConfigDescriptor;
    while (pD->bLength) {
      switch (pD->bDescriptorType) {
        case USB_CONFIGURATION_DESCRIPTOR_TYPE:
          if (((USB_CONFIGURATION_DESCRIPTOR
              *)pD)->bConfigurationValue !=
              USB_Configuration) {
            (uint8_t *)pD +=
                ((USB_CONFIGURATION_DESCRIPTOR
                *)pD)->wTotalLength;
            continue;
          }
          break;
        case USB_INTERFACE_DESCRIPTOR_TYPE:
          ifn = ((USB_INTERFACE_DESCRIPTOR
              *)pD)->bInterfaceNumber;
          alt = ((USB_INTERFACE_DESCRIPTOR
              *)pD)->bAlternateSetting;
          msk = 0;
          if ((ifn == SetupPacket.wIndex.WB.L)
              && (alt ==
              SetupPacket.wValue.WB.L)) {
            set = TRUE;
            old = USB_AltSetting[ifn];
            USB_AltSetting[ifn] = (uint8_t)alt;
          }
          break;
        case USB_ENDPOINT_DESCRIPTOR_TYPE:
          if (ifn == SetupPacket.wIndex.WB.L) {
            n = ((USB_ENDPOINT_DESCRIPTOR
                *)pD)->bEndpointAddress & 0x8F;
            m = (n & 0x80) ? ((1 << 16) << (n &
                0x0F)) : (1 << n);
            if (alt == SetupPacket.wValue.WB.L)
                {
              USB_EndPointMask |= m;
              USB_EndPointHalt &= ~m;
              USB_ConfigEP((USB_ENDPOINT_DESCRIPTOR
                  *)pD);
              USB_EnableEP(n);
              USB_ResetEP(n);
              msk |= m;
            }
            else if ((alt == old) && ((msk & m)
                == 0)) {
              USB_EndPointMask &= ~m;
              USB_EndPointHalt &= ~m;
              USB_DisableEP(n);
            }
          }
          break;
      }
      (uint8_t *)pD += pD->bLength;
    }
    break;
  default:
    return (FALSE);
}
  return (set);
}

/*
 * USB Endpoint 0 Event Callback
 *   Parameters:  event
 *   Return Value: none
 */

void USB_EndPoint0 (uint32_t event) {

  switch (event) {
    case USB_EVT_SETUP:
      USB_SetupStage();
      USB_DirCtrlEP(SetupPacket.bmRequestType.BM.Dir);
      EP0Data.Count = SetupPacket.wLength; /*
          Number of bytes to transfer */
      switch
          (SetupPacket.bmRequestType.BM.Type) {

        case REQUEST_STANDARD:
          switch (SetupPacket.bRequest) {
            case USB_REQUEST_GET_STATUS:
              if (!USB_ReqGetStatus()) {
                goto stall_i;
              }
              USB_DataInStage();
              break;

            case USB_REQUEST_CLEAR_FEATURE:
              if (!USB_ReqSetClrFeature(0)) {
                goto stall_i;
              }
              USB_StatusInStage();
#if USB_FEATURE_EVENT
              USB_Feature_Event();
#endif
              break;

            case USB_REQUEST_SET_FEATURE:
              if (!USB_ReqSetClrFeature(1)) {
                goto stall_i;
              }
              USB_StatusInStage();
#if USB_FEATURE_EVENT
              USB_Feature_Event();
#endif
              break;

            case USB_REQUEST_SET_ADDRESS:
              if (!USB_ReqSetAddress()) {
                goto stall_i;
              }
              USB_StatusInStage();
              break;

            case USB_REQUEST_GET_DESCRIPTOR:
              if (!USB_ReqGetDescriptor()) {
                goto stall_i;
              }
              USB_DataInStage();
              break;

            case USB_REQUEST_SET_DESCRIPTOR:
```

```
/*stall_o:*/ USB_SetStallEP(0x00); /* not
    supported */
          EP0Data.Count = 0;
          break;

        case USB_REQUEST_GET_CONFIGURATION:
          if (!USB_ReqGetConfiguration()) {
            goto stall_i;
          }
          USB_DataInStage();
          break;

        case USB_REQUEST_SET_CONFIGURATION:
          if (!USB_ReqSetConfiguration()) {
            goto stall_i;
          }
          USB_StatusInStage();
#if USB_CONFIGURE_EVENT
          USB_Configure_Event();
#endif
          break;

        case USB_REQUEST_GET_INTERFACE:
          if (!USB_ReqGetInterface()) {
            goto stall_i;
          }
          USB_DataInStage();
          break;

        case USB_REQUEST_SET_INTERFACE:
          if (!USB_ReqSetInterface()) {
            goto stall_i;
          }
          USB_StatusInStage();
#if USB_INTERFACE_EVENT
          USB_Interface_Event();
#endif
          break;

        default:
          goto stall_i;
      }
      break; /* end case REQUEST_STANDARD */

#if USB_CLASS
    case REQUEST_CLASS:
      switch
          (SetupPacket.bmRequestType.BM.Recipient)
          {

        case REQUEST_TO_DEVICE:
          goto stall_i;

            /* not supported */

        case REQUEST_TO_INTERFACE:
#if USB_HID
          if (SetupPacket.wIndex.WB.L ==
              USB_HID_IF_NUM) { /* IF number
              correct? */
            switch (SetupPacket.bRequest) {
              case HID_REQUEST_GET_REPORT:
                if (HID_GetReport()) {
                  EP0Data.pData = EP0Buf;
                                  /* point to
                      data to be sent */
                  USB_DataInStage();
                                  /*
                      send requested data */
                  goto setup_class_ok;
                }
                break;
              case HID_REQUEST_SET_REPORT:
                EP0Data.pData = EP0Buf;
                                  /* data to
                    be received */
                goto setup_class_ok;
              case HID_REQUEST_GET_IDLE:
                if (HID_GetIdle()) {
                  EP0Data.pData = EP0Buf;
                                  /* point to
                      data to be sent */
                  USB_DataInStage();
                                  /*
                      send requested data */
                  goto setup_class_ok;
                }
                break;
              case HID_REQUEST_SET_IDLE:
                if (HID_SetIdle()) {
                  USB_StatusInStage();
                                  /* send
                      Acknowledge */
                  goto setup_class_ok;
                }
                break;
              case HID_REQUEST_GET_PROTOCOL:
                if (HID_GetProtocol()) {
                  EP0Data.pData = EP0Buf;
                                  /* point to
                      data to be sent */
                  USB_DataInStage();
                                  /*
                      send requested data */
                  goto setup_class_ok;
                }
                break;
              case HID_REQUEST_SET_PROTOCOL:
                if (HID_SetProtocol()) {
                  USB_StatusInStage();
                                  /* send
                      Acknowledge */
                  goto setup_class_ok;
                }
                break;
            }
          }
#endif /* USB_HID */
#if USB_MSC
          if (SetupPacket.wIndex.WB.L ==
              USB_MSC_IF_NUM) { /* IF number
              correct? */
            switch (SetupPacket.bRequest) {
              case MSC_REQUEST_RESET:
                if ((SetupPacket.wValue.W ==
                    0) &&       /* RESET with
                    invalid parameters ->
                    STALL */
                    (SetupPacket.wLength == 0))
                      {
                  if (MSC_Reset()) {
                    USB_StatusInStage();
                    goto setup_class_ok;
```

```
                }
              }
              break;
            case MSC_REQUEST_GET_MAX_LUN:
              if ((SetupPacket.wValue.W ==
                  0) &&       /* GET_MAX_LUN
                  with invalid parameters ->
                  STALL */
                  (SetupPacket.wLength == 1))
                    {
                if (MSC_GetMaxLUN()) {
                  EP0Data.pData = EP0Buf;
                  USB_DataInStage();
                  goto setup_class_ok;
                }
              }
              break;
          }
        }
#endif /* USB_MSC */
#if USB_AUDIO
        if ((SetupPacket.wIndex.WB.L ==
            USB_ADC_CIF_NUM) || /* IF
            number correct? */
            (SetupPacket.wIndex.WB.L ==
              USB_ADC_SIF1_NUM) ||
            (SetupPacket.wIndex.WB.L ==
              USB_ADC_SIF2_NUM)) {
          switch (SetupPacket.bRequest) {
            case AUDIO_REQUEST_GET_CUR:
            case AUDIO_REQUEST_GET_MIN:
            case AUDIO_REQUEST_GET_MAX:
            case AUDIO_REQUEST_GET_RES:
              if (ADC_IF_GetRequest()) {
                EP0Data.pData = EP0Buf;
                                /* point to
                  data to be sent */
                USB_DataInStage();
                                /*
                  send requested data */
                goto setup_class_ok;
              }
              break;
            case AUDIO_REQUEST_SET_CUR:
//            case AUDIO_REQUEST_SET_MIN:
//            case AUDIO_REQUEST_SET_MAX:
//            case AUDIO_REQUEST_SET_RES:
              EP0Data.pData = EP0Buf;
                                /* data to
                  be received */
              goto setup_class_ok;
          }
        }
#endif /* USB_AUDIO */
#if USB_CDC
        if ((SetupPacket.wIndex.WB.L ==
            USB_CDC_CIF_NUM) || /* IF
            number correct? */
            (SetupPacket.wIndex.WB.L ==
              USB_CDC_DIF_NUM)) {
          switch (SetupPacket.bRequest) {
            case
              CDC_SEND_ENCAPSULATED_COMMAND:
              EP0Data.pData = EP0Buf;
                                /* data to
                  be received, see
                  USB_EVT_OUT */
              goto setup_class_ok;
            case
              CDC_GET_ENCAPSULATED_RESPONSE:
              if
                (CDC_GetEncapsulatedResponse())
                {
                EP0Data.pData = EP0Buf;
                                /* point to
                  data to be sent */
                USB_DataInStage();
                                /*
                  send requested data */
                goto setup_class_ok;
              }
              break;
            case CDC_SET_COMM_FEATURE:
              EP0Data.pData = EP0Buf;
                                /* data to
                  be received, see
                  USB_EVT_OUT */
              goto setup_class_ok;
            case CDC_GET_COMM_FEATURE:
              if
                (CDC_GetCommFeature(SetupPacket.wValu
                {
                EP0Data.pData = EP0Buf;
                                /* point to
                  data to be sent */
                USB_DataInStage();
                                /*
                  send requested data */
                goto setup_class_ok;
              }
              break;
            case CDC_CLEAR_COMM_FEATURE:
              if
                (CDC_ClearCommFeature(SetupPacket.wVa
                {
                USB_StatusInStage();
                                /* send
                  Acknowledge */
                goto setup_class_ok;
              }
              break;
            case CDC_SET_LINE_CODING:
              EP0Data.pData = EP0Buf;
                                /* data to
                  be received, see
                  USB_EVT_OUT */
              goto setup_class_ok;
            case CDC_GET_LINE_CODING:
              if (CDC_GetLineCoding()) {
                EP0Data.pData = EP0Buf;
                                /* point to
                  data to be sent */
                USB_DataInStage();
                                /*
                  send requested data */
                goto setup_class_ok;
              }
              break;
            case CDC_SET_CONTROL_LINE_STATE:
              if
                (CDC_SetControlLineState(SetupPacket.
                {
                USB_StatusInStage();
                                /* send
```

```
                    Acknowledge */
                  goto setup_class_ok;
                }
                break;
              case CDC_SEND_BREAK:
                if
                    (CDC_SendBreak(SetupPacket.wValue.W))
                    {
                  USB_StatusInStage();
                                    /* send
                      Acknowledge */
                  goto setup_class_ok;
                }
                break;
            }
          }
#endif /* USB_CDC */
          goto stall_i;

              /* not supported */
          /* end case REQUEST_TO_INTERFACE */

        case REQUEST_TO_ENDPOINT:
#if USB_AUDIO
          switch (SetupPacket.bRequest) {
            case AUDIO_REQUEST_GET_CUR:
            case AUDIO_REQUEST_GET_MIN:
            case AUDIO_REQUEST_GET_MAX:
            case AUDIO_REQUEST_GET_RES:
              if (ADC_EP_GetRequest()) {
                EP0Data.pData = EP0Buf;
                                /* point to
                    data to be sent */
                USB_DataInStage();
                                    /*
                    send requested data */
                goto setup_class_ok;
              }
              break;
            case AUDIO_REQUEST_SET_CUR:
//          case AUDIO_REQUEST_SET_MIN:
//          case AUDIO_REQUEST_SET_MAX:
//          case AUDIO_REQUEST_SET_RES:
              EP0Data.pData = EP0Buf;
                                /* data to
                  be received */
              goto setup_class_ok;
          }
#endif /* USB_AUDIO */
          goto stall_i;
          /* end case REQUEST_TO_ENDPOINT */

        default:
          goto stall_i;
      }
setup_class_ok:
                                        /*
   request finished successfully */
      break; /* end case REQUEST_CLASS */
#endif /* USB_CLASS */

#if USB_VENDOR
    case REQUEST_VENDOR:
      switch
          (SetupPacket.bmRequestType.BM.Recipient)
          {
        case REQUEST_TO_DEVICE:
          if (!USB_ReqVendorDev(TRUE)) {
            goto stall_i;

                /* not supported */
          }
          break;

        case REQUEST_TO_INTERFACE:
          if (!USB_ReqVendorIF(TRUE)) {
            goto stall_i;

                /* not supported */
          }
          break;

        case REQUEST_TO_ENDPOINT:
          if (!USB_ReqVendorEP(TRUE)) {
            goto stall_i;

                /* not supported */
          }
          break;

        default:
          goto stall_i;
      }

      if (SetupPacket.wLength) {
        if (SetupPacket.bmRequestType.BM.Dir
            == REQUEST_DEVICE_TO_HOST) {
          USB_DataInStage();
        }
      } else {
        USB_StatusInStage();
      }

      break; /* end case REQUEST_VENDOR */
#endif /* USB_VENDOR */

    default:
stall_i: USB_SetStallEP(0x80);
      EP0Data.Count = 0;
      break;
  }
  break; /* end case USB_EVT_SETUP */

case USB_EVT_OUT:
  if (SetupPacket.bmRequestType.BM.Dir ==
      REQUEST_HOST_TO_DEVICE) {
    if (EP0Data.Count) {
                                        /* still
      data to receive ? */
      USB_DataOutStage();
                                         /*
      receive data */
      if (EP0Data.Count == 0) {
                                /* data
        complete ? */
        switch
            (SetupPacket.bmRequestType.BM.Type)
            {

        case REQUEST_STANDARD:
          goto stall_i;

              /* not supported */
```

```
#if (USB_CLASS)
          case REQUEST_CLASS:
            switch
                (SetupPacket.bmRequestType.BM.Recipient)
                {
              case REQUEST_TO_DEVICE:
                goto stall_i;

                  /* not supported */

              case REQUEST_TO_INTERFACE:
#if USB_HID
                if (SetupPacket.wIndex.WB.L ==
                    USB_HID_IF_NUM) { /* IF
                    number correct? */
                  switch (SetupPacket.bRequest)
                      {
                    case HID_REQUEST_SET_REPORT:
                      if (HID_SetReport()) {
                        USB_StatusInStage();
                                        /* send
                            Acknowledge */
                        goto out_class_ok;
                      }
                      break;
                  }
                }
#endif /* USB_HID */
#if USB_AUDIO
                if ((SetupPacket.wIndex.WB.L
                    == USB_ADC_CIF_NUM) || /*
                    IF number correct? */
                    (SetupPacket.wIndex.WB.L ==
                       USB_ADC_SIF1_NUM) ||
                    (SetupPacket.wIndex.WB.L ==
                       USB_ADC_SIF2_NUM)) {
                  switch (SetupPacket.bRequest)
                      {
                    case AUDIO_REQUEST_SET_CUR:
//                  case AUDIO_REQUEST_SET_MIN:
//                  case AUDIO_REQUEST_SET_MAX:
//                  case AUDIO_REQUEST_SET_RES:
                      if (ADC_IF_SetRequest()) {
                        USB_StatusInStage();
                                        /* send
                            Acknowledge */
                        goto out_class_ok;
                      }
                      break;
                  }
                }
#endif /* USB_AUDIO */
#if USB_CDC
                if ((SetupPacket.wIndex.WB.L
                    == USB_CDC_CIF_NUM) || /*
                    IF number correct? */
                    (SetupPacket.wIndex.WB.L ==
                       USB_CDC_DIF_NUM)) {
                  switch (SetupPacket.bRequest)
                      {
                    case
                        CDC_SEND_ENCAPSULATED_COMMAND:
                      if
                          (CDC_SendEncapsulatedCommand())
                          {
                        USB_StatusInStage();
                                        /* send
                            Acknowledge */
                        goto out_class_ok;
                      }
                      break;
                    case CDC_SET_COMM_FEATURE:
                      if
                          (CDC_SetCommFeature(SetupPacket.w
                          {
                        USB_StatusInStage();
                                        /* send
                            Acknowledge */
                        goto out_class_ok;
                      }
                      break;
                    case CDC_SET_LINE_CODING:
                      if (CDC_SetLineCoding()) {
                        USB_StatusInStage();
                                        /* send
                            Acknowledge */
                        goto out_class_ok;
                      }
                      break;
                  }
                }
#endif /* USB_CDC */
                goto stall_i;
                /* end case
                    REQUEST_TO_INTERFACE */

              case REQUEST_TO_ENDPOINT:
#if USB_AUDIO
                switch (SetupPacket.bRequest) {
                  case AUDIO_REQUEST_SET_CUR:
//                case AUDIO_REQUEST_SET_MIN:
//                case AUDIO_REQUEST_SET_MAX:
//                case AUDIO_REQUEST_SET_RES:
                    if (ADC_EP_SetRequest()) {
                      USB_StatusInStage();
                                        /* send
                          Acknowledge */
                      goto out_class_ok;
                    }
                    break;
                }
#endif /* USB_AUDIO */
                goto stall_i;
                /* end case
                    REQUEST_TO_ENDPOINT */

              default:
                goto stall_i;
            }
out_class_ok:

    /* request finished successfully */
            break; /* end case REQUEST_CLASS
                */
#endif /* USB_CLASS */

#if USB_VENDOR
          case REQUEST_VENDOR:
            switch
                (SetupPacket.bmRequestType.BM.Recipient)
                {
```

```
            case REQUEST_TO_DEVICE:
              if (!USB_ReqVendorDev(FALSE)) {
                goto stall_i;

                  /* not supported */
              }
              break;

            case REQUEST_TO_INTERFACE:
              if (!USB_ReqVendorIF(FALSE)) {
                goto stall_i;

                  /* not supported */
              }
              break;

            case REQUEST_TO_ENDPOINT:
              if (!USB_ReqVendorEP(FALSE)) {
                goto stall_i;

                  /* not supported */
              }
              break;

            default:
              goto stall_i;
            }

            USB_StatusInStage();

            break; /* end case REQUEST_VENDOR
                */
#endif /* USB_VENDOR */

          default:
            goto stall_i;
          }
        }
      }
    } else {
      USB_StatusOutStage();
                                    /*
        receive Acknowledge */
    }
    break; /* end case USB_EVT_OUT */

  case USB_EVT_IN :
    if (SetupPacket.bmRequestType.BM.Dir ==
        REQUEST_DEVICE_TO_HOST) {
      USB_DataInStage();
                                      /*
        send data */
    } else {
      if (USB_DeviceAddress & 0x80) {
        USB_DeviceAddress &= 0x7F;
        USB_SetAddress(USB_DeviceAddress);
      }
    }
    break; /* end case USB_EVT_IN */

  case USB_EVT_OUT_STALL:
    USB_ClrStallEP(0x00);
    break;

  case USB_EVT_IN_STALL:
    USB_ClrStallEP(0x80);
    break;
  }
}
```

J. usbcore.h

```
/*----------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------
 * Name: usbcore.h
 * Purpose: USB Core Definitions
 * Version: V1.20
 *----------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------

#ifndef __USBCORE_H__
#define __USBCORE_H__


/* USB Endpoint Data Structure */
typedef struct _USB_EP_DATA {
  uint8_t *pData;
  uint16_t Count;
} USB_EP_DATA;

/* USB Core Global Variables */
extern uint16_t USB_DeviceStatus;
extern uint8_t USB_DeviceAddress;
extern uint8_t USB_Configuration;
extern uint32_t USB_EndPointMask;
extern uint32_t USB_EndPointHalt;
extern uint32_t USB_EndPointStall;
extern uint8_t USB_AltSetting[USB_IF_NUM];

/* USB Endpoint 0 Buffer */
extern uint8_t EP0Buf[USB_MAX_PACKET0];

/* USB Endpoint 0 Data Info */
extern USB_EP_DATA EP0Data;

/* USB Setup Packet */
extern USB_SETUP_PACKET SetupPacket;

/* USB Core Functions */
extern void USB_ResetCore (void);
```

```
#endif /* __USBCORE_H__ */
```

K. usbuser.c

```
/*----------------------------------------------------------------------------
 *      U S B - K e r n e l
 *----------------------------------------------------------------------------
 * Name: usbuser.c
 * Purpose: USB Custom User Module
 * Version: V1.20
 *----------------------------------------------------------------------------
 *      This software is supplied "AS IS" without any warranties, express,
 *      implied or statutory, including but not limited to the implied
 *      warranties of fitness for purpose, satisfactory quality and
 *      noninfringement. Keil extends you a royalty-free right to reproduce
 *      and distribute executable files created using this software for use
 *      on NXP Semiconductors LPC family microcontroller devices only. Nothing
 *      else gives you the right to use this software.
 *
 * Copyright (c) 2009 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------*/

#include "type.h"

#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbuser.h"

#include "usbaudio.h"


/*
 * USB Power Event Callback
 *  Called automatically on USB Power Event
 *   Parameter:   power: On(TRUE)/Off(FALSE)
 */

#if USB_POWER_EVENT
void USB_Power_Event (uint32_t power) {
}
#endif


/*
 * USB Reset Event Callback
 *  Called automatically on USB Reset Event
 */

#if USB_RESET_EVENT
void USB_Reset_Event (void) {
  USB_ResetCore();
}
#endif


/*
 * USB Suspend Event Callback
 *  Called automatically on USB Suspend Event
 */

