DSP pitch detector API

The following document will describe and explain the application programming interface for the “Pitch detection” application implemented in this assignment.

* Functions in “Audio\_Pitch\_detection.c”
  + Void main(void)

|  |  |
| --- | --- |
| Description:  The main method runs the actual pitch detection sequence. This is the function which is automatically called, when the DSP chip is powered up, assuming the chip is programmed already, or when the program is loaded onto the chip itself.  The pitch detector is designed to detect, identify and display a pitch between 10 Hz and 4kHz. It works with a sampling rate of 8kHz.  Any pitch lower or higher will be displayed as a “high” pitch. This is due to the resolution of the detector which is limited by the chosen frame size (here 256). | |
| Dependencies:  - <p33FJ256GP506.h>  - <dsp\h\dsp.h>  - <board\h\sask.h>  - <peripherals\adc\h\ADCChannelDrv.h>  - <peripherals\pwm\h\OCPWMDrv.h>  - <board\inc\ex\_sask\_generic.h>  - <board\inc\ex\_sask\_led.h>  - <peripherals\timers\inc\ex\_timer.h>  - "..\inc\ex\_audio\_process.h"  - "..\inc\LED\_control.h"  - "..\inc\FFT\_processing.h"  Definitions:  - FRAME\_SIZE 256 | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * Audio [signal]:   The pitch detector requires an audio signal as its input variable.  The signal is to be send into the appropriate peripheral slot with the jumper set to either “MIC” for a microphone input or “LINE IN” when using an external device to input a signal directly (i.e. phone, mp3 player, etc.)  The pitch detector was tested with a signal generator using a sinus wave at different input frequencies. | |
| Output:   * LED [visual]:   The pitch detector returns its results by using the LED peripherals provided by the DSP StarterKit.  It will display a certain combination of the LEDs, depending on its results:   * GREEN: Indicates a LOW pitch (between 10 and 800 Hz) * GREEN/YELLOW: Indicates a LOW to MEDIUM pitch (between 801 and 1600 Hz) * YELLOW: Indicates a MEDIUM pitch (between 1601 and 2400 Hz) * YELLOW/RED: Indicates a MEDIUM to HIGH pitch (between 2401 and 3200 Hz) * RED: Indicates a HIGH pitch (between 3201 and 4000 Hz)   \*NOTE\*: The RED indicator will also show for a pitch below 10 Hz and above 4kHz. This is the results of insufficient resolution for lower frequencies and sampling for higher frequencies | |
| Error handling:   * Program calculates a pitch higher then 4kHz:   If the pitch level returns a value that is not handled by the switch/case statement, the program is getting kicked into an “ERROR\_STATE”, which will flash all 3 LEDs at the same time until it is reset to “READY\_STATE” by pressing switch 1 and switch 2 at the same time. | |
| Functions from other sources (see their documentation for further details):   * FFT\_Processing->FFT() * FFT\_Processing->pitchDetection() * LED\_control->displayLED() * LED\_control->displayState() | |

* Functions in “LED\_control.c”
  + Int displayState(int)

|  |  |
| --- | --- |
| Description:  “displayState” is a function used for program control. It provides options to set the program into a desired state and can therefore be used as an exception handler or to test for successful programming of the chip.  It provides clear and distinct visual feedback, describing what state the program is currently in.  The ready state can be left by pressing switch S1.  The error state can be reset to ready by pressing both switches S1 and S2. | |
| Dependencies:   * NONE   Definitions:   * NONE | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * State [int]:   Defines the state the program will be set into.   * 0: Ready state (Cycling all three LEDs). * 1: Run state (All LEDs off+output from the pitch detector). * 2: Error state (Flashing all 3 LEDs at the same time). | |
| Output:   * currentState [int]:   Returns the current state of the program back to the calling point. | |
| Error handling:   * If an integer values is provided that does not correspond to either state, the program kicks into error state as a default. | |
| Functions from other sources (see their documentation for further details):   * NONE | |

* + Void readyState(void)

|  |  |
| --- | --- |
| Description:  This function shows the ready state on the external peripherals provided by the DSP kit.  It generates a sequence of all 3 LEDs, swapping them every .3 seconds.  This state can be cancelled by pressing the switch S1. | |
| Dependencies:   * <board\h\sask.h> * <peripherals\timers\inc\ex\_timer.h> * "..\inc\LED\_control.h"   Definitions:   * clock\_frequency 40e6 * timeout 0.3 | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * S1 [switch]:   Pressing the switch will terminate the ready state and continue the program from the calling point. | |
| Output:   * LED [visual]:   The ready state displays the current state of the program via the LED peripherals.  The ready state is shown as a cycling of the 3 LEDs with a delay of .3 seconds between each swap. | |
| Error handling:  In case of a calculation error which will determine that the next LED to be turned on is a LED that does not exist (integer exceeding the range of 0-2), the program will jump into the error state. | |
| Functions from other sources (see their documentation for further details):   * ex\_timer\_init() * ex\_timer\_wait() | |

