

Digital audio filter for short wave CW listening

Submitted by Gabriel Patulea for the Van Drebbel contest co-sponsored by NEC Electronics America and EDN Magazine. – December 2004

1) Description of the application

The application presented herein is an adjustable bandwidth digital filter that employs the feature rich 78F0338 NEC microcontroller installed on the EV0338 Micro Board. The main purpose of the filter is to increase the readability of the CW code signals received by short wave transceivers. Usually the HF bands are very noisy and affected by interference. By narrowing the audio bandwidth of the receiver it is possible to remove a significant portion of the unwanted noise and improve the signal to noise ratio of the received signal.

Some of the well known advantages of digital filters are flexibility and stability; one can easily change the filter characteristics by simply changing a few constants in the software. Although there are complex issues associated with digital filters, their performance is hardly imagined to be achieved by their analog counterparts, either passive or OpAmp based active filters. In this respect, the application could be seen as an example of how a digital filter could be built based on the 78F0338 microcontroller.

2) Design considerations

While designing the filter the first element to be taken into account was the central frequency of the filter. Then, depending on the receiving conditions, it is convenient for the operator to be able to modify the bandwidth of the filter on the fly. Usually the operator starts with a wider bandwidth. This choice helps tuning the receiver on the appropriate frequency. As the signal gets into the filter's pass band, the bandwidth can be progressively made narrower in order to keep the signal within the pass-band limits and cleaned up from the adjacent noise and interference. Taking into account the considerations above the following parameters have been chosen: Central Frequency = 750Hz, BW1 = 500Hz, BW2 = 250Hz and BW3 = 100Hz. The central frequency is fixed, but the bandwidth can be adjusted while the filter is in operation.

Digital filters could be divided in 2 main categories: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. Although the FIRs are well known for their inherent stability and phase linearity, they are computationally intensive and more difficult to implement in a general purpose, low speed microcontroller. That is why an IIR configuration has been chosen for the application. The main advantage is that less real time computation is required, and with careful simulation the other various challenges could be overcome as well.

The filter type chosen for the application was a second order type 1 Chebychev with a pass band ripple of 3dB. The choice was based on the steeper roll-off curve of the filter in the proximity of the pass band compared to the Butterworth (maximally flat pass-band)

filter. The design and simulation of the filter were performed using the Matlab package from Mathworks.

Another important step in the design was the choice of the sampling frequency. The minimum allowable sampling frequency is dictated by the Nyquist's sampling theorem. Based on that, and assuming ideal brick-wall filter characteristics, the minimum theoretical sampling frequency would have been $2 \cdot (750 + 500/2) = 2\text{KHz}$, but over-sampling is usually required because of the finite roll-off of the anti-alias filter. On the other hand, the microcontroller should be able to perform the required processing in the time interval between 2 consecutive samples. This requirement puts a high limit on the sampling frequency. Through simulation and optimization of the code the author has come to the conclusion that 5KHz is an acceptable choice for the sampling frequency. The decision was also based on the availability of the 5MHz crystal that is by default installed on the EV0338 Micro Board. The chosen sampling frequency is a trade-off that meets both the sampling theorem requirement as well as it allows for sufficient time for the CPU to process the signal between consecutive samples.

As with any sampled system the analog input signal has to be bandwidth limited. In order to achieve the bandwidth limiting, a low pass filter has been attached to the input of the A/D converter. This filter is also known as the anti-alias filter. It has been built around the TL082 OpAmp and the roll-off characteristic has 18dB/octave based on a third order Chebychev filter that has been implemented. The cutoff frequency has been chosen around 1KHz. The frequency components at half the sampling frequency (2.5KHz) are attenuated in excess of 30dB.

