Media Center

*COE718: Embedded System Design

Syed Maisam Abbas
ECB Department (Electrical, Computer, and Biomedical Engineering))
Toronto Metropolitan University - Electrical Engineering (BEng))
Toronto, Ontario
maisam.abbas@torontomu.ca

***Abstract*—Using the MCB1700 board, uVision, and every programming idea you have studied this semester, the goal of this final project is to build a media center. A photo gallery that can display multiple BMP files, an MP3 player that streams music and MP3 files from the PC, and a gaming center with one or more animated games the user can play are among the features of the media center.**

***Index Terms*—Hardware, Software, Real-time, MCB1700, LPC1768, Assembly/C, MP3 Player, Photo Gallery, Game**

I. Introduction

The general requirements for this media center project call for a graphical user interface (GUI) that is shown on an LCD screen and provides the user with a menu of navigable features. The MCB1700's joystick must be used largely for navigation. A photo gallery, an MP3 player, and at least one game are the essential features, though other features are optional. The design should enable interactive use (e.g., using the joystick for browsing) by enabling the user to pick any menu choice with the joystick, at which time the system must do the appropriate task. The system must automatically return to the main menu after the task or user interaction is finished. Using LEDs to provide visual feedback during execution or navigation, and integrating a USB keyboard designed for gaming.

This project introduces the Keil uVision IDE and some of its features using the ARM Cortex-M3 embedded processor. The ARM Cortex-M3 embedded processor's barrel shifting, conditional branching, and bit banding are the concepts that designers are expected to develop as per the learning objectives for this project. The coding basics of configuring the LEDs and LCDs of the ARM Cortex M3 and its NXP LPC1768 microcontroller will be simulated, debugged, and analyzed.

These concepts will furthermore help in the understanding of the Cortex M-series processor, including how to run a simple program on the MCB1700 dev board, which is frequently used in embedded systems. As the majority of embedded systems use ARM processors for low-power consumption and competitive performance, designers will find the skill sets obtained from this project especially useful. More specifically, students will acquire practical experience with barrel shifting, conditional branching, and bit banding to analyze their effectiveness in both Debug and Target mode using performance evaluation methods covered in the lectures.

The lab equipment will allow engineers to become familiar with the uVision environment, its simulation capabilities, and the tools needed to assess various CPU performance factors. The following coding information referenced in the appendix will be compiled to construct an LED bit-band application. Depending on how the bit is toggled at the hardware level, the LED control methods' execution speeds vary greatly. Furthermore, it introduces the concept of how to use RTX, a Real-Time Operating System (RTOS), for basic multitasking. Students will specifically learn how to use the round-robin scheduling mechanism that is a feature of RTX to schedule threads of work.

Understanding real-time scheduling with µVision for ARM Cortex-M3 is the aim of this project. In particular, you will discover how to schedule and apply the widely used Rate Monotonic Scheduling (RMS) method for Fixed Priority Scheduling (FPS). Additionally, priority inversion and its resolution will be practically experienced by you. You will learn about virtual timers, inter-thread communication techniques (such as signals and thread waits), and static and dynamic priority inversions to put these strategies into practice. To complete this project, students must have a solid understanding of the previous labs, which dealt with µVision and RTX. For background information on RMS and priority inversion, consult the course notes.Students learn how to create multi-threaded apps for ARM Cortex-M3/M4 processors with RTX in this project. Using preemptive and non-preemptive scheduling techniques enabled by the RTX operating system, CMSIS libraries, and µVision, the students will learn how to schedule multi-threaded applications.

II. PAST WORK/REVIEW

The SysTick Timer on the ARM Cortex-M microcontroller is used to manage and measure the execution time of Round-Robin (RR) scheduling (Masking, ThreadYield() Function, and Direct Bit-Band) in the code.

A. Hardware Implementations

- Controllers
- Potentiometer
- KBD directional joystick
- KBD select button
- Reset button

- LCD Screen
- Speakers
- Port LEDs

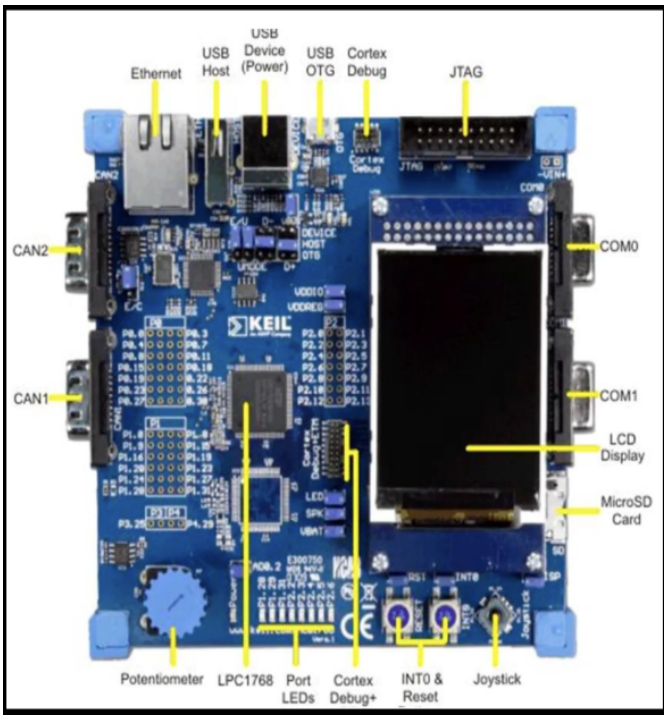

Fig. 1. MCB1700 board - LPC1768 Microcontroller

B. Software Implementations

A collection of instructions that the user processes must be carried out by the program component. Keil uVision, a program written in the C programming language, was utilized in this project. A functional system is made up of both software and hardware components that are coupled to one another. In order to transmit signals to the hardware and carry out the required tasks, the software continuously waits for human input. By obtaining the data from the program and outputting the outcomes, the hardware completes the necessary task.

$$https://www.gimp.org/ \quad (1)$$

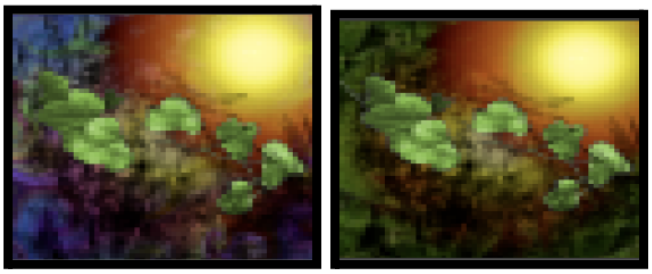

Fig. 2. GIMP (GNU Image Manipulation Program)

In embedded systems, where a file system is not available and images must be stored directly in the flash memory of the micro controller, the process of converting GIMP XCF image files into.c files is crucial for utilizing graphical assets. The image data must be converted from its complicated format into a raw, static C array that the micro controller's display driver can read. This specific process is frequently mentioned in embedded projects like your Media Center. To ensure the proper color depth, the image must first be processed in GIMP (or another program), such as the project's 16 BPP (R-5 bits, G-6 bits, B-5 bits) standard; the file's header information must then be eliminated using a hex editor, leaving only the contiguous raw pixel data; finally, a conversion utility wraps this raw data into a C syntax array (such as const unsigned char ARM_Ani_16bpp[]) to make the image memory-resident and directly accessible for writing to the LCD controller's buffer.

III. METHODOLOGY

A. Importing an Image onto the Keil MCB1700 Board

Using a customized version of GIMP to transform images for LCD is required per the instructions for importing images into the Keil MCB1700 Board. Since transparent pixels will appear as black, the image must first be loaded into GIMP. The image must be flipped vertically using $Image > Transform > FlipVertically$ to compensate for the board's upside-down design. Because of the limited screen size and memory constraints, it might also need to be reduced using Image ¿ Scale Image.

$File > ExportAs...$ is then used to convert the image to the "C Source code (*.c)" format. The important options "Save as RGB565 (16-bit)" and "Save macros instead of struct" must be chosen in the export dialogue, along with a prefixed name. #include "yourCFile.c" is then used to include the generated.c file in the source code, add it to a new "Resources" group, and copy it into the working directory of the Keil project. GLCD_Bitmap(x, y, w, h, bitmap) is the function that renders the image on the LCD, where bitmap is the pointer to the image data array.

Using the GIMP macro, the file extension should be changed to .h and included as such for more recent versions of GIMP that export to structs. GIMP is advised; however, you can also use the external tool:

$$https://notisrac.github.io/FileToCArray/ \quad (2)$$

B. Memory Space Issues on the Keil MCB1700 Board when Importing Images

Programmers must distinguish between defining and declaring variables in order to prevent duplicate memory allocation and potential memory space issues when importing a large number of photos. A variable must only be defined once within a specified scope, and it is defined when the compiler allots storage for it. On the other hand, a variable is declared if the compiler is only told that it exists, which can happen more than once. The image variable (defined in its associated C code, such as "image.c") must only be declared in other files, such as "main.c," using the extern declaration keyword in order to conserve board memory and guarantee that the image data

is not specified twice. Declaring the variable as extern in the header file and providing the definition in only one source file is the proper method; placing a definition in a header file is discouraged because it results in many definitions when multiple source files include that header.

To gain a competitive edge, distinguish between the standard GNU GPL, which limits use to free software programs, and the GNU Lesser GPL (LGPL), which allows use in proprietary products. The strategic objective is to release powerful libraries with unique features, like GNU Readline, under the regular GPL, whereas the LGPL is favoured for non-unique libraries whose functionality is easily copied (like the C library). Applying this logic to the GNU Image Manipulation Program (GIMP), every special or unique library that gives GIMP a clear, potent benefit needs to be licensed under the GPL. The whole free software community gains from this deliberate exclusivity by making GIMP and other applications more appealing than their proprietary counterparts; other projects are encouraged to employ free software licenses in order to obtain and utilize those special, GPL-covered building blocks.

C. Memory System

The IDLE process exhibits few execution blocks, as can be seen in the waveform, suggesting that the CPU is running at close to 100the observed period. Therefore, the running thread must willingly give up the CPU to the next thread of equal or higher priority when osThreadYield() is used.

The GUI follows a strict control loop: selection of any task initiates the feature; upon task completion or user termination, the system automatically transitions back to the main menu (Figure 1.0) to maintain high availability and predictable operation.

1) Run-time Environment (RTE): Threads are only intended to run for the minimum of $\approx 2.52\mu s$ needed to increment their counter in this cooperatively scheduled environment before instantly handing over to the next task.

a) Performance Analyzer : Displays CPU time split among several program modules and functions as opposed to the previous situation, in which the processor was monopolized by a single infinite loop.

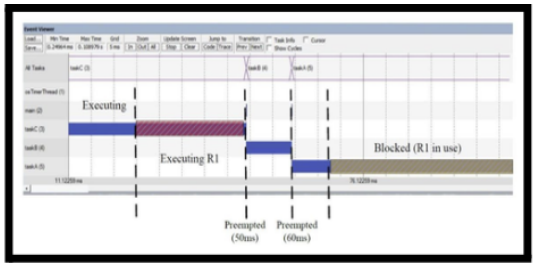

Fig. 3. Function based on RTX (Real-Time Execution)

b) Event Viewer: A state of indefinite postponement (starvation) for the lower-priority thread in this proactive, priority-based scheduling system is confirmed by the Event Viewer's visual proof.

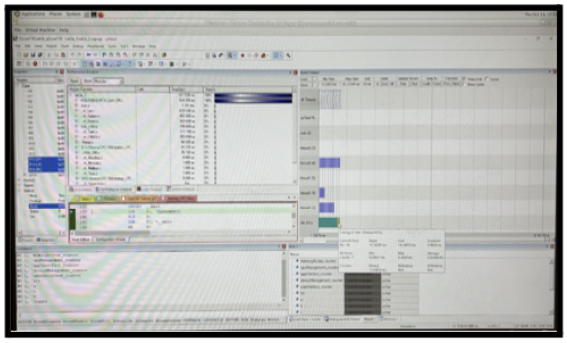

Fig. 4. ThreadYield() function of the RTX for the ARM Cortex-M3

2) Real-time Execution (RTX): Using the SysTick_Handler Interrupt Service Routine (ISR) to continuously increment the global time counter, msTicks, and then using the delay_ms function to create a software delay that pauses execution until the required number of ticks (BASE_DELAY)(BASE_ADD×curentShift) has passed, the method entails initializing the SysTick counter to run at a high frequency. This method effectively employs the SysTick as a consistent.

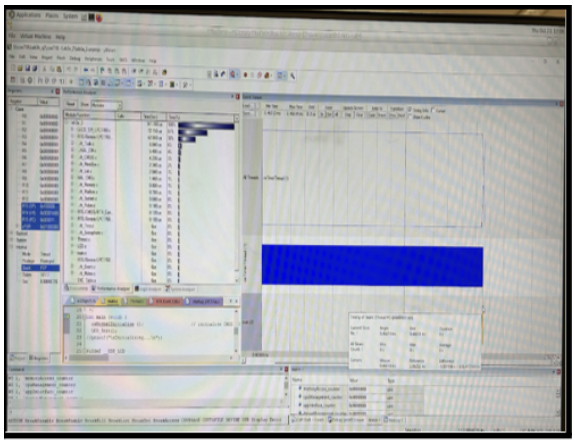

Fig. 5. Performance Analyzer of the memory system

The Direct Bit-Band method is the fastest ($\approx$ 1 - 2 cycles) because the hardware can set or clear the bit in a single, atomic operation, due to its pre-calculated alias address. It calculates the complex bit-band alias address during program execution, adding runtime overhead, the Function Mode (Macro) is a little slower ($\approx$ 2 - 4 cycles). Because it necessitates a multi-step, non-atomic Read-Modify-Write (R-M-W) sequence on the entire 32-bit register, the Mask Mode is the slowest ($\approx 3 + cycles$). The ternary conditional operator (? :) is also used in comparison to conventional if/else statements; the ARM compiler can frequently optimize into effective, branchless conditional execution assembly instructions, reducing pipeline pauses and enhancing efficiency.

IV. DESIGN

Utilizing the computational and peripheral capabilities of the NXP LPC1768 microcontroller, which has an ARM Cortex-M3 core and is housed on the MCB1700 development board, the Media Center is designed as a multithreaded, interactive embedded system. Under the guidance of a Real-Time Operating System (RTOS), the architecture is intended to offer a responsive and adaptable foundation for multimedia functions.

A. System Architecture & Control

The system utilizes a client-centric design with a primary Graphical User Interface (GUI) displayed on the on-board LCD screen.