#if USB_SUSPEND_EVENT
void USB_Suspend_Event (void) {
}
#endif


/*
 * USB Resume Event Callback
 *  Called automatically on USB Resume Event
 */

#if USB_RESUME_EVENT
void USB_Resume_Event (void) {
}
#endif


/*
 * USB Remote Wakeup Event Callback
 *  Called automatically on USB Remote Wakeup Event
 */

#if USB_WAKEUP_EVENT
void USB_WakeUp_Event (void) {
}
#endif


/*
 * USB Start of Frame Event Callback
 *  Called automatically on USB Start of Frame Event
 */

#if USB_SOF_EVENT
void USB_SOF_Event (void) {
#if USB_DMA == 0
  if (USB_ReadEP(0x03, (BYTE *)&DataBuf[DataIn])) {
    /* Data Available */
    DataIn += P_S;                  /* Update Data In Index */
    DataIn &= B_S - 1;              /* Adjust Data In Index */
    if (((DataIn - DataOut) & (B_S - 1)) == (B_S/2)) {
      DataRun = 1;                  /* Data Stream running */
    }
  } else {
    /* No Data */
    DataRun = 0;                    /* Data Stream not running */
    DataOut = DataIn;               /* Initialize Data Indexes */
  }
#endif
}
#endif
```

```
/*
 * USB Error Event Callback
 *  Called automatically on USB Error Event
 *   Parameter:   error: Error Code
 */

#if USB_ERROR_EVENT
void USB_Error_Event (uint32_t error) {
}
#endif


/*
 * USB Set Configuration Event Callback
 *  Called automatically on USB Set
    Configuration Request
 */

#if USB_CONFIGURE_EVENT
void USB_Configure_Event (void) {

  if (USB_Configuration) {        /* Check if
      USB is configured */
    /* add your code here */
  }
}
#endif


/*
 * USB Set Interface Event Callback
 *  Called automatically on USB Set Interface
    Request
 */

#if USB_INTERFACE_EVENT
void USB_Interface_Event (void) {
}
#endif


/*
 * USB Set/Clear Feature Event Callback
 *  Called automatically on USB Set/Clear
    Feature Request
 */

#if USB_FEATURE_EVENT
void USB_Feature_Event (void) {
}
#endif


#define P_EP(n) ((USB_EP_EVENT & (1 << (n)))
    ? USB_EndPoint##n : NULL)

/* USB Endpoint Events Callback Pointers */
void (* const USB_P_EP[16]) (uint32_t event)
    = {
  P_EP(0),
  P_EP(1),
  P_EP(2),
  P_EP(3),
  P_EP(4),
  P_EP(5),
  P_EP(6),
  P_EP(7),
  P_EP(8),
  P_EP(9),
  P_EP(10),
  P_EP(11),
  P_EP(12),
  P_EP(13),
  P_EP(14),
  P_EP(15),
};


/*
 * USB Endpoint 1 Event Callback
 *  Called automatically on USB Endpoint 1
    Event
 *   Parameter:   event
 */

void USB_EndPoint1 (uint32_t event) {
}


/*
 * USB Endpoint 2 Event Callback
 *  Called automatically on USB Endpoint 2
    Event
 *   Parameter:   event
 */

void USB_EndPoint2 (uint32_t event) {
}

/*
 * USB Endpoint 3 Event Callback
 *  Called automatically on USB Endpoint 3
    Event
 *   Parameter:   event
 */

void USB_EndPoint3 (uint32_t event) {
#if USB_DMA
  USB_DMA_DESCRIPTOR DD;

  if (event & USB_EVT_OUT_DMA_EOT) {
    /* End of Transfer */
    if (USB_DMA_BufAdr(0x03) !=
        ((uint32_t)DataBuf + 2*DataIn)) {
      /* Data Available */
      DataIn += P_C*P_S;           /* Update
          Data In Index */
      DataIn &= B_S - 1;           /* Adjust
          Data In Index */
      if (((DataIn - DataOut) & (B_S - 1)) ==
          (B_S/2)) {
        DataRun = 1;               /* Data
            Stream running */
      }
    } else {
      /* No Data */
      DataRun = 0;                 /* Data
          Stream not running */
      DataOut = DataIn;            /*
          Initialize Data Indexes */
    }
  }
  if (event & (USB_EVT_OUT_DMA_EOT) |
      (USB_EVT_OUT_DMA_NDR)) {
```

```
    /* End of Transfer or New Descriptor
       Request */
    DD.BufAdr = (uint32_t)DataBuf + 2*DataIn;
       /* DMA Buffer Address */
    DD.BufLen = P_C;              /* DMA
       Packet Count */
    DD.MaxSize = 0;               /* Must be 0
       for Iso Transfer */
    DD.InfoAdr = (uint32_t)InfoBuf; /* Packet
       Info Buffer Address */
    DD.Cfg.Val = 0;               /* Initial
       DMA Configuration */
    DD.Cfg.Type.IsoEP = 1;        /* Iso
       Endpoint */
    USB_DMA_Setup (0x03, &DD);    /* Setup DMA
       */
    USB_DMA_Enable(0x03);         /* Enable
       DMA */
  }
#else
  event = event;
#endif
}


/*
 * USB Endpoint 4 Event Callback
 *  Called automatically on USB Endpoint 4
    Event
 *  Parameter:   event
 */

void USB_EndPoint4 (uint32_t event) {
}


/*
 * USB Endpoint 5 Event Callback
 *  Called automatically on USB Endpoint 5
    Event
 *  Parameter:   event
 */

void USB_EndPoint5 (uint32_t event) {
}


/*
 * USB Endpoint 6 Event Callback
 *  Called automatically on USB Endpoint 6
    Event
 *  Parameter:   event
 */

void USB_EndPoint6 (uint32_t event) {
}


/*
 * USB Endpoint 7 Event Callback
 *  Called automatically on USB Endpoint 7
    Event
 *  Parameter:   event
 */

void USB_EndPoint7 (uint32_t event) {
}


/*
 * USB Endpoint 8 Event Callback
 *  Called automatically on USB Endpoint 8
    Event
 *  Parameter:   event
 */

void USB_EndPoint8 (uint32_t event) {
}


/*
 * USB Endpoint 9 Event Callback
 *  Called automatically on USB Endpoint 9
    Event
 *  Parameter:   event
 */

void USB_EndPoint9 (uint32_t event) {
}


/*
 * USB Endpoint 10 Event Callback
 *  Called automatically on USB Endpoint 10
    Event
 *  Parameter:   event
 */

void USB_EndPoint10 (uint32_t event) {
}


/*
 * USB Endpoint 11 Event Callback
 *  Called automatically on USB Endpoint 11
    Event
 *  Parameter:   event
 */

void USB_EndPoint11 (uint32_t event) {
}


/*
 * USB Endpoint 12 Event Callback
 *  Called automatically on USB Endpoint 12
    Event
 *  Parameter:   event
 */

void USB_EndPoint12 (uint32_t event) {
}


/*
 * USB Endpoint 13 Event Callback
 *  Called automatically on USB Endpoint 13
    Event
 *  Parameter:   event
 */

void USB_EndPoint13 (uint32_t event) {
}
```

```
/*
 * USB Endpoint 14 Event Callback
 *  Called automatically on USB Endpoint 14
    Event
 *   Parameter:   event
 */

void USB_EndPoint14 (uint32_t event) {
}


/*
 * USB Endpoint 15 Event Callback
 *  Called automatically on USB Endpoint 15
    Event
 *   Parameter:   event
 */

void USB_EndPoint15 (uint32_t event) {
}
```

L. usbuser.h

```
/*------------------------------------------------------------------------------
 *    U S B - K e r n e l
 *------------------------------------------------------------------------------
 * Name: usbuser.h
 * Purpose: USB Custom User Definitions
 * Version: V1.20
 *------------------------------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------------------------------*/

#ifndef __USBUSER_H__
#define __USBUSER_H__


/* USB Device Events Callback Functions */
extern void USB_Power_Event (uint32_t power);
extern void USB_Reset_Event (void);
extern void USB_Suspend_Event (void);
extern void USB_Resume_Event (void);
extern void USB_WakeUp_Event (void);
extern void USB_SOF_Event (void);
extern void USB_Error_Event (uint32_t error);

/* USB Endpoint Callback Events */
#define USB_EVT_SETUP 1 /* Setup Packet */
#define USB_EVT_OUT   2  /* OUT Packet */
#define USB_EVT_IN    3  /* IN Packet */
#define USB_EVT_OUT_NAK 4 /* OUT Packet - Not
    Acknowledged */
#define USB_EVT_IN_NAK 5 /* IN Packet - Not
    Acknowledged */
#define USB_EVT_OUT_STALL 6 /* OUT Packet -
    Stalled */
#define USB_EVT_IN_STALL 7 /* IN Packet -
    Stalled */
#define USB_EVT_OUT_DMA_EOT 8 /* DMA OUT EP -
    End of Transfer */
#define USB_EVT_IN_DMA_EOT 9 /* DMA IN EP -
    End of Transfer */
#define USB_EVT_OUT_DMA_NDR 10 /* DMA OUT EP
    - New Descriptor Request */
#define USB_EVT_IN_DMA_NDR 11 /* DMA IN EP -
    New Descriptor Request */
#define USB_EVT_OUT_DMA_ERR 12 /* DMA OUT EP
    - Error */
#define USB_EVT_IN_DMA_ERR 13 /* DMA IN EP -
    Error */

/* USB Endpoint Events Callback Pointers */
extern void (* const USB_P_EP[16])(uint32_t
    event);

/* USB Endpoint Events Callback Functions */
extern void USB_EndPoint0 (uint32_t event);
extern void USB_EndPoint1 (uint32_t event);
extern void USB_EndPoint2 (uint32_t event);
extern void USB_EndPoint3 (uint32_t event);
extern void USB_EndPoint4 (uint32_t event);
extern void USB_EndPoint5 (uint32_t event);
extern void USB_EndPoint6 (uint32_t event);
extern void USB_EndPoint7 (uint32_t event);
extern void USB_EndPoint8 (uint32_t event);
extern void USB_EndPoint9 (uint32_t event);
extern void USB_EndPoint10 (uint32_t event);
extern void USB_EndPoint11 (uint32_t event);
extern void USB_EndPoint12 (uint32_t event);
extern void USB_EndPoint13 (uint32_t event);
extern void USB_EndPoint14 (uint32_t event);
extern void USB_EndPoint15 (uint32_t event);

/* USB Core Events Callback Functions */
extern void USB_Configure_Event (void);
extern void USB_Interface_Event (void);
extern void USB_Feature_Event (void);

#endif /* __USBUSER_H__ */
```

M. usbdesc.c

```
/*------------------------------------------------------------------------------
 *    U S B - K e r n e l
 *------------------------------------------------------------------------------
 * Name: usbdesc.c
 * Purpose: USB Descriptors
 * Version: V1.20
 *------------------------------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
```

```
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------
 * History:
 *       V1.20 Changed string descriptor
    handling
 *       V1.00 Initial Version
 *----------------------------------------------------*/
#include "type.h"

#include "usb.h"
#include "audio.h"
#include "usbcfg.h"
#include "usbdesc.h"


/* USB Standard Device Descriptor */
const uint8_t USB_DeviceDescriptor[] = {
  USB_DEVICE_DESC_SIZE,      /* bLength */
  USB_DEVICE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  WBVAL(0x0200), /* 2.00 */ /* bcdUSB */
  0x00,                     /* bDeviceClass */
  0x00,                     /*
     bDeviceSubClass */
  0x00,                     /*
     bDeviceProtocol */
  USB_MAX_PACKET0,          /*
     bMaxPacketSize0 */
  WBVAL(0x1FC9),            /* idVendor */
  WBVAL(0x4002),            /* idProduct */
  WBVAL(0x0100), /* 1.00 */ /* bcdDevice */
  0x01,                     /* iManufacturer
     */
  0x02,                     /* iProduct */
  0x03,                     /* iSerialNumber
     */
  0x01                      /*
     bNumConfigurations: one possible
     configuration*/
};

/* USB Configuration Descriptor */
/*  All Descriptors (Configuration,
    Interface, Endpoint, Class, Vendor */
const uint8_t USB_ConfigDescriptor[] = {
/* Configuration 1 */
  USB_CONFIGUARTION_DESC_SIZE, /* bLength */
  USB_CONFIGURATION_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  WBVAL(                        /* wTotalLength
     */
    USB_CONFIGUARTION_DESC_SIZE +
    USB_INTERFACE_DESC_SIZE     +
    AUDIO_CONTROL_INTERFACE_DESC_SZ(1) +
    AUDIO_INPUT_TERMINAL_DESC_SIZE +
    AUDIO_FEATURE_UNIT_DESC_SZ(1,1) +
    AUDIO_OUTPUT_TERMINAL_DESC_SIZE +
    USB_INTERFACE_DESC_SIZE     +
    USB_INTERFACE_DESC_SIZE     +
    AUDIO_STREAMING_INTERFACE_DESC_SIZE +
    AUDIO_FORMAT_TYPE_I_DESC_SZ(1) +
    AUDIO_STANDARD_ENDPOINT_DESC_SIZE +
    AUDIO_STREAMING_ENDPOINT_DESC_SIZE
  ),
  0x02,                         /*
     bNumInterfaces */
  0x01,                         /*
     bConfigurationValue */
  0x00,                         /*
     iConfiguration */
  USB_CONFIG_BUS_POWERED,       /* bmAttributes
     */
  USB_CONFIG_POWER_MA(100),     /* bMaxPower */
/* Interface 0, Alternate Setting 0, Audio
   Control */
  USB_INTERFACE_DESC_SIZE,      /* bLength */
  USB_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  0x00,                         /*
     bInterfaceNumber */
  0x00,                         /*
     bAlternateSetting */
  0x00,                         /*
     bNumEndpoints */
  USB_DEVICE_CLASS_AUDIO,       /*
     bInterfaceClass */
  AUDIO_SUBCLASS_AUDIOCONTROL, /*
     bInterfaceSubClass */
  AUDIO_PROTOCOL_UNDEFINED,     /*
     bInterfaceProtocol */
  0x00,                         /* iInterface */
/* Audio Control Interface */
  AUDIO_CONTROL_INTERFACE_DESC_SZ(1), /*
     bLength */
  AUDIO_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_CONTROL_HEADER,         /*
     bDescriptorSubtype */
  WBVAL(0x0100), /* 1.00 */     /* bcdADC */
  WBVAL(                        /* wTotalLength
     */
    AUDIO_CONTROL_INTERFACE_DESC_SZ(1) +
    AUDIO_INPUT_TERMINAL_DESC_SIZE +
    AUDIO_FEATURE_UNIT_DESC_SZ(1,1) +
    AUDIO_OUTPUT_TERMINAL_DESC_SIZE
  ),
  0x01,                         /*
     bInCollection */
  0x01,                         /*
     baInterfaceNr */
/* Audio Input Terminal */
  AUDIO_INPUT_TERMINAL_DESC_SIZE, /* bLength */
  AUDIO_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_CONTROL_INPUT_TERMINAL, /*
     bDescriptorSubtype */
  0x01,                         /* bTerminalID
     */
  WBVAL(AUDIO_TERMINAL_USB_STREAMING), /*
     wTerminalType */
  0x00,                         /*
     bAssocTerminal */
```

```
  0x01,                         /* bNrChannels
     */
  WBVAL(AUDIO_CHANNEL_M),       /*
     wChannelConfig */
  0x00,                         /*
     iChannelNames */
  0x00,                         /* iTerminal */
/* Audio Feature Unit */
  AUDIO_FEATURE_UNIT_DESC_SZ(1,1), /* bLength
     */
  AUDIO_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_CONTROL_FEATURE_UNIT,   /*
     bDescriptorSubtype */
  0x02,                         /* bUnitID */
  0x01,                         /* bSourceID */
  0x01,                         /* bControlSize
     */
  AUDIO_CONTROL_MUTE |
  AUDIO_CONTROL_VOLUME,         /*
     bmaControls(0) */
  0x00,                         /*
     bmaControls(1) */
  0x00,                         /* iTerminal */
/* Audio Output Terminal */
  AUDIO_OUTPUT_TERMINAL_DESC_SIZE, /* bLength
     */
  AUDIO_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_CONTROL_OUTPUT_TERMINAL, /*
     bDescriptorSubtype */
  0x03,                         /* bTerminalID
     */
  WBVAL(AUDIO_TERMINAL_SPEAKER), /*
     wTerminalType */
  0x00,                         /*
     bAssocTerminal */
  0x02,                         /* bSourceID */
  0x00,                         /* iTerminal */
/* Interface 1, Alternate Setting 0, Audio
    Streaming - Zero Bandwith */
  USB_INTERFACE_DESC_SIZE,      /* bLength */
  USB_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  0x01,                         /*
     bInterfaceNumber */
  0x00,                         /*
     bAlternateSetting */
  0x00,                         /*
     bNumEndpoints */
  USB_DEVICE_CLASS_AUDIO,       /*
     bInterfaceClass */
  AUDIO_SUBCLASS_AUDIOSTREAMING, /*
     bInterfaceSubClass */
  AUDIO_PROTOCOL_UNDEFINED,     /*
     bInterfaceProtocol */
  0x00,                         /* iInterface */
/* Interface 1, Alternate Setting 1, Audio
    Streaming - Operational */
  USB_INTERFACE_DESC_SIZE,      /* bLength */
  USB_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  0x01,                         /*
     bInterfaceNumber */
  0x01,                         /*
     bAlternateSetting */
  0x01,                         /*
     bNumEndpoints */
  USB_DEVICE_CLASS_AUDIO,       /*
     bInterfaceClass */
  AUDIO_SUBCLASS_AUDIOSTREAMING, /*
     bInterfaceSubClass */
  AUDIO_PROTOCOL_UNDEFINED,     /*
     bInterfaceProtocol */
  0x00,                         /* iInterface */
/* Audio Streaming Interface */
  AUDIO_STREAMING_INTERFACE_DESC_SIZE, /*
     bLength */
  AUDIO_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_STREAMING_GENERAL,      /*
     bDescriptorSubtype */
  0x01,                         /*
     bTerminalLink */
  0x01,                         /* bDelay */
  WBVAL(AUDIO_FORMAT_PCM),      /* wFormatTag */
/* Audio Type I Format */
  AUDIO_FORMAT_TYPE_I_DESC_SZ(1), /* bLength */
  AUDIO_INTERFACE_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_STREAMING_FORMAT_TYPE, /*
     bDescriptorSubtype */
  AUDIO_FORMAT_TYPE_I,          /* bFormatType
     */
  0x01,                         /* bNrChannels
     */
  0x02,                         /*
     bSubFrameSize */
  16,                           /*
     bBitResolution */
  0x01,                         /* bSamFreqType
     */
  B3VAL(32000),                 /* tSamFreq */
/* Endpoint - Standard Descriptor */
  AUDIO_STANDARD_ENDPOINT_DESC_SIZE, /*
     bLength */
  USB_ENDPOINT_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  USB_ENDPOINT_OUT(3),          /*
     bEndpointAddress */
  USB_ENDPOINT_TYPE_ISOCHRONOUS, /*
     bmAttributes */
  WBVAL(64),                    /*
     wMaxPacketSize */
  0x01,                         /* bInterval */
  0x00,                         /* bRefresh */
  0x00,                         /*
     bSynchAddress */
/* Endpoint - Audio Streaming */
  AUDIO_STREAMING_ENDPOINT_DESC_SIZE, /*
     bLength */
  AUDIO_ENDPOINT_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  AUDIO_ENDPOINT_GENERAL,       /* bDescriptor
     */
  0x00,                         /* bmAttributes
     */
  0x00,                         /*
     bLockDelayUnits */
  WBVAL(0x0000),                /* wLockDelay */
/* Terminator */
  0                             /* bLength */
};
```

```
/* USB String Descriptor (optional) */
const uint8_t USB_StringDescriptor[] = {
/* Index 0x00: LANGID Codes */
  0x04,                       /* bLength */
  USB_STRING_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  WBVAL(0x0409), /* US English */ /* wLANGID */
/* Index 0x01: Manufacturer */
  (13*2 + 2),                 /* bLength (13
     Char + Type + lenght) */
  USB_STRING_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  'N',0,
  'X',0,
  'P',0,
  ' ',0,
  'S',0,
  'e',0,
  'm',0,
  'i',0,
  'c',0,
  'o',0,
  'n',0,
  'd',0,
  ' ',0,
/* Index 0x02: Product */
  (20*2 + 2),                 /* bLength ( 20
     Char + Type + lenght) */
  USB_STRING_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  'N',0,
  'X',0,
  'P',0,
  ' ',0,
  'L',0,
  'P',0,
  'C',0,
  '1',0,
  '7',0,
  'x',0,
  'x',0,
  ' ',0,
  'S',0,
  'p',0,
  'e',0,
  'a',0,
  'k',0,
  'e',0,
  'r',0,
  ' ',0,
/* Index 0x03: Serial Number */
  (12*2 + 2),                 /* bLength (12
     Char + Type + lenght) */
  USB_STRING_DESCRIPTOR_TYPE, /*
     bDescriptorType */
  'D',0,
  'E',0,
  'M',0,
  'O',0,
  '0',0,
  '0',0,
  '0',0,
  '0',0,
  '0',0,
  '0',0,
  '0',0,
  '0',0,
};
```

N. usbdesc.h

```
/*-------------------------------------------------------
 *    U S B - K e r n e l
 *-------------------------------------------------------
 * Name: usbdesc.h
 * Purpose: USB Descriptors Definitions
 * Version: V1.20
 *-------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *-------------------------------------------------------

#ifndef __USBDESC_H__
#define __USBDESC_H__


#define WBVAL(x) (x & 0xFF),((x >> 8) & 0xFF)
#define B3VAL(x) (x & 0xFF),((x >> 8) &
    0xFF),((x >> 16) & 0xFF)

