

14 Tratamiento digital de señales

El tratamiento digital de señales es una línea de la tecnología demasiado extensa para ser tratada en un solo capítulo; sin embargo, este capítulo se centra en las nociones de adquisición, tratamiento y reconstrucción de señales. Cabe denotar que el alcance de los microcontroladores PICMicro de baja gama como los 16F y 18F sólo permiten realizar tratamientos sobre señales de baja frecuencia. De todos modos, muchas aplicaciones son de utilidad bajo estas características tales como sistemas de comunicación, generación de tonos DTMF y adquisición de señales bioeléctricas, entre otras. Si el desarrollador desea trabajar y tratar señales de mayor espectro, realizando tratamientos de mayor complejidad será necesario trabajar con unidades de procesamiento más robustas como la familia de microcontroladores PICMicro ds. Estos microcontroladores no pueden ser programados con el compilador MikroC PRO, para estos se debe utilizar el compilador MikroC PRO para dsPIC.

Un diagrama en bloques del procesamiento digital de una señal se puede apreciar en la siguiente figura:

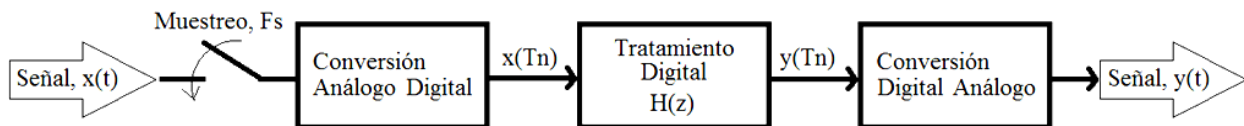


Figura 14-1

14.1 Muestreo

Como primera medida se requiere adquirir muestras periódicamente de la señal analógica, esto se logra haciendo una conversión analógica digital, con una frecuencia constante denominada frecuencia de muestreo F_s . El periodo de muestreo T , es el tiempo en segundos que existe entre muestra y muestra, y a su vez es el inverso de la frecuencia de muestreo $T = 1/F_s$. El muestreo permite convertir una señal continua en una señal discreta, es decir que pasa de $x(t)$ a $x(Tn)$, donde T es el periodo de muestreo, y n es el número de la muestra digitalizada. Para definir el periodo de muestreo T , o lo que es igual la frecuencia de muestreo F_s , se debe tener la siguiente consideración: La frecuencia de muestreo debe ser mayor que la máxima componente espectral contenida en la señal $x(t)$. Por ejemplo, si la máxima frecuencia contenida en la señal $x(t)$ es 500Hz, la frecuencia de muestreo debe ser como mínimo el doble, es decir $F_s = 2 \times 500\text{Hz} = 1000\text{Hz}$. Este concepto es conocido como teorema de muestreo de Nyquist. Sin embargo la frecuencia de muestreo puede ser mayor al doble del máximo en frecuencia de la señal $x(t)$, de hecho cuanto mayor sea la frecuencia de muestreo mayor será la fidelidad de la señal digitalizada, pero a su vez mayor serán los recursos requeridos en velocidad, y memoria de procesamiento.

$$F_s \geq 2F_{MAX} \quad \text{Ecuación 14-1}$$

14.2 Funciones de transferencia

Una función de transferencia es la relación que existe entre la entrada de una señal y su salida al pasar por el bloque de proceso. Las funciones de transferencia se estudian y se manipulan en términos de la frecuencia compleja y no en el dominio del tiempo continuo. Esto se debe a que al convertir las funciones a la frecuencia compleja las operaciones que implican manipulación de números complejos y sistemas integro diferenciales que se hacen lineales, y esto simplifica su tratamiento. Las funciones de transferencia para el caso de los sistemas de tiempo discreto se hacen en términos de la variable compleja z , para el tratamiento de estos sistemas se recurre a la transformación z , que representa la frecuencia compleja para el tiempo discreto. Las funciones de transferencia se denotan por excelencia con la letra H , para las funciones en términos de la variable z , y con h , para las funciones en términos del tiempo discreto T_n . A continuación se puede ver un ejemplo:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{A}{B} \quad \text{Ecuación 14-2}$$

