# 11098 DP3

## Using software design tools

## to design a DSP application

## on the dsPIC® Digital Signal Controller (DSC)

# DSP Hands-on Class Series

- DSP: Part 1
  - Introduction to Digital Signal Processing using the dsPIC® DSC
- DSP: Part 2
  - Using the DSP Features of the dsPIC DSC Architecture
- DSP: Part 3
  - Using Software Design Tools to Design a DSP Application on the dsPIC DSC

# Session Objective

This session prepares you to do the following:

- Use the dsPIC® DSC Filter Design tool to design FIR/IIR filters.
- Learn various DSP library functions for filtering and spectrum analysis.
- Use dsPICworks™ signal analysis tool.

University of Microchip
U of M
MASTERs 2007

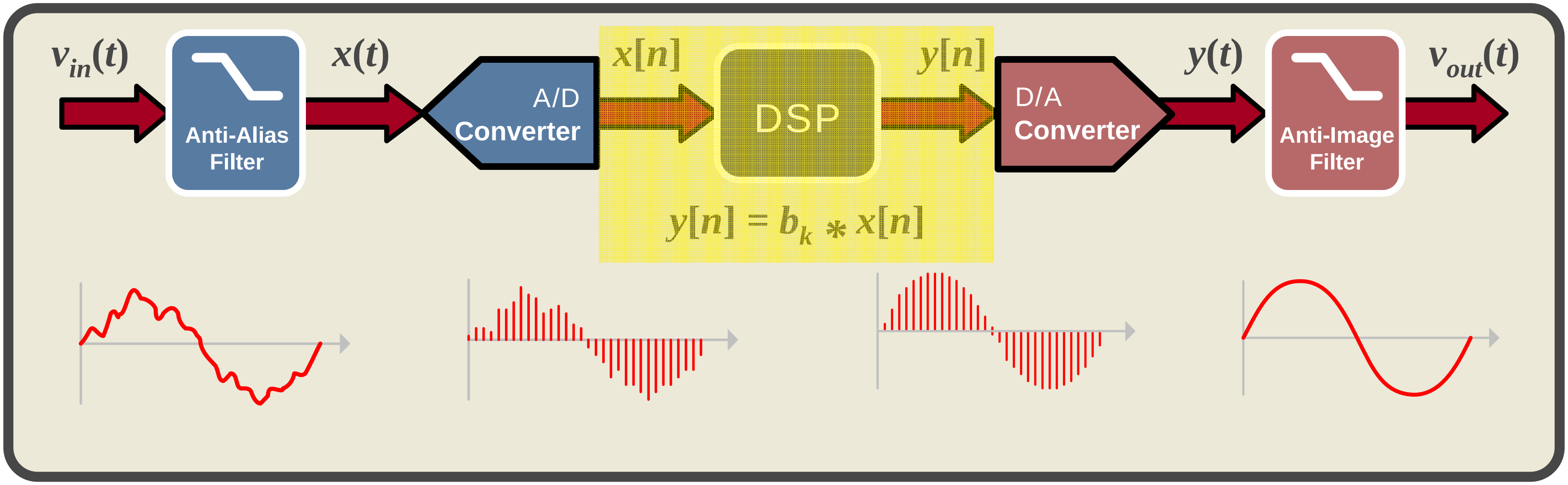
# Session Agenda

The session covers following topics:

- Digital Signal Chain Overview
- FIR Filter Design Lab1
- IIR Filter Design Lab2
- FFT Application and Result Interpretation Lab3

# Digital Signal Chain Overview

# Digital Signal Chain

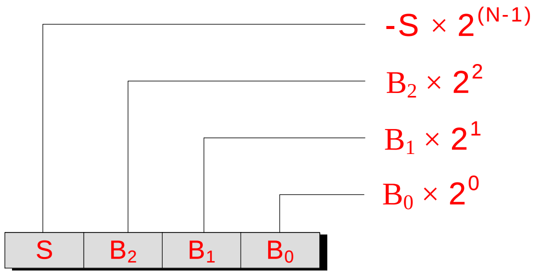


**A typical DSP system consists of five basic blocks:**

- Anti-Aliasing Filter
- Analog to Digital Converter
- Digital Signal Processor
- Digital to Analog Converter
- Reconstruction Filter

# Binary Number Representation

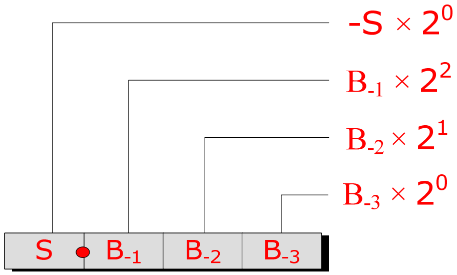
## ❑ Interpretation of bits in a 4-bit integer



0111 = (-0 x 8) + (1 x 4) + (1 x 2) + (1 x 1) = 7

1000 =(-1 x 8) + (0 x 4) + (0 x 2) + (0 x 1) = -8

## ❑ Interpretation of bits in a 4-bit fixed point number (Q3 format)



0111 = (-0 x 1) + (1 x 0.5) + (1 x 0.25) + (1 x 0.125) = =0.875

1000 =(-1 x 1) + (0 x 0.5) + (0 x 0.25) + (0 x 0.125) = -1

## ❑ Conversion of Fixed Point Number

**X = round(**x * $2^{Q}$)

**= round(**0.3 * $2^{12}$**) = round(**1228.8**) = 1229 in Q12 format**

**= round(**0.3 * $2^{15}$**) = round(**9830.4**) = 9830 in Q15 format**

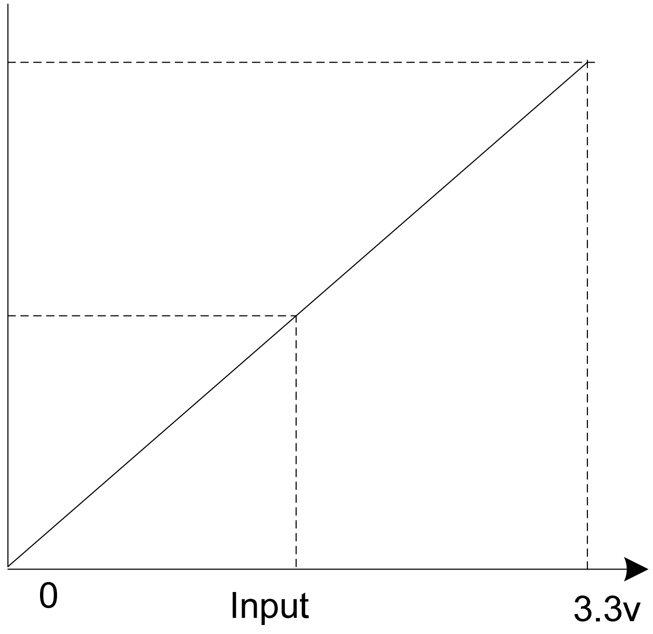
# ADC Result Format

## Unsigned Integer

0000 1111 1111 1111 (4095)

0000 1000 0000 0000 (2048)

0000 0000 0000 0000 (0)

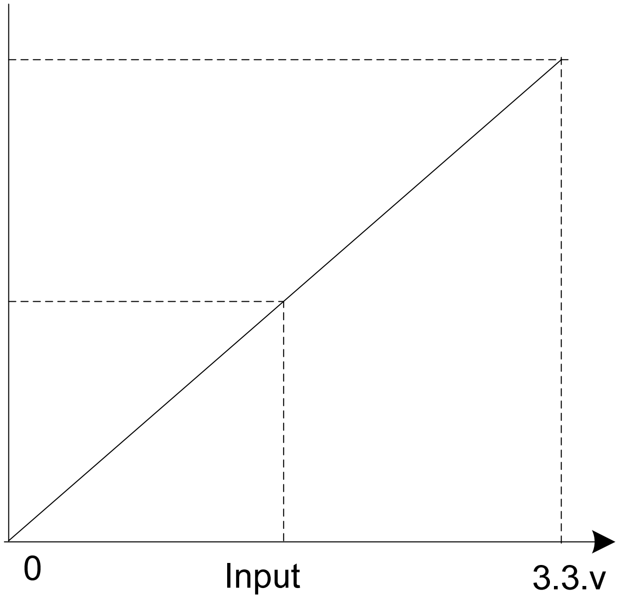


## Signed Fractional

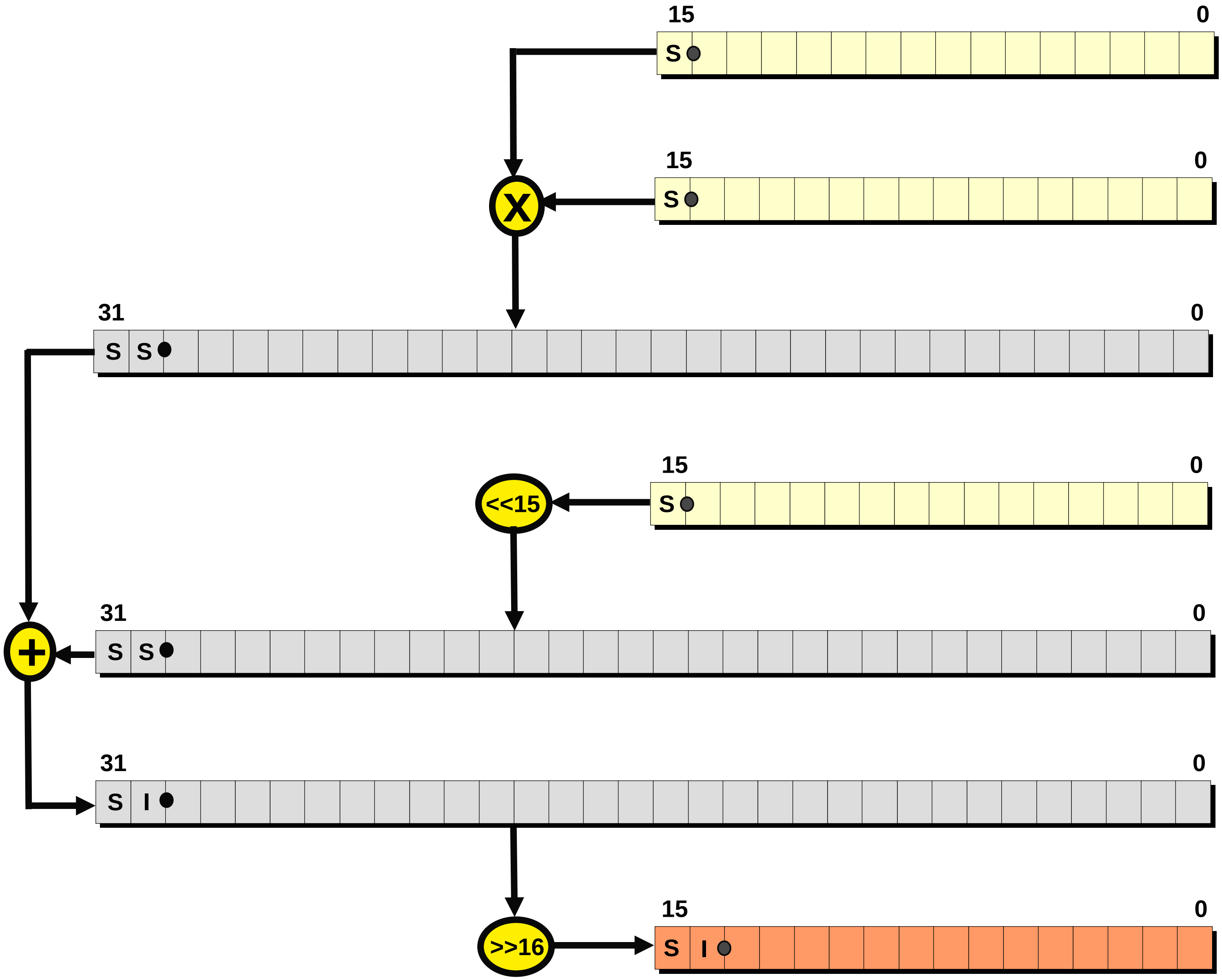
0111 1111 1111 0000 (+0.999)

0000 0000 0000 0000 (0)

1000 0000 0000 0000 (-1)



# Fixed Point Computation



M = m . 2^15
(Q15)

X = x . 2^15
(Q15)

M . X = m . x . 2^30
(Q30)

B = x . 2^15
(Q15)

B = b . 2^30
(Q30)

Y = (M . X) + B
= (m . x + b) 2^30

Y = (M . X) + B
= (m . x + b) 2^14

# Fixed Point Computation

## Advantages

- Uses Integer Arithmetic Unit
- Uses minimum memory space to represent fractional number

## Limitations

- May generate *Overflow* or *Underflow*
- Limited dynamic range *(Trade-off range for resolution)*

# FIR Filter Design

# Analog vs Digital Filter

## Analog Filter

- ✖ Made up from components such as resistors, capacitors, and op amps
- ✖ Subject to drift and are dependent on temperature
- ✖ Fixed hardware
- ✓ Well-established standard techniques for designing analog filter circuitry

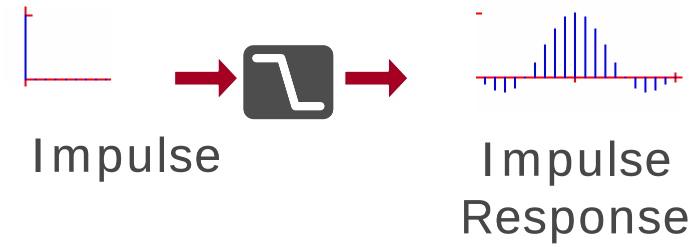
## Digital Filter

- ✓ Algorithm running on a processing core
- ✓ Are not subject to drift or dependent on temperature
- ✓ Programmable
- ✓ Can build on analog filter design techniques

# FIR vs IIR Filter

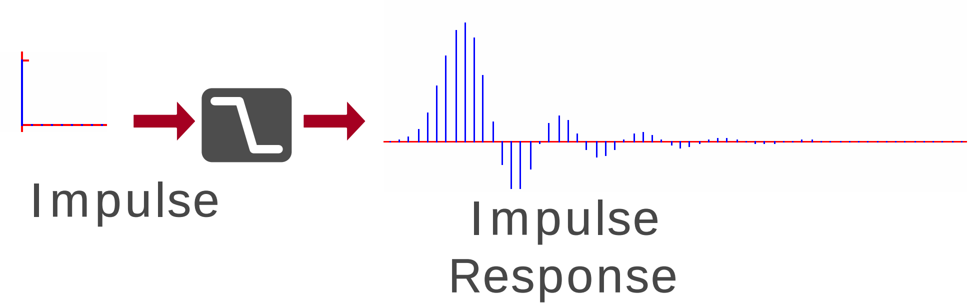
### Finite Impulse Response Filter

- ✓ Linear phase easy to obtain
- ✓ FIR filters always stable
- ✖ Large number of filter taps may be required for sharp responses
- ✖ No analog history to build upon



### Infinite Impulse Response Filter

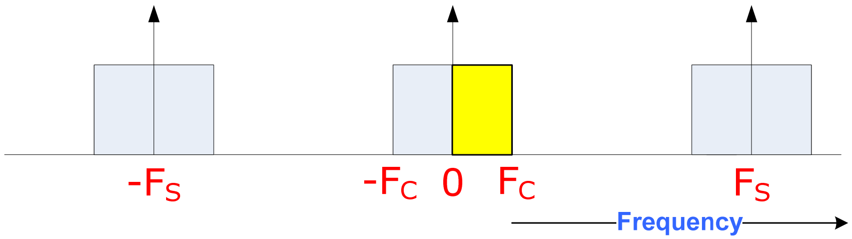
- ✖ Phase is difficult to control
- ✖ Possibility for instability
- ✓ Fewer numerical operations than FIR filters for sharp responses
- ✓ Can build on analog filter design techniques