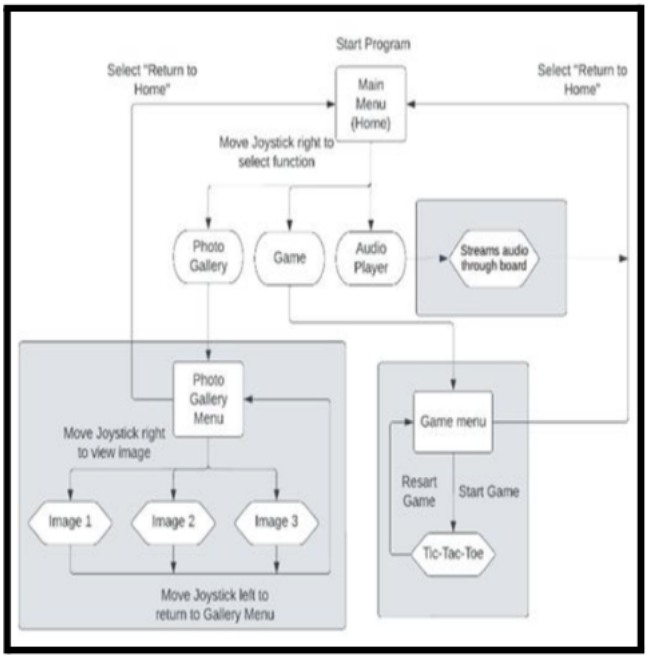

Fig. 6. Flow chart of the Media Center

1) Input & Navigation: Interactions are managed via dedicated peripherals: Input and Navigation: The core of user interaction is the MCB1700's directional joystick, which enables users to scroll through the main menu options (Photo Gallery, MP3 Player, Games) and initiate feature selection. A USB keyboard is integrated specifically to support interactive input for the gaming feature.

2) Real-Time Operating System (RTOS): The RTX RTOS is employed to manage concurrent tasks, ensuring responsiveness and resource allocation. The design incorporates multi-threaded applications scheduled using both basic round-robin and advanced Rate Monotonic Scheduling (RMS) techniques, providing a robust, predictable environment necessary for real-time media processing.

3) Implementations & Feedback: Port LEDs are used to provide immediate visual feedback on the system's state during navigation and execution. Analog control is provided

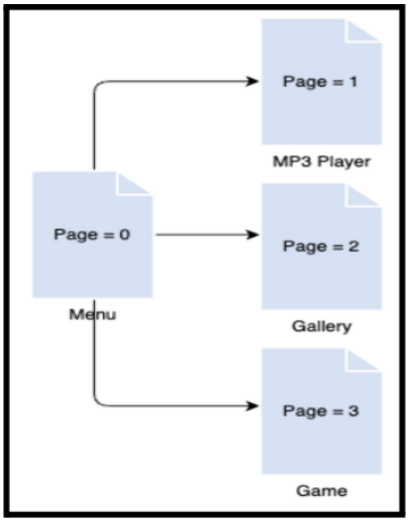

Fig. 7. The global variable, 'Page', determines the application displayed to the LCD

by the Potentiometer, which is dedicated to volume control for the MP3 Player component via an Analog-to-Digital Converter (ADC). A specialized process used in embedded systems development to convert graphical assets created in GIMP into a format that can be directly compiled and stored in the microcontroller's memory (like the one used in your media center project) is the conversion of a GIMP XCF image file to a.c file. The image must be transformed into a static C array with the raw pixel data because microcontrollers sometimes lack file systems or the capacity to load complicated image formats like XCF at runtime.

This procedure usually entails exporting the image data from GIMP into a raw binary or hex format, which is then wrapped in a C syntax and defined as a const unsigned char or const unsigned short array using a conversion tool (or occasionally GIMP itself via a script/plugin). This gives the embedded C code access to the image data. It can then read the array values directly and write them to the buffer of the LCD controller for display. In order to guarantee that the array only contains the contiguous, loadable pixel values, the conversion step, which is mentioned in your project's Abstract.txt is essential for determining the proper color depth (e.g., 16 bits per pixel) and eliminating superfluous header information.

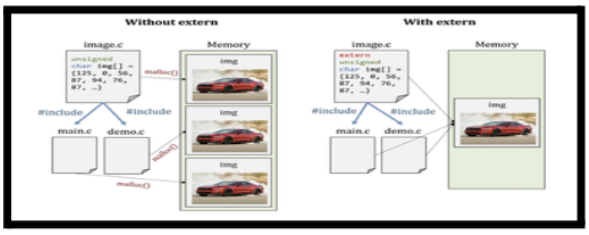

Fig. 8. Declaration of a variable using extern with the definition of int foo to convert image.xcf files into .c files

4) Game Execution (Real-time): The scheduler virtually never discovers the ready queue empty because of this quick yielding. As a result, the CPU is constantly occupied switching between running application threads, which causes the CPU utilization time to approach 100This is supported by the Idle Demon variable (countIDLE), which, when monitored over time, records a proportionately low count because the scheduler hardly ever, if ever, allots CPU resources to the low-priority IDLE task.

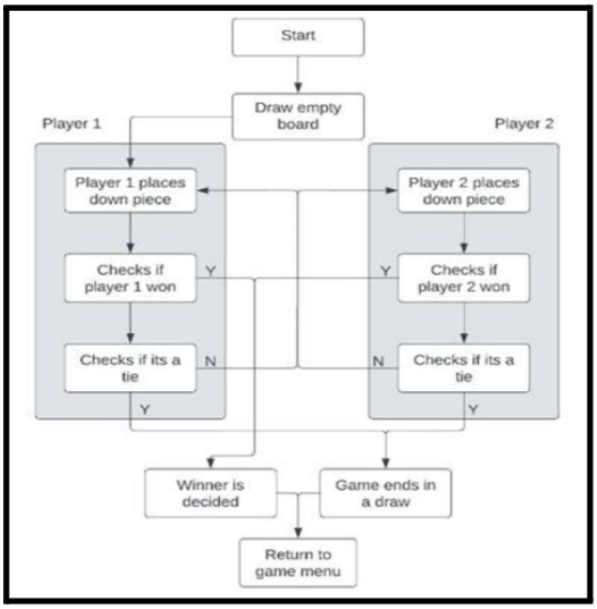

Fig. 9. Block diagram of the game

B. Graphical Asset Management & Memory Strategy

Once a game starts, an empty 3 by 3 square grid is drawn on the board. To make sure that the pieces are placed in the correct spots, a grid of the coordinates was created which contains the spaces in which only the players can place a game piece down.

TABLE I
COORDINATE GRID OF THE GAME

Game Grid Layout (X, Y) Coordinates		
Coordinate 1	**Coordinate 2**	**Coordinate 3**
(1, 5)	(1, 9)	(1, 13)
(4, 5)	(4, 9)	(4, 13)
(7, 5)	(7, 9)	(7, 13)

[a]Tic-Tac-Toe Coordinates.

Table 1 represents the exact places on the LCD where the game pieces could only be placed in. By default, the game starts at the coordinate (1, 5) and is defined by setting the values 5 and 1 to x and y respectively. The values of x or y are then incremented or decremented by a value that will move the cursor to the next defined coordinate space. In this case, y increases or decreases by a value of 3, and x increases or decreases by a value of 4. To prevent the cursor from leaving the defined coordinate space, conditions were made to ensure that the value of y and x do not exceed their maximum values or go below their minimum value. This would stop incrementing or decrementing the values of x or y.

C. Core Feature Implementation

The game can now identify when a player wins, loses, or draws since a two-dimensional array is used to record where each game piece is placed and by which player. Every time a game piece is placed, the placeholder values are assigned. A ”if” statement was developed that examines each player’s eight potential winning strategies. When the placeholder value is the same in three rows for any of the winning strategies, the player is deemed the winner. When a player wins, the game goes back to the menu and shows the winner’s name on the board. The game returns to the game menu and declares a draw if no player connects three of their pieces in a row.

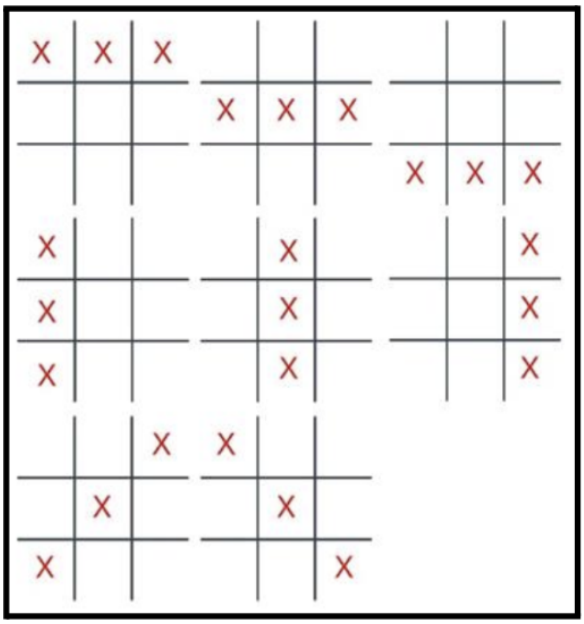

Fig. 10. Tic-Tac-Toe winning sequences (8 arrangements)

V. EXPERIMENTAL RESULTS

A. Media Center

A thread-based display would function similarly to a page-based display, but each app would have its own thread and handle the joystick and LCD from either the app’s own file or the file that manages each thread, rather than a variable controlling the LCD output and handling joystick inputs from the same thread.

To build a thread-based strategy, all four threads would need to be initialized. The main menu would be initialized with high priority, while the other three programs would be initialized with standard priority out of these four threads. The main menu would be the first thread to run when the project was launched. The code turned out to be more clearer and more adaptable when additional features were added because each app’s code was divided into its own thread.

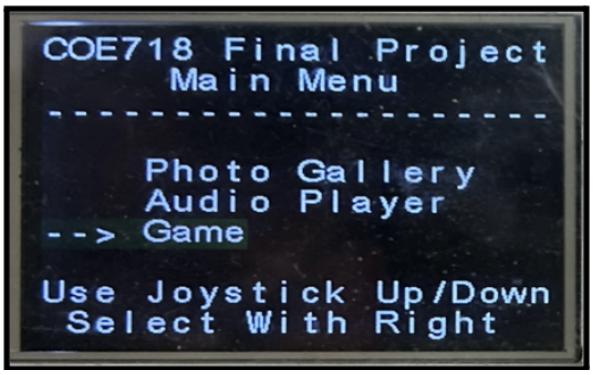

Fig. 11. Main Menu of the Embedded System

- Photo Gallery
- mp3 Player
- Games(s)

B. Photo Gallery

According to the Photo Gallery standards, the LCD must be able to show different bitmap (.bmp) images one at a time. The user interface design for viewing these images is adaptable and creative; some ideas include a menu with file names or a straightforward carousel display that can be manipulated with a joystick. Before being displayed, the .bmp files must be converted, which is a crucial technical requirement. A C source file with the image data as an array of unsigned characters must be created from each .bmp file. GIMP with the export options "Use macros instead of struct" and "Save as RGB565(16-bit)" is the suggested conversion tool. For compatibility with the Keil compiler, all guint8 variable types in the produced C file must be manually changed to unsigned char after conversion. The resulting file is of type static; thus, the #include "filename.c" preprocessor directive must be used to integrate the converted C file containing the picture data into the main program.

Fig. 12. Photo Gallery of the Media Center

Fig. 13. Image database - Photo Gallery

C. Audio Player

The MP3 player component of the media center project is in charge of using a USB to stream audio from a PC to the on-board speaker of the development board. This connection must be made when the option is selected, and it must be disconnected when the menu is closed. The potentiometer controls the volume; rotation in both clockwise and counterclockwise directions raises and reduces the volume, respectively. The system must show a splash screen on the LCD during playing, and the audio must come from an MP3 or YouTube video on the PC. The implementation should make use of the USB audio example and USB interrupt resources from the course.

Fig. 14. Audio Mode

D. Game Selection

The programmer has total discretion over implementation thanks to the game(s) section specifications, which call for the MCB1700's LCD and joystick to be integrated and used interactively. Simple word games are definitely prohibited, and the game or games must be notable graphics arcade-style games (such as Pac-Man, Snake, or Tetris). One game is required at the very least, and bonus points can be earned for implementing two or more games.

The project's technical complexity is immediately increased by the strict demand to integrate well-known visual arcade-style games like Pac-Man, Snake, or Tetris, while expressly forbidding basic word games. In order to produce smooth, real-time graphics and interaction, this mandate intentionally tests the programmer's capacity to handle constrained resources, particularly the NXP LPC1768 microcontroller's CPU speed and memory. Arcade games require constant screen updates, pixel manipulation, collision detection, and quick state management, in contrast to word games. Developing a powerful graphics engine that can quickly create sprites, update scoreboards, and refresh the game map on the MCB1700's LCD at a fast enough frame rate to guarantee playability replaces

the primary goal of simple textual output. This limitation puts the basic knowledge of embedded systems timing and optimization to the test.

Furthermore, the game selection necessitates a deep dive into the interfacing of the LCD and joystick. Because the game must be ”interactively” integrated, the programmer must build effective, low-level driver code to read the joystick's state with minimal latency (perhaps via GPIO pins or the Analog-to-Digital Converter, as the literature suggests with the LPC_ADC_TypeDef). Smooth and responsive movement is crucial for a game like Pac-Man; the game becomes unplayable if there is a noticeable lag between a joystick input and the character's movement. Data structures such as the LPC ADC TypeDef from the LPC17xx.h header file enable direct register manipulation in this operation. The difficulty is in converting the joystick's physical, frequently analog input into accurate, debounced digital signals that efficiently manage the game's logic and graphics updates.

The header file LPC17xx.h, which provides information on the LPC1768 chip's registers and access to them through well-formatted C data structures, is typically used when programming with the NXP LPC1768 chip. This allows operations to be carried out on these registers quickly and conveniently. The LPC_ADC_TypeDef, a general data structure for accessing analog to digital converter registers, is one example of such a data structure. That data structure initially provides access to a few control registers during project setup.

Fig. 15. Game Menu and operation of Player 1 vs Player 2 in tic-tac-toe

Lastly, a crucial architectural factor—modularity and resource partitioning—is introduced by the possibility of additional points for implementing two or more games. A successful strategy would entail creating a single game framework or engine that is readily reusable across several game titles since the programmer has complete control over implementation. While enabling separate game loops and assets for games like Snake and Tetris, this structure must effectively handle shared resources, such as memory for graphics buffers and the interrupt routines for the joystick. This design decision is essential for minimizing code duplication and showcasing sophisticated software engineering concepts in the embedded context, demonstrating that the design is a scalable platform that can support numerous unique, resource-intensive graphical applications rather than just a single, monolithic application.

VI. CONCLUSIONS

The project's development was greatly aided by the knowledge acquired from the semester's lectures and laboratories. This entails comprehending the characteristics of the MCB1700 Board, uVision ideas, debugging, and multi-threading to solve large programs featured in a fully operational media center that makes use of both software and hardware issues, as well as software and hardware interaction. A photo gallery, an animated game, and an MP3 player all contributed to the media center's user-friendliness, simplicity, and enjoyment.

