Development of an MP3 player using an MP3 hardware decoder

Aleksandar L. Zoranović, Goran M. Stojanović and Veljko D. Malbaša
Department of Electronics, Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia
E-mail: sgoran@uns.ac.rs

Abstract This paper presents an implementation of a digital audio player with the MP3 hardware decoder VS1033 as a result of a project-based course. The aim of this course is to motivate students to integrate knowledge from hardware and software in the field of microcomputer electronics and to apply them in order to create a functional device. This device uses a very low consumption MSP430F2417 microcontroller, which has built-in peripherals for serial communication with decoder, MMC/SDC memory card and dot matrix display with a resolution of 128×64 dots. It is able to play MP3, WMA and MIDI files and has voice recording capability.

Keywords electronics teaching; hardware decoder; microcontroller; MMC/SDC card; MP3 player

The fourth-year project-based course Microcomputer electronics, presented in this paper, aims at letting the students apply knowledge obtained in previous courses and exposing them to real-life engineering work and problems. For a successful education in electronics, it is crucial that students have opportunities to perform practical activities and to 'feel like engineers'. Therefore, this project-based course is designed to encourage students to create both the hardware and the software part of a functional device. In this paper we describe a project course based on the example of an MP3 player using an MP3 hardware decoder.

Recently, there has been a big increase in the popularity of devices capable of reproducing compressed audio formats. MPEG-1/2 Audio layer-3 (MP3) is a widespread format for playback of high-quality compressed audio for portable devices such as audio players and mobile phones. The attractiveness of the MP3 audio format is mostly due to its compactness compared to standard audio formats. Also, development of internet, on-line music services and file sharing clients has had a large influence on its popularity, with associated legal issues of copyright protection.[1] Because compressed audio decoding is processor demanding, in the past personal computers were mostly used in the decoding process. However, today, specialised microprocessors for decoding and reproduction of digital audio formats can be found in almost every home multimedia device and especially in portable devices. Sound compression algorithms (MP3, WMA, AAC, OGG and similar) are based on the statistical features of music and human hearing: they discard all non-audible sound details. The MP3 audio algorithm applies a psycho-acoustic model to a hybrid subband/transform coding scheme. In this way, less data is used for storing sound in digital format. In this process some parts of the original signal are inevitably lost in the decoding process, but an important aim is to make this loss almost imperceptible to a listener. By using a microcomputer system, decoding can be done in software

or in hardware. Software decoding is usually performed on a general-purpose 32-bit RISC processor executing DSP algorithms or on a DSP processor.[2] Hardware implementation usually implies usage of specially designed ASIC or FPGA chips.[3–5] Moreover, to consider the demand for a wireless connection to a portable MP3 player, an integrated design of an MP3 player with a Bluetooth hands-free set has been presented.[6]

In this work we present the use of a specialised microprocessor for compressed audio decoding containing communication interfaces for a simple glue-less connection to a microcontroller, as the best way for students to learn principles of hardware and software co-design.

Microcomputer system

Configuration of the microcontroller system including basic components is shown in Fig. 1. The control unit is the MSP430 family microcontroller from Texas Instruments.[7] It is connected with peripheral units using serial communication interfaces and synchronises their work. An SD/MMC card is used as a memory medium. The VS1033 is a single-chip hardware MP3 / WMA / AAC / MIDI decoder and ADPCM (adaptive differential pulse-code modulation) encoder, containing DSP core, memory, serial control and input data interfaces, stereo DAC, mono ADC and earphone amplifier. This device contains a dot matrix monochrome LCD with 128×64 dots resolution and a built-in widely used KS0108 display controller.

The MSP430F2417 microcontroller presents a new generation of a very low consumption microcontrollers, incorporating the following elements: 16-bit RISC processor, memory (92 kB + 256 B Flash, 8 KB RAM), peripheral units, flexible clock system. It includes two serial communication peripheral units supporting UART, SPI, I2C and IrDA interfaces.