De la ecuación anterior se puede deducir que:

$$H(z)X(z) = Y(z) \quad \text{Ecuación 14-3}$$

Como se puede ver en la anterior ecuación la salida Y , de un sistema discreto es igual a la entrada X , multiplicada por la función de transferencia H . Se debe recordar que el tratamiento real y la programación se realizan en términos del tiempo discreto, por esta razón siempre se deberá realizar la transformada inversa de z . La transformación inversa de la ecuación anterior da como resultado la siguiente ecuación en función del tiempo discreto T_n :

$$h(n) * x(n) = y(n) \quad \text{Ecuación 14-4}$$

De la anterior ecuación se puede observar que se omite la escritura del periodo T , dado que este es un valor constante, y no tiene relevancia escribirlo siempre. Por otro lado se puede observar que la función h y la señal x , no se multiplican como en la frecuencia compleja, en este caso al realizar la transformación su equivalencia es la convolución. Para el caso del tratamiento de señales, la longitud de $h(n)$, es finita, y el número máximo que asume n se denomina M , este a su vez determina el orden de la función de transferencia. Por otra parte la señal de entrada $x(n)$, es una serie de muestras de longitud indeterminada, igual que la salida $y(n)$.

De lo anterior se puede concluir que para realizar un tratamiento digital sobre una señal se debe realizar la operación de convolución continua entre los coeficiente de la función $h(n)$ y la señal $x(n)$. Sin embargo la operación de convolución puede ser de dos formas, una en la cual la convolución se hace con las muestras pasadas y actuales de la señal $x(n)$, esto se conoce como sistema sin memoria o sistemas FIR (Respuesta Finita al Impulso). Por otra parte cuando la convolución requiere información de las muestras pasadas de $x(n)$, y de $y(n)$, estos sistemas se denominan sistemas con memoria o IIR (Respuesta Infinita al Impulso). Los sistemas IIR son sistemáticamente más simples, en su implementación y requieren de campos de memoria pequeños, por otra parte realizar su diseño y lograr su estabilidad es más complicado y requiere de tediosos procesos matemáticos. Los sistemas FIR, ofrecen características contrarias a los sistemas IIR, los sistemas FIR, son de fácil implementación y diseño, pero consumen recursos de

memoria y procesamiento mayor, una de las virtudes de estos sistemas es que siempre son estables. Los análisis y ejemplos de este capítulo se concentrarán en los sistemas FIR e IIR.

14.3 Convolución

Para iniciar es importante conocer la estructura de una convolución continua en forma matemática y por medio de un segmento de código en lenguaje C:

$$y(n) = \sum_{k=-\infty}^{+\infty} [h(k)x(n-k)] \quad \text{Ecuación 14-5}$$

La ecuación anterior describe la forma general de la convolución, sin embargo se debe recordar que la longitud de la función $h(n)$, es finita y su máximo es M , por lo tanto la ecuación se puede reescribir de la siguiente forma:

$$y(n) = \sum_{k=0}^M [h(k)x(n-k)] \quad \text{Ecuación 14-6}$$

Cada vez que una muestra de la salida $y(n)$, es calculada por medio de la convolución, se requieren M , muestras de la señal $x(n)$, incluida la muestra actual, esto quiere decir que para hacer la convolución se debe tener presente la necesidad de un campo de memoria igual a M para guardar las últimas muestras durante el proceso.

A continuación se puede observar un ejemplo de cómo se implementar una convolución, en lenguaje C:

```
//Orden del sistema FIR de orden 17, M=17.
#define M 17
float x[M];
float h[M];

//Rutina para hacer la convolución.

float yn=0.0;
short k;
for( k=M-1; k>=1; k-- )x[n]=x[n-1];
x[0]=x0;
for( k=0; k<M; k++ )yn += h[k]*x[k];
```

14.4 Filtros FIR

El diseño de los filtros FIR, es relativamente simple y utiliza estructuras ya definidas para cada tipo de filtro. Los filtros pueden ser de cinco naturalezas: Pasa bajas, pasa altas, pasa bandas, rechaza banda, y multi banda. Para el tratamiento de los filtros se debe tener presente las siguientes consideraciones:

Los cálculos de los filtros se hacen en radianes y no en hercios.

La frecuencia de muestreo en hercios F_s , es equivalente a una frecuencia en radianes/segundo de 2π .

La máxima frecuencia tratada por los filtros es $F_s/2$, o π Radianes.

La relación que existe entre radianes y hercios es: $\omega = 2\pi F$, y $F = \omega/2\pi$.

Las frecuencias de corte de los filtros se denotan como ω_c , y F_c .

La frecuencia de corte se calcula como $\omega_c = 2\pi F_c / F_s$.

La banda de transición del filtro se denota como $d\omega$ en radianes/segundo y dF en hercios.

