LED VISUAL MUSIC EQ

Christian Knight

Nikko Noble

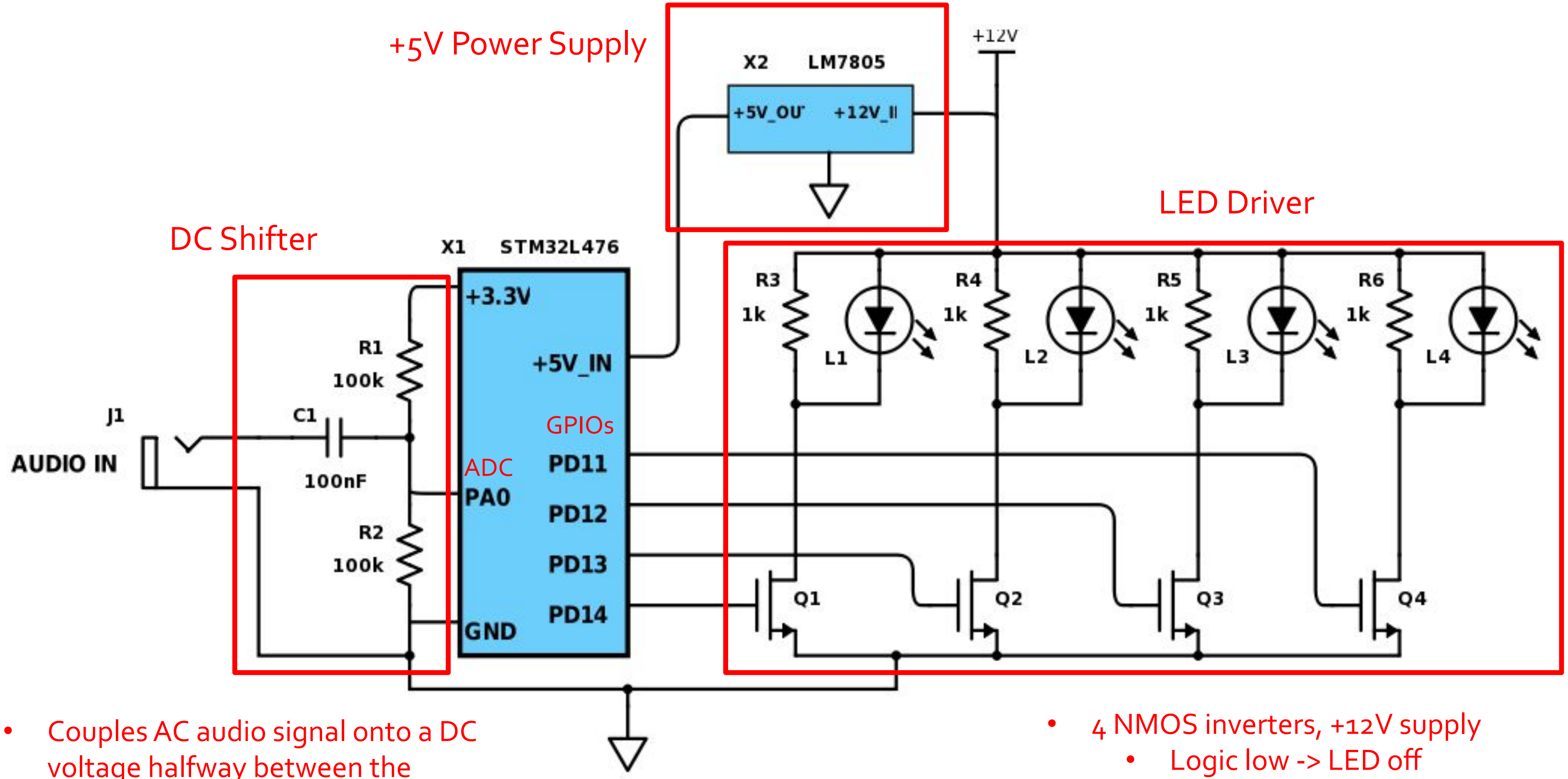
WHAT DOES IT DO?

- Get line-level audio signal (from phone, laptop, etc.)
- Use real-time DSP to split audio signal into 4 frequency bands
  - Bass
  - Lower mid-band
  - Upper mid-band
  - Treble
- Pulse LEDs based on intensity of music in each set frequency band
  - Create visual lightshow to “see” the music!

WHAT DOES IT REQUIRE?