Practical experience in embedded systems design using the ARM Cortex-M3 CPU and the Keil uVision IDE, this project effectively achieved its main goals. Creating and evaluating code using processor capabilities like bit banding, conditional branching, implementing the RTX RTOS, knowledge about NXP LPC1768 microcontroller low-level performance enhancement, and barrel shifting were the early learning objectives.

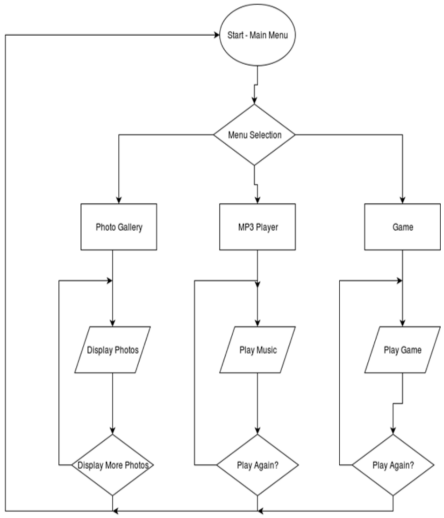

Fig. 16. Block Diagram of the Media Center

Overall, these fundamental and sophisticated ideas came together to create a Media Center application that included a graphical Gaming Center, a USB Audio Player with ADC-controlled volume, a Photo Gallery requiring specific GIMP image conversion techniques, and a navigable GUI (controlled by the joystick). Furthermore, the Keil uVision software and ARM Cortex M3 device, as displayed on the liquid crystal display (LCD) using the Debug simulation session with optimization levels to control the LEDs.

VII. REFERENCES

The bibliography for the COE718 Final Project, "Media Center," is very informative; it functions as a map of the project's development workflow, toolchain, and academic limitations in addition to being a collection of references. From the low-level embedded programming requirements to the high-level graphical assets and configuration management, the three references, GIMP, GitHub, and the course materials. Collectively describing the full ecosystem. The effective implementation of the GUI-based media center on the MCB1700 board depends on this synthesis of technical, logistical, and educational resources.

This reference emphasizes the use of GIMP, the GNU Image Manipulation Program, a potent open-source tool for image editing and graphic design. GIMP was probably used in the context of the Embedded Systems Design project to produce and improve any visual assets required for the Graphical User Interface (GUI) of the media center that was shown on the LCD of the MCB1700 board. This entails creating unique icons, splash screens, or diagrams and making sure they are appropriate for the embedded technology in terms of color depth and resolution. This reference's inclusion highlights the methodical procedures used to prepare the system's non-code components and shows how multimedia components were incorporated into the finished output. GIMP made it possible for the programmer to carry out important functions like color depth reduction, which guarantees that icons and images follow the restricted color schemes that the integrated display hardware supports. It was utilized to produce and improve visual assets, including schematic diagrams for the final report, splash screens shown at launch, and unique icons for the media center's many programs. This careful preparation of the non-code components reduced the processing cost on the NXP LPC1768 processor during runtime by ensuring that the graphical elements were both aesthetically beautiful and computationally efficient. The reference emphasizes this meticulous planning, which is proof of a comprehensive approach to embedded development where software optimization encompasses graphical asset pipeline management.

When a GitHub repository is cited, it means that a version control system was used to professionally manage the project's source code. As the focal point for the whole development lifecycle, GitHub allows for effective change tracking, safe code backups, and possibly even collaboration if the project includes several team members. This reference, which indicates the location of the final C/C++ code, design files, and embedded media center configurations, is essential for transparency and reproducibility. It confirms that industry best practices for software engineering were used in the development and upkeep of the code covered in the report. The reference to a particular GitHub repository attests to the "Media Center" project's adherence to professional software engineering standards and its early integration of a strong version control system. The entire development lifecycle was centered around GitHub. Its use offered crucial infrastructure for safe, off-device code backup and change tracking, which enabled the programmer to audit the evolution of the code base and return to earlier stable stages. This is especially important in embedded development, where months of work can be easily lost due to corrupted development environments or physical hardware failures. Citing the repository also creates a foundation for project reproducibility and transparency. The final C/C++ source code, low-level driver implementations, hardware initialization procedures, and embedded media center configuration files are all explicitly pointed to by the reference. This repository is the final record for assessors or future developers, attesting to the application of industry best practices in the creation and maintenance of the code. With this integration, the project becomes a professionally managed technical deliverable rather than just an academic exercise.

The entire endeavor is grounded in its educational and technical context thanks to this scholarly reference. Citing the official course page for "COE 718: Embedded Systems Design," the report makes it clear that the project complies with Dr. Gul N. Khan's technical standards and curriculum criteria. The project's objectives, such as the requirement to install a basic GUI-based media center on the MCB1700 board and make use of its hardware capabilities like the joystick and LCD, are based on this source. The theoretical framework, design limitations, and pedagogical goal that eventually guided all technical choices and project outcomes described in the final report are established in the reference. The official course materials for Dr. Gul N. Khan's "COE 718: Embedded Systems Design" are the most fundamental source. The entire endeavor is grounded in the necessary technical and educational context by this scholarly citation. The project's scope and design constraints were clearly defined by adhering to the course requirements. The main goal was to directly target the NXP LPC1768 microcontroller and its related peripherals on the MCB1700 board rather than just create a broad piece of software.

The reference mandates a number of integration points, including the implementation of a basic GUI-based media center, the need to interface with the LCD for display, and—most importantly—the joystick's interactive control mechanism. Low-level programming that directly accesses the chip's registers—typically via structures described in the LPC17xx.h header, like the LPC ADC TypeDef used for joystick input—was required by this framework. The project thoroughly tested the student's capacity to go beyond high-level application programming and engage in actual embedded systems design, handling interrupts, memory, and direct hardware access, according to the course's mandate. These three references work together to highlight the project's all-encompassing implementation approach. The GIMP reference guarantees that the finished project satisfies the visual criteria of a contemporary GUI-based program, the academic reference establishes the technical standard, and the GitHub reference guarantees the development process is responsible and repeatable. Thus, the "Media Center" project's ultimate functionality—the seamless display and interactive control—as well as its adherence to

this established methodology are used to gauge its success. The report's specific naming of these sources attests to the programmer's methodical approach to the problem, which addressed the LPC1768 chip's technological limitations while upholding a high standard for graphical display and software administration. This multi-layered strategy is a sign of an advanced engineering method used in the difficult limitations of an embedded environment.

REFERENCES

[1] GIMP, https://www.gimp.org/ (accessed Nov. 27, 2025).
[2] GitHub, https://github.com/syed-maisam (accessed Nov. 8, 2025).
[3] G. N. Khan, "COE 718: Embedded Systems Design," Electrical, Computer and Biomedical Engineering. https://www.ecb.torontomu.ca/ courses/coe718/index.html

Syed Maisam Abbas, it is crucial that you carefully go over your "COE718 Final Project, Media Center" report and eliminate any instructional or guidance content that came from the original IEEE conference form before submitting it. This covers any non-content elements, such as usage notes, formatting advice, and placeholder text. If you don't remove this template material entirely from your final paper, your submission can be considered incomplete or formatted wrongly, which could eventually prevent the report from being approved or published in the conference proceedings. Please consider this removal to be the last, required check.

VIII. APPENDIX

A. ADC.c

```
/*----------------------------------------------------------------------------
 * Name:    ADC.c
 * Purpose: low level ADC functions
 * Note(s): possible defines select the used ADC interface:
 *          __ADC_IRQ   - ADC works in Interrupt mode
 *                      - ADC works in polling mode (default)
 *----------------------------------------------------------------------------
 * This file is part of the uVision/ARM development tools.
 * This software may only be used under the terms of a valid, current,
 * end user licence from KEIL for a compatible version of KEIL software
 * development tools. Nothing else gives you the right to use this software.
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2008-2011 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------*/

#include "LPC17xx.H"                    /* LPC17xx definitions          */
#include "ADC.h"

uint16_t AD_last;                       /* Last converted value         */
uint8_t  AD_done = 0;                   /* AD conversion done flag      */

/*----------------------------------------------------------------------------
  Function that initializes ADC
 *----------------------------------------------------------------------------*/
void ADC_Init (void) {

  LPC_SC->PCONP |= ((1 << 12) | (1 << 15));  /* enable power to ADC & IOCON */

  LPC_PINCON->PINSEL1  &= ~( 3 << 18);
  LPC_PINCON->PINSEL1  |=  ( 1 << 18);  /* P0.25 is AD0.2               */
  LPC_PINCON->PINMODE1 &= ~( 3 << 18);
  LPC_PINCON->PINMODE1 |=  ( 2 << 18);  /* P0.25 no pull up/down        */

  LPC_ADC->ADCR       =  ( 1 <<  2) |   /* select AD0.2 pin             */
                         ( 4 <<  8) |   /* ADC clock is 25MHz/5         */
                         ( 1 << 21);    /* enable ADC                   */

#ifdef __ADC_IRQ
  LPC_ADC->ADINTEN     =  ( 1 <<  8);  /* global enable interrupt      */

  NVIC_EnableIRQ(ADC_IRQn);             /* enable ADC Interrupt         */
#endif
}
```

```
/*----------------------------------------------------------------------------
  start AD Conversion

*----------------------------------------------------------------------------*/
void ADC_StartCnv (void) {
  LPC_ADC->ADCR &= ~( 7 << 24);
/* stop conversion              */
  LPC_ADC->ADCR |=  ( 1 << 24);
/* start conversion             */
}


/*----------------------------------------------------------------------------
  stop AD Conversion

*----------------------------------------------------------------------------*/
void ADC_StopCnv (void) {

  LPC_ADC->ADCR &= ~( 7 << 24);
/* stop conversion              */
}


/*----------------------------------------------------------------------------
  get converted AD value

*----------------------------------------------------------------------------*/
uint16_t ADC_GetCnv (void) {

#ifndef __ADC_IRQ
  while (!(LPC_ADC->ADGDR & ( 1UL << 31)));  /* Wait for Conversion end       */
  AD_last = (LPC_ADC->ADGDR >> 4) & ADC_VALUE_MAX; /* Store converted value   */

  AD_done = 1;
#endif

  return(AD_last);
}


/*----------------------------------------------------------------------------
  A/D IRQ: Executed when A/D Conversion is done

*----------------------------------------------------------------------------*/
#ifdef __ADC_IRQ
void ADC_IRQHandler(void) {
  volatile uint32_t adstat;

  adstat = LPC_ADC->ADSTAT;
/* Read ADC clears interrupt     */

  AD_last = (LPC_ADC->ADGDR >> 4) & ADC_VALUE_MAX; /* Store converted value   */

  AD_done = 1;
}
#endif
```

B. adcurser.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 */
}
```

C. Blinky.c

```
/*-------------------------------------------------
----------------------
* Name: Blinky.c
* Purpose: Media Center Project
* By: Syed Maisam Abbas
*-------------------------------------------------
----------------------*/

#include <LPC17xx.h>
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"

#define __FI 1

//Menu logos
extern unsigned char music_logo[];
extern unsigned char gallery_logo[];
extern unsigned char game_logo[];

//Initailize functions
extern int mp3_player(void);
extern int game(void);
extern int gallery(void);

void main_menu(void);

int button_pressed;

//Main Menu
void main_menu() {
	int active = 0;

	GLCD_Clear(White);
	while(1) {

GLCD_SetBackColor(White);
		GLCD_SetTextColor(Red);
GLCD_DisplayString(0,0,__FI,"     MAIN
MENU    ");
		GLCD_SetTextColor(Blue);

GLCD_DisplayString(8,0,__FI,"
UP/DOWN - Scroll  ");

GLCD_DisplayString(9,0,__FI,"   SELECT
- Select   ");

		button_pressed =
get_button();
		if(button_pressed ==
KBD_DOWN) {
			if(active == 3) {
				active = 0;
			}
			else {
				active++;
			}
		}
		else if(button_pressed ==
KBD_UP){
			if(active ==0) {
					active
= 3;
			}
			else {
				active--;
			}
		}

		//Gallery selected
		if(active==0) {

GLCD_SetBackColor(Yellow);

GLCD_SetTextColor(Blue);
```

```
GLCD_DisplayString(3,0,__FI,"1) GALLERY          ");

GLCD_SetBackColor(White);

GLCD_SetTextColor(Blue);

GLCD_DisplayString(4,0,__FI,"2) MUSIC ");

GLCD_DisplayString(5,0,__FI,"3) GAME ");



if(button_pressed == KBD_SELECT) {

GLCD_Clear(Blue);

gallery();
                                    }
                            }

                    //MP3 selected
                    if(active==1) {

GLCD_SetBackColor(White);

GLCD_SetTextColor(Blue);

GLCD_DisplayString(3,0,__FI,"1) GALLERY          ");

GLCD_SetBackColor(Yellow);

GLCD_SetTextColor(Blue);
```

```
GLCD_DisplayString(4,0,__FI,"2) MUSIC ");

GLCD_SetBackColor(White);

GLCD_SetTextColor(Blue);

GLCD_DisplayString(5,0,__FI,"3) GAME ");



if(button_pressed == KBD_SELECT) {

GLCD_Clear(White);

mp3_player();

GLCD_Clear(White);
                                    }
                            }

                            //Game selected
                    if(active==2) {

GLCD_SetBackColor(White);

GLCD_SetTextColor(Blue);

GLCD_DisplayString(3,0,__FI,"1) GALLERY          ");

GLCD_DisplayString(4,0,__FI,"2) MUSIC ");

GLCD_SetBackColor(Yellow);

GLCD_SetTextColor(Blue);
```

```
GLCD_DisplayString(5,0,__FI,"3) GAME
");




if(button_pressed == KBD_SELECT) {

GLCD_Clear(White);

game();
                                        }
                                }
                        }
                }

//Main
int main(void) {
                        LED_Init();
                        GLCD_Init();
                        KBD_Init();

                        GLCD_Clear(White);

GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);

GLCD_DisplayString(5,0,__FI,"    MEDIA
CENTER    ");
                        GLCD_SetTextColor(Blue);


GLCD_DisplayString(7,0,__FI,"     Mohib
Khan   ");
                        GLCD_SetTextColor(Red);


GLCD_DisplayString(9,0,__FI,"   Select to
Start   ");
                                while(1) {
                                        if(get_button() ==
KBD_SELECT) {

main_menu();
                                        }
                                }
                        }
```

D. gallery.c

```
/*-------------------------------------------------
------------------------
 * Name:    gallery.c
 * Purpose: Gallery
 * By: Syed Maisam Abbas

*-------------------------------------------------
------------------------*/


#include <LPC17xx.H>
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"