#define USB_DEVICE_DESC_SIZE
    (sizeof(USB_DEVICE_DESCRIPTOR))
#define USB_CONFIGUARTION_DESC_SIZE
    (sizeof(USB_CONFIGURATION_DESCRIPTOR))
#define USB_INTERFACE_DESC_SIZE
    (sizeof(USB_INTERFACE_DESCRIPTOR))
#define USB_ENDPOINT_DESC_SIZE
    (sizeof(USB_ENDPOINT_DESCRIPTOR))

extern const uint8_t USB_DeviceDescriptor[];
extern const uint8_t USB_ConfigDescriptor[];
extern const uint8_t USB_StringDescriptor[];


#endif /* __USBDESC_H__ */
```

O. usbaudio.h

```
/*-------------------------------------------------------
 *    Name:  usbaudio.h
 *    Purpose: USB Audio Demo Definitions
 *    Version: V1.10
 *-------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
```

```
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------

/* Audio Definitions */
#define DATA_FREQ 32000       /* Audio Data
    Frequency */
#define P_S   32              /* Packet Size
    */
#if USB_DMA
#define P_C   4               /* Packet Count
    */
#else
#define P_C   1               /* Packet Count
    */
#endif
#define B_S   (8*P_C*P_S)     /* Buffer Size
    */

/* Push Button Definitions */
// #define PBINT 0x00004000     /* P0.14 */

/* LED Definitions */
#define LEDMSK 0x000000FF     /* P2.0..7 */

/* Audio Demo Variables */
extern uint8_t Mute;            /* Mute State
    */
extern uint32_t Volume;         /* Volume
    Level */
extern uint16_t VolCur;          /* Volume
    Current Value */
#if !USB_DMA
extern uint32_t InfoBuf[P_C];   /* Packet
    Info Buffer */
extern short DataBuf[B_S];    /* Data Buffer
    */
#else
extern uint32_t *InfoBuf;
extern short *DataBuf;
#endif
extern uint16_t DataOut;         /* Data Out
    Index */
extern uint16_t DataIn;          /* Data In
    Index */
extern uint8_t DataRun;          /* Data
    Stream Run State */
```

P. usbcfg.h

```
/*----------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------
 * Name: usbcfg.h
 * Purpose: USB Custom Configuration
 * Version: V1.20
 *----------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------
 * History:
 *       V1.20 Added vendor specific support
 *       V1.00 Initial Version
 *----------------------------------------------

#ifndef __USBCFG_H__
#define __USBCFG_H__

//*** <<< Use Configuration Wizard in Context
    Menu >>> ***

/*
// <h> USB Configuration
//  <o0> USB Power
//      <i> Default Power Setting
//      <0=> Bus-powered
//      <1=> Self-powered
//  <o1> Max Number of Interfaces <1-256>
//  <o2> Max Number of Endpoints <1-32>
//  <o3> Max Endpoint 0 Packet Size
//      <8=> 8 Bytes <16=> 16 Bytes <32=> 32
    Bytes <64=> 64 Bytes
//  <e4> DMA Transfer
//    <i> Use DMA for selected Endpoints
//    <o5.0> Endpoint 0 Out
//    <o5.1> Endpoint 0 In
//    <o5.2> Endpoint 1 Out
//    <o5.3> Endpoint 1 In
//    <o5.4> Endpoint 2 Out
//    <o5.5> Endpoint 2 In
//    <o5.6> Endpoint 3 Out
//    <o5.7> Endpoint 3 In
//    <o5.8> Endpoint 4 Out
//    <o5.9> Endpoint 4 In
//    <o5.10> Endpoint 5 Out
//    <o5.11> Endpoint 5 In
//    <o5.12> Endpoint 6 Out
//    <o5.13> Endpoint 6 In
//    <o5.14> Endpoint 7 Out
//    <o5.15> Endpoint 7 In
//    <o5.16> Endpoint 8 Out
//    <o5.17> Endpoint 8 In
```

```
//    <o5.18> Endpoint 9 Out
//    <o5.19> Endpoint 9 In
//    <o5.20> Endpoint 10 Out
//    <o5.21> Endpoint 10 In
//    <o5.22> Endpoint 11 Out
//    <o5.23> Endpoint 11 In
//    <o5.24> Endpoint 12 Out
//    <o5.25> Endpoint 12 In
//    <o5.26> Endpoint 13 Out
//    <o5.27> Endpoint 13 In
//    <o5.28> Endpoint 14 Out
//    <o5.29> Endpoint 14 In
//    <o5.30> Endpoint 15 Out
//    <o5.31> Endpoint 15 In
//  </e>
// </h>
*/

#define USB_POWER    0
#define USB_IF_NUM   4
#define USB_EP_NUM   32
#define USB_MAX_PACKET0 64
#define USB_DMA      1
#define USB_DMA_EP   0x00000040


/*
// <h> USB Event Handlers
//  <h> Device Events
//    <o0.0> Power Event
//    <o1.0> Reset Event
//    <o2.0> Suspend Event
//    <o3.0> Resume Event
//    <o4.0> Remote Wakeup Event
//    <o5.0> Start of Frame Event
//    <o6.0> Error Event
//  </h>
//  <h> Endpoint Events
//    <o7.0> Endpoint 0 Event
//    <o7.1> Endpoint 1 Event
//    <o7.2> Endpoint 2 Event
//    <o7.3> Endpoint 3 Event
//    <o7.4> Endpoint 4 Event
//    <o7.5> Endpoint 5 Event
//    <o7.6> Endpoint 6 Event
//    <o7.7> Endpoint 7 Event
//    <o7.8> Endpoint 8 Event
//    <o7.9> Endpoint 9 Event
//    <o7.10> Endpoint 10 Event
//    <o7.11> Endpoint 11 Event
//    <o7.12> Endpoint 12 Event
//    <o7.13> Endpoint 13 Event
//    <o7.14> Endpoint 14 Event
//    <o7.15> Endpoint 15 Event
//  </h>
//  <h> USB Core Events
//    <o8.0> Set Configuration Event
//    <o9.0> Set Interface Event
//    <o10.0> Set/Clear Feature Event
//  </h>
// </h>
*/

#define USB_POWER_EVENT 0
#define USB_RESET_EVENT 1
#define USB_SUSPEND_EVENT 0
#define USB_RESUME_EVENT 0
#define USB_WAKEUP_EVENT 0
#define USB_SOF_EVENT 1
#define USB_ERROR_EVENT 0
#define USB_EP_EVENT 0x0009
#define USB_CONFIGURE_EVENT 0
#define USB_INTERFACE_EVENT 0
#define USB_FEATURE_EVENT 0


/*
// <e0> USB Class Support
//  <i> enables USB Class specific Requests
//  <e1> Human Interface Device (HID)
//    <o2> Interface Number <0-255>
//  </e>
//  <e3> Mass Storage
//    <o4> Interface Number <0-255>
//  </e>
//  <e5> Audio Device
//    <o6> Control Interface Number <0-255>
//    <o7> Streaming Interface 1 Number <0-255>
//    <o8> Streaming Interface 2 Number <0-255>
//  </e>
//  <e9> Communication Device
//    <o10> Control Interface Number <0-255>
//    <o11> Bulk Interface Number <0-255>
//    <o12> Max Communication Device Buffer
    Size
//      <8=> 8 Bytes <16=> 16 Bytes <32=> 32
    Bytes <64=> 64 Bytes
//  </e>
// </e>
*/

#define USB_CLASS    1
#define USB_HID      0
#define USB_HID_IF_NUM 0
#define USB_MSC      0
#define USB_MSC_IF_NUM 0
#define USB_AUDIO    1
#define USB_ADC_CIF_NUM 0
#define USB_ADC_SIF1_NUM 1
#define USB_ADC_SIF2_NUM 2
#define USB_CDC    0
#define USB_CDC_CIF_NUM 0
#define USB_CDC_DIF_NUM 1
#define USB_CDC_BUFSIZE 64

/*
// <e0> USB Vendor Support
//  <i> enables USB Vendor specific Requests
// </e>
*/
#define USB_VENDOR   0


#endif /* __USBCFG_H__ */
```

Q. audio.h

```
/*----------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------
 *    Name:  AUDIO.H
 *    Purpose: USB Audio Device Class
    Definitions
```

```
 *    Version: V1.10
 *----------------------------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on Philips LPC2xxx microcontroller
    devices only. Nothing else gives
 *    you the right to use this software.
 *
 *    Copyright (c) 2005-2006 Keil Software.
 *----------------------------------------------------------------------*/

#ifndef __AUDIO_H__
#define __AUDIO_H__


/* Audio Interface Subclass Codes */
#define AUDIO_SUBCLASS_UNDEFINED    0x00
#define AUDIO_SUBCLASS_AUDIOCONTROL 0x01
#define AUDIO_SUBCLASS_AUDIOSTREAMING 0x02
#define AUDIO_SUBCLASS_MIDISTREAMING 0x03

/* Audio Interface Protocol Codes */
#define AUDIO_PROTOCOL_UNDEFINED    0x00


/* Audio Descriptor Types */
#define AUDIO_UNDEFINED_DESCRIPTOR_TYPE 0x20
#define AUDIO_DEVICE_DESCRIPTOR_TYPE 0x21
#define AUDIO_CONFIGURATION_DESCRIPTOR_TYPE
    0x22
#define AUDIO_STRING_DESCRIPTOR_TYPE 0x23
#define AUDIO_INTERFACE_DESCRIPTOR_TYPE 0x24
#define AUDIO_ENDPOINT_DESCRIPTOR_TYPE 0x25


/* Audio Control Interface Descriptor
    Subtypes */
#define AUDIO_CONTROL_UNDEFINED     0x00
#define AUDIO_CONTROL_HEADER        0x01
#define AUDIO_CONTROL_INPUT_TERMINAL 0x02
#define AUDIO_CONTROL_OUTPUT_TERMINAL 0x03
#define AUDIO_CONTROL_MIXER_UNIT    0x04
#define AUDIO_CONTROL_SELECTOR_UNIT 0x05
#define AUDIO_CONTROL_FEATURE_UNIT  0x06
#define AUDIO_CONTROL_PROCESSING_UNIT 0x07
#define AUDIO_CONTROL_EXTENSION_UNIT 0x08

/* Audio Streaming Interface Descriptor
    Subtypes */
#define AUDIO_STREAMING_UNDEFINED   0x00
#define AUDIO_STREAMING_GENERAL     0x01
#define AUDIO_STREAMING_FORMAT_TYPE 0x02
#define AUDIO_STREAMING_FORMAT_SPECIFIC 0x03

/* Audio Endpoint Descriptor Subtypes */
#define AUDIO_ENDPOINT_UNDEFINED    0x00
#define AUDIO_ENDPOINT_GENERAL      0x01

/* Audio Descriptor Sizes */
#define AUDIO_CONTROL_INTERFACE_DESC_SZ(n)
    0x08+n
#define AUDIO_STREAMING_INTERFACE_DESC_SIZE
    0x07
#define AUDIO_INPUT_TERMINAL_DESC_SIZE 0x0C
#define AUDIO_OUTPUT_TERMINAL_DESC_SIZE 0x09
#define AUDIO_MIXER_UNIT_DESC_SZ(p,n) 0x0A+p+n
#define AUDIO_SELECTOR_UNIT_DESC_SZ(p) 0x06+p
#define AUDIO_FEATURE_UNIT_DESC_SZ(ch,n)
    0x07+(ch+1)*n
#define AUDIO_PROCESSING_UNIT_DESC_SZ(p,n,x)
    0x0D+p+n+x
#define AUDIO_EXTENSION_UNIT_DESC_SZ(p,n)
    0x0D+p+n
#define AUDIO_STANDARD_ENDPOINT_DESC_SIZE 0x09
#define AUDIO_STREAMING_ENDPOINT_DESC_SIZE
    0x07----------------------*/


/* Audio Processing Unit Process Types */
#define AUDIO_UNDEFINED_PROCESS     0x00
#define AUDIO_UP_DOWN_MIX_PROCESS   0x01
#define AUDIO_DOLBY_PROLOGIC_PROCESS 0x02
#define AUDIO_3D_STEREO_PROCESS     0x03
#define AUDIO_REVERBERATION_PROCESS 0x04
#define AUDIO_CHORUS_PROCESS        0x05
#define AUDIO_DYN_RANGE_COMP_PROCESS 0x06


/* Audio Request Codes */
#define AUDIO_REQUEST_UNDEFINED     0x00
#define AUDIO_REQUEST_SET_CUR       0x01
#define AUDIO_REQUEST_GET_CUR       0x81
#define AUDIO_REQUEST_SET_MIN       0x02
#define AUDIO_REQUEST_GET_MIN       0x82
#define AUDIO_REQUEST_SET_MAX       0x03
#define AUDIO_REQUEST_GET_MAX       0x83
#define AUDIO_REQUEST_SET_RES       0x04
#define AUDIO_REQUEST_GET_RES       0x84
#define AUDIO_REQUEST_SET_MEM       0x05
#define AUDIO_REQUEST_GET_MEM       0x85
#define AUDIO_REQUEST_GET_STAT      0xFF


/* Audio Control Selector Codes */
#define AUDIO_CONTROL_UNDEFINED     0x00  /*
    Common Selector */

/* Terminal Control Selectors */
#define AUDIO_COPY_PROTECT_CONTROL  0x01

/* Feature Unit Control Selectors */
#define AUDIO_MUTE_CONTROL          0x01
#define AUDIO_VOLUME_CONTROL        0x02
#define AUDIO_BASS_CONTROL          0x03
#define AUDIO_MID_CONTROL           0x04
#define AUDIO_TREBLE_CONTROL        0x05
#define AUDIO_GRAPHIC_EQUALIZER_CONTROL 0x06
#define AUDIO_AUTOMATIC_GAIN_CONTROL 0x07
#define AUDIO_DELAY_CONTROL         0x08
#define AUDIO_BASS_BOOST_CONTROL    0x09
#define AUDIO_LOUDNESS_CONTROL      0x0A

/* Processing Unit Control Selectors: */
#define AUDIO_ENABLE_CONTROL        0x01  /*
    Common Selector */
```

```
#define AUDIO_MODE_SELECT_CONTROL   0x02  /*
    Common Selector */

/* - Up/Down-mix Control Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */
/*    AUDIO_MODE_SELECT_CONTROL     0x02
    Common Selector */

/* - Dolby Prologic Control Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */
/*    AUDIO_MODE_SELECT_CONTROL     0x02
    Common Selector */

/* - 3D Stereo Extender Control Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */
#define AUDIO_SPACIOUSNESS_CONTROL  0x02

/* - Reverberation Control Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */
#define AUDIO_REVERB_LEVEL_CONTROL  0x02
#define AUDIO_REVERB_TIME_CONTROL   0x03
#define AUDIO_REVERB_FEEDBACK_CONTROL 0x04

/* - Chorus Control Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */
#define AUDIO_CHORUS_LEVEL_CONTROL  0x02
#define AUDIO_SHORUS_RATE_CONTROL   0x03
#define AUDIO_CHORUS_DEPTH_CONTROL  0x04

/* - Dynamic Range Compressor Control
    Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */
#define AUDIO_COMPRESSION_RATE_CONTROL 0x02
#define AUDIO_MAX_AMPL_CONTROL      0x03
#define AUDIO_THRESHOLD_CONTROL     0x04
#define AUDIO_ATTACK_TIME_CONTROL   0x05
#define AUDIO_RELEASE_TIME_CONTROL  0x06

/* Extension Unit Control Selectors */
/*    AUDIO_ENABLE_CONTROL          0x01
    Common Selector */

/* Endpoint Control Selectors */
#define AUDIO_SAMPLING_FREQ_CONTROL 0x01
#define AUDIO_PITCH_CONTROL         0x02


/* Audio Format Specific Control Selectors */

/* MPEG Control Selectors */
#define AUDIO_MPEG_CONTROL_UNDEFINED 0x00
#define AUDIO_MPEG_DUAL_CHANNEL_CONTROL 0x01
#define AUDIO_MPEG_SECOND_STEREO_CONTROL 0x02
#define AUDIO_MPEG_MULTILINGUAL_CONTROL 0x03
#define AUDIO_MPEG_DYN_RANGE_CONTROL 0x04
#define AUDIO_MPEG_SCALING_CONTROL  0x05
#define AUDIO_MPEG_HILO_SCALING_CONTROL 0x06

/* AC-3 Control Selectors */
#define AUDIO_AC3_CONTROL_UNDEFINED 0x00
#define AUDIO_AC3_MODE_CONTROL      0x01
#define AUDIO_AC3_DYN_RANGE_CONTROL 0x02
#define AUDIO_AC3_SCALING_CONTROL   0x03
#define AUDIO_AC3_HILO_SCALING_CONTROL 0x04


/* Audio Format Types */
#define AUDIO_FORMAT_TYPE_UNDEFINED 0x00
#define AUDIO_FORMAT_TYPE_I         0x01
#define AUDIO_FORMAT_TYPE_II        0x02
#define AUDIO_FORMAT_TYPE_III       0x03


/* Audio Format Type Descriptor Sizes */
#define AUDIO_FORMAT_TYPE_I_DESC_SZ(n)
    0x08+(n*3)
#define AUDIO_FORMAT_TYPE_II_DESC_SZ(n)
    0x09+(n*3)
#define AUDIO_FORMAT_TYPE_III_DESC_SZ(n)
    0x08+(n*3)
#define AUDIO_FORMAT_MPEG_DESC_SIZE 0x09
#define AUDIO_FORMAT_AC3_DESC_SIZE  0x0A


/* Audio Data Format Codes */

/* Audio Data Format Type I Codes */
#define AUDIO_FORMAT_TYPE_I_UNDEFINED 0x0000
#define AUDIO_FORMAT_PCM            0x0001
#define AUDIO_FORMAT_PCM8           0x0002
#define AUDIO_FORMAT_IEEE_FLOAT     0x0003
#define AUDIO_FORMAT_ALAW           0x0004
#define AUDIO_FORMAT_MULAW          0x0005

/* Audio Data Format Type II Codes */
#define AUDIO_FORMAT_TYPE_II_UNDEFINED 0x1000
#define AUDIO_FORMAT_MPEG           0x1001
#define AUDIO_FORMAT_AC3            0x1002

/* Audio Data Format Type III Codes */
#define AUDIO_FORMAT_TYPE_III_UNDEFINED 0x2000
#define AUDIO_FORMAT_IEC1937_AC3    0x2001
#define AUDIO_FORMAT_IEC1937_MPEG1_L1 0x2002
#define AUDIO_FORMAT_IEC1937_MPEG1_L2_3 0x2003
#define AUDIO_FORMAT_IEC1937_MPEG2_NOEXT
    0x2003
#define AUDIO_FORMAT_IEC1937_MPEG2_EXT 0x2004
#define AUDIO_FORMAT_IEC1937_MPEG2_L1_LS
    0x2005
#define AUDIO_FORMAT_IEC1937_MPEG2_L2_3 0x2006


/* Predefined Audio Channel Configuration
    Bits */
#define AUDIO_CHANNEL_M             0x0000 /*
    Mono */
#define AUDIO_CHANNEL_L             0x0001 /*
    Left Front */
#define AUDIO_CHANNEL_R             0x0002 /*
    Right Front */
#define AUDIO_CHANNEL_C             0x0004 /*
    Center Front */
#define AUDIO_CHANNEL_LFE           0x0008 /*
    Low Freq. Enhance. */
#define AUDIO_CHANNEL_LS            0x0010 /*
    Left Surround */
#define AUDIO_CHANNEL_RS            0x0020 /*
    Right Surround */
```

```
#define AUDIO_CHANNEL_LC            0x0040 /*
    Left of Center */
#define AUDIO_CHANNEL_RC            0x0080 /*
    Right of Center */
#define AUDIO_CHANNEL_S             0x0100 /*
    Surround */
#define AUDIO_CHANNEL_SL            0x0200 /*
    Side Left */
#define AUDIO_CHANNEL_SR            0x0400 /*
    Side Right */
#define AUDIO_CHANNEL_T             0x0800 /*
    Top */


/* Feature Unit Control Bits */
#define AUDIO_CONTROL_MUTE          0x0001
#define AUDIO_CONTROL_VOLUME        0x0002
#define AUDIO_CONTROL_BASS          0x0004
#define AUDIO_CONTROL_MID           0x0008
#define AUDIO_CONTROL_TREBLE        0x0010
#define AUDIO_CONTROL_GRAPHIC_EQUALIZER 0x0020
#define AUDIO_CONTROL_AUTOMATIC_GAIN 0x0040
#define AUDIO_CONTROL_DEALY         0x0080
#define AUDIO_CONTROL_BASS_BOOST    0x0100
#define AUDIO_CONTROL_LOUDNESS      0x0200

/* Processing Unit Control Bits: */
#define AUDIO_CONTROL_ENABLE        0x0001 /*
    Common Bit */
#define AUDIO_CONTROL_MODE_SELECT   0x0002 /*
    Common Bit */

/* - Up/Down-mix Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */
/*    AUDIO_CONTROL_MODE_SELECT     0x0002
    Common Bit */

/* - Dolby Prologic Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */
/*    AUDIO_CONTROL_MODE_SELECT     0x0002
    Common Bit */

/* - 3D Stereo Extender Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */
#define AUDIO_CONTROL_SPACIOUSNESS  0x0002

/* - Reverberation Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */
#define AUDIO_CONTROL_REVERB_TYPE   0x0002
#define AUDIO_CONTROL_REVERB_LEVEL  0x0004
#define AUDIO_CONTROL_REVERB_TIME   0x0008
#define AUDIO_CONTROL_REVERB_FEEDBACK 0x0010

/* - Chorus Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */
#define AUDIO_CONTROL_CHORUS_LEVEL  0x0002
#define AUDIO_CONTROL_SHORUS_RATE   0x0004
#define AUDIO_CONTROL_CHORUS_DEPTH  0x0008

/* - Dynamic Range Compressor Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */
#define AUDIO_CONTROL_COMPRESSION_RATE 0x0002
#define AUDIO_CONTROL_MAX_AMPL      0x0004
#define AUDIO_CONTROL_THRESHOLD     0x0008
#define AUDIO_CONTROL_ATTACK_TIME   0x0010
#define AUDIO_CONTROL_RELEASE_TIME  0x0020

