EL5483
Real-Time Embedded System Design
Final Project

MUSIC PLAYER

Luxiang Yin
Chung-Lin Sha

List of Contents

1. Introduction

Our goal is to make a music player with C8051F020, VS1053 (audio codec chip) and SD/micro-SD card. We use C8051F020 as the main controller to read music data from SD/micro-SD card and sent it to VS1053 for decoding and playing.

We connect two development boards together to build the whole hardware system. One is Silabs MCUniversity Starter Kit with C8051020, several buttons and LEDs on-board. The other is Sparkfun MP3 Player Shield, with VS1053 and micro-SD card slot.

We implement a layered software structure for this music player. Hardware Abstraction Layer (HAL) provides functions and other accessories to directly access hardware; FAT File System provides file operation interfaces; Event System abstract events from hardware and other activities and provide interface to give responds to these events.

2. Hardware

2.1 Hardware Connection

We connect the two broads with DuPont Cables. The hardware connection and system diagram are shown in figure 2.1-1 and figure 2.1-2.

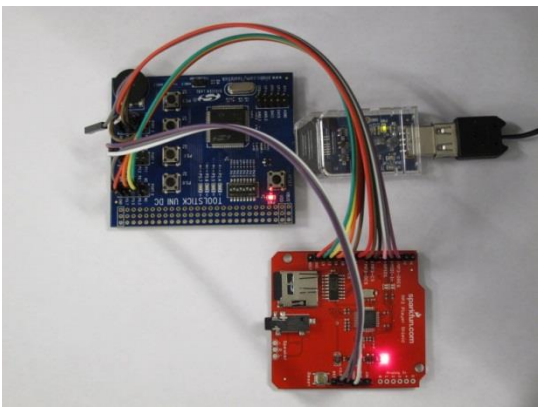


Figure 2.1-1 Hardware Connection

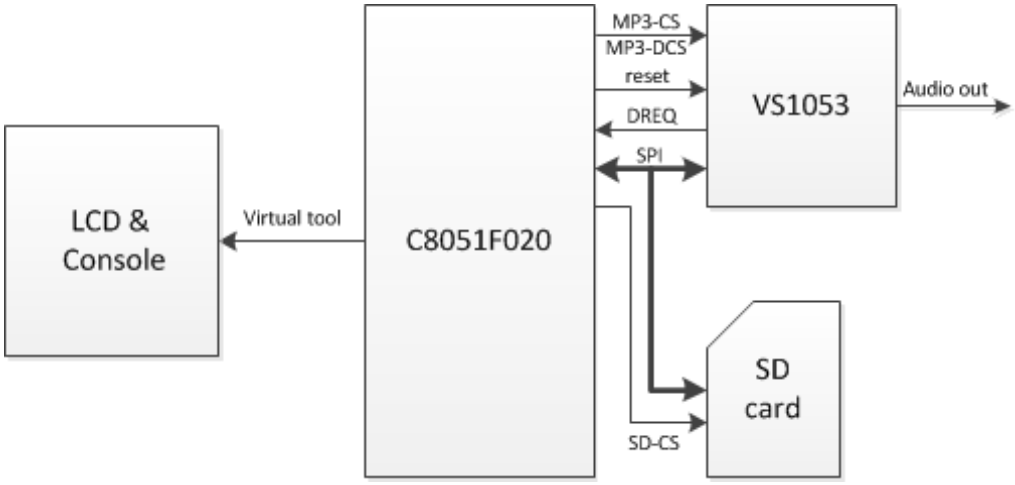


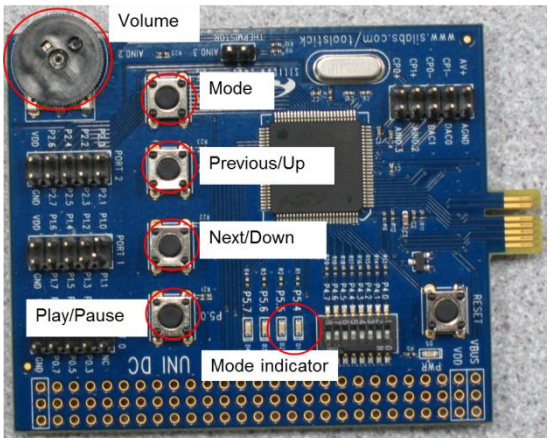
Figure 2.1-2 System Diagram

MCU communicate with VS1053 chip and SD card via SPI. MP3-CS, MP3-DCS, and SD-CS are used as Chip-select. MP3-CS is for instructions and MP3-DCS is for data. VS1053 pull down DREQ when it is ready for new instructions or data so that MCU can get to know. MCU can pull down Reset to put VS1053 into hardware reset.