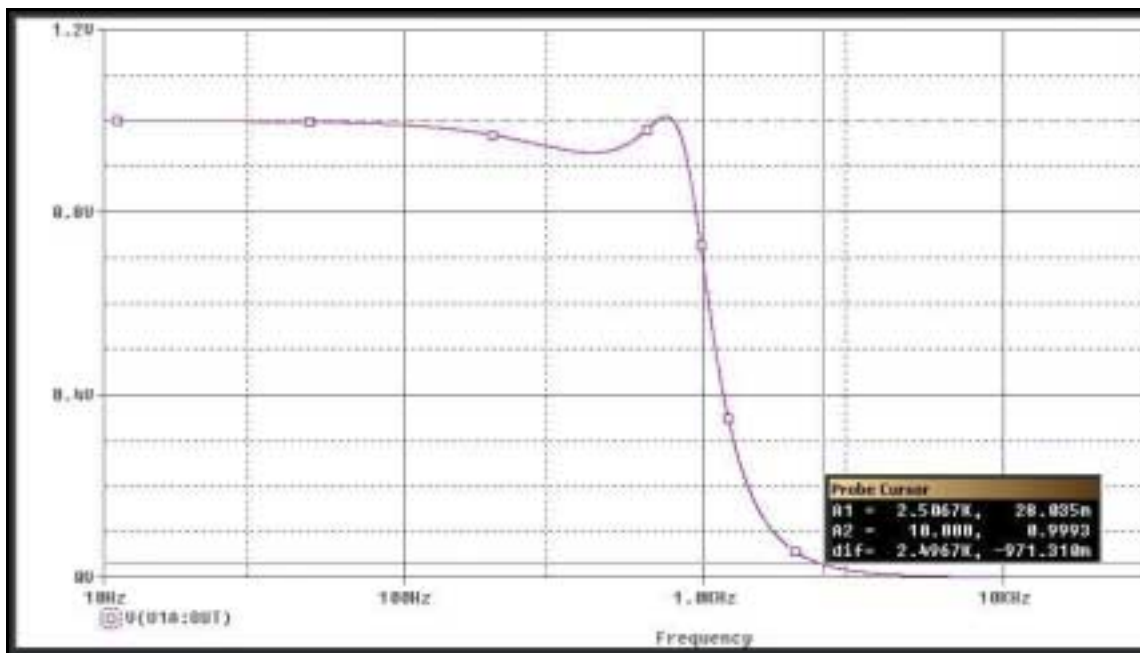


Fig. 1 Transfer characteristic of the anti-alias and reconstruction filters

The digital filter output samples are sent to the internal D/A converter provided inside the microcontroller. The output signal still contains a large amount of harmonics of the sampling frequency which might be perceived as noise and distortions. A second analog filter has been provided at the output that constitutes the so called reconstruction filter. The output filter has the same configuration as the anti-alias filter, but its purpose is different in the signal processing sequence.

3) Digital Filter Design and Algorithm

As previously mentioned, the digital filter has been designed using the Signal Processing Toolbox from Matlab, a very popular math oriented, general purpose package.

The general form of a digital filter formula expressed in terms of Z transform is:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 \cdot z^{-1} + \dots + b_n \cdot z^{-n}}{1 + a_1 \cdot z^{-1} + \dots + a_n \cdot z^{-n}} \quad (\text{Eq.1})$$

For practical reasons, in order to reduce the calculation errors the n-th order form needs to be converted into a product of first and second order factors. The conversion also groups together the poles and zeros that are close to each other. Therefore, the formula that is usually implemented in software is similar to:

$$H(z) = g \cdot \prod_{k=1}^L \frac{b_{0k} + b_{1k} \cdot z^{-1} + b_{2k} \cdot z^{-2}}{1 + a_{1k} \cdot z^{-1} + a_{2k} \cdot z^{-2}} \quad (\text{Eq.2})$$

In the application presented herein, a second order filter has been implemented, therefore L=1. From the Z domain transform formula above, the time domain recursive equation that needs to be performed by the software will become:

$$y(n) = g \cdot [b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2)] - [a_1 \cdot y(n-1) + a_2 \cdot y(n-2)] \quad (\text{Eq.3})$$

where g is the gain of the filter section, a_i and b_i are the filter coefficients and $y(n)$, $x(n)$, $y(n-1)$, $x(n-1)$,... and so on are the current and the delayed samples of the output (y) signal and the input (x) signal respectively.

