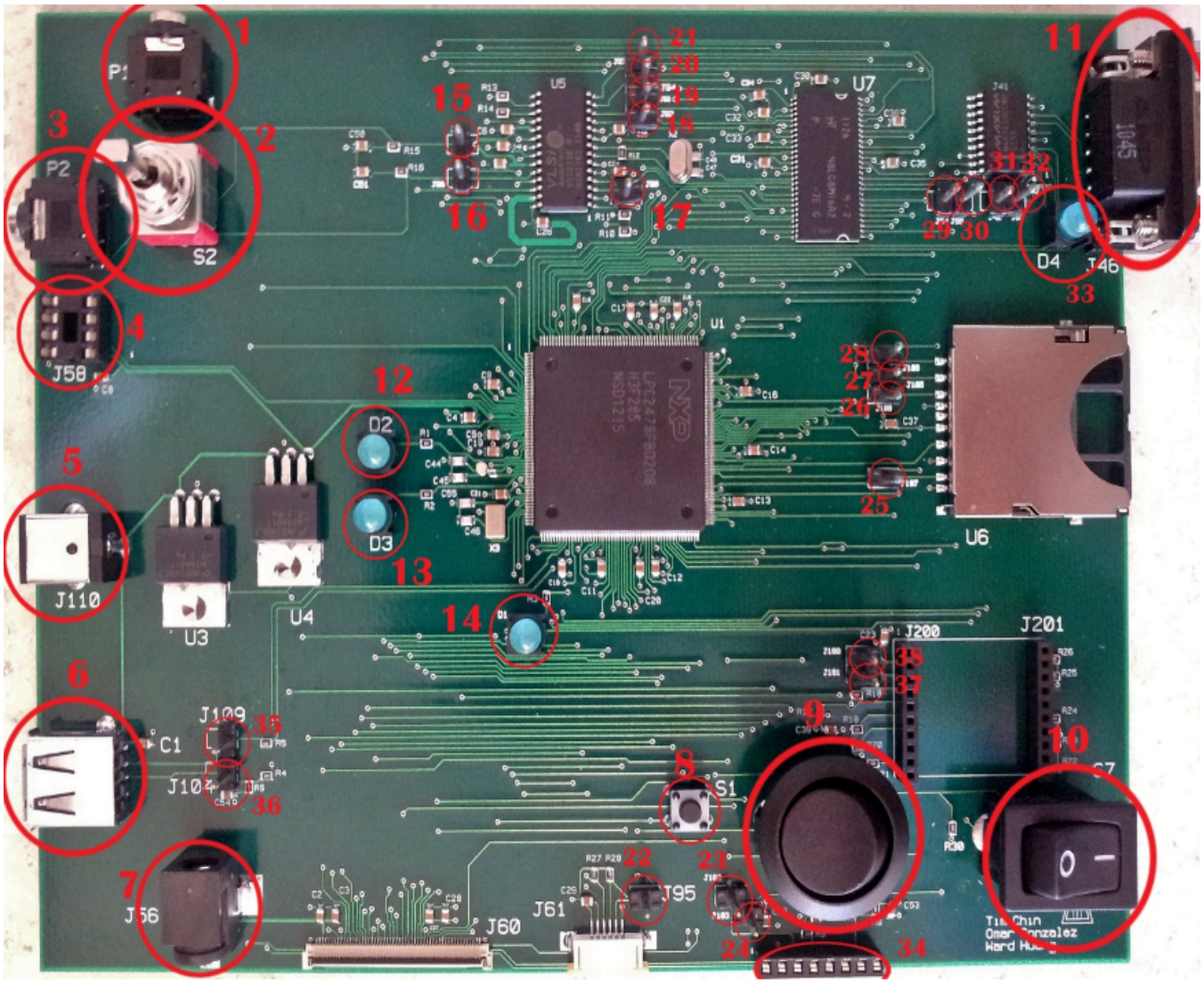
Mu.S.E. Operating Manual



Controls and Indicators

1 3.5 Audio jack line input (P1)

2 Pull-Throw switch (S2)

This switch toggles between line in and MP3 decoder to audio jack output P1. Toggling switch “up” towards north end of board means playback from MP3 decoder. Toggling the switch “down” toward the south end of the board passes audio through the audio auxiliary input (1).
(NOTE: Switch must be held at position desired, if not it will move back to original position)

3 3.5 Audio jack line output (P2)

4 Light sensor connector (J58)

Must be visible to collect ambient light readings.

