# WORKING with DIGITAL AUDIO - Code Snips

**Note: These Code Snips are taken straight from the book chapter; i.e. the “Program Examples”. In some cases therefore they are not complete programs.**

/\* Program Example 13.1: MIDI messaging with variable scroll speed

\*/

#include "mbed.h"

#include "USBMIDI.h”

USBMIDI midi; // initialise MIDI interface

AnalogIn Ain(p19); // create analog input

int main() {

while (1) {

for(int i=48; i<72; i++) { // step through notes

midi.write(MIDIMessage::NoteOn(i)); // note on

wait(Ain); // pause

midi.write(MIDIMessage::NoteOff(i)); // note off

wait(2\*Ain); // pause

}

}

}

Program Example 13.1: MIDI messaging with variable scroll speed

/\* Program Example 13.2: MIDI messaging controlled by switch and potentiometer

\*/

#include "mbed.h"

#include "USBMIDI.h"

USBMIDI midi;                        // initialise MIDI interface

DigitalIn Switch(p14);

AnalogIn Ain(p20);

int main(){

    while (1) {

        if (Switch==1) {

            int note = 48+72\*Ain;                     // calculate note value

            midi.write(MIDIMessage::NoteOn(note));    // note on

            wait(0.2);

            midi.write(MIDIMessage::NoteOff(note));    // note on

        }

    }

}

Program Example 13.2: MIDI messaging controlled by a switch and a potentiometer

/\* Program Example 13: Read MIDI messages and display key and velocity data to a host terminal application

\*/

#include "mbed.h"

#include "USBMIDI.h"

USBMIDI midi;

void read\_message(MIDIMessage msg){

switch (msg.type()) {

case MIDIMessage::NoteOnType:

printf("Note On key:%d vel:%d\n", msg.key(),msg.velocity());

break;

case MIDIMessage::NoteOffType:

printf("Note Off key:%d vel:%d\n", msg.key(),msg.velocity());

break;

}

}

int main()

{

midi.attach(read\_message); // call back for messages receive

while (1) {

}

}

Program Example 13.3: Reading and displaying MIDI messages

/\* Program Example 13.4: Simple mbed synthesizer

\*/

#include "mbed.h"

#include "USBMIDI.h"

USBMIDI midi;

PwmOut sound(p21);

void read\_message(MIDIMessage msg){

float freq=440\*pow(2,(msg.key()-69)/12.0); // calculate note frequency

switch (msg.type()) {

case MIDIMessage::NoteOnType:

printf("NoteOn key:%d vel: %d%d\n", msg.key(), msg.velocity());

sound.period(1 / freq); // Set PWM frequency

sound = 0.5; // Switch PWM on (50% duty cycle)

break;

case MIDIMessage::NoteOffType:

printf("NoteOff key:%d, vel: %d\n", msg.key(), msg.velocity());

sound = 0; // Switch PWM off (0% duty cycle)

break;

}

}

int main()

{

midi.attach(read\_message); // attach read\_message function to interrupt

while (1) {

}

}

Program Example 13.4: Simple mbed MIDI synthesizer

/\* Program Example 13.5 Audio signal input and output

\*/

#include "mbed.h"

//mbed objects

AnalogIn Ain(p15); // audio signal in

AnalogOut Aout(p18); // audio signal out

Ticker s20khz\_tick;

//function prototypes

void s20khz\_task(void);

//variables and data

float data\_in, data\_out;

//main program start here

int main() {

s20khz\_tick.attach\_us(&s20khz\_task,50); // attach task to 50us tick

}

// function 20khz\_task

void s20khz\_task(void){

data\_in=Ain;

data\_out=data\_in;

Aout=data\_out;