The Matlab script (.m file) that has been used for design and simulation of the filter is provided on the CD. The script generated the coefficients a_i and b_i for each filter bandwidth. In order to test the behavior of the filter, a chirp signal has been fed to each configuration of the digital filter. The chirp signal is essentially a constant magnitude sine wave whose frequency is continuously increasing from a minimum to a maximum value. Plotting the output samples versus time, one can see the frequency characteristic of the filter. There are other methods to plot the frequency characteristic of a digital filter, such as taking the Fourier transform of the impulse response of the filter. In this case, using the chirp signal, the filter algorithm was tested step by step and compared to the results obtained in the microcontroller simulation. Instability and overflow conditions were tested as well. The results of the filter simulations in Matlab can be seen in the picture below.

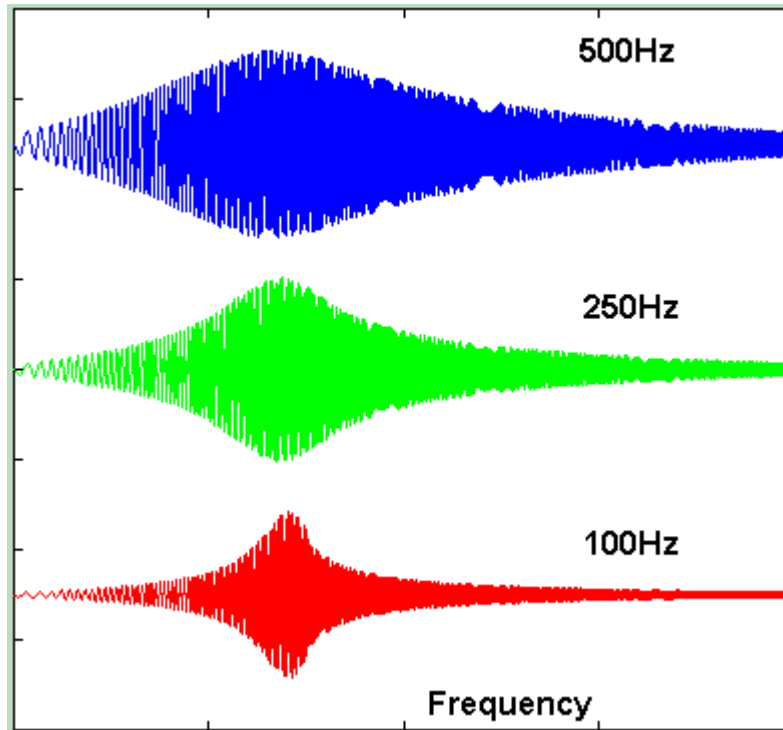


Fig. 2 Magnitude vs. frequency plots of the Matlab filter simulations for each filter bandwidth: 500Hz, 250Hz and 100Hz

Inspecting the recursive formula, it can be seen that for both the input and the output signals the last three samples need to be stored in the memory in order for the computation to be possible.

Since neither the C language nor the assembler language of the processor allows for circular buffers to be implemented, the double delay line technique has been used. This technique consists of using twice the minimum amount of memory buffering required and double-storing the samples at two different locations.

A possible code example that illustrates the double delay line update for the current i (the buffer index) value could be seen below.

```
x(i-3)=x(i)=ADC_result;           // update 2 input samples
y(i-3)=y(i)=g*(b0*x(i)+b1*x(i-1)+b1*x(i-2)) // update 2 output samples
          -(a1*y(i-1)+a2*y(i-2));
```

For processing speed considerations, arrays and pointer arithmetic have not been used in the actual digital filter routine, but the algorithm implementation is essentially the same.

