Real-Time Embedded System Design
Final Project

MUSIC PLAYER

Luxiang Yin

Chung-Lin Sha

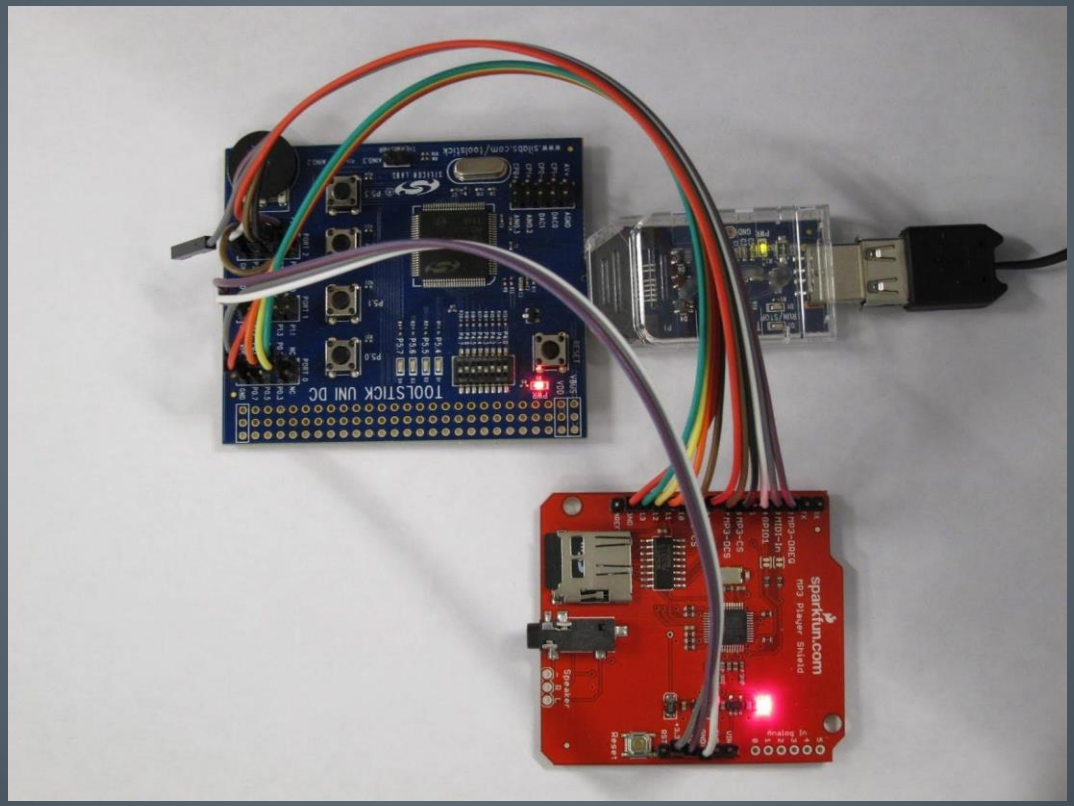
Outline

- Introduction & Overview