#define __FI        1

//Gallery Pictures
#include "csgo_logo.c"
#include "valorant_logo.c"
#include "cod_logo.c"
#include "david.c"


//Picture Scroll
int gallery(void){
        int current = 0;
        int button_pressed = 0;



        GLCD_SetBackColor(White);
  GLCD_SetTextColor(Blue);

                while(1){


GLCD_DisplayString(0, 0, __FI, "     -
GALLERY -     ");

GLCD_DisplayString(8, 0, __FI, "
RIGHT/LEFT - SCROLL ");

GLCD_DisplayString(9, 0, __FI, " SELECT
- MAIN MENU ");

                        button_pressed =
get_button();

                        if(button_pressed ==
KBD_LEFT){
                                if(current ==
0){
                                        current
= 3;

GLCD_Clear(Blue);
                                }
                                else {

current--;

GLCD_Clear(Blue);
                                }
                        }
                        else if(button_pressed
== KBD_RIGHT) {
                                if(current ==
3){
                                        current
= 0;

GLCD_Clear(Blue);
                                }
                                else {

current++;

GLCD_Clear(Blue);
```

```
                        }
                    }
                    else if(button_pressed
== KBD_SELECT){

GLCD_Clear(White);
                                return
0;

                    }

                    if(current == 0)  {

GLCD_Bitmap (80, 55, 160,  90, (unsigned
char *)CSGO_LOGO_pixel_data);
                    }
                    else if (current == 1)
{

GLCD_Bitmap (80, 55, 160,  99, (unsigned
char *)GIMP_IMAGE_pixel_data);
                    }
                    else if (current == 2)
{

GLCD_Bitmap (80, 55, 160,  90, (unsigned
char *)VALORANT_LOGO_pixel_data);
                    }
                    else if (current == 3)
                    {

GLCD_Bitmap (80, 25, 160,  166, (unsigned
char *)DAVID_pixel_data);
                    }
                }


}
```

E. apple.c

```
/* GIMP RGBA C-Source image dump
(apple.c) */

//16x16
const unsigned char apple[] =
("\303>;\205\302=:\234\274<\071\212\272;\
070)\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000"

"\000\000\000\000\000\000\000\000\301)f\2
36&\226A\031A\010\346A\210b\343\040\0
00\000\000\000\000\000\000\000\000"

"\000\000\000\000\000\000\000\000\000\34
3[f\236\244tGZ\003!\000\000\000\000\000\
000\000\000\000\000\000\000\000\000\000\
000\000\000"

"\000\000\000\000\000\000\"\062%l\205\07
1\000\000\000\000\000\000\000\000\000\00
0\000\000\000\000\000\000\000\000\000\00
0Ea\010\262"

"i\322(\302\007\252\347\251\347\311\007\3
22\206\241\242H\000\000\000\000\000\000
\000\000\000\000\010"

"\232(\322H\322\211\322\007\322\007\322\
007\322\007\322\007\322\007\322\007\322\
004i\000\000\000"

"\000\000\000\004IH\322(\322\014\333(\32
2\007\322\007\322\007\322\007\322\007\32
2\007\322\007\322"

"\007\322A\030\000\000\000\000\347\211\0
07\322\312\332\312\332\007\322\007\322\0
07\322\007\322\007"
```

"\322\007\322\007\322\007\322\007\322\343X\000\000\000\000(\252\007\322\013\333i\322\007\322\007\322"

"\007\322\007\322\007\322\007\322\007\322\007\322\007\322$q\000\000\000\000(\252\007\322\014\333\007\322"

"\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322$q\000\000\000\000\347\221\007"

"\322\353\332\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\343"

"`\000\000\000\000\205i\007\322\212\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322\007\322"

"\007\322\007\312\202\070\000\000\000\000a\030i\322(\322\010\322\007\322\007\322\007\322\007\322\007\322"

"\322\007\322\007\322\007\322\306\261\000\000\000\000\000\000\000\000\206q\007\322\007\322\007\322\007\322"

"\007\322\007\322\007\322\007\322\007\322\347\311\242@\000\000\000\000\000\000\000\000\000\000\000\000e\201\007\322"

"\007\322\007\322\007\322\007\322\007\322\007\322\347\301\343X\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000"

"\000\000a(\004q$q\343X\003a$q\003aA\030\000\000\000\000\000\000\000\000\000");

F. GLCD_SPI_LPC1700.C

```
/*********************************************************************/
/* GLCD_SPI_LPC1700.c: LPC1700 low level Graphic LCD (240x320 pixels) driven */
/*                     with SPI functions                            */
/*********************************************************************/
/* This file is part of the uVision/ARM development tools.           */
/* Copyright (c) 2005-2011 Keil - An ARM Company. All rights reserved.  */
/* This software may only be used under the terms of a valid, current,  */
/* end user licence from KEIL for a compatible version of KEIL software  */
/* development tools. Nothing else gives you the right to use this software.  */
/*********************************************************************/


#include <lpc17xx.h>
#include "GLCD.h"
#include "Font_6x8_h.h"
#include "Font_16x24_h.h"

/*************************
Orientation  configuration
*************************/

#define LANDSCAPE   1               /* 1 for landscape, 0 for portrait    */
#define ROTATE180   0               /* 1 to rotate the screen for 180 deg */
```

```
/********************** Hardware specific configuration **********************/

/* SPI Interface: SPI3

   PINS:
   - CS     = P0.6 (GPIO pin)
   - RS     = GND
   - WR/SCK = P0.7 (SCK1)
   - RD     = GND
   - SDO    = P0.8 (MISO1)
   - SDI    = P0.9 (MOSI1)
*/

#define PIN_CS      (1 << 6)
#define PIN_CLK     (1 << 7)
#define PIN_DAT     (1 << 9)

#define IN          0x00
#define OUT         0x01

/* SPI_SR - bit definitions
*/
#define TFE         0x01
#define RNE         0x04
#define BSY         0x10

/*------------------------- Speed dependant settings -------------------------*/

/* If processor works on high frequency delay has to be increased, it can be
   increased by factor 2^N by this constant
*/
#define DELAY_2N    18

/*---------------------- Graphic LCD size definitions -----------------------*/

#if (LANDSCAPE == 1)
#define WIDTH       320             /* Screen Width (in pixels)           */
#define HEIGHT      240             /* Screen Hight (in pixels)           */
#else
#define WIDTH       240             /* Screen Width (in pixels)           */
#define HEIGHT      320             /* Screen Hight (in pixels)           */
#endif
#define BPP         16              /* Bits per pixel                     */
#define BYPP        ((BPP+7)/8)     /* Bytes per pixel                    */

/*--------------- Graphic LCD interface hardware definitions ---------------*/

/* Pin CS setting to 0 or 1
*/
#define LCD_CS(x)   ((x) ? (LPC_GPIO0->FIOSET = PIN_CS)    : (LPC_GPIO0->FIOCLR = PIN_CS))
#define LCD_CLK(x)  ((x) ? (LPC_GPIO0->FIOSET = PIN_CLK)   : (LPC_GPIO0->FIOCLR = PIN_CLK))
#define LCD_DAT(x)  ((x) ? (LPC_GPIO0->FIOSET = PIN_DAT)   : (LPC_GPIO0->FIOCLR = PIN_DAT))

#define DAT_MODE(x) ((x == OUT) ? (LPC_GPIO0->FIODIR |= PIN_DAT) : (LPC_GPIO0->FIODIR &= ~PIN_DAT))
#define BUS_VAL()   ((LPC_GPIO0->FIOPIN  & PIN_DAT) != 0)
```

```
#define SPI_START   (0x70)          /*
Start byte for SPI transfer       */
#define SPI_RD      (0x01)          /* WR
bit 1 within start            */
#define SPI_WR      (0x00)          /* WR
bit 0 within start            */
#define SPI_DATA    (0x02)          /* RS
bit 1 within start byte       */
#define SPI_INDEX   (0x00)          /* RS
bit 0 within start byte       */

#define BG_COLOR  0                 /*
Background color              */
#define TXT_COLOR 1                 /* Text
color                         */


/*---------------------------- Global variables
-----------------------------*/

/**********************************
**********************************
*******/
static volatile unsigned short Color[2] =
{White, Black};
static unsigned char Himax;

/************************ Local
auxiliary functions
*************************/

/**********************************
**********************************
********
* Delay in while loop cycles
*
*   Parameter:    cnt:    number of while
cycles to delay                 *
*   Return:
*
**********************************
**********************************
*******/

static void delay (int cnt) {
  cnt <<= DELAY_2N;
  while (cnt--);
}


/**********************************
**********************************
********
* Transfer 1 byte over the serial
communication                   *
*   Parameter:    byte:   byte to be sent
*
*             mode:   OUT = transmit byte, IN
= receive byte          *
*   Return:           byte read while sending
*
**********************************
**********************************
*******/
static unsigned char spi_tran_man (unsigned
char byte, unsigned int mode) {
  unsigned char val = 0;
  int i;

  if (mode == OUT) { DAT_MODE (OUT);
}
  else          { DAT_MODE (IN);  }

  for (i = 7; i >= 0; i--) {
    LCD_CLK(0);
    delay(1);
    if (mode == OUT) {
      LCD_DAT((byte & (1 << i)) != 0);
    }
    else {
```

```
    val |= (BUS_VAL() << i);
   }
   LCD_CLK(1);
   delay(1);
  }
  return (val);
}


/*********************************
*********************************
********
* Transfer 1 byte over the serial
communication                    *
*   Parameter:    byte:   byte to be sent
*
*   Return:               byte read while sending
*
*********************************
*********************************
*******/

static __inline unsigned char spi_tran
(unsigned char byte) {

  LPC_SSP1->DR = byte;
  while (!(LPC_SSP1->SR & RNE));       /*
Wait for send to finish          */
  return (LPC_SSP1->DR);
}


/*********************************
*********************************
********
* Write a command the LCD controller
*
*   Parameter:    cmd:    command to be
written                          *
*   Return:
*
*********************************
*********************************
*******/

static __inline void wr_cmd (unsigned char
cmd) {
  LCD_CS(0);
  spi_tran(SPI_START | SPI_WR |
SPI_INDEX);   /* Write : RS = 0, RW = 0
*/
  spi_tran(0);
  spi_tran(cmd);
  LCD_CS(1);
}


/*********************************
*********************************
********
* Write data to the LCD controller
*
*   Parameter:    dat:    data to be written
*
*   Return:
*
*********************************
*********************************
*******/

static __inline void wr_dat (unsigned short
dat) {
  LCD_CS(0);
  spi_tran(SPI_START | SPI_WR |
SPI_DATA);    /* Write : RS = 1, RW = 0
*/
  spi_tran((dat >>   8));                /* Write
D8..D15              */
```

```
  spi_tran((dat & 0xFF));                 /*
Write D0..D7              */
  LCD_CS(1);
}


/**********************************
**********************************
********
* Start of data writing to the LCD controller
*
*   Parameter:
*
*   Return:
*
**********************************
**********************************
*******/

static __inline void wr_dat_start (void) {
  LCD_CS(0);
  spi_tran(SPI_START | SPI_WR |
SPI_DATA);    /* Write : RS = 1, RW = 0
*/
}


/**********************************
**********************************
********
* Stop of data writing to the LCD controller
*
*   Parameter:
*
*   Return:
*
**********************************
**********************************
*******/

static __inline void wr_dat_stop (void) {

  LCD_CS(1);
}


/**********************************
**********************************
********
* Data writing to the LCD controller
*
*   Parameter:    dat:    data to be written
*
*   Return:
*
**********************************
**********************************
*******/

static __inline void wr_dat_only (unsigned
short dat) {

  spi_tran((dat >>   8));                 /* Write
D8..D15              */
  spi_tran((dat & 0xFF));                  /*
Write D0..D7              */
}