2.2 User Interfaces

2.2.1 On-board Control & Indicators

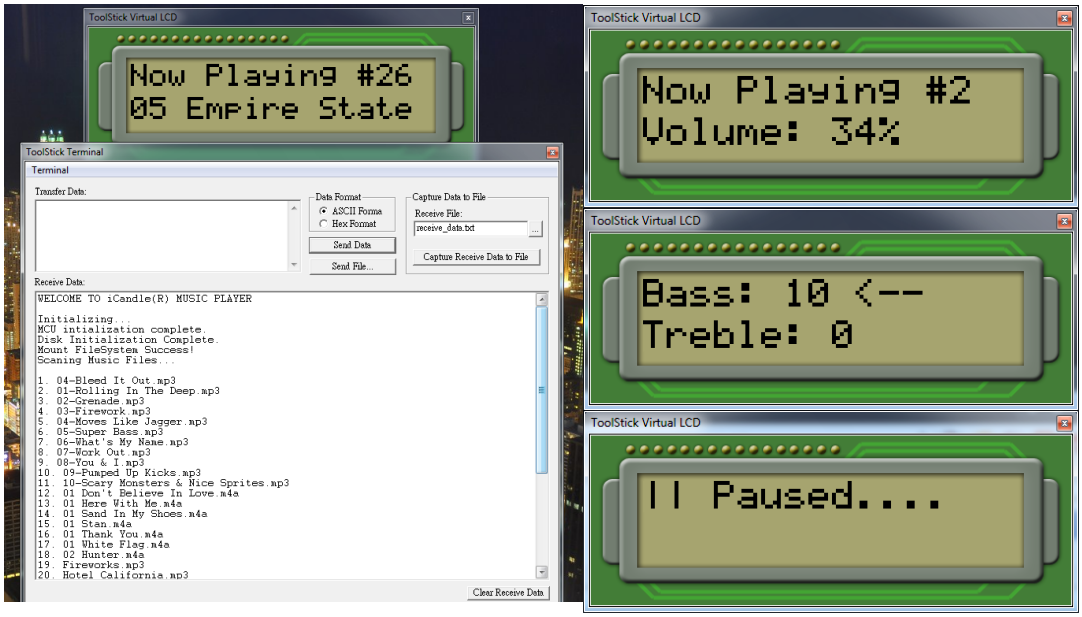
Volume is controlled by potentiometer. The button at P5.3 is used to change the mode of player. The buttons at P5.2 and P5.1 are used to switch songs and change the level of bass and treble. The button at P5.0 is used to pause or resume playing. LEDs at P5.4 and P5.5 are configured as mode indicator. The LED-Mode table is shown below ('1'=on; '0'=off).