/* Extension Unit Control Bits */
/*    AUDIO_CONTROL_ENABLE          0x0001
    Common Bit */

/* Endpoint Control Bits */
#define AUDIO_CONTROL_SAMPLING_FREQ 0x01
#define AUDIO_CONTROL_PITCH         0x02
#define AUDIO_MAX_PACKETS_ONLY      0x80


/* Audio Terminal Types */

/* USB Terminal Types */
#define AUDIO_TERMINAL_USB_UNDEFINED 0x0100
#define AUDIO_TERMINAL_USB_STREAMING 0x0101
#define AUDIO_TERMINAL_USB_VENDOR_SPECIFIC
    0x01FF

/* Input Terminal Types */
#define AUDIO_TERMINAL_INPUT_UNDEFINED 0x0200
#define AUDIO_TERMINAL_MICROPHONE   0x0201
#define AUDIO_TERMINAL_DESKTOP_MICROPHONE
    0x0202
#define AUDIO_TERMINAL_PERSONAL_MICROPHONE
    0x0203
#define AUDIO_TERMINAL_OMNI_DIR_MICROPHONE
    0x0204
#define AUDIO_TERMINAL_MICROPHONE_ARRAY 0x0205
#define AUDIO_TERMINAL_PROCESSING_MIC_ARRAY
    0x0206

/* Output Terminal Types */
#define AUDIO_TERMINAL_OUTPUT_UNDEFINED 0x0300
#define AUDIO_TERMINAL_SPEAKER      0x0301
#define AUDIO_TERMINAL_HEADPHONES   0x0302
#define AUDIO_TERMINAL_HEAD_MOUNTED_AUDIO
    0x0303
#define AUDIO_TERMINAL_DESKTOP_SPEAKER 0x0304
#define AUDIO_TERMINAL_ROOM_SPEAKER 0x0305
#define AUDIO_TERMINAL_COMMUNICATION_SPEAKER
    0x0306
#define AUDIO_TERMINAL_LOW_FREQ_SPEAKER 0x0307

/* Bi-directional Terminal Types */
#define
    AUDIO_TERMINAL_BIDIRECTIONAL_UNDEFINED
    0x0400
#define AUDIO_TERMINAL_HANDSET      0x0401
#define AUDIO_TERMINAL_HEAD_MOUNTED_HANDSET
    0x0402
#define AUDIO_TERMINAL_SPEAKERPHONE 0x0403
#define
    AUDIO_TERMINAL_SPEAKERPHONE_ECHOSUPRESS
    0x0404
#define
    AUDIO_TERMINAL_SPEAKERPHONE_ECHOCANCEL
    0x0405

/* Telephony Terminal Types */
#define AUDIO_TERMINAL_TELEPHONY_UNDEFINED
    0x0500
```

```
#define AUDIO_TERMINAL_PHONE_LINE    0x0501
#define AUDIO_TERMINAL_TELEPHONE     0x0502
#define AUDIO_TERMINAL_DOWN_LINE_PHONE 0x0503

/* External Terminal Types */
#define AUDIO_TERMINAL_EXTERNAL_UNDEFINED
    0x0600
#define AUDIO_TERMINAL_ANALOG_CONNECTOR 0x0601
#define
    AUDIO_TERMINAL_DIGITAL_AUDIO_INTERFACE
    0x0602
#define AUDIO_TERMINAL_LINE_CONNECTOR 0x0603
#define AUDIO_TERMINAL_LEGACY_AUDIO_CONNECTOR
    0x0604
#define AUDIO_TERMINAL_SPDIF_INTERFACE 0x0605
#define AUDIO_TERMINAL_1394_DA_STREAM 0x0606
#define AUDIO_TERMINAL_1394_DA_STREAM_TRACK
    0x0607

/* Embedded Function Terminal Types */
#define AUDIO_TERMINAL_EMBEDDED_UNDEFINED
    0x0700
#define AUDIO_TERMINAL_CALIBRATION_NOISE
    0x0701
#define AUDIO_TERMINAL_EQUALIZATION_NOISE
    0x0702
#define AUDIO_TERMINAL_CD_PLAYER     0x0703
#define AUDIO_TERMINAL_DAT           0x0704
#define AUDIO_TERMINAL_DCC           0x0705
#define AUDIO_TERMINAL_MINI_DISK     0x0706
#define AUDIO_TERMINAL_ANALOG_TAPE   0x0707
#define AUDIO_TERMINAL_PHONOGRAPH    0x0708
#define AUDIO_TERMINAL_VCR_AUDIO     0x0709
#define AUDIO_TERMINAL_VIDEO_DISC_AUDIO 0x070A
#define AUDIO_TERMINAL_DVD_AUDIO     0x070B
#define AUDIO_TERMINAL_TV_TUNER_AUDIO 0x070C
#define
    AUDIO_TERMINAL_SATELLITE_RECEIVER_AUDIO
    0x070D
#define AUDIO_TERMINAL_CABLE_TUNER_AUDIO
    0x070E
#define AUDIO_TERMINAL_DSS_AUDIO     0x070F
#define AUDIO_TERMINAL_RADIO_RECEIVER 0x0710
#define AUDIO_TERMINAL_RADIO_TRANSMITTER
    0x0711
#define AUDIO_TERMINAL_MULTI_TRACK_RECORDER
    0x0712
#define AUDIO_TERMINAL_SYNTHESIZER   0x0713


#endif /* __AUDIO_H__ */
```

R. type.h

```
/*****************************************************
 *  type.h: Type definition Header file for
     NXP LPC17xx Family
 *  Microprocessors
 *
 *  Copyright(C) 2009, NXP Semiconductor
 *  All rights reserved.
 *
 *  History
 *  2009.05.25 ver 1.00 Prelimnary version,
     first Release
 *
*****************************************************/
#include <stdint.h>

#ifndef __TYPE_H__
#define __TYPE_H__

#ifndef NULL
#define NULL ((void *)0)
#endif

#ifndef FALSE
#define FALSE (0)
#endif

#ifndef TRUE
#define TRUE (1)
#endif


typedef enum {RESET = 0, SET = !RESET}
    FlagStatus, ITStatus;
typedef enum {DISABLE = 0, ENABLE = !DISABLE}
    FunctionalState;

#endif /* __TYPE_H__ */
```

S. adcuser.c

```
/*----------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------
 *    Name:  ADCUSER.C
 *    Purpose: Audio Device Class Custom User
    Module
 *    Version: V1.10
 *----------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC family
    microcontroller devices only. Nothing
 *    else gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------
 *****************************************************/

#include "type.h"

#include "usb.h"
#include "audio.h"
#include "usbcfg.h"
#include "usbcore.h"
#include "adcuser.h"

#include "usbaudio.h"
```

```
    uint16_t VolCur = 0x0100; /* Volume
        Current Value */
const uint16_t VolMin = 0x0000; /* Volume
    Minimum Value */
const uint16_t VolMax = 0x0100; /* Volume
    Maximum Value */
const uint16_t VolRes = 0x0004; /* Volume
    Resolution */

/*
 * Audio Device Class Interface Get Request
     Callback
 *  Called automatically on ADC Interface Get
     Request
 *   Parameters:  None (global SetupPacket and
     EP0Buf)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t ADC_IF_GetRequest (void) {

/*
  Interface = SetupPacket.wIndex.WB.L;
  EntityID = SetupPacket.wIndex.WB.H;
  Request = SetupPacket.bRequest;
  Value   = SetupPacket.wValue.W;
  ...
*/

  if (SetupPacket.wIndex.W == 0x0200) {
    /* Feature Unit: Interface = 0, ID = 2 */
    if (SetupPacket.wValue.WB.L == 0) {
      /* Master Channel */
      switch (SetupPacket.wValue.WB.H) {
        case AUDIO_MUTE_CONTROL:
          switch (SetupPacket.bRequest) {
            case AUDIO_REQUEST_GET_CUR:
              EP0Buf[0] = Mute;
              return (TRUE);
          }
          break;
        case AUDIO_VOLUME_CONTROL:
          switch (SetupPacket.bRequest) {
            case AUDIO_REQUEST_GET_CUR:
              *((__packed uint16_t *)EP0Buf) =
                  VolCur;
              return (TRUE);
            case AUDIO_REQUEST_GET_MIN:
              *((__packed uint16_t *)EP0Buf) =
                  VolMin;
              return (TRUE);
            case AUDIO_REQUEST_GET_MAX:
              *((__packed uint16_t *)EP0Buf) =
                  VolMax;
              return (TRUE);
            case AUDIO_REQUEST_GET_RES:
              *((__packed uint16_t *)EP0Buf) =
                  VolRes;
              return (TRUE);
          }
          break;
      }
    }
  }
  return (FALSE); /* Not Supported */
}


/*
 * Audio Device Class Interface Set Request
     Callback
 *  Called automatically on ADC Interface Set
     Request
 *   Parameters:  None (global SetupPacket and
     EP0Buf)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t ADC_IF_SetRequest (void) {

/*
  Interface = SetupPacket.wIndex.WB.L;
  EntityID = SetupPacket.wIndex.WB.H;
  Request = SetupPacket.bRequest;
  Value   = SetupPacket.wValue.W;
  ...
*/

  if (SetupPacket.wIndex.W == 0x0200) {
    /* Feature Unit: Interface = 0, ID = 2 */
    if (SetupPacket.wValue.WB.L == 0) {
      /* Master Channel */
      switch (SetupPacket.wValue.WB.H) {
        case AUDIO_MUTE_CONTROL:
          switch (SetupPacket.bRequest) {
            case AUDIO_REQUEST_SET_CUR:
              Mute = EP0Buf[0];
              return (TRUE);
          }
          break;
        case AUDIO_VOLUME_CONTROL:
          switch (SetupPacket.bRequest) {
            case AUDIO_REQUEST_SET_CUR:
              VolCur = *((__packed uint16_t
                  *)EP0Buf);
              return (TRUE);
          }
          break;
      }
    }
  }
  return (FALSE); /* Not Supported */
}


/*
 * Audio Device Class EndPoint Get Request
     Callback
 *  Called automatically on ADC EndPoint Get
     Request
 *   Parameters:  None (global SetupPacket and
     EP0Buf)
 *   Return Value: TRUE - Success, FALSE -
     Error
 */

uint32_t ADC_EP_GetRequest (void) {

/*
  EndPoint = SetupPacket.wIndex.WB.L;
  Request = SetupPacket.bRequest;
```

```
  Value  = SetupPacket.wValue.W;
  ...
*/
  return (FALSE); /* Not Supported */
}


/*
 * Audio Device Class EndPoint Set Request
    Callback
 *  Called automatically on ADC EndPoint Set
    Request
 *   Parameters:  None (global SetupPacket and
    EP0Buf)
 *   Return Value: TRUE - Success, FALSE -
    Error
 */

uint32_t ADC_EP_SetRequest (void) {

/*
  EndPoint = SetupPacket.wIndex.WB.L;
  Request = SetupPacket.bRequest;
  Value  = SetupPacket.wValue.W;
  ...
*/
  return (FALSE); /* Not Supported */
}
```

T. adcuser.h

```
/*----------------------------------------------------
 *    U S B - K e r n e l
 *----------------------------------------------------
 *    Name:  ADCUSER.H
 *    Purpose: Audio Device Class Custom User
    Definitions
 *    Version: V1.10
 *----------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on Philips LPC2xxx microcontroller
    devices only. Nothing else gives
 *    you the right to use this software.
 *
 *    Copyright (c) 2005-2006 Keil Software.
 *----------------------------------------------------

#ifndef __ADCUSER_H__
#define __ADCUSER_H__


/* Audio Device Class Requests Callback
    Functions */
extern uint32_t ADC_IF_GetRequest (void);
extern uint32_t ADC_IF_SetRequest (void);
extern uint32_t ADC_EP_GetRequest (void);
extern uint32_t ADC_EP_SetRequest (void);


#endif /* __ADCUSER_H__ */
```

U. usbdmain.c

```
/*----------------------------------------------------
 * Name: usbmain.c
 * Purpose: USB Audio Class Demo
 * Version: V1.20
 *----------------------------------------------------
 *    This software is supplied "AS IS"
    without any warranties, express,
 *    implied or statutory, including but not
    limited to the implied
 *    warranties of fitness for purpose,
    satisfactory quality and
 *    noninfringement. Keil extends you a
    royalty-free right to reproduce
 *    and distribute executable files created
    using this software for use
 *    on NXP Semiconductors LPC
    microcontroller devices only. Nothing else
 *    gives you the right to use this
    software.
 *
 * Copyright (c) 2009 Keil - An ARM Company.
    All rights reserved.
 *----------------------------------------------------

#include "LPC17xx.h"             /* LPC17xx
    definitions */
#include "type.h"
#include "stdint.h"
#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbaudio.h"

// include KBD input to exit
#include "KBD.h"
// include LCD and pic for audio screen
#include "GLCD.h"
#include "MP3_player.c"

extern void SystemCoreClockUpdate(void);
extern uint32_t SystemCoreClock;
uint8_t Mute;                        /* Mute
    State */
uint32_t Volume;                     /* Volume
    Level */

#if USB_DMA
uint32_t *InfoBuf = (uint32_t*)(DMA_BUF_ADR);
short *DataBuf = (short *)(DMA_BUF_ADR +
    4*P_C);
#else
uint32_t InfoBuf[P_C];
short DataBuf[B_S];                  /* Data
    Buffer */
#endif

uint16_t DataOut;                     /* Data
    Out Index */
```

```
uint16_t DataIn;                        /* Data
    In Index */

uint8_t DataRun;                        /* Data
    Stream Run State */
uint16_t PotVal;                        /*
    Potenciometer Value */
uint32_t VUM;                           /* VU
    Meter */
uint32_t Tick;                          /* Time
    Tick */


// initialize USB Audio Screen:
void init_usb_screen(void){
  GLCD_Clear(White);
  GLCD_Bitmap(20, 0, 23, 240, (unsigned
      char*)MP3_SPLASH_pixel_data);
}

/*
 * Get Potenciometer Value
 */

void get_potval (void) {
  uint32_t val;

  LPC_ADC->CR |= 0x01000000;    /* Start A/D
      Conversion */
  do {
    val = LPC_ADC->GDR;         /* Read A/D
        Data Register */
  } while ((val & 0x80000000) == 0); /* Wait
      for end of A/D Conversion */
  LPC_ADC->CR &= ~0x01000000;   /* Stop A/D
      Conversion */
  PotVal = ((val >> 8) & 0xF8) + /* Extract
      Potenciometer Value */
         ((val >> 7) & 0x08);
}



/*
 * Timer Counter 0 Interrupt Service Routine
 *  executed each 31.25us (32kHz frequency)
 */

void TIMER0_IRQHandler(void)
{
  long val;
  uint32_t cnt;

  if (DataRun) {                    /* Data
      Stream is running */
    val = DataBuf[DataOut];         /* Get Audio
        Sample */
    cnt = (DataIn - DataOut) & (B_S - 1); /*
        Buffer Data Count */
    if (cnt == (B_S - P_C*P_S)) { /* Too much
        Data in Buffer */
      DataOut++;                    /* Skip one
          Sample */
    }
    if (cnt > (P_C*P_S)) {          /* Still
        enough Data in Buffer */
      DataOut++;                    /* Update
          Data Out Index */
    }
    DataOut &= B_S - 1;             /* Adjust
        Buffer Out Index */
    if (val < 0) VUM -= val;        /*
        Accumulate Neg Value */
    else         VUM += val;        /*
        Accumulate Pos Value */
    val *= Volume;                  /* Apply
        Volume Level */
    val >>= 16;                     /* Adjust
        Value */
    val += 0x8000;                  /* Add Bias
        */
    val &= 0xFFFF;                  /* Mask
        Value */
  } else {
    val = 0x8000;                   /* DAC
        Middle Point */
  }

  if (Mute) {
    val = 0x8000;                   /* DAC
        Middle Point */
  }

  LPC_DAC->CR = val & 0xFFC0;    /* Set Speaker
      Output */

  if ((Tick++ & 0x03FF) == 0) {   /* On every
      1024th Tick */
    get_potval();                   /* Get
        Potenciometer Value */
    if (VolCur == 0x8000) {         /* Check for
        Minimum Level */
      Volume = 0;                   /* No Sound
          */
    } else {
      Volume = VolCur * PotVal;     /* Chained
          Volume Level */
    }
    val = VUM >> 20;                /* Scale
        Accumulated Value */
    VUM = 0;                        /* Clear VUM
        */
    if (val > 7) val = 7;           /* Limit
        Value */
  }

  LPC_TIM0->IR = 1;                 /* Clear
      Interrupt Flag */

   // If I were to redo this project, I would
       check for KBD input over here and
   // then issue a hardware interrupt in order
       to allow for interrupting usb audio

}

// suspend audio
void suspendUsbAudio(void){
  USB_Suspend();
}

/*************************************************
**  Main Function usbAudio()
*************************************************/
int usbAudio(void)
```

```
{
 volatile uint32_t pclkdiv, pclk;
  init_usb_screen();
 /* SystemCoreClockUpdate() updates the
     SystemCoreClock variable */
 SystemCoreClockUpdate();

 LPC_PINCON->PINSEL1
     &=~((0x03<<18)|(0x03<<20));
 /* P0.25, A0.0, function 01, P0.26 AOUT,
     function 10 */
 LPC_PINCON->PINSEL1 |=
     ((0x01<<18)|(0x02<<20));

 /* Enable CLOCK into ADC controller */
 LPC_SC->PCONP |= (1 << 12);

 LPC_ADC->CR = 0x00200E04;  /* ADC: 10-bit
     AIN2 @ 4MHz */
 LPC_DAC->CR = 0x00008000;  /* DAC Output set
     to Middle Point */

 /* By default, the PCLKSELx value is zero,
     thus, the PCLK for
 all the peripherals is 1/4 of the
     SystemCoreClock. */
 /* Bit 2~3 is for TIMER0 */
 pclkdiv = (LPC_SC->PCLKSEL0 >> 2) & 0x03;
 switch ( pclkdiv )
 {
  case 0x00:
  default:
    pclk = SystemCoreClock/4;
  break;
  case 0x01:
    pclk = SystemCoreClock;
  break;
  case 0x02:
    pclk = SystemCoreClock/2;
  break;
  case 0x03:
    pclk = SystemCoreClock/8;
  break;
 }

 LPC_TIM0->MR0 = pclk/DATA_FREQ - 1; /* TC0
     Match Value 0 */
 LPC_TIM0->MCR = 3;      /* TC0 Interrupt and
     Reset on MR0 */
 LPC_TIM0->TCR = 1;      /* TC0 Enable */
 NVIC_EnableIRQ(TIMER0_IRQn);

 USB_Init();      /* USB Initialization */
 USB_Connect(TRUE);  /* USB Connect */

 /********** The main Function is an endless
     loop ***********/
 while(1){
    int kbd = get_button();
    if(kbd == KBD_LEFT) {
      USB_Connect(FALSE);
      //LPC_USB -> DevIntEn = 1;
      //LPC_SC->PCONP |= (1UL<<31);
      break;
    }

  }
  //USB_Connect(FALSE);
  //USB_Suspend();
  //USB_Reset();
  return 1;
}

/*****************************************************
**                    End Of File
*****************************************************/
```

V. *usbdmain.h*

```
#include "LPC17xx.h"              /* LPC17xx
    definitions */
#include "type.h"
#include "stdint.h"
#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbaudio.h"

extern void SystemCoreClockUpdate(void);
extern uint32_t SystemCoreClock;

int usbAudio(void);
void suspendUsbAudio(void);
```

W. *game.c*

```
#include "stdio.h"
#include "stdlib.h"
#include "LPC17xx.h"
#include "KBD.h"
#include "GLCD.h"
#include "LED.h"

/* A basic snake game, with or without
    borders, taken from
    https://github.com/mfoee/MCB1700
*/

#define DELAY_2N 20

int xpos; //horizontal position
int ypos; //vertical position
int size; //size of the body
int direct = 0; //current direction
int prev_direct = 0; //previous direction
int joystick_val = 0; //current joystick val
int joystick_prev_val = KBD_RIGHT; //previous
    joystick val
char str[20],str1[20],str2[20],str3[20];
int snake[100][2]; //snake coordinates.
int delx, dely; //used to figure out where to
    turn
int speed; //how fast the snake will move
int xfood, yfood; //food coordinates
int tempx, tempy; // for the point at which
    the body turns
int collision = 0;
int border = 0;
int score = 0;
int exit_game = 0;
```

```
void food(){
  int i;
  xfood = rand()%9;
  yfood = rand()%20;
  for(i=0;i<size;i++){
    if(xfood == snake[i][0])
      if(yfood == snake[i][1])
        food();
  }
  GLCD_DisplayChar(xfood,yfood,1,0x81);
}

void gameDelay (int count){
  count <<= DELAY_2N;
  while(count--);
}

void setbody(){
  int i; //counting

  for(i=0;i<size;i++){
    switch(direct){
      case 0://right
        snake[i][0] = xpos;
        snake[i][1] = ypos-i;
        break;
      case 1://left
        snake[i][0] = xpos;
        snake[i][1] = ypos+i;
        break;
      case 2://down
        snake[i][0] = xpos+i;
        snake[i][1] = ypos;
        break;
      case 3://up
        snake[i][0] = xpos-1;
        snake[i][1] = ypos;
        break;
    }
  }
}

void addbody(){
  int n=1;
  size++;
  score = score + 2*n;
  if(speed != 0)
    speed--;
  n++;
}

void check(){
  int i;

  //food check
  if(xfood == snake[0][0])
    if(yfood == snake[0][1]){
      addbody();
      food();
    }

  //tail collision check
  for(i=1;i<size;i++){
    if(snake[0][0] == snake[i][0])
      if(snake[0][1] == snake[i][1])
        collision = 1;
  }