5 5V barrel jack connector (J110)

This is the 5V barrel jack connector which provides the main source of power for most modules on the board.

6 USB input port

7 19V barrel jack connector (J56)

This barrel jack connector will supply the 19V requirement to light up the LCD display.

8 Push button (S1)

Push button used as a reset for different modules on the board, including the CPU, music decoder, the LCD and the motion detector.

9 On-On switch

Switch that toggles between External Interrupt 0 (Capacitive Touch Panel) and boot loader pin.

10 On-Off switch

Switch that toggles between high/low voltage for boot loading.

11 RS232 Connector (J46)

Connector used to program the CPU.

12 LED (D2)

LED indicator used to indicate when the CPU is in Reset state. This is a 3.3 V pin. LOW on this pin indicates LPC2478 being in Reset state.

13 LED (D3)

LED indicator used to indicate when RTC alarm is generated. This is a 1.8 V pin. It goes HIGH when a RTC alarm is generated.

14 LED (D1)

USB port 1 GoodLink LED indicator. It is LOW when device is configured (non-control endpoints enabled), or when host is enabled and has detected a device on the bus. It is HIGH when the device is not configured or when host is enabled and has not detected a device on the bus, or during global suspend. It transitions between LOW and HIGH (flashes) when host is enabled and detects activity on the bus.

15 Header pin (J99)

Header connected to the music decoder’s right channel output.

16 Header pin (J98)

Header connected to the music decoder’s left channel output.

17 Header pin (J96)

Header connected to the music decoder’s byte synchronization signal.

18 Header pin (J92)

Header connected to the music decoder’s chip select input, which is active LOW.

19 Header pin (J93)

Header connected to the music decoder’s clock for serial bus.

20 Header pin (J94)

Header connected to the music decoder’s serial input.

21 Header pin (J97)
Header connected to the music decoder's serial output, which is active when XCS=0, regardless of XRESET.
22 Header pin (J95)
Header connected to the capacitive touch panel's serial I2C data pin.
23 Header pin (J102)
Header connected to the motion detector's RXD receive data.
24 Header pin (J103)
Header connected to the motion detector's TXD transmit data.
25 Header pin (J107)
Header connected to the SD connector's command pin.
26 Header pin (J108)
Header connected to the SD connector's data input/output.
27 Header pin (J105)
Header connected to the SD connector's data input/output.
28 Header pin (J106)
Header connected to the SD connector's data input/output.
29 Header pin (J54)
Header connected to MAX Level shifter T2-out.
30 Header pin (J55)
Header connected to MAX Level shifter R2-in.
31 Header pin (J42)
Header connected to MAX Level shifter R2-out.
32 Header pin (J53)
Header connected to MAX Level shifter T2-in.
33 LED (D4)
Indicator LED, turns on when data is detected from RS232 connector.
34 PIR connector
Connector used to mount motion detector module onto the board.
35 Header pin (J109)
Header connected to USB port's data receive (RXD) line.
36 Header pin (J104)
Header connected to USB port's data transmit (TXD) line.
37 Header pin (J101)
Header connected to the Wifi's UART-TX (8-mA drive, 3.3-V tolerant).
38 Header pin (J100)
Header connected to the Wifi's UART-RX (3.3-V tolerant).