}

Program Example 13.5: Audio signal input and output

/\*Program Example 13.6 Low pass filter function

\*/

float LPF(float LPF\_in){

float a[4]={1,2.6235518066,-2.3146825811,0.6855359773};

float b[4]={0.0006993496,0.0020980489,0.0020980489,0.0006993496};

static float LPF\_out;

static float x[4], y[4];

x[3] = x[2]; x[2] = x[1]; x[1] = x[0]; // move x values by one sample

y[3] = y[2]; y[2] = y[1]; y[1] = y[0]; // move y values by one sample

x[0] = LPF\_in; // new value for x[0]

y[0] = (b[0]\*x[0]) + (b[1]\*x[1]) + (b[2]\*x[2]) + (b[3]\*x[3])

+ (a[1]\*y[1]) + (a[2]\*y[2]) + (a[3]\*y[3]);

LPF\_out = y[0];

return LPF\_out; // output filtered value

}

Program Example 13.6: Low-pass filter function

/\* Program Example 13.7 Delay / Echo Effect

\*/

#include "mbed.h"

AnalogIn Ain(p15); //object definitions

AnalogOut Aout(p18);

AnalogIn delay\_pot(p16);

AnalogIn feedback\_pot(p17);

Ticker s20khz\_tick;

void s20khz\_task(void); // function prototypes

#define MAX\_BUFFER 14000 // max data samples

signed short data\_in; // signed allows positive and negative

unsigned short data\_out; // unsigned just allows positive values

float delay=0;

float feedback=0;

signed short buffer[MAX\_BUFFER]={0}; // define buffer and set values to 0

int i=0;

//main program start here

int main() {

s20khz\_tick.attach\_us(&s20khz\_task,50);

}

// function 20khz\_task

void s20khz\_task(void){

data\_in=Ain.read\_u16()-0x7FFF; // read data and normalise

buffer[i]=data\_in+(buffer[i]\*feedback); // add data to buffer data

data\_out=buffer[i]+0x7FFF; // output buffer data value

Aout.write\_u16(data\_out); // write output

if (i>(delay)){ // if delay loop has completed

i=0; // reset counter

delay=delay\_pot\*MAX\_BUFFER; // calculate new delay buffer size

feedback=(1-feedback\_pot)\*0.9; // calculate feedback gain value

}else{

i=i+1; // otherwise increment delay counter

}

}

Program Example 13.7: Delay / echo effect

/\* Program Example 13.8 Wave file header reader

\*/

#include "mbed.h"

#include "SDFileSystem.h"

SDFileSystem sd(p5, p6, p7, p8, "sd");

char c1, c2, c3, c4; // chars for reading data in

int AudioFormat, NumChannels, SampleRate, BitsPerSample ;

int main() {

printf("\n\rWave file header reader\n\r");

FILE \*fp = fopen("/sd/sinewave.wav", "rb");

fseek(fp, 20, SEEK\_SET); // set pointer to byte 20

fread(&AudioFormat, 2, 1, fp); // check file is PCM

if (AudioFormat==0x01) {

printf("Wav file is PCM data\n\r");

}

else {

printf("Wav file is not PCM data\n\r");

}

fread(&NumChannels, 2, 1, fp); // find number of channels

printf("Number of channels: %d\n\r",NumChannels);

fread(&SampleRate, 4, 1, fp); // find sample rate

printf("Sample rate: %d\n\r",SampleRate);

fread(&BitsPerSample, 2, 1, fp); // find resolution

printf("Bits per sample: %d\n\r",BitsPerSample);

fclose(fp);

}

Program Example 13.8: Wave header reader

/\* Program Example 13.9: 16-bit mono wave player

\*/

#include "mbed.h"

#include "SDFileSystem.h"

#define BUFFERSIZE 0xfff // number of data in circular buffer = 4096

SDFileSystem sd(p11, p12, p13, p14, "sd"); //SD Card object

AnalogOut DACout(p18);