  //collision to wall
  if(border == 1){
    //check right wall
    if(snake[0][1] == 19 && snake[1][1] == 18)
      collision = 1;
    //check left wall
    if(snake[0][1] == 0 && snake[1][1] == 1)
      collision = 1;
    //check bottom wall
    if(snake[0][0] == 9 && snake[1][0] == 8)
      collision = 1;
    //check top wall
    if(snake[0][0] == 0 && snake[1][0] == 1)
      collision = 1;
  }
}

void updatebody(){
  int i;
  if(direct == 0){//move right
    for(i=size;i>0;i--){
      if(i -1 == 0){
        snake[0][1] = ypos;
        snake[0][0] = xpos;
      }else{
        GLCD_DisplayChar(snake[i-1][0],snake[i-1][1],1,
            ');
        snake[i-1][1] = snake[i-2][1];
        snake[i-1][0] = snake[i-2][0];
      }
    }
    for(i=1;i<size;i++){
      GLCD_DisplayChar(snake[0][0],snake[0][1],1,0x8B);
      GLCD_DisplayChar(snake[i][0],snake[i][1],1,0x82);
    }
    gameDelay(speed);
  }else if(direct == 1){//move left
    for(i=size;i>0;i--){
      if(i -1 == 0){
        snake[0][1] = ypos;
        snake[0][0] = xpos;
      }else{
        GLCD_DisplayChar(snake[i-1][0],snake[i-1][1],1,
            ');
        snake[i-1][1] = snake[i-2][1];
        snake[i-1][0] = snake[i-2][0];
      }
    }
    for(i=1;i<size;i++){
      GLCD_DisplayChar(snake[0][0],snake[0][1],1,0x89);
      GLCD_DisplayChar(snake[i][0],snake[i][1],1,0x82);
    }
    gameDelay(speed);
  }else if(direct == 2){//move down
    for(i=size;i>0;i--){
      if(i -1 == 0){
        snake[0][1] = ypos;
        snake[0][0] = xpos;
      }else{
        GLCD_DisplayChar(snake[i-1][0],snake[i-1][1],1,
            ');
        snake[i-1][1] = snake[i-2][1];
        snake[i-1][0] = snake[i-2][0];
      }
    }
```

```
    for(i=1;i<size;i++){
      GLCD_DisplayChar(snake[0][0],snake[0][1],1,0x87);
      GLCD_DisplayChar(snake[i][0],snake[i][1],1,0x82);
    }
    gameDelay(speed);
  }else if(direct == 3){//move up
    for(i=size;i>0;i--){
      if(i-1 == 0){
        snake[0][1] = ypos;
        snake[0][0] = xpos;
      }else{
        GLCD_DisplayChar(snake[i-1][0],snake[i-1][1],1,'
            ');
        snake[i-1][1] = snake[i-2][1];
        snake[i-1][0] = snake[i-2][0];
      }
    }
    for(i=1;i<size;i++){
      GLCD_DisplayChar(snake[0][0],snake[0][1],1,0x85);
      GLCD_DisplayChar(snake[i][0],snake[i][1],1,0x82);
    }
    gameDelay(speed);
  }
  check();
}

void direction(int joyval){

  switch(joyval){
    case KBD_UP:
      if (joystick_prev_val == KBD_LEFT ||
          joystick_prev_val == KBD_RIGHT){
        xpos--;
        if (xpos < 0){
          xpos = 9;
        }
        direct = 3;
        prev_direct = direct;
        joystick_prev_val = joystick_val;
        updatebody();
      }
      break;
    case KBD_DOWN:
      if (joystick_prev_val == KBD_LEFT ||
          joystick_prev_val == KBD_RIGHT){
        xpos++;
        if (xpos > 9){
          xpos = 0;
        }
        direct = 2;
        prev_direct = direct;
        joystick_prev_val = joystick_val;
        updatebody();
      }
      break;
    case KBD_LEFT:
      if (joystick_prev_val == KBD_UP ||
          joystick_prev_val == KBD_DOWN){
        ypos--;
        if (ypos < 0){
          ypos = 20;
        }
        direct = 1;
        prev_direct = direct;
        joystick_prev_val = joystick_val;
        updatebody();
      }
      break;
    case KBD_RIGHT:
      if (joystick_prev_val == KBD_UP ||
          joystick_prev_val == KBD_DOWN){
        ypos++;
        if (ypos > 20 ){
          ypos = 0;
        }
        direct = 0;
        prev_direct = direct;
        joystick_prev_val = joystick_val;
        updatebody();
      }
      break;
    case KBD_SELECT:
      exit_game = 1;
      collision = 1;
      break;
    default:
      switch(direct){
        case 0://right
          ypos++;
          if (ypos > 20){
            ypos = 0;
          }
          updatebody();
          check();
          break;
        case 1://left
          ypos--;
          if (ypos < 0){
            ypos = 20;
          }
          updatebody();
          check();
          break;
        case 2://down
          xpos++;
          if (xpos > 9){
            xpos = 0;
          }
          updatebody();
          check();
          break;
        case 3://up
          xpos--;
          if (xpos < 0){
            xpos = 9;
          }
          updatebody();
          check();
          break;
      }
      break;
  }
}

void clearsnake(){
  int i;
  for(i=0;i<size;i++){
    snake[i][0]=1;
    snake[i][1]=1;
  }
}

int game(){
  int joy_difficulty, joy_try;
```

```
  int mode, modesel;
  int highscore=0;
  int gameover, tryagain;
  char scores[54];
//   char fd[20];
  int done=0;

  GLCD_Init();
  KBD_Init();
  LED_Init();

  while(!done){
    modesel = 1;
    mode = 1;
    tryagain = 1;
    direct = 0;
    prev_direct = 0;
    joystick_val = 0;
    joystick_prev_val = KBD_RIGHT;
    speed = 15;
    size = 2;
    xpos = 5;
    ypos = 10;
    GLCD_Clear(Black);
    GLCD_SetBackColor(Black);
    GLCD_SetTextColor(Green);
    GLCD_DisplayString(2,0,1,"Select Border
        mode: ");
    GLCD_DisplayString(4,0,1,"--> ON   ");
    GLCD_DisplayString(5,0,1," OFF     ");
    GLCD_DisplayString(29,0,0," press
        joystick to select the mode, left to
        exit ");

    while(modesel == 1){
      joy_difficulty = get_button();
      switch(joy_difficulty){
        case KBD_DOWN:
          GLCD_DisplayString(4,0,1," ON
              ");
          GLCD_DisplayString(5,0,1,"--> OFF
              ");
          mode = 2;
          break;
        case KBD_UP:
          GLCD_DisplayString(4,0,1,"--> ON
              ");
          GLCD_DisplayString(5,0,1," OFF
              ");
          mode = 1;
          break;
        case KBD_SELECT:
          if(mode == 1)
            border = 1;
          if(mode == 2)
            border = 0;
          modesel = 0;
          GLCD_Clear(White);
          GLCD_SetBackColor(White);
          GLCD_SetTextColor(Black);
          break;
        case KBD_LEFT:
          return 0;
      }
    }

    setbody();
    food();

    while(collision == 0){
      joystick_val = get_button();
      direction(joystick_val);
      sprintf(str," score:[%d]",score);
      GLCD_DisplayString(0,0,0,(unsigned
          char *)str);
    }

    if(collision == 1){
      GLCD_Clear(Red);
      GLCD_SetBackColor(Red);
      GLCD_SetTextColor(White);
      if(score>=highscore)
          highscore = score;
      sprintf(scores," [SCORE: %d]     [HIGH
          SCORE: %d]",score,highscore);
      GLCD_DisplayString(0,0,0,(unsigned
          char *)scores);
      GLCD_DisplayString(2,0,1," GAME OVER
          ");
      GLCD_DisplayString(7,0,1,"Wanna try
          again? :P ");
      GLCD_DisplayString(8,0,1," --> YES");
      GLCD_DisplayString(9,0,1," NO ");
      gameover = 0;
      while(gameover == 0){
        joy_try = get_button();
        switch(joy_try){
          case KBD_DOWN:
            GLCD_DisplayString(8,0,1," YES");
            GLCD_DisplayString(9,0,1," -->
                NO ");
            tryagain = 0;
            break;
          case KBD_UP:
            GLCD_DisplayString(8,0,1," -->
                YES");
            GLCD_DisplayString(9,0,1," NO ");
            tryagain = 1;
            break;
          case KBD_SELECT:
            if(tryagain == 0){
              GLCD_Clear(White);
              done =1;
              return 0;
            }
            if(tryagain == 1){
              gameover = 1;
              collision = 0;
              border = 0;
              clearsnake();
            }
            break;
          case KBD_LEFT:
            done = 1;
            gameover = 1;
            break;
        }
      }
    }
    gameDelay(5);
  }
  return 0;
}
```

X. *game.h*

```
#ifndef __game_h
#define __game_h

extern int game();
extern void gameDelay();

#endif
```

Y. *LPC17xx.h*

```
/**************************************************************************//**
 * @file: LPC17xx.h
 * @purpose: CMSIS Cortex-M3 Core Peripheral Access Layer Header File for
 *         NXP LPC17xx Device Series
 * @version: V1.10
 * @date: 24. September 2010
 *------------------------------------------------------------------------------
 *
 * @note
 * Copyright (C) 2010 ARM Limited. All rights reserved.
 *
 * @par
 * ARM Limited (ARM) is supplying this software for use with Cortex-M3
 * processor based microcontrollers. This file can be freely distributed
 * within development tools that are supporting such ARM based processors.
 *
 * @par
 * THIS SOFTWARE IS PROVIDED "AS IS". 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.
 * ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 *****************************************************************************/


#ifndef __LPC17xx_H__
#define __LPC17xx_H__

/*
 *
 * ========== Interrupt Number Definition ==========
 *
 */

/** @addtogroup LPC17xx_System
 * @{
 */

/** @brief IRQ interrupt source definition */
typedef enum IRQn
{
/****** Cortex-M3 Processor Exceptions Numbers ***********************************************/
  NonMaskableInt_IRQn    = -14,    /*!< 2 Non Maskable Interrupt                */
  MemoryManagement_IRQn  = -12,    /*!< 4 Cortex-M3 Memory Management Interrupt */
  BusFault_IRQn          = -11,    /*!< 5 Cortex-M3 Bus Fault Interrupt         */
  UsageFault_IRQn        = -10,    /*!< 6 Cortex-M3 Usage Fault Interrupt       */
  SVCall_IRQn            = -5,     /*!< 11 Cortex-M3 SV Call Interrupt          */
  DebugMonitor_IRQn      = -4,     /*!< 12 Cortex-M3 Debug Monitor Interrupt    */
  PendSV_IRQn            = -2,     /*!< 14 Cortex-M3 Pend SV Interrupt          */
  SysTick_IRQn           = -1,     /*!< 15 Cortex-M3 System Tick Interrupt      */

/****** LPC17xx Specific Interrupt Numbers ***************************************************/
  WDT_IRQn               = 0,      /*!< Watchdog Timer Interrupt                */
  TIMER0_IRQn            = 1,      /*!< Timer0 Interrupt                        */
  TIMER1_IRQn            = 2,      /*!< Timer1 Interrupt                        */
  TIMER2_IRQn            = 3,      /*!< Timer2 Interrupt                        */
  TIMER3_IRQn            = 4,      /*!< Timer3 Interrupt                        */
  UART0_IRQn             = 5,      /*!< UART0 Interrupt                         */
  UART1_IRQn             = 6,      /*!< UART1 Interrupt                         */
  UART2_IRQn             = 7,      /*!< UART2 Interrupt                         */
  UART3_IRQn             = 8,      /*!< UART3 Interrupt                         */
  PWM1_IRQn              = 9,      /*!< PWM1 Interrupt                          */
  I2C0_IRQn              = 10,     /*!< I2C0 Interrupt                          */
  I2C1_IRQn              = 11,     /*!< I2C1 Interrupt                          */
  I2C2_IRQn              = 12,     /*!< I2C2 Interrupt                          */
  SPI_IRQn               = 13,     /*!< SPI Interrupt                           */
  SSP0_IRQn              = 14,     /*!< SSP0 Interrupt                          */
  SSP1_IRQn              = 15,     /*!< SSP1 Interrupt                          */
  PLL0_IRQn              = 16,     /*!< PLL0 Lock (Main PLL) Interrupt          */
  RTC_IRQn               = 17,     /*!< Real Time Clock Interrupt               */
  EINT0_IRQn             = 18,     /*!< External Interrupt 0 Interrupt          */
  EINT1_IRQn             = 19,     /*!< External Interrupt 1 Interrupt          */
  EINT2_IRQn             = 20,     /*!< External Interrupt 2 Interrupt          */
```

```
  EINT3_IRQn            = 21,     /*!<
      External Interrupt 3 Interrupt      */
  ADC_IRQn              = 22,     /*!< A/D
      Converter Interrupt                 */
  BOD_IRQn              = 23,     /*!<
      Brown-Out Detect Interrupt          */
  USB_IRQn              = 24,     /*!< USB
      Interrupt                           */
  CAN_IRQn              = 25,     /*!< CAN
      Interrupt                           */
  DMA_IRQn              = 26,     /*!< General
      Purpose DMA Interrupt       */
  I2S_IRQn              = 27,     /*!< I2S
      Interrupt                           */
  ENET_IRQn             = 28,     /*!<
      Ethernet Interrupt                  */
  RIT_IRQn              = 29,     /*!<
      Repetitive Interrupt Timer Interrupt */
  MCPWM_IRQn            = 30,     /*!< Motor
      Control PWM Interrupt         */
  QEI_IRQn              = 31,     /*!<
      Quadrature Encoder Interface Interrupt */
  PLL1_IRQn             = 32,     /*!< PLL1
      Lock (USB PLL) Interrupt      */
  USBActivity_IRQn      = 33,     /*!< USB
      Activity Interrupt(For wakeup only) */
  CANActivity_IRQn      = 34      /*!< CAN
      Activity Interrupt(For wakeup only) */
} IRQn_Type;


/*
 *
    ===========================================================
 * ----------- Processor and Core Peripheral
    Section -----------------------
 *
    ===========================================================
 */

/* Configuration of the Cortex-M3 Processor
    and Core Peripherals */
#define __MPU_PRESENT    1       /*!< MPU
    present or not                */
#define __NVIC_PRIO_BITS  5      /*!< Number
    of Bits used for Priority Levels */
#define __Vendor_SysTickConfig 0 /*!< Set to
    1 if different SysTick Config is used */


#include "core_cm3.h"            /* Cortex-M3
    processor and core peripherals */
#include "system_LPC17xx.h"      /* System
    Header                        */


/******************************************************************************
/*            Device Specific Peripheral
    registers structures */
/******************************************************************************

#if defined ( __CC_ARM )
#pragma anon_unions
#endif

/*------------- System Control (SC)
    ------------------------------------------*/
```

```
/** @brief System Control (SC) register
    structure definition */
typedef struct
{
  __IO uint32_t FLASHCFG;         /*!< Offset:
      0x000 (R/W) Flash Accelerator
      Configuration Register */
      uint32_t RESERVED0[31];
  __IO uint32_t PLL0CON;          /*!< Offset:
      0x080 (R/W) PLL0 Control Register */
  __IO uint32_t PLL0CFG;          /*!< Offset:
      0x084 (R/W) PLL0 Configuration Register
      */
  __I uint32_t PLL0STAT;          /*!< Offset:
      0x088 (R/ ) PLL0 Status Register */
  __O uint32_t PLL0FEED;          /*!< Offset:
      0x08C ( /W) PLL0 Feed Register */
      uint32_t RESERVED1[4];
  __IO uint32_t PLL1CON;          /*!< Offset:
      0x0A0 (R/W) PLL1 Control Register */
  __IO uint32_t PLL1CFG;          /*!< Offset:
      0x0A4 (R/W) PLL1 Configuration Register
      */
  __I uint32_t PLL1STAT;          /*!< Offset:
      0x0A8 (R/ ) PLL1 Status Register */
  __O uint32_t PLL1FEED;          /*!< Offset:
      0x0AC ( /W) PLL1 Feed Register */
      uint32_t RESERVED2[4];
  __IO uint32_t PCON;             /*!< Offset:
      0x0C0 (R/W) Power Control Register */
  __IO uint32_t PCONP;            /*!< Offset:
      0x0C4 (R/W) Power Control for
      Peripherals Register */
      uint32_t RESERVED3[15];
  __IO uint32_t CCLKCFG;          /*!< Offset:
      0x104 (R/W) CPU Clock Configure Register
      */
  __IO uint32_t USBCLKCFG;        /*!< Offset:
      0x108 (R/W) USB Clock Configure Register
      */
  __IO uint32_t CLKSRCSEL;        /*!< Offset:
      0x10C (R/W) Clock Source Select Register
      */
  __IO uint32_t CANSLEEPCLR;      /*!< Offset:
      0x110 (R/W) CAN Sleep Clear Register */
  __IO uint32_t CANWAKEFLAGS;     /*!< Offset:
      0x114 (R/W) CAN Wake-up Flags Register */
      uint32_t RESERVED4[10];
  __IO uint32_t EXTINT;           /*!< Offset:
      0x140 (R/W) External Interrupt Flag
      Register */
      uint32_t RESERVED5[1];
  __IO uint32_t EXTMODE;          /*!< Offset:
      0x148 (R/W) External Interrupt Mode
      Register */
  __IO uint32_t EXTPOLAR;         /*!< Offset:
      0x14C (R/W) External Interrupt Polarity
      Register */
      uint32_t RESERVED6[12];
  __IO uint32_t RSID;             /*!< Offset:
      0x180 (R/W) Reset Source Identification
      Register */
      uint32_t RESERVED7[7];
  __IO uint32_t SCS;              /*!< Offset:
      0x1A0 (R/W) System Controls and Status
      Register */
```

```
  __IO uint32_t IRCTRIM;       /* Clock
     Dividers            */
  __IO uint32_t PCLKSEL0;         /*!< Offset:
     0x1A8 (R/W) Peripheral Clock Select 0
     Register */
  __IO uint32_t PCLKSEL1;         /*!< Offset:
     0x1AC (R/W) Peripheral Clock Select 1
     Register */
     uint32_t RESERVED8[4];
  __IO uint32_t USBIntSt;         /*!< Offset:
     0x1C0 (R/W) USB Interrupt Status
     Register */
  __IO uint32_t DMAREQSEL;        /*!< Offset:
     0x1C4 (R/W) DMA Request Select Register
     */
  __IO uint32_t CLKOUTCFG;        /*!< Offset:
     0x1C8 (R/W) Clock Output Configuration
     Register */

 } LPC_SC_TypeDef;

/*------------- Pin Connect Block (PINCON)
    ------------------------------------*/
/** @brief Pin Connect Block (PINCON)
    register structure definition */
typedef struct
{
  __IO uint32_t PINSEL0;     /* !< Offset:
     0x000 PIN Select0 (R/W) */
  __IO uint32_t PINSEL1;     /* !< Offset:
     0x004 PIN Select1 (R/W) */
  __IO uint32_t PINSEL2;     /* !< Offset:
     0x008 PIN Select2 (R/W) */
  __IO uint32_t PINSEL3;     /* !< Offset:
     0x00C PIN Select3 (R/W) */
  __IO uint32_t PINSEL4;     /* !< Offset:
     0x010 PIN Select4 (R/W) */
  __IO uint32_t PINSEL5;     /* !< Offset:
     0x014 PIN Select5 (R/W) */
  __IO uint32_t PINSEL6;     /* !< Offset:
     0x018 PIN Select6 (R/W) */
  __IO uint32_t PINSEL7;     /* !< Offset:
     0x01C PIN Select7 (R/W) */
  __IO uint32_t PINSEL8;     /* !< Offset:
     0x020 PIN Select8 (R/W) */
  __IO uint32_t PINSEL9;     /* !< Offset:
     0x024 PIN Select9 (R/W) */
  __IO uint32_t PINSEL10;    /* !< Offset:
     0x028 PIN Select20 (R/W) */
     uint32_t RESERVED0[5];
  __IO uint32_t PINMODE0;    /* !< Offset:
     0x040 PIN Mode0 (R/W) */
  __IO uint32_t PINMODE1;    /* !< Offset:
     0x044 PIN Mode1 (R/W) */
  __IO uint32_t PINMODE2;    /* !< Offset:
     0x048 PIN Mode2 (R/W) */
  __IO uint32_t PINMODE3;    /* !< Offset:
     0x04C PIN Mode3 (R/W) */
  __IO uint32_t PINMODE4;    /* !< Offset:
     0x050 PIN Mode4 (R/W) */
  __IO uint32_t PINMODE5;    /* !< Offset:
     0x054 PIN Mode5 (R/W) */
  __IO uint32_t PINMODE6;    /* !< Offset:
     0x058 PIN Mode6 (R/W) */
  __IO uint32_t PINMODE7;    /* !< Offset:
     0x05C PIN Mode7 (R/W) */
  __IO uint32_t PINMODE8;    /* !< Offset:
     0x060 PIN Mode8 (R/W) */
  __IO uint32_t PINMODE9;    /* !< Offset:
     0x064 PIN Mode9 (R/W) */
  __IO uint32_t PINMODE_OD0;   /* !< Offset:
     0x068 Open Drain PIN Mode0 (R/W) */
  __IO uint32_t PINMODE_OD1;   /* !< Offset:
     0x06C Open Drain PIN Mode1 (R/W) */
  __IO uint32_t PINMODE_OD2;   /* !< Offset:
     0x070 Open Drain PIN Mode2 (R/W) */
  __IO uint32_t PINMODE_OD3;   /* !< Offset:
     0x074 Open Drain PIN Mode3 (R/W) */
  __IO uint32_t PINMODE_OD4;   /* !< Offset:
     0x078 Open Drain PIN Mode4 (R/W) */
  __IO uint32_t I2CPADCFG;    /* !< Offset:
     0x07C I2C Pad Configure (R/W) */
} LPC_PINCON_TypeDef;