* + Void errorState(void)

|  |  |
| --- | --- |
| Description:  This function displays the error state to the LED peripherals provided by the DSP kit.  It generates a sequence of all 3 LEDs, turning them on and off every .3 seconds.  This state can be cancelled by pressing the switches S1+S2. This will reset the program back to the ready state. | |
| Dependencies:   * <board\h\sask.h> * <peripherals\timers\inc\ex\_timer.h> * "..\inc\LED\_control.h"   Definitions:   * clock\_frequency 40e6 * timeout 0.3 | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * S1 & S2 [switch]:   Pressing both switches at the same time will exit the error state and reset the program back into ready state. | |
| Output:   * LED [visual]:   The error state displays the current state of the program via the LED peripherals.  The error state is shown by turning the 3 LEDs off and on, with a delay of .3 seconds between each swap. | |
| Error handling:   * NONE | |
| Functions from other sources (see their documentation for further details):   * ex\_timer\_init() * ex\_timer\_wait() | |

* + Void displayLED(int, int, int)

|  |  |
| --- | --- |
| Description:  The function enables the user to program a variable function output.  It supplies an option to target each LED separately and turn them on or off.  The first input connects to the red LED, the second to the yellow LED and the third value to the green LED.  The input value 0=>OFF, 10=>ON. | |
| Dependencies:   * <board\h\sask.h> * "..\inc\LED\_control.h"   Definitions:   * NONE | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * Red led [int]:   The first variable defines the red led (0=off, 1=on).   * Yellow led [int]:   The second variable defines the yellow led (0=off, 1=on).   * Green led [int]:   The third variable defines the green led (0=off, 1=on). | |
| Output:   * LED [visual]:   The function turns on the chosen LEDs (red, yellow, green) | |
| Error handling:   * NONE. | |
| Functions from other sources (see their documentation for further details):   * NONE. | |

* Functions in “FFT\_processing.c”
  + Void FFT(int, fractional \*, fractcomplex\*)

|  |  |
| --- | --- |
| Description:  The FFT function transforms an input signal into the frequency domain.  The input signal is being transformed into a number (given frame size) of frequency pins which have the height of the amplitude corresponding to this specific frequency. | |
| Dependencies:   * "..\inc\FFT\_processing.h" * <dsp.h>   Definitions:   * FFT\_FRAME\_SIZE 256 * SAMPLING\_RATE 8000 * COEFFS\_IN\_DATA 0xFF00 | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * Framesize [int]:   Defines the amount of data points given in the audioIN array.   * audioIN [fractional \*]:   Contains the recorded input signal gathered from the peripheral port.   * compX [fractcomplex \*]:   Address of an array with the size “framesize” which will contain the results of the FFT | |
| Output:   * compX [fractcomplex \*]:   Contains the FFT results in the frequency domain. | |
| Error handling:  NONE | |
| Functions from other sources (see their documentation for further details):   * dsp->TwidFactorInit() * dsp->FFTComplex() | |

* + Int pitchDetection(fractcomplex \*)

|  |  |
| --- | --- |
| Description:  This function calculates the pitch of a given frequency domain signal.  The pitch is detected by finding the highest value contained in the array and using the position as the frequency pin.  The detected pin is the multiplied by the value of a pin, which is calculated by dividing the sampling rate by the frame size.  Therefore the resolution and accuracy is dependent on the frame size of the system, here 256. This frame size achieves an error margin of ~ +-16Hz  This pitch detector is designed for a real signal and therefore ignores the mirrored upper half of the FFT results. | |
| Dependencies:   * "..\inc\FFT\_processing.h" * <dsp.h>   Definitions:   * FFT\_FRAME\_SIZE 256 * SAMPLING\_RATE 8000 | The listed header files and definitions are required to compile and run the pitch detector.  For details regarding each header file see the comments at the beginning of the desired file content. |
| Input parameters:   * compXFftResults [fractcomplex \*]:   Address of the array containing the frequency domain values for the processed input signal. | |
| Output:   * detectedPitchLvl [int]:   The detected pitch is converted into a single digit number and returned to the original program call. | |
| Error handling:  NONE | |
| Functions from other sources (see their documentation for further details):   * dsp->SquareMagnitudeCplx() * dsp->VectorMax() | |