- Hardware
- Software
- Development History

INTRODUCTION & OVERVIEW

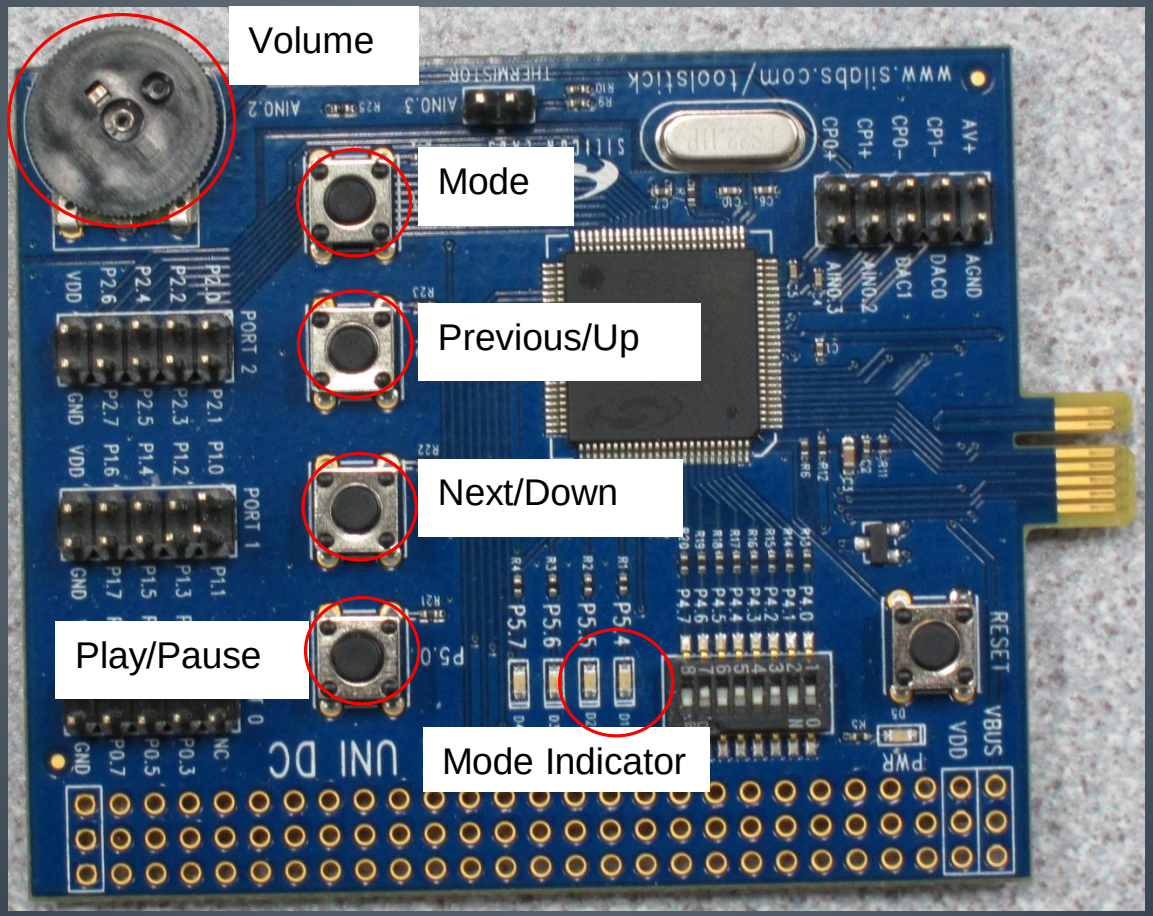
- Hardware
- User Interface
- Virtual Tools