- Microcontroller capable of real-time DSP
  - STM32L476G-Discovery Evaluation Board
    - Same board used in ECE486 DSP
    - ARM optimized DSP libraries & ECE486 sampling library used
    - 1 ADC used for mono audio input
    - 4 digital I/O pins used to control 4 LEDs

STM32L476G-Discovery Evaluation Board



- Couples AC audio signal onto a DC voltage halfway between the supplies to measure "negative" part of audio

- 4 NMOS inverters, +12V supply
  - Logic low -> LED off
  - Logic high -> LED on

HARDWARE SCHEMATIC

DIGITAL FILTERING

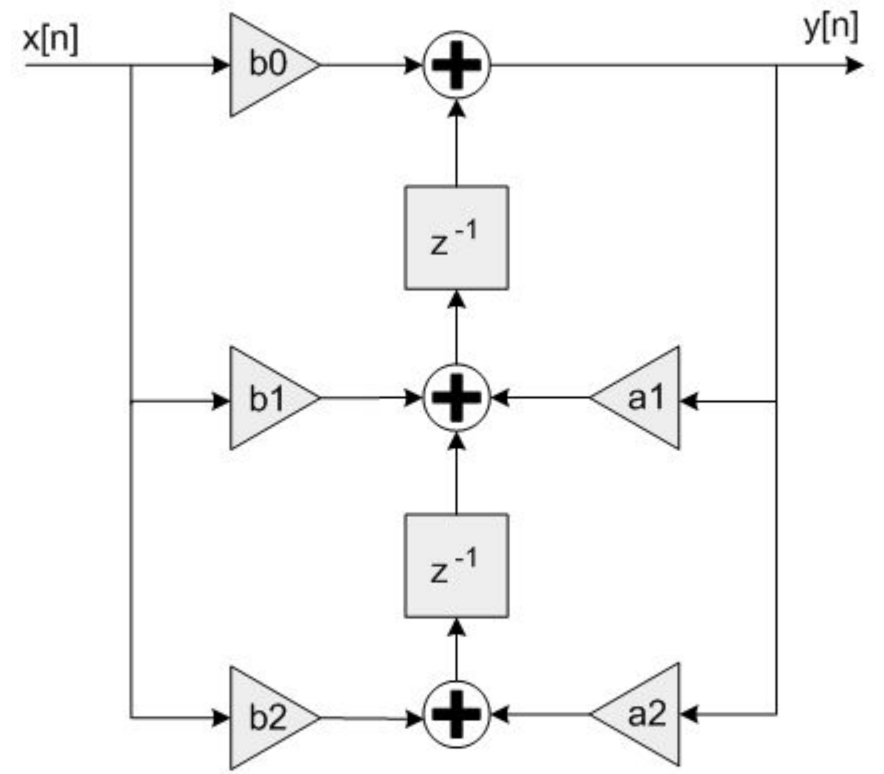
- Cascaded second order Biquad sections
- Direct form II transposed structure used

Algorithm

Each Biquad stage implements a second order filter using the difference equation:

```
y[n] = b0 * x[n] + d1
d1 = b1 * x[n] + a1 * y[n] + d2
d2 = b2 * x[n] + a2 * y[n]
```

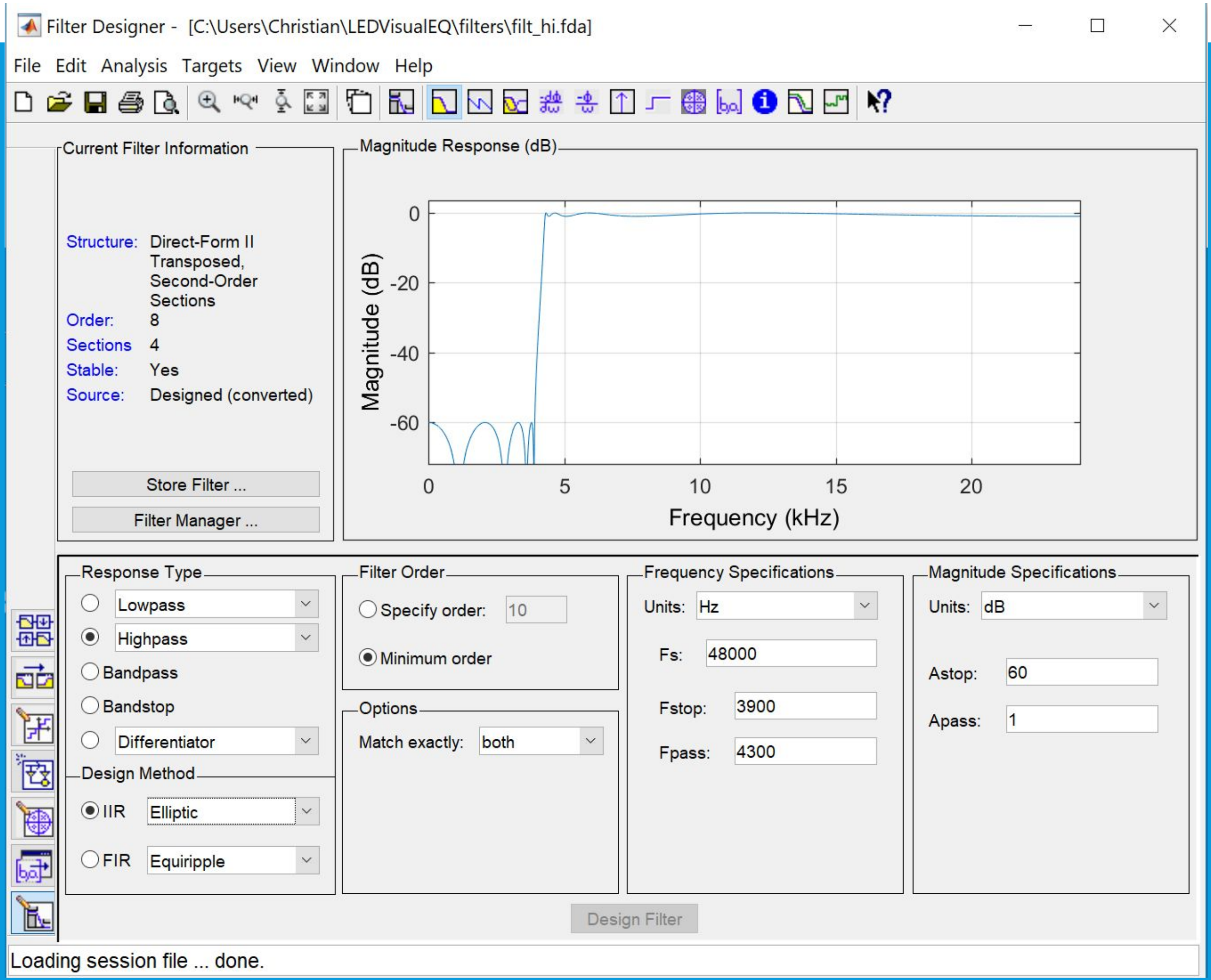
where d1 and d2 represent the two state values.