/*------------- General Purpose Input/Output
    (GPIO) ------------------------------*/
/** @brief General Purpose Input/Output
    (GPIO) register structure definition */
typedef struct
{
  union {
    __IO uint32_t FIODIR;    /* !< Offset:
       0x00 Port direction (R/W) */
    struct {
      __IO uint16_t FIODIRL;
      __IO uint16_t FIODIRH;
    };
    struct {
      __IO uint8_t FIODIR0;
      __IO uint8_t FIODIR1;
      __IO uint8_t FIODIR2;
      __IO uint8_t FIODIR3;
    };
  };
  uint32_t RESERVED0[3];
  union {
    __IO uint32_t FIOMASK;    /* !< Offset:
       0x10 Port mask (R/W) */
    struct {
      __IO uint16_t FIOMASKL;
      __IO uint16_t FIOMASKH;
    };
    struct {
      __IO uint8_t FIOMASK0;
      __IO uint8_t FIOMASK1;
      __IO uint8_t FIOMASK2;
      __IO uint8_t FIOMASK3;
    };
  };
  union {
    __IO uint32_t FIOPIN;    /* !< Offset:
       0x14 Port value (R/W) */
    struct {
      __IO uint16_t FIOPINL;
      __IO uint16_t FIOPINH;
    };
    struct {
      __IO uint8_t FIOPIN0;
      __IO uint8_t FIOPIN1;
      __IO uint8_t FIOPIN2;
      __IO uint8_t FIOPIN3;
    };
  };
```

```
  union {
    __IO uint32_t FIOSET;     /* !< Offset:
        0x18 Port output set (R/W) */
    struct {
      __IO uint16_t FIOSETL;
      __IO uint16_t FIOSETH;
    };
    struct {
      __IO uint8_t FIOSET0;
      __IO uint8_t FIOSET1;
      __IO uint8_t FIOSET2;
      __IO uint8_t FIOSET3;
    };
  };
  union {
    __O uint32_t FIOCLR;      /* !< Offset:
        0x1C Port output clear (R/W) */
    struct {
      __O uint16_t FIOCLRL;
      __O uint16_t FIOCLRH;
    };
    struct {
      __O uint8_t FIOCLR0;
      __O uint8_t FIOCLR1;
      __O uint8_t FIOCLR2;
      __O uint8_t FIOCLR3;
    };
  };
} LPC_GPIO_TypeDef;

/** @brief General Purpose Input/Output
    interrupt (GPIOINT) register structure
    definition */
typedef struct
{
  __I uint32_t IntStatus;         /*!< Offset:
      0x000 (R/ ) GPIO overall Interrupt
      Status Register */
  __I uint32_t IO0IntStatR;       /*!< Offset:
      0x004 (R/ ) GPIO Interrupt Status
      Register 0 for Rising edge */
  __I uint32_t IO0IntStatF;       /*!< Offset:
      0x008 (R/ ) GPIO Interrupt Status
      Register 0 for Falling edge */
  __O uint32_t IO0IntClr;         /*!< Offset:
      0x00C (R/W) GPIO Interrupt Clear
      Register 0 */
  __IO uint32_t IO0IntEnR;        /*!< Offset:
      0x010 ( /W) GPIO Interrupt Enable
      Register 0 for Rising edge */
  __IO uint32_t IO0IntEnF;        /*!< Offset:
      0x014 (R/W) GPIO Interrupt Enable
      Register 0 for Falling edge */
      uint32_t RESERVED0[3];
  __I uint32_t IO2IntStatR;       /*!< Offset:
      0x000 (R/ ) GPIO Interrupt Status
      Register 2 for Rising edge */
  __I uint32_t IO2IntStatF;       /*!< Offset:
      0x000 (R/ ) GPIO Interrupt Status
      Register 2 for Falling edge */
  __O uint32_t IO2IntClr;         /*!< Offset:
      0x000 ( /W) GPIO Interrupt Clear
      Register 2 */
  __IO uint32_t IO2IntEnR;        /*!< Offset:
      0x000 (R/W) GPIO Interrupt Enable
      Register 2 for Rising edge */
  __IO uint32_t IO2IntEnF;        /*!< Offset:
      0x000 (R/W) GPIO Interrupt Enable
      Register 2 for Falling edge */
} LPC_GPIOINT_TypeDef;

/*------------- Timer (TIM)
    ----------------------------------------------*/
/** @brief Timer (TIM) register structure
    definition */
typedef struct
{
  __IO uint32_t IR;               /*!< Offset:
      0x000 (R/W) Interrupt Register */
  __IO uint32_t TCR;              /*!< Offset:
      0x004 (R/W) Timer Control Register */
  __IO uint32_t TC;               /*!< Offset:
      0x008 (R/W) Timer Counter Register */
  __IO uint32_t PR;               /*!< Offset:
      0x00C (R/W) Prescale Register */
  __IO uint32_t PC;               /*!< Offset:
      0x010 (R/W) Prescale Counter Register */
  __IO uint32_t MCR;              /*!< Offset:
      0x014 (R/W) Match Control Register */
  __IO uint32_t MR0;              /*!< Offset:
      0x018 (R/W) Match Register 0 */
  __IO uint32_t MR1;              /*!< Offset:
      0x01C (R/W) Match Register 1 */
  __IO uint32_t MR2;              /*!< Offset:
      0x020 (R/W) Match Register 2 */
  __IO uint32_t MR3;              /*!< Offset:
      0x024 (R/W) Match Register 3 */
  __IO uint32_t CCR;              /*!< Offset:
      0x028 (R/W) Capture Control Register */
  __I uint32_t CR0;               /*!< Offset:
      0x02C (R/ ) Capture Register 0 */
  __I uint32_t CR1;               /*!< Offset:
      0x030 (R/ ) Capture Register */
      uint32_t RESERVED0[2];
  __IO uint32_t EMR;              /*!< Offset:
      0x03C (R/W) External Match Register */
      uint32_t RESERVED1[12];
  __IO uint32_t CTCR;             /*!< Offset:
      0x070 (R/W) Count Control Register */
} LPC_TIM_TypeDef;

/*------------- Pulse-Width Modulation (PWM)
    ----------------------------------*/
/** @brief Pulse-Width Modulation (PWM)
    register structure definition */
typedef struct
{
  __IO uint32_t IR;               /*!< Offset:
      0x000 (R/W) Interrupt Register */
  __IO uint32_t TCR;              /*!< Offset:
      0x004 (R/W) Timer Control Register.
      Register */
  __IO uint32_t TC;               /*!< Offset:
      0x008 (R/W) Timer Counter Register */
  __IO uint32_t PR;               /*!< Offset:
      0x00C (R/W) Prescale Register */
  __IO uint32_t PC;               /*!< Offset:
      0x010 (R/W) Prescale Counter Register */
  __IO uint32_t MCR;              /*!< Offset:
      0x014 (R/W) Match Control Register */
  __IO uint32_t MR0;              /*!< Offset:
      0x018 (R/W) Match Register 0 */
```

```
  __IO uint32_t MR1;              /*!< Offset:
     0x01C (R/W) Match Register 1 */
  __IO uint32_t MR2;              /*!< Offset:
     0x020 (R/W) Match Register 2 */
  __IO uint32_t MR3;              /*!< Offset:
     0x024 (R/W) Match Register 3 */
  __IO uint32_t CCR;              /*!< Offset:
     0x028 (R/W) Capture Control Register */
  __I uint32_t CR0;               /*!< Offset:
     0x02C (R/ ) Capture Register 0 */
  __I uint32_t CR1;               /*!< Offset:
     0x030 (R/ ) Capture Register 1 */
  __I uint32_t CR2;               /*!< Offset:
     0x034 (R/ ) Capture Register 2 */
  __I uint32_t CR3;               /*!< Offset:
     0x038 (R/ ) Capture Register 3 */
     uint32_t RESERVED0;
  __IO uint32_t MR4;              /*!< Offset:
     0x040 (R/W) Match Register 4 */
  __IO uint32_t MR5;              /*!< Offset:
     0x044 (R/W) Match Register 5 */
  __IO uint32_t MR6;              /*!< Offset:
     0x048 (R/W) Match Register 6 */
  __IO uint32_t PCR;              /*!< Offset:
     0x04C (R/W) PWM Control Register */
  __IO uint32_t LER;              /*!< Offset:
     0x050 (R/W) Load Enable Register */
     uint32_t RESERVED1[7];
  __IO uint32_t CTCR;             /*!< Offset:
     0x070 (R/W) Count Control Register */
} LPC_PWM_TypeDef;

/*------------- Universal Asynchronous
    Receiver Transmitter (UART) -----------*/
/** @brief Universal Asynchronous Receiver
    Transmitter (UART) register structure
    definition */
typedef struct
{
  union {
  __I uint32_t RBR;           /*!< Offset:
     0x000 Receiver Buffer Register (R/ ) */
  __O uint32_t THR;           /*!< Offset:
     0x000 Transmit Holding Register ( /W) */
  __IO uint32_t DLL;          /*!< Offset:
     0x000 Divisor Latch LSB (R/W) */
  };
  union {
  __IO uint32_t DLM;          /*!< Offset:
     0x004 Divisor Latch MSB (R/W) */
  __IO uint32_t IER;          /*!< Offset:
     0x004 Interrupt Enable Register (R/W) */
  };
  union {
  __I uint32_t IIR;           /*!< Offset:
     0x008 Interrupt ID Register (R/ ) */
  __O uint32_t FCR;           /*!< Offset:
     0x008 FIFO Control Register ( /W) */
  };
  __IO uint32_t LCR;          /*!< Offset:
     0x00C Line Control Register (R/W) */
     uint32_t RESERVED0;
  __I uint32_t LSR;           /*!< Offset:
     0x014 Line Status Register (R/ ) */
     uint32_t RESERVED1;
  __IO uint32_t SCR;          /*!< Offset:
     0x01C Scratch Pad Register (R/W) */
  __IO uint32_t ACR;          /*!< Offset:
     0x020 Auto-baud Control Register (R/W) */
  __IO uint32_t ICR;          /*!< Offset:
     0x024 IrDA Control Register (R/W) */
  __IO uint32_t FDR;          /*!< Offset:
     0x028 Fractional Divider Register (R/W)
     */
     uint32_t RESERVED2;
  __IO uint32_t TER;          /*!< Offset:
     0x030 Transmit Enable Register (R/W) */
} LPC_UART_TypeDef;

/** @brief Universal Asynchronous Receiver
    Transmitter 0 (UART0) register structure
    definition */
typedef struct
{
  union {
  __I uint32_t RBR;           /*!< Offset:
     0x000 Receiver Buffer Register (R/ ) */
  __O uint32_t THR;           /*!< Offset:
     0x000 Transmit Holding Register ( /W) */
  __IO uint32_t DLL;          /*!< Offset:
     0x000 Divisor Latch LSB (R/W) */
  };
  union {
  __IO uint32_t DLM;          /*!< Offset:
     0x004 Divisor Latch MSB (R/W) */
  __IO uint32_t IER;          /*!< Offset:
     0x000 Interrupt Enable Register (R/W) */
  };
  union {
  __I uint32_t IIR;           /*!< Offset:
     0x008 Interrupt ID Register (R/ ) */
  __O uint32_t FCR;           /*!< Offset:
     0x008 FIFO Control Register ( /W) */
  };
  __IO uint32_t LCR;          /*!< Offset:
     0x00C Line Control Register (R/W) */
  __IO uint32_t MCR;          /*!< Offset:
     0x010 Modem control Register (R/W) */
  __I uint32_t LSR;           /*!< Offset:
     0x014 Line Status Register (R/ ) */
  __I uint32_t MSR;           /*!< Offset:
     0x018 Modem status Register (R/ ) */
  __IO uint32_t SCR;          /*!< Offset:
     0x01C Scratch Pad Register (R/W) */
  __IO uint32_t ACR;          /*!< Offset:
     0x020 Auto-baud Control Register (R/W) */
     uint32_t RESERVED0;
  __IO uint32_t FDR;          /*!< Offset:
     0x028 Fractional Divider Register (R/W)
     */
     uint32_t RESERVED1;
  __IO uint32_t TER;          /*!< Offset:
     0x030 Transmit Enable Register (R/W) */
     uint32_t RESERVED2[6];
  __IO uint32_t RS485CTRL;    /*!< Offset:
     0x04C RS-485/EIA-485 Control Register
     (R/W) */
  __IO uint32_t ADRMATCH;     /*!< Offset:
     0x050 RS-485/EIA-485 address match
     Register (R/W) */
  __IO uint32_t RS485DLY;     /*!< Offset:
     0x054 RS-485/EIA-485 direction control
     delay Register (R/W) */
} LPC_UART1_TypeDef;
```

```
/*------------- Serial Peripheral Interface
    (SPI) ----------------------------*/
/** @brief Serial Peripheral Interface (SPI)
    register structure definition */
typedef struct
{
  __IO uint32_t SPCR;             /*!< Offset:
     0x000 SPI Control Register (R/W) */
  __I uint32_t SPSR;              /*!< Offset:
     0x004 SPI Status Register (R/) */
  __IO uint32_t SPDR;             /*!< Offset:
     0x008 SPI Data Register (R/W) */
  __IO uint32_t SPCCR;            /*!< Offset:
     0x00C SPI Clock Counter Register (R/W) */
     uint32_t RESERVED0[3];
  __IO uint32_t SPINT;            /*!< Offset:
     0x01C SPI Interrupt Flag Register (R/W)
     */
} LPC_SPI_TypeDef;

/*------------- Synchronous Serial
    Communication (SSP)
    -----------------------*/
/** @brief Synchronous Serial Communication
    (SSP) register structure definition */
typedef struct
{
  __IO uint32_t CR0;              /*!< Offset:
     0x000 (R/W) Control Register 0 */
  __IO uint32_t CR1;              /*!< Offset:
     0x004 (R/W) Control Register 1 */
  __IO uint32_t DR;               /*!< Offset:
     0x008 (R/W) Data Register */
  __I uint32_t SR;                /*!< Offset:
     0x00C (R/ ) Status Register */
  __IO uint32_t CPSR;             /*!< Offset:
     0x010 (R/W) Clock Prescale Register */
  __IO uint32_t IMSC;             /*!< Offset:
     0x014 (R/W) Interrupt Mask Set and Clear
     Register */
  __IO uint32_t RIS;              /*!< Offset:
     0x018 (R/W) Raw Interrupt Status
     Register */
  __IO uint32_t MIS;              /*!< Offset:
     0x01C (R/W) Masked Interrupt Status
     Register */
  __IO uint32_t ICR;              /*!< Offset:
     0x020 (R/W) SSPICR Interrupt Clear
     Register */
  __IO uint32_t DMACR;            /*!< Offset:
     0x024 (R/W) DMA Control Register */
} LPC_SSP_TypeDef;

/*------------- Inter-Integrated Circuit
    (I2C) --------------------------------*/
/** @brief Inter-Integrated Circuit (I2C)
    register structure definition */
typedef struct
{
  __IO uint32_t CONSET;           /*!< Offset:
     0x000 (R/W) I2C Control Set Register */
  __I uint32_t STAT;              /*!< Offset:
     0x004 (R/ ) I2C Status Register */
  __IO uint32_t DAT;              /*!< Offset:
     0x008 (R/W) I2C Data Register */
  __IO uint32_t ADR0;             /*!< Offset:
     0x00C (R/W) I2C Slave Address Register 0
     */
  __IO uint32_t SCLH;             /*!< Offset:
     0x010 (R/W) SCH Duty Cycle Register High
     Half Word */
  __IO uint32_t SCLL;             /*!< Offset:
     0x014 (R/W) SCL Duty Cycle Register Low
     Half Word */
  __O uint32_t CONCLR;            /*!< Offset:
     0x018 (R/W) I2C Control Clear Register */
  __IO uint32_t MMCTRL;           /*!< Offset:
     0x01C (R/W) Monitor mode control
     register */
  __IO uint32_t ADR1;             /*!< Offset:
     0x020 (R/W) I2C Slave Address Register 1
     */
  __IO uint32_t ADR2;             /*!< Offset:
     0x024 (R/W) I2C Slave Address Register 2
     */
  __IO uint32_t ADR3;             /*!< Offset:
     0x028 (R/W) I2C Slave Address Register 3
     */
  __I uint32_t DATA_BUFFER;       /*!< Offset:
     0x02C (R/ ) Data buffer Register */
  __IO uint32_t MASK0;            /*!< Offset:
     0x030 (R/W) I2C Slave address mask
     register 0 */
  __IO uint32_t MASK1;            /*!< Offset:
     0x034 (R/W) I2C Slave address mask
     register 1 */
  __IO uint32_t MASK2;            /*!< Offset:
     0x038 (R/W) I2C Slave address mask
     register 2 */
  __IO uint32_t MASK3;            /*!< Offset:
     0x03C (R/W) I2C Slave address mask
     register 3 */
} LPC_I2C_TypeDef;

/*------------- Inter IC Sound (I2S)
    ----------------------------------------*/
/** @brief Inter IC Sound (I2S) register
    structure definition */
typedef struct
{
  __IO uint32_t DAO;              /*!< Offset:
     0x000 (R/W) Digital Audio Output
     Register */
  __IO uint32_t DAI;              /*!< Offset:
     0x004 (R/W) Digital Audio Input Register
     */
  __O uint32_t TXFIFO;            /*!< Offset:
     0x008 ( /W) Transmit FIFO */
  __I uint32_t RXFIFO;            /*!< Offset:
     0x00C (R/ ) Receive FIFO */
  __I uint32_t STATE;             /*!< Offset:
     0x010 (R/W) Status Feedback Register */
  __IO uint32_t DMA1;             /*!< Offset:
     0x014 (R/W) DMA Configuration Register 1
     */
  __IO uint32_t DMA2;             /*!< Offset:
     0x018 (R/W) DMA Configuration Register 2
     */
  __IO uint32_t IRQ;              /*!< Offset:
     0x01C (R/W) Interrupt Request Control
     Register */
```

```
  __IO uint32_t TXRATE;          /*!< Offset:
     0x020 (R/W) Transmit reference clock
     divider Register */
  __IO uint32_t RXRATE;          /*!< Offset:
     0x024 (R/W) Receive reference clock
     divider Register */
  __IO uint32_t TXBITRATE;       /*!< Offset:
     0x028 (R/W) Transmit bit rate divider
     Register */
  __IO uint32_t RXBITRATE;       /*!< Offset:
     0x02C (R/W) Receive bit rate divider
     Register */
  __IO uint32_t TXMODE;          /*!< Offset:
     0x030 (R/W) Transmit mode control
     Register */
  __IO uint32_t RXMODE;          /*!< Offset:
     0x034 (R/W) Receive mode control
     Register */
} LPC_I2S_TypeDef;

/*------------- Repetitive Interrupt Timer
    (RIT) -----------------------------*/
/** @brief Repetitive Interrupt Timer (RIT)
    register structure definition */
typedef struct
{
  __IO uint32_t RICOMPVAL;
  __IO uint32_t RIMASK;
  __IO uint32_t RICTRL;
  __IO uint32_t RICOUNTER;
} LPC_RIT_TypeDef;

/*------------- Real-Time Clock (RTC)
    ----------------------------------------*/
/** @brief Real-Time Clock (RTC) register
    structure definition */
typedef struct
{
  __IO uint32_t ILR;             /*!< Offset:
     0x000 (R/W) Interrupt Location Register
     */
     uint32_t RESERVED0;
  __IO uint32_t CCR;             /*!< Offset:
     0x008 (R/W) Clock Control Register */
  __IO uint32_t CIIR;            /*!< Offset:
     0x00C (R/W) Counter Increment Interrupt
     Register */
  __IO uint32_t AMR;             /*!< Offset:
     0x010 (R/W) Alarm Mask Register */
  __I uint32_t CTIME0;           /*!< Offset:
     0x014 (R/ ) Consolidated Time Register 0
     */
  __I uint32_t CTIME1;           /*!< Offset:
     0x018 (R/ ) Consolidated Time Register 1
     */
  __I uint32_t CTIME2;           /*!< Offset:
     0x01C (R/ ) Consolidated Time Register 2
     */
  __IO uint32_t SEC;             /*!< Offset:
     0x020 (R/W) Seconds Counter Register */
  __IO uint32_t MIN;             /*!< Offset:
     0x024 (R/W) Minutes Register */
  __IO uint32_t HOUR;            /*!< Offset:
     0x028 (R/W) Hours Register */
  __IO uint32_t DOM;             /*!< Offset:
     0x02C (R/W) Day of Month Register */
  __IO uint32_t DOW;             /*!< Offset:
     0x030 (R/W) Day of Week Register */
  __IO uint32_t DOY;             /*!< Offset:
     0x034 (R/W) Day of Year Register */
  __IO uint32_t MONTH;           /*!< Offset:
     0x038 (R/W) Months Register */
  __IO uint32_t YEAR;            /*!< Offset:
     0x03C (R/W) Years Register */
  __IO uint32_t CALIBRATION;     /*!< Offset:
     0x040 (R/W) Calibration Value Register */
  __IO uint32_t GPREG0;          /*!< Offset:
     0x044 (R/W) General Purpose Register 0 */
  __IO uint32_t GPREG1;          /*!< Offset:
     0x048 (R/W) General Purpose Register 1 */
  __IO uint32_t GPREG2;          /*!< Offset:
     0x04C (R/W) General Purpose Register 2 */
  __IO uint32_t GPREG3;          /*!< Offset:
     0x050 (R/W) General Purpose Register 3 */
  __IO uint32_t GPREG4;          /*!< Offset:
     0x054 (R/W) General Purpose Register 4 */
  __IO uint32_t RTC_AUXEN;       /*!< Offset:
     0x058 (R/W) RTC Auxiliary Enable
     Register */
  __IO uint32_t RTC_AUX;         /*!< Offset:
     0x05C (R/W) RTC Auxiliary Control
     Register */
  __IO uint32_t ALSEC;           /*!< Offset:
     0x060 (R/W) Alarm value for Seconds */
  __IO uint32_t ALMIN;           /*!< Offset:
     0x064 (R/W) Alarm value for Minutes */
  __IO uint32_t ALHOUR;          /*!< Offset:
     0x068 (R/W) Alarm value for Hours */
  __IO uint32_t ALDOM;           /*!< Offset:
     0x06C (R/W) Alarm value for Day of Month
     */
  __IO uint32_t ALDOW;           /*!< Offset:
     0x070 (R/W) Alarm value for Day of Week
     */
  __IO uint32_t ALDOY;           /*!< Offset:
     0x074 (R/W) Alarm value for Day of Year
     */
  __IO uint32_t ALMON;           /*!< Offset:
     0x078 (R/W) Alarm value for Months */
  __IO uint32_t ALYEAR;          /*!< Offset:
     0x07C (R/W) Alarm value for Year */
} LPC_RTC_TypeDef;

