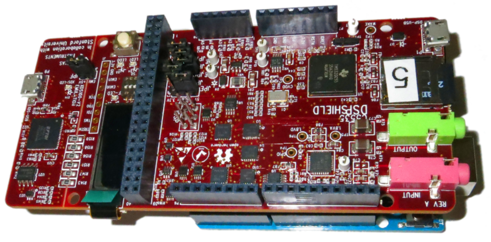
DSP Shield App Library Functions

Manual by Theo Diamandis and William Esposito



This document describes the methods included in the "DSPShield.h" library for Arduino, which can be found at github.com/wespo/DSPShield. The library works in conjunction with the DSP Shield Application / Energia sketch, which can be found at github.com/wespo/DSP-Shield-Mode-Application. Additionally, to use this library, the user must also include the "mailbox.h" DSP Shield communication library in their Arduino sketch. The mailbox library can be found at github.com/wespo/mailbox.

To run the DSP Shield application, the DSP Shield Mode Application must be installed on the DSP Shield's SD Card by copying bootimg.bin to the root directory of the card and restarting the DSP Shield.

Included below is a brief description of each function supported by the DSP Shield library, the syntax for the function call, and a description of all parameters used in the call. To invoke these functions, one prepends each call with "`DSPShield.`". For example, the initializer function would be invoked by writing "`DSPShield.init();`"

Contents:

init() 2
startLoopback() 2
stopLoopback() 2
disableFilter() 2
setInputGain() 3
setOutputVolume() 3
setIIRFilter() 3
setIIRCoefficients() 4
setFIRFilter() 4
setFIRCoefficients() 5
chirpStart() 6
toneStart() 7
toneStop() 7
spectrumStart() 8
spectrumStop() 8
displayPrint() 9

init()

Description:
Initializes the DSP Shield.

Syntax:
DSPShield.init()

Parameters:
None

startLoopback()

Description:
Starts DSP Shield's audio loopback

Syntax:
DSPShield.startLoopback()

Parameters:
None

stopLoopback()

Description:
Stops the DSP Shield's audio loopback

Syntax:
DSPShield.stopLoopback()

Parameters:
NonedisableFilter()

Description:
Disables the current filter on the selected channel

Syntax:
DSPShield.disableFilter(channel)
DSPShield.disableFilter()

Parameters:
channel: codec channel. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

setInputGain()

Description:
Amplifies the input

Syntax:
DSPShield.setInputGain(db)

Parameters:
db: gain by which to amplify the input, in decibels

setOutputVolume()

Description:
Sets output volume of DSP Shield

Syntax:
DSPShield.setOutputVolume(percent)

Parameters:
percent: an integer between 0 and 100

setIIRFilter()

Description:
Loads an IIR Filter from the SD card

Syntax:
DSPShield.setIIRFilter(channel, pass, response, order, cuttoff1, cutoff2)
DSPShield.setIIRFilter(channel, pass, response, order, cutoff)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

filter: type of filter to use. Options are LOW_PASS, HIGH_PASS, BAND_PASS, BAND_STOP

order: 41, 101, 201, or 511

cutoff: Increments are 1Hz steps from 10Hz to 20000Hz

setIIRCoefficients()

Description:
Send and IIR filter from the arduino.

Syntax:
DSPShield.setIIRCoefficients(channel, type, order, coefficients)
DSPShield.setIIRCoefficients(channel, type, order1, coefficients1, order2, coefficients2)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

type: HIGH_PASS or LOW_PASS

order: 41, 101, 201, or 511

coefficients: array of coefficients to be used as a filter. They are 16bit signed integers and can be generated in the DSP Shield FIT filter designer tool, matlab, or python.

setFIRFilter()

Description:
Loads an FIR filter from the SD Card

Syntax:

For a band/notch filter:
DSPShield.setFIRFilter(channel, filter, order, cutoff1, cutoff2)

For a low/high filter:
DSPShield.setFIRFilter(channel, filter, order, cutoff)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

filter: type of filter to use. Options are LOW_PASS, HIGH_PASS, BAND_PASS, BAND_STOP

order: 41, 101, 201, or 511

cutoff: Increments are 1Hz steps from 10Hz to 20000Hz

setFIRCoefficients()