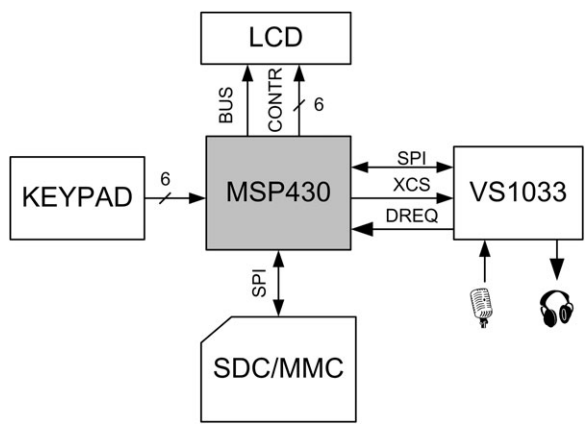

Fig. 1 *Block diagram of developed MP3 player.*

This microcontroller is a good solution for this purpose because it contains enough RAM for data buffering and has all required communication interfaces realised as hardware modules. Processing speed is not sufficient for software implementation of the decoding algorithm, which is the main reason for use of a separate hardware decoder. This is a good illustration that during the project, students need to have awareness of many important hardware design issues such as trade-offs between area, speed and power consumption. The MP3 hardware decoder VS1033 uses the SPI communication protocol to communicate with the main microcontroller working as a master.[8] The DSP core processes input data. After decoding, data proceeds through sound enhancement modules and volume control to an 18-bit sigma-delta DAC. Besides standard SPI (serial peripheral interface) protocol lines two additional lines are used – DREQ (Data Request) and XCS (Function Select). SPI is used for both VS1033's SDI (Serial Data Interface) and SCI (Serial Control Interface). The SCI interface enables writing and reading internal registers, enabling decoder configuration and reading important data during reproduction (such as reproduction time). The SDI interface is meant for transferring compressed audio data (a block of 32 bytes of data can be sent to the decoder). Besides reproduction, the chip can be used in recording mode, using microphone or line input and IMA ADPCM compression algorithm. The dot matrix LCD display includes two built-in KS0108 display controllers, each of them controlling one 64×64 pixel segment.[9] The display incorporates 1024 bytes of internal RAM, holding each pixel state. Multiplexed output and picture refreshing are carried by included KS0108 controllers. There are two modes of operation – control and data mode.

The communication interface is parallel with one 8-bit data bus and six control lines. A great advantage of the MSP430 microcontroller is that all the interfaces are realised in hardware, therefore eliminating the need for the user to take care of each bit transmission and clock generation. During initialisation, the user has to set the clock frequency and important interface parameters (bit order, idle state and active clock edge). Data transfer is initiated when data is moved to the transmit buffer and completed when data is received in the receive data buffer. Each of these events can generate system interrupt. The complete electrical scheme of the device is shown in Fig. 2. An illustration of the MP3 player in function can be seen in Fig. 3.

Software

The complete software is written in C programming language. It has hierarchical organization and includes three software layers (a driver layer, a middle layer and an application layer, Fig. 4.). The driver layer includes display driver module, VS1033 driver, MMC/SDC driver and keypad driver. The middle layer incorporates three modules: GUI – graphical user interface implementation, PLAYER – common player functions (opening file for reproduction, reproduction, next and previous song, directory and file list generation in File Browser mode) and FS – file system module. The application layer realises finite state machine and uses functions provided by the middle software layer.