# FIR Filter Design using ‘Windows’

- **Ideal frequency response is**

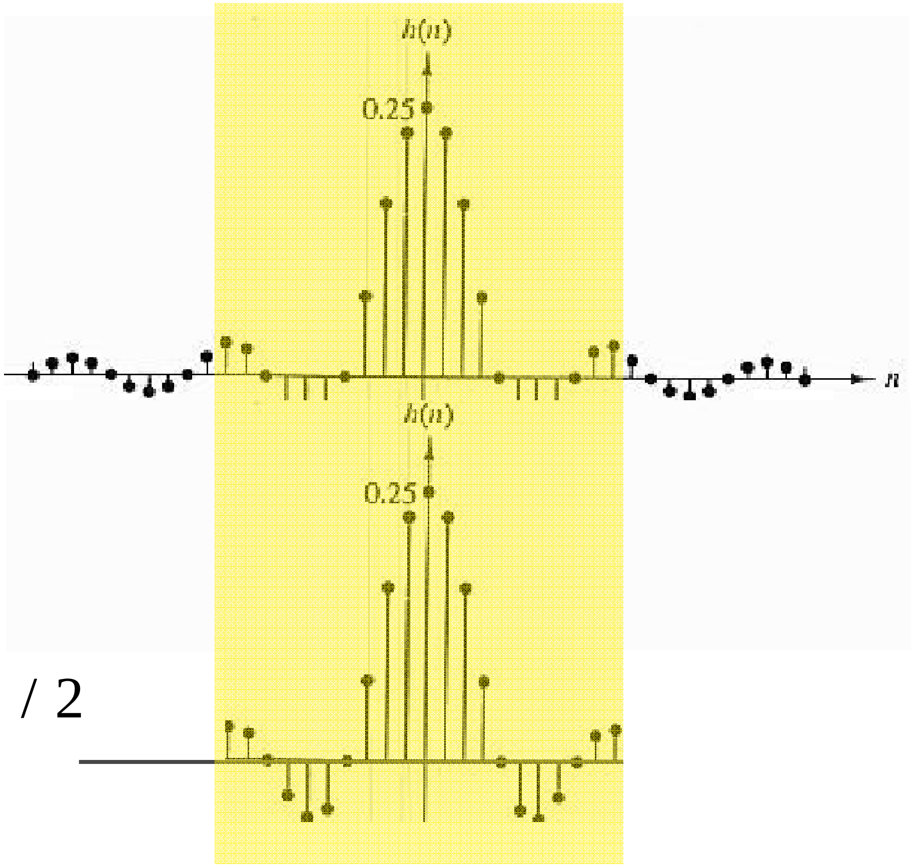
$$H_d(f) = \begin{cases} 1 & |f| < f_C \\ 0 & f_C < |f| < \dfrac{fs}{2} \end{cases}$$



- **Ideal Impulse response is**

$$h_d[n] = \frac{1}{2\pi}\int_{-\pi}^{+\pi} H_d(e^{j\omega}).e^{j\omega.k}\,d\omega$$

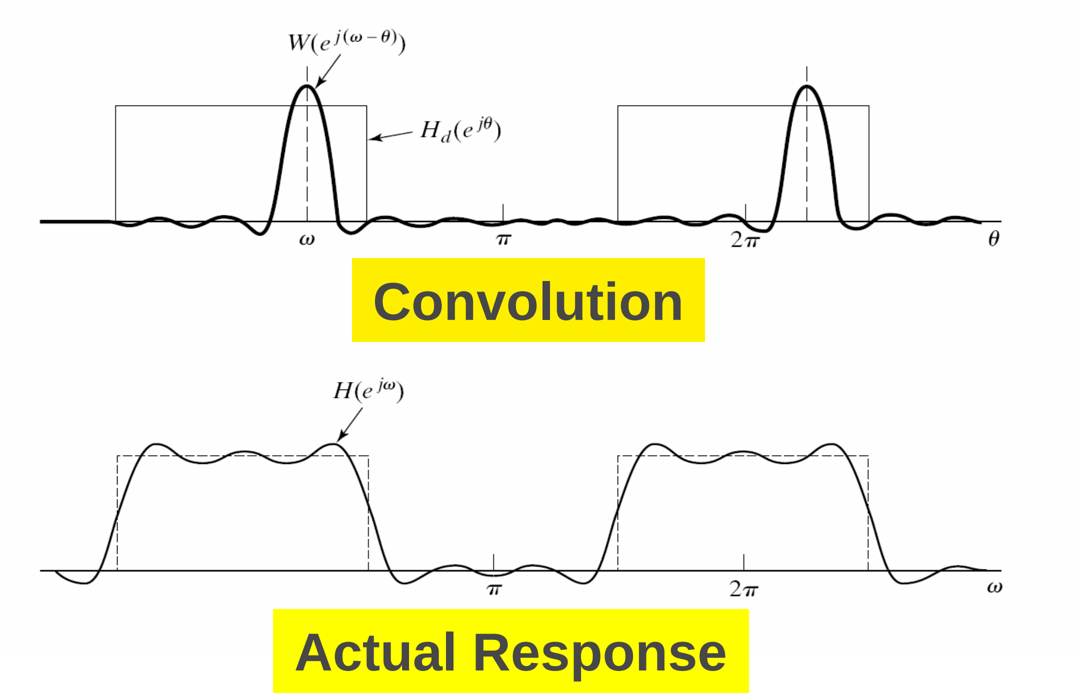
$$= \frac{\sin(\omega_c n)}{\pi n} \qquad -\infty < n < \infty$$



- **Truncated impulse response is**

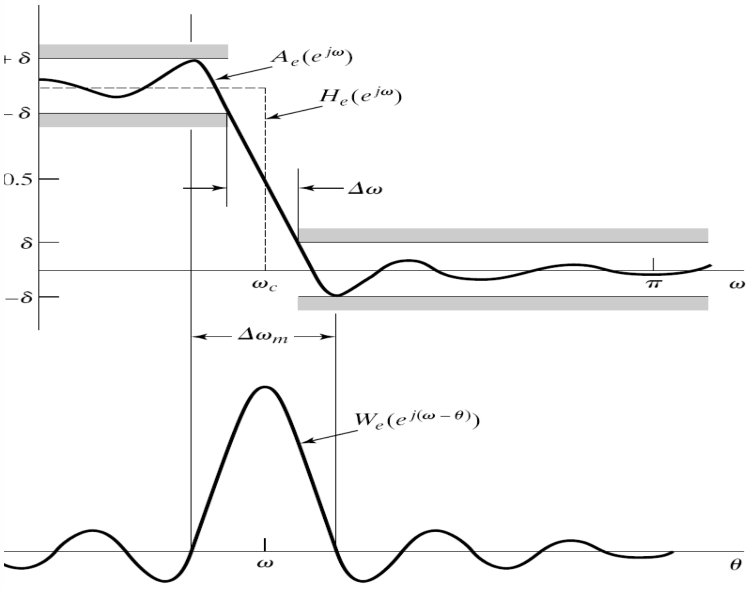
$$h[n] = \begin{cases} h_d[n] & -M/2 < n < M/2 \\ 0 & \text{otherwise} \end{cases}$$

# Windowing Effect (Gibbs Phenomena)

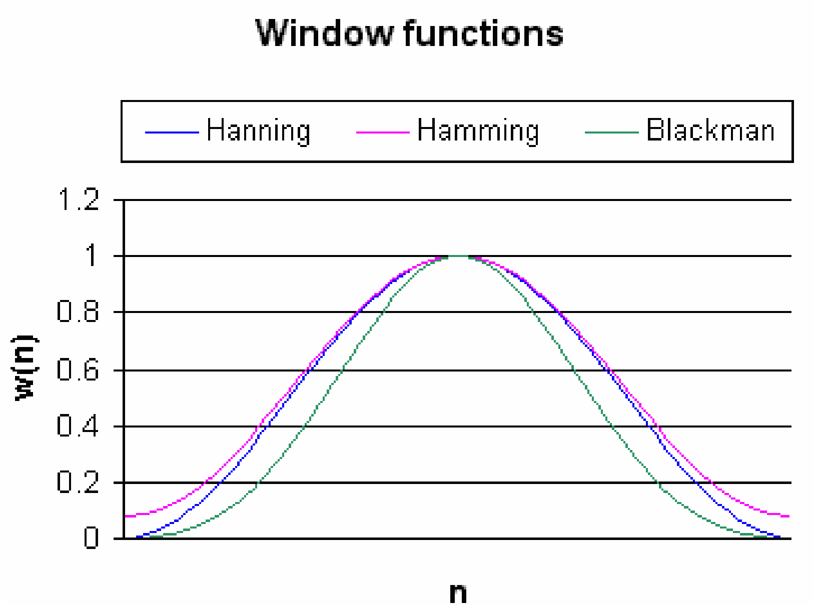


Convolution

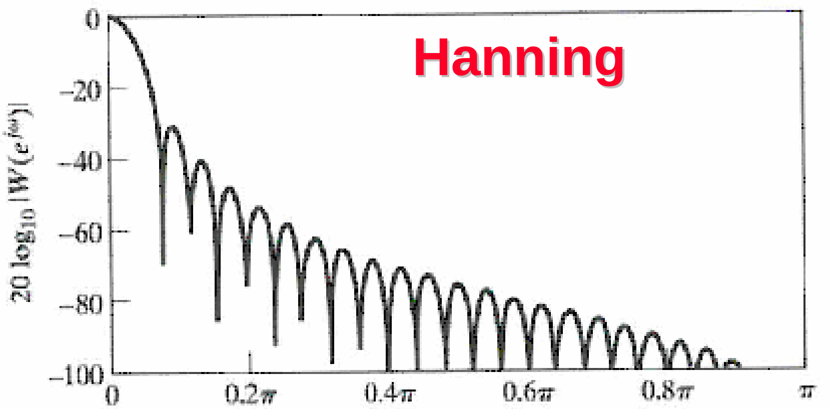
Actual Response

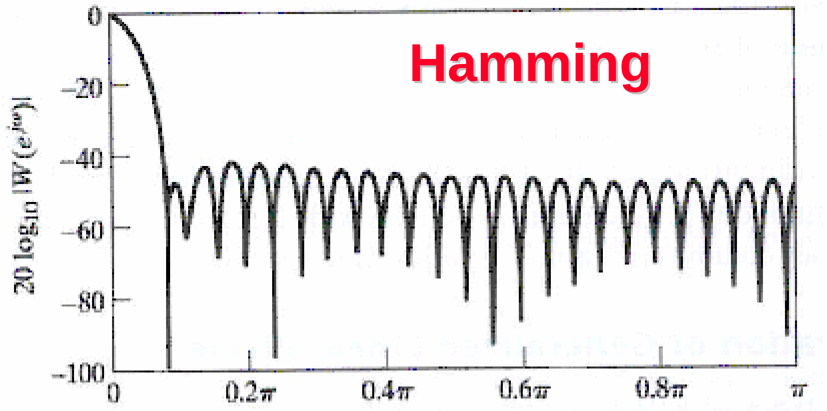


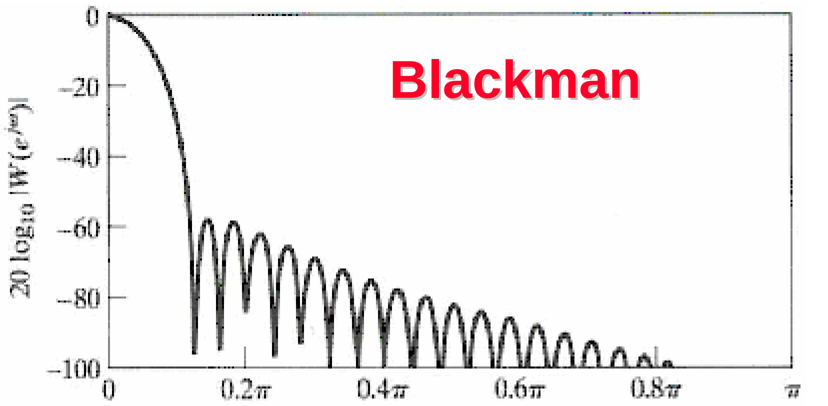
# Window Properties



| Window | Transition Width | Peak Sidelobe (dB) |
|---|---|---|
| Rectangular | 4π/M | -13 |
| Hanning | 8π/M | -32 |
| Hamming | 8π/M | -43 |
| Blackman | 12π/M | -58 |







# FIR Filter Implementation

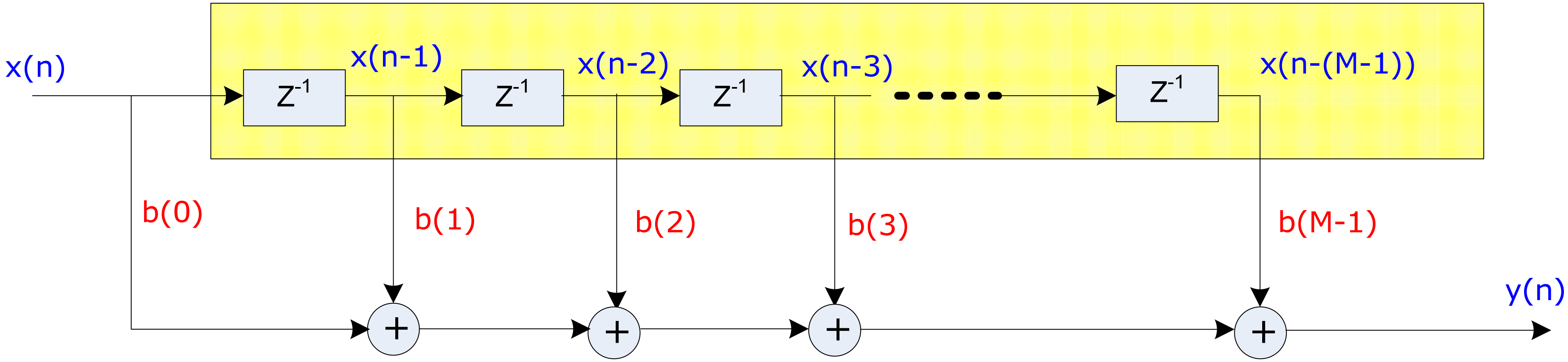
You can describe FIR filter using any of the following equation:

- The difference equation

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k)$$

- The transfer function

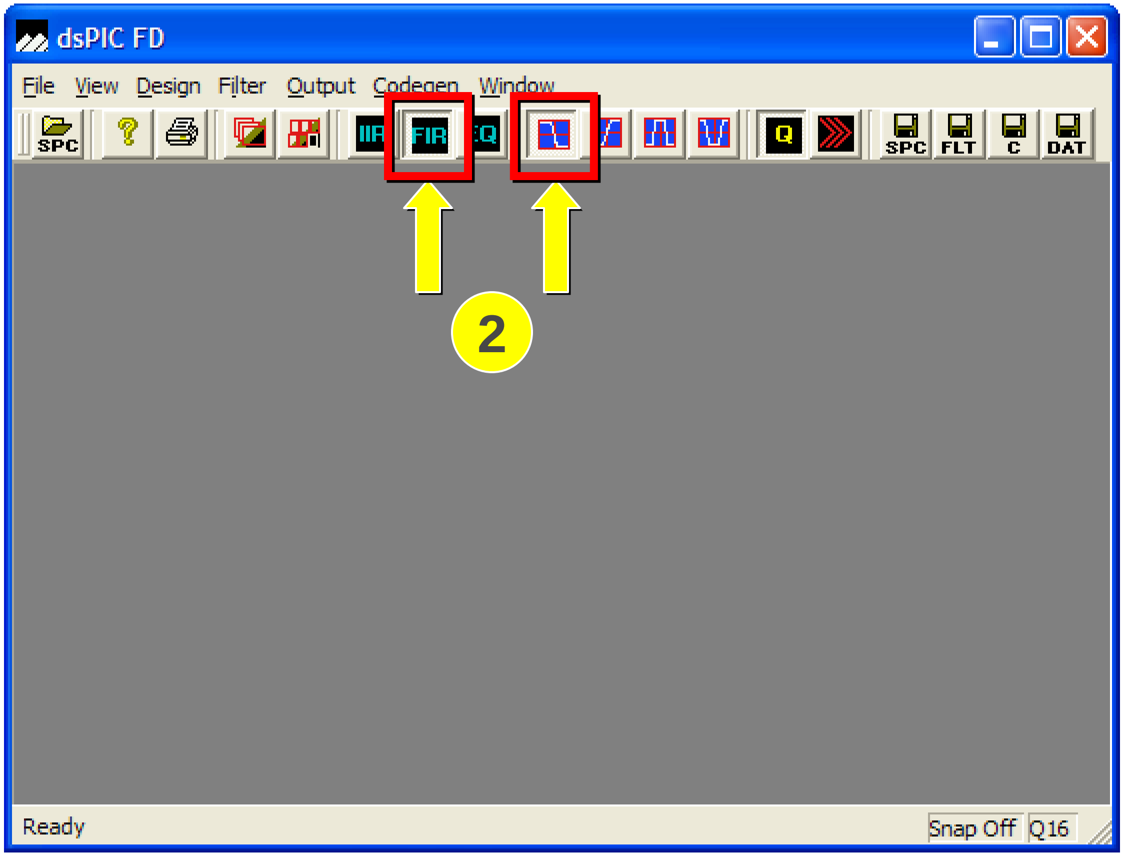
$$H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{M-1} z^{-(M-1)}$$



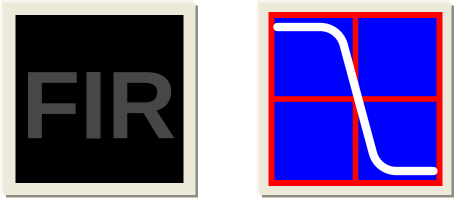
# Using dsPIC® DSC Filter Design to Generate Filter Coefficients

## ❶ Launch dsPIC DSC Filter Design

start ▶ All Programs ▶ mds ▶ dsPIC Filter Design ▶ dsPIC Filter Design

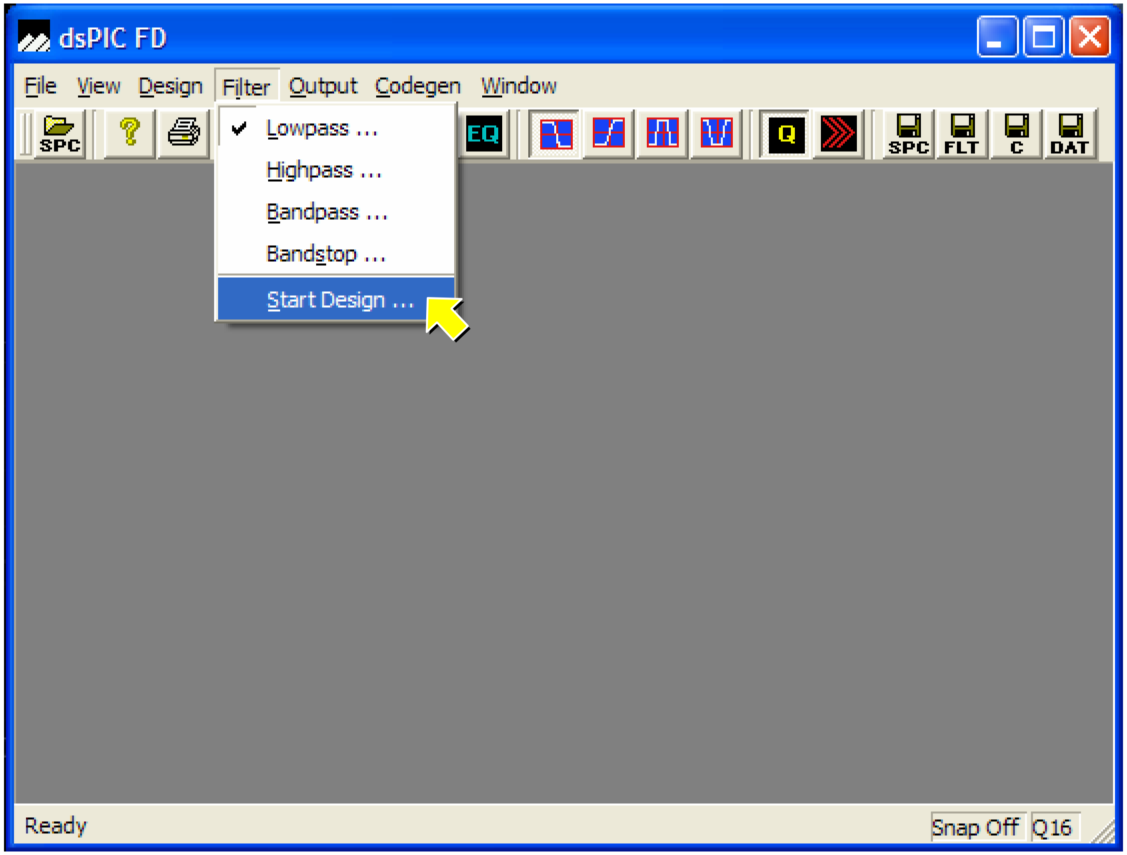


## ❷ Select FIR and Low Pass from the tool bar



Note:
FIR selects window method FIR design
EQ selects equiripple method FIR design

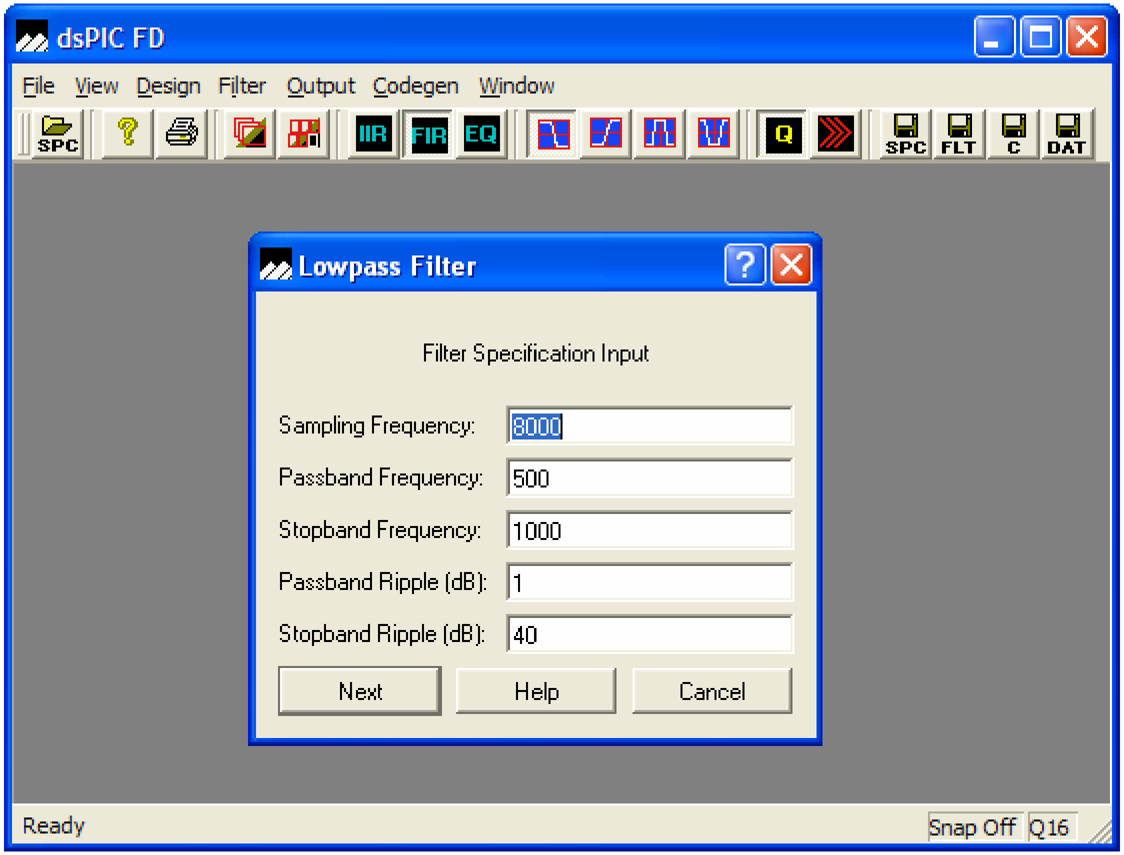
# Using dsPIC® DSC Filter Design to Generate Filter Coefficients



❸ Select from the menu

Filter ▶ Start Design…

# Using dsPIC® Filter Design to Generate Filter Coefficients



Filter Specification Input

Sampling Frequency: 8000

Passband Frequency: 500

Stopband Frequency: 1000

Passband Ripple (dB): 1

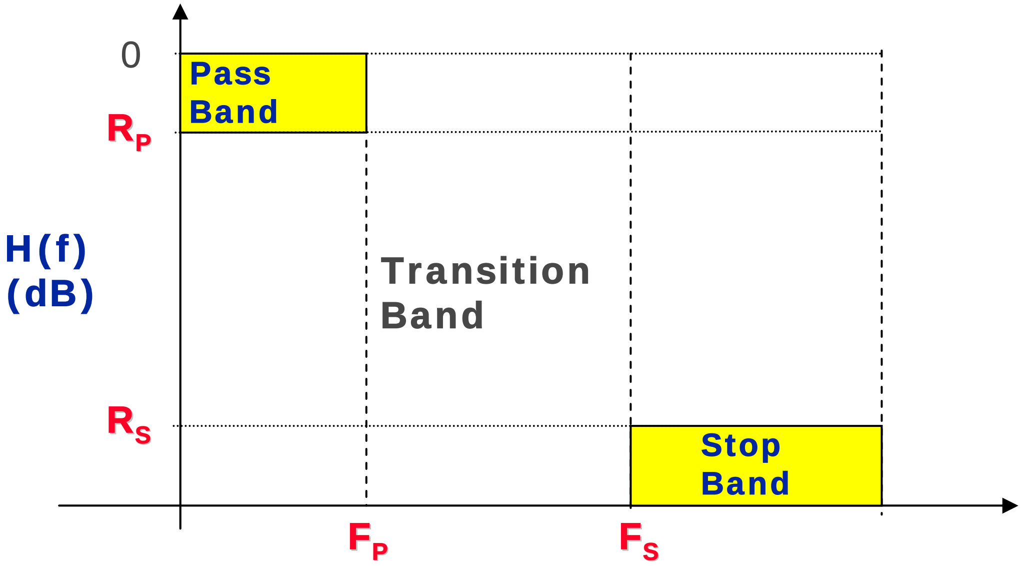
Stopband Ripple (dB): 40

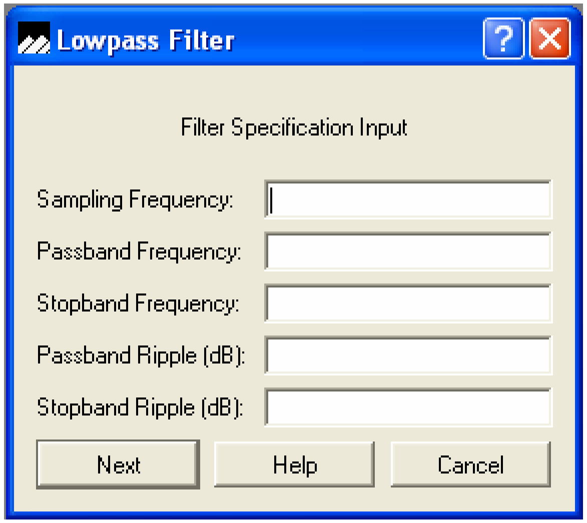
❹ Enter the following filter specifications:

Sampling Frequency = 8000 Hz
Passband Frequency = 500 Hz
Stopband Frequency = 1000 Hz
Passband Ripple = 1 dB
Stopband Ripple = 40 dB

Click Next

# Filter Specification





Filter Specification Input

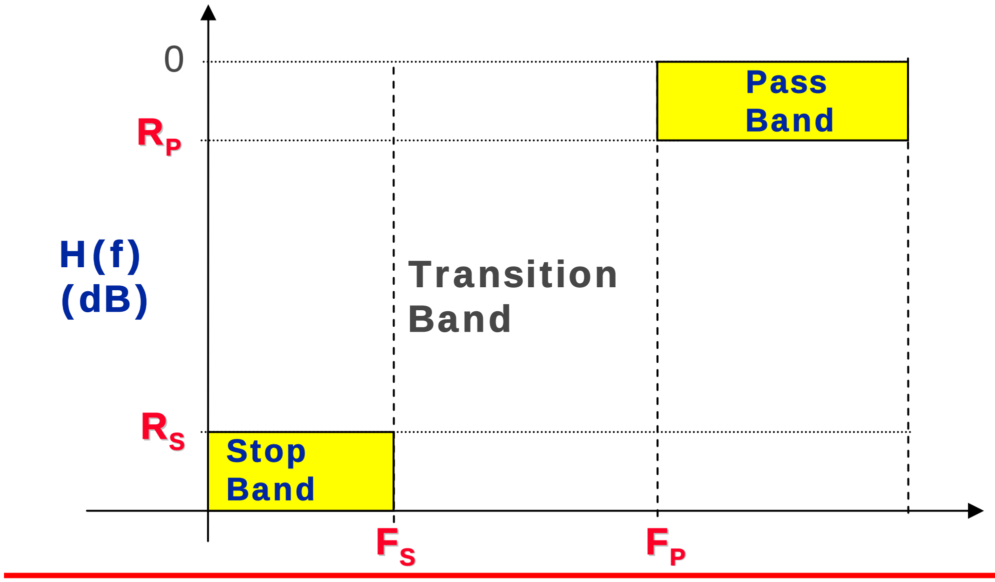
Sampling Frequency:

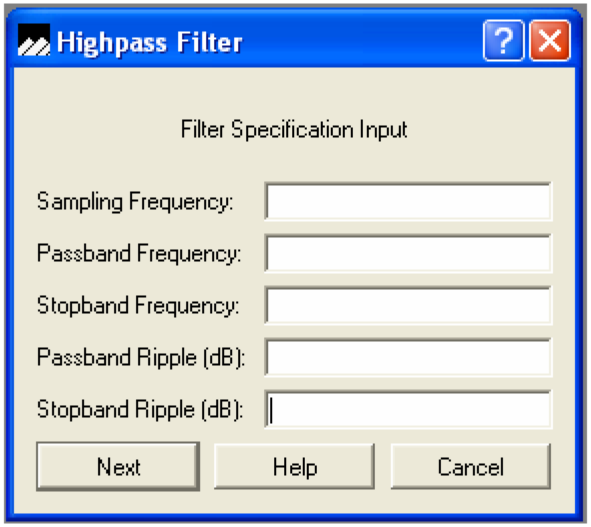
Passband Frequency:

Stopband Frequency:

Passband Ripple (dB):

Stopband Ripple (dB):





Filter Specification Input

Sampling Frequency:

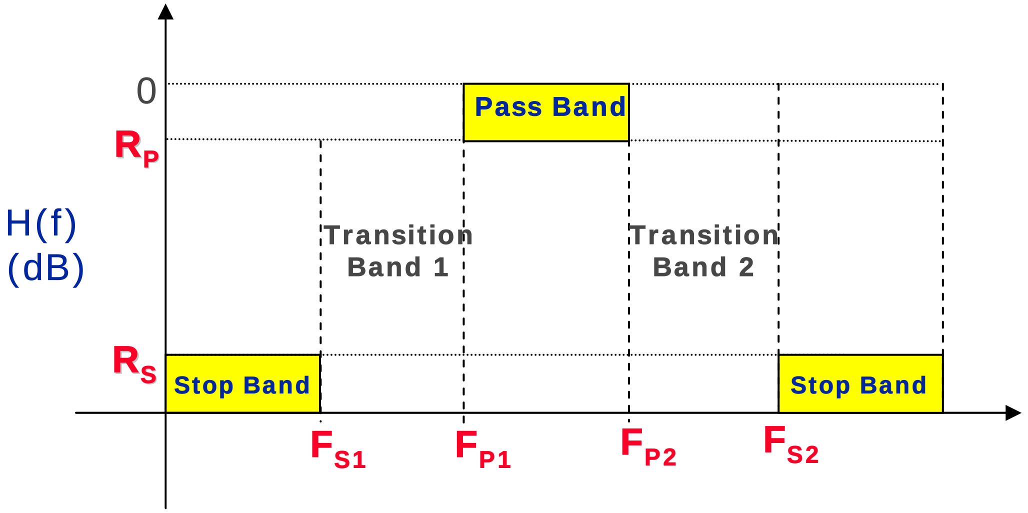
Passband Frequency:

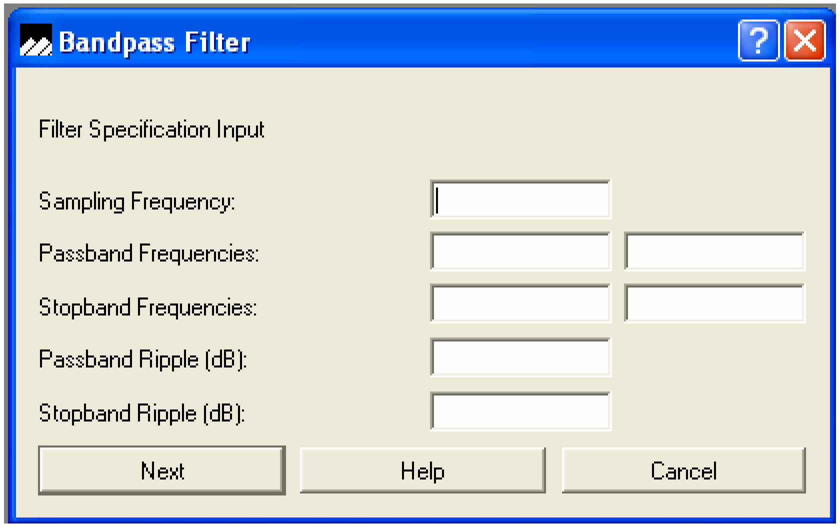
Stopband Frequency:

Passband Ripple (dB):

Stopband Ripple (dB):

# Filter Specification





**Bandpass Filter**

Filter Specification Input

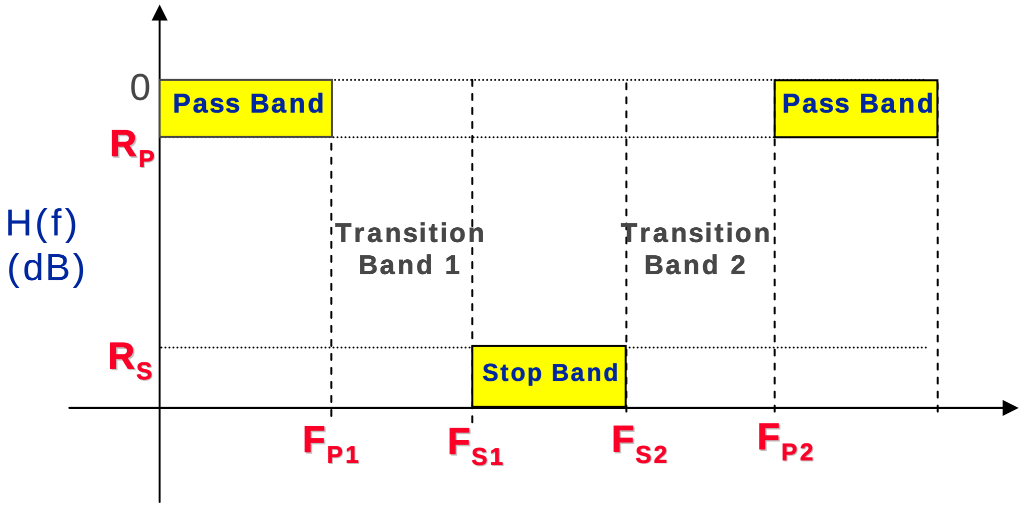
Sampling Frequency:

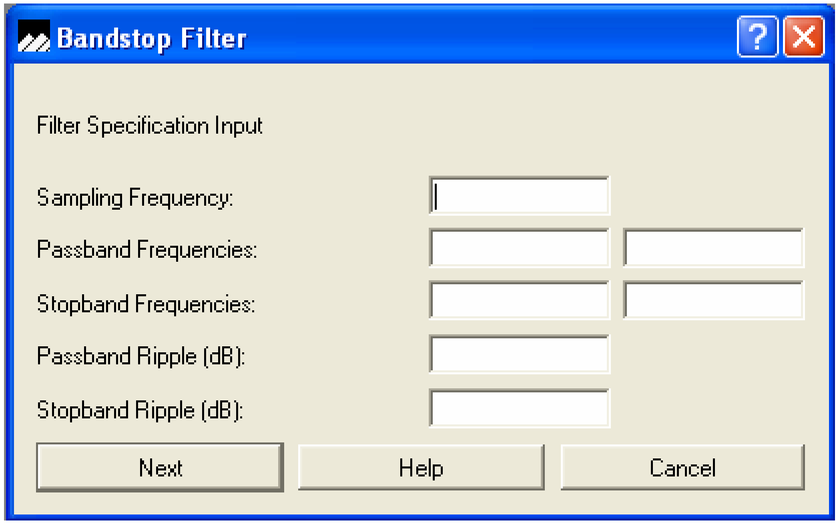
Passband Frequencies:

Stopband Frequencies:

Passband Ripple (dB):

Stopband Ripple (dB):





**Bandstop Filter**

Filter Specification Input

Sampling Frequency:

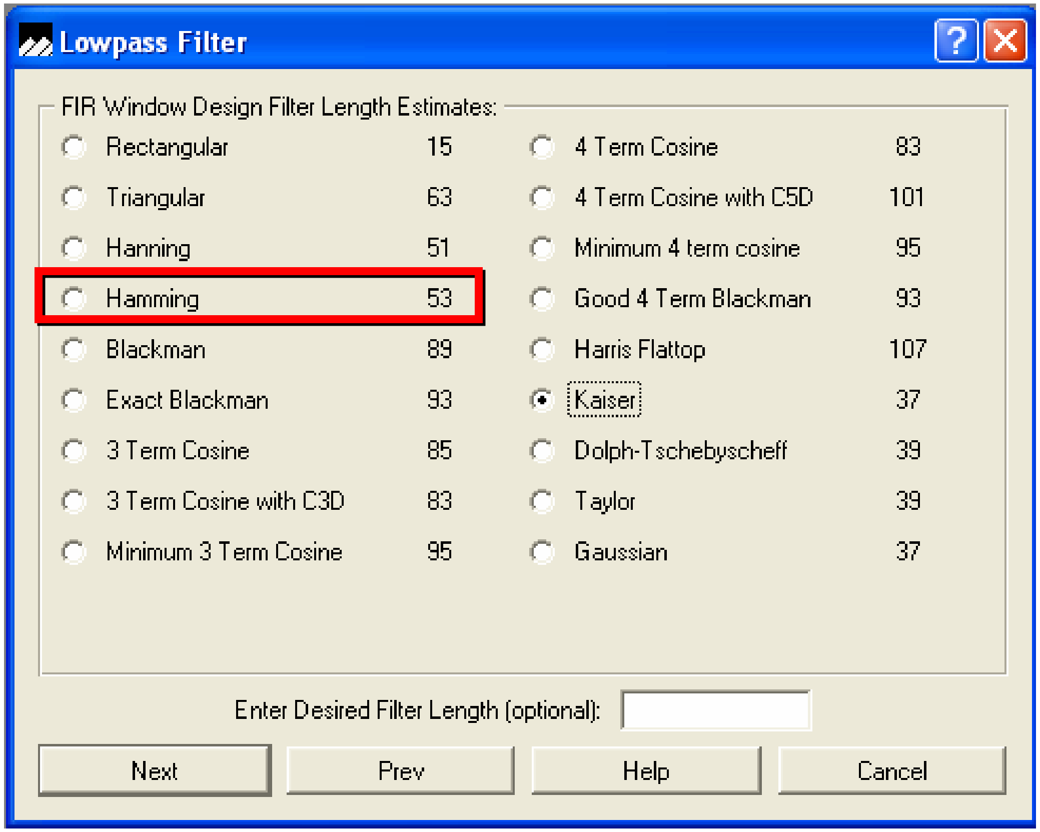
Passband Frequencies:

Stopband Frequencies:

Passband Ripple (dB):

Stopband Ripple (dB):

# Using dsPIC® DSC Filter Design to Generate Filter Coefficients



Lowpass Filter

FIR Window Design Filter Length Estimates:

| | | | | | |
|---|---|---|---|---|---|
| ○ | Rectangular | 15 | ○ | 4 Term Cosine | 83 |
| ○ | Triangular | 63 | ○ | 4 Term Cosine with C5D | 101 |
| ○ | Hanning | 51 | ○ | Minimum 4 term cosine | 95 |
| ○ | Hamming | 53 | ○ | Good 4 Term Blackman | 93 |
| ○ | Blackman | 89 | ○ | Harris Flattop | 107 |
| ○ | Exact Blackman | 93 | ● | Kaiser | 37 |
| ○ | 3 Term Cosine | 85 | ○ | Dolph-Tschebyscheff | 39 |
| ○ | 3 Term Cosine with C3D | 83 | ○ | Taylor | 39 |
| ○ | Minimum 3 Term Cosine | 95 | ○ | Gaussian | 37 |

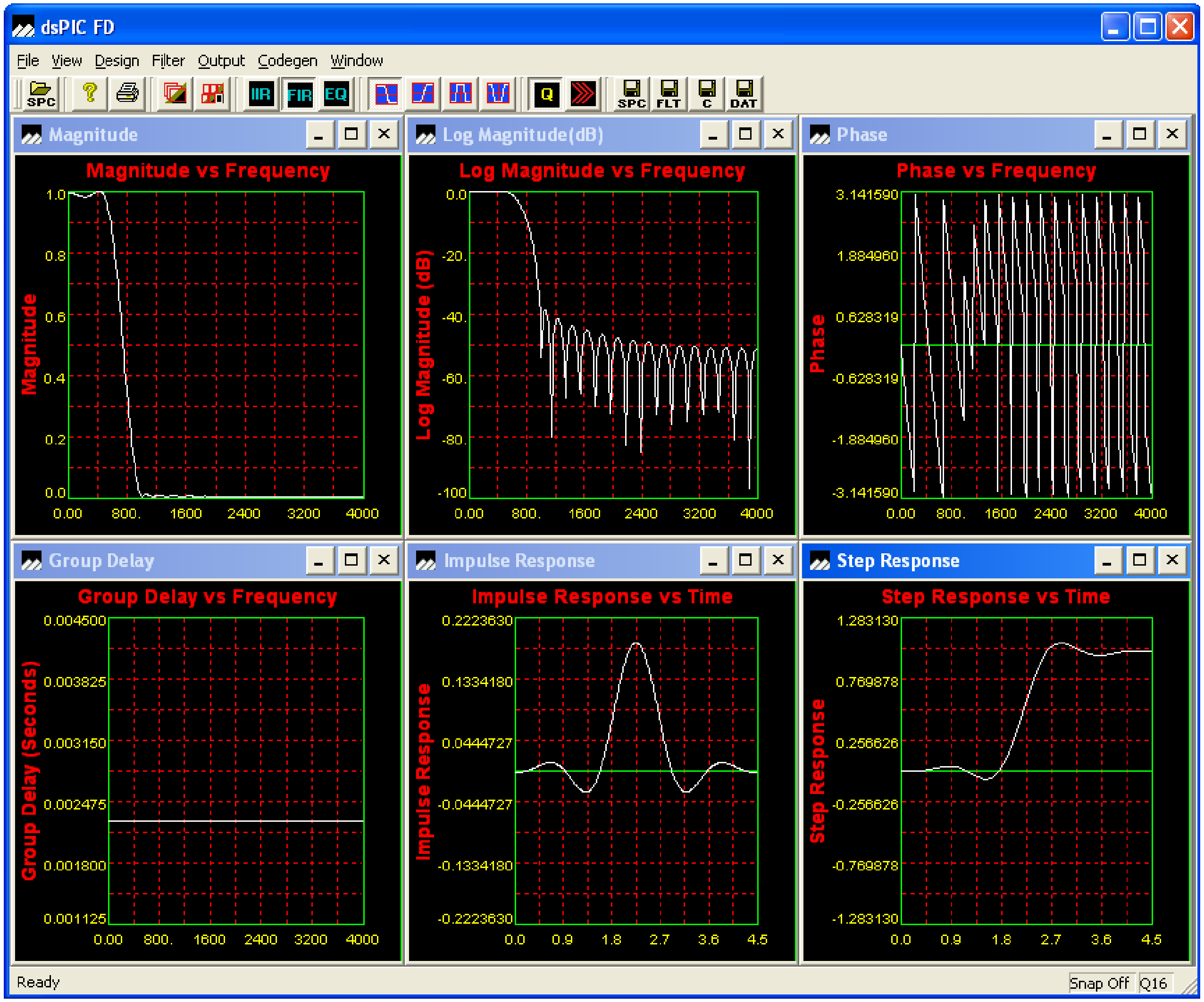
Enter Desired Filter Length (optional):

❺ Select the *Hamming* window

Click Next

Note the estimated filter length of 53 taps.