/*------------- Watchdog Timer (WDT)
    -----------------------------------------*/
/** @brief Watchdog Timer (WDT) register
    structure definition */
typedef struct
{
  __IO uint32_t MOD;             /*!< Offset:
     0x000 (R/W) Watchdog mode Register */
  __IO uint32_t TC;              /*!< Offset:
     0x004 (R/W) Watchdog timer constant
     Register */
  __O uint32_t FEED;             /*!< Offset:
     0x008 ( /W) Watchdog feed sequence
     Register */
  __I uint32_t TV;               /*!< Offset:
     0x00C (R/ ) Watchdog timer value
     Register */
  __IO uint32_t WDCLKSEL;
} LPC_WDT_TypeDef;
```

```
/*------------- Analog-to-Digital Converter
    (ADC) -----------------------------*/
/** @brief Analog-to-Digital Converter (ADC)
    register structure definition */
typedef struct
{
  __IO uint32_t CR;               /*!< Offset:
      0x000 (R/W) A/D Control Register */
  __IO uint32_t GDR;              /*!< Offset:
      0x004 (R/W) A/D Global Data Register */
      uint32_t RESERVED0;
  __IO uint32_t INTEN;            /*!< Offset:
      0x00C (R/W) A/D Interrupt Enable
      Register */
  __I uint32_t DR[8];             /*!< Offset:
      0x010 (R/ ) A/D Channel # Data Register
      */
  __I uint32_t STAT;              /*!< Offset:
      0x030 (R/ ) A/D Status Register */
  __IO uint32_t ADTRM;            /*!< Offset:
      0x034 (R/W) ADC trim Register */
   __IO uint32_t ADCR;
  __IO uint32_t ADGDR;
  __IO uint32_t ADINTEN;
  __I uint32_t ADDR0;
  __I uint32_t ADDR1;
  __I uint32_t ADDR2;
  __I uint32_t ADDR3;
  __I uint32_t ADDR4;
  __I uint32_t ADDR5;
  __I uint32_t ADDR6;
  __I uint32_t ADDR7;
  __I uint32_t ADSTAT;
} LPC_ADC_TypeDef;

/*------------- Digital-to-Analog Converter
    (DAC) -----------------------------*/
/** @brief Digital-to-Analog Converter (DAC)
    register structure definition */
typedef struct
{
  __IO uint32_t CR;               /*!< Offset:
      0x000 (R/W) D/A Converter Register */
  __IO uint32_t CTRL;             /*!< Offset:
      0x004 (R/W) DAC Control register */
  __IO uint32_t CNTVAL;           /*!< Offset:
      0x008 (R/W) DAC Counter Value Register */
    __IO uint32_t DACR;
  __IO uint32_t DACCTRL;
  __IO uint16_t DACCNTVAL;
} LPC_DAC_TypeDef;


/*------------- Motor Control Pulse-Width
    Modulation (MCPWM) ------------------*/
/** @brief Motor Control Pulse-Width
    Modulation (MCPWM) register structure
    definition */
typedef struct
{
  __I uint32_t CON;               /*!< Offset:
      0x000 (R/ ) PWM Control read address
      Register */
  __O uint32_t CON_SET;           /*!< Offset:
      0x004 ( /W) PWM Control set address
      Register */
  __O uint32_t CON_CLR;           /*!< Offset:
      0x008 ( /W) PWM Control clear address
      Register */
  __I uint32_t CAPCON;            /*!< Offset:
      0x00C (R/ ) Capture Control read address
      Register */
  __O uint32_t CAPCON_SET;        /*!< Offset:
      0x010 ( /W) Capture Control set address
      Register */
  __O uint32_t CAPCON_CLR;        /*!< Offset:
      0x014 ( /W) Event Control clear address
      Register */
  __IO uint32_t TC0;              /*!< Offset:
      0x018 (R/W) Timer Counter Register,
      channel 0 */
  __IO uint32_t TC1;              /*!< Offset:
      0x01C (R/W) Timer Counter Register,
      channel 1 */
  __IO uint32_t TC2;              /*!< Offset:
      0x020 (R/W) Timer Counter Register,
      channel 2 */
  __IO uint32_t LIM0;             /*!< Offset:
      0x024 (R/W) Limit Register, channel 0 */
  __IO uint32_t LIM1;             /*!< Offset:
      0x028 (R/W) Limit Register, channel 1 */
  __IO uint32_t LIM2;             /*!< Offset:
      0x02C (R/W) Limit Register, channel 2 */
  __IO uint32_t MAT0;             /*!< Offset:
      0x030 (R/W) Match Register, channel 0 */
  __IO uint32_t MAT1;             /*!< Offset:
      0x034 (R/W) Match Register, channel 1 */
  __IO uint32_t MAT2;             /*!< Offset:
      0x038 (R/W) Match Register, channel 2 */
  __IO uint32_t DT;               /*!< Offset:
      0x03C (R/W) Dead time Register */
  __IO uint32_t CP;               /*!< Offset:
      0x040 (R/W) Commutation Pattern Register
      */
  __IO uint32_t CAP0;             /*!< Offset:
      0x044 (R/W) Capture Register, channel 0
      */
  __IO uint32_t CAP1;             /*!< Offset:
      0x048 (R/W) Capture Register, channel 1
      */
  __IO uint32_t CAP2;             /*!< Offset:
      0x04C (R/W) Capture Register, channel 2
      */
  __I uint32_t INTEN;             /*!< Offset:
      0x050 (R/ ) Interrupt Enable read
      Register */
  __O uint32_t INTEN_SET;         /*!< Offset:
      0x054 ( /W) Interrupt Enable set address
      Register */
  __O uint32_t INTEN_CLR;         /*!< Offset:
      0x058 ( /W) Interrupt Enable clear
      address Register */
  __I uint32_t CNTCON;            /*!< Offset:
      0x05C (R/ ) Count Control read address
      Register */
  __O uint32_t CNTCON_SET;        /*!< Offset:
      0x060 ( /W) Count Control set address
      Register */
  __O uint32_t CNTCON_CLR;        /*!< Offset:
      0x064 ( /W) Count Control clear address
      Register */
  __I uint32_t INTF;              /*!< Offset:
      0x068 (R/ ) Interrupt flags read address
```

```
    Register */
  __O uint32_t INTF_SET;          /*!< Offset:
    0x06C ( /W) Interrupt flags set address
    Register */
  __O uint32_t INTF_CLR;          /*!< Offset:
    0x070 ( /W) Interrupt flags clear
    address Register */
  __O uint32_t CAP_CLR;           /*!< Offset:
    0x074 ( /W) Capture clear address
    Register */
} LPC_MCPWM_TypeDef;

/*------------- Quadrature Encoder Interface
    (QEI) ---------------------------*/
/** @brief Quadrature Encoder Interface (QEI)
    register structure definition */
typedef struct
{
  __O uint32_t CON;               /*!< Offset:
    0x000 ( /W) Control Register */
  __I uint32_t STAT;              /*!< Offset:
    0x004 (R/ ) Encoder Status Register */
  __IO uint32_t CONF;             /*!< Offset:
    0x008 (R/W) Configuration Register */
  __I uint32_t POS;               /*!< Offset:
    0x00C (R/ ) Position Register */
  __IO uint32_t MAXPOS;           /*!< Offset:
    0x010 (R/W) Maximum position Register */
  __IO uint32_t CMPOS0;           /*!< Offset:
    0x014 (R/W) Position compare Register 0
    */
  __IO uint32_t CMPOS1;           /*!< Offset:
    0x018 (R/W) Position compare Register 1
    */
  __IO uint32_t CMPOS2;           /*!< Offset:
    0x01C (R/W) Position compare Register 2
    */
  __I uint32_t INXCNT;            /*!< Offset:
    0x020 (R/ ) Index count Register */
  __IO uint32_t INXCMP0;          /*!< Offset:
    0x024 (R/W) Index compare Register 0 */
  __IO uint32_t LOAD;             /*!< Offset:
    0x028 (R/W) Velocity timer reload
    Register */
  __I uint32_t TIME;              /*!< Offset:
    0x02C (R/ ) Velocity timer Register */
  __I uint32_t VEL;               /*!< Offset:
    0x030 (R/ ) Velocity counter Register */
  __I uint32_t CAP;               /*!< Offset:
    0x034 (R/ ) Velocity capture Register */
  __IO uint32_t VELCOMP;          /*!< Offset:
    0x038 (R/W) Velocity compare Register */
  __IO uint32_t FILTER;
    uint32_t RESERVED0[998];
  __O uint32_t IEC;               /*!< Offset:
    0xFD8 ( /W) Interrupt enable clear
    Register */
  __O uint32_t IES;               /*!< Offset:
    0xFDC ( /W) Interrupt enable set
    Register */
  __I uint32_t INTSTAT;           /*!< Offset:
    0xFE0 (R/ ) Interrupt status Register */
  __I uint32_t IE;                /*!< Offset:
    0xFE4 (R/ ) Interrupt enable Register */
  __O uint32_t CLR;               /*!< Offset:
    0xFE8 ( /W) Interrupt status clear
    Register */
  __O uint32_t SET;               /*!< Offset:
    0xFEC ( /W) Interrupt status set
    Register */
} LPC_QEI_TypeDef;

/*------------- Controller Area Network (CAN)
    ---------------------------------*/
/** @brief Controller Area Network Acceptance
    Filter RAM (CANAF_RAM)structure
    definition */
typedef struct
{
  __IO uint32_t mask[512];        /*!< Offset:
    0x000 (R/W) Acceptance Filter RAM */
} LPC_CANAF_RAM_TypeDef;

/** @brief Controller Area Network Acceptance
    Filter(CANAF) register structure
    definition */
typedef struct                  /* Acceptance
    Filter Registers */
{
  __IO uint32_t AFMR;             /*!< Offset:
    0x000 (R/W) Acceptance Filter Register */
  __IO uint32_t SFF_sa;           /*!< Offset:
    0x004 (R/W) Standard Frame Individual
    Start Address Register */
  __IO uint32_t SFF_GRP_sa;       /*!< Offset:
    0x008 (R/W) Standard Frame Group Start
    Address Register */
  __IO uint32_t EFF_sa;           /*!< Offset:
    0x00C (R/W) Extended Frame Start Address
    Register */
  __IO uint32_t EFF_GRP_sa;       /*!< Offset:
    0x010 (R/W) Extended Frame Group Start
    Address Register */
  __IO uint32_t ENDofTable;       /*!< Offset:
    0x014 (R/W) End of AF Tables Register */
  __I uint32_t LUTerrAd;          /*!< Offset:
    0x018 (R/ ) LUT Error Address Register */
  __I uint32_t LUTerr;            /*!< Offset:
    0x01C (R/ ) LUT Error Register */
  __IO uint32_t FCANIE;           /*!< Offset:
    0x020 (R/W) Global FullCANInterrupt
    Enable Register */
  __IO uint32_t FCANIC0;          /*!< Offset:
    0x024 (R/W) FullCAN Interrupt and
    Capture Register 0 */
  __IO uint32_t FCANIC1;          /*!< Offset:
    0x028 (R/W) FullCAN Interrupt and
    Capture Register 1 */
} LPC_CANAF_TypeDef;

/** @brief Controller Area Network Central
    (CANCR) register structure definition */
typedef struct                  /* Central
    Registers        */
{
  __I uint32_t TxSR;              /*!< Offset:
    0x000 (R/ ) CAN Central Transmit Status
    Register */
  __I uint32_t RxSR;              /*!< Offset:
    0x004 (R/ ) CAN Central Receive Status
    Register */
  __I uint32_t MSR;               /*!< Offset:
    0x008 (R/ ) CAN Central Miscellaneous
    Register */
```

```
} LPC_CANCR_TypeDef;

/** @brief Controller Area Network Controller
    (CAN) register structure definition */
typedef struct                  /*
    Controller Registers      */
{
  __IO uint32_t MOD;            /*!< Offset:
      0x000 (R/W) CAN Mode Register */
  __O uint32_t CMR;             /*!< Offset:
      0x004 ( /W) CAN Command Register */
  __IO uint32_t GSR;            /*!< Offset:
      0x008 (R/W) CAN Global Status Register */
  __I uint32_t ICR;             /*!< Offset:
      0x00C (R/ ) CAN Interrupt and Capture
      Register */
  __IO uint32_t IER;            /*!< Offset:
      0x010 (R/W) CAN Interrupt Enable
      Register */
  __IO uint32_t BTR;            /*!< Offset:
      0x014 (R/W) CAN Bus Timing Register */
  __IO uint32_t EWL;            /*!< Offset:
      0x018 (R/W) CAN Error Warning Limit
      Register */
  __I uint32_t SR;              /*!< Offset:
      0x01C (R/ ) CAN Status Register */
  __IO uint32_t RFS;            /*!< Offset:
      0x020 (R/W) CAN Receive Frame Status
      Register */
  __IO uint32_t RID;            /*!< Offset:
      0x024 (R/W) CAN Receive Identifier
      Register */
  __IO uint32_t RDA;            /*!< Offset:
      0x028 (R/W) CAN Receive Data Register A
      */
  __IO uint32_t RDB;            /*!< Offset:
      0x02C (R/W) CAN Receive Data Register B
      */
  __IO uint32_t TFI1;           /*!< Offset:
      0x030 (R/W) CAN Transmit Frame
      Information Register 1 */
  __IO uint32_t TID1;           /*!< Offset:
      0x034 (R/W) CAN Transmit Identifier
      Register 1 */
  __IO uint32_t TDA1;           /*!< Offset:
      0x038 (R/W) CAN Transmit Data Register A
      1 */
  __IO uint32_t TDB1;           /*!< Offset:
      0x03C (R/W) CAN Transmit Data Register B
      1 */
  __IO uint32_t TFI2;           /*!< Offset:
      0x040 (R/W) CAN Transmit Frame
      Information Register 2 */
  __IO uint32_t TID2;           /*!< Offset:
      0x044 (R/W) CAN Transmit Identifier
      Register 2 */
  __IO uint32_t TDA2;           /*!< Offset:
      0x048 (R/W) CAN Transmit Data Register A
      2 */
  __IO uint32_t TDB2;           /*!< Offset:
      0x04C (R/W) CAN Transmit Data Register B
      2 */
  __IO uint32_t TFI3;           /*!< Offset:
      0x050 (R/W) CAN Transmit Frame
      Information Register 3 */
  __IO uint32_t TID3;           /*!< Offset:
      0x054 (R/W) CAN Transmit Identifier
      Register 3 */
  __IO uint32_t TDA3;           /*!< Offset:
      0x058 (R/W) CAN Transmit Data Register A
      3 */
  __IO uint32_t TDB3;           /*!< Offset:
      0x05C (R/W) CAN Transmit Data Register B
      3 */
} LPC_CAN_TypeDef;

/*------------- General Purpose Direct Memory
    Access (GPDMA) ----------------*/
/** @brief General Purpose Direct Memory
    Access (GPDMA) register structure
    definition */
typedef struct                  /* Common
    Registers         */
{
  __I uint32_t IntStat;         /*!< Offset:
      0x000 (R/ ) DMA Interrupt Status
      Register */
  __I uint32_t IntTCStat;       /*!< Offset:
      0x004 (R/ ) DMA Interrupt Terminal Count
      Request Status Register */
  __O uint32_t IntTCClear;      /*!< Offset:
      0x008 ( /W) DMA Interrupt Terminal Count
      Request Clear Register */
  __I uint32_t IntErrStat;      /*!< Offset:
      0x00C (R/ ) DMA Interrupt Error Status
      Register */
  __O uint32_t IntErrClr;       /*!< Offset:
      0x010 ( /W) DMA Interrupt Error Clear
      Register */
  __I uint32_t RawIntTCStat;    /*!< Offset:
      0x014 (R/ ) DMA Raw Interrupt Terminal
      Count Status Register */
  __I uint32_t RawIntErrStat;   /*!< Offset:
      0x018 (R/ ) DMA Raw Error Interrupt
      Status Register */
  __I uint32_t EnbldChns;       /*!< Offset:
      0x01C (R/ ) DMA Enabled Channel Register
      */
  __IO uint32_t SoftBReq;       /*!< Offset:
      0x020 (R/W) DMA Software Burst Request
      Register */
  __IO uint32_t SoftSReq;       /*!< Offset:
      0x024 (R/W) DMA Software Single Request
      Register */
  __IO uint32_t SoftLBReq;      /*!< Offset:
      0x028 (R/W) DMA Software Last Burst
      Request Register */
  __IO uint32_t SoftLSReq;      /*!< Offset:
      0x02C (R/W) DMA Software Last Single
      Request Register */
  __IO uint32_t Config;         /*!< Offset:
      0x030 (R/W) DMA Configuration Register */
  __IO uint32_t Sync;           /*!< Offset:
      0x034 (R/W) DMA Synchronization Register
      */
} LPC_GPDMA_TypeDef;

/** @brief General Purpose Direct Memory
    Access Channel (GPDMACH) register
    structure definition */
typedef struct                  /* Channel
    Registers        */
{
```

```
  __IO uint32_t CSrcAddr;         /*!< Offset:
     0x000 (R/W) DMA Channel # Source Address
     Register */
  __IO uint32_t CDestAddr;        /*!< Offset:
     0x004 (R/W) DMA Channel # Destination
     Address Register */
  __IO uint32_t CLLI;             /*!< Offset:
     0x008 (R/W) DMA Channel # Linked List
     Item Register */
  __IO uint32_t CControl;         /*!< Offset:
     0x00C (R/W) DMA Channel # Control
     Register */
  __IO uint32_t CConfig;          /*!< Offset:
     0x010 (R/W) DMA Channel # Configuration
     Register */
} LPC_GPDMACH_TypeDef;

/*------------- Universal Serial Bus (USB)
    ----------------------------------*/
/** @brief Universal Serial Bus (USB)
    register structure definition */
typedef struct
{
  __I uint32_t Revision;          /*!< Offset:
     0x000 (R/ ) Revision Register */
  __IO uint32_t Control;          /*!< Offset:
     0x004 (R/W) Control Register */
  __IO uint32_t CommandStatus;    /*!< Offset:
     0x008 (R/W) Command / Status Register */
  __IO uint32_t InterruptStatus; /*!< Offset:
     0x00C (R/W) Interrupt Status Register */
  __IO uint32_t InterruptEnable; /*!< Offset:
     0x010 (R/W) Interrupt Enable Register */
  __IO uint32_t InterruptDisable; /*!< Offset:
     0x014 (R/W) Interrupt Disable Register */
  __IO uint32_t HCCA;             /*!< Offset:
     0x018 (R/W) Host Controller
     communication Area Register */
  __I uint32_t PeriodCurrentED; /*!< Offset:
     0x01C (R/ ) Register */
  __IO uint32_t ControlHeadED;    /*!< Offset:
     0x020 (R/W) Register */
  __IO uint32_t ControlCurrentED; /*!< Offset:
     0x024 (R/W) Register */
  __IO uint32_t BulkHeadED;       /*!< Offset:
     0x028 (R/W) Register */
  __IO uint32_t BulkCurrentED;    /*!< Offset:
     0x02C (R/W) Register */
  __I uint32_t DoneHead;          /*!< Offset:
     0x030 (R/ ) Register */
  __IO uint32_t FmInterval;       /*!< Offset:
     0x034 (R/W) Register */
  __I uint32_t FmRemaining;       /*!< Offset:
     0x038 (R/ ) Register */
  __I uint32_t FmNumber;          /*!< Offset:
     0x03C (R/ ) Register */
  __IO uint32_t PeriodicStart;    /*!< Offset:
     0x040 (R/W) Register */
  __IO uint32_t LSTreshold;       /*!< Offset:
     0x044 (R/W) Register */
  __IO uint32_t RhDescriptorA;    /*!< Offset:
     0x048 (R/W) Register */
  __IO uint32_t RhDescriptorB;    /*!< Offset:
     0x04C (R/W) Register */
  __IO uint32_t RhStatus;         /*!< Offset:
     0x050 (R/W) Register */
  __IO uint32_t RhPortStatus1;    /*!< Offset:
     0x054 (R/W) Register */
  __IO uint32_t RhPortStatus2;    /*!< Offset:
     0x05C (R/W) Register */
     uint32_t RESERVED0[40];
  __I uint32_t Module_ID;         /*!< Offset:
     0x0FC (R/ ) Module ID / Version
     Reverence ID Register */
                                  /* USB
                                     On-The-Go
                                     Registers
                                      */
  __I uint32_t IntSt;             /*!< Offset:
     0x100 (R/ ) OTG Interrupt Status
     Register */
  __IO uint32_t IntEn;            /*!< Offset:
     0x104 (R/W) OTG Interrupt Enable
     Register */
  __O uint32_t IntSet;            /*!< Offset:
     0x108 ( /W) OTG Interrupt Set Register */
  __O uint32_t IntClr;            /*!< Offset:
     0x10C ( /W) OTG Interrupt Clear Register
     */
  __IO uint32_t StCtrl;           /*!< Offset:
     0x110 (R/W) OTG Status and Control
     Register */
  __IO uint32_t Tmr;              /*!< Offset:
     0x114 (R/W) OTG Timer Register */
     uint32_t RESERVED1[58];
                                  /* USB Device
                                     Interrupt
                                     Registers
                                     */
  __I uint32_t DevIntSt;          /*!< Offset:
     0x200 (R/ ) USB Device Interrupt Status
     Register */
  __IO uint32_t DevIntEn;         /*!< Offset:
     0x204 (R/W) USB Device Interrupt Enable
     Register */
  __O uint32_t DevIntClr;         /*!< Offset:
     0x208 ( /W) USB Device Interrupt Clear
     Register */
  __O uint32_t DevIntSet;         /*!< Offset:
     0x20C ( /W) USB Device Interrupt Set
     Register */
                                  /* USB Device
                                     SIE
                                     Command
                                     Registers
                                     */
  __O uint32_t CmdCode;           /*!< Offset:
     0x210 (R/W) USB Command Code Register */
  __I uint32_t CmdData;           /*!< Offset:
     0x214 (R/W) USB Command Data Register */
                                  /* USB Device
                                     Transfer
                                     Registers
                                     */
  __I uint32_t RxData;            /*!< Offset:
     0x218 (R/ ) USB Receive Data Register */
  __O uint32_t TxData;            /*!< Offset:
     0x21C ( /W) USB Transmit Data Register */
  __I uint32_t RxPLen;            /*!< Offset:
     0x220 (R/ ) USB Receive Packet Length
     Register */
  __O uint32_t TxPLen;            /*!< Offset:
     0x224 ( /W) USB Transmit Packet Length
```

