"A Digital, real time Audio Frequency Spectrum Analyzer for audio devices"

Microchip 16-bit Embedded Control Design Contest

Registration Number: MT1750

"Eligible Parts": dsPIC30F6012A
"Bonus Parts" : MCP6022

Spectrum analyzers are employed in most of the modern signal processing systems for measuring the distribution of signal energy in frequency. An audio spectrum analyzer is used for measurements in the audible frequency spectrum (from 0 to 20000 Hz) and can be a very powerful tool for keeping a studio well tuned. A Digital, real-time Audio Frequency Spectrum analyzer circuit for audio devices is presented in this project. This circuit aims to professionals or hobbyists who would like to embed it in an audio device or use it as a stand-alone unit.

The Digital, real-time Audio Spectrum analyzer circuit for audio devices can be connected to any audio device. It accepts an analog audio signal as input. It digitizes and processes this audio signal using a DSP. It computes the distribution of the audio signal energy to 20 specific frequency bands and displays it on a 20x20 LED display in real time. The circuit is based on a dsPIC30F6012A, which is chosen for a feature set which matches the requirements of this project. The DSP engine, the 12-bit A/D converter, the timer3 timer, 42 I/O ports and the large internal data memory of the dsPIC30F6012A are used in this project. A quad of dual MCP6022 op-amps for anti-aliasing filtering and gain are also used in the analog section of the circuit.

The built-in 12-bit A/D converter, the timer3 timer (for ADC timing) and a quad of dual MCP6022 op-amps for anti-aliasing filtering and gain, digitise the incoming analog audio signal. The digital representation of the analog audio signal is processed in real time using the processor's DSP engine. The DSP processing includes Windowing, Fast Fourier Transform and Square-magnitude (power) computation. This way, a discrete digital representation of the signal power in frequency is obtained. Furthermore, a Filter-Banking algorithm is applied to the obtained discrete spectral-power, resulting in the implementation of a 20 non uniform-band parallel analysis filter- bank. This filter bank is used to decompose the audio signal power into a set of 20 sub band signals. Each sub-band signal represents the total audio signal power in each of the 20 spectral bands and is displayed on a 20-LED bar graph. There are 20 bar graphs (one for each spectral band), each one consisting of 20 LEDs and, as such, a 20x20 Led display is used.

The main goals of the circuit are as follows:

- It is a low-cost circuit and can be embedded in any audio device.
- It uses a simple, low cost 20x20 LED display
- It supports four display modes ("0-Climbing-Bars", "1-Bars", "2-Climbing-Bars+Rain", "3-Bars+Peaks")
- It provides a very good frequency and amplitude resolution (0.431 octaves from 31 to 15000Hz and 1.3 db from 0 to –28db.
- Can be powered from a single 5V power supply.

- Offers simultaneous monitoring of the entire audio signal band in real time.
- Uses a single processor both for DSP and display controlling (there is no need for a separate display controller).