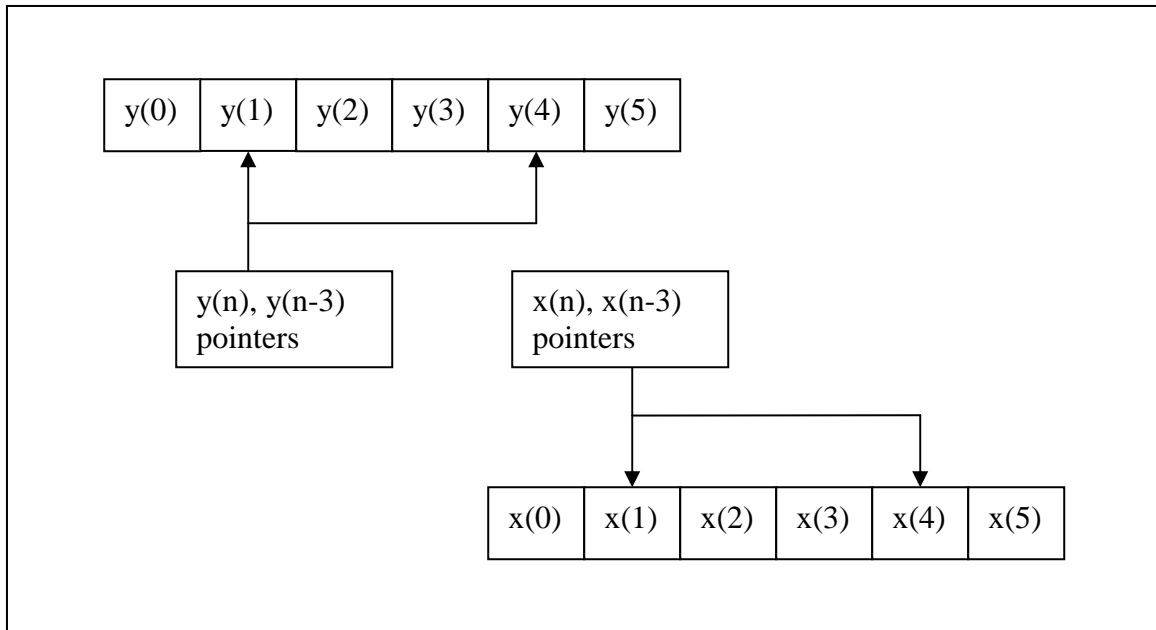


Fig. 3 Double delay line technique

4) Specific functions used in the microcontroller

The following functions and peripherals have been used in the microcontroller:

4.1) The A/D converter. Its purpose is to generate a sampled digitized replica of the analog signal applied at its input.

4.2) Timer 0. Timer 0 generates the precise time intervals at which the sampling process must take place. The timer is configured in such a way that allows it to generate the 5KHz sampling frequency.

4.3) The CPU's multiplier. The K0 CPU performs the actual computation of the samples as well as input switches reading, peripheral configuration and LCD display update according to the current status of the program. It is important to note that the multiply (MULU) instruction is highly required in a signal processing application. The availability of the hardware multiplier increases the speed of the algorithm execution.

4.4) The D/A converter. The results of the filter computation are being sent out to the D/A converter which returns the signal to the analog world.

4.5) The LCD controller. The bandwidth of the currently operating filter is being displayed on the LCD. The alphanumeric LCD display module is connected to the LCD controller located inside the 78F0338 device.

4.6) Parallel port P0. The purpose of the parallel port is to act as an input device. The switches SW2, SW3 and SW4 are being used to select the desired bandwidth of the filter. The status of each switch is being read periodically through the parallel port.