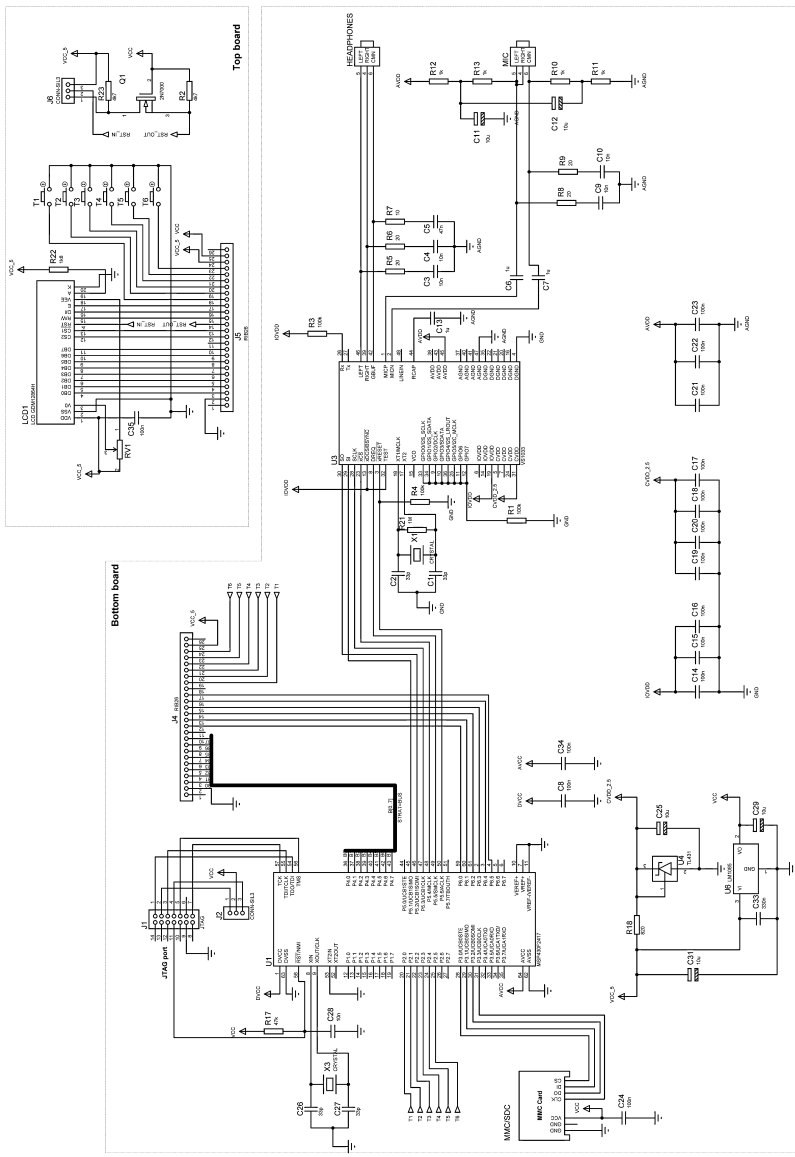

Fig. 2 *The complete electrical scheme of developed MP3 player.*

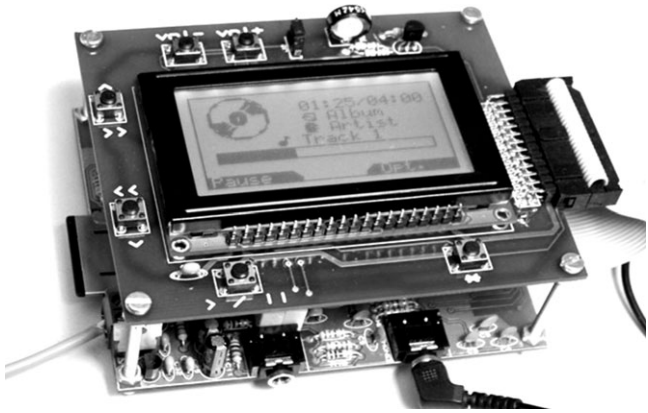

Fig. 3 *MP3 player in function.*

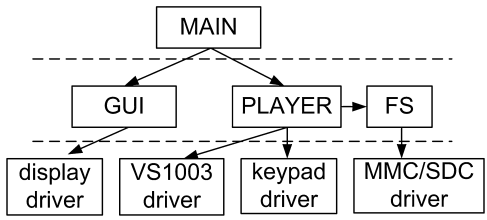

Fig. 4 *Software hierarchical organisation.*

MMC/SDC card communication protocol

MMC (multimedia card) or SDC (secure digital card) memory cards are used as a storage medium. These cards have a flash memory block and built-in microcontroller controlling read and write operations. The data is transferred between memory card and host controller in units of 512 bytes per block in default, and from the point of view of application programs, it can be seen as a generic hard disk drive.