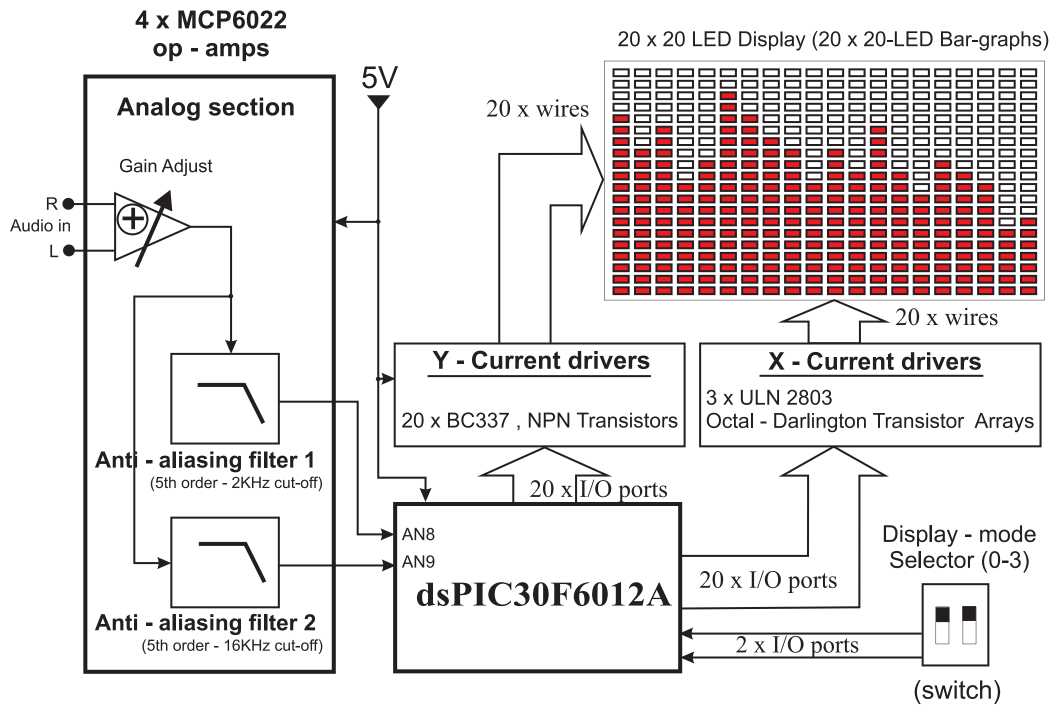

Figure 1. Overall System Block Diagram (The system consists of 2 boards, the main-board and the display-board. All units are located in the main board, except for the display)

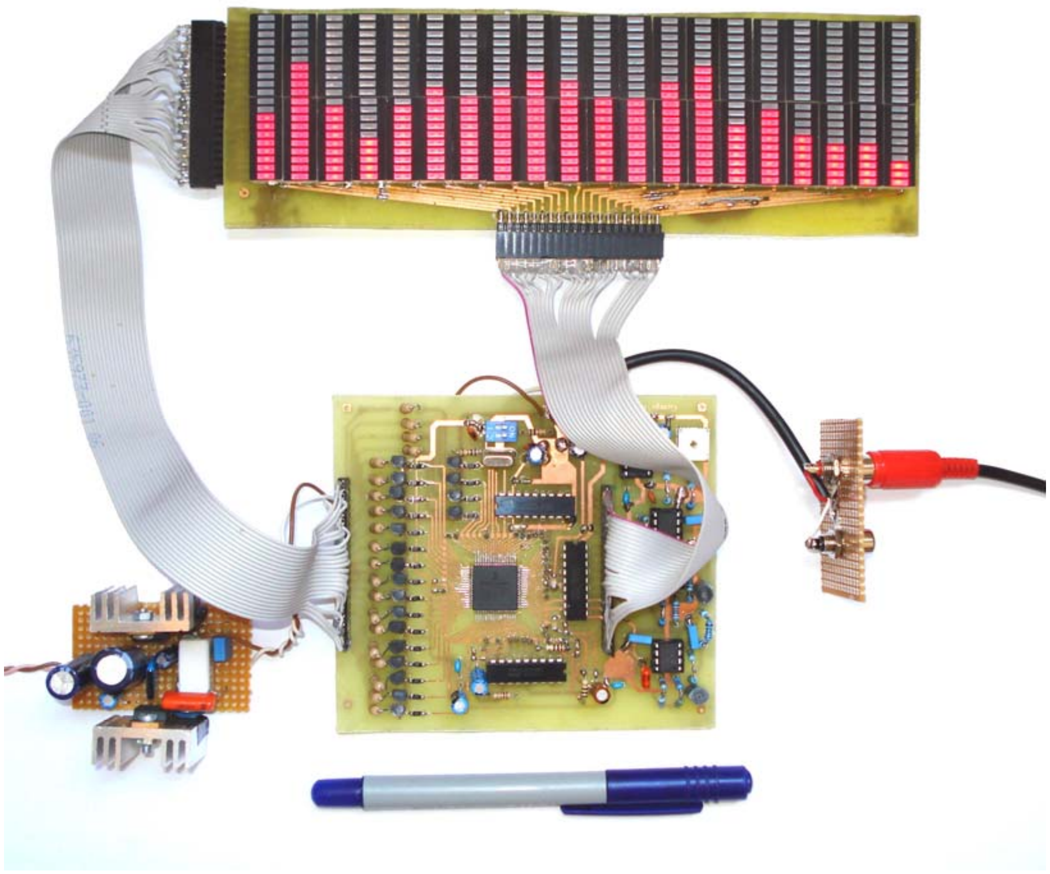

Photo 1. The complete assembly (The main board is located at the center of the photo. The display is at the top. A 5V/1.5A power supply unit is at the left – not included in the project)