5) Block diagram of the application

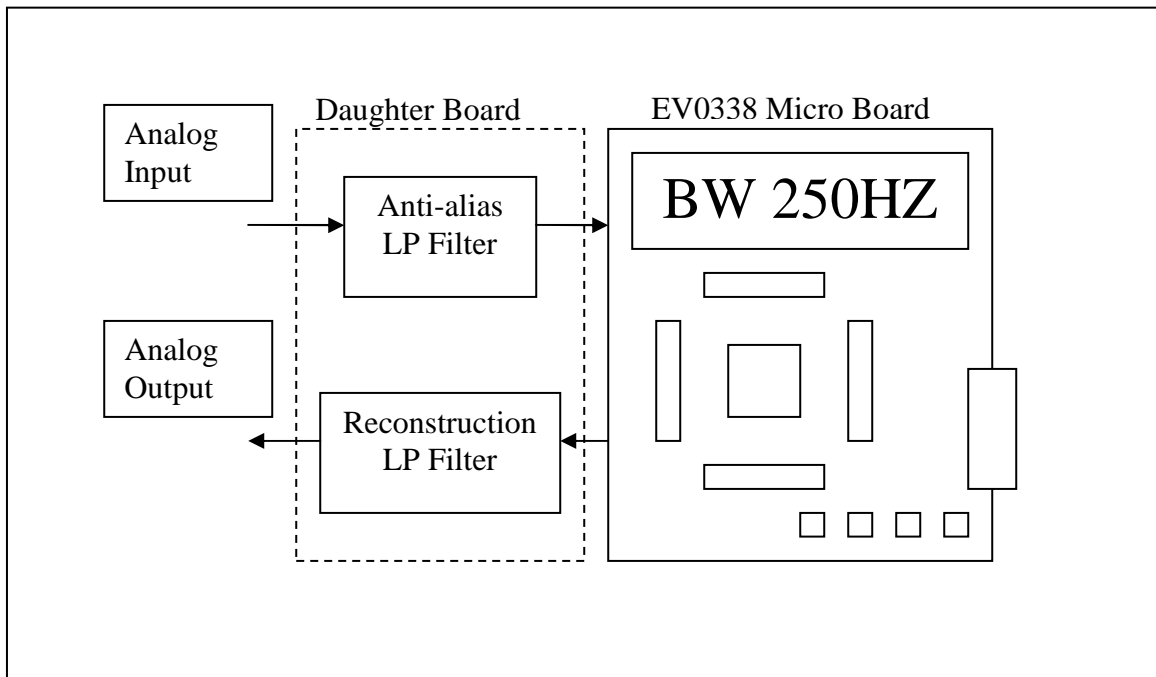


Fig. 4 Block diagram of the application

Figure 3 presents the block diagram and the signal flow of the application. The audio analog signal source is applied to the input of the anti alias filter. The output of the anti alias filter is connected to the ANI0/P10 input of the microcontroller. Upon processing, the output AO0/P120 of the D/A converter is supplied to the reconstruction filter. Further, the reconstruction filter provides the filtered analog signal at its output.