The currently defined file system is FAT12/16, whereas FAT32 is defined for high capacity cards (>= 4 GB). These cards support two communication interfaces: SD Card bus and SPI bus. SPI bus has been used, due to its simplicity and ease of connection to the microcontroller and the fact that it is mostly used in low cost embedded applications. For MP3 file reproduction, great transfer speed is not needed and SPI bus achieves necessary speed. SD cards typically operate at 3.3 V (the same as the microcontroller) therefore voltage level conversion is not needed. In SPI mode, command frames, usually called tokens, are sent to the card.[10] When a command frame is transmitted to the card, a response to the command will be sent back to the host. There are three command response formats, R1, R2 and R3, depending on each

command. A byte of response R1 is returned for most commands, as can be seen in Fig. 5. Because data transfer is driven by serial clock generated by the host, the host must continue to read bytes until it receives any valid response. The CS signal must be held low during a transaction (command, response and data transfer if it exists).

The CRC field is optional in SPI mode, but is required as a bit field to compose a command frame. The command frame structure is shown in Fig. 6. Commands are sent to the card in six-byte format including specific command code, 32-bit argument and CRC. After power on reset, MMC/SDC enters its native operating mode. To put it in SPI mode, the host must set DI and CS high and apply more than 78 clock pulses and the card will be ready to accept commands. The most important commands are listed in Table 1.

First, it is necessary to perform a card software reset sending command 0. Afterwards, the host sends an initialisation command which differs in SDC and MMC. MMC is initialised by sending command 1. Sending command 55 (application command prefix) followed by command 41 resets the SD card. After the initialization process is done, the card is ready for data transfer. Single or multiple data blocks can be sent/received after command response. The data block is transferred as a data packet that consists of start token, data block and CRC. Figure 7 shows timing diagrams for single block read whereas Fig. 8 illustrates diagrams for single block write.

File system

Files transferred to the memory card from a PC can be read by the device because it has FAT file system implementation. The typical file system module has the

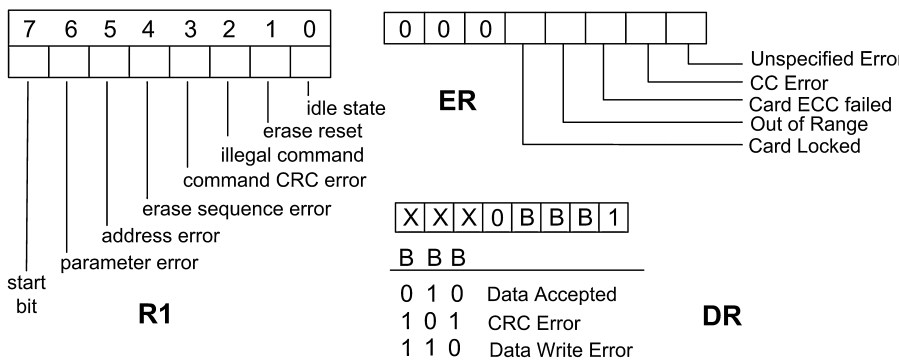

Fig. 5 *R1 response, ER (error) and DR (data response) tokens.*

BYTE 1			BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	
0	1	command	argument	argument	argument	argument	CRC	1