/**********************************
**********************************
********
* Read data from the LCD controller
*
*   Parameter:
*
*   Return:             read data
*
```

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

static __inline unsigned short rd_dat (void) {
  unsigned short val = 0;

  LCD_CS(0);
  spi_tran(SPI_START | SPI_RD |
SPI_DATA);    /* Read: RS = 1, RW = 1
*/
  spi_tran(0);                          /* Dummy
read 1               */
  val   = spi_tran(0);                   /* Read
D8..D15              */
  val <<= 8;
  val  |= spi_tran(0);                   /* Read
D0..D7               */
  LCD_CS(1);
  return (val);
}


/************************************
************************************
********
* Write a value to the to LCD register
*
*   Parameter:    reg:    register to be written
*
*               val:    value to write to the
register                    *
************************************
************************************
*******/

static __inline void wr_reg (unsigned char
reg, unsigned short val) {

  wr_cmd(reg);
  wr_dat(val);
}


/************************************
************************************
********
* Read from the LCD register
*
*   Parameter:    reg:    register to be read
*
*   Return:             value read from the
register                    *
************************************
************************************
*******/

static unsigned short rd_reg (unsigned char
reg) {

  wr_cmd(reg);
  return(rd_dat());
}


/************************************
************************************
********
* Read LCD controller ID (Himax GLCD)
*
*   Parameter:    (none)
*
*   Return:       controller ID
*
************************************
************************************
*******/

static unsigned short rd_id_man (void) {
  unsigned short val;
```

```
  /* Set MOSI, MISO and SCK as GPIO
pins, with pull-down/pull-up disabled     */
  LPC_PINCON->PINSEL0  &= ~((3 <<
18) | (3 << 16) | (3 << 14));
  LPC_PINCON->PINMODE0 |=
0x000AA000;
  LPC_GPIO0->FIODIR    |= PIN_CLK;
/* SCK pin is GPIO output            */
  LCD_CS (1);                       /* Set chip
select high             */
  LCD_CLK(1);                        /* Set clock
high                  */

  LCD_CS(0);
  spi_tran_man (SPI_START | SPI_WR |
SPI_INDEX, OUT);
  spi_tran_man (0x00, OUT);
  LCD_CS(1);

  LCD_CS(0);
  spi_tran_man (SPI_START | SPI_RD |
SPI_DATA, OUT);
  val = spi_tran_man(0, IN);
  LCD_CS(1);

  /* Connect MOSI, MISO, and SCK to SSP
peripheral                        */
  LPC_GPIO0->FIODIR    &= ~PIN_CLK;
  LPC_PINCON->PINSEL0  |= (2 << 18) |
(2 << 16) | (2 << 14);
  LPC_PINCON->PINMODE0 &=
~0x000FF000;

  return (val);
}
```

```
/************************ Exported
functions
*********************************/

/***********************************
***********************************
********
* Initialize the Graphic LCD controller
*
*   Parameter:
*
*   Return:
*
***********************************
***********************************
*******/

void GLCD_Init (void) {
  unsigned short driverCode;

  /* Enable clock for SSP1, clock = CCLK /
2                                 */
  LPC_SC->PCONP       |= 0x00000400;
  LPC_SC->PCLKSEL0    |= 0x00200000;

  /* Configure the LCD Control pins
*/
  LPC_PINCON->PINSEL9 &=
0xF0FFFFFF;
  LPC_GPIO4->FIODIR   |= 0x30000000;
  LPC_GPIO4->FIOSET    = 0x20000000;

  /* SSEL1 is GPIO output set to high
*/
  LPC_GPIO0->FIODIR   |= 0x00000040;
  LPC_GPIO0->FIOSET    = 0x00000040;
  LPC_PINCON->PINSEL0 &=
0xFFF03FFF;
  LPC_PINCON->PINSEL0 |=
0x000A8000;
```

```
  /* Enable SPI in Master Mode, CPOL=1,
CPHA=1                              */
  /* Max. 12.5 MBit used for Data Transfer
@ 100MHz                            */
  LPC_SSP1->CR0        = 0x01C7;
  LPC_SSP1->CPSR       = 0x02;
  LPC_SSP1->CR1        = 0x02;

  driverCode = rd_id_man ();
  if (driverCode == 0) {
    driverCode = rd_reg(0x00);
  }

  if (driverCode == 0x47) {             /* LCD
with HX8347-D LCD Controller   */
    Himax = 1;                        /* Set Himax
LCD controller flag      */
    /* Driving ability settings
----------------------------------------------*/
    wr_reg(0xEA, 0x00);               /* Power
control internal used (1)    */
    wr_reg(0xEB, 0x20);               /* Power
control internal used (2)    */
    wr_reg(0xEC, 0x0C);               /* Source
control internal used (1)   */
    wr_reg(0xED, 0xC7);               /* Source
control internal used (2)   */
    wr_reg(0xE8, 0x38);               /* Source
output period Normal mode   */
    wr_reg(0xE9, 0x10);               /* Source
output period Idle mode     */
    wr_reg(0xF1, 0x01);               /* RGB
18-bit interface ;0x0110       */
    wr_reg(0xF2, 0x10);

    /* Adjust the Gamma Curve
----------------------------------------------*/
    wr_reg(0x40, 0x01);
    wr_reg(0x41, 0x00);
    wr_reg(0x42, 0x00);
    wr_reg(0x43, 0x10);
    wr_reg(0x44, 0x0E);
    wr_reg(0x45, 0x24);
    wr_reg(0x46, 0x04);
    wr_reg(0x47, 0x50);
    wr_reg(0x48, 0x02);
    wr_reg(0x49, 0x13);
    wr_reg(0x4A, 0x19);
    wr_reg(0x4B, 0x19);
    wr_reg(0x4C, 0x16);

    wr_reg(0x50, 0x1B);
    wr_reg(0x51, 0x31);
    wr_reg(0x52, 0x2F);
    wr_reg(0x53, 0x3F);
    wr_reg(0x54, 0x3F);
    wr_reg(0x55, 0x3E);
    wr_reg(0x56, 0x2F);
    wr_reg(0x57, 0x7B);
    wr_reg(0x58, 0x09);
    wr_reg(0x59, 0x06);
    wr_reg(0x5A, 0x06);
    wr_reg(0x5B, 0x0C);
    wr_reg(0x5C, 0x1D);
    wr_reg(0x5D, 0xCC);

    /* Power voltage setting
----------------------------------------------*/
    wr_reg(0x1B, 0x1B);
    wr_reg(0x1A, 0x01);
    wr_reg(0x24, 0x2F);
    wr_reg(0x25, 0x57);
    wr_reg(0x23, 0x88);

    /* Power on setting
-----------------------------------------------------
*/
    wr_reg(0x18, 0x36);               /* Internal
oscillator frequency adj  */
```

```
   wr_reg(0x19, 0x01);              /* Enable
internal oscillator        */
   wr_reg(0x01, 0x00);              /* Normal
mode, no scrool            */
   wr_reg(0x1F, 0x88);              /* Power
control 6 - DDVDH Off      */
   delay(20);
   wr_reg(0x1F, 0x82);              /* Power
control 6 - Step-up: 3 x VCI */
   delay(5);
   wr_reg(0x1F, 0x92);              /* Power
control 6 - Step-up: On    */
   delay(5);
   wr_reg(0x1F, 0xD2);               /* Power
control 6 - VCOML active   */
   delay(5);

   /* Color selection
-------------------------------------------------------
-*/
   wr_reg(0x17, 0x55);              /* RGB,
System interface: 16 Bit/Pixel*/
   wr_reg(0x00, 0x00);              /*
Scrolling off, no standby        */

   /* Interface config
-------------------------------------------------------
*/
   wr_reg(0x2F, 0x11);              /* LCD
Drive: 1-line inversion      */
   wr_reg(0x31, 0x00);
   wr_reg(0x32, 0x00);              /* DPL=0,
HSPL=0, VSPL=0, EPL=0      */

   /* Display on setting
-----------------------------------------------------*
/
   wr_reg(0x28, 0x38);              /* PT(0,0)
active, VGL/VGL          */
   delay(20);
   wr_reg(0x28, 0x3C);              /* Display
active, VGL/VGL          */

  #if (LANDSCAPE == 1)
   #if (ROTATE180 == 0)
    wr_reg (0x16, 0xA8);
   #else
    wr_reg (0x16, 0x68);
   #endif
  #else
   #if (ROTATE180 == 0)
    wr_reg (0x16, 0x08);
   #else
    wr_reg (0x16, 0xC8);
   #endif
  #endif

   /* Display scrolling settings
-------------------------------------------*/
   wr_reg(0x0E, 0x00);              /* TFA
MSB                        */
   wr_reg(0x0F, 0x00);              /* TFA
LSB                        */
   wr_reg(0x10, 320 >> 8);           /* VSA
MSB                        */
   wr_reg(0x11, 320 &  0xFF);         /* VSA
LSB                        */
   wr_reg(0x12, 0x00);              /* BFA
MSB                        */
   wr_reg(0x13, 0x00);              /* BFA
LSB                        */
  }
  else {
   Himax = 0;                       /* This is not
Himax LCD controller   */
   /* Start Initial Sequence
-----------------------------------------------*/
  #if (ROTATE180 == 1)
   wr_reg(0x01, 0x0000);             /* Clear
SS bit                     */
```

```
  #else
   wr_reg(0x01, 0x0100);            /* Set SS
bit                    */
  #endif
   wr_reg(0x02, 0x0700);            /* Set 1
line inversion             */
   wr_reg(0x04, 0x0000);            /* Resize
register                */
   wr_reg(0x08, 0x0207);            /* 2 lines
front, 7 back porch       */
   wr_reg(0x09, 0x0000);            /* Set
non-disp area refresh cyc ISC  */
   wr_reg(0x0A, 0x0000);             /*
FMARK function                */
   wr_reg(0x0C, 0x0000);             /* RGB
interface setting             */
   wr_reg(0x0D, 0x0000);             /* Frame
marker Position             */
   wr_reg(0x0F, 0x0000);            /* RGB
interface polarity            */

   /* Power On sequence
-----------------------------------------------------
*/
   wr_reg(0x10, 0x0000);            /* Reset
Power Control 1             */
   wr_reg(0x11, 0x0000);            /* Reset
Power Control 2             */
   wr_reg(0x12, 0x0000);            /* Reset
Power Control 3             */
   wr_reg(0x13, 0x0000);            /* Reset
Power Control 4             */
   delay(20);                    /* Discharge
cap power voltage (200ms)*/
   wr_reg(0x10, 0x12B0);             /* SAP,
BT[3:0], AP, DSTB, SLP, STB   */
   wr_reg(0x11, 0x0007);             /*
DC1[2:0], DC0[2:0], VC[2:0]       */
   delay(5);                     /* Delay 50 ms
*/
   wr_reg(0x12, 0x01BD);             /*
VREG1OUT voltage                */
   delay(5);                     /* Delay 50 ms
*/
   wr_reg(0x13, 0x1400);             /*
VDV[4:0] for VCOM amplitude       */
   wr_reg(0x29, 0x000E);             /*
VCM[4:0] for VCOMH              */
   delay(5);                     /* Delay 50 ms
*/
   wr_reg(0x20, 0x0000);             /*
GRAM horizontal Address          */
   wr_reg(0x21, 0x0000);             /*
GRAM Vertical Address           */

   /* Adjust the Gamma Curve
-----------------------------------------------*/
   switch (driverCode) {
     case 0x5408:                  /* LCD with
SPFD5408 LCD Controller   */
       wr_reg(0x30, 0x0B0D);
       wr_reg(0x31, 0x1923);
       wr_reg(0x32, 0x1C26);
       wr_reg(0x33, 0x261C);
       wr_reg(0x34, 0x2419);
       wr_reg(0x35, 0x0D0B);
       wr_reg(0x36, 0x1006);
       wr_reg(0x37, 0x0610);
       wr_reg(0x38, 0x0706);
       wr_reg(0x39, 0x0304);
       wr_reg(0x3A, 0x0E05);
       wr_reg(0x3B, 0x0E01);
       wr_reg(0x3C, 0x010E);
       wr_reg(0x3D, 0x050E);
       wr_reg(0x3E, 0x0403);
       wr_reg(0x3F, 0x0607);
       break;

     case 0x9325:                  /* LCD with
RM68050 LCD Controller    */
```

```
        wr_reg(0x0030,0x0000);
        wr_reg(0x0031,0x0607);
        wr_reg(0x0032,0x0305);
        wr_reg(0x0035,0x0000);
        wr_reg(0x0036,0x1604);
        wr_reg(0x0037,0x0204);
        wr_reg(0x0038,0x0001);
        wr_reg(0x0039,0x0707);
        wr_reg(0x003C,0x0000);
        wr_reg(0x003D,0x000F);
        break;

      case 0x9320:                  /* LCD with
ILI9320 LCD Controller    */
      default:                      /* LCD with
other LCD Controller      */
        wr_reg(0x30, 0x0006);
        wr_reg(0x31, 0x0101);
        wr_reg(0x32, 0x0003);
        wr_reg(0x35, 0x0106);
        wr_reg(0x36, 0x0B02);
        wr_reg(0x37, 0x0302);
        wr_reg(0x38, 0x0707);
        wr_reg(0x39, 0x0007);
        wr_reg(0x3C, 0x0600);
        wr_reg(0x3D, 0x020B);
        break;
    }

    /* Set GRAM area
-------------------------------------------------------
---*/
    wr_reg(0x50, 0x0000);             /*
Horizontal GRAM Start Address      */
    wr_reg(0x51, (HEIGHT-1));         /*
Horizontal GRAM End   Address      */
    wr_reg(0x52, 0x0000);             /*
Vertical   GRAM Start Address      */
    wr_reg(0x53, (WIDTH-1));          /*
Vertical   GRAM End   Address      */

    /* Set Gate Scan Line
-------------------------------------------------------*
/
    switch (driverCode) {
      case 0x5408:                  /* LCD with
SPFD5408 LCD Controller   */
      case 0x9325:                  /* LCD with
RM68050 LCD Controller    */
        #if (LANDSCAPE ^ ROTATE180)
         wr_reg(0x60, 0x2700);
        #else
         wr_reg(0x60, 0xA700);
        #endif
         break;

      case 0x9320:                  /* LCD with
ILI9320 LCD Controller    */
      default:                      /* LCD with
other LCD Controller      */
        #if (LANDSCAPE ^ ROTATE180)
         wr_reg(0x60, 0xA700);
        #else
         wr_reg(0x60, 0x2700);
        #endif
         break;
    }
    wr_reg(0x61, 0x0001);             /*
NDL,VLE, REV                      */
    wr_reg(0x6A, 0x0000);             /* Set
scrolling line                */

    /* Partial Display Control
-----------------------------------------------*/
    wr_reg(0x80, 0x0000);
    wr_reg(0x81, 0x0000);
    wr_reg(0x82, 0x0000);
    wr_reg(0x83, 0x0000);
    wr_reg(0x84, 0x0000);
    wr_reg(0x85, 0x0000);
```

```
   /* Panel Control
-------------------------------------------------
---*/
   wr_reg(0x90, 0x0010);
   wr_reg(0x92, 0x0000);
   wr_reg(0x93, 0x0003);
   wr_reg(0x95, 0x0110);
   wr_reg(0x97, 0x0000);
   wr_reg(0x98, 0x0000);

   /* Set GRAM write direction
      I/D=11 (Horizontal : increment, Vertical
: increment)                */
  #if (LANDSCAPE == 1)
   /* AM=1   (address is updated in vertical
writing direction)          */
   wr_reg(0x03, 0x1038);
  #else
   /* AM=0   (address is updated in
horizontal writing direction)          */
   wr_reg(0x03, 0x1030);
  #endif

   wr_reg(0x07, 0x0137);            /* 262K
color and display ON         */
  }
  LPC_GPIO4->FIOSET = 0x10000000;
}



/*********************************
*********************************
********
* Set draw window region
*
*   Parameter:      x:       horizontal position
*
*                   y:       vertical position
*
*                   w:       window width in pixel
*
*                   h:       window height in pixels
*
*   Return:
*
*********************************
*********************************
*******/

void GLCD_SetWindow (unsigned int x,
unsigned int y, unsigned int w, unsigned int
h) {
  unsigned int xe, ye;

  if (Himax) {
    xe = x+w-1;
    ye = y+h-1;

    wr_reg(0x02, x  >>    8);          /* Column
address start MSB          */
    wr_reg(0x03, x  &  0xFF);          /*
Column address start LSB          */
    wr_reg(0x04, xe >>    8);          /*
Column address end MSB          */
    wr_reg(0x05, xe &  0xFF);          /*
Column address end LSB          */

    wr_reg(0x06, y  >>    8);          /* Row
address start MSB          */
    wr_reg(0x07, y  &  0xFF);          /* Row
address start LSB          */
    wr_reg(0x08, ye >>    8);          /* Row
address end MSB          */
    wr_reg(0x09, ye &  0xFF);          /* Row
address end LSB          */
  }
  else {
   #if (LANDSCAPE == 1)
```

```
   wr_reg(0x50, y);                /* Vertical
GRAM Start Address      */
   wr_reg(0x51, y+h-1);             /* Vertical
GRAM End   Address (-1) */
   wr_reg(0x52, x);                /*
Horizontal GRAM Start Address      */
   wr_reg(0x53, x+w-1);             /*
Horizontal GRAM End   Address (-1) */
   wr_reg(0x20, y);
   wr_reg(0x21, x);
  #else
   wr_reg(0x50, x);                /*
Horizontal GRAM Start Address      */
   wr_reg(0x51, x+w-1);             /*
Horizontal GRAM End   Address (-1) */
   wr_reg(0x52, y);                /* Vertical
GRAM Start Address      */
   wr_reg(0x53, y+h-1);             /* Vertical
GRAM End   Address (-1) */
   wr_reg(0x20, x);
   wr_reg(0x21, y);
  #endif
 }
}



/*********************************
**********************************
********
* Set draw window region to whole screen
*
*   Parameter:
*
*   Return:
*
*********************************
**********************************
*******/

void GLCD_WindowMax (void) {
  GLCD_SetWindow (0, 0, WIDTH,
HEIGHT);
}


/*********************************
**********************************
********
* Draw a pixel in foreground color
*
*   Parameter:      x:        horizontal position
*
*                   y:        vertical position
*
*   Return:
*
*********************************
**********************************
*******/

void GLCD_PutPixel (unsigned int x,
unsigned int y) {

  if (Himax) {
   wr_reg(0x02, x >>    8);          /* Column
address start MSB          */
   wr_reg(0x03, x &  0xFF);          /*
Column address start LSB          */
   wr_reg(0x04, x >>    8);          /* Column
address end MSB            */
   wr_reg(0x05, x &  0xFF);          /*
Column address end LSB            */

   wr_reg(0x06, y >>    8);          /* Row
address start MSB             */
   wr_reg(0x07, y &  0xFF);          /* Row
address start LSB             */
   wr_reg(0x08, y >>    8);          /* Row
address end MSB               */
```

```
   wr_reg(0x09, y &  0xFF);          /* Row
address end LSB              */
  }
  else {
   #if (LANDSCAPE == 1)
    wr_reg(0x20, y);
    wr_reg(0x21, x);
   #else
    wr_reg(0x20, x);
    wr_reg(0x21, y);
   #endif
  }

  wr_cmd(0x22);
  wr_dat(Color[TXT_COLOR]);
}


/*********************************
*********************************
********
* Set foreground color
*
*   Parameter:      color:    foreground color
*
*   Return:
*
*********************************
*********************************
*******/

void GLCD_SetTextColor (unsigned short
color) {

  Color[TXT_COLOR] = color;
}


/*********************************
*********************************
********
* Set background color
*
*   Parameter:      color:    background color
*
*   Return:
*
*********************************
*********************************
*******/

void GLCD_SetBackColor (unsigned short
color) {

  Color[BG_COLOR] = color;
}


/*********************************
*********************************
********
* Clear display
*
*   Parameter:      color:    display clearing
color                *
*   Return:
*
*********************************
*********************************
*******/

void GLCD_Clear (unsigned short color) {
  unsigned int i;

  GLCD_WindowMax();
  wr_cmd(0x22);
  wr_dat_start();
```

```
  for(i = 0; i < (WIDTH*HEIGHT); i++)
    wr_dat_only(color);
  wr_dat_stop();
}


/*********************************
*********************************
********
* Draw character on given position
*
*   Parameter:      x:        horizontal position
*
*                   y:        vertical position
*
*                   cw:       character width in pixel
*
*                   ch:       character height in
pixels                        *
*                   c:        pointer to character
bitmap                        *
*   Return:
*
*********************************
*********************************
*******/

void GLCD_DrawChar (unsigned int x,
unsigned int y, unsigned int cw, unsigned int
ch, unsigned char *c) {
  unsigned int i, j, k, pixs;

  GLCD_SetWindow(x, y, cw, ch);

  wr_cmd(0x22);
  wr_dat_start();

  k  = (cw + 7)/8;

  if (k == 1) {
    for (j = 0; j < ch; j++) {
      pixs = *(unsigned char  *)c;
      c += 1;

      for (i = 0; i < cw; i++) {
        wr_dat_only (Color[(pixs >> i) & 1]);
      }
    }
  }
  else if (k == 2) {
    for (j = 0; j < ch; j++) {
      pixs = *(unsigned short *)c;
      c += 2;

      for (i = 0; i < cw; i++) {
        wr_dat_only (Color[(pixs >> i) & 1]);
      }
    }
  }
  wr_dat_stop();
}


/*********************************
*********************************
********
* Disply character on given line
*
*   Parameter:      ln:       line number
*
*                   col:      column number
*
*                   fi:       font index (0 = 6x8, 1 =
16x24)                        *
*                   c:        ascii character
*
*   Return:
*
```

