RTDSP Lab 4

Yong Wen Chua (ywc110) & Ryan Savitski (rs5010)

1 Matlab Filter Design

1.1 Matlab Code

Based on the specification given, the following Matlab code was used to generate the filter:

```
clear;

rp = 0.4;                       % passband ripple
rs = 50;                        % stopband ripple
f = [0.065 0.1125 0.5625 0.625]; % Normalised frequencies
a = [0 1 0];                    % amplitude
fs = 8000;                      % sampling frequency

% calculate deviation
dev = [10^(-rs/20) (10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];

% determine the order
[n,fo,ao,w] = firpmord(f,a,dev);

b = firpm(n+3, fo, ao, w);

% time to plot
figure

% linear gain plot
subplot(2,2,[1 3]);
% [h,f] = freqz(b,a,n,fs)
[h, omega] = freqz(b, 1, 2048, fs);
plot( fo.*(fs/2), ao, omega, abs(h));
legend('Ideal', 'Design');
grid minor;
xlabel('Frequency (Hz)');
ylabel('Gain');

% magnitude bode plot
subplot(2,2,2)
plot (omega, mag2db(abs(h)));
grid minor;
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');

% phase bode plot
subplot(2,2,4)
plot (omega, unwrap(angle(h)));
```

```
grid minor;
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');

% write to file
format long e
save ('fir_coef.txt', 'b', '-ascii', '-double', '-tabs');
```