Fig. 6 *SDC/MMC SPI command format.*

TABLE 1 *Selected memory card commands in SPI mode*

Command	Mnemonic	Argument	Response	Description
0 (0x00)	GO_IDLE_STATE	< none >	R1	Software reset.
9 (0x09)	SEND_CSD	< none >	R1	Sending card-specific data.
10 (0x0A)	SEND_CID	< none >	R1	Sending card identification.
12 (0x0B)	STOP_TRANSMISSION	< none >	R1	Stop transmission.
17 (0x11)	READ_SINGLE_BLOCK	address	R1	Reading a block at byte address.
18 (0x12)	READ_MULTIPLE_BLOCK	address	R1	Reading multiple blocks at address.
24 (0x18)	WRITE_BLOCK	address	R1	Writing a block at byte address.
25 (0x19)	WRITE_MULTIPLE_BLOCK	address	R1	Writing multiple blocks at address.
55 (0x37)	APP_CMD	< none >	R1	Prefix for application command.
59 (0x3B)	CRC_ON_OFF	bit 0 only	R1	Argument sets CRC on or off.
41 (0x29)	SEND_OP_COND	< none >	R1	Start card initialisation.

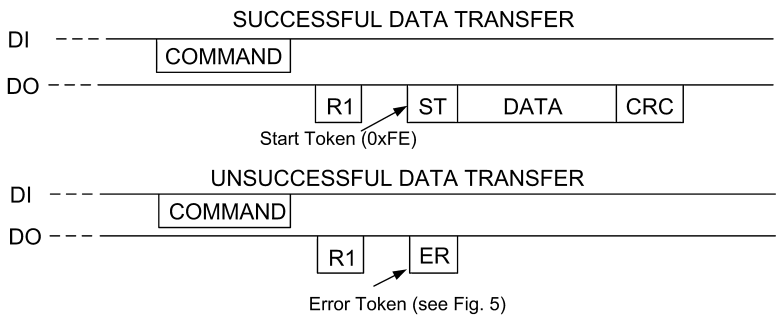

Fig. 7 *Timing diagrams for successful and unsuccessful transfer from the card.*

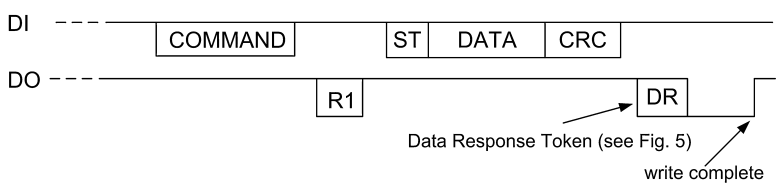

Fig. 8 *Timing diagrams for single block write.*

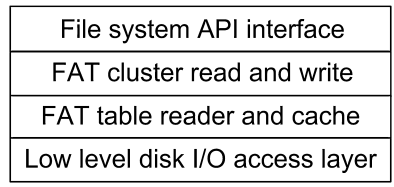

Fig. 9 *Typical file system module organisation.*

structure depicted in Fig. 9. The top level functions create a uniform application interface regardless of the memory medium being used. Medium level functions allow cluster access, FAT (file allocation table) operations and caching for better performance. The lowest level functions provide interface to storage medium (driver). The device uses a generic FAT file system library called Tiny FatFs.[11] This module is free software, written in ANSI C, making it portable and is optimised for 8-bit and 16-bit microcontrollers. It supports FAT12, FAT16 and FAT32, single drive, 8.3 format file name (long file names are not supported) and has very low memory consumption (less than 1 kB of RAM). Since the FatFs module is completely separated from the disk I/O layer, various storage media are supported (hard disks, CDs, memory cards). Low level medium I/O functions are not included in the module and must be provided by the user (functions for drive initialization, reading and writing sectors). These functions rely on a MMC/SDC driver module written by a user knowing previously noted usage of MMC/SDC in SPI mode. The most important functions provided by the file system module are: f_mount – mounts and initializes work area, f_readdir – reads directory entries in sequence, f_opendir – opens a directory, f_open – opens or creates a file, f_read – read data from file, f_write – writes data to file, f_close – closes a file, f_seek – moves file R/W pointer.

Graphical User Interface

The keen competition in the market for MP3 players has resulted in the need for innovative GUI design suitable for a variety of customers.[12] The graphical user interface (GUI) module enables an intuitive interface to interact with the user. The developed GUI, within this project, is based on the fact that each system state has its own interface which is defined by calling specific dispatcher function with specified parameters. Dispatcher function pseudo code for the GUI in File Browser state is shown in Fig. 10, whereas the GUI screen is illustrated in Fig. 11.

