1.

**float** a[] = {};

**float** b[] = {};

**float** x[3] = {0, 0, 0};

**float** y[3] = {0, 0, 0};

adcd0result = AdcdResultRegs.ADCRESULT0;

x[0] = (**float**)3.0\*(adcd0result / 4095.0);

y[0] = (b[0]\*x[0] + b[1]\*x[1] + b[2]\*x[2] - a[1]\*y[1] - a[2]\*y[2])/a[0];

setDACA(y[0]);

x[3] = x[2];

x[2] = x[1];

y[3] = y[2];

y[2] = y[1];

2.

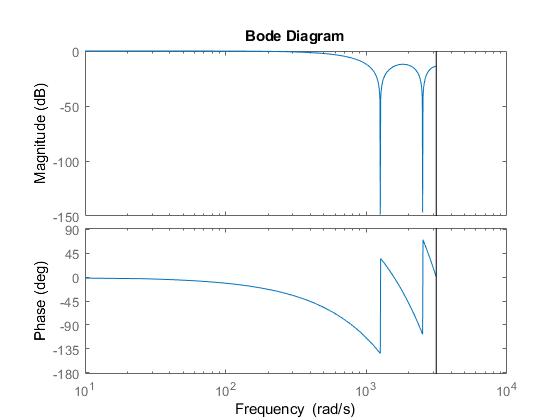
To calculated the output of the filter, it needs the input of this time step and previous output. The previous output is determined by previous previous output. So each output is determined by all the inputs. In others words, the output of this time step yk has the contribution of all the previous inputs: x1, x2, …xk. And the parameter of each items are the same. So it is called the average infinite impulse response

FIR:

0.2 z^4 + 0.2 z^3 + 0.2 z^2 + 0.2 z + 0.2

-----------------------------------------

z^4



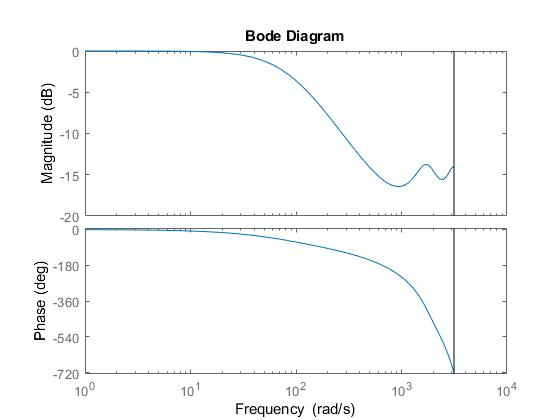
IIR：

tfIIR =

0.2

-------------------------------------

z^4 - 0.2 z^3 - 0.2 z^2 - 0.2 z - 0.2



3.

EPwm5Regs.TBPRD = 1000;

AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;

AdcbRegs.ADCSOC2CTL.bit.CHSEL = 0;

AdcbRegs.ADCSOC4CTL.bit.CHSEL = 0;

AdcbRegs.ADCSOC6CTL.bit.CHSEL = 0;

AdcbRegs.ADCSOC8CTL.bit.CHSEL = 0;

AdcbRegs.ADCSOCACTL.bit.CHSEL = 0;

AdcbRegs.ADCSOCCCTL.bit.CHSEL = 0;

AdcbRegs.ADCSOCECTL.bit.CHSEL = 0;

AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;

AdcbRegs.ADCSOC3CTL.bit.CHSEL = 1;

AdcbRegs.ADCSOC5CTL.bit.CHSEL = 1;

AdcbRegs.ADCSOC7CTL.bit.CHSEL = 1;

AdcbRegs.ADCSOC9CTL.bit.CHSEL = 1;

AdcbRegs.ADCSOCBCTL.bit.CHSEL = 1;

AdcbRegs.ADCSOCDCTL.bit.CHSEL = 1;

AdcbRegs.ADCSOCFCTL.bit.CHSEL = 1;

**\_\_interrupt** **void** **ADCD\_ISR** (**void**)

{

uint32\_t adcd0result = 0;

uint32\_t adcd1result = 1;

uint32\_t adcd0result = 2;

uint32\_t adcd1result = 3;

uint32\_t adcd0result = 4;

uint32\_t adcd1result = 5;

uint32\_t adcd0result = 6;

uint32\_t adcd1result = 7;

uint32\_t adcd0result = 8;

uint32\_t adcd1result = 9;

uint32\_t adcd0result = 10;

uint32\_t adcd1result = 11;

uint32\_t adcd0result = 12;

uint32\_t adcd1result = 13;

uint32\_t adcd0result = 14;

uint32\_t adcd1result = 15;

adcd0result = AdcdResultRegs.ADCRESULT0;

adcd1result = AdcdResultRegs.ADCRESULT1;

adcd2result = AdcdResultRegs.ADCRESULT2;

adcd3result = AdcdResultRegs.ADCRESULT3;

adcd4result = AdcdResultRegs.ADCRESULT4;

adcd5result = AdcdResultRegs.ADCRESULT5;

adcd6result = AdcdResultRegs.ADCRESULT6;

adcd7result = AdcdResultRegs.ADCRESULT7;

adcd8result = AdcdResultRegs.ADCRESULT8;

adcd9result = AdcdResultRegs.ADCRESULT9;

adcd10result = AdcdResultRegs.ADCRESULT10;

adcd11result = AdcdResultRegs.ADCRESULT11;

adcd12result = AdcdResultRegs.ADCRESULT12;

adcd13result = AdcdResultRegs.ADCRESULT13;

adcd14result = AdcdResultRegs.ADCRESULT14;

adcd15result = AdcdResultRegs.ADCRESULT15;

avg0 = 0.125\*adcd0result + 0.125\*adcd2result + 0.125\*adcd4result + 0.125\*adcd6result + 0.125\*adcd8result + 0.125\*adcd10result + 0.125\*adcd12result + 0.125\*adcd14result;

avg1 = 0.125\*adcd1result + 0.125\*adcd3result + 0.125\*adcd5result + 0.125\*adcd7result + 0.125\*adcd9result + 0.125\*adcd11result + 0.125\*adcd13result + 0.125\*adcd15result;

setDACA(avg0);

setDACA(avg1);

AdcdRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; //clear interrupt flag

PieCtrlRegs.PIEACK.all = PIEACK\_GROUP1;

}