LED	Mode
00	Song
01	Bass
10	Treble

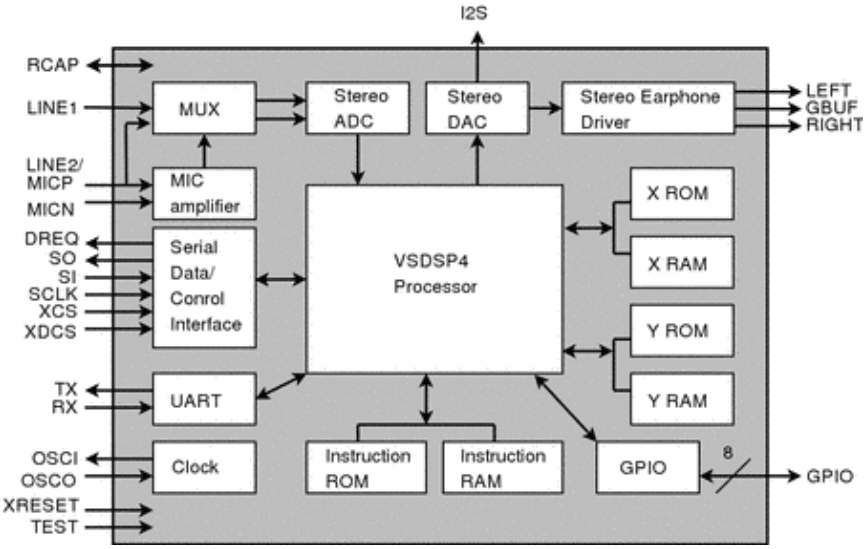
Figure 2.2-1

2.2.2 Virtual Tools (Terminal and LCD)



Virtual tools are used to show the status of the music player. The Toolstick Terminal shows all the console messages of the music player and the list of files in the SD card. Virtual LCD shows current status of the music player.

2.3 VS1053



VS1053 is an audio codec chip that supports large varieties of formats. Its control and data interface is SPI combined with several other pins. The connection pins we need is shown in Figure 2.3-1

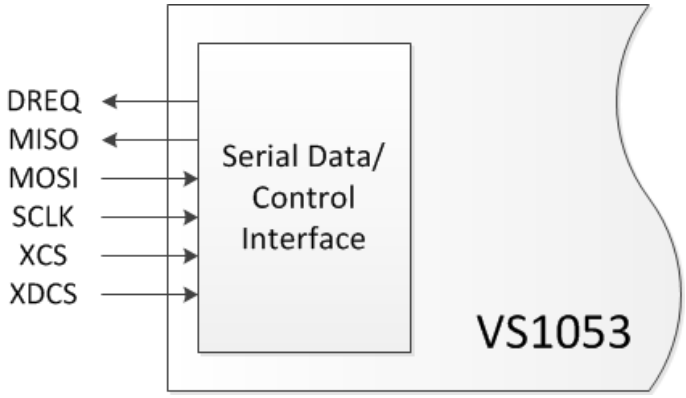


Figure 2.3-1

MISO, MOSI, and SCLK are SPI-compatible pins. XCS and XDCS are Chip-select of Control and Data. DREQ indicates whether VS1053 requires new data/instructions.

2.4 Micro SD Card

The micro SD card is configured to work in SPI mode. During initialization, the SPI bus works at a low speed with

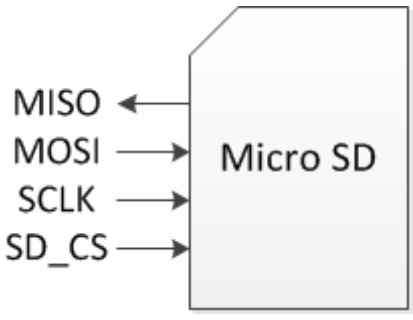


Figure 2.4-1

114KHz clock, to guarantee the highest compatibility. The clock will go up to 3.7MHz after initialization. Pin connection and descriptions are shown in Figure 2.4-1 and Figure 2.4-2.

SPI Bus

Pin	Name	I/O	Logic	Description
1	nCS	I	PP	Card Select (Neg True)
2	DI	I	PP	Data In [MOSI]
3	VSS	S	S	Ground
4	VDD	S	S	Power
5	CLK	I	PP	Clock [SCLK]
6	VSS	S	S	Ground
7	DO	O	PP	Data Out [MISO]
8	NC nIRQ	. O	. OD	NC (Memory Cards) Interrupt (SDIO Cards)
9	NC	.	.	NC

Figure 2.4-2

3. Software

3.1 Introduction

In our system we divide it into three different layers: application layer, event system and hardware layer. Hardware communicates to the event layer. Event system layer handles all the events in our system such as pause, volume change, next song… etc. Then application layer gives responds to events that event system provides.