```
    Register */
__IO uint32_t Ctrl;           /*!< Offset:
    0x228 (R/W) USB Control Register */
__O uint32_t DevIntPri;       /*!< Offset:
    0x22C (R/W) USB Device Interrupt
    Priority Register */
                              /* USB Device
                                 Endpoint
                                 Interrupt
                                 Regs */
__I uint32_t EpIntSt;         /*!< Offset:
    0x230 (R/ ) USB Endpoint Interrupt
    Status Register */
__IO uint32_t EpIntEn;        /*!< Offset:
    0x234 (R/W) USB Endpoint Interrupt
    Enable Register */
__O uint32_t EpIntClr;        /*!< Offset:
    0x238 ( /W) USB Endpoint Interrupt Clear
    Register */
__O uint32_t EpIntSet;        /*!< Offset:
    0x23C ( /W) USB Endpoint Interrupt Set
    Register */
__O uint32_t EpIntPri;        /*!< Offset:
    0x240 ( /W) USB Endpoint Interrupt
    Priority Register */
                              /* USB Device
                                 Endpoint
                                 Realization
                                 Reg*/
__IO uint32_t ReEp;           /*!< Offset:
    0x244 (R/W) USB Realize Endpoint
    Register */
__O uint32_t EpInd;           /*!< Offset:
    0x248 ( /W) USB Endpoint Index Register
    */
__IO uint32_t MaxPSize;       /*!< Offset:
    0x24C (R/W) USB MaxPacketSize Register */
                              /* USB Device
                                 DMA
                                 Registers
                                 */
__I uint32_t DMARSt;          /*!< Offset:
    0x250 (R/ ) USB DMA Request Status
    Register */
__O uint32_t DMARClr;         /*!< Offset:
    0x254 ( /W) USB DMA Request Clear
    Register */
__O uint32_t DMARSet;         /*!< Offset:
    0x258 ( /W) USB DMA Request Set Register
    */
    uint32_t RESERVED2[9];
__IO uint32_t UDCAH;          /*!< Offset:
    0x280 (R/W) USB UDCA Head Register */
__I uint32_t EpDMASt;         /*!< Offset:
    0x284 (R/ ) USB EP DMA Status Register */
__O uint32_t EpDMAEn;         /*!< Offset:
    0x288 ( /W) USB EP DMA Enable Register */
__O uint32_t EpDMADis;        /*!< Offset:
    0x28C ( /W) USB EP DMA Disable Register
    */
__I uint32_t DMAIntSt;        /*!< Offset:
    0x290 (R/ ) USB DMA Interrupt Status
    Register */
__IO uint32_t DMAIntEn;       /*!< Offset:
    0x294 (R/W) USB DMA Interrupt Enable
    Register */
    uint32_t RESERVED3[2];
__I uint32_t EoTIntSt;        /*!< Offset:
    0x2A0 (R/ ) USB End of Transfer
    Interrupt Status Register */
__O uint32_t EoTIntClr;       /*!< Offset:
    0x2A4 ( /W) USB End of Transfer
    Interrupt Clear Register */
__O uint32_t EoTIntSet;       /*!< Offset:
    0x2A8 ( /W) USB End of Transfer
    Interrupt Set Register */
__I uint32_t NDDRIntSt;       /*!< Offset:
    0x2AC (R/ ) USB New DD Request Interrupt
    Status Register */
__O uint32_t NDDRIntClr;      /*!< Offset:
    0x2B0 ( /W) USB New DD Request Interrupt
    Clear Register */
__O uint32_t NDDRIntSet;      /*!< Offset:
    0x2B4 ( /W) USB New DD Request Interrupt
    Set Register */
__I uint32_t SysErrIntSt;     /*!< Offset:
    0x2B8 (R/ ) USB System Error Interrupt
    Status Register */
__O uint32_t SysErrIntClr;    /*!< Offset:
    0x2BC ( /W) USB System Error Interrupt
    Clear Register */
__O uint32_t SysErrIntSet;    /*!< Offset:
    0x2C0 ( /W) USB System Error Interrupt
    Set Register */
    uint32_t RESERVED4[15];
                              /* USB OTG I2C
                                 Registers
                                    */
  union {
  __I uint32_t I2C_RX;        /*!< Offset:
      0x300 (R/ ) OTG I2C Receive Register */
  __O uint32_t I2C_TX;        /*!< Offset:
      0x300 ( /W) OTG I2C Transmit Register */
  };
  __I uint32_t I2C_STS;       /*!< Offset:
      0x304 (R/ ) OTG I2C Status Register */
  __IO uint32_t I2C_CTL;      /*!< Offset:
      0x308 (R/W) OTG I2C Control Register */
  __IO uint32_t I2C_CLKHI;    /*!< Offset:
      0x30C (R/W) OTG I2C Clock High Register
      */
  __O uint32_t I2C_CLKLO;     /*!< Offset:
      0x310 ( /W) OTG I2C Clock Low Register */
      uint32_t RESERVED5[824];
                              /* USB Clock
                                 Control
                                 Registers
                                 */
  union {
  __IO uint32_t USBClkCtrl;   /*!< Offset:
      0xFF4 (R/W) OTG clock controller
      Register */
  __IO uint32_t OTGClkCtrl;   /*!< Offset:
      0xFF4 (R/W) USB clock controller
      Register */
  };
  union {
  __I uint32_t USBClkSt;      /*!< Offset:
      0xFF8 (R/ ) OTG clock status Register */
  __I uint32_t OTGClkSt;      /*!< Offset:
      0xFF8 (R/ ) USB clock status Register */
  };
} LPC_USB_TypeDef;
```

```
/*------------- Ethernet Media Access
    Controller (EMAC) ----------------------*/
/** @brief Ethernet Media Access Controller
    (EMAC) register structure definition */
typedef struct
{
  __IO uint32_t MAC1;             /*!< Offset:
     0x000 (R/W) MAC Configuration Register 1
     */
  __IO uint32_t MAC2;             /*!< Offset:
     0x004 (R/W) MAC Configuration Register 2
     */
  __IO uint32_t IPGT;             /*!< Offset:
     0x008 (R/W) Back-to-Back
     Inter-Packet-Gap Register */
  __IO uint32_t IPGR;             /*!< Offset:
     0x00C (R/W) Non Back-to-Back
     Inter-Packet-Gap Register */
  __IO uint32_t CLRT;             /*!< Offset:
     0x010 (R/W) Collision Window / Retry
     Register */
  __IO uint32_t MAXF;             /*!< Offset:
     0x014 (R/W) Maximum Frame Register */
  __IO uint32_t SUPP;             /*!< Offset:
     0x018 (R/W) PHY Support Register */
  __IO uint32_t TEST;             /*!< Offset:
     0x01C (R/W) Test Register */
  __IO uint32_t MCFG;             /*!< Offset:
     0x020 (R/W) MII Mgmt Configuration
     Register */
  __IO uint32_t MCMD;             /*!< Offset:
     0x024 (R/W) MII Mgmt Command Register */
  __IO uint32_t MADR;             /*!< Offset:
     0x028 (R/W) MII Mgmt Address Register */
  __O uint32_t MWTD;              /*!< Offset:
     0x02C ( /W) MII Mgmt Write Data Register
     */
  __I uint32_t MRDD;              /*!< Offset:
     0x030 (R/ ) MII Mgmt Read Data Register
     */
  __I uint32_t MIND;              /*!< Offset:
     0x034 (R/ ) MII Mgmt Indicators Register
     */
     uint32_t RESERVED0[2];
  __IO uint32_t SA0;              /*!< Offset:
     0x040 (R/W) Station Address 0 Register */
  __IO uint32_t SA1;              /*!< Offset:
     0x044 (R/W) Station Address 1 Register */
  __IO uint32_t SA2;              /*!< Offset:
     0x048 (R/W) Station Address 2 Register */
     uint32_t RESERVED1[45];
  __IO uint32_t Command;          /*!< Offset:
     0x100 (R/W) Command Register */
  __I uint32_t Status;            /*!< Offset:
     0x104 (R/ ) Status Register */
  __IO uint32_t RxDescriptor;     /*!< Offset:
     0x108 (R/W) Receive Descriptor Base
     Address Register */
  __IO uint32_t RxStatus;         /*!< Offset:
     0x10C (R/W) Receive Status Base Address
     Register */
  __IO uint32_t RxDescriptorNumber; /*!<
     Offset: 0x110 (R/W) Receive Number of
     Descriptors Register */
  __I uint32_t RxProduceIndex;    /*!< Offset:
     0x114 (R/ ) Receive Produce Index
     Register */
  __IO uint32_t RxConsumeIndex;   /*!< Offset:
     0x118 (R/W) Receive Consume Index
     Register */
  __IO uint32_t TxDescriptor;     /*!< Offset:
     0x11C (R/W) Transmit Descriptor Base
     Address Register */
  __IO uint32_t TxStatus;         /*!< Offset:
     0x120 (R/W) Transmit Status Base Address
     Register */
  __IO uint32_t TxDescriptorNumber; /*!<
     Offset: 0x124 (R/W) Transmit Number of
     Descriptors Register */
  __IO uint32_t TxProduceIndex;   /*!< Offset:
     0x128 (R/W) Transmit Produce Index
     Register */
  __I uint32_t TxConsumeIndex;    /*!< Offset:
     0x12C (R/ ) Transmit Consume Index
     Register */
     uint32_t RESERVED2[10];
  __I uint32_t TSV0;              /*!< Offset:
     0x158 (R/ ) Transmit Status Vector 0
     Register */
  __I uint32_t TSV1;              /*!< Offset:
     0x15C (R/ ) Transmit Status Vector 1
     Register */
  __I uint32_t RSV;               /*!< Offset:
     0x160 (R/ ) Receive Status Vector
     Register */
     uint32_t RESERVED3[3];
  __IO uint32_t FlowControlCounter; /*!<
     Offset: 0x170 (R/W) Flow Control Counter
     Register */
  __I uint32_t FlowControlStatus; /*!< Offset:
     0x174 (R/ ) Flow Control Status egister
     */
     uint32_t RESERVED4[34];
  __IO uint32_t RxFilterCtrl;     /*!< Offset:
     0x200 (R/W) Receive Filter Control
     Register */
  __I uint32_t RxFilterWoLStatus; /*!< Offset:
     0x204 (R/ ) Receive Filter WoL Status
     Register */
  __O uint32_t RxFilterWoLClear;  /*!< Offset:
     0x208 ( /W) Receive Filter WoL Clear
     Register */
     uint32_t RESERVED5;
  __IO uint32_t HashFilterL;      /*!< Offset:
     0x210 (R/W) Hash Filter Table LSBs
     Register */
  __IO uint32_t HashFilterH;      /*!< Offset:
     0x214 (R/W) Hash Filter Table MSBs
     Register */
     uint32_t RESERVED6[882];
  __I uint32_t IntStatus;         /*!< Offset:
     0xFE0 (R/ ) Interrupt Status Register */
  __IO uint32_t IntEnable;        /*!< Offset:
     0xFE4 (R/W) Interrupt Enable Register */
  __O uint32_t IntClear;          /*!< Offset:
     0xFE8 ( /W) Interrupt Clear Register */
  __O uint32_t IntSet;            /*!< Offset:
     0xFEC ( /W) Interrupt Set Register */
     uint32_t RESERVED7;
  __IO uint32_t PowerDown;        /*!< Offset:
     0xFF4 (R/W) Power-Down Register */
} LPC_EMAC_TypeDef;

#if defined ( __CC_ARM )
```

```
#pragma no_anon_unions
#endif


/*****************************************************************************
/*                  Peripheral memory map
                    */
/*****************************************************************************
/* Base addresses
                                                        */
#define LPC_FLASH_BASE (0x00000000UL)
#define LPC_RAM_BASE   (0x10000000UL)
#ifdef __LPC17XX_REV00
#define LPC_AHBRAM0_BASE (0x20000000UL)
#define LPC_AHBRAM1_BASE (0x20004000UL)
#else
#define LPC_AHBRAM0_BASE (0x2007C000UL)
#define LPC_AHBRAM1_BASE (0x20080000UL)
#endif
#define LPC_GPIO_BASE  (0x2009C000UL)
#define LPC_APB0_BASE  (0x40000000UL)
#define LPC_APB1_BASE  (0x40080000UL)
#define LPC_AHB_BASE   (0x50000000UL)
#define LPC_CM3_BASE   (0xE0000000UL)

/* APB0 peripherals
                                                      */
#define LPC_WDT_BASE   (LPC_APB0_BASE +
    0x00000)
#define LPC_TIM0_BASE  (LPC_APB0_BASE +
    0x04000)
#define LPC_TIM1_BASE  (LPC_APB0_BASE +
    0x08000)
#define LPC_UART0_BASE (LPC_APB0_BASE +
    0x0C000)
#define LPC_UART1_BASE (LPC_APB0_BASE +
    0x10000)
#define LPC_PWM1_BASE  (LPC_APB0_BASE +
    0x18000)
#define LPC_I2C0_BASE  (LPC_APB0_BASE +
    0x1C000)
#define LPC_SPI_BASE   (LPC_APB0_BASE +
    0x20000)
#define LPC_RTC_BASE   (LPC_APB0_BASE +
    0x24000)
#define LPC_GPIOINT_BASE (LPC_APB0_BASE +
    0x28080)
#define LPC_PINCON_BASE (LPC_APB0_BASE +
    0x2C000)
#define LPC_SSP1_BASE  (LPC_APB0_BASE +
    0x30000)
#define LPC_ADC_BASE   (LPC_APB0_BASE +
    0x34000)
#define LPC_CANAF_RAM_BASE (LPC_APB0_BASE +
    0x38000)
#define LPC_CANAF_BASE (LPC_APB0_BASE +
    0x3C000)
#define LPC_CANCR_BASE (LPC_APB0_BASE +
    0x40000)
#define LPC_CAN1_BASE  (LPC_APB0_BASE +
    0x44000)
#define LPC_CAN2_BASE  (LPC_APB0_BASE +
    0x48000)
#define LPC_I2C1_BASE  (LPC_APB0_BASE +
    0x5C000)

/* APB1 peripherals
                                                      */
#define LPC_SSP0_BASE  (LPC_APB1_BASE +
    0x08000)
#define LPC_DAC_BASE   (LPC_APB1_BASE +
    0x0C000)
#define LPC_TIM2_BASE  (LPC_APB1_BASE +
    0x10000)
#define LPC_TIM3_BASE  (LPC_APB1_BASE +
    0x14000)
#define LPC_UART2_BASE (LPC_APB1_BASE +
    0x18000)
#define LPC_UART3_BASE (LPC_APB1_BASE +
    0x1C000)
#define LPC_I2C2_BASE  (LPC_APB1_BASE +
    0x20000)
#define LPC_I2S_BASE   (LPC_APB1_BASE +
    0x28000)
#define LPC_RIT_BASE   (LPC_APB1_BASE +
    0x30000)
#define LPC_MCPWM_BASE (LPC_APB1_BASE +
    0x38000)
#define LPC_QEI_BASE   (LPC_APB1_BASE +
    0x3C000)
#define LPC_SC_BASE    (LPC_APB1_BASE +
    0x7C000)

/* AHB peripherals
                                                      */
#define LPC_EMAC_BASE  (LPC_AHB_BASE + 0x00000)
#define LPC_GPDMA_BASE (LPC_AHB_BASE +
    0x04000)
#define LPC_GPDMACH0_BASE (LPC_AHB_BASE +
    0x04100)
#define LPC_GPDMACH1_BASE (LPC_AHB_BASE +
    0x04120)
#define LPC_GPDMACH2_BASE (LPC_AHB_BASE +
    0x04140)
#define LPC_GPDMACH3_BASE (LPC_AHB_BASE +
    0x04160)
#define LPC_GPDMACH4_BASE (LPC_AHB_BASE +
    0x04180)
#define LPC_GPDMACH5_BASE (LPC_AHB_BASE +
    0x041A0)
#define LPC_GPDMACH6_BASE (LPC_AHB_BASE +
    0x041C0)
#define LPC_GPDMACH7_BASE (LPC_AHB_BASE +
    0x041E0)
#define LPC_USB_BASE   (LPC_AHB_BASE + 0x0C000)

/* GPIOs

    */
#define LPC_GPIO0_BASE (LPC_GPIO_BASE +
    0x00000)
#define LPC_GPIO1_BASE (LPC_GPIO_BASE +
    0x00020)
#define LPC_GPIO2_BASE (LPC_GPIO_BASE +
    0x00040)
#define LPC_GPIO3_BASE (LPC_GPIO_BASE +
    0x00060)
#define LPC_GPIO4_BASE (LPC_GPIO_BASE +
    0x00080)


/*****************************************************
```

```
/*                  Peripheral declaration                  */
/*****************************************************************/
#define LPC_SC          ((LPC_SC_TypeDef  *)
    LPC_SC_BASE  )
#define LPC_GPIO0       ((LPC_GPIO_TypeDef *)
    LPC_GPIO0_BASE )
#define LPC_GPIO1       ((LPC_GPIO_TypeDef *)
    LPC_GPIO1_BASE )
#define LPC_GPIO2       ((LPC_GPIO_TypeDef *)
    LPC_GPIO2_BASE )
#define LPC_GPIO3       ((LPC_GPIO_TypeDef *)
    LPC_GPIO3_BASE )
#define LPC_GPIO4       ((LPC_GPIO_TypeDef *)
    LPC_GPIO4_BASE )
#define LPC_WDT         ((LPC_WDT_TypeDef *)
    LPC_WDT_BASE )
#define LPC_TIM0        ((LPC_TIM_TypeDef *)
    LPC_TIM0_BASE )
#define LPC_TIM1        ((LPC_TIM_TypeDef *)
    LPC_TIM1_BASE )
#define LPC_TIM2        ((LPC_TIM_TypeDef *)
    LPC_TIM2_BASE )
#define LPC_TIM3        ((LPC_TIM_TypeDef *)
    LPC_TIM3_BASE )
#define LPC_RIT         ((LPC_RIT_TypeDef *)
    LPC_RIT_BASE )
#define LPC_UART0       ((LPC_UART_TypeDef *)
    LPC_UART0_BASE )
#define LPC_UART1       ((LPC_UART1_TypeDef *)
    LPC_UART1_BASE )
#define LPC_UART2       ((LPC_UART_TypeDef *)
    LPC_UART2_BASE )
#define LPC_UART3       ((LPC_UART_TypeDef *)
    LPC_UART3_BASE )
#define LPC_PWM1        ((LPC_PWM_TypeDef *)
    LPC_PWM1_BASE )
#define LPC_I2C0        ((LPC_I2C_TypeDef *)
    LPC_I2C0_BASE )
#define LPC_I2C1        ((LPC_I2C_TypeDef *)
    LPC_I2C1_BASE )
#define LPC_I2C2        ((LPC_I2C_TypeDef *)
    LPC_I2C2_BASE )
#define LPC_I2S         ((LPC_I2S_TypeDef *)
    LPC_I2S_BASE )
#define LPC_SPI         ((LPC_SPI_TypeDef *)
    LPC_SPI_BASE )
#define LPC_RTC         ((LPC_RTC_TypeDef *)
    LPC_RTC_BASE )
#define LPC_GPIOINT     ((LPC_GPIOINT_TypeDef
    *) LPC_GPIOINT_BASE )
#define LPC_PINCON      ((LPC_PINCON_TypeDef *)
    LPC_PINCON_BASE )
#define LPC_SSP0        ((LPC_SSP_TypeDef *)
    LPC_SSP0_BASE )
#define LPC_SSP1        ((LPC_SSP_TypeDef *)
    LPC_SSP1_BASE )
#define LPC_ADC         ((LPC_ADC_TypeDef *)
    LPC_ADC_BASE )
#define LPC_DAC         ((LPC_DAC_TypeDef *)
    LPC_DAC_BASE )
#define LPC_CANAF_RAM   ((LPC_CANAF_RAM_TypeDef
    *) LPC_CANAF_RAM_BASE)
#define LPC_CANAF       ((LPC_CANAF_TypeDef *)
    LPC_CANAF_BASE )
#define LPC_CANCR       ((LPC_CANCR_TypeDef *)
    LPC_CANCR_BASE )
#define LPC_CAN1        ((LPC_CAN_TypeDef *)
    LPC_CAN1_BASE )
#define LPC_CAN2        ((LPC_CAN_TypeDef *)
    LPC_CAN2_BASE )
#define LPC_MCPWM       ((LPC_MCPWM_TypeDef *)
    LPC_MCPWM_BASE )
#define LPC_QEI         ((LPC_QEI_TypeDef *)
    LPC_QEI_BASE )
#define LPC_EMAC        ((LPC_EMAC_TypeDef *)
    LPC_EMAC_BASE )
#define LPC_GPDMA       ((LPC_GPDMA_TypeDef *)
    LPC_GPDMA_BASE )
#define LPC_GPDMACH0    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH0_BASE )
#define LPC_GPDMACH1    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH1_BASE )
#define LPC_GPDMACH2    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH2_BASE )
#define LPC_GPDMACH3    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH3_BASE )
#define LPC_GPDMACH4    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH4_BASE )
#define LPC_GPDMACH5    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH5_BASE )
#define LPC_GPDMACH6    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH6_BASE )
#define LPC_GPDMACH7    ((LPC_GPDMACH_TypeDef
    *) LPC_GPDMACH7_BASE )
#define LPC_USB         ((LPC_USB_TypeDef *)
    LPC_USB_BASE )

/**
 * @}
 */

#endif // __LPC17xx_H__
```