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

void GLCD_DisplayChar (unsigned int ln,
unsigned int col, unsigned char fi, unsigned
char c) {

  c -= 32;
  switch (fi) {
    case 0:  /* Font 6 x 8 */
      GLCD_DrawChar(col *  6, ln *  8,  6,  8,
(unsigned char *)&Font_6x8_h  [c * 8]);
      break;
    case 1:  /* Font 16 x 24 */
      GLCD_DrawChar(col * 16, ln * 24, 16,
24, (unsigned char *)&Font_16x24_h[c *
24]);
      break;
  }
}


/**********
**********
********
* Disply string on given line
*
*   Parameter:      ln:       line number
*
*                   col:      column number
*
*                   fi:       font index (0 = 6x8, 1 =
16x24)                *
*                   s:        pointer to string
*
*   Return:
*
**********
**********
*******/

void GLCD_DisplayString (unsigned int ln,
unsigned int col, unsigned char fi, unsigned
char *s) {

  while (*s) {
    GLCD_DisplayChar(ln, col++, fi, *s++);
  }
}


/**********
**********
********
* Clear given line
*
*   Parameter:      ln:       line number
*
*                   fi:       font index (0 = 6x8, 1 =
16x24)                *
*   Return:
*
**********
**********
*******/

void GLCD_ClearLn (unsigned int ln,
unsigned char fi) {
  unsigned char i;
  unsigned char buf[60];

  GLCD_WindowMax();
  switch (fi) {
    case 0:  /* Font 6 x 8 */
      for (i = 0; i < (WIDTH+5)/6; i++)
        buf[i] = ' ';
      buf[i+1] = 0;
```

```
    break;
   case 1:  /* Font 16 x 24 */
    for (i = 0; i < (WIDTH+15)/16; i++)
     buf[i] = ' ';
    buf[i+1] = 0;
    break;
  }
  GLCD_DisplayString (ln, 0, fi, buf);
}

/*********************************
*********************************
********
* Draw bargraph
*
*   Parameter:      x:        horizontal position
*
*                   y:        vertical position
*
*                   w:        maximum width of
bargraph (in pixels)           *
*                   h:        bargraph height
*
*                   val:      value of active bargraph
(in 1/1024)           *
*   Return:
*
*********************************
*********************************
*******/

void GLCD_Bargraph (unsigned int x,
unsigned int y, unsigned int w, unsigned int
h, unsigned int val) {
  int i,j;

  val = (val * w) >> 10;              /* Scale
value                     */
  GLCD_SetWindow(x, y, w, h);
  wr_cmd(0x22);
  wr_dat_start();
  for (i = 0; i < h; i++) {
   for (j = 0; j <= w-1; j++) {
    if(j >= val) {
     wr_dat_only(Color[BG_COLOR]);
    } else {
     wr_dat_only(Color[TXT_COLOR]);
    }
   }
  }
  wr_dat_stop();
}


/*********************************
*********************************
********
* Display graphical bitmap image at position
x horizontally and y vertically   *
* (This function is optimized for 16 bits per
pixel format, it has to be       *
*  adapted for any other bits per pixel
format)                          *
*   Parameter:      x:        horizontal position
*
*                   y:        vertical position
*
*                   w:        width of bitmap
*
*                   h:        height of bitmap
*
*                   bitmap:   address at which the
bitmap data resides         *
*   Return:
*
*********************************
*********************************
*******/
```

```
void GLCD_Bitmap (unsigned int x,
unsigned int y, unsigned int w, unsigned int
h, unsigned char *bitmap) {
  int i, j;
  unsigned short *bitmap_ptr = (unsigned
short *)bitmap;

  GLCD_SetWindow (x, y, w, h);

  wr_cmd(0x22);
  wr_dat_start();
  for (i = (h-1)*w; i > -1; i -= w) {
    for (j = 0; j < w; j++) {
      wr_dat_only (bitmap_ptr[i+j]);
    }
  }
  wr_dat_stop();
}



/**********************************
**********************************
********
* Scroll content of the whole display for dy
pixels vertically                *
*   Parameter:      dy:       number of pixels
for vertical scroll           *
*   Return:
*
**********************************
**********************************
*******/

void GLCD_ScrollVertical (unsigned int dy)
{
#if (LANDSCAPE == 0)
  static unsigned int y = 0;

  y = y + dy;

  while (y >= HEIGHT)
    y -= HEIGHT;

  if (Himax) {
    wr_reg(0x01, 0x08);
    wr_reg(0x14, y>>8);              /* VSP
MSB                          */
    wr_reg(0x15, y&0xFF);             /* VSP
LSB                          */
  }
  else {
    wr_reg(0x6A, y);
    wr_reg(0x61, 3);
  }
#endif
}



/**********************************
**********************************
********
* Write a command to the LCD controller
*
*   Parameter:      cmd:      command to write
to the LCD                    *
*   Return:
*
**********************************
**********************************
*******/
void GLCD_WrCmd (unsigned char cmd) {
  wr_cmd (cmd);
}



/**********************************
**********************************
********
* Write a value into LCD controller register
*
```

```
*   Parameter:      reg:      lcd register address
*
*                   val:      value to write into reg
*
*   Return:
*
************************************
************************************
*******/
void GLCD_WrReg (unsigned char reg,
unsigned short val) {
  wr_reg (reg, val);
}
/************************************
************************************
*******/
```

G. IRQ.c

```
/************************************
************************************
*******/
/* IRQ.C: IRQ Handlers
*/
/************************************
************************************
*******/
/* This file is part of the uVision/ARM
development tools.                  */
/* Copyright (c) 2005-2009 Keil Software.
All rights reserved.                */
/* This software may only be used under the
terms of a valid, current,          */
/* end user licence from KEIL for a
compatible version of KEIL software       */
/* development tools. Nothing else gives
you the right to use this software.  */
/************************************
************************************
*******/


unsigned long ticks = 0;
unsigned char ClockLEDOn;             /* On
for 30 ms every 50 ms          */
unsigned char ClockLEDOff;            /* On
for 20 ms every 50 ms          */
unsigned char ClockANI;               /* Clock
for Animation 150 ms        */


void SysTick_Handler (void) {         /*
SysTick Interrupt Handler (10ms)   */
  ticks++;
  switch (ticks) {
    case  3:
      ClockLEDOff = 1;
      break;
```

```
    case  5:
      ClockLEDOn  = 1;
      break;
    case  8:
      ClockLEDOff = 1;
      break;
    case 10:
      ClockLEDOn  = 1;
      break;
    case 13:
      ClockLEDOff = 1;
      break;
    case 15:
      ticks       = 0;
      ClockANI    = 1;
      ClockLEDOn  = 1;
    default:
      break;
  }
}
```

H. KBD.c

```
/*----------------------------------------------------------------------------
 * Name:    KBD.c
 * Purpose: MCB1700 low level Joystick
 * Version: V2.0

*----------------------------------------------------------------------------
 * This file is part of the uVision/ARM development tools.
 * This software may only be used under the terms of a valid, current,
 * end user licence from KEIL for a compatible version of KEIL software
 * development tools. Nothing else gives you the right to use this software.
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2008 Keil - An ARM Company. All rights reserved.

*----------------------------------------------------------------------------
 * History:
 *          V2.0 - updated by Anita Tino for LPC1768

*----------------------------------------------------------------------------*/
#include <LPC17xx.H>                    /* LPC17xx definitions               */
#include "KBD.h"
uint32_t KBD_val  = 0;
/*----------------------------------------------------------------------------
  initialize Joystick
```

```
*-------------------------------------------------
-----------------------*/
void KBD_Init (void) {
  LPC_SC->PCONP     |= (1 << 15);
/* enable power to GPIO & IOCON  */
/* P1.20, P1.23..26 is GPIO (Joystick) */
  LPC_PINCON->PINSEL3 &= ~((3<<
8)|(3<<14)|(3<<16)|(3<<18)|(3<<20));
/* P1.20, P1.23..26 is input */
  LPC_GPIO1->FIODIR   &=
~((1<<20)|(1<<23)|(1<<24)|(1<<25)|(1<<26
));
}
/*-------------------------------------------------
-------------------------
  Get Joystick value.. part of get_button

*-------------------------------------------------
-----------------------*/
uint32_t KBD_get  (void) {
  uint32_t kbd_val;
  kbd_val = (LPC_GPIO1->FIOPIN >> 20)
& KBD_MASK;
  return (kbd_val);
}
/*-------------------------------------------------
-------------------------
  Get Joystick value

*-------------------------------------------------
-----------------------*/
uint32_t get_button (void) {
   uint32_t val = 0;
   val = KBD_get();                    /* read
Joystick state    */
   val = (~val & KBD_MASK);            /*
key pressed is read as a non '0' value*/
   return (val);
}
```

I. LCD.c

```
/*-------------------------------------------------
-------------------------
 * Name:    LED.c
 * Purpose: low level LED functions
 * Note(s):

*-------------------------------------------------
-------------------------
 * This file is part of the uVision/ARM
development tools.
 * This software may only be used under the
terms of a valid, current,
 * end user licence from KEIL for a
compatible version of KEIL software
 * development tools. Nothing else gives
you the right to use this software.
 *
 * This software is supplied "AS IS" without
warranties of any kind.
 *
 * Copyright (c) 2009-2011 Keil - An ARM
Company. All rights reserved.