Overview
Hardware



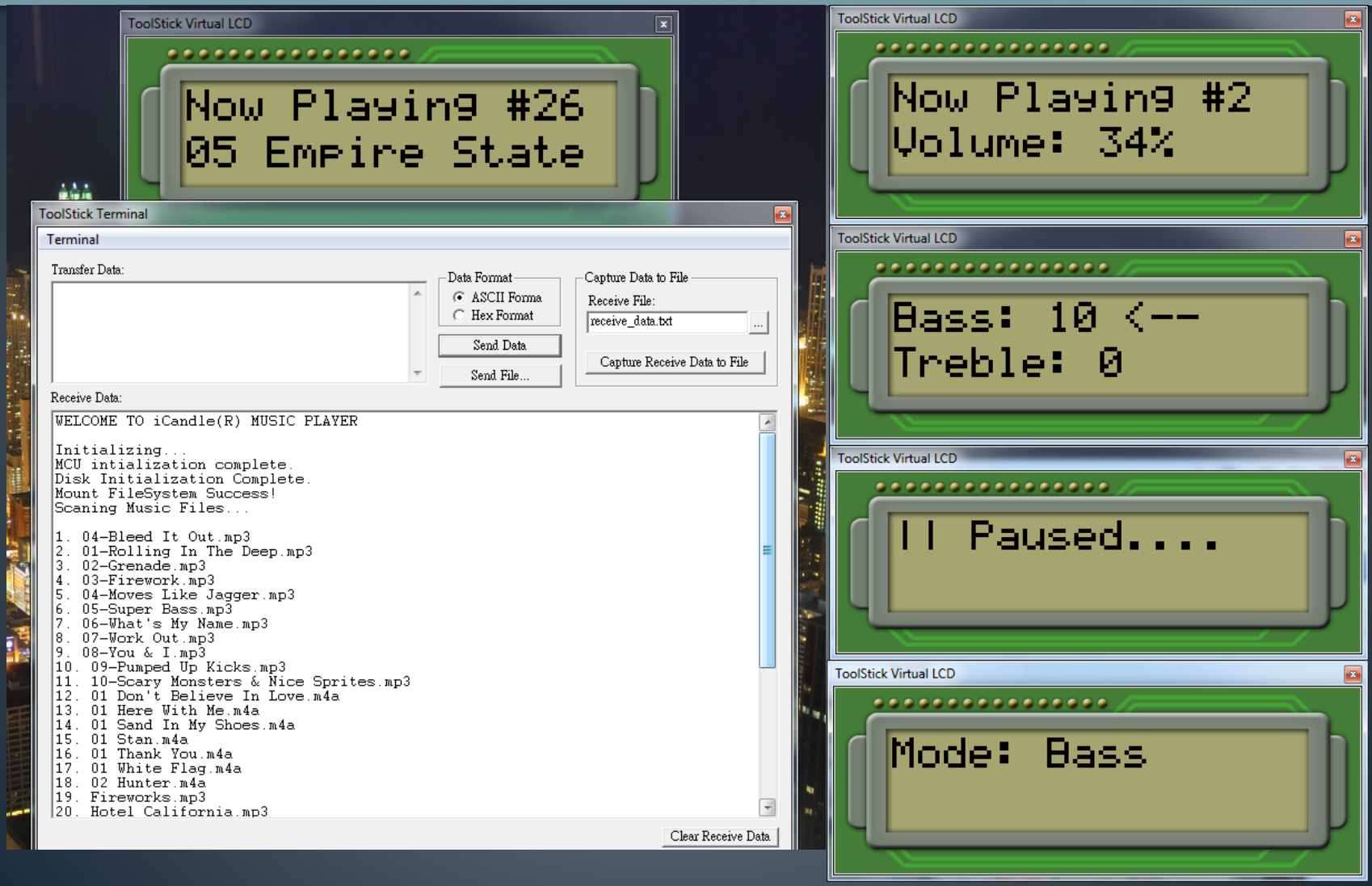
Overview

User Interface



Overview

Virtual Tools

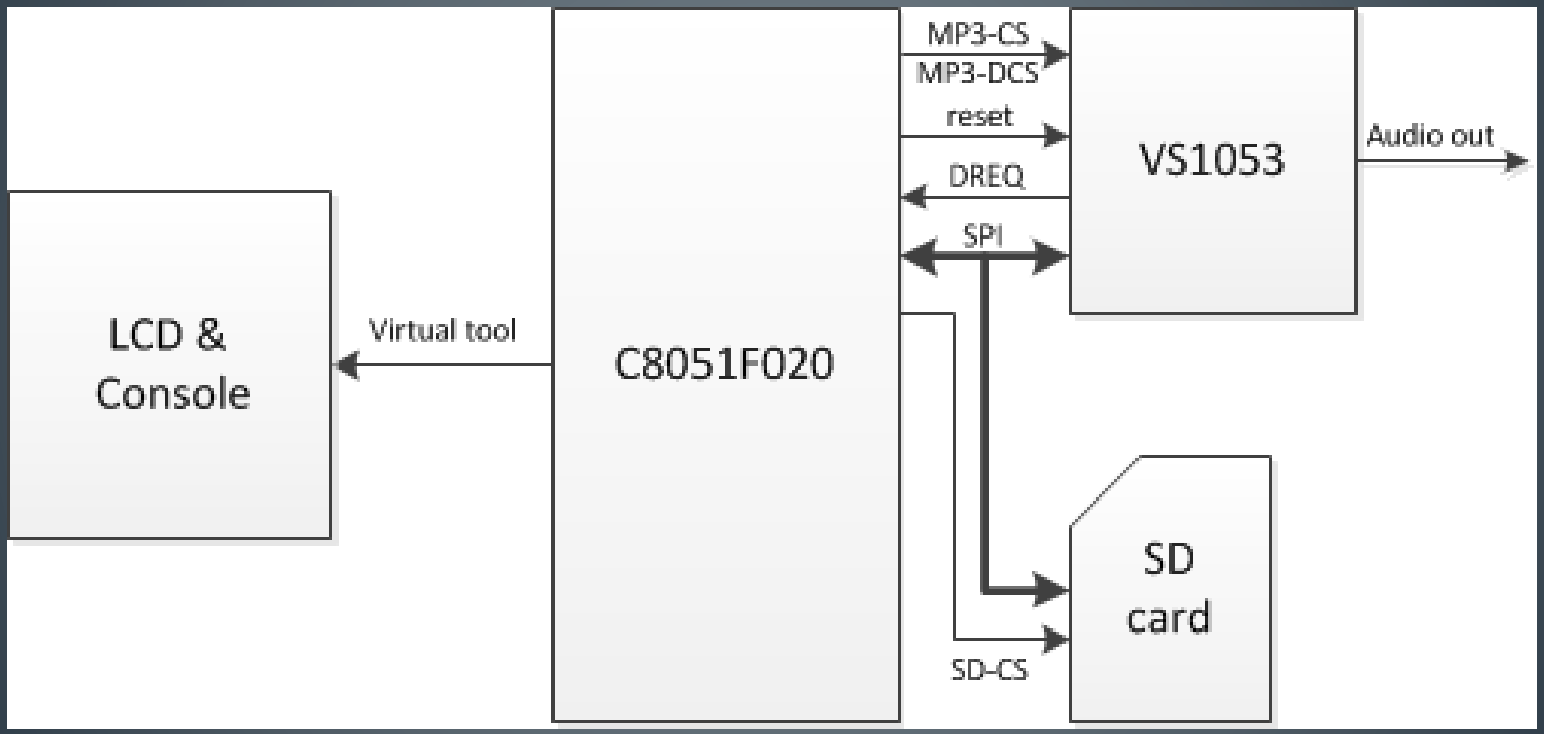


HARDWARE

- System Diagram
- VS1053
- Micro SD Card

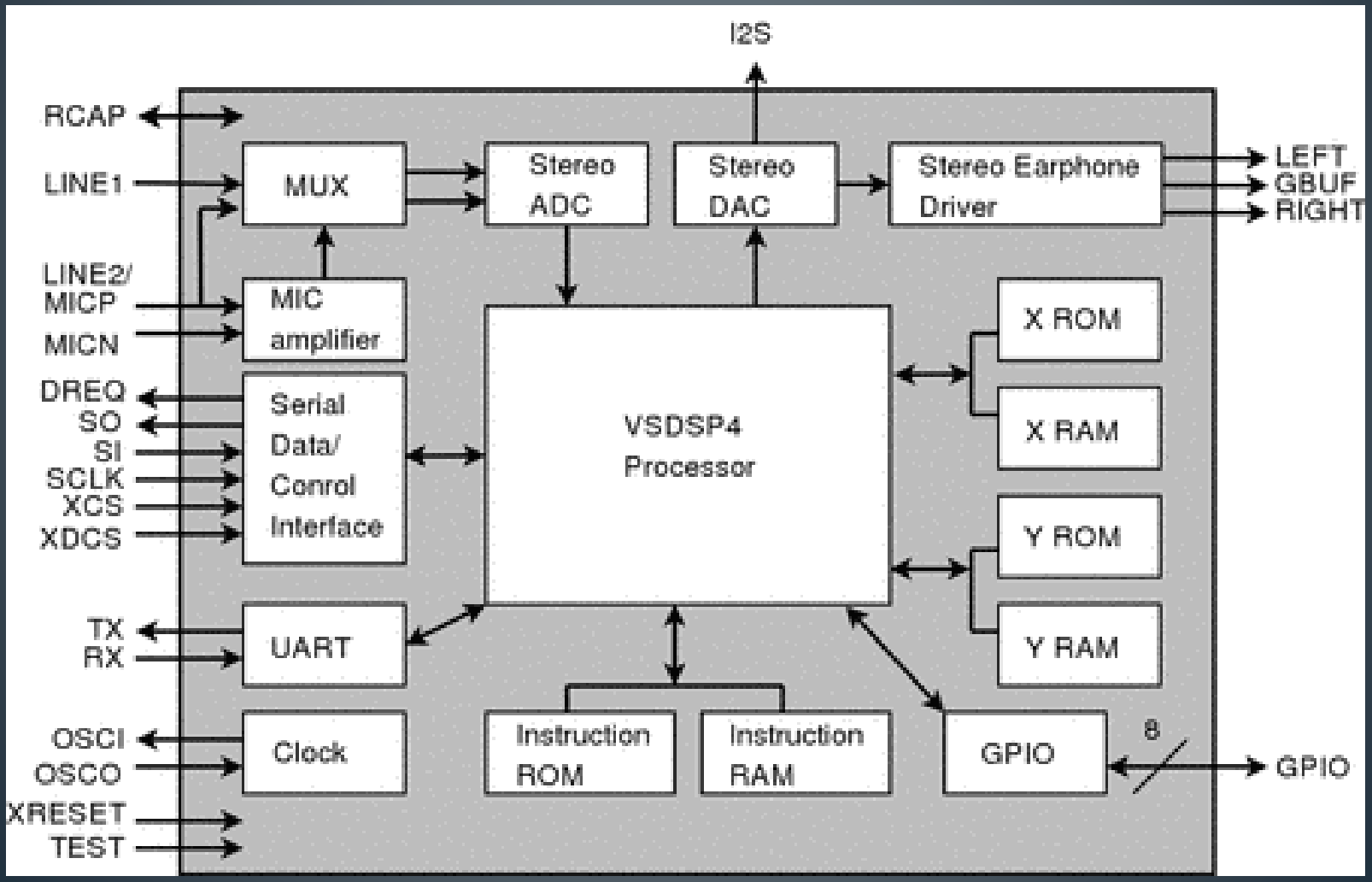
