DSP auralisation API

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

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

|  |  |
| --- | --- |
| Description:  The main method runs the auralisation program. This is the function 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 auralisation will show the frequency spectrum between 0 and 4000 Hz by playing back the individual components of the spectrum as a sound. It works with a sampling rate of 8kHz. | |
| 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\Peripheral\_control.h"  - "..\inc\Signal\_processing.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:   * Audio [signal]:   The auralisation 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 auralisation was tested with a signal generator using a sinus wave at different input frequencies. | |
| Output:   * LED [visual]:   The auralisation will output multiple LED variations which depict the current state of the program. More information on this can be found in the Peripheral control section.   * Audio Signal:   The frequency spectrum transformed into multiple sounds. 1 for each frequency bin. | |
| Error handling:   * If any runtime error occurs the program will switch into error mode:   “ERROR\_STATE”: 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):   * Signal\_Processing->FFT() * Signal\_Processing->generateAuralisation() * Signal\_Processing->inverseFFT() * Peripheral\_control->turnOffAll() * Peripheral\_control->analysingState() * Peripheral\_control->playbackState() * Peripheral\_control->displayState() | |

* Functions in “Peripheral\_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 analysing state can be left by pressing switch S2.  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). * analysingState[int]   Defines the type of display during the analysation process   * 0:analysation is finished(Yellow LED shows) * 1:Red LED shows(stage 1) * 2:Red/Yellow LEDs shows(stage 2) * 3:Red/Yellow/Green LEDs shows(stage 3) | |
| 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 shows the green LED until the program is continued.  This state can be cancelled by pressing the switch S1. | |
| Dependencies:   * <board\h\sask.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:   * 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 by the green LED. | |
| Error handling:  -NONE | |
| Functions from other sources (see their documentation for further details): | |

* + 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 analysingState(int)

|  |  |
| --- | --- |
| Description:  The function is used to depict the current state of the analysation process for the auralisation  It will show a combination of all 3 LEDs depending on what stage is currently being processed.  RED: Stage 1 - FFT function  RED/YELLOW: Stage 2 - Auralisation  RED/YELLOW/GREEN: Stage 3 – inverse FFT function  YELLOW: The analysation is finished and the program is ready to continue | |
| 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:   * 0 [int]:   Completion state.   * 1 [int]:   Display stage 1.   * 2 [int]:   Display stage 2.   * 3 [int]:   Display stage 3.   * S2 [switch]:   Pressing the switch while in state 0 will terminate the analysing state and continue the program from the calling point. | |
| Output:   * LED [visual]:   The function turns on the chosen LEDs (red, yellow, green) | |
| Error handling:   * If the input does not match the values 0-3 the program is kicked into error state:   This state can be left by resetting to the Ready state by pressing S1 and S2. | |
| Functions from other sources (see their documentation for further details):   * errorState() * turnOffAll() | |

* + Void playbackState()

|  |  |
| --- | --- |
| Description:  The function runs while the program plays back the auralisation.  While the audio is being played all 3 LEDs will cycle with an interval of .3 seconds per LED.  After the play back finishes the program is reset into 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:   * NONE | |
| Output:   * LED [visual]:   The function turns on the chosen LEDs (red, yellow, green) | |
| Error handling:   * If a calculation error occurs during the cycle the program is kicked into error state:   This state can be left by resetting to the Ready state by pressing S1 and S2. | |
| Functions from other sources (see their documentation for further details):   * errorState() * turnOffAll() * ex\_timer\_init() * ex\_timer\_wait() | |

* + Void turnOffAll()

|  |  |
| --- | --- |
| Description:  The function will turn off all 3 LEDs. | |
| Dependencies:   * <board\h\sask.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:   * NONE | |
| 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 | |

* + Void turnOnAll()

|  |  |
| --- | --- |
| Description:  This function will turn on all 3 LEDs. | |
| Dependencies:   * <board\h\sask.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:   * NONE | |
| 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 “Signal\_processing.c”
  + Void FFT(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 16 * LOG2N 4 * 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:   * 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() | |

* + void inverseFFT(fractional \*,fractcomplex \*)

|  |  |
| --- | --- |
| Description:  The inverse FFT function transforms the frequency domain into an audio signal.  The process is the inverse of the FFT function. | |
| Dependencies:   * "..\inc\FFT\_processing.h" * <dsp.h>   Definitions:   * FFT\_FRAME\_SIZE 16 * LOG2N 4 * 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:   * frctAudioWorkSpace [fractional \*]:   Address of an array with the size “framesize” which will contain the results of the inverse FFT   * compX [fractcomplex \*]:   Address of an array with the size “framesize” which contains the frequency domain to be transformed into an audio signal. | |
| Output:   * frctAudioWorkSpace [fractcomplex \*]:   Contains the inverse FFT results . | |
| Error handling:  NONE | |
| Functions from other sources (see their documentation for further details):   * dsp->TwidFactorInit() * dsp->IFFTComplex() | |

* + void generateAuralisation(fractcomplex[FFT\_FRAME\_SIZE][FFT\_FRAME\_SIZE], fractcomplex[FFT\_FRAME\_SIZE])

|  |  |
| --- | --- |
| Description:  This function splits the results of the FFT into its individual frequency pins in preparation for the inverse FFT function. | |
| Dependencies:   * NONE   Definitions:   * FFT\_FRAME\_SIZE 16 | 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:   * auralisation [fractcomplex[FFT\_FRAME\_SIZE][FFT\_FRAME\_SIZE]]:   A 2d Array with the size “framesize” which will contain the individual frequencies from the FFT results   * fftResults [fractcomplex[FFT\_FRAME\_SIZE]]:   An array with the size “framesize” which contains the results from the FFT function | |
| Output:   * auralisation [fractcomplex[FFT\_FRAME\_SIZE][FFT\_FRAME\_SIZE]]:   Contains the results of the function. | |
| Error handling:  NONE | |
| Functions from other sources (see their documentation for further details):   * NONE | |