*-------------------------------------------------
-----------------------*/

#include "LPC17xx.H"                    /*
LPC17xx definitions          */
#include "LED.h"

const unsigned long led_mask[] = {
1UL<<28, 1UL<<29, 1UL<<31, 1UL<< 2,
                          1UL<< 3, 1UL<< 4,
1UL<< 5, 1UL<< 6 };


/*-------------------------------------------------
-------------------------
  initialize LED Pins
```

```
*----------------------------------------------------
-----------------------*/
void LED_Init (void) {

  LPC_SC->PCONP     |= (1 << 15);
/* enable power to GPIO & IOCON  */

  LPC_GPIO1->FIODIR |= 0xB0000000;
/* LEDs on PORT1 are output      */
  LPC_GPIO2->FIODIR |= 0x0000007C;
/* LEDs on PORT2 are output      */
}

/*----------------------------------------------------
-------------------------
  Function that turns on requested LED

*----------------------------------------------------
-----------------------*/
void LED_On (unsigned int num) {

  if (num < 3) LPC_GPIO1->FIOPIN |=
led_mask[num];
  else         LPC_GPIO2->FIOPIN |=
led_mask[num];
}

/*----------------------------------------------------
-------------------------
  Function that turns off requested LED

*----------------------------------------------------
-----------------------*/
void LED_Off (unsigned int num) {

  if (num < 3) LPC_GPIO1->FIOPIN &=
~led_mask[num];
  else         LPC_GPIO2->FIOPIN &=
~led_mask[num];
}

/*----------------------------------------------------
-------------------------
  Function that outputs value to LEDs

*----------------------------------------------------
-----------------------*/
void LED_Out(unsigned int value) {
  int i;

  for (i = 0; i < LED_NUM; i++) {
    if (value & (1<<i)) {
      LED_On (i);
    } else {
      LED_Off(i);
    }
  }
}
```

J. system_LPC17xx.c

```
/**************************************************************************//**
 * @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.
 *
 ******************************************************************************/


#include <stdint.h>
#include "LPC17xx.h"


/** @addtogroup LPC17xx_System
 * @{
 */

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

/*---------------------- Clock Configuration ---------------------------------
//
// <e> Clock Configuration
//   <h> System Controls and Status Register (SCS)
//     <o1.4>    OSCRANGE: Main Oscillator Range Select
//                 <0=>  1 MHz to 20 MHz
//                 <1=> 15 MHz to 25 MHz
//     <e1.5>      OSCEN: Main Oscillator Enable
//     </e>
//   </h>
//
//   <h> Clock Source Select Register (CLKSRCSEL)
//     <o2.0..1>   CLKSRC: PLL Clock Source Selection
//                 <0=> Internal RC oscillator
//                 <1=> Main oscillator
```

```
//                 <2=> RTC oscillator
//   </h>
//
//   <e3> PLL0 Configuration (Main PLL)
//     <h> PLL0 Configuration Register
(PLL0CFG)
//                 <i> F_cco0 = (2 * M * F_in) /
N
//                 <i> F_in must be in the range
of 32 kHz to 50 MHz
//                 <i> F_cco0 must be in the
range of 275 MHz to 550 MHz
//       <o4.0..14>  MSEL: PLL Multiplier
Selection
//                 <6-32768><#-1>
//                 <i> M Value
//       <o4.16..23> NSEL: PLL Divider
Selection
//                 <1-256><#-1>
//                 <i> N Value
//     </h>
//   </e>
//
//   <e5> PLL1 Configuration (USB PLL)
//     <h> PLL1 Configuration Register
(PLL1CFG)
//                 <i> F_usb = M * F_osc or
F_usb = F_cco1 / (2 * P)
//                 <i> F_cco1 = F_osc * M * 2 *
P
//                 <i> F_cco1 must be in the
range of 156 MHz to 320 MHz
//       <o6.0..4>   MSEL: PLL Multiplier
Selection
//                 <1-32><#-1>
//                 <i> M Value (for USB
maximum value is 4)
//       <o6.5..6>   PSEL: PLL Divider
Selection
//                 <0=> 1
//                 <1=> 2
//                 <2=> 4
//                 <3=> 8
//                 <i> P Value
//     </h>
//   </e>
//
//   <h> CPU Clock Configuration Register
(CCLKCFG)
//     <o7.0..7>  CCLKSEL: Divide Value for
CPU Clock from PLL0
//                 <1-256><#-1>
//   </h>
//
//   <h> USB Clock Configuration Register
(USBCLKCFG)
//     <o8.0..3>   USBSEL: Divide Value for
USB Clock from PLL0
//                 <0-15>
//                 <i> Divide is USBSEL + 1
//   </h>
//
//   <h> Peripheral Clock Selection Register
0 (PCLKSEL0)
//     <o9.0..1>    PCLK_WDT: Peripheral
Clock Selection for WDT
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.2..3>    PCLK_TIMER0: Peripheral
Clock Selection for TIMER0
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.4..5>    PCLK_TIMER1: Peripheral
Clock Selection for TIMER1
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
```

```
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.6..7>    PCLK_UART0: Peripheral
Clock Selection for UART0
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.8..9>    PCLK_UART1: Peripheral
Clock Selection for UART1
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.12..13>  PCLK_PWM1: Peripheral
Clock Selection for PWM1
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.14..15>  PCLK_I2C0: Peripheral
Clock Selection for I2C0
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.16..17>  PCLK_SPI: Peripheral
Clock Selection for SPI
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.20..21>  PCLK_SSP1: Peripheral
Clock Selection for SSP1
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.22..23>  PCLK_DAC: Peripheral
Clock Selection for DAC
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.24..25>  PCLK_ADC: Peripheral
Clock Selection for ADC
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
//     <o9.26..27>  PCLK_CAN1: Peripheral
Clock Selection for CAN1
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 6
//     <o9.28..29>  PCLK_CAN2: Peripheral
Clock Selection for CAN2
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 6
//     <o9.30..31>  PCLK_ACF: Peripheral
Clock Selection for ACF
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 6
//   </h>
//
//   <h> Peripheral Clock Selection Register
1 (PCLKSEL1)
//     <o10.0..1>   PCLK_QEI: Peripheral
Clock Selection for the Quadrature Encoder
Interface
//                 <0=> Pclk = Cclk / 4
//                 <1=> Pclk = Cclk
//                 <2=> Pclk = Cclk / 2
//                 <3=> Pclk = Cclk / 8
```

```
//     <o10.2..3>   PCLK_GPIO: Peripheral
Clock Selection for GPIOs
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.4..5>   PCLK_PCB: Peripheral
Clock Selection for the Pin Connect Block
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.6..7>   PCLK_I2C1: Peripheral
Clock Selection for I2C1
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.10..11> PCLK_SSP0: Peripheral
Clock Selection for SSP0
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.12..13> PCLK_TIMER2:
Peripheral Clock Selection for TIMER2
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.14..15> PCLK_TIMER3:
Peripheral Clock Selection for TIMER3
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.16..17> PCLK_UART2:
Peripheral Clock Selection for UART2
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.18..19> PCLK_UART3:
Peripheral Clock Selection for UART3
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.20..21> PCLK_I2C2: Peripheral
Clock Selection for I2C2
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.22..23> PCLK_I2S: Peripheral
Clock Selection for I2S
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.26..27> PCLK_RIT: Peripheral
Clock Selection for the Repetitive Interrupt
Timer
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.28..29> PCLK_SYSCON:
Peripheral Clock Selection for the System
Control Block
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
//     <o10.30..31> PCLK_MC: Peripheral
Clock Selection for the Motor Control PWM
//                  <0=> Pclk = Cclk / 4
//                  <1=> Pclk = Cclk
//                  <2=> Pclk = Cclk / 2
//                  <3=> Pclk = Cclk / 8
```

```
//   </h>
//
//   <h> Power Control for Peripherals Register (PCONP)
//     <o11.1>     PCTIM0: Timer/Counter 0 power/clock enable
//     <o11.2>     PCTIM1: Timer/Counter 1 power/clock enable
//     <o11.3>     PCUART0: UART 0 power/clock enable
//     <o11.4>     PCUART1: UART 1 power/clock enable
//     <o11.6>     PCPWM1: PWM 1 power/clock enable
//     <o11.7>     PCI2C0: I2C interface 0 power/clock enable
//     <o11.8>     PCSPI: SPI interface power/clock enable
//     <o11.9>     PCRTC: RTC power/clock enable
//     <o11.10>    PCSSP1: SSP interface 1 power/clock enable
//     <o11.12>    PCAD: A/D converter power/clock enable
//     <o11.13>    PCCAN1: CAN controller 1 power/clock enable
//     <o11.14>    PCCAN2: CAN controller 2 power/clock enable
//     <o11.15>    PCGPIO: GPIOs power/clock enable
//     <o11.16>    PCRIT: Repetitive interrupt timer power/clock enable
//     <o11.17>    PCMC: Motor control PWM power/clock enable
//     <o11.18>    PCQEI: Quadrature encoder interface power/clock enable
//     <o11.19>    PCI2C1: I2C interface 1 power/clock enable
//     <o11.21>    PCSSP0: SSP interface 0 power/clock enable
//     <o11.22>    PCTIM2: Timer 2 power/clock enable
//     <o11.23>    PCTIM3: Timer 3 power/clock enable
//     <o11.24>    PCUART2: UART 2 power/clock enable
//     <o11.25>    PCUART3: UART 3 power/clock enable
//     <o11.26>    PCI2C2: I2C interface 2 power/clock enable
//     <o11.27>    PCI2S: I2S interface power/clock enable
//     <o11.29>    PCGPDMA: GP DMA function power/clock enable
//     <o11.30>    PCENET: Ethernet block power/clock enable
//     <o11.31>    PCUSB: USB interface power/clock enable
//   </h>
//
//   <h> Clock Output Configuration Register (CLKOUTCFG)
//     <o12.0..3>  CLKOUTSEL: Selects clock source for CLKOUT
//                 <0=> CPU clock
//                 <1=> Main oscillator
//                 <2=> Internal RC oscillator
//                 <3=> USB clock
//                 <4=> RTC oscillator
//     <o12.4..7>  CLKOUTDIV: Selects clock divider for CLKOUT
//                 <1-16><#-1>
//     <o12.8>     CLKOUT_EN: CLKOUT enable control
//   </h>
//
// </e>
*/
```

```
/** @addtogroup
LPC17xx_System_Defines  LPC17xx
System Defines
  @{
 */

#define CLOCK_SETUP          1
#define SCS_Val              0x00000020
#define CLKSRCSEL_Val        0x00000001
#define PLL0_SETUP           1
#define PLL0CFG_Val          0x00050063
#define PLL1_SETUP           1
#define PLL1CFG_Val          0x00000023
#define CCLKCFG_Val          0x00000003
#define USBCLKCFG_Val
0x00000000
#define PCLKSEL0_Val         0x00000000
#define PCLKSEL1_Val         0x00000000
#define PCONP_Val            0x042887DE
#define CLKOUTCFG_Val
0x00000000


/*---------------------- Flash Accelerator
Configuration ----------------------
//
// <e> Flash Accelerator Configuration
//   <o1.12..15> FLASHTIM: Flash Access
Time
//             <0=> 1 CPU clock (for CPU
clock up to 20 MHz)
//             <1=> 2 CPU clocks (for CPU
clock up to 40 MHz)
//             <2=> 3 CPU clocks (for CPU
clock up to 60 MHz)
//             <3=> 4 CPU clocks (for CPU
clock up to 80 MHz)
//             <4=> 5 CPU clocks (for CPU
clock up to 100 MHz)
//             <5=> 6 CPU clocks (for any CPU
clock)
// </e>
*/
#define FLASH_SETUP          1
#define FLASHCFG_Val         0x00004000

/*
//-------- <<< end of configuration section
>>> ------------------------------
*/

/*----------------------------------------------
------------------------
  Check the register settings

*----------------------------------------------
------------------------*/
#define CHECK_RANGE(val, min, max)
((val < min) || (val > max))
#define CHECK_RSVD(val, mask)
(val & mask)

/* Clock Configuration
----------------------------------------------------
-*/
#if (CHECK_RSVD((SCS_Val),
~0x00000030))
  #error "SCS: Invalid values of reserved
bits!"
#endif

#if (CHECK_RANGE((CLKSRCSEL_Val),
0, 2))
  #error "CLKSRCSEL: Value out of
range!"
#endif

#if (CHECK_RSVD((PLL0CFG_Val),
~0x00FF7FFF))
```

```
  #error "PLL0CFG: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PLL1CFG_Val), ~0x0000007F))
  #error "PLL1CFG: Invalid values of reserved bits!"
#endif

#if (PLL0_SETUP)            /* if PLL0 is used */
 #if (CCLKCFG_Val < 2)     /* CCLKSEL must be greater then 1 */
   #error "CCLKCFG: CCLKSEL must be greater then 1 if PLL0 is used!"
 #endif
#endif

#if (CHECK_RANGE((CCLKCFG_Val), 0, 255))
  #error "CCLKCFG: Value out of range!"
#endif

#if (CHECK_RSVD((USBCLKCFG_Val), ~0x0000000F))
  #error "USBCLKCFG: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PCLKSEL0_Val), 0x000C0C00))
  #error "PCLKSEL0: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PCLKSEL1_Val), 0x03000300))
  #error "PCLKSEL1: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PCONP_Val), 0x10100821))
  #error "PCONP: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((CLKOUTCFG_Val), ~0x000001FF))
  #error "CLKOUTCFG: Invalid values of reserved bits!"
#endif

/* Flash Accelerator Configuration ----------------------------------------*/
#if (CHECK_RSVD((FLASHCFG_Val), ~0x0000F000))
  #error "FLASHCFG: Invalid values of reserved bits!"
#endif


/*----------------------------------------------------------------------------
  DEFINES

*----------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------
  Define clocks

*----------------------------------------------------------------------------*/
#define XTAL        (12000000UL)       /* Oscillator frequency             */
#define OSC_CLK     (      XTAL)       /* Main oscillator frequency        */
```

```
#define RTC_CLK     (   32768UL)        /*
RTC oscillator frequency          */
#define IRC_OSC     ( 4000000UL)        /*
Internal RC oscillator frequency   */


/* F_cco0 = (2 * M * F_in) / N  */
#define __M             (((PLL0CFG_Val      )
& 0x7FFF) + 1)
#define __N             (((PLL0CFG_Val >>
16) & 0x00FF) + 1)
#define __FCCO(__F_IN)    ((2ULL * __M
* __F_IN) / __N)
#define __CCLK_DIV
(((CCLKCFG_Val      ) & 0x00FF) + 1)

/* Determine core clock frequency
according to settings */
 #if (PLL0_SETUP)
   #if   ((CLKSRCSEL_Val & 0x03) == 1)
      #define __CORE_CLK
(__FCCO(OSC_CLK) / __CCLK_DIV)
   #elif ((CLKSRCSEL_Val & 0x03) == 2)
      #define __CORE_CLK
(__FCCO(RTC_CLK) / __CCLK_DIV)
   #else
      #define __CORE_CLK
(__FCCO(IRC_OSC) / __CCLK_DIV)
   #endif
 #else
   #if   ((CLKSRCSEL_Val & 0x03) == 1)
      #define __CORE_CLK (OSC_CLK
/ __CCLK_DIV)
   #elif ((CLKSRCSEL_Val & 0x03) == 2)
      #define __CORE_CLK (RTC_CLK
/ __CCLK_DIV)
   #else
      #define __CORE_CLK (IRC_OSC
/ __CCLK_DIV)
   #endif
 #endif

/**
 * @}
 */


/** @addtogroup
LPC17xx_System_Public_Variables
LPC17xx System Public Variables
  @{
 */
/*----------------------------------------------
-------------------------
  Clock Variable definitions

*----------------------------------------------
-------------------------*/
uint32_t SystemCoreClock =
__CORE_CLK;/*!< System Clock
Frequency (Core Clock)*/

/**
 * @}
 */


/** @addtogroup
LPC17xx_System_Public_Functions
LPC17xx System Public Functions
  @{
 */