Reproduction

Blocks of 512 bytes of data are transferred from the memory card to a buffer during reproduction, as can be seen in Fig. 12. As decoder accepts blocks of 32 bytes of data to be decoded, the microcontroller sends blocks of this size from the 512 byte buffer. Data is read from the card in blocks of 512 bytes because that gives the best performance (reading single byte from the flash memory card is significantly slower, because the NAND flash memory array used in cards is optimised for fast reading of blocks of data). The GUI screen during reproduction is shown in Fig. 13.

```
unsigned char GUI_Browser(uchar p)
{
  case p:
  {
     _FRAME :     show frame (background);
     _BUTTONS:  write button function;
     _ROWS:       show dir entries;
     _TITLE:      show title;
     _FONT:       set font style;
  }
}
```

Fig. 10 *GUI dispatcher function pseudo code.*

Fig. 11 *File browser state GUI screen.*

Several concurrent tasks are performed during reproduction – data transfer, updating reproduction information on display (progress bar, song duration) – thus efficient processor time management is needed. The microcontroller's Timer A is used for the time sharing mechanism. This timer is also applied for the generation of delay intervals used in serial communication with peripherals.

Main program

The developed device represents a finite state machine (FSM), thus the main program flow is an infinite iteration in which the current state is checked and depending on the current state and switch condition, the transition to a new state is generated (a form of Mealy FSM). The transition diagram is presented in Fig. 14, whereas the program flow, which starts with initialisation, is shown in Fig. 15. Each state has