Hardware

System Diagram



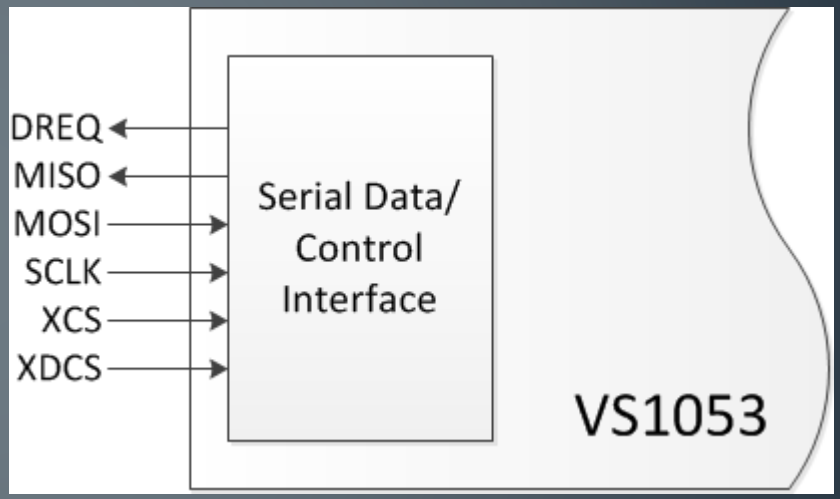
Hardware

VS1053



Hardware
VS1053

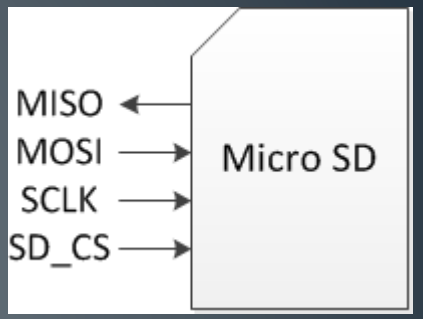
- **MISO/MOSI/SCLK**: SPI Compatible Pins
- **XCS/XDCS**: Chip-select of Data or Control