# Using dsPIC® DSC Filter Design to Generate Filter Coefficients

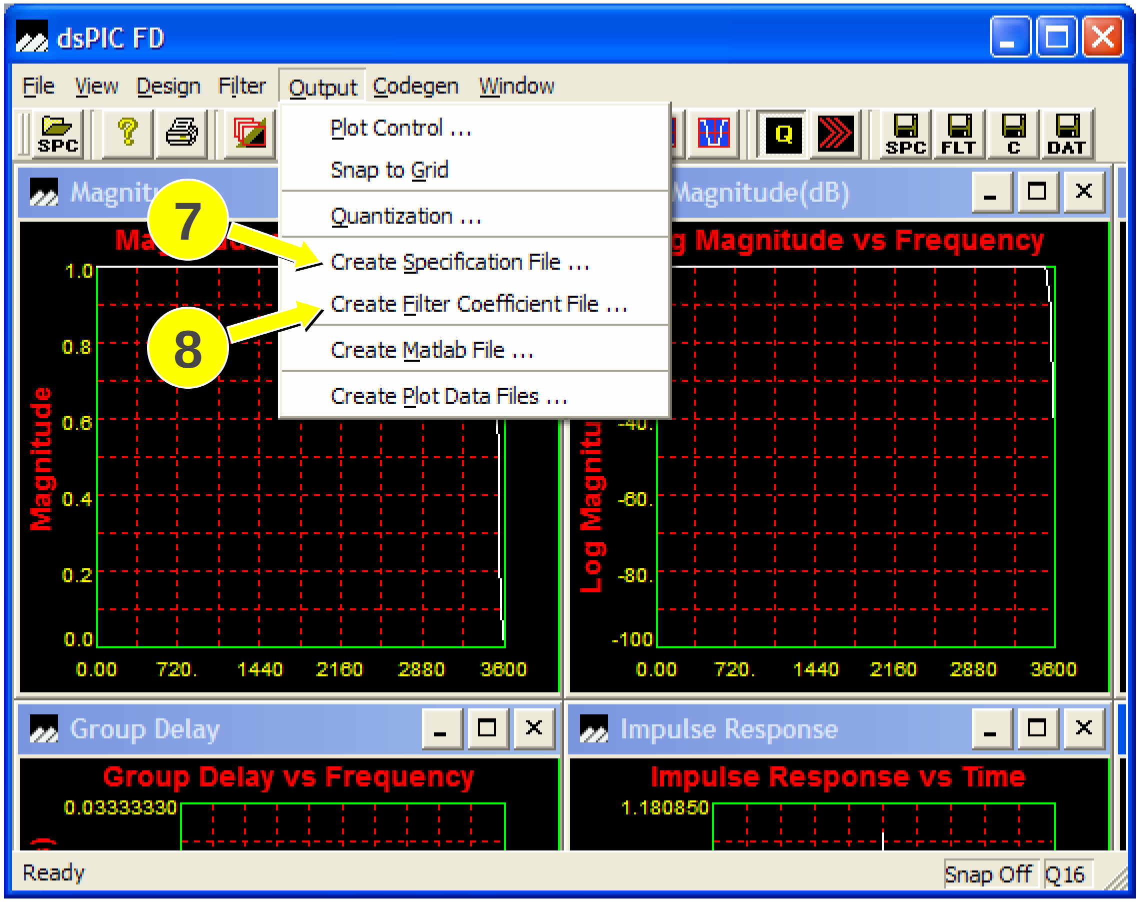


❻ **View graphical display of your filter's characteristics**

Note that in the top two graphs the line of interest (in white) travels along the very top and then down the right side.

There is very little ripple and the transition band is very steep (< 100Hz wide)

# Using dsPIC® DSC Filter Design to Generate Filter Coefficients



❼ Select from the menu:

Output ▶ Create Specification File…

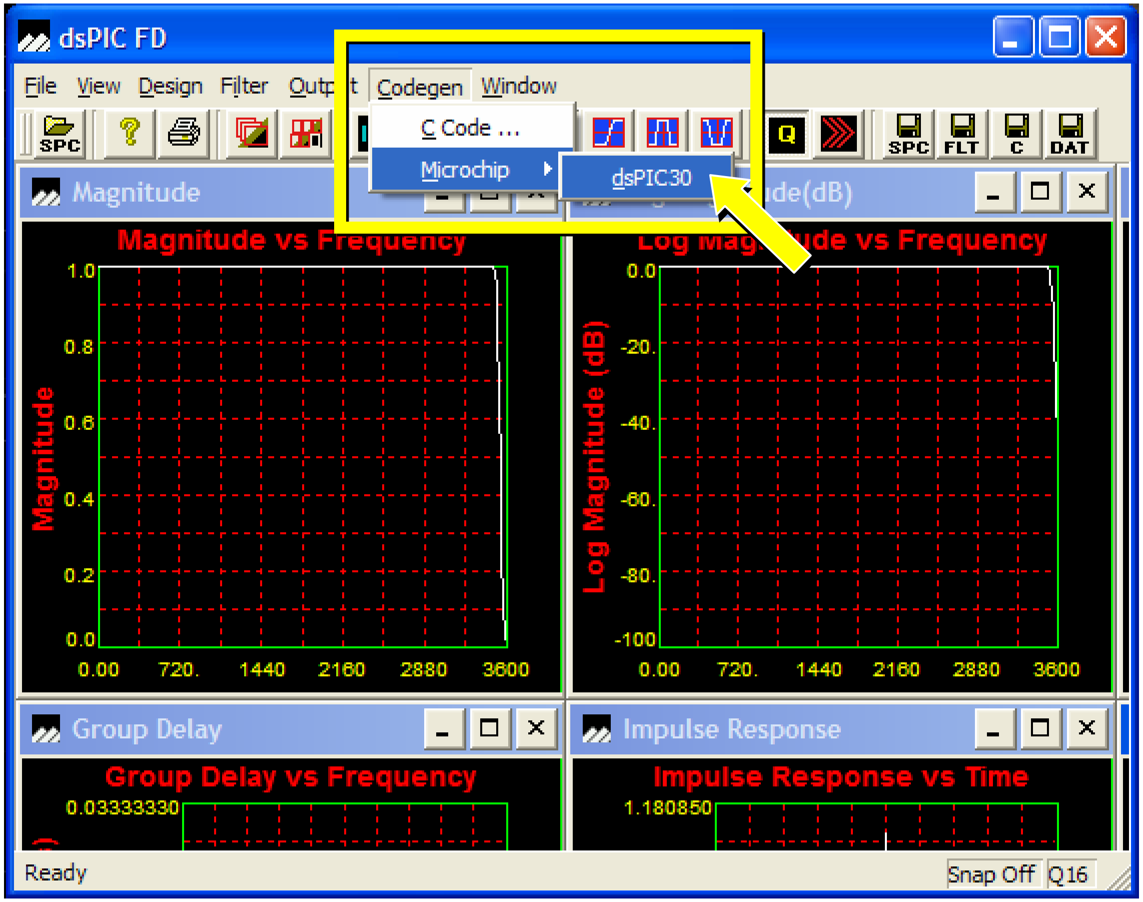
Name the file: `Filter1.spc`
Save the file to the class directory:
`C:\MASTERS\11098_DP3\`

❽ Select from the menu:

Output ▶ Create Filter Coefficient File…

Name the file: Filter1.flt
Save the file to the class directory:
C:\MASTERS\11093_DP3\

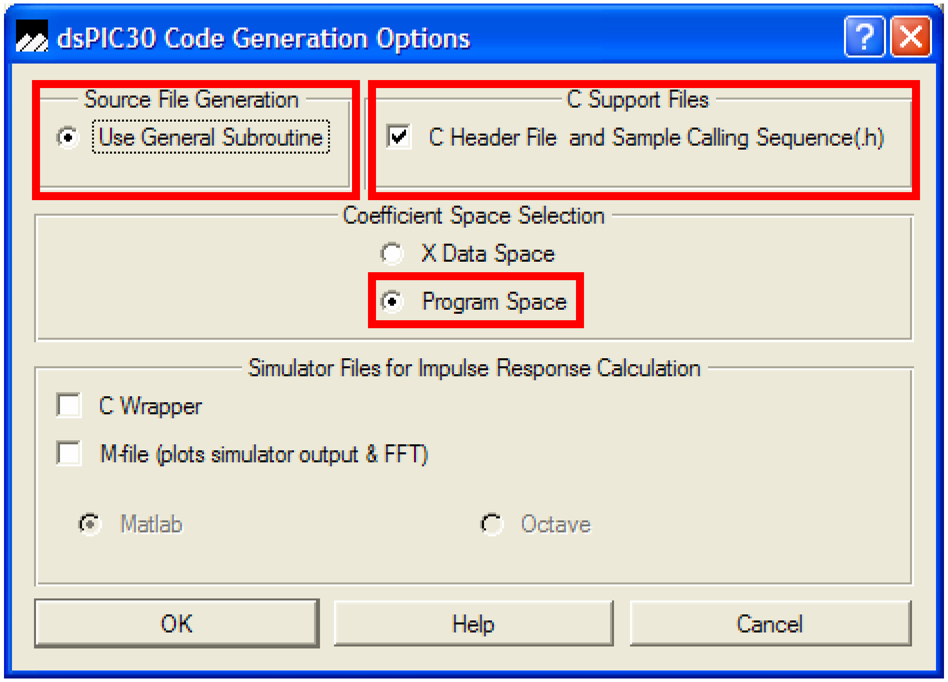
# Using dsPIC® DSC Filter Design to Generate Filter Coefficients



❾ **Select from the menu:**

**Codegen ▶Microchip ▶dsPIC30**

# Using dsPIC® DSC Filter Design to Generate Filter Coefficients



dsPIC30 Code Generation Options

Source File Generation
- ● Use General Subroutine

C Support Files
- ☑ C Header File and Sample Calling Sequence(.h)

Coefficient Space Selection
- ○ X Data Space
- ● Program Space

Simulator Files for Impulse Response Calculation
- □ C Wrapper
- □ M-file (plots simulator output & FFT)
- ● Matlab ○ Octave

## ❿ Select the following options:

- Use General Subroutine
- C Header File and Sample Calling Sequence (.h)
- Program Space

**Click** OK

**Name the file: Filter1**
**Save the file to the class directory:**
**C:\MASTERS\11098_DP3\**

**The program generate two files:**
**lpf.h**
**lpf.s**

# FIR Filter
# DSP Library Interface

## ❑ Filter Data Type

```
typedef struct {
        int numCoeffs;
        fractional* coeffsBase;
        fractional* coeffsEnd;
        int coeffsPage;
        fractional* delayBase;
        fractional* delayEnd;
        fractional* delay;
} FIRStruct;
```

## ❑ Filter Interface

```
extern void FIRDelayInit (FIRStruct* filter);

extern fractional* FIR (int numSamps,
        fractional* dstSamps, fractional* srcSamps,
        FIRStruct* filter);
```

# dsPIC® DSC Filter Design (coefficient in RAM)

- Co-efficient Buffer placement in X RAM

```
;    . section .xdata                      ; Old Syntax
     .section .xdata, data, xmemory       ; New Syntax

xxxxTaps:
    .hword    0x005F, 0x004E, …..0x0100
    .hword    0x02A3, 0x03D8, ….0xE9B7
```

- Delay Buffer placement in Y RAM

```
;   .section .ybss,  "b"                   ; old syntax
    .section .ybss, bss, ymemory          ; new syntax

xxxxDelay:
    .space xxxxNumTaps*2
```

- Filter definition

```
    .section .data
    .global _xxxxFilter
_xxxxFilter:
```

# dsPIC® DSC Filter Design (coefficient in Flash)

- Co-efficient placement in Program memory

```
;   .section .xxxxconst, "x“                ; old syntax
    .section .xxxxconst, code               ; new syntax

xxxxTaps:
    .hword    0x005F, 0x004E, …..0x0100
    .hword    0x02A3, 0x03D8, ….0xE9B7
```

- Delay Buffer placement in Y RAM

```
;   .section .ybss,  "b"                    ; old syntax
    .section .ybss, bss, ymemory            ; new syntax

xxxxDelay:
     .space xxxxNumTaps*2
```

- Filter definition

```
    .section .data
    .global _xxxxFilter
_xxxxFilter:
```

UNIVERSITY OF MICROCHIP
UOFM
MASTERS 2007

# FIR Filter Usage

- ❑ **Reference the Filter data instance**

```
extern FIRStruct xxxxFilter;
```

- ❑ **Initialize the Filter internal states**

```
FIRDelayInit (&xxxxFilter);
```

- ❑ **Invoke the Filter**

```
FIR(1, &output, &input, &xxxxFilter);
```

# Hands-on Lab #1

## Objective

- ❑ Design a filter using dsPIC® DSC Filter Design tool.
- ❑ Use DSP Library to perform the filtering.
- ❑ Test the Digital Filter.

## Procedure

- ❑ Refer to the Hand out.

## Result

- ❑ Vary the frequency in Audio Generator and observe the input and output of the Digital Filter in DMCI window.

# IIR Filter Design

# IIR Filter Design

- Transform an **analog prototype** (Butterworth, Chebyshev, Elliptic) into a **digital filter**
- Supports the **analog features**
- IIR Filter Types:
  - Digital Butterworth Filter is monotonic in both the Pass band and Stop band
  - Digital Chebyshev Filter has ripple in the Pass band but are monotonic in the Stop band (or vice versa)
  - Digital Elliptic Filter has equiripple in both Pass band and Stop band

# Butterworth Filter (5 pole low pass)

- **Pass Band**

  Maximum flat magnitude response in pass-band.

- **Transition Region**

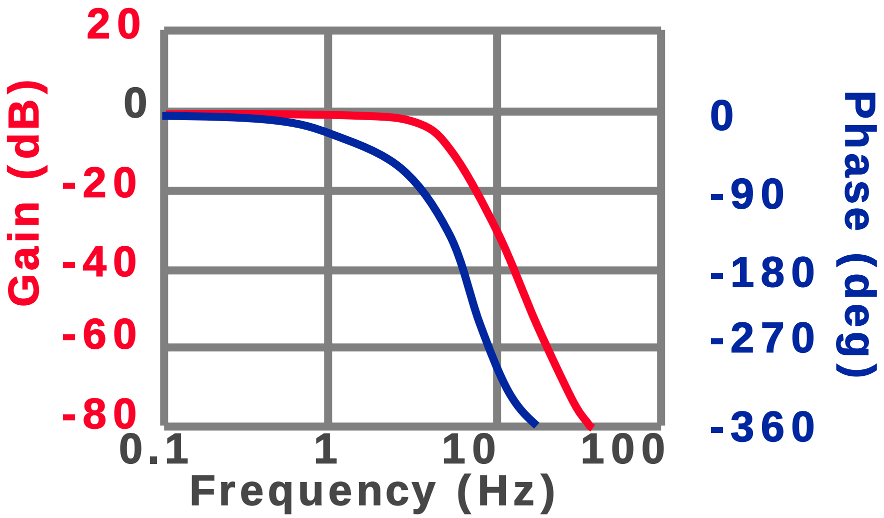
  Steeper than Bessel, not as good as Chebyshev filter.

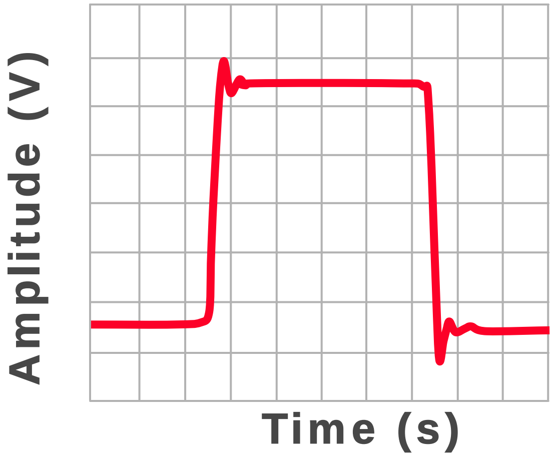
- **Stop Band**

  No ringing.

- **Step Response**

  Some overshoot and ringing, but less than the Chebyshev filter.





# Bessel Filter (5 pole low pass)

- **Pass Band**

  Flat magnitude response in pass-band.

- **Transition Region**

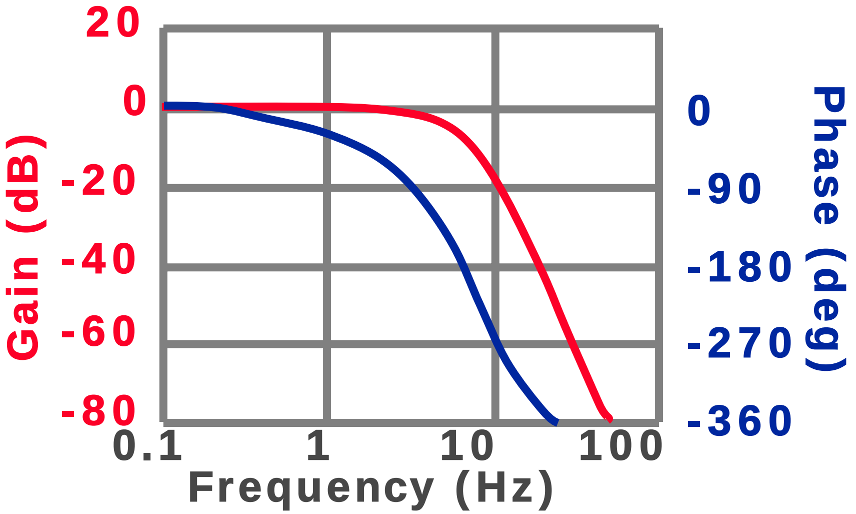
  Slower than the Butterworth or Chebyshev filters.