El orden de los sistemas FIR se denota como M .

Se recomienda usar un número impar para M .

A continuación se estudiará la forma de diseño para cada uno de estos filtros.

14.4.1 Filtro Pasa bajas

Los filtros pasa bajas se caracterizan por tener una frecuencia de corte a partir de la cual las frecuencias inferiores son permitidas, y las frecuencias superiores son atenuadas. La función de transferencia $h(n)$, para este filtro es:

$$h(n) = \begin{cases} \frac{\text{Sen}(\omega_c n)}{\pi n}, & n \neq 0 \\ \frac{\omega_c}{\pi}, & n = 0 \end{cases} \quad \text{Ecuación 14-7}$$

$$-\frac{M}{2} < n < \frac{M}{2}$$

La respuesta espectral de este filtro de orden 7, en escala lineal es la siguiente:

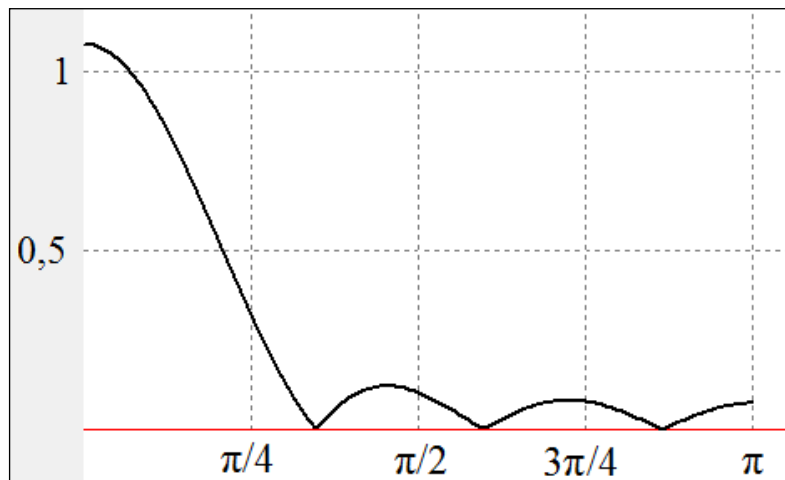


Figura 14-2

14.4.2 Filtros Pasa Altas

Los filtros pasa altos permiten el paso de las frecuencias superiores a la frecuencia de corte, y suprimen las inferiores a la misma. El cálculo de la función de transferencia $h(n)$, para estos filtros está definida por la siguiente ecuación:

$$h(n) = \begin{cases} \frac{-\text{Sen}(w_c n)}{\pi n}, & n \neq 0 \\ 1 - \frac{w_c}{\pi}, & n = 0 \end{cases} \quad \text{Ecuación 14-8}$$

$$-\frac{M}{2} < n < \frac{M}{2}$$

La respuesta espectral en escala lineal de este filtro en orden 17, es la siguiente:

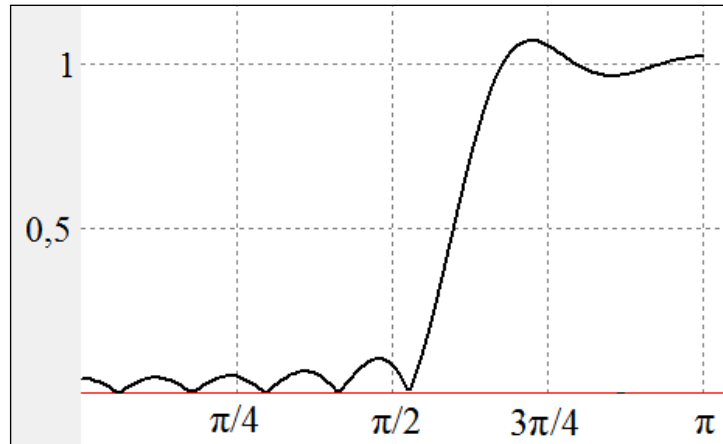


Figura 14-3

14.4.3 Filtro Pasa Banda

Los filtros pasa banda permiten el paso de una porción de frecuencias entre W_{c1} y W_{c2} , y el resto de frecuencias son eliminadas. La función de transferencia para este tipo de filtros se calcula por medio de la siguiente ecuación:

$$h(n) = \begin{cases} \frac{\text{Sen}(w_{c2}n) - \text{Sen}(w_{c1}n)}{\pi n}, & n \neq 0 \\ \frac{w_{c2} - w_{c1}}{\pi}, & n = 0 \end{cases} \quad \text{Ecuación 14-9}$$

$$-\frac{M}{2} < n < \frac{M}{2}$$

La respuesta espectral en escala lineal de este filtro en orden 17, es la siguiente:

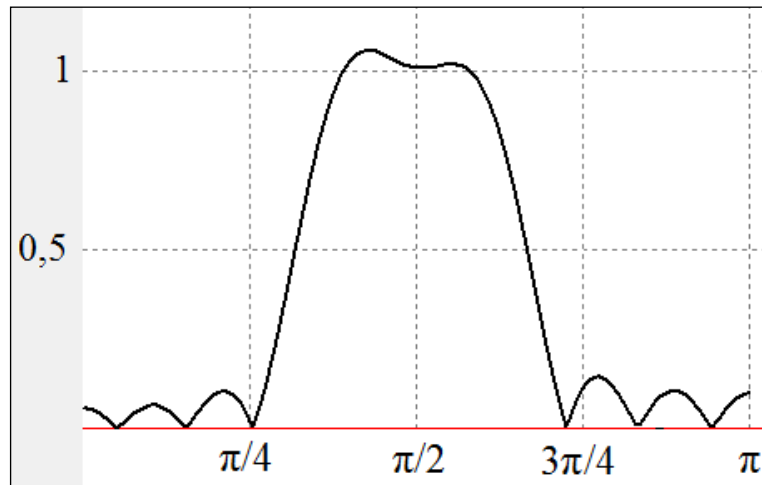


Figura 14-4

14.4.4 Filtro Rechaza Banda

Los filtros rechaza banda tienen un comportamiento similar a los filtros pasa banda, y su proceder es inverso a los filtros pasa banda, la función que caracteriza los coeficientes de la función de transferencia es la siguiente:

$$h(n) = \begin{cases} \frac{\text{Sen}(w_{c1}n) - \text{Sen}(w_{c2}n)}{\pi n}, & n \neq 0 \\ 1 - \frac{w_{c2} - w_{c1}}{\pi}, & n = 0 \end{cases}$$

Ecuación 14-10

$$-\frac{M}{2} < n < \frac{M}{2}$$

La respuesta espectral en escala lineal de este filtro en orden 17, es la siguiente:

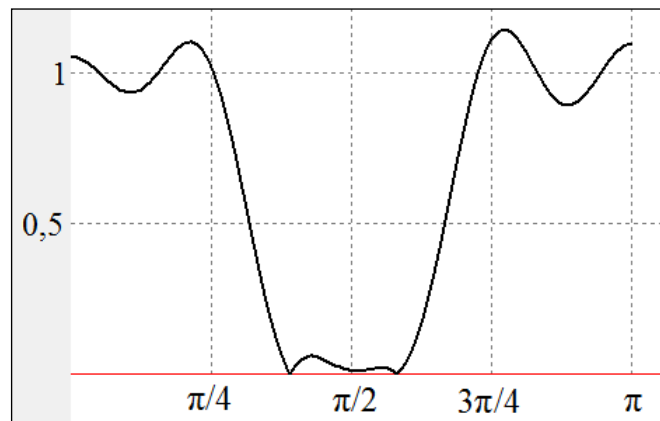


Figura 14-5

14.4.5 Filtro Multi Band

Los filtros multi banda son arreglos que integran diferentes filtros en uno solo, estos pueden dar ganancias arbitrarias en un rango de frecuencias. La siguiente ecuación integra las ganancias A del filtro, y las frecuencias de corte w :

$$h(n) = \begin{cases} \sum_{l=1}^L \left[(A_l - A_{l+1}) \frac{\text{sen}(w_l n)}{\pi n} \right], & n \neq 0 \\ \sum_{l=1}^L \left[(A_l - A_{l+1}) \frac{w_l}{\pi} \right], & n = 0 \end{cases}$$

Ecuación 14-11

$$-\frac{M}{2} < n < \frac{M}{2}$$

La siguiente gráfica muestra la respuesta, de un filtro multi banda de orden 65, con cuatro bandas diferentes:

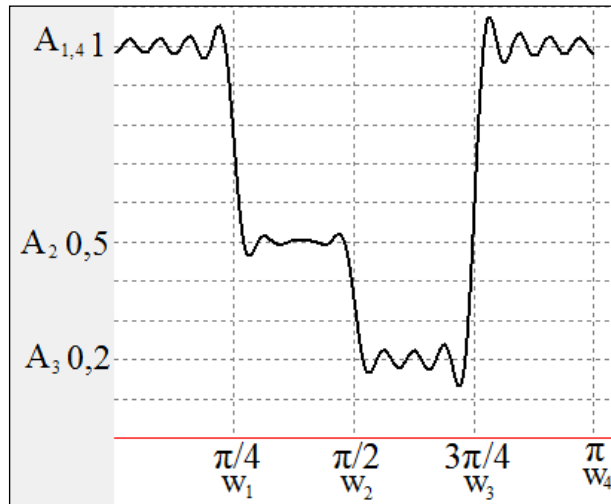


Figura 14-6

Para fines de diseño se debe tener presente que la ultima frecuencia calculada debe ser igual a π . Este tipo de filtros son ideales para el diseño de sistemas que requieran efectuar ecualización, sobre una señal.

14.4.6 Ventanas fijas

Las ventanas se aplican a las funciones de transferencia $h(n)$, el objetivo de las ventanas es mejorar y suavizar la respuesta espectral de los filtros FIR. Las ventanas de mayor uso son las siguientes:

- Rectangular
- Hamming
- Hanning
- Blackman

Los filtros que se demostraron anteriormente por defecto son filtros con ventana rectangular. Estos filtros tienen la menor transición en la frecuencia de corte, lo que los hace más cercanos a los filtros ideales, sin embargo esta propiedad produce en los filtros sobresaltos y oscilaciones en el espectro. Este efecto es conocido como: fenómeno Gibbs. Este efecto puede apreciarse en la siguiente gráfica:

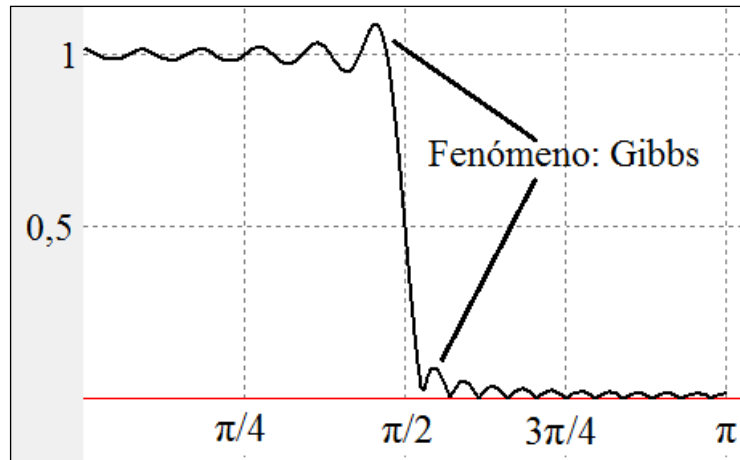


Figura 14-7

La ventana Rectangular ofrece una banda de transición igual a la siguiente relación: $dW = 1,8\pi/M$

14.4.7 Ventanas Rectangulares

Una ventana rectangular ofrece un patrón lineal, y constante. Se debe recordar que al realizar el cálculo de un filtro FIR, por defecto ya se encuentra con una ventana de este tipo.

La aplicación de las ventanas involucra la creación de una nueva función de factores $w(n)$, que posteriormente debe ser multiplicada término a término con la función de transferencia $h(n)$, para crear la función $h(n)$ definitiva.

Una ventana Rectangular se representa por medio de la siguiente función $w(n)$:

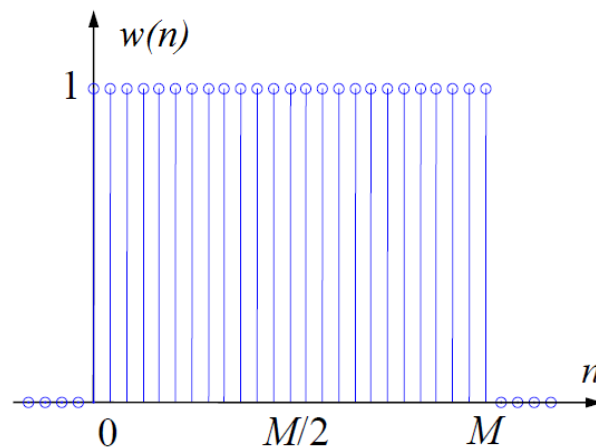


Figura 14-8

La respuesta espectral de la ventana Rectangular en escala logarítmica es la siguiente:

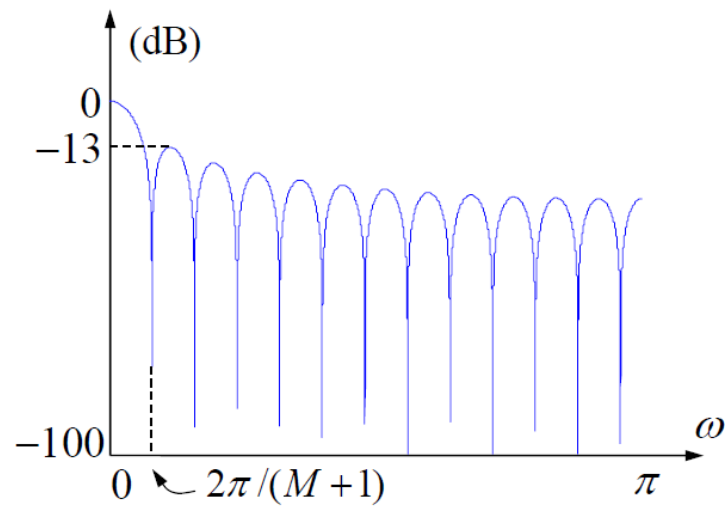


Figura 14-9

La ventana Rectangular ofrece una atenuación del fenómeno Gibbs de -13dB.

14.4.8 Ventanas Hamming

Para el caso de la ventana Hamming, los factores $w(n)$, se calculan de la siguiente forma:

$$w(n) = \frac{1 - \cos(2\pi n / M)}{2}, 0 \leq n \leq M \quad \text{Ecuación 14-12}$$

La misma respuesta espectral aplicando una ventana Hamming, en la figura (14.7) del fenómeno Gibbs se ve de la siguiente forma:

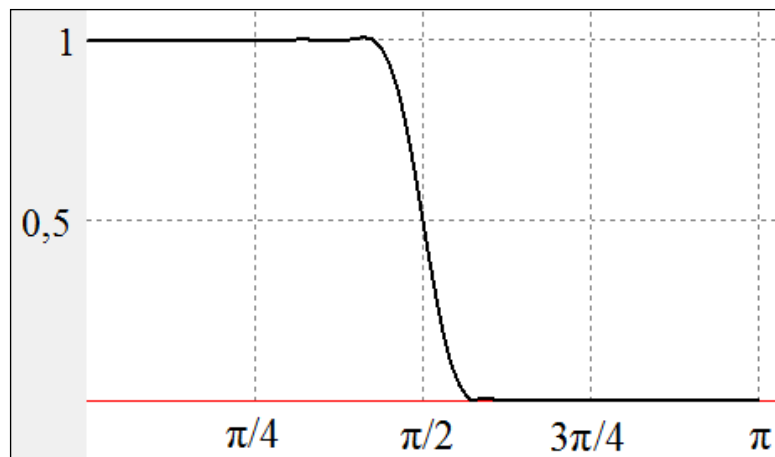


Figura 14-10

La ventana Hamming ofrece una banda de transición igual a la siguiente relación: $dW = 6,2\pi/M$

Una ventana Hamming se representa por medio de la siguiente función $w(n)$:

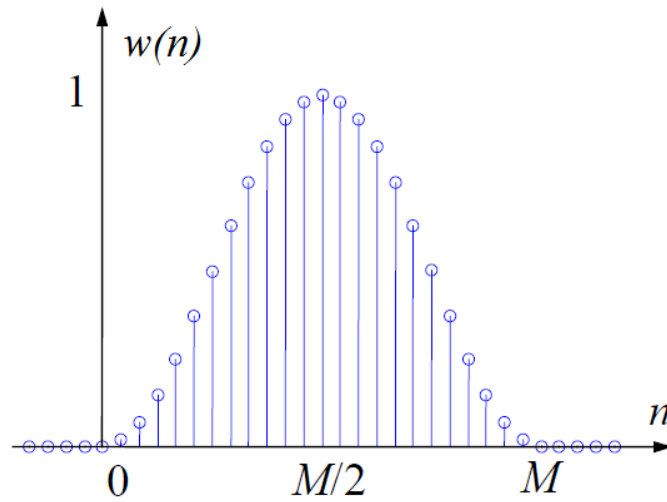


Figura 14-11

La respuesta espectral de la ventana Hamming en escala logarítmica es la siguiente:

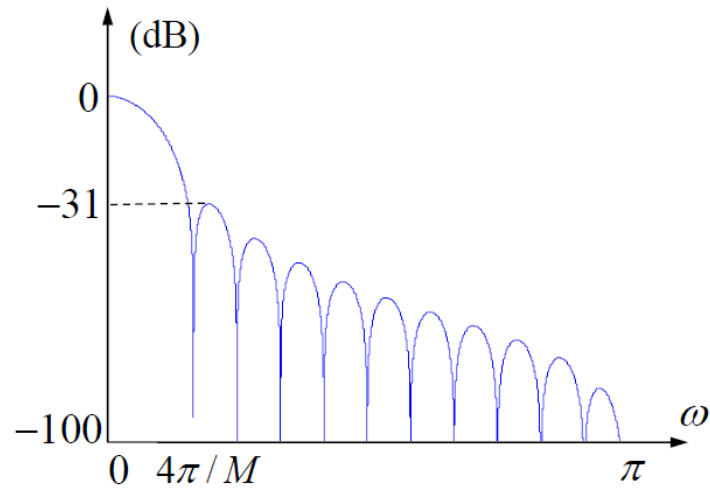


Figura 14-12

En la ventana Hamming se ofrece una atenuación del fenómeno Gibbs de -31dB.

14.4.9 Ventanas Hanning

Para el cálculo de la ventana Hanning, los factores $w(n)$, se deducen de la siguiente forma:

$$w(n) = \frac{27 - 23\cos(2\pi n/M)}{50}, 0 \leq n \leq M \quad \text{Ecuación 14-13}$$

La misma respuesta espectral aplicando una ventana Hanning, en la figura (14.7) del fenómeno Gibbs se ve de la siguiente forma:

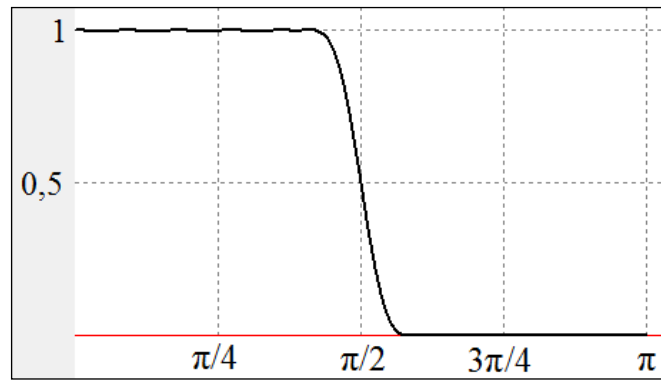


Figura 14-13

La ventana Hanning ofrece una banda de transición igual a la siguiente relación: $dW = 6,6\pi/M$

Una ventana Hanning se representa por medio de la siguiente función $w(n)$:

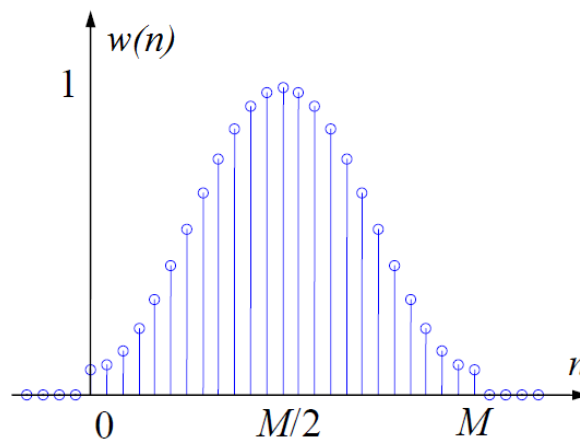


Figura 14-14

La respuesta espectral de la ventana Hanning en escala logarítmica es la siguiente:

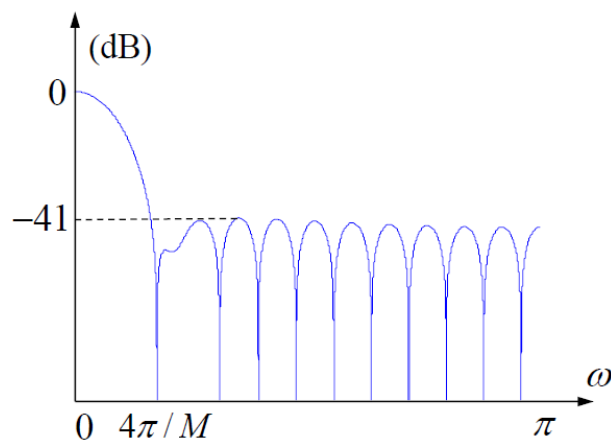


Figura 14-15

En la ventana Hanning se ofrece una atenuación del fenómeno Gibbs de -41dB.