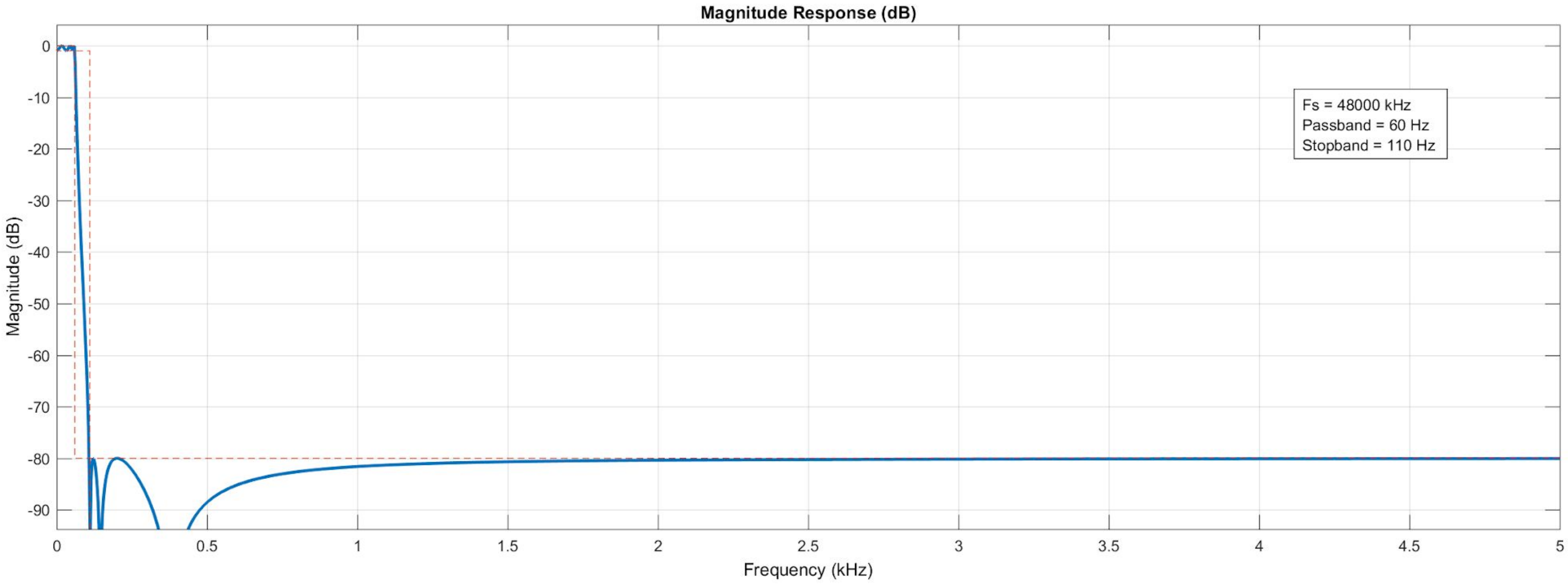
Single transposed Direct Form II Biquad

FILTER DESIGN & IMPLEMENTATION

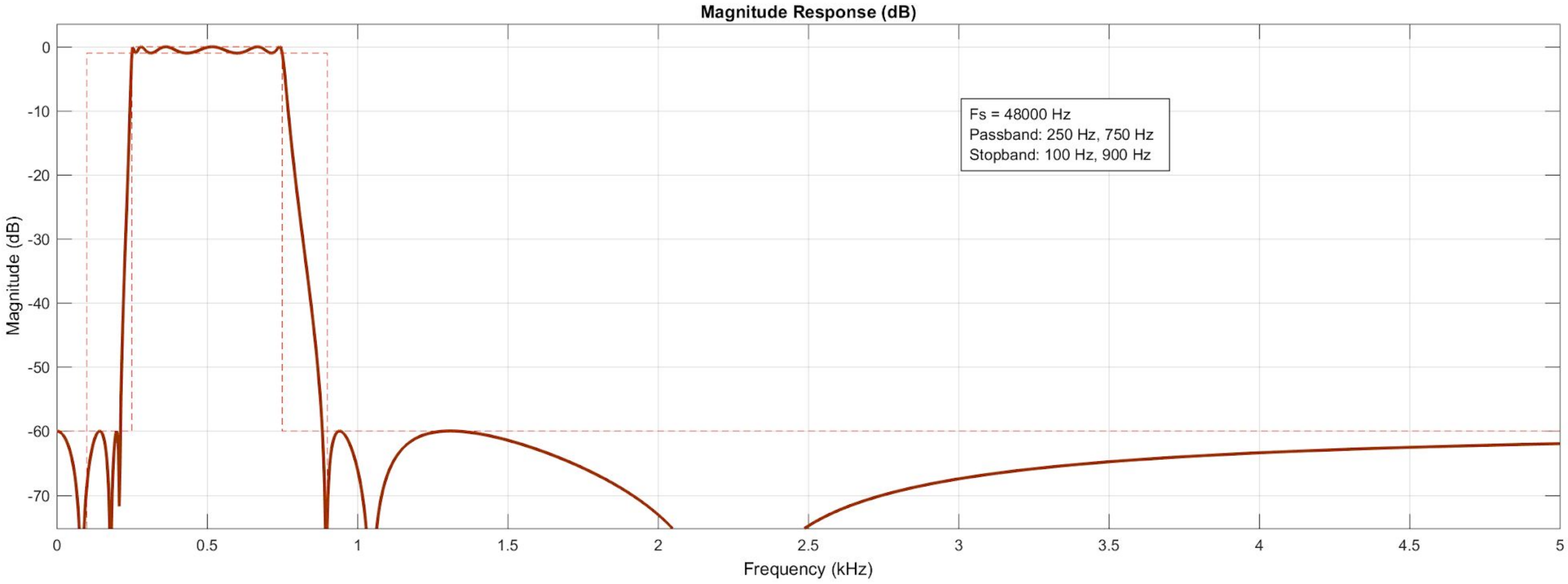
- Filter Design and Analysis Tool (fdatool)
  - Matlab DSP Toolbox
- Design 4 eliptic IIR biquad filters
- Export filters in a format usable by ARM's DSP library (b & a coeffs)