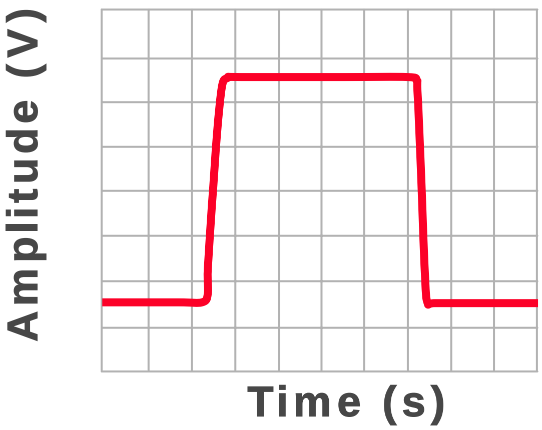
- **Stop Band**

  No ringing.

- **Step Response**

  Very little overshoot or ringing as compared to the Butterworth and Chebyshev filters.





# Chebyshev Filter (5 pole low pass)

- **Pass Band**

  Ripple in the pass-band.

- **Transition Region**

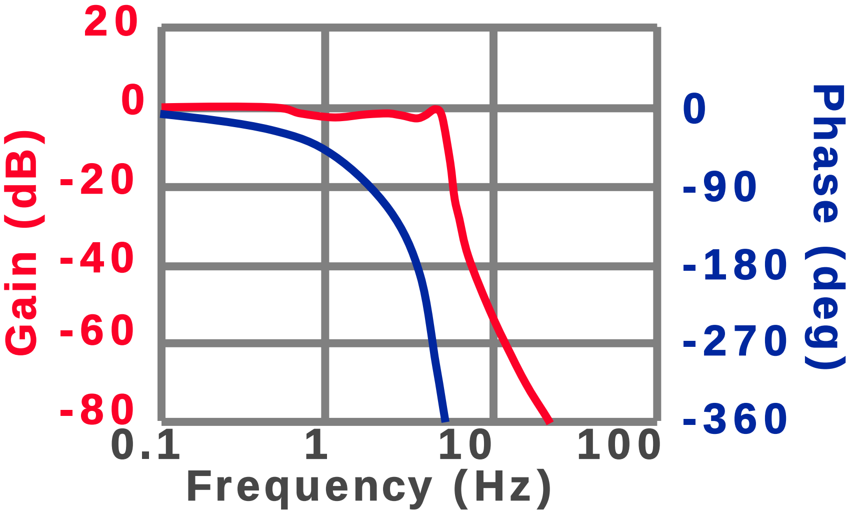
  Steeper than Butterworth and Bessel filters.

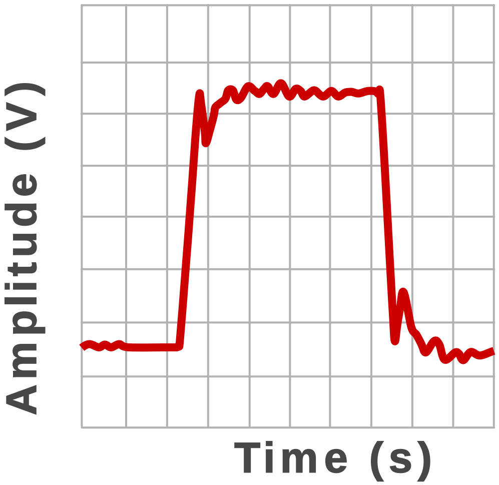
- **Stop Band**

  No ringing.

- **Step Response**

  Fair degree of overshoot and ringing.





UNIVERSITY OF MICROCHIP UOFM MASTERs 2007

# Filter Properties

| Filter Type | Passband | Transition Region | Stopband | Step Response |
|---|---|---|---|---|
| Butterworth | Maximum flat magnitude response in pass-band. | Steeper than Bessel, but not as good as Chebyshev or Inverse Chebyshev filters. | No ringing. | Some overshoot and ringing, but less than the Chebyshev or Inverse Chebyshev filters. |
| Chebyshev | Ripple in the pass-band. | Steeper than Butterworth and Bessel filters, but not as steep as the Inverse Chebyshev filter. | No ringing. | Fair degree of overshoot and ringing, but less than the Inverse Chebyshev filter. |
| Bessel | Flat magnitude response in pass-band. | Slower than the Butterworth, Chebyshev or Inverse Chebyshev filters. | No ringing. | Very little overshoot or ringing as compared to the Butterworth, Chebyshev and Inverse Chebyshev filters. |
| Inverse Chebyshev | Flat magnitude response in pass-band. | Steeper than Butterworth, Bessel, and Chebyshev filters. | More ringing than other filters. | More overshoot and ringing than Butterworth, Bessel, and Chebyshev filters. |

# IIR Filter Implementation

**You can describe IIR filter using any of the following equation:**

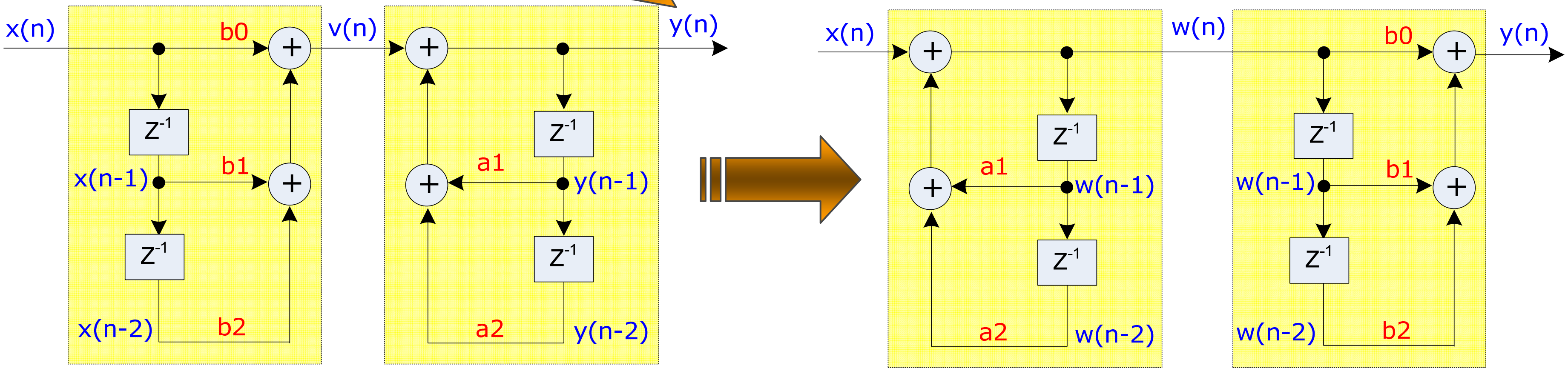
- The difference equation

$$y(n) = -\sum_{k=1}^{N} a_k \times y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

- The transfer function

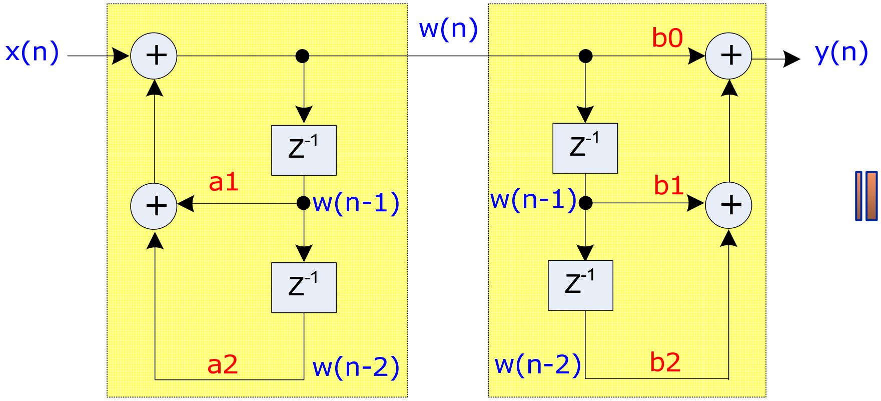
$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} = \left(\frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}\right) \bullet \left(\frac{b'_0 + b'_1 z^{-1} + b'_2 z^{-2}}{a'_0 + a'_1 z^{-1} + a'_2 z^{-2}}\right) \cdots$$

## Direct Form I

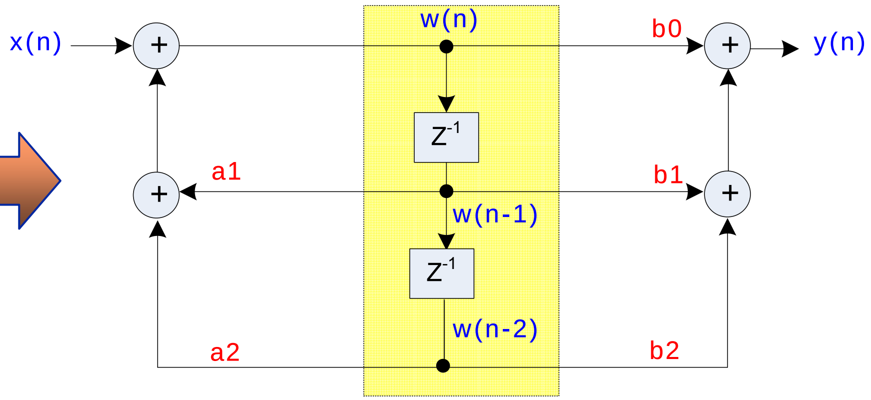


# IIR Filter Implementation

## Direct Form I



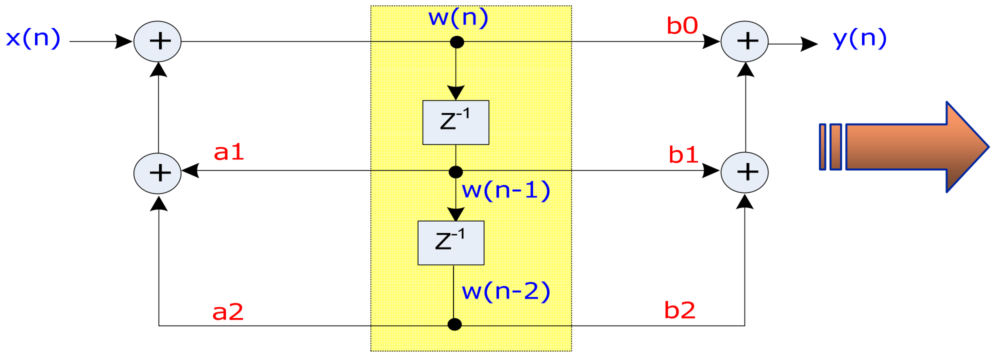
## Direct Form II



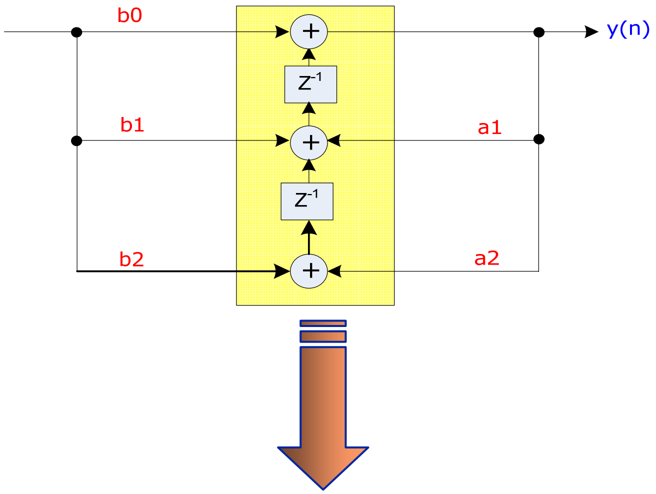
- Theoretically, Direct Form I and II are not different.
- From Implementation perspective:
  - Direct II requires Lesser memory space.
  - Direct Form II requires input scaling.

# IIR Filter Implementation

## Direct Form II

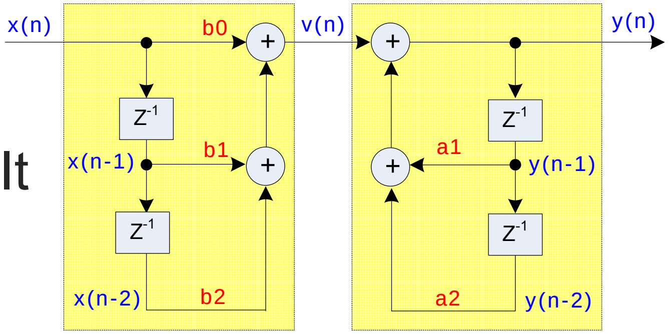


## Transposed Direct Form II



- Theoretically, Direct Form I and Transposed Direct Form II are not different.
- From Implementation perspective:
  - Direct Form II requires input scaling. It reduces SNR.

## Direct Form I



# Transposed Direct Form II
# DSP Library Interface

## ❑ Filter Data Type

```
typedef struct {
        int numSectionsLess1;
        fractional* coeffsBase;  // {b0,b1,a1,b2,a2}
        int coeffsPage;
        fractional* delayBase1;
        fractional* delayBase2;
        int finalShift;
} IIRTransposedStruct;
```

## ❑ Filter Interface

```
extern fractional* IIRTransposed (int numSamps,
        fractional* dstSamps, fractional* srcSamps,
              IIRTransposedStruct* filter);

extern void IIRTransposedInit (IIRTransposedStruct* filter);
```

# Direct Form II

# DSP Library Interface

## ❑ Filter Data Type

```
typedef struct {
        int numSectionsLess1;
        fractional* coeffsBase;  // {a2,a1,b2,b1,b0}        int
coeffsPage;
        fractional* delayBase;
        fractional initialGain;
        int finalShift;
} IIRCanonicStruct;
```

## ❑ Filter Interface

```
extern void IIRCanonicInit (IIRCanonicStruct* filter);

extern fractional* IIRCanonic (int numSamps,
        fractional* dstSamps, fractional* srcSamps,  IIRCanonicStruct*
filter);
```

# dsPIC® DSC Filter Design (Coefficient in RAM)

- Coefficient Buffer placement in X RAM

```
;   .section .xdata                         ; old Syntax
    .section .xdata, data, xmemory          ; new Syntax

xxxxCoefs:
   .hword    0x242E
```

- Delay Buffer placement in Y RAM

```
;  .section .ybss,  "b"                     ; old syntax
   .section .ybss, bss, ymemory             ; new syntax

xxxxStates:
   .space testNumSections*2*2
```

- Filter definition

```
   .section .data
   .global _xxxxFilter
_xxxxFilter:
```

# dsPIC$^{®}$ DSC Filter Design (Coefficient in Flash)

- **Filter coefficient placement in program memory**

```
;   .section .xxxxconst, "x“           ; old syntax
    .section .xxxxconst, code          ; new syntax

xxxxCoefs:
    .hword    0xCE06 ; a( 1,2)/2
```

- **Delay Buffer placement in Y RAM**

```
;   .section .ybss,  "b"                ; old syntax
    .section .ybss, bss, ymemory        ; new syntax

xxxxStates:
    .space testNumSections*2*2
```

- **Filter definition**

```
    .section .data
    .global _xxxxFilter
_xxxxFilter:
```

# IIR Filter Usage

- □ **Reference the Filter data instance**

```
extern IIRCanonicStruct xxxxFilter;

extern IIRTransposedStruct xxxxFilter;
```

- □ **Initialize the Filter internal states**

```
IIRCanonicInit(&xxxxFilter);

IIRTransposedInit(&xxxxFilter);
```

- □ **Invoke the Filter**

```
IIRCanonic(1,  &output, &input, &xxxxFilter);
IIRTransposed(1,  &output, &input, &xxxxFilter);
```

# Hands-on Lab #2

**Objective**

- ❑ Design a filter using dsPIC Filter Design tool.
- ❑ Use DSP Library to perform the filtering.
- ❑ Test the Digital Filter.