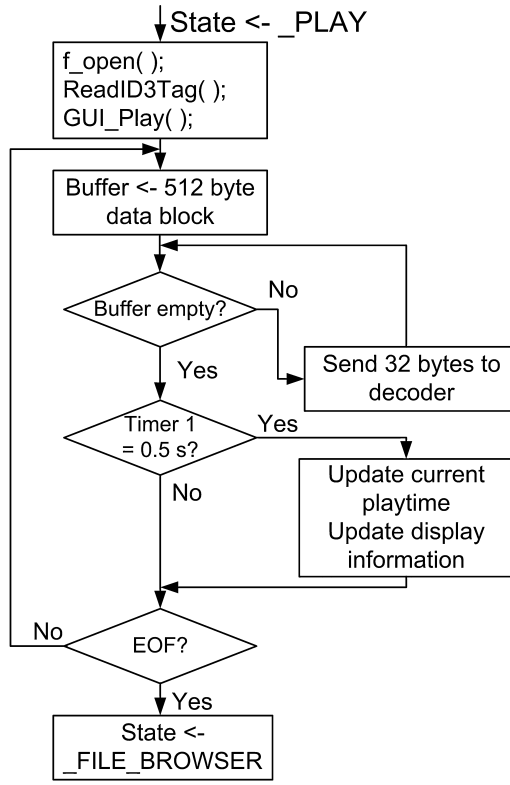

Fig. 12 *Reproduction steps.*

Fig. 13 *Play state GUI screen.*

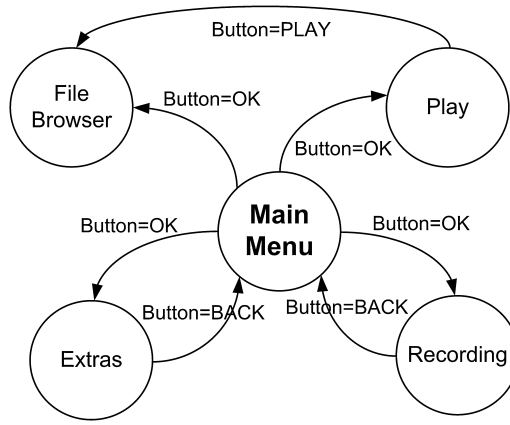

Fig. 14 *State transition diagram.*

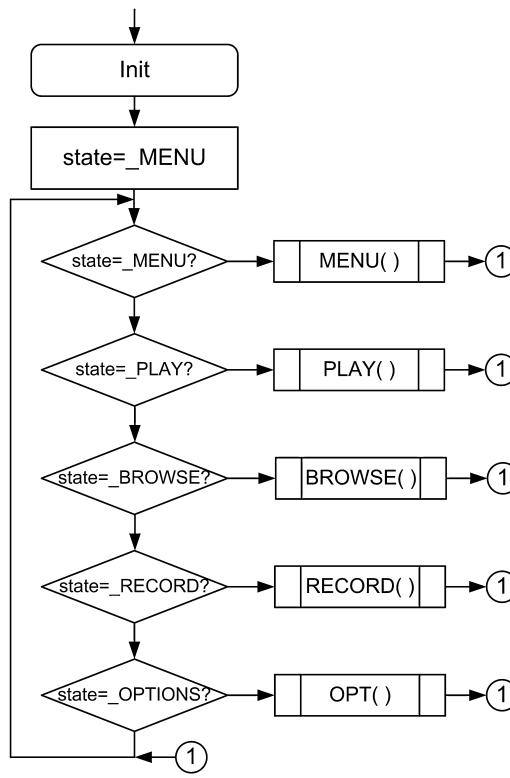

Fig. 15 *The main program flow.*

descriptive structure carrying system parameters in that state which are being used in middle layer functions.

Specific learning outcomes

The project described in this paper has been designed to give an excellent opportunity for students to learn about microcomputer electronics in a 'real world' scenario. Upon successful completion of this course and realisation of the described (or similar) project, the student should be able to:

- Identify the functional units of a computer system (i.e., the processor, peripherals, memory, etc.), their roles and to combine them in a device;
- Demonstrate the ability to access a variety of software applications using the graphical user interface (GUI) features and to understand the importance of the GUI for users (and customers);
- Write medium-level code to make a functional microcomputer;
- Demonstrate the ability to use File System, different audio formats and compression algorithms;
- Describe how all people are being affected by microcomputer electronics in their daily lives with interesting and useful application-oriented devices.

Appealing and practically oriented projects have a central role in engineering, so the presented example represents a typical project-based learning model as an educational tool. This model actively involves students in the learning process and they have opportunities to make own decisions in different project phases such as design, selection of electronic components, communication protocols, combining hardware parts, testing, etc.). This approach results in increased motivation, satisfaction and confidence of the students. To evaluate students' satisfaction and their opinions about this projects-based course, an evaluation form was prepared. The evaluation form was completed by 23 students in spring 2007 and 29 students in the spring semester of 2008. The feedback from the students was quite positive, and their responses are presented in Table 2.

Almost 60% of the enrolled students strongly agreed that they gained valuable practical hands-on skills through realised projects. More than 45% of the students responded SA (strongly agree) and around 40% A (agree) that they enjoyed carrying out the practical work and being given opportunities to demonstrate individual initiative and creativity. Students had also a chance to express their own comments and suggestions in free form. Among the comments received were:

- '*The projects-oriented course has been interesting*';
- '*This project has increased my affinity to consumer electronics in general*';
- '*I gave a real-world experience in electronics and I will recommend this course to other students*'.

Through these projects, students also develop the following skills: time management, teamwork, organisation, innovative approach, critical and creative thinking,

TABLE 2 *Student responses associated with the Microcomputer electronics course (SA – Strongly Agree, A – Agree, D – Disagree, SD – Strongly Disagree)*

No.	Questions	SA	A	D	SD
1	I gained practical skills through this project-oriented course				
	Spring 2007 (23)	15	5	3	0
	Spring 2008 (29)	14	10	5	0
2	The project-based hands-on experience helped me learn hardware and software co-design				
	Spring 2007 (23)	13	7	3	0
	Spring 2008 (29)	11	12	5	1
3	I enjoyed carrying out the practical work				
	Spring 2007 (23)	15	5	2	1
	Spring 2008 (29)	12	11	3	3
4	Performing the realisation of the practical work gave me the opportunity to demonstrate individual initiative and creativity				
	Spring 2007 (23)	14	5	3	1
	Spring 2008 (29)	14	10	4	1
5	I find this project-based course relevant for my future profession				
	Spring 2007 (23)	16	5	2	0
	Spring 2008 (29)	12	12	3	2
6	Overall, how would you rate this project?				
	Spring 2007 (23)	9.43			
	Spring 2008 (29)	8.97			

etc. Through this approach, students face the reality of the multidisciplinary problems of a typical consumer electronics application.