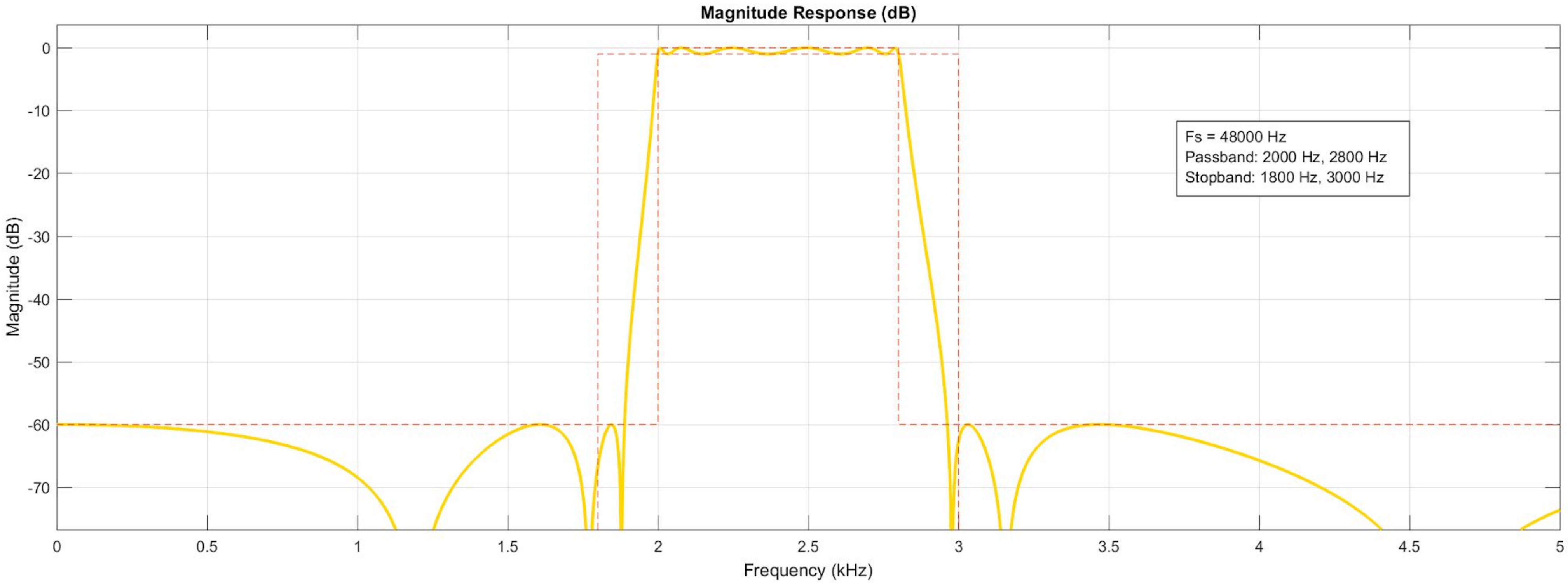
LOW PASS FILTER



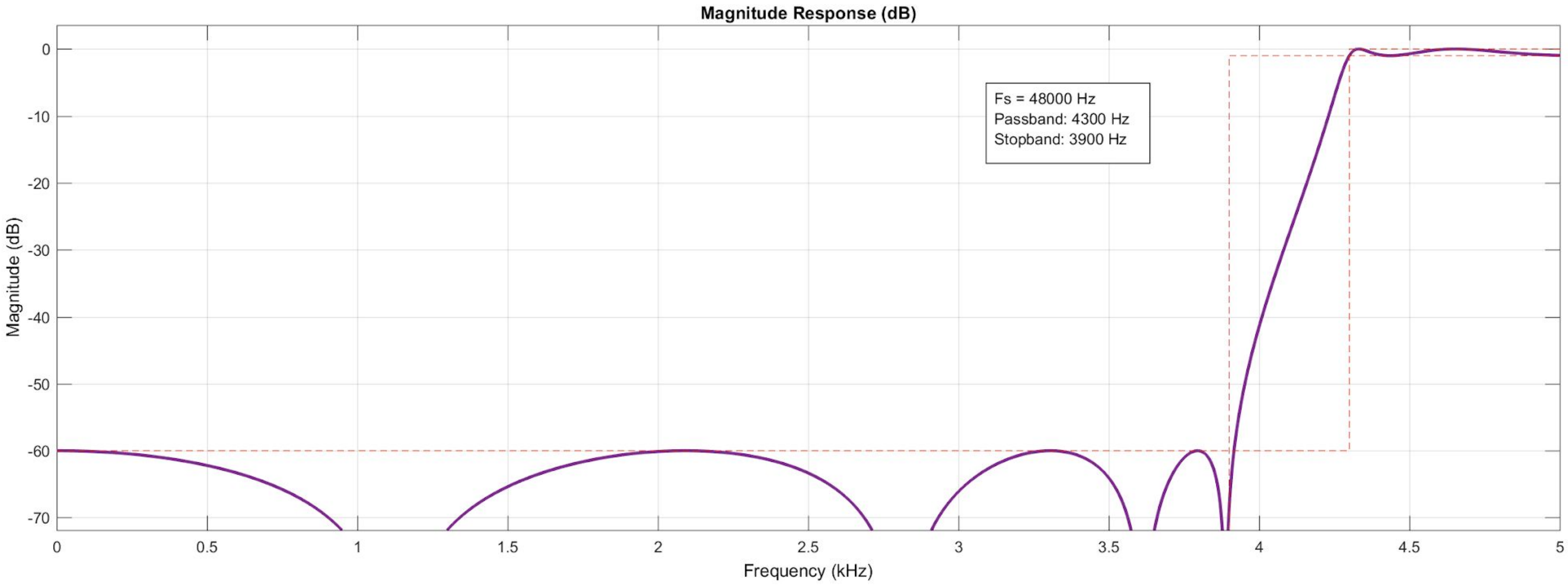
LOW-MID PASS FILTER



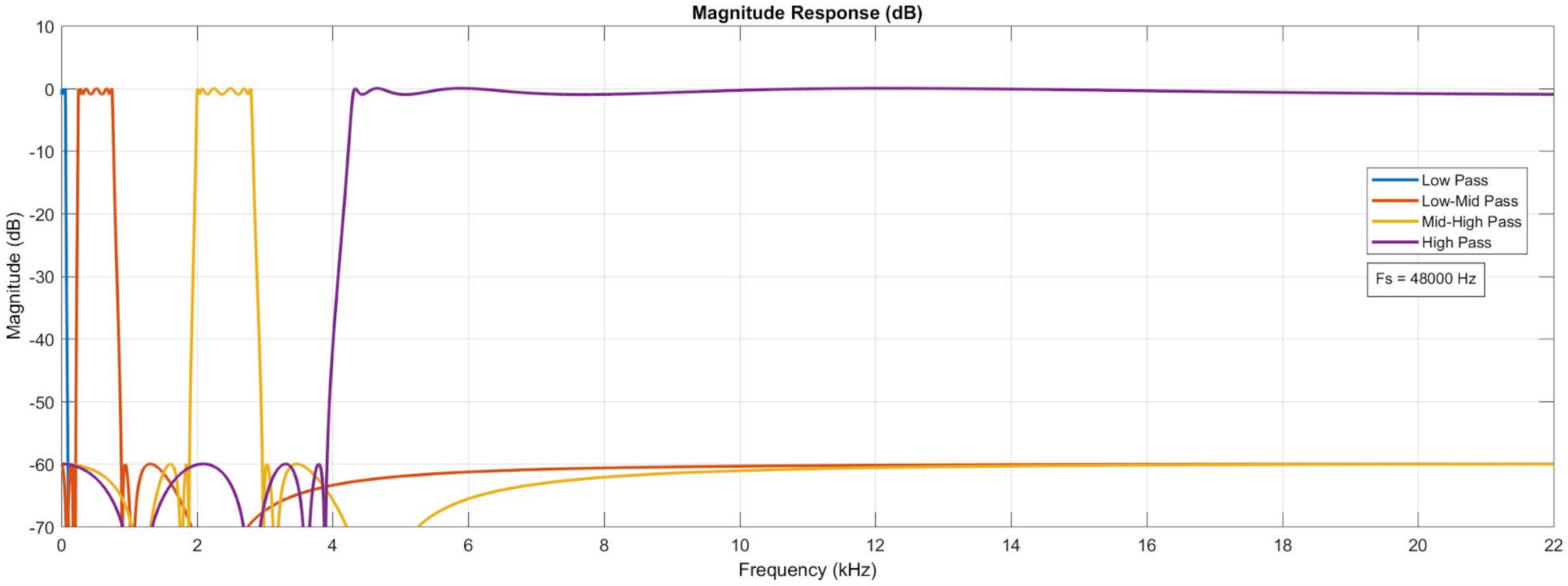
MID-HIGH PASS FILTER



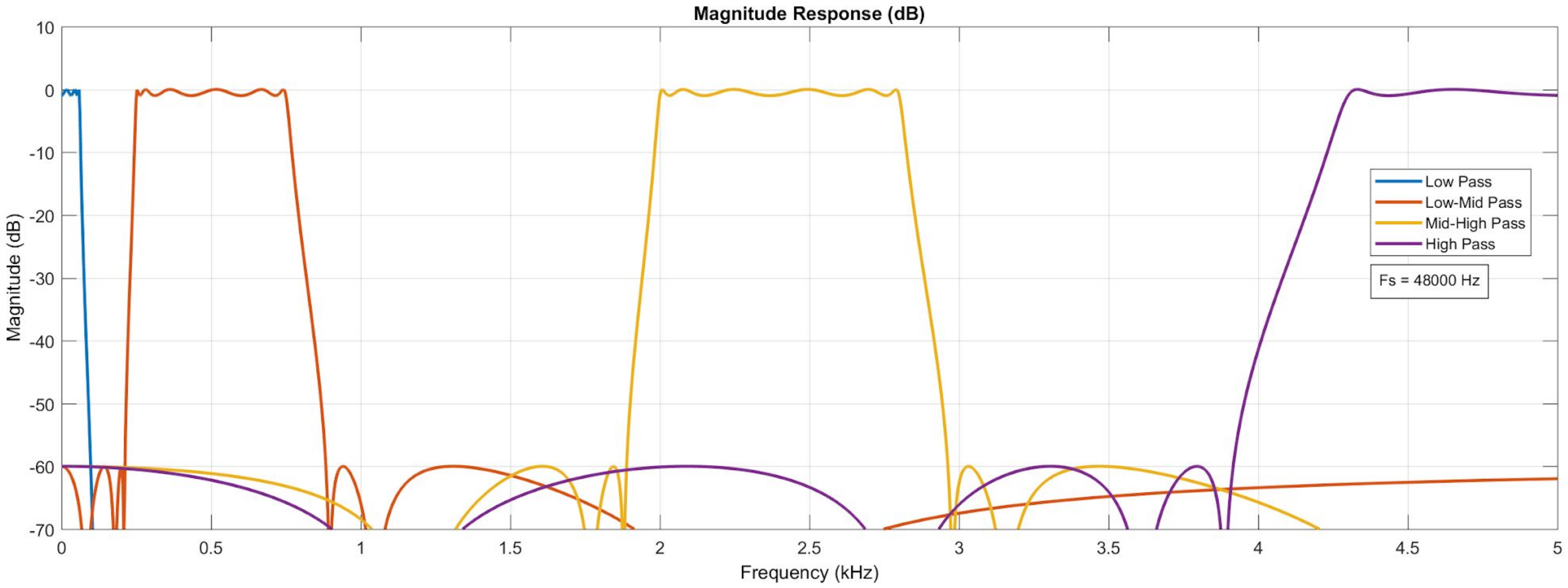
HIGH PASS FILTER



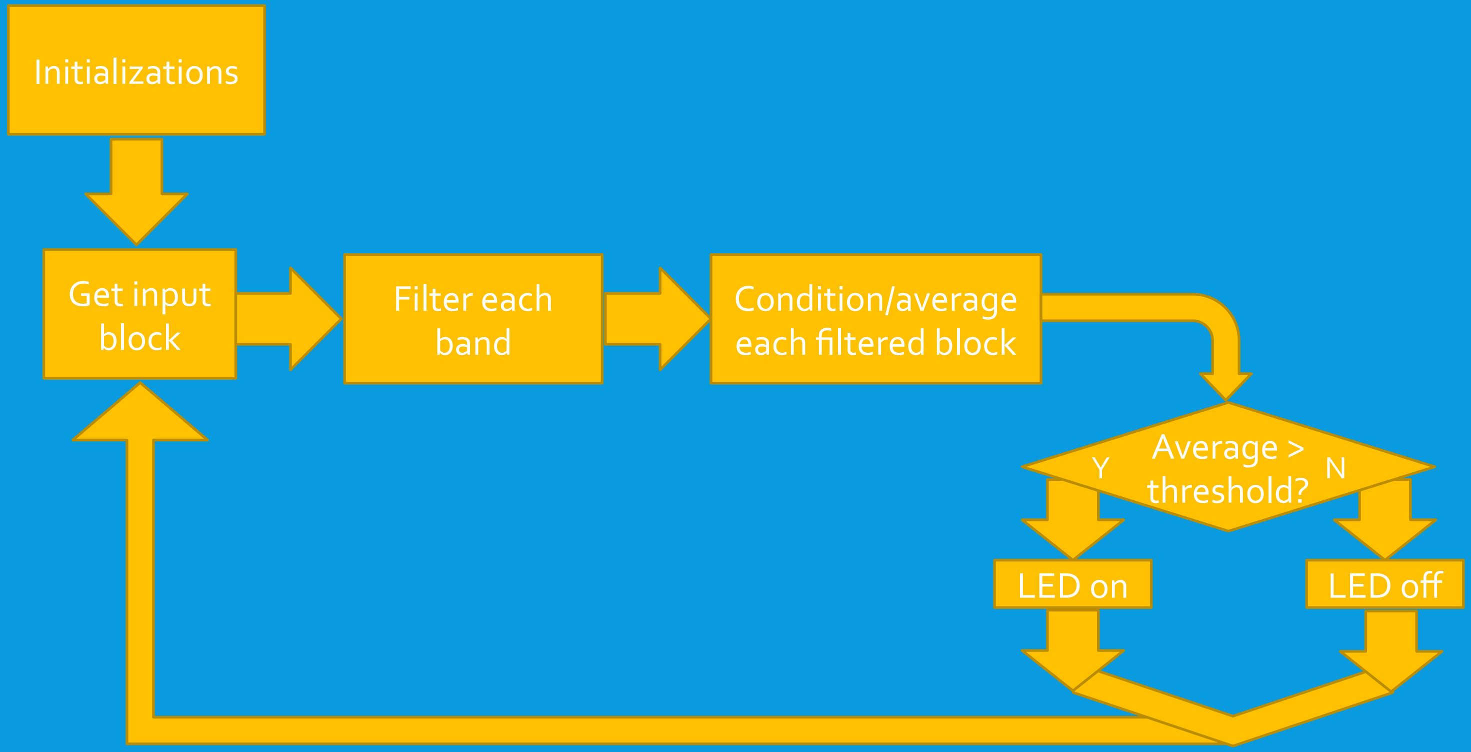
ALL FILTER RESPONSE



ALL FILTER RESPONSE



MAIN PROGRAM



ARM BIQUAD FILTER INITIALIZATION

```
void arm_biquad_cascade_df2T_init_f32 ( arm_biquad_cascade_df2T_instance_f32 *  S,
