// FIR FILTERS

// sampling frequency 12.5khz

void main()

{

// variable declaration for the filtering action

short imp[20],\*inp,i,\*oup;

int index,j,x,k,sum;

unsigned int \*soc,\*adc1,\*dac1;

// address assignmenat for the variables

inp = (short \*)0x80001000;

oup=(short \*)0x80002000;

soc = (unsigned int \*)0x9004000c;

adc1 = (unsigned int \*)0x90040008;

dac1 = (unsigned int \*)0x90040008;

// coefficient for fir low pass filter with the sampling frequency of 12.5Khz

// cutoff frequency of 1Khz

// filter order 16

// LPF

imp[0] = 0x0FCCF;

imp[1] = 0x0FF37;

imp[2] = 0x02BA;

imp[3] = 0x06FA;

imp[4] = 0x0B6F;

imp[5] = 0x0F80;

imp[6] = 0x01298;

imp[7] = 0x01444;

imp[8] = 0x01444;

imp[9] = 0x01298;

imp[10] = 0x0F80;

imp[11] = 0x0B6F;

imp[12] = 0x06FA;

imp[13] = 0x02BA;

imp[14] = 0x0FF37;

imp[15] = 0x0FCCF;

// HPF

/\*

imp[0] = 0x0FDC3;

imp[1] = 0x070E;

imp[2] = 0x0F5DE;

imp[3] = 0x0214;

imp[4] = 0x0E8ED;

imp[5] = 0x0CC;

imp[6] = 0x0D23F;

imp[7] = 0x03D38;

imp[8] = 0x03D38;

imp[9] = 0x0D23F;

imp[10] = 0x0CC;

imp[11] = 0x0E8ED;

imp[12] = 0x0214;

imp[13] = 0x0F5DE;

imp[14] = 0x070E;

imp[15] = 0x0FDC3;

\*/

// rect window 5khz cut off

// fs = 32 khz

/\*imp[0] = 0x0F5C7;

imp[1] = 0x05A7;

imp[2] = 0x0FE52;

imp[3] = 0x011B8;

imp[4] = 0x0F7BD;

imp[5] = 0x05F5;

imp[6] = 0x0C9CF;

imp[7] = 0x02B13;

imp[8] = 0x02B13;

imp[9] = 0x0C9CF;

imp[10] = 0x05F5;

imp[11] = 0x0F7BD;

imp[12] = 0x011B8;

imp[13] = 0x0FE52;

imp[14] = 0x05A7;

imp[15] = 0x0F5C7;

\*/// BPF

/\*

imp[0] = 0x00;

imp[1] = 0x0FB4E;

imp[2] = 0x0F3A7;

imp[3] = 0x0EEEA;

imp[4] = 0x0F2BF;

imp[5] = 0x00;

imp[6] = 0x0111C;

imp[7] = 0x01D0B;

imp[8] = 0x01D0B;

imp[9] = 0x0111C;

imp[10] = 0x00;

imp[11] = 0x0F2BF;

imp[12] = 0x0EEEA;

imp[13] = 0x0F3A7;

imp[14] = 0x0FB4E;

imp[15] = 0x00;

\*/

// BRF

/\*

imp[0] = 0x0F235;

imp[1] = 0x03EC;

imp[2] = 0x064;

imp[3] = 0x018EC;

imp[4] = 0x0414;

imp[5] = 0x01638;

imp[6] = 0x0DC53;

imp[7] = 0x03E7E;

imp[8] = 0x03E7E;

imp[9] = 0x0DC53;

imp[10] = 0x01638;

imp[11] = 0x0414;

imp[12] = 0x018EC;

imp[13] = 0x064;

imp[14] = 0x03EC;

imp[15] = 0x0F235;

\*/

// index variable for circular buffer action

index = 0;

// initialise the input array to zero

for(i=0;i<20;i++)

\*(inp+i)=0;

// infinite loop for the filter starts here

while(1)

{

x = \*(soc);

for(k=0;k<50;k++);

x = \*(adc1);

x = x & 0x0fff;

x = x ^ 0x0800;

x = x - 0x0800;

inp[(index & 0xf)] = x;

index = (index +1) & 0xf;

sum = 0;

for (j = 0; j < 16; j++)

{

sum += inp[(index + j) & 0xf] \* imp[j];

}

// calculated output is getting output variable & added with offset of 0x800

\*oup= (sum>>15) + 0x800 ;

// out the filtered sample to dac

\*dac1 = \*oup;

// \*dac1 = x + 0x800;

// port4 = \*oup;

}

}