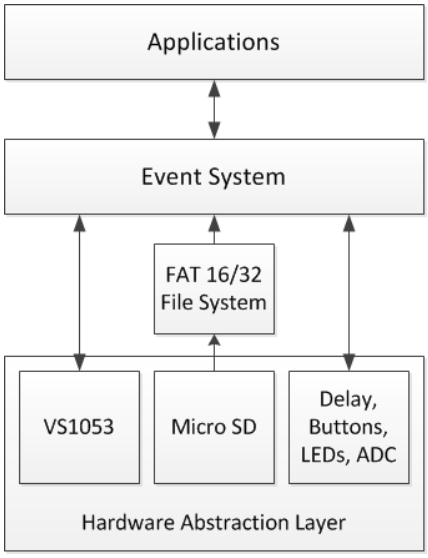


Figure 3.1-1

3.2 FAT File System

We use an open source project, FatFs, to implement our file system. It has two layers. The lowest level is Disk I/O and Disk-oriented Function layer, which directly read or write physical sectors of the storage media. File-oriented Function Layer is provided by the original project FatFs. It contains a series of functions and accessories for handling files.

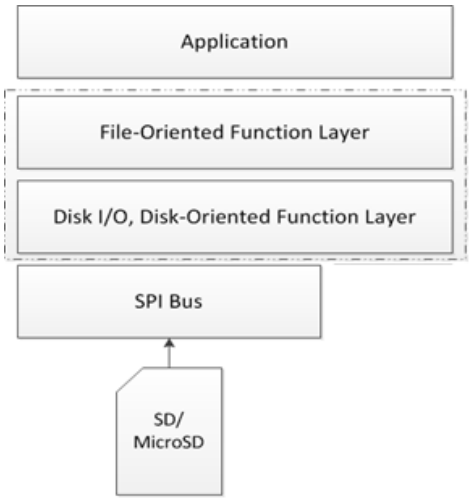


Figure 3.2-1

3.2.1 Disk I/O Layer

Disk I/O Layer consists of several basic functions that directly Read/Write Physical Sectors. Functions are shown below:

- disk_initialize() – Initialize the disk
- disk_read() – Read sectors, given start sector number and sectors count
- disk_status() – Get disk status

3.3 Event system

Event system avoids large sequence of code in a single loop. It abstracts Events from hardware activities. It also separates scanning hardware changes and responds, to avoid confusion. The main loop will become very simple if applied event system, as shown below.

```
//Main loop
while(1)
{
      //Get system event
      Player.Event = GetEvent();
      //Handle Events
      HandleEvent();
}
```

3.3.1 Event Type Definition

We define all the possible status into 11 different kinds of the events. It can help us easily manage all the situations in the music player. All events are shown below.

```
typedef enum{
      EV_NULL = 0,     //Nothing happened
      EV_DREQ,         //VS10xx requires more data
      EV_BUFEMPTY,     //Buffer is empty
      EV_NEXT,         //Next song
      EV_PREVIOUS,     //Previous song
      EV_MODE,         //Mode change
      EV_BASSTREB,     //Bass/Treble adjustment
      EV_VOLUME,       //Volume adjustment
      EV_PAUSE,        //Pause
      EV_PLAY,         //Resume playing
      EV_END                 //File end
}EVENT;
```

3.3.2 Event flow

Our music player is implemented as the event flow which shows below.