- **DREQ**: Indicate whether VS1053 requires new data/instruction



Hardware
Micro SD

- BUS: SPI Mode

- Clock Rate:
- 114KHz during initialization,
- 3.7MHz during data transmission



SPI Bus

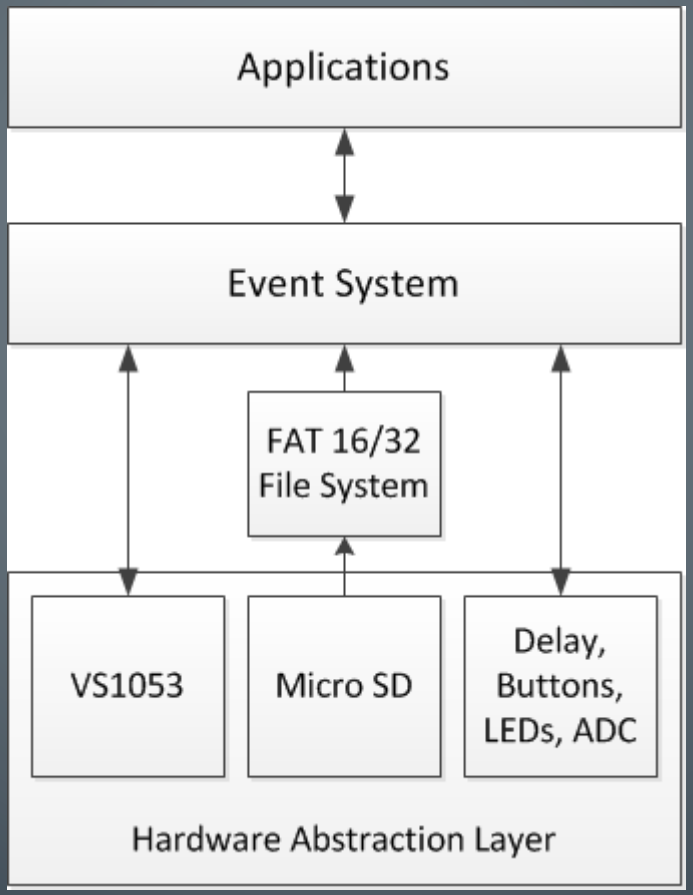
Pin	Name	I/O	Logic	Description
1	nCS	I	PP	Card Select (Neg True)
2	DI	I	PP	Data In [MOSI]
3	VSS	S	S	Ground
4	VDD	S	S	Power
5	CLK	I	PP	Clock [SCLK]
6	VSS	S	S	Ground
7	DO	O	PP	Data Out [MISO]
8	NC nIRQ	. O	. OD	NC (Memory Cards) Interrupt (SDIO Cards)
9	NC	.	.	NC