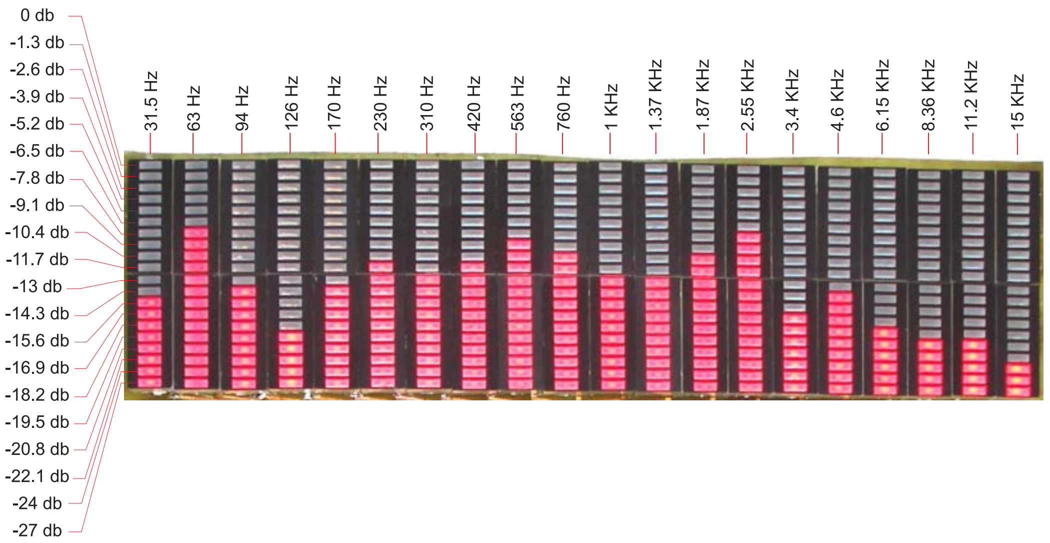

Photo 2. The distribution of the audio signal energy is plotted in 20 bar – graphs. 4 display-modes are supported (This photo was taken during normal operation at "Bars" display mode)