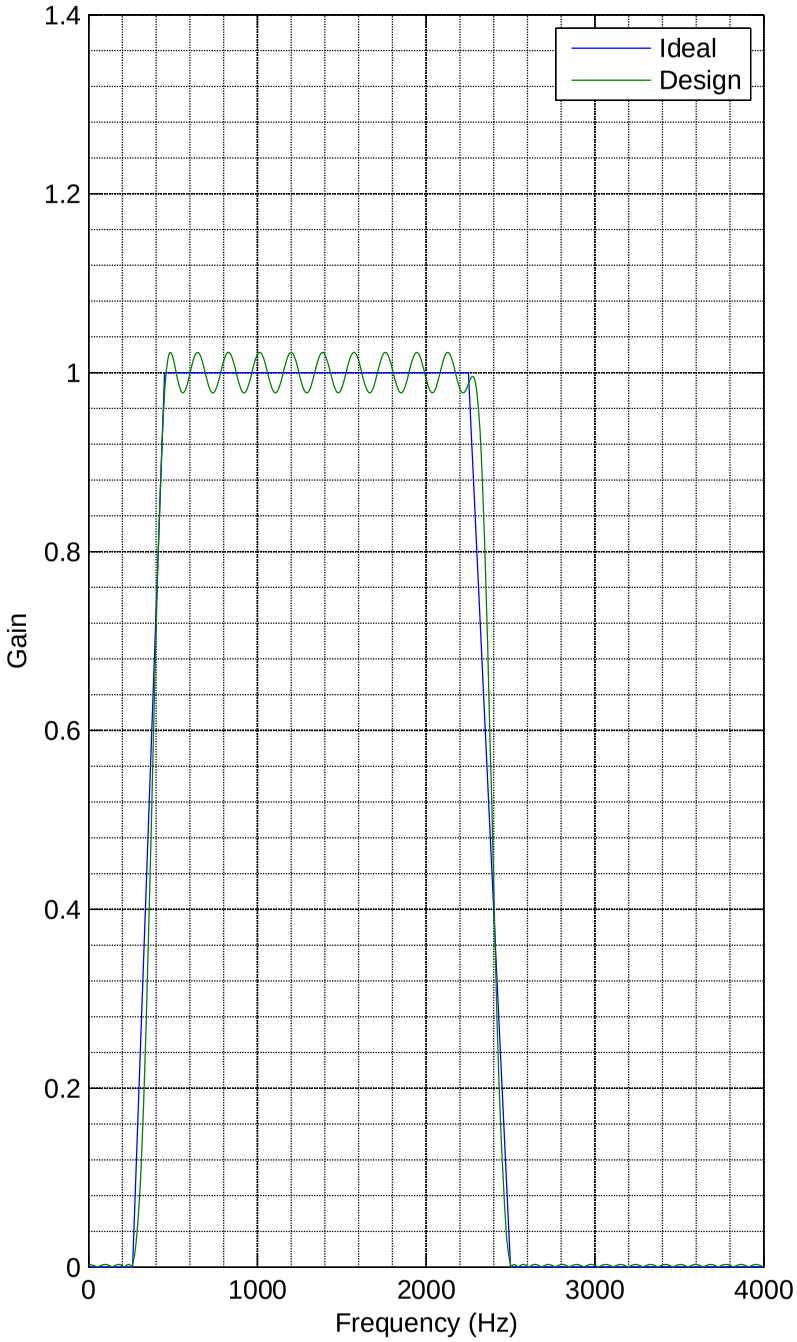
1.2 Coefficients

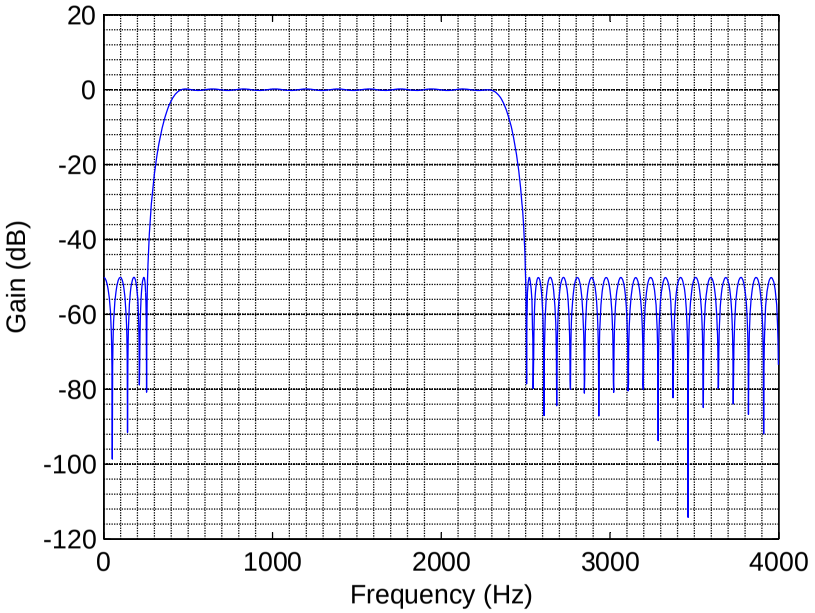
The coefficients generated by the Order 87 filter used is given by:

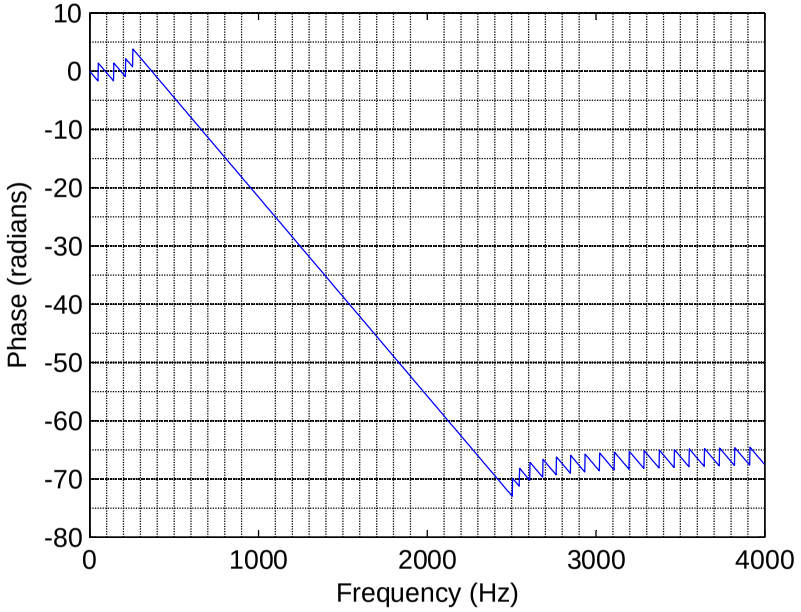
```
  -5.623823486158163e-03   -4.814285150867136e-03    3.237747605309768e-03    5.262307736662378e-03
3.832767802377313e-04    2.522852408070471e-03    6.642759430555022e-03    2.019154091723755e-03
6.083815421606797e-04    6.019507451326197e-03    2.258885469945656e-03   -4.058165617474114e-03
1.105369803703248e-03    8.757033068230690e-04   -9.438909556934223e-03   -6.713337183199348e-03
-4.491209456137727e-04   -1.078114100877992e-02   -1.283302504481474e-02    7.435733855308850e-04
-3.647574495656666e-03   -1.228547240652902e-02    4.906921613346250e-03    1.791976942964414e-02
-3.512499629985368e-03    8.432845956606996e-03    2.824214003399047e-02    9.541442788719748e-03
4.652770513821219e-03    3.318119501420717e-02    1.916147197952098e-02   -1.164093838147069e-02
1.490581695337271e-02    1.864062674743676e-02   -3.839051552509087e-02   -3.097763574266678e-02
6.989123352177381e-03   -6.226551473176629e-02   -1.010544436236774e-01   -9.643722538302999e-03
-5.129503215550500e-02   -2.109183819768691e-01   -2.156221212042710e-02    4.213315377569838e-01
4.213315377569838e-01   -2.156221212042710e-02   -2.109183819768691e-01   -5.129503215550500e-02
-9.643722538302999e-03   -1.010544436236774e-01   -6.226551473176629e-02    6.989123352177381e-03
-3.097763574266678e-02   -3.839051552509087e-02    1.864062674743676e-02    1.490581695337271e-02
-1.164093838147069e-02    1.916147197952098e-02    3.318119501420717e-02    4.652770513821219e-03
9.541442788719748e-03    2.824214003399047e-02    8.432845956606996e-03   -3.512499629985368e-03
1.791976942964414e-02    4.906921613346250e-03   -1.228547240652902e-02   -3.647574495656666e-03
7.435733855308850e-04   -1.283302504481474e-02   -1.078114100877992e-02   -4.491209456137727e-04
-6.713337183199348e-03   -9.438909556934223e-03    8.757033068230690e-04    1.105369803703248e-03
-4.058165617474114e-03    2.258885469945656e-03    6.019507451326197e-03    6.083815421606797e-04
2.019154091723755e-03    6.642759430555022e-03    2.522852408070471e-03    3.832767802377313e-04
5.262307736662378e-03    3.237747605309768e-03   -4.814285150867136e-03   -5.623823486158163e-03
```

1.3 Frequency Response

The frequency response of the generated filter is given on the following page.


Ideal
Design
Gain
Frequency (Hz)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1000
2000
3000
4000


Gain (dB)
Frequency (Hz)
20
0
-20
-40
-60
-80
-100
-120
0
1000
2000
3000
4000


Phase (radians)
Frequency (Hz)
10
0
-10
-20
-30
-40
-50
-60
-70
-80
0
1000
2000
3000
4000

2 Non-Circular FIR Filter

2.1 Code Description

The coefficients for the filter is kept in a global `double` array with the name `b`. An array of size 88, `buffer`, is used as the storage for the previous inputs, required for the convolution. At the start of every ISR, the sample is first read from the input port.

```
Int16 sample = mono_read_16Bit(); // read
```

The buffer is then updated as though it's a shift register.

```
// Handle the buffer
for (i = N-1; i > 0; i--)
  buffer[i] = buffer[i-1];
buffer[0] = sample;
```

Finally, the convolution is done by a call to the `convoluteNonCircular` function and the output is written to. The convolution is done simply according to the following equation

$$output = \sum_{i=0}^{87} b[i] \times buffer[i]$$

and is implemented in code as below:

```
for (i = 0; i < N; i++)
  output += b[i] * buffer[i];
```

2.2 Oscilloscpe Traces

The oscilloscope trace of the filter implemented on the DSP behave as expected with the amplitude changing accordingly.

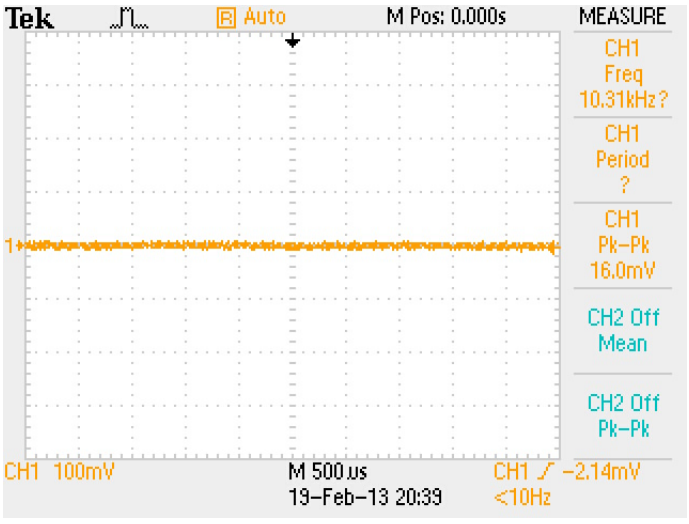


Figure 1: 200 Hz input, with almost zero output. This is in the stopband.