Ticker SampleTicker;

int SampleRate;

float SamplePeriod; // sample period in microseconds

int CircularBuffer[BUFFERSIZE]; // circular buffer array

int ReadPointer=0;

int WritePointer=0;

bool EndOfFileFlag=0;

void DACFunction(void); // function prototype

int main() {

FILE \*fp = fopen("/sd/testa.wav", "rb"); // open wave file

fseek(fp, 24, SEEK\_SET); // move to byte 24

fread(&SampleRate, 4, 1, fp); // get sample frequency

SamplePeriod=(float)1/SampleRate; // calculate sample period as float

SampleTicker.attach(&DACFunction,SamplePeriod); // start output tick

while (!feof(fp)) { // loop until end of file is encountered

fread(&CircularBuffer[WritePointer], 2, 1, fp);

WritePointer=WritePointer+1; // increment Write Pointer

if (WritePointer>=BUFFERSIZE) { // if end of circular buffer

WritePointer=0; // go back to start of buffer

}

}

EndOfFileFlag=1;

fclose(fp);

}

// DAC function called at rate SamplePeriod

void DACFunction(void) {

if ((EndOfFileFlag==0)|(ReadPointer>0)) { // output while data available

DACout.write\_u16(CircularBuffer[ReadPointer]); // output to DAC

ReadPointer=ReadPointer+1; // increment pointer

if (ReadPointer>=BUFFERSIZE) {

ReadPointer=0; // reset pointer if necessary

}

}

}

Program Example 13.9: Wave file player utilising a circular buffer

// Program Example 13.10 program structure for outputting high-quality audio

//

#include "mbed.h"

#include "TLV320.h" // the I2SSlave header is linked from within TLV320.h

#define BUFFERSIZE 0xff // number of data in circular buffer = 256

#define PACKETSIZE 8 // number of data values in a single audio package

TLV320 audio(p9, p10, 52, p5, p6, p7, p8, p29); //TLV320 object

int CircularBuffer[BUFFERSIZE]; // circular buffer array

int ReadPointer=0;

int WritePointer=0;

// \*\* Additional variables to be declared here

// \*\* Function prototypes

void SetupAudio (void);

void PlayAudio(void);

void FillBuffer(void);

// \*\* Main function

int main(){

SetupAudio(); // call SetupAudio function

while (1) {

FillBuffer(); //continually fill circular buffer

}

}

// \*\* Additional functions to be added here

Program Example 13.10 program structure for outputting high-quality audio

// Program Example 13.11 functions for initialising high-quality audio output

//

// \*\* Function to setup TLV320 \*\*\*

void SetupAudio(void){

audio.power(0x02); //power up TLV to audio input/output mode

audio.frequency(44100); //set sample frequency

audio.format(16, 0); //set transfer protocol - 16 bit, stereo

audio.outputVolume(0.8, 0.8);

audio.attach(&PlayAudio); //attach interrupt to send data to TLV320

audio.start(TRANSMIT); //interrupt come from the I2STXFIFO only

}

// \*\* Function to read from circular buffer and send data to TLV320 \*\*\*

void PlayAudio(void){

audio.write(CircularBuffer, ReadPointer, PACKETSIZE);// write to buffer

ReadPointer += PACKETSIZE; // increment read pointer

ReadPointer &= BUFFERSIZE; // if end of buffer then reset to 0

theta -= PACKETSIZE; // decrement theta

}

Program Example 13.11: Functions for initialising high-quality audio output

// Program Example 13.12 Function to load data to circular buffer

//

void FillBuffer(void){

if (theta < BUFFERSIZE) {

for (int i=0; i<PACKETSIZE; i++) { // loop for PACKETSIZE samples

// create squarewave with freq=fs/(2\*halfperiod)

if ((counter+i)<halfperiod) {

x=0x8000; // 2s compliment for -1 (16 bit)

} else if ((counter+i)<(halfperiod\*2)) {

x=0x7fff; // 2s compliment for +1 (16 bit)

} else {

counter=0;

}

//put data in buffer (right = MS 16 bits, left = LS 16 bits)

CircularBuffer[WritePointer+i]=(x<<16)|(x);

}

theta+=PACKETSIZE; // increment theta

WritePointer+=PACKETSIZE; // increment write pointer

WritePointer &=BUFFERSIZE; // if end of buffer then reset to 0

counter+=PACKETSIZE; // increment square wave counter

}

}

Program Example 13.12: Function to load square wave data to the circular buffer

// Program Example 13.13: Code structure for a high-quality wave file player.