Operating Instructions

For some reason our voltage dividers do a bad job of actually regulating voltage, so optimal operating conditions are to plug the board in with the barrel jack connector (5) connected to a

power supply that supplies a 3.9V supply. The barrel jack (7) takes a standard ~19V laptop charger and powers the backlight of the LCD screen. When trying to program the board, the RS232 connection from the computer should not be directly connected to the RS232 connector(11) on the board when using a null-modem cable. Instead, pin2 and pin3 need to be swapped.
To begin operation, first put the Python.serial code into a test.py file in the same folder as the combined mp3s that must under 128Mbits. Then, reset the board and wait until board begins printing ...waiting. Now you can close putty, and begin running the test.py file to transfer over the mp3 data. Once the data is transferred, serial will close and the mp3 will be on the sdram. The PIR will send a signal to the mp3 that tells it to play music when it detects hand motion nearby, and the light sensor will toggle the music played between the two addresses based on light conditions.

Source Code

Python.Serial code (For file transfer through rs232):

```
#this file should be put in test.py in same folder as test.mp3 after py.serial is installed
#!/mp3directory
import serial, time, sys, argv
test = open('./test.mp3');
ser = serial.Serial(
        port = "/COM1",
        baudrate = 9600,
        bytesize = serial.EIGHTBITS,
        parity = serial.PARITYNONE,
        stopbits = serial.STOPBITS_ONE,
        timeout = 1,
        #disable software flow control
        xonxoff = False,
        #disable hardware flow control
        rtscts = False,
        dsrdtr = False,
        writeTimeout = 2
        )
try ser.open()
except Exception, e:
        print "open serial port: " + str(e)
        exit()
if ser.isOpen():
        try:
        ser.flushInput()
        ser.flushOutput()
```

```
        ser.write(test)
        while True:
                print “Writing...”
        #for main to verify data is sent
        ser.write(0b11111111)
        ser.write(0b11111111)
        ser.write(0b11111111)
        ser.write(0b11111111)
        ser.write(0b11111111)
        ser.close()
else:
        print “cannot open port”
```

Light Sensor – TI OPT101

The light sensor takes in light levels through the photodiode on top of the chip. It outputs a voltage that scales up linearly with light intensity. As the light intensity is at max, the voltage of the chip output is at Vcc (normally ~3.3V). When the sensor detects no light, the output voltage is 0. This is connected to the CPU's embedded ADC converter to line 0. This value is polled in an infinite main that checks to see if the light level is changing. Calling **ADC_out()** returns the current value of the ADC converted light level.

SD Card Socket

The MCI code is able to detect an SDSC card using the **MCICard_Init()** function, but we were not able to get it to successfully write to the fifo. Significant changes include adding new commands to fit updated specifications, and revision of testing strategy.

Motion Sensor - ZMOTION DETECTION MODULE Z8FS040

The motion sensor is also polled to make the music start and stop playing as it receives a signal. The sensitivity is set low so that it detects near field motion and should not detect things beyond a few inches range. Using a detection function in main called **PIR_detection()**, we can check the status register of the PIR to determine whether it has detected a hand swipe. Then, in the infinite loop in the main function, we can determine whether the music should play or stop using a counter to count the number of times motion has been detected.

SDRAM – Micron MT48LC4M16A2P-7E:G TR

The SDRAM is used to temporarily host the music files since the SD card is not working. In order to put music on the SDRAM you need to first check the file size of the two mp3s that you want to send. They need to be combined into one file called test.mp3 and put in the same folder as the python code for the rs232. Then, you set the address of each mp3 in ex_sdram.h based on the base address of the sdram and size of the mp3 so that the main.c code knows where each mp3 starts. **SDRAM_Init()** initializes the sdram and **SDRAMTest()** verifies that it is

working correctly.

Music Decoder – VLSI VS1011

The music decoder has all functions (test and not) that can be played around with in the file “spitest.c”. In the function MP3_Test(), you can choose to comment out sections for the sine test and volume test. The function MP3_Play() begins playing the music from an address in the SDRAM, and the function MP3_Stop() stops the music from playing. This is generally called when motion is detected in main.

LCD/Touchscreen

The LCD screen has little functionality for our project. First off, due to an error in our design, the capacitive touch panel was taken out of the picture. Therefore, the LCD simply displays the song name for the current song being played. After initialization using the **lcd_hw_init()** function, found in lcd_hw.c, we pull the song information from the SDRAM to display the song title using the **lcd_putString(WORD x, WORD y, BYTE *pStr)** function found in lcd_grph.c.