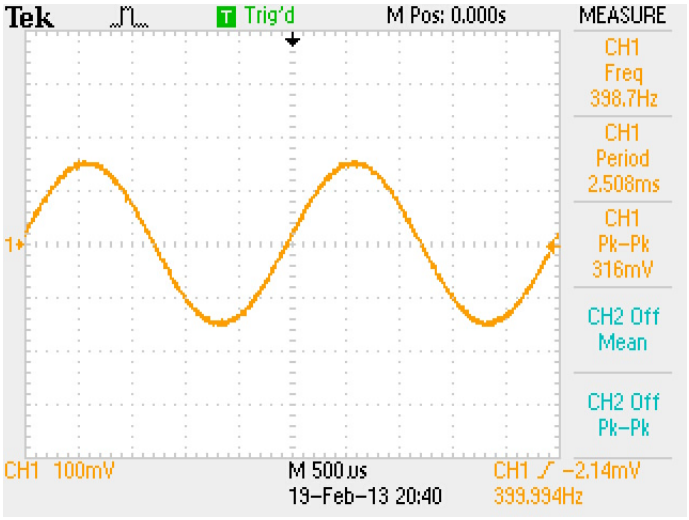


Figure 2: 400 Hz input, with increasing output amplitude. This is in the first transition band.

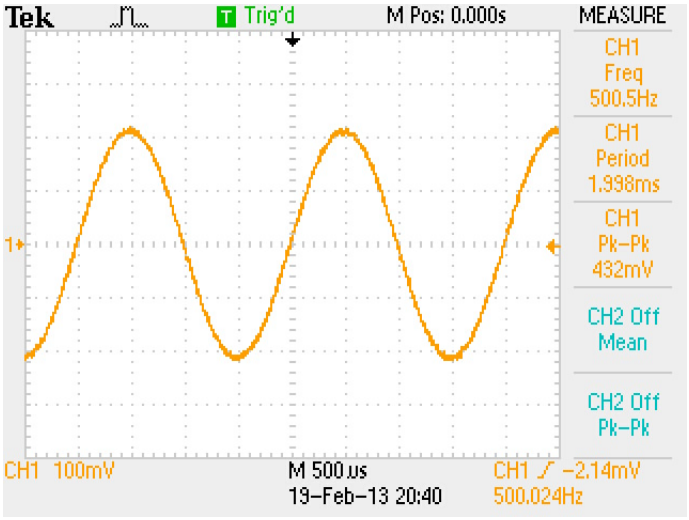


Figure 3: 500 Hz input, with maximum output amplitude. This is within the passband.

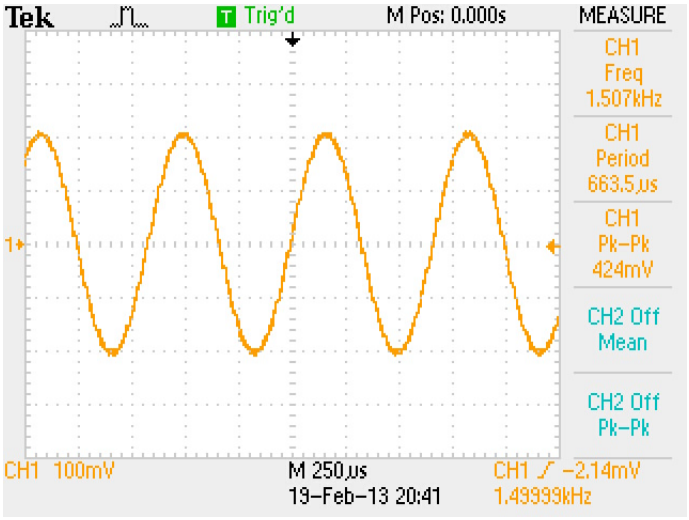


Figure 4: 1500 Hz input, with maximum amplitude. This is within the passband.