Conclusion

This paper has presented the most suitable implementation of a digital audio player for lectures on microcomputer electronics through a project-based course. The microcontroller used for the project proves to be an adequate solution for this task, because it has all the necessary interfaces for communication with peripherals and a sufficient amount of RAM. The highly integrated single chip MP3 decoder has very low power consumption, needs very few external components and includes a programmable DSP core, making it possible for the user to add custom effects and even new compression formats. Graphic LCD enables implementation of a graphical user interface. SDC or MMC are widespread and cheap types of memory cards. The FAT file system module makes the memory card compatible with PCs. Components are well synchronised, thus the gapless playback of MP3 files with a bit rate of up to 320 kbps has been achieved. The device has small power consumption through using a new generation of components and sleep modes of the MSP430 microcontroller. All components demand a 3 V working voltage, except the display which needs 5 V, therefore an external voltage source of 5 V has been used. An alternative

solution could be usage of two 1.5 V AA batteries and boost converter or 3 V compatible display. The necessity for students to pay close attention to all these aspects, during the design of one complex microcomputer system, is a very positive side of the project course. The assessment results, together with individual discussions with students, indicate that they are very satisfied with acquired practical knowledge and experience through popular project-based learning.

References

1 M. McCandless, 'The MP3 revolution', *IEEE Intelligent Systems and Their Applications*, **14** (1999), 8.
2 V. Gurkhe, 'Optimization of an MP3 decoder on the ARM processor', *Conf. Convergent Technologies for Asia-Pacific Region*, *TENCON 2003*, Bangalore, India, Oct. 2003, Vol. 4, pp. 1475–1478.
3 T. Sakamoto, M. Taruki and T. Hase, 'A fast MPEG-audio layer III algorithm for a 32-bit MCU', *IEEE Trans. Consumer Electron.*, **45** (1999), 986.
4 H. Hedberg, T. Lenart, H. Svensson, P. Nilsson and V. Owall, 'Teaching digital HW-design by implementing a complete MP3 decoder', *IEEE Int. Conf. on Microelectronic Systems Education* (MSE'03), Anaheim, CA, June 2003 (IEEE, 2003), pp. 31–32.
5 H. Hedberg, T. Lenart, and H. Svensson, 'A complete MP3 decoder on a chip', *IEEE Int. Conf. on Microelectronic Systems Education*, (MSE'05), Anaheim, CA, June 2005. (IEEE, 2005), pp. 103–104.
6 Y. W. Bai and Ch. L. Chiang, 'Design and implementation of the integration applications for a portable MP3 player with a Bluetooth hand-free/set', *IEEE Trans. Consumer Electron*., **51** (2005), 849.
7 *MSP430x2xx Family User's Guide*, Texas Instruments, 2006.
8 *VS1033 – MP3/WMA/AAC/MIDI audio codec*, datasheet, VLSI Solutions, 2007.
9 *GDM12864H LCD module datasheet*, Xiamen Ocular, 2005.
10 *SD Media Format Expands the MAXQ2000's Space for Nonvolatile Data Storage*, Maxim Application Note 3969, Dec. 2006.
11 Tiny-FatFs – FAT file system module, © ChaN (August, 2008). [Online]. Available at http://elmchan.org/fsw/ff/00index_e.html.
12 Y. W. Bai and F. E. Tsai, 'Design and implementation of a table-based GUI for MP3 players', *IEEE Trans. Consumer Electron*., **53** (2007), 554.