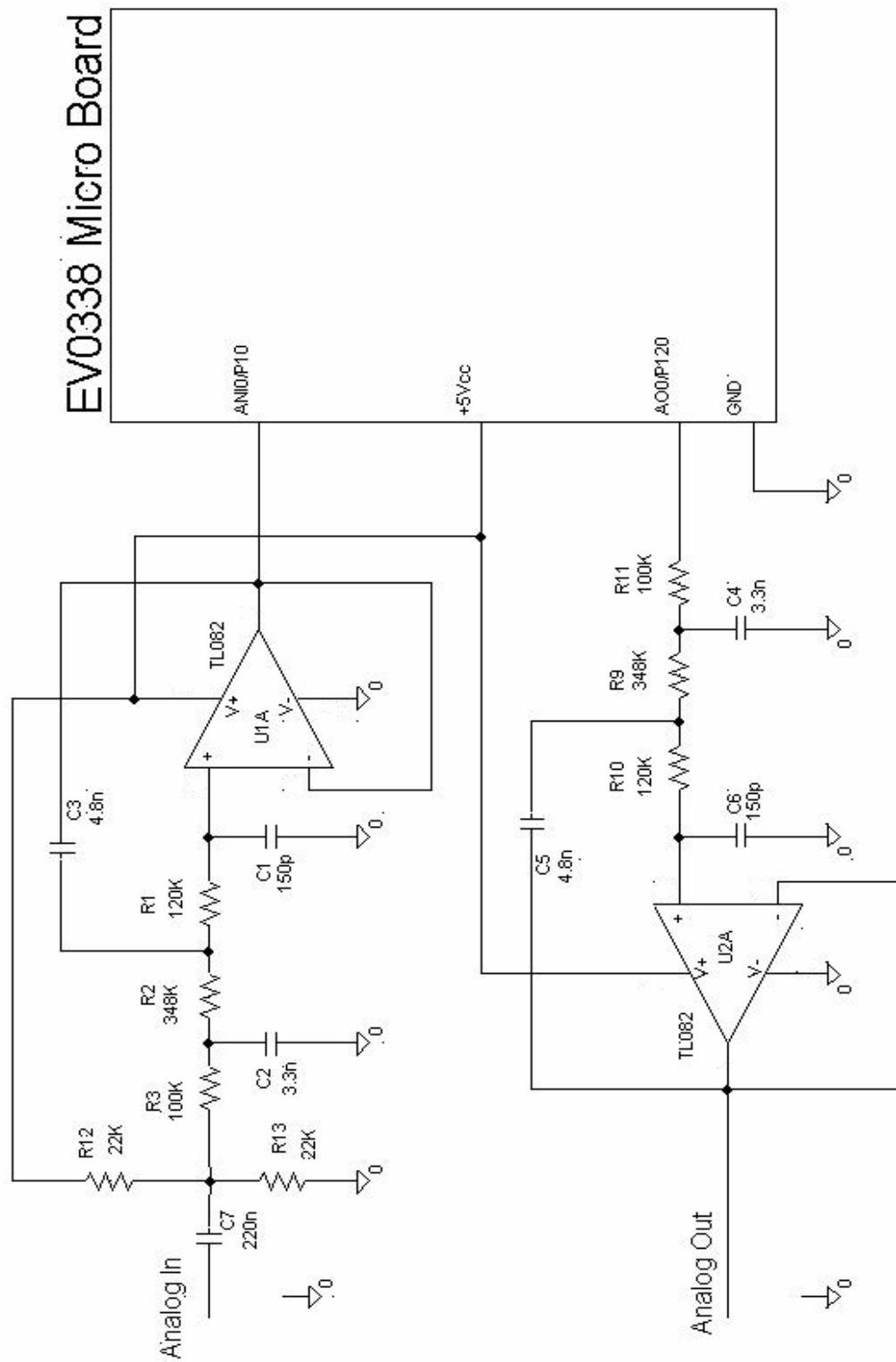


Fig. 5 Schematic diagram

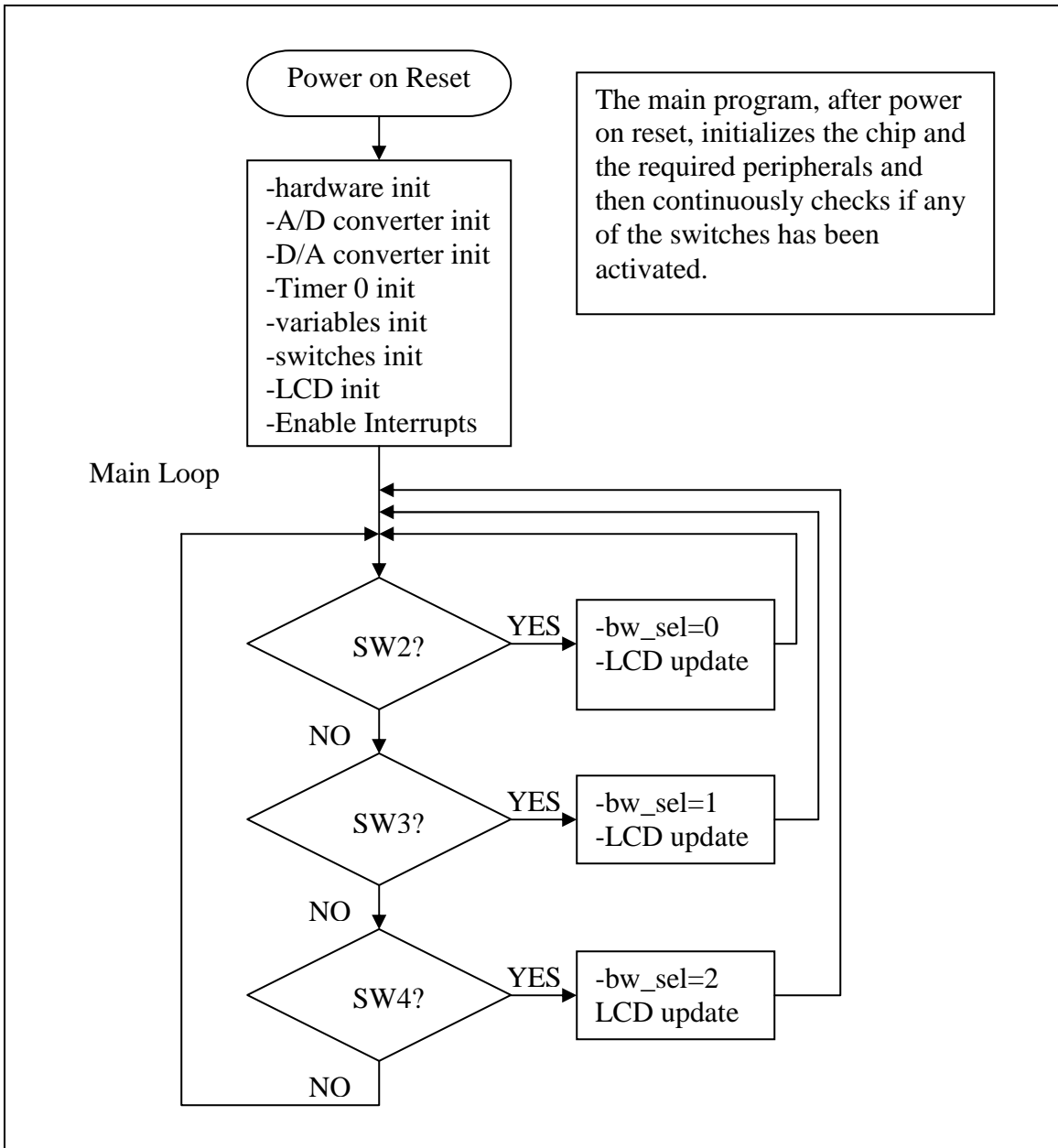


Fig. 6 Main program flowchart

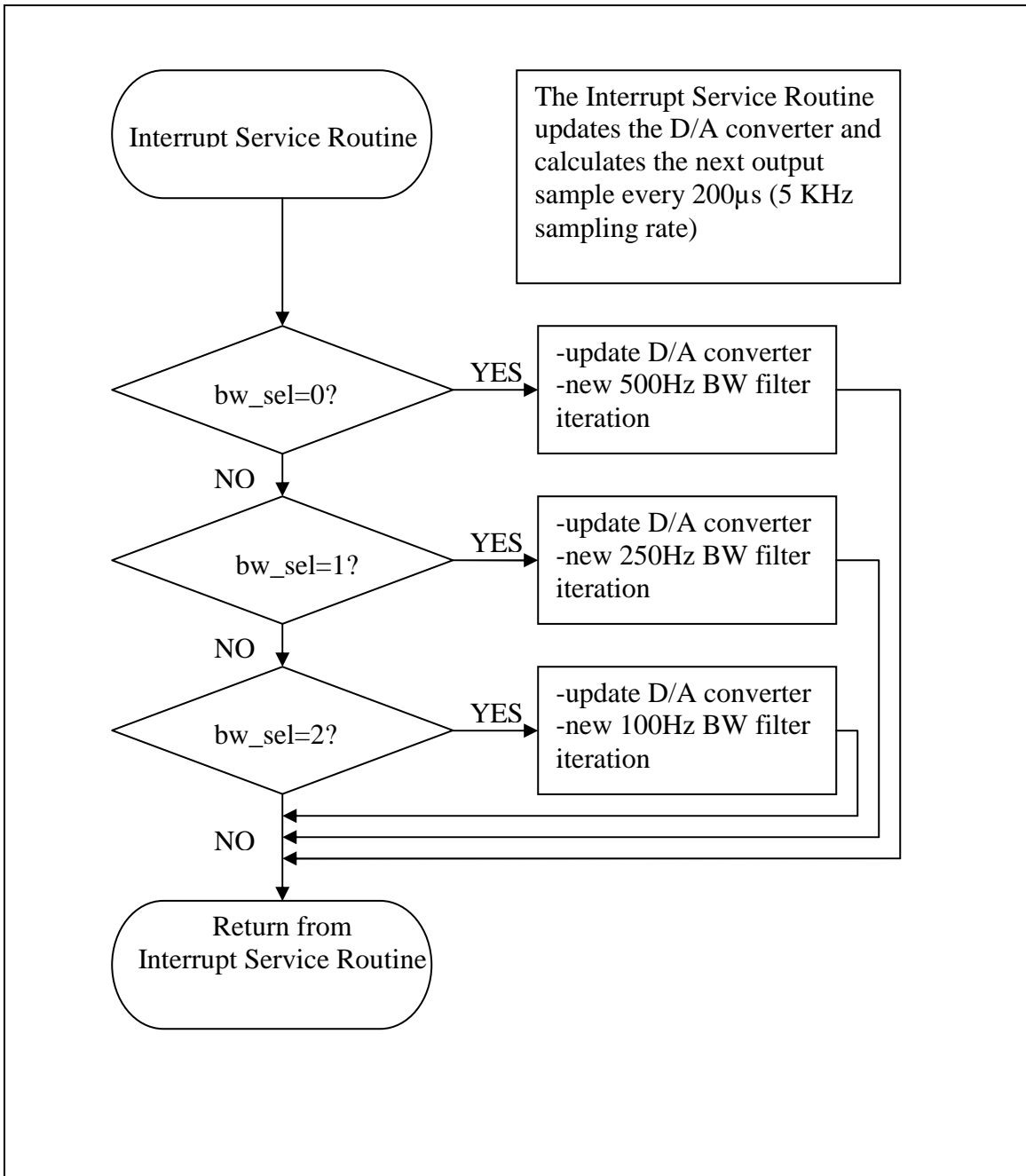


Fig. 7 Interrupt Service Routine flowchart

Conclusion and comments

The application has demonstrated the ability of the 78F0338 NEC microcontroller to perform digital signal processing even at audio frequency range, although the device was not primarily designed for this purpose.

The author has determined that the amount of time required for one iteration of the digital filter is around 185us. Adding 10% time margin, the sampling interval becomes 200us, i.e. 5KHz sampling frequency. The microprocessor on the EV0338 Micro Board has been originally provided with a 5MHz crystal. It is expected that the performance of the filter could be improved (such as increase the filter order) if the crystal was 10MHz.

Processing at even slower sampling rates (if and where appropriate) could benefit from increased processing resolution, by increasing the number of bytes allocated for number representation. The output signal to noise ratio will be improved at the expense of a longer processing time required between consecutive samples.

In order to fully benefit from the trade offs applied in the current application, it is recommended that the input signal uses as much as possible from the A/D converter range. Low signal amplitudes might be affected by the reduced precision used for the sample storage in order to increase processing speed.