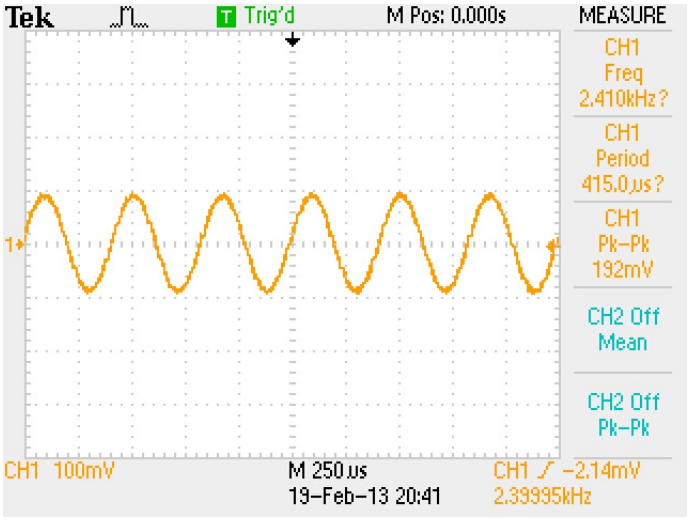


Figure 5: 2400 Hz, with decreasing amplitude. This is within the second transition band.

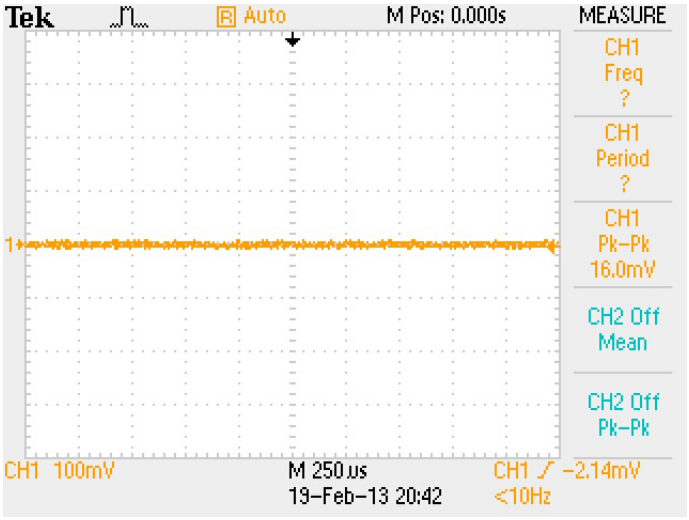


Figure 6: 3000 Hz input, with zero output. This is within the second stopband.

2.3 Code Performance

The number of cycles to run each iteration of the ISR was benchmarked in terms of the number of instruciton cycles. Measurements were taken several times for each optimisation level.The results are laid out below:

Optimisation Level	Number of Clock Cycles
None	9566, 9353, 9347, 9348, 9343, **9342**
Level 0	7662, 7443, 7439, 7438, **7438**
Level 2	5477, 5254, 5248, 5244, 5238, **5238**