SOFTWARE

- Diagram
- FAT File System
- Event System

Software
Diagram

Layered Structure



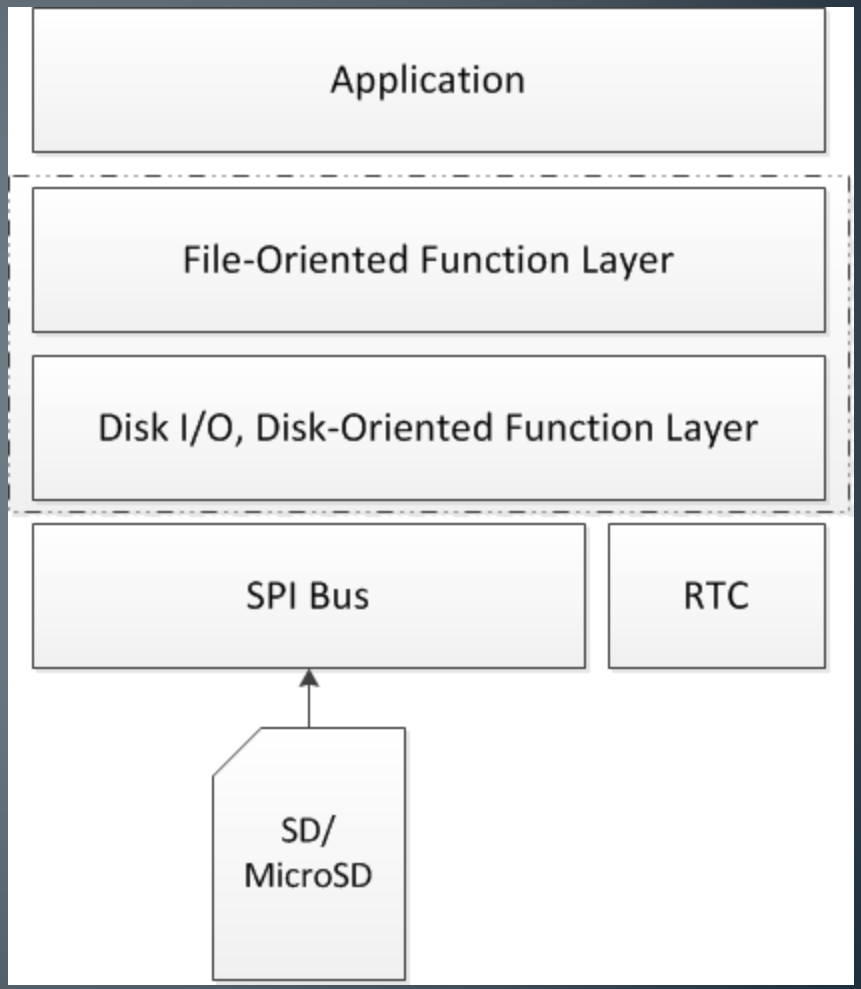
Software

FAT16/32 File System

- Open Source Project – FatFs
- Required Implementation:

  Disk I/O layer and

  SPI communication.



FAT16/32 File System

Disk I/0 Layer

- Disk I/O Layer consists of several basic functions that directly Read/Write Physical Sectors.

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

Software
Event System

The main loop is now like this which is applied an Event System.

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

Software

Event System

- Avoid large sequence of code in a single loop;
- Abstract Events from hardware activities;
- Separate scanning hardware changes and responds, to avoid confusion.