**Procedure**

- ❑ Refer to the Hand out.

**Result**

- ❑ Vary the frequency in Audio Generator and observe the input and output of the Digital Filter in DMCI window.

UNIVERSITY OF MICROCHIP
U OF M
MASTERs 2007

# FFT Application and Result Interpretation

# Fast Fourier Transform

DFT Equation: $$X(k)=\sum_{n=0}^{N-1} x(n)\times w_N^{kn}$$ Where, $w_N^{kn} = e^{-j2\pi nk/N}$

Complexity:

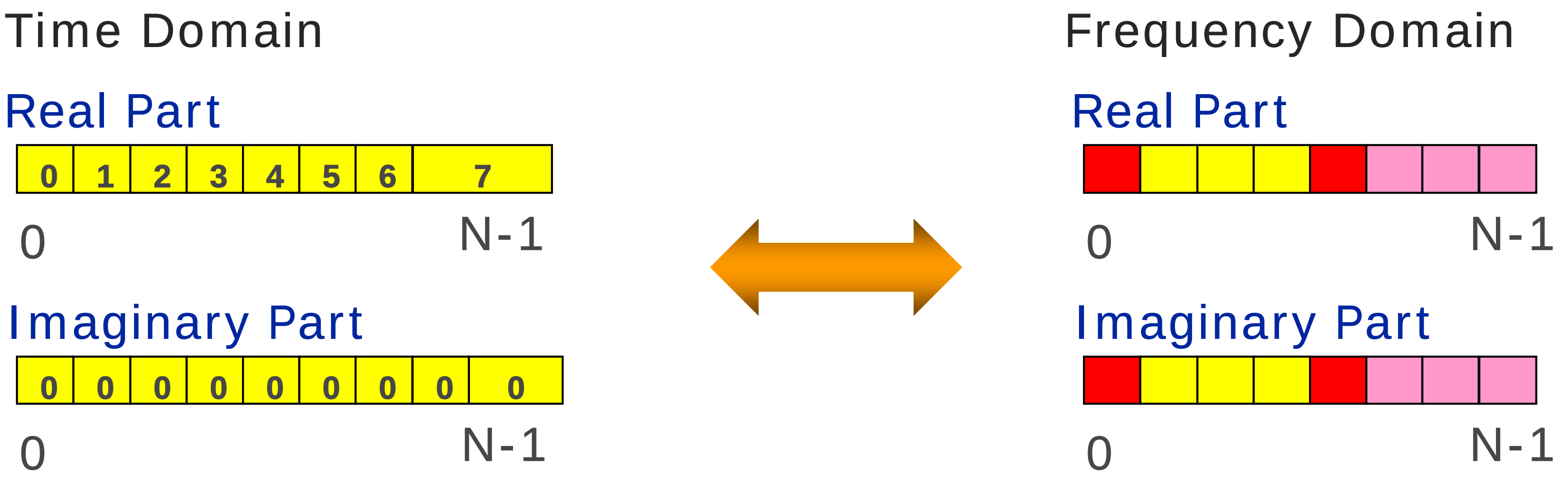
Complex Addition $\left(N/2\right)\times\log_2(N)$

Complex Multiplication $(N)\times\log_2(N)$

| N | DFT | | FFT | |
|---|---|---|---|---|
| | Multiplication | Additions | Multiplication | Additions |
| 128 | 16,384 | 16,256 | 448 | 896 |
| 256 | 65,536 | 65280 | 1,024 | 2048 |
| 512 | 262,144 | 261,632 | 2,304 | 4608 |

# Real/Complex FFT

## N point Complex FFT



Time Domain

Real Part

0 1 2 3 4 5 6 7

0 N-1

Imaginary Part

0 0 0 0 0 0 0 0 0

0 N-1

Frequency Domain

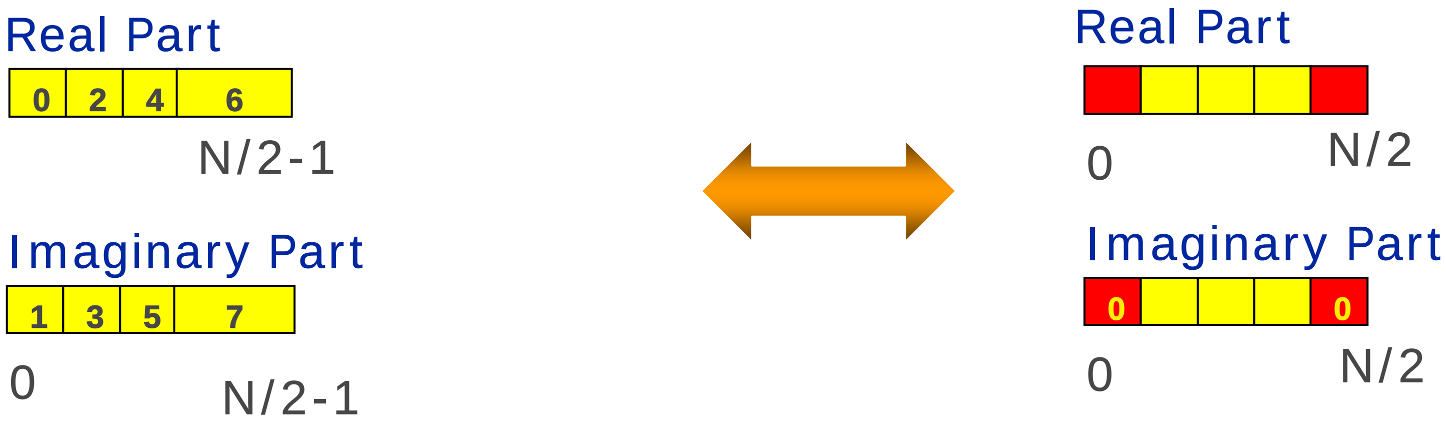
Real Part

0 N-1

Imaginary Part

0 N-1

## N point Real FFT



Real Part

0 2 4 6

N/2-1

Imaginary Part

1 3 5 7

0 N/2-1

Real Part

0 N/2

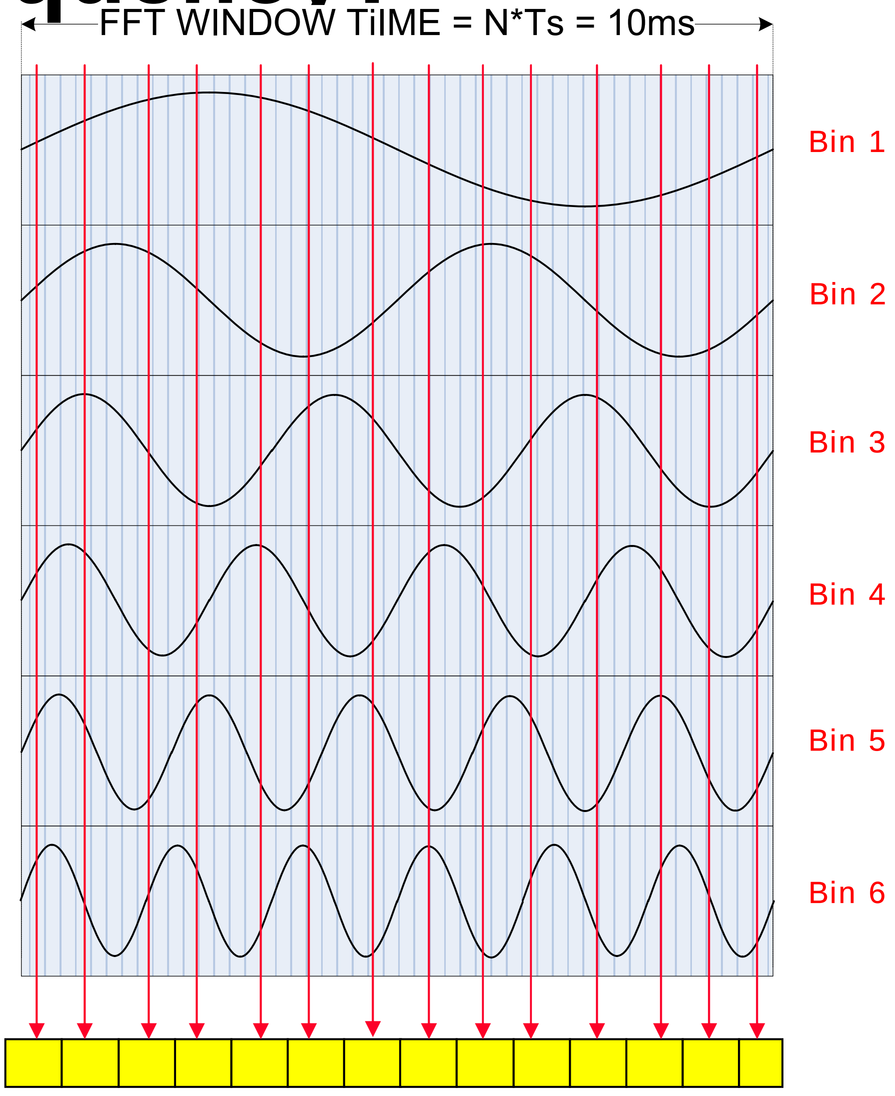
Imaginary Part

0 0

0 N/2

# How the bins are related to signal frequency?

- Frequency resolution $= \dfrac{1}{N \times T_S} = \dfrac{F_S}{N}$
- For example:
  - N=128 point FFT
  - Fs=1280Hz sampling frequency
  - Frequency resolution=10Hz

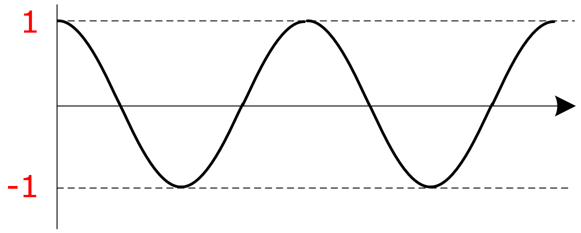


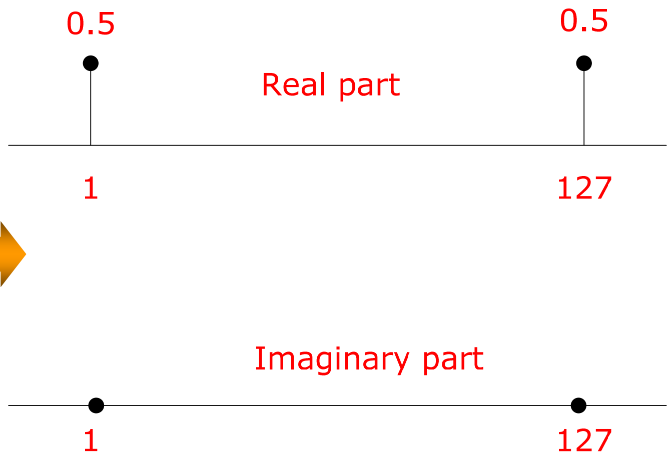
FFT Buffer

# Why FFT bin is complex number?

$$\cos(\omega t) = \frac{e^{j\omega t} + e^{-j\omega t}}{2}$$

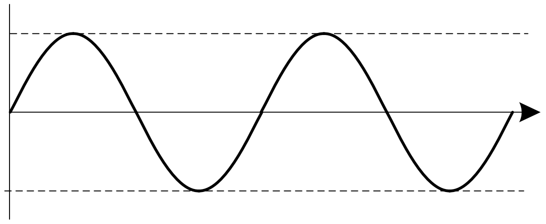
$\cos(2\times\pi\times10)$

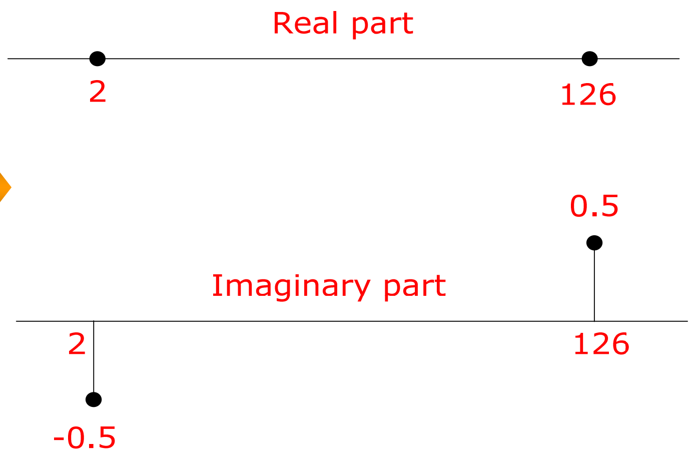




$$\sin(\omega t) = \frac{e^{j\omega t} - e^{-j\omega t}}{2\cdot j} = j\times\left(\frac{-e^{j\omega t} + e^{-j\omega t}}{2}\right)$$

$\sin(2\times\pi\times20)$



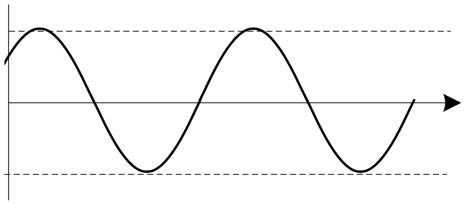


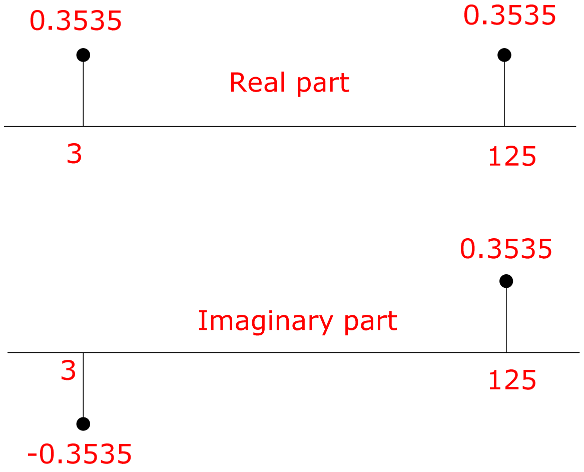
# Why FFT bin is complex number?