Description:
Send an FIR filter from the Arduino

Syntax:
DSPShield.setFIRCoefficients(channel, coeffNum, coefficients)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

coeffNum: number of coefficients. This value should be no more than 511 as you will overload the DSP and Arduino.

coefficients: array of coefficients to be used as a filter. They are 16bit signed integers and can be generated in the DSP Shield FIT filter designer tool, matlab, or python.

chirpStart()

Description:
Generates a chirp

Syntax:
DSPShield.chirpStart(channel, startingFrequency, endingFrequency, duration, amplitude, loopMode, mixMode, waveType)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

startingFreuency: floating point frequency at which to begin chirp. DC to 20,000Hz

endingFreuency: floating point frequency at which to finish chirp. DC to 20,000Hz

duration: in 10ms steps

amplitude: How load the output should be (a floating point number 0-1). For headphones, small values (~0.01) are appropriate volumes. For oscilloscopes, values as high as 1.0 are reasonable and allow for maximum dynamic range.

loopMode: what to do at the end of the chirp. Options are CHIRP_LOOP, CHIRP_HOLD, CHIRP_STOP

mixMode: Determines whether or not the output will be mixed with the codec input or simple overwrite it. Options are MODE_SUM, MODE_REPLACE

waveType: type of wave to output. Options are WAV_SIN, WAV_SAW, WAV_TRI, WAV_SQU. Note that loading a new DDS Waveform (changing shape) will briefly interrupt audio. This protects against occasional DSP resource conflicts over the DMA buffer caused when the SD card load and Audio interrupt collide.

toneStart()

Description:
Generates a tone

Syntax:
DSPShield.toneStart(channel, frequency, amplitude, mixMode, waveType)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

frequency: floating point frequency to output. DC to 20,000Hz

amplitude: How load the output should be (a floating point number 0-1). For headphones, small values (~0.01) are an appropriate volume. For oscilloscopes, values as high as 1.0 are reasonable and allow for maximum dynamic range.

mixMode: Determines whether or not the output will be mixed with the codec input or simple overwrite it. Options are MODE_SUM, MODE_REPLACE

waveType: type of wave to output. Options are WAV_SIN, WAV_SAW, WAV_TRI, WAV_SQU. Note that loading a new DDS Waveform (changing shape) will briefly interrupt audio. This protects against occasional DSP resource conflicts over the DMA buffer caused when the SD card load and Audio interrupt collide.

toneStop()

Description:
Stops the current tone on the specified channel.

Syntax:
DSPShield.toneStop(channel)

Parameters:
channel: codec channel. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

spectrumStart()

Description:
Causes the DSP Shield to start sending back the Spectrum (FFT) of the received sound to the Arduino for use

Syntax:
DSPShield.spectrumStart(channel, interval, source, callbackFunction)
DSPShield.spectrumStart(channel, interval, source, complex, numPoints, windowType, callbackFunction)

Parameters:
channel: codec channel to use. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

interval: milliseconds between transmitted spectrum frames. Aliased to 10ms intervals. Avoid low intervals with large FFT sizes.

source: Options are SOURCE_INPUT, SOURCE_OUTPUT, SOURCE_CODEC. Source input and output provide the spectrum of the signal going into or coming out of the filter chain. Source codec provides the signal coming from the codec only, ignoring or coming out of the filter chain

complex: SPECTRUM_MAGNITUDE means the DSP will return the magnitude squared of each frequency bin. SPECTRUM_COMPLEX means that the DSP will return the complex value for each frequency bin as an array of ints stored in the format: [real(0), imaginary(0), real(1), imaginary(1), … real(n), imaginary(n)]

numPoints: length of FFT. A power of two from 32 to 512. numPoints/2 bins will be returned

windowType: Options are WINDOW_NONE (Rectangular Windowing), WINDOW_BLACKMAN (Blackman Window), WINDOW_HAMM (Hamming Window), WINDOW_HANN (Hanning Window)

callbackFunction: name of a function on the Arduino to call when a new spectrum is ready

spectrumStop()

Description:
Stops the DSP shield from sending back the spectrum of the received sound

Syntax:
DSPShield.spectrumStop(channel)

Parameters:
channel: codec channel. Options are CHAN_LEFT, CHAN_RIGHT, CHAN_BOTH

displayPrint()

Description:
Displays a string to the DSP Shield’s OLED display.

Syntax:
DSPShield.displayPrint(string)

Parameters:
string: a string to display on the OLED display