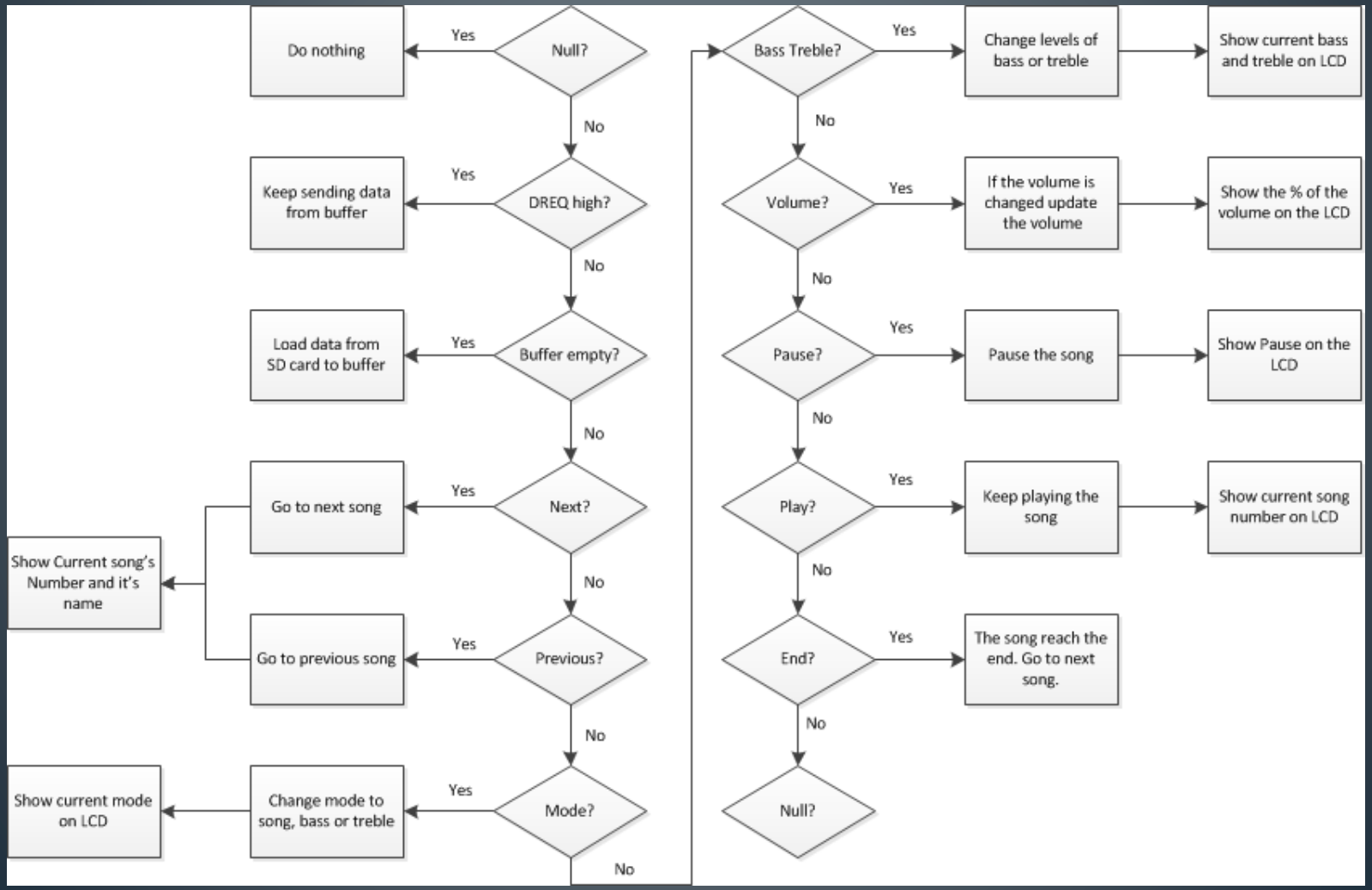
Software
Event System

- Event Type Definition

```
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 sone
    EV_MODE,            //Mode change
    EV_BASSTREB,        //Bass/Treble ajustment
    EV_VOLUME,          //Volume ajustment
    EV_PAUSE,           //Pause
    EV_PLAY,            //Resume playing
    EV_END              //File end
}EVENT;
```

Software

Event System



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 a 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.

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).

Thank You