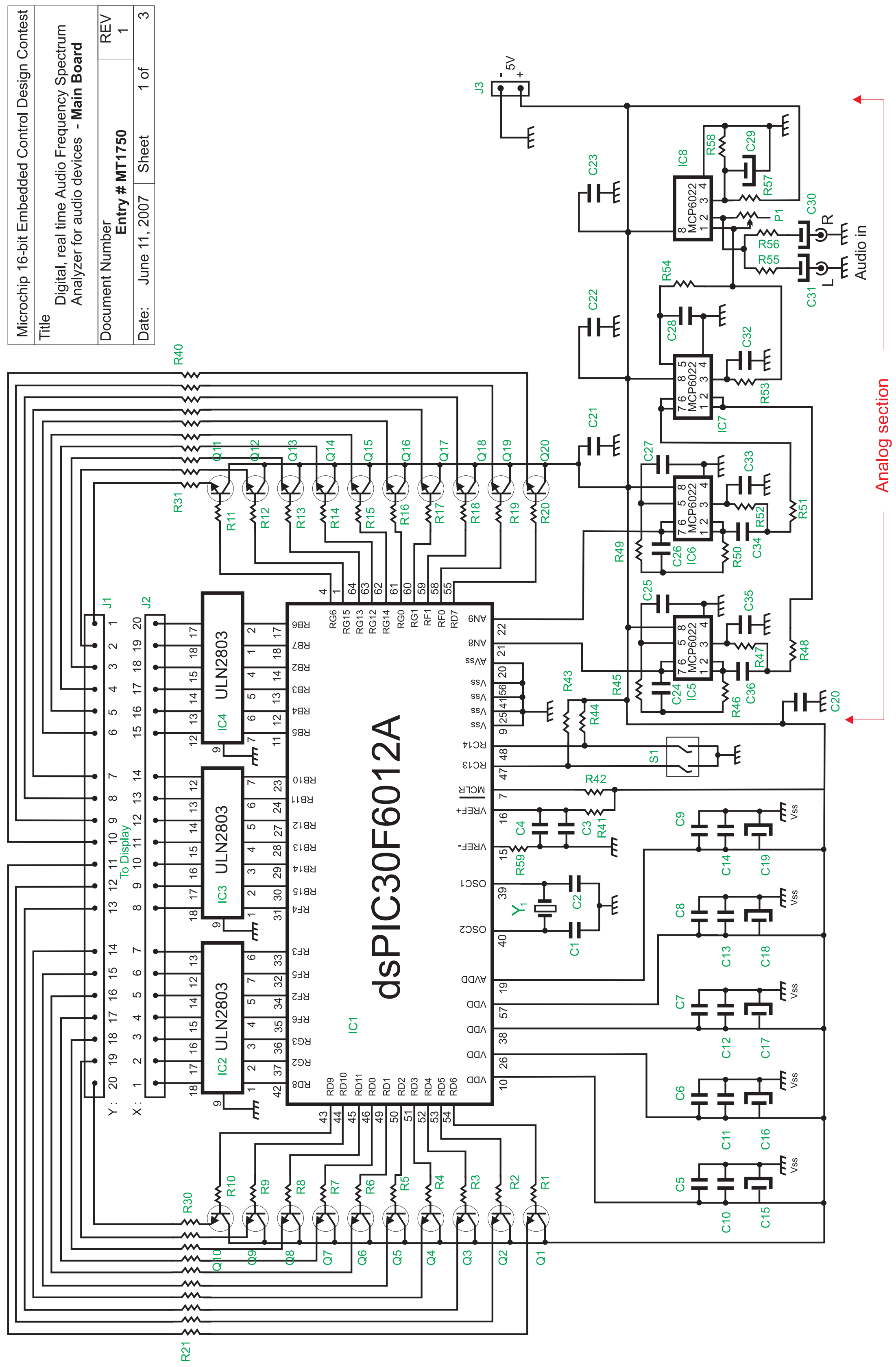

Figure 2. Main – Board Schematic

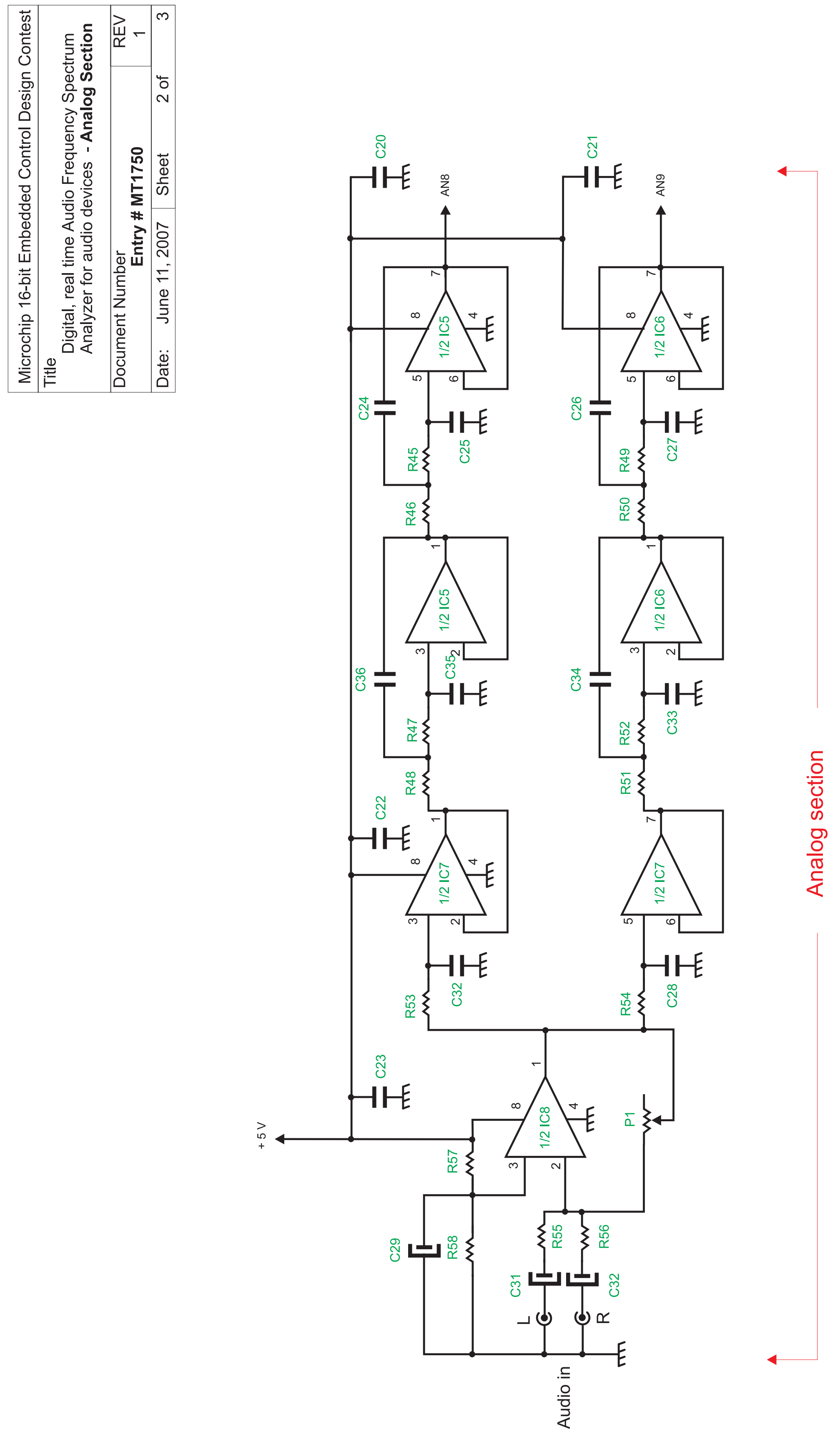

Figure 3. Analog – Section Schematic

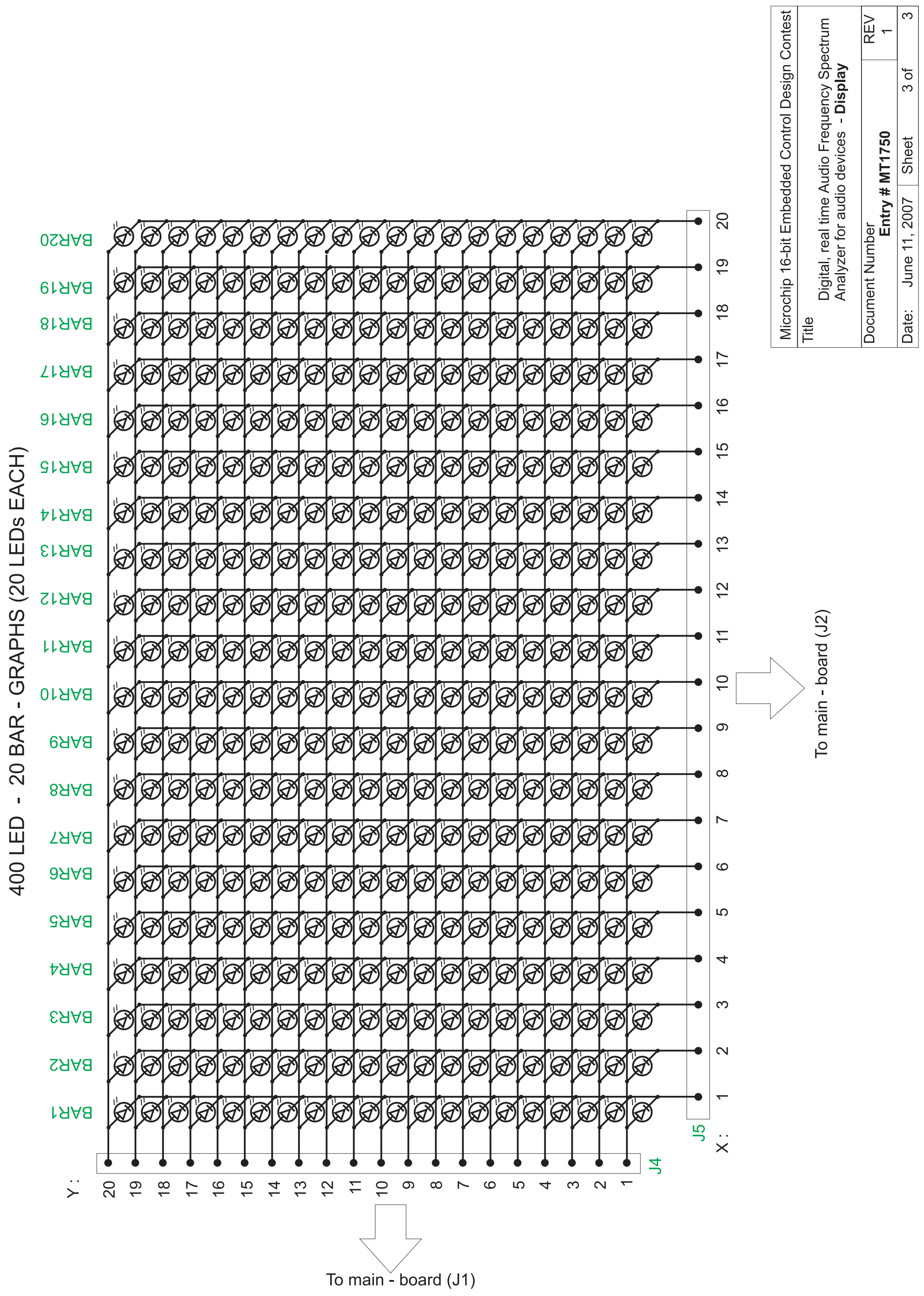

Figure 4. Display Schematic

Microchip 16-bit Embedded Control Design Contest Entry # **MT1750**

Digital, real-time Audio Frequency Spectrum Analyzer for audio devices

Materials list

Qty	Description	Ref Des
1	Microchip dsPIC30F6012A	IC1
3	ULN2803 – Darlington transistor array	IC2, IC3, IC4
4	Microchip MCP6022 dual op-amp	IC5, IC6, IC7, IC8
20	BC337 NPN transistor	Q1, Q2, Q3, ... Q20