2.4 Code Listing

```
/*************************************************************************************
             DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
                    IMPERIAL COLLEGE LONDON

              EE 3.19: Real Time Digital Signal Processing
                 Dr Paul Mitcheson and Daniel Harvey

                                LAB 4 - Non-circular FIR
 *************************************************************************************/

/*************************** Pre-processor statements ****************************/

#include <stdlib.h>
#include <stdio.h>
//  Included so program can make use of DSP/BIOS configuration tool.
#include "dsp_bios_cfg.h"

/* The file dsk6713.h must be included in every program that uses the BSL.  This
   example also includes dsk6713_aic23.h because it uses the
   AIC23 codec module (audio interface). */
#include "dsk6713.h"
#include "dsk6713_aic23.h"
```

```
23
24 // math library (trig functions)
25 #include <math.h>
26
27 // Some functions to help with writing/reading the audio ports when using interrupts.
28 #include <helper_functions_ISR.h>
29
30 /*********************************** Global declarations ****************************************/
31
32 /* Audio port configuration settings: these values set registers in the AIC23 audio
33    interface to configure it. See TI doc SLWS106D 3-3 to 3-10 for more info. */
34 DSK6713_AIC23_Config Config = { \
35        /*****************************************************************/
36        /*   REGISTER              FUNCTION            SETTINGS         */
37        /*****************************************************************/\
38     0x0017,  /* 0 LEFTINVOL  Left line input channel volume  0dB                  */\
39     0x0017,  /* 1 RIGHTINVOL Right line input channel volume 0dB                  */\
40     0x01f9,  /* 2 LEFTHPVOL  Left channel headphone volume   0dB                  */\
41     0x01f9,  /* 3 RIGHTHPVOL Right channel headphone volume  0dB                  */\
42     0x0011,  /* 4 ANAPATH    Analog audio path control       DAC on, Mic boost 20dB*/\
43     0x0000,  /* 5 DIGPATH    Digital audio path control      All Filters off       */\
44     0x0000,  /* 6 DPOWERDOWN Power down control              All Hardware on       */\
45     0x0043,  /* 7 DIGIF      Digital audio interface format  16 bit                */\
46     0x008d,  /* 8 SAMPLERATE Sample rate control             8 KHZ                 */\
47     0x0001   /* 9 DIGACT     Digital interface activation    On                    */\
48        /*****************************************************************/
49 };
50
51
52 // Codec handle:- a variable used to identify audio interface
53 DSK6713_AIC23_CodecHandle H_Codec;
54
55
56 /*********************************** Filter Stuff ***************************************/
57 // The order of the FIR filter +1
58 #define N 88
59
60 // include the coefficients
61 #include "fir_coef.txt"
62
63 // define the buffer
64 Int16 buffer[N] = {0};
65
66  /*********************************** Function prototypes ********************************/
67 void init_hardware(void);
68 void init_HWI(void);
69 void ISR_AIC(void);
70 Int16 convoluteNonCircular(void);
71 /********************************** Main routine ****************************************/
72 void main(){
73
74
75   // initialize board and the audio port
76   init_hardware();
77
78   /* initialize hardware interrupts */
79   init_HWI();
```

```
80
81    /* loop indefinitely, waiting for interrupts */
82    while(1)
83    {};
84
85  }
86
87  /******************************** init_hardware() ********************************/
88  void init_hardware()
89  {
90      // Initialize the board support library, must be called first
91      DSK6713_init();
92
93      // Start the AIC23 codec using the settings defined above in config
94      H_Codec = DSK6713_AIC23_openCodec(0, &Config);
95
96    /* Function below sets the number of bits in word used by MSBSP (serial port) for
97    receives from AIC23 (audio port). We are using a 32 bit packet containing two
98    16 bit numbers hence 32BIT is set for  receive */
99    MCBSP_FSETS(RCR1, RWDLEN1, 32BIT);
100
101   /* Configures interrupt to activate on each consecutive available 32 bits
102   from Audio port hence an interrupt is generated for each L & R sample pair */
103   MCBSP_FSETS(SPCR1, RINTM, FRM);
104
105   /* These commands do the same thing as above but applied to data transfers to
106   the audio port */
107   MCBSP_FSETS(XCR1, XWDLEN1, 32BIT);
108   MCBSP_FSETS(SPCR1, XINTM, FRM);
109
110
111 }
112
113 /******************************** init_HWI() ************************************/
114 void init_HWI(void)
115 {
116   IRQ_globalDisable();      // Globally disables interrupts
117   IRQ_nmiEnable();        // Enables the NMI interrupt (used by the debugger)
118   IRQ_map(IRQ_EVT_RINT1,4);   // Maps an event to a physical interrupt
119   IRQ_enable(IRQ_EVT_RINT1);    // Enables the event
120   IRQ_globalEnable();       // Globally enables interrupts
121
122 }
123
124 /****************** WRITE YOUR INTERRUPT SERVICE ROUTINE HERE***********************/
125
126 void ISR_AIC(void){
127     int i;
128     Int16 output;
129     Int16 sample = mono_read_16Bit(); // read
130
131     // Handle the buffer
132     for (i = N-1; i > 0; i--)
133       buffer[i] = buffer[i-1];
134
135     buffer[0] = sample;
136     output = convoluteNonCircular();
```

```
137     mono_write_16Bit(output); // write
138 }
139 
140 // Perform convolution
141 Int16 convoluteNonCircular(void){
142   double output = 0;
143   int i;
144 
145   for (i = 0; i < N; i++)
146     output += b[i] * buffer[i];
147 
148   return (Int16) round(output);
149 }
```