//

#include "mbed.h"

#include "SDHCFileSystem.h"

#include "TLV320.h"

SDFileSystem sd(p11, p12, p13, p14, "sd"); //SD Card object

TLV320 audio(p9, p10, 52, p5, p6, p7, p8, p29); //TLV320 object

#define BUFFERSIZE 0xfff // number of data in circular buffer = 4096

#define PACKETSIZE 8 // number of data values in a single audio package

int CircularBuffer[BUFFERSIZE];

int ReadPointer = 0;

int WritePointer = 0;

int theta = 0;

int WavData[PACKETSIZE];

FILE \*WavFile;

// \*\* Function prototypes

void SetupAudio (void);

void PlayAudio(void);

void FillBuffer(void);

void OpenFiles(void);

// \*\* Main function

int main() {

OpenFiles();

SetupAudio();

while (1) {

FillBuffer(); //continually fill circular buffer

}

}

// \*\* Additional functions to be added here

Program Example 13.13: Code structure for a high-quality wave file player.

// Program Example 13.14 Functions to open SD wave file and read audio data

//

// \*\*\* function to open file \*\*\*

void OpenFiles(void) {

wait(1);

wavFile = fopen("/sd/Audio.wav", "r"); //open file

fseek(wavFile, 44, SEEK\_SET); // offset to data

wait(1);

}

// \*\*\* Function to load circular buffer from SD Card \*\*\*

void FillBuffer(void) {

if (theta < BUFFERSIZE) {

fread(&WavData, 4, PACKETSIZE, WavFile); //read 4 byte (i.e. 32 bit)

// data from the wav file

for (int i=0; i<PACKETSIZE; i++) {

CircularBuffer[WritePointer]=WavData[i];

theta++;

WritePointer++;

WritePointer &= BUFFERSIZE;

}

}

}

Program Example 13.14 Functions to open SD wave file and read audio data

// Program Example 13.15: Audio input / output with the TLV320

//

#include "mbed.h"

#include "TLV320.h"

TLV320 audio(p9, p10, 52, p5, p6, p7, p8, p29); //TLV320 object

#define BUFFERSIZE 0xff // number of data in circular buffer = 256

#define PACKETSIZE 4 // number of data values in a single audio package

int CircularBuffer[BUFFERSIZE];

int ReadPointer = 0;

int WritePointer = 0;

// \*\* Function to read from circular buffer and send data to TLV320 \*\*\*

void PlayAudio(void){

// read data to circular buffer and increment pointers

audio.read(); // read 4 values to rxBuffer

for (int i=0; i<PACKETSIZE; i++) {

CircularBuffer[WritePointer+i] = audio.rxBuffer[i]; // transfer data

}

WritePointer+=PACKETSIZE;

WritePointer &= BUFFERSIZE;

// write data from circular buffer and increment counters

audio.write(CircularBuffer, ReadPointer, PACKETSIZE);

ReadPointer += PACKETSIZE;

ReadPointer &= BUFFERSIZE;

}

// \*\* Function to setup TLV320 \*\*\*

void SetupAudio(void){

audio.power(0x02); //power up TLV

audio.frequency(44100); //set sample frequency

audio.format(16, 0); //set transfer protocol - 16 bit, stereo

audio.outputVolume(0.8, 0.8);

audio.inputVolume(0.8, 0.8);

audio.attach(&playAudio); //attach interrupt handler

audio.start(BOTH);

}

// \*\* Main function

int main(){

SetupAudio();

while (1) {

}

}

Program Example 13.15: Audio input / output with the TLV320