20	20 – LED Bar – graph *	BAR1, BAR2, …BAR20
1	14.74560 MHz Crystal	Y1
20	1K SMD resistor	R1, R2, R3 … R20
20	0 – 180 Ohm ** 1/2W resistor	R21, R22 … R40
3	10K 5%, 1/4W resistor	R42, R43, R44
1	6.98K 1%, 1/4W resistor	R45
1	3.48K 1%, 1/4W resistor	R46
1	18.7K 1%, 1/4W resistor	R47
1	8.87K 1%, 1/4W resistor	R48
1	6.49K 1%, 1/4W resistor	R49
1	3.09K 1%, 1/4W resistor	R50
1	7.87K 1%, 1/4W resistor	R51
1	17.4K 1%, 1/4W resistor	R52
1	16.9K 1%, 1/4W resistor	R53
1	15.8K 1%, 1/4W resistor	R54
2	4.7K 5%, 1/4W resistor	R55, R56
1	47K 5%, 1/4W resistor	R57, R58
2	10 Ohm 5%, 1/4W resistor	R41, R59
1	100K trimmer	P1
2	22pF ceramic capacitor	C1, C2
6	10nF SMD capacitor	C4, C5, C6, C7, C8, C9
6	100nF SMD capacitor	C3, C10, C11 ... C14
4	100nF multi-layer capacitor	C20, C21, C22, C23
1	56nF mica capacitor	C24
3	4.7nF mica capacitor	C25, C32, C35
2	8.2nF multi-layer capacitor	C26, C36
3	680pF ceramic capacitor	C27, C28, C33
6	1uF / 10V electrolytic capacitor	C29, C15, C16 … C19
2	10uF / 10V electrolytic capacitor	C30, C31
1	1.2nF multi-layer capacitor	C34
1	Dual dip-switch	S1
4	20-pin male connector	J1, J2, J4, J5
1	2-pin male connector	J3

Note: Capacitors are 5%, unless specified.