                                        uint8_t                                 numStages,
                                        float32_t *                             pCoeffs,
                                        float32_t *                             pState
                                      )
```

Parameters

[in,out]	***S**	points to an instance of the filter data structure.
[in]	**numStages**	number of 2nd order stages in the filter.
[in]	***pCoeffs**	points to the filter coefficients.
[in]	***pState**	points to the state buffer.

Returns

none

Coefficient and State Ordering:

The coefficients are stored in the array `pCoeffs` in the following order:

```
{b10, b11, b12, a11, a12, b20, b21, b22, a21, a22, ...}
```

where `b1x` and `a1x` are the coefficients for the first stage, `b2x` and `a2x` are the coefficients for the second stage, and so on. The `pCoeffs` array contains a total of `5*numStages` values.

The `pState` is a pointer to state array. Each Biquad stage has 2 state variables `d1`, and `d2`. The 2 state variables for stage 1 are first, then the 2 state variables for stage 2, and so on. The state array has a total length of `2*numStages` values. The state variables are updated after each block of data is processed; the coefficients are untouched.

References **arm_biquad_cascade_df2T_instance_f32::numStages**, **arm_biquad_cascade_df2T_instance_f32::pCoeffs**, and **arm_biquad_cascade_df2T_instance_f32::pState**.