/**
 * Update SystemCoreClock variable
 *
 * @param  none
 * @return none
 *
```

```
 * @brief  Updates the SystemCoreClock
with current core Clock
 *         retrieved from cpu registers.
 */void SystemCoreClockUpdate (void)
/* Get Core Clock Frequency      */
{
  /* Determine clock frequency according to
clock register values             */
  if (((LPC_SC->PLL0STAT >> 24) & 3) ==
3) { /* If PLL0 enabled and connected */
    switch (LPC_SC->CLKSRCSEL & 0x03)
{
     case 0:                          /* Int. RC
oscillator => PLL0    */
     case 3:                          /* Reserved,
default to Int. RC  */
       SystemCoreClock = (IRC_OSC *
                        ((2ULL *
((LPC_SC->PLL0STAT & 0x7FFF) + 1)))  /
                        (((LPC_SC->PLL0STAT >>
16) & 0xFF) + 1)       /
                        ((LPC_SC->CCLKCFG &
0xFF)+ 1));
       break;
     case 1:                          /* Main
oscillator => PLL0       */
       SystemCoreClock = (OSC_CLK *
                        ((2ULL *
((LPC_SC->PLL0STAT & 0x7FFF) + 1)))  /
                        (((LPC_SC->PLL0STAT >>
16) & 0xFF) + 1)       /
                        ((LPC_SC->CCLKCFG &
0xFF)+ 1));
       break;
     case 2:                          /* RTC
oscillator => PLL0       */
       SystemCoreClock = (RTC_CLK *
                        ((2ULL *
((LPC_SC->PLL0STAT & 0x7FFF) + 1)))  /
                        (((LPC_SC->PLL0STAT >>
16) & 0xFF) + 1)       /
                        ((LPC_SC->CCLKCFG &
0xFF)+ 1));
       break;
    }
  } else {
    switch (LPC_SC->CLKSRCSEL & 0x03)
{
     case 0:                          /* Int. RC
oscillator => PLL0    */
     case 3:                          /* Reserved,
default to Int. RC  */
       SystemCoreClock = IRC_OSC /
((LPC_SC->CCLKCFG & 0xFF)+ 1);
       break;
     case 1:                          /* Main
oscillator => PLL0       */
       SystemCoreClock = OSC_CLK /
((LPC_SC->CCLKCFG & 0xFF)+ 1);
       break;
     case 2:                          /* RTC
oscillator => PLL0       */
       SystemCoreClock = RTC_CLK /
((LPC_SC->CCLKCFG & 0xFF)+ 1);
       break;
    }
  }

}

/**
 * Initialize the system
 *
 * @param  none
 * @return none
 *
 * @brief  Setup the microcontroller system.
 *         Initialize the System.
 */
```

```
void SystemInit (void)
{
#if (CLOCK_SETUP)                     /*
Clock Setup                        */
  LPC_SC->SCS       = SCS_Val;
  if (LPC_SC->SCS & (1 << 5)) {           /*
If Main Oscillator is enabled  */
    while ((LPC_SC->SCS & (1<<6)) ==
0);/* Wait for Oscillator to be ready    */
  }

  LPC_SC->CCLKCFG   = CCLKCFG_Val;
/* Setup Clock Divider                */
  /* Periphral clock must be selected before
PLL0 enabling and connecting
   * - according
errata.lpc1768-16.March.2010 -
   */
  LPC_SC->PCLKSEL0  =
PCLKSEL0_Val;     /* Peripheral Clock
Selection         */
  LPC_SC->PCLKSEL1  =
PCLKSEL1_Val;

  LPC_SC->CLKSRCSEL =
CLKSRCSEL_Val;    /* Select Clock Source
sysclk / PLL0  */

#if (PLL0_SETUP)
  LPC_SC->PLL0CFG   = PLL0CFG_Val;
/* configure PLL0                     */
  LPC_SC->PLL0FEED  = 0xAA;
  LPC_SC->PLL0FEED  = 0x55;

  LPC_SC->PLL0CON   = 0x01;           /*
PLL0 Enable                        */
  LPC_SC->PLL0FEED  = 0xAA;
  LPC_SC->PLL0FEED  = 0x55;
  while (!(LPC_SC->PLL0STAT &
(1<<26)));/* Wait for PLOCK0
*/

  LPC_SC->PLL0CON   = 0x03;           /*
PLL0 Enable & Connect              */
  LPC_SC->PLL0FEED  = 0xAA;
  LPC_SC->PLL0FEED  = 0x55;
  while ((LPC_SC->PLL0STAT & ((1<<25)
| (1<<24))) != ((1<<25) | (1<<24)));  /* Wait
for PLLC0_STAT & PLLE0_STAT */
#endif

#if (PLL1_SETUP)
  LPC_SC->PLL1CFG   = PLL1CFG_Val;
  LPC_SC->PLL1FEED  = 0xAA;
  LPC_SC->PLL1FEED  = 0x55;

  LPC_SC->PLL1CON   = 0x01;           /*
PLL1 Enable                        */
  LPC_SC->PLL1FEED  = 0xAA;
  LPC_SC->PLL1FEED  = 0x55;
  while (!(LPC_SC->PLL1STAT &
(1<<10)));/* Wait for PLOCK1
*/

  LPC_SC->PLL1CON   = 0x03;           /*
PLL1 Enable & Connect              */
  LPC_SC->PLL1FEED  = 0xAA;
  LPC_SC->PLL1FEED  = 0x55;
  while ((LPC_SC->PLL1STAT & ((1<< 9) |
(1<< 8))) != ((1<< 9) | (1<< 8)));  /* Wait for
PLLC1_STAT & PLLE1_STAT */
#else
  LPC_SC->USBCLKCFG =
USBCLKCFG_Val;    /* Setup USB Clock
Divider           */
#endif
```

```
  LPC_SC->PCONP     = PCONP_Val;
/* Power Control for Peripherals      */

  LPC_SC->CLKOUTCFG =
CLKOUTCFG_Val;    /* Clock Output
Configuration         */
#endif

#if (FLASH_SETUP == 1)                /*
Flash Accelerator Setup            */
  LPC_SC->FLASHCFG  =
(LPC_SC->FLASHCFG & ~0x0000F000) |
FLASHCFG_Val;
#endif
}


/**
 * Determine the CPU Clock Frequency
(SystemFrequency)
 *
 * @param none
 * @return none
 *
 * @brief        Function determines the
microcontroller's CPU clock frequency
 *                           It populates this value
into the "SystemFrequeny" variable.
 *                           Recommended to call
this function in main()
 */
#define IRC_OSC     ( 4000000UL)        /*
Internal RC oscillator frequency   */


/*----------------------------------------------------
------------------------
  Clock Variable definitions
*----------------------------------------------------
------------------------*/
uint32_t SystemFrequency = IRC_OSC;
/*!< System Clock Frequency (Core Clock)
*/

void SystemClockUpdate (void) {
          /* Determine clock frequency
according to clock register values             */
          if (((LPC_SC->PLL0STAT >>
24)&3)==3) {/* If PLL0 enabled and
connected      */
                    switch
(LPC_SC->CLKSRCSEL & 0x03) {
                              case 0:
/* Internal RC oscillator => PLL0     */
                              case 3:
/* Reserved, default to Internal RC   */

SystemFrequency = (IRC_OSC *
                                                  (((2
* ((LPC_SC->PLL0STAT & 0x7FFF) + 1)))
/

(((LPC_SC->PLL0STAT >> 16) & 0xFF) +
1))   /

((LPC_SC->CCLKCFG & 0xFF)+ 1));
                                        break;
                              case 1:
/* Main oscillator => PLL0            */

SystemFrequency = (OSC_CLK *
                                                  (((2
* ((LPC_SC->PLL0STAT & 0x7FFF) + 1)))
/

(((LPC_SC->PLL0STAT >> 16) & 0xFF) +
1))   /
```

```
                                                                    break;
((LPC_SC->CCLKCFG & 0xFF)+ 1));                                   }
                                    break;                      }
                            case 2:                     }
/* RTC oscillator => PLL0           */                 /**
                                                        * @}
SystemFrequency = (RTC_CLK *                            */
                                                (((2
* ((LPC_SC->PLL0STAT & 0x7FFF) + 1)))                   /**
/                                                       * @}
                                                        */
(((LPC_SC->PLL0STAT >> 16) & 0xFF) +
1))   /

((LPC_SC->CCLKCFG & 0xFF)+ 1));
                                    break;
                    }

          } else {
                    switch
(LPC_SC->CLKSRCSEL & 0x03) {
                            case 0:
/* Internal RC oscillator => PLL0     */
                            case 3:
/* Reserved, default to Internal RC   */

SystemFrequency = IRC_OSC /
((LPC_SC->CCLKCFG & 0xFF)+ 1);
                                    break;
                            case 1:
/* Main oscillator => PLL0          */

SystemFrequency = OSC_CLK /
((LPC_SC->CCLKCFG & 0xFF)+ 1);
                                    break;
                            case 2:
/* RTC oscillator => PLL0           */

SystemFrequency = RTC_CLK /
((LPC_SC->CCLKCFG & 0xFF)+ 1);
```

K. 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
 */

__inline 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
 */

__inline 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
 */

__inline 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
 */

__inline 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_DESCRIPTO
R *)pD)->bLength != 0) {
              pD +=
((USB_CONFIGURATION_DESCRIPTOR
*)pD)->wTotalLength;
            }
          }
          if
(((USB_CONFIGURATION_DESCRIPTO
R *)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
 */

__inline 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
 */

__inline 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_DESCRIPTO
R *)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_DESCRIPTO
R *)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_DESC
RIPTOR *)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
 */

__inline 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
 */

__inline 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_DESCRIPTO
R *)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_DESC
RIPTOR *)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.bmRequestTy
pe.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_CONFIGURATIO
N:
            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_COMMA
ND:
              EP0Data.pData = EP0Buf;
/* data to be received, see USB_EVT_OUT
*/
              goto setup_class_ok;
            case
CDC_GET_ENCAPSULATED_RESPONS
E:
```

```
            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.wVal
ue.W)) {
              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.wV
alue.W)) {
              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.w
Value.W)) {
              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_COMMA
ND:
                      if
(CDC_SendEncapsulatedCommand()) {
                        USB_StatusInStage();
/* send Acknowledge */
                        goto out_class_ok;
                      }
                      break;
                    case
CDC_SET_COMM_FEATURE:
                      if
(CDC_SetCommFeature(SetupPacket.wValu
e.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;

  }
}
```

L. 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_SI
ZE     +
    USB_INTERFACE_DESC_SIZE
+
    USB_INTERFACE_DESC_SIZE
+

AUDIO_STREAMING_INTERFACE_DES
C_SIZE +

AUDIO_FORMAT_TYPE_I_DESC_SZ(1)
+

AUDIO_STANDARD_ENDPOINT_DESC
_SIZE   +
```

```
AUDIO_STREAMING_ENDPOINT_DES
C_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_TY
PE,      /* 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_SI
ZE
  ),
  0x01,                         /* bInCollection
*/
  0x01,                         /* baInterfaceNr
*/
/* Audio Input Terminal */

AUDIO_INPUT_TERMINAL_DESC_SIZE
,       /* bLength */

AUDIO_INTERFACE_DESCRIPTOR_TY
PE,      /* 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_AUDIOSTREAMIN
G,       /* bInterfaceSubClass */
  AUDIO_PROTOCOL_UNDEFINED,
/* bInterfaceProtocol */
  0x00,                         /* iInterface */
/* Audio Streaming Interface */

AUDIO_STREAMING_INTERFACE_DES
C_SIZE,  /* bLength */

AUDIO_INTERFACE_DESCRIPTOR_TY
PE,      /* 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_TY
PE,      /* 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_ISOCHRONOU
S,       /* bmAttributes */
  WBVAL(64),                    /*
wMaxPacketSize */
  0x01,                         /* bInterval */
  0x00,                         /* bRefresh */
  0x00,                         /*
bSynchAddress */
/* Endpoint - Audio Streaming */

AUDIO_STREAMING_ENDPOINT_DES
C_SIZE,  /* bLength */

AUDIO_ENDPOINT_DESCRIPTOR_TYP
E,       /* 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,
};
```

M. 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 "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbaudio.h"
#include "kbd.h"
#include "GLCD.h"
#include "music_waves.c"

#define __FI        1                   /* Font
index 16x24               */


//Flag to allow user to exit function
int loop_audio;

extern  void SystemClockUpdate(void);
extern uint32_t SystemFrequency;
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 */


/*
 * Get Potentiometer 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)
{
           int joystick_val;
  long  val;
  uint32_t cnt;
           loop_audio = 1;
           while(loop_audio){
                    joystick_val = get_button();
                    if(joystick_val ==
KBD_SELECT) {

NVIC_DisableIRQ(TIMER0_IRQn);

NVIC_DisableIRQ(USB_IRQn);

USB_Connect(FALSE);
                                        USB_Reset();
                                        loop_audio =
0;
                                        return;
                    }
                    else {
                                        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 */
                                loop_audio =
0;
                                return;
                            }
                        }
}


/*-------------------------------------------------
-------------------------
  MP3 Player - Play audio, if up pressed, exit
function via TIMER0_IRQn
```

```
*-------------------------------------------------
-----------------------*/
int mp3_player (void)
{
        volatile uint32_t pclkdiv, pclk;
        loop_audio = 1;
  /* SystemClockUpdate() updates the
SystemFrequency variable */
  SystemClockUpdate();

  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
SystemFrequency. */
  /* Bit 2~3 is for TIMER0 */
  pclkdiv = (LPC_SC->PCLKSEL0 >> 2) &
0x03;
  switch ( pclkdiv )
  {
        case 0x00:
        default:
          pclk = SystemFrequency/4;
        break;
        case 0x01:
          pclk = SystemFrequency;
        break;
        case 0x02:
          pclk = SystemFrequency/2;
        break;
        case 0x03:
          pclk = SystemFrequency/8;
        break;

        }

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

        GLCD_Bitmap (0, 70, 320, 92,
(unsigned char
*)MUSIC_WAVES_pixel_data);

        GLCD_DisplayString(0, 0, __FI, "
USE POTENTIOMETER ");

        GLCD_DisplayString(1, 0, __FI, "
TO ADJUST VOLUME  ");

        GLCD_DisplayString(8, 0, __FI, "
NOW PLAYING AUDIO ");
        GLCD_DisplayString(9, 0, __FI, "
SELECT - MAIN MENU ");

  NVIC_EnableIRQ(TIMER0_IRQn);
  USB_Init();                           /* USB
Initialization */
        USB_Connect(TRUE);
/* USB Connect */
```

```
    return 0;
}
```

N. 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(E
PNum)), 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(E
PNum)),
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(E
PNum)), 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(E
PNum)), 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(E
PNum)), 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_E
OT);
        }
      } 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_N
DR);
        }
      } 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_E
RR);
         }
       } 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;
}
```

O. 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) {
}
```