* Any LED type can be used

** The value of R21-40 resistors depends on the LED type (0 ohm used in this project for 18mA, common brightness LEDs)

The code has been developed in C, using Microchip’s C30 Compiler. A brief sample of the code is listed below (figure 5). This sample contains the main routine of the program, which uses interrupt driven state-machine logic. The main routine is interrupt driven from the A/D converter. The 12 bit A/D converter acquires NUMSAMP samples from AN8 and AN9 pins at a sampling rate of 7998.6981 Hz and 87252.071 Hz respectively and puts the A/D conversion results to the complex inputSignal[] array. The inputSignal[] array is processed by means of Windowing, Fast Fourier Transform and Square-magnitude (power) computation. The discrete power spectrum of the audio signal is stored in the Power_Vector[] array. A filter banking algorithm is applied to the Power_Vector[] and the results are stored to the real Bar[] array. The Bar[] array is finally displayed in 1 of the 4 available display – modes depending on the S1 switch select.

```
int main (void)
{

  // Initialization
  initializeIO();
  initialize_rest();

      while (1)              //Loop Endlessly
      {                      //from this point on. (interrupt driven state-machine)

          if (SamplesReadyFlag==1)  // NUMSAMP samples have been acquired (sampling rate:7998.6981 Hz)
             {

               /*Apply HammingFactors to the input sample array and store the resulting windowed vector in place*/
               VectorMultiply (NUMSAMP*2,&inputSignal[0].real,&inputSignal[0].real,&HammingFactors[0]);

               /* Perform FFT operation */
              FFTComplexIP (LOG2_BLOCK_LENGTH, &inputSignal[0], &twiddleFactors[0], COEFFS_IN_DATA);

              /* Store output samples in bit-reversed order of their addresses */
              BitReverseComplex (LOG2_BLOCK_LENGTH, &inputSignal[0]);

              /* Compute the square magnitude of the complex FFT output array so we have a Real output vector */
              SquareMagnitudeCplx(NUMSAMP, &inputSignal[0], &Power_Vector[0]);

              Filter_Bank(SamplesReadyFlag);       // Find the Bar[0-9] samples which will be displayed in a 20-bars led display

              p_inputSignal = &inputSignal[0];         //Initialize pointer for the A/D convertion
              SamplesReadyFlag = 2;                    //Initialize A/D converter's Samples-Ready flag for 87252.071Hz sampling rate
               ADC_Init(2);                            //Initialize the A/D converter for 87252.071 Hz sampling rate


              }

          if (SamplesReadyFlag==3)  // NUMSAMP samples have been acquired (sampling rate:87252.071 Hz)
              {

               /*Apply HammingFactors to the  input sample array  and store the resulting windowed vector in place*/
               VectorMultiply (NUMSAMP*2,&inputSignal[0].real,&inputSignal[0].real,&HammingFactors[0]);

              /* Perform FFT operation */
              FFTComplexIP (LOG2_BLOCK_LENGTH, &inputSignal[0], &twiddleFactors[0], COEFFS_IN_DATA);

              /* Store output samples in bit-reversed order of their addresses */
              BitReverseComplex (LOG2_BLOCK_LENGTH, &inputSignal[0]);

             /* Compute the square magnitude of the complex FFT output array so we have a Real output vector */
             SquareMagnitudeCplx(NUMSAMP, &inputSignal[0], &Power_Vector[0]);

               Filter_Bank(SamplesReadyFlag);         // Find the Bar[10-19] samples which will be displayed in a 20-bars led display

               p_inputSignal = &inputSignal[0];           //Initialize pointer for the A/D convertion
               SamplesReadyFlag = 0;                      //Initialize A/D converter's Samples-Ready flag for 7998.6981 Hz sampling rate
               ADC_Init(1);                               //Initialize the A/D converter for 7998.6981 Hz sampling rate

              }


          display (display_mode());                          // Dislay routine (for display_mode 0, 1, 2, 3)

      }
    return 0;

}
```

Figure 5. The main routine