ARM BIQUAD FILTERING FUNCTION

```
LOW_OPTIMIZATION_ENTER void arm_biquad_cascade_df2T_f32 ( const arm_biquad_cascade_df2T_instance_f32 *  S,
                                                          float32_t *                                   pSrc,
                                                          float32_t *                                   pDst,
                                                          uint32_t                                      blockSize
                                                        )
```

Parameters

[in]	***S**	points to an instance of the filter data structure.
[in]	***pSrc**	points to the block of input data.
[out]	***pDst**	points to the block of output data
[in]	**blockSize**	number of samples to process.

Returns

none.

References blockSize, arm_biquad_cascade_df2T_instance_f32::numStages, arm_biquad_cascade_df2T_instance_f32::pCoeffs, and arm_biquad_cascade_df2T_instance_f32::pState.

VECTOR SCALE, OFFSET, ABSOLUTE VALUE

```
void arm_scale_f32 ( float32_t *  pSrc,
                     float32_t    scale,
                     float32_t *  pDst,
                     uint32_t     blockSize
                   )
```

Parameters

[in]	***pSrc**	points to the input vector
[in]	**scale**	scale factor to be applied
[out]	***pDst**	points to the output vector
[in]	**blockSize**	number of samples in the vector

Returns
none.

References **blockSize**.

Referenced by **arm_dct4_f32()**, and **main()**.

```
void arm_offset_f32 ( float32_t *  pSrc,
                      float32_t    offset,
                      float32_t *  pDst,
                      uint32_t     blockSize
                    )
```

Parameters

[in]	***pSrc**	points to the input vector
[in]	**offset**	is the offset to be added
[out]	***pDst**	points to the output vector
[in]	**blockSize**	number of samples in the vector

Returns
none.

References **blockSize**.

```
void arm_abs_f32 ( float32_t *  pSrc,
                   float32_t *  pDst,
                   uint32_t     blockSize
                 )
```

Parameters

[in]	***pSrc**	points to the input buffer
[out]	***pDst**	points to the output buffer
[in]	**blockSize**	number of samples in each vector

Returns
none.

References **blockSize**.

Referenced by **main()**.

- These functions used to manipulate input audio block

VECTOR MEAN

```
void arm_mean_f32 ( float32_t *  pSrc,
                    uint32_t     blockSize,
                    float32_t *  pResult
                  )
```

Parameters

[in]	***pSrc**	points to the input vector
[in]	**blockSize**	length of the input vector
[out]	***pResult**	mean value returned here

Returns

none.

References **blockSize**.

Referenced by **main()**.

- After manipulating the input block, take average to get music intensity

QUESTIONS?

```c
while(1)    {
    getblock(input);

    DIGITAL_IO_SET();

    if (scale_input > 1)    arm_scale_f32(input,scale_input,input,nsamp);

    arm_biquad_cascade_df2T_f32(&filter_lo,input,output_lo,nsamp);
    arm_biquad_cascade_df2T_f32(&filter_lo_mid,input,output_lo_mid,nsamp);
    arm_biquad_cascade_df2T_f32(&filter_mid_hi,input,output_mid_hi,nsamp);
    arm_biquad_cascade_df2T_f32(&filter_hi,input,output_hi,nsamp);

    arm_scale_f32(output_lo,scale_lo,output_lo,nsamp);
    arm_scale_f32(output_lo_mid,scale_lo_mid,output_lo_mid,nsamp);
    arm_scale_f32(output_mid_hi,scale_mid_hi,output_mid_hi,nsamp);
    arm_scale_f32(output_hi,scale_hi,output_hi,nsamp);

    arm_abs_f32(output_lo,output_lo,nsamp);
    arm_abs_f32(output_lo_mid,output_lo_mid,nsamp);
    arm_abs_f32(output_mid_hi,output_mid_hi,nsamp);
    arm_abs_f32(output_hi,output_hi,nsamp);

    arm_offset_f32(output_lo,offset,output_lo,nsamp);
    arm_offset_f32(output_lo_mid,offset,output_lo_mid,nsamp);
    arm_offset_f32(output_mid_hi,offset,output_mid_hi,nsamp);
    arm_offset_f32(output_hi,offset,output_hi,nsamp);

    arm_mean_f32(output_lo,nsamp,&mean);
    if(mean > thresh_lo) LO_SET();
    else LO_RESET();
    arm_mean_f32(output_lo_mid,nsamp,&mean);
    if(mean > thresh_lo_mid) LO_MID_SET();
    else LO_MID_RESET();
    arm_mean_f32(output_mid_hi,nsamp,&mean);
    if(mean > thresh_mid_hi) MID_HI_SET();
    else MID_HI_RESET();
    arm_mean_f32(output_hi,nsamp,&mean);
    if(mean > thresh_hi) HI_SET();
    else HI_RESET();

    if (KeyPressed) {
        KeyPressed = RESET;
        scale_input *= increment;
    }

    DIGITAL_IO_RESET();
}
```