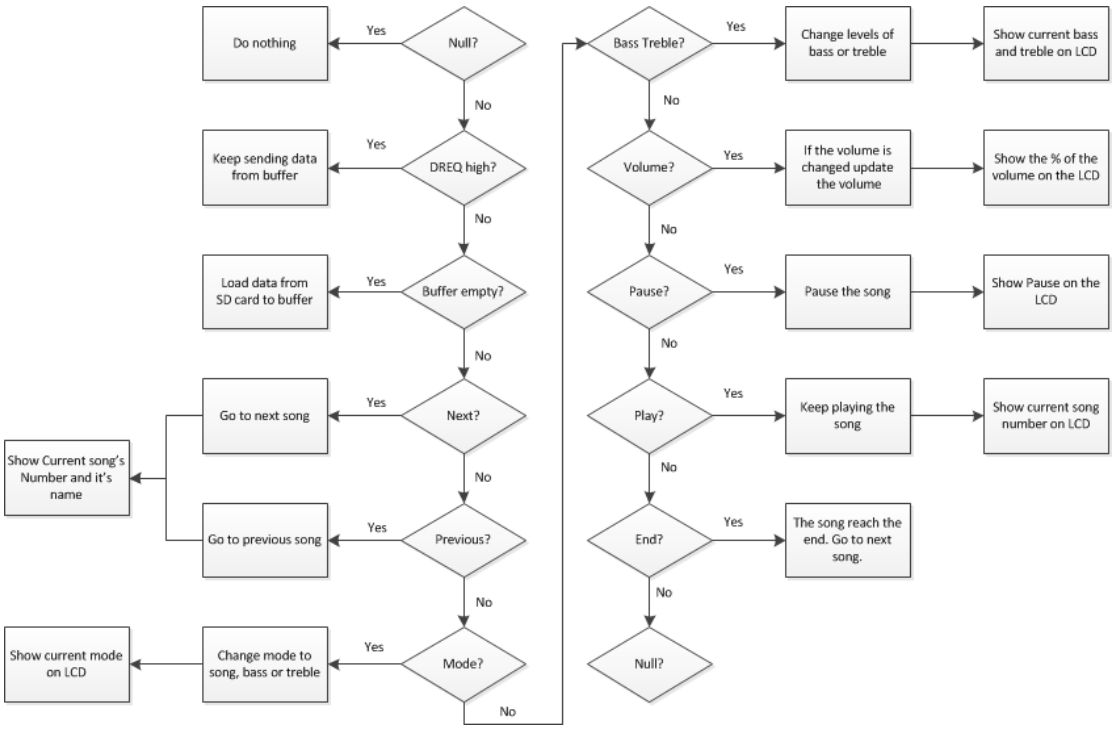


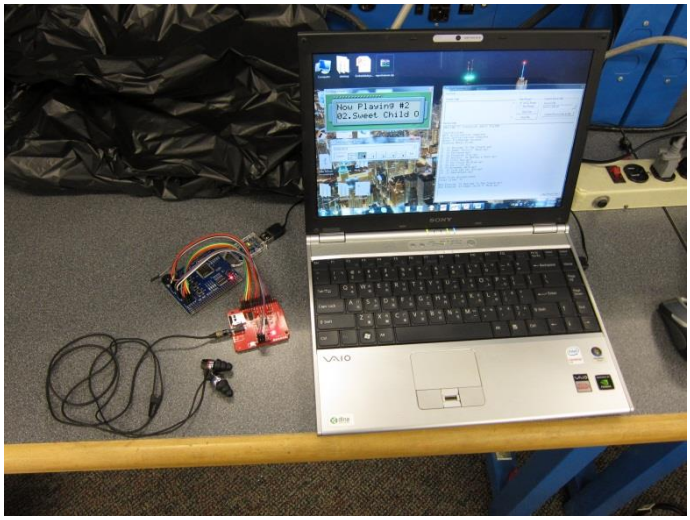
Figure 3.3-1

4. Result

In this project we implement almost all the function which a music player needs to provide. This music player below functions

- Play/Pause.
- Next/ previous song.
- Audio enhancement: bass and treble change.
- Mode change function.
- Volume change by potentiometer.
- Show mode on LED
- Show status on LCD
- Show initialization status and song list in Terminal

Our implementation is shown below.



5. Development History

- Ver. 1. Basic connection and function tests.
- Ver. 2. Read directly from sectors of a SD card but failed.
- Ver. 3. Applied open source project – FatFs as an implementation of file system.
- Ver. 4. Initial implementation of playing a single song.
- Ver. 5. Added an event control system.
- Ver. 6. Added tracks scan, next & previous tracks changing function.
- Ver. 7. Added volume control and Bass/Treble enhancement.
- Ver. 8. Added a virtual LCD as a display device.

6. Possible Further Improvements

- Add Full Control of the Whole System From Console.
- Apply FIFO Organization to Event Buffer (so that new events can be preserved while there are unhandled events pending).
- ID3 Tag Recognition.
- Apply Folder Traversal (to do full scan of music files).