$$\sin(\omega t+45)=\sin\left(\omega t\right)\times\cos\left(45\right)+\cos\left(\omega t\right)\times\sin\left(45\right)$$

$$=0.707\times\sin\left(\omega t\right)+0.707\times\cos\left(\omega t\right)$$

$$\sin(\left(2\times\pi\times30\right)+45)$$



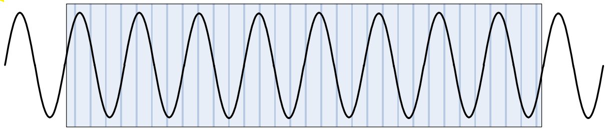


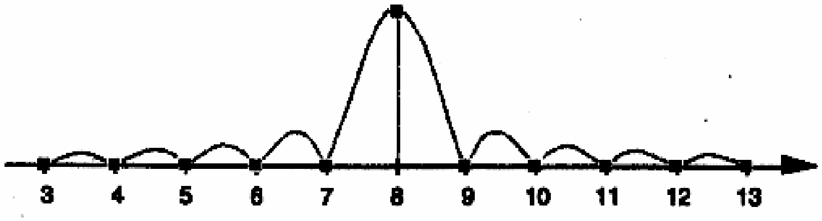
# Spectrum Leakage in FFT

80 Hz

**Signal continuous periodically beyond data window**

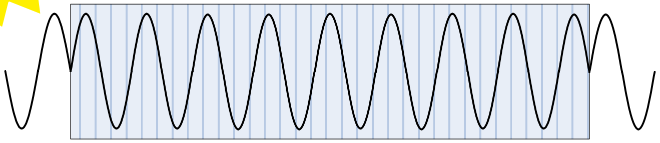
8 cycles

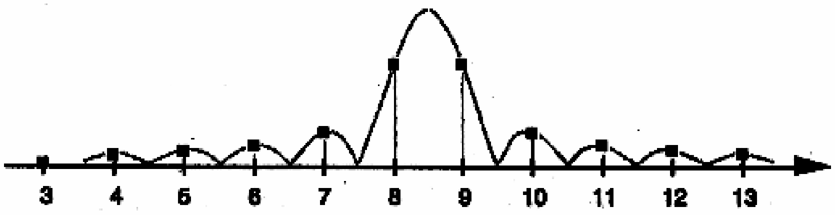




85 Hz

8.5 cycles



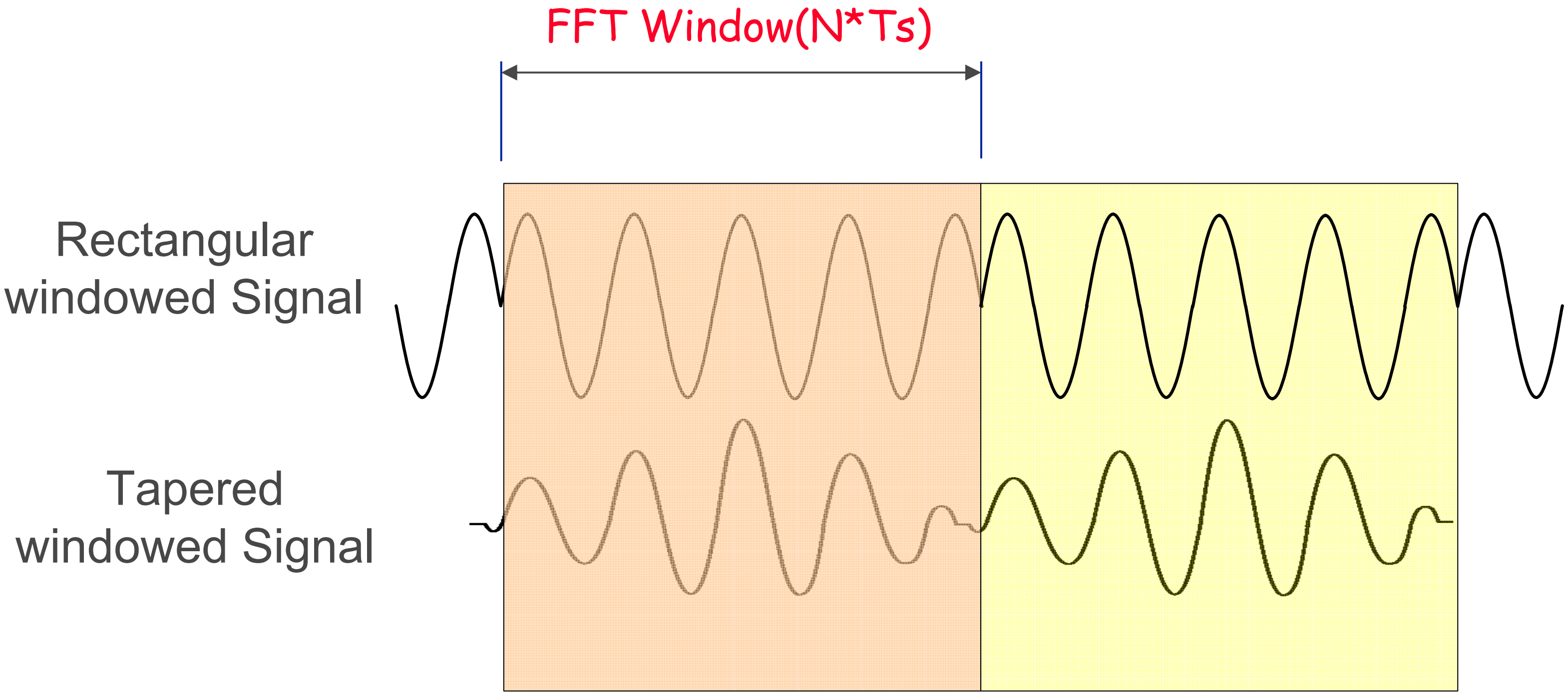


Spectral components associated with frequencies between two successive frequency bins propagate to all bins.

# How to minimize leakage

Windowing reduces:

- Leakage by minimising sidelobes magnitude.
- End-points discontinuities, in time domain.

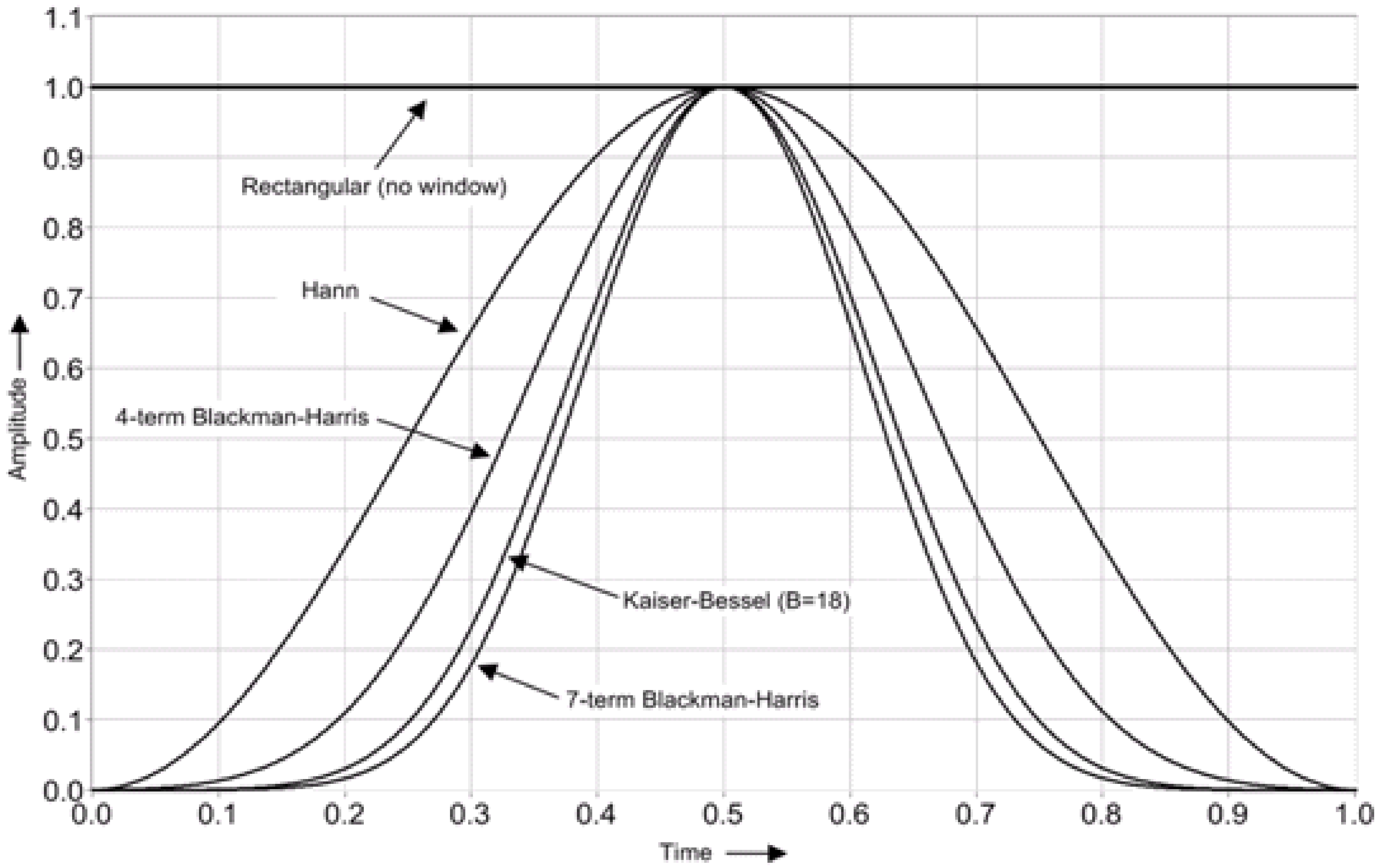


FFT Window(N*Ts)

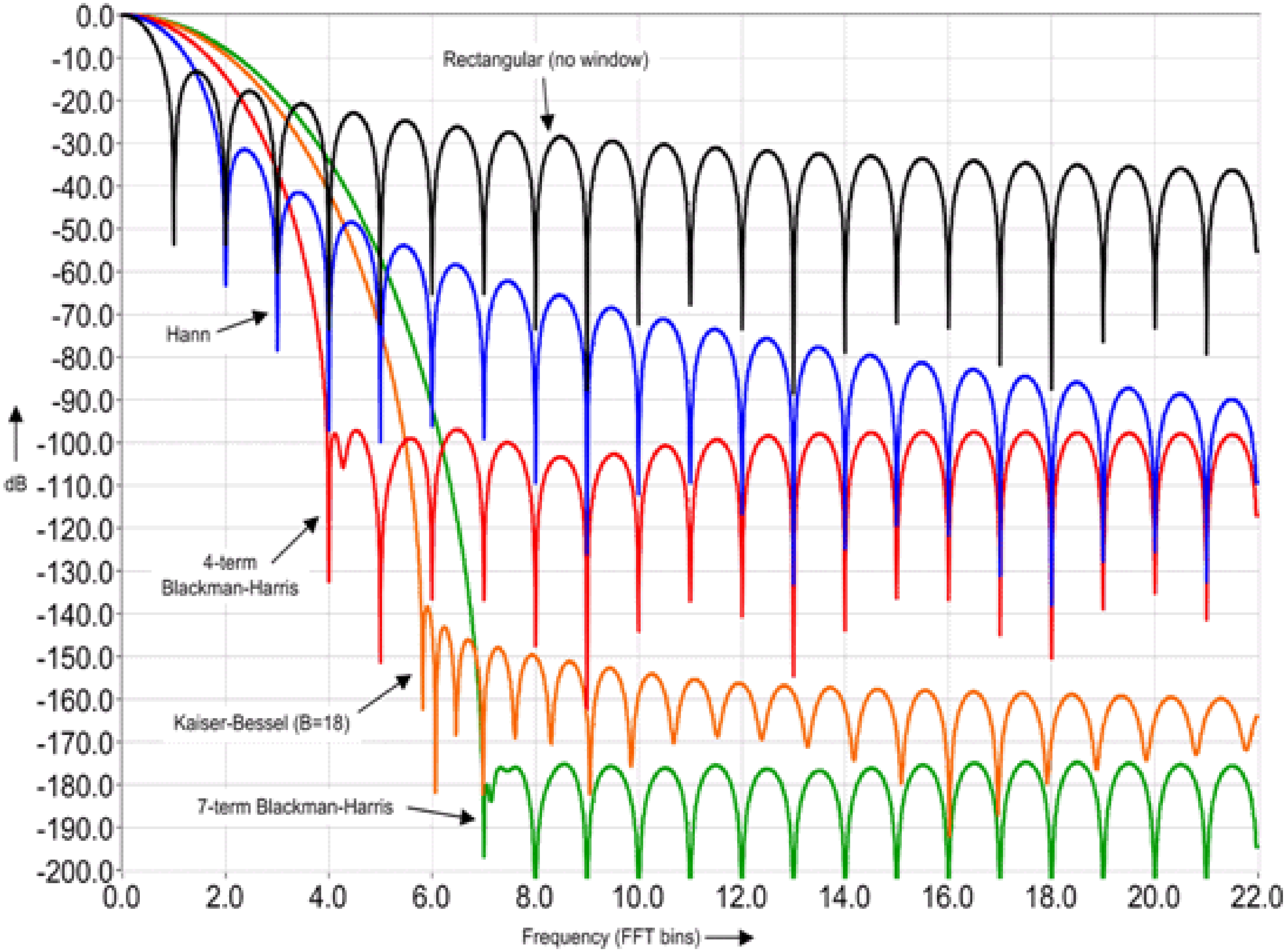
Rectangular windowed Signal

Tapered windowed Signal

# Windows Time domain plot



# Windows Frequency domain plot



# Real FFT usage

- **N-Point real valued sequence in contiguous locations**
- **Bit reverse the N/2 point complex input**

  void realBrev16b(int N, int *buffer);

- **Performs N/2 point complex FFT**

  void cplxFft16b(int logN, int *Buffer, int *twiddle, int twiddlePsv);

- **Process real FFT split function**

  void realFft16b(int logN, int *Buffer, int *twiddle, int twiddlePsv);

# Real FFT Configuration

- **Configures the following parameters in FFT16B.H file**

```
#define PSV_TF            1      // 1 - Twiddle Factor in PSV
#define REAL_N            128    // Specify real FFT size N
#define REAL_LOGN         6      // FFT = LOG2(N)/2
```

- **Refer to Twiddle Factor Buffer**

| | |
|---|---|
| Twiddle Factor Buffer: | psvTwdlFctr16b |
| Twiddle Factor Buffer Page: | __builtin_psvpage(&psvTwdlFctr16b) |

# Hands-on Lab #3

## Objective

- Use 16-bit real FFT function to perform spectrum analysis
- Generate basic test vectors to test real FFT using dsPICworks™
- Interpret the FFT results

## Procedure

- Refer to the hand out

## Result

- Check the FFT result with basic test vectors

# DSP Library Functions

- DSP Library functions has 49 functions.
  - Vector Functions (13 functions)
  - Matrix Functions (6 functions)
  - Filtering Functions (15 functions)
  - Transform Functions (9 functions)
  - Windowing Functions (6 functions)

# DSP Library Functions

## Vector Functions

1. VectorAdd ( )
2. VectorConvolve ( )
3. VectorCopy ( )
4. VectorCorrelate ( )
5. VectorDotProduct ( )
6. VectorMax ( )
7. VectorMin ( )
8. VectorMultiply ( )
9. VectorNegate ( )
10. VectorPower ( )
11. VectorScale ( )
12. VectorSubtract ( )
13. VectorZeroPad ( )

## Matrix Functions

1. MatrixAdd ( )
2. MatrixInvert ( )
3. MatrixMultiply ( )
4. MatrixScale ( )
5. MatrixSubtract ( )
6. MatrixTranspose( )

# DSP Library Functions

## Filter Functions

1. FIR ( )
2. FIRDecimate ( )
3. FIRDelayInit ( )
4. FIRInterpolate ( )
5. FIRInterpDelayInit ( )
6. FIRLattice ( )
7. FIRLMS ( )
8. FIRLMSNorm ( )
9. FIRStructInit ( )
10. IIRCanonic ( )
11. IIRCanonicInit ( )
12. IIRLattice ( )
13. IIRLatticeInit ( )
14. IIRTransposed ( )
15. IIRTransposedInit ( )

## Transform Functions

1. BitReverseComplex ( )
2. CosFactorInit ( )
3. DCT ( )
4. DCTIP ( )
5. FFTComplex ( )
6. FFTComplexIP ( )
7. IFFTComplex ( )
8. IFFTComplexIP ( )
9. TwidFactorInit ( )

## Windowing Functions

1. BartlettInit ( )
2. BlackmanInit ( )
3. HammingInit ( )
4. HanningInit ( )
5. KaiserInit ( )
6. VectorWindow ( )

# Summary

- dsPIC® DSC Filter Design tool to design Digital filters.

- Using DSP library functions to implement Digital Filters

- Using DSP library functions to perform spectrum analysis

Thank You

# Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KeeLoq, KeeLoq logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.