14.4.10 Ventanas Blackman

Para el diseño de una ventana Blackman, los factores $w(n)$, se calculan de la siguiente forma:

$$w(n) = \frac{21 - 25\cos(2\pi n/M) + 4\cos(4\pi n/M)}{50}, 0 \leq n \leq M \quad \text{Ecuación 14-14}$$

La misma respuesta espectral aplicando una ventana Blackman, en la figura (14.7) del fenómeno Gibbs se ve de la siguiente forma:

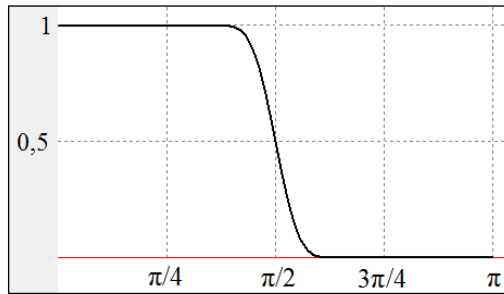


Figura 14-16

La ventana Blackman ofrece una banda de transición igual a la siguiente relación: $dW = 11\pi/M$

Una ventana Blackman se representa por medio de la siguiente función $w(n)$:

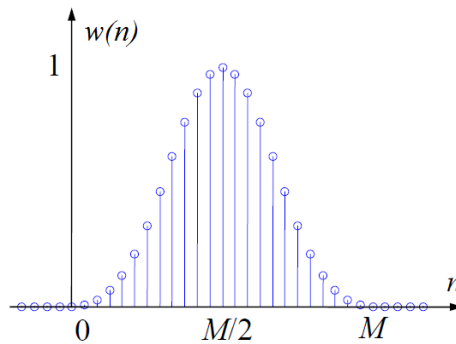


Figura 14-17

La respuesta espectral de la ventana Blackman en escala logarítmica es la siguiente:

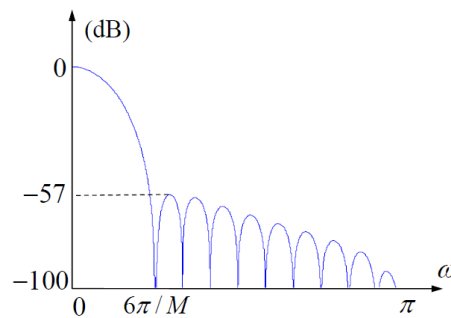



Figura 14-18

La ventana Blackman ofrece una atenuación del fenómeno Gibbs de -57dB.

14.5 Ejemplos de diseño para filtros FIR

En la siguiente sección se mostrarán ejemplos de diseño para filtros FIR, implementando un microcontrolador 18F452, con una fuente de 40MHz como reloj. Para fines prácticos reales la frecuencia de 40MHz, se logra utilizando un cristal de 10MHz, y activando la fuente de reloj HS-PLL en el PIC. Esta opción implementa internamente en el PIC, un PLL que multiplica la frecuencia externa por un factor de cuatro, y el resultado es usado como fuente de reloj para el procesador del microcontrolador. Para todos los ejemplos FIR que se mostrarán se deben simular con el mismo arreglo en ISIS, para este fin se implementa un circuito con los dispositivos 18F452, RES, OSCILLOSCOPE, Generador virtual Seno.

Para usar los generadores de señal virtual en ISIS, se debe picar en la barra de herramientas de la izquierda el icono de Generator Mode, este tiene la siguiente apariencia visual: , dentro de este se escoge la opción SINE, y se pega dentro del área de trabajo. Este generador tiene por defecto una frecuencia de 1Hz, y una amplitud de 1 voltio. La adquisición de señales se hace en el microcontrolador por medio del módulo AD, los niveles de tensión que las entradas análogas admiten, no pueden salirse de los confines de la polarización del microcontrolador. En otras palabras los niveles de las entradas análogas no pueden ser superiores a 5 voltios, ni voltajes negativos. Para evitar las circunstancias antes nombradas, se debe manipular la configuración del generador virtual de señal, para este propósito, se hace doble clic, sobre el generador y se editan los parámetros: Offset, y Amplitud. El parámetro offset se configura a 2,5 voltios, y el parámetro Amplitud a 2 voltios. Esta acción evita que las señales generadas se salgan de los parámetros de PIC. Para fines de simulación el parámetro Frequency, puede ser alterado al gusto del desarrollador.

Una vista de esta edición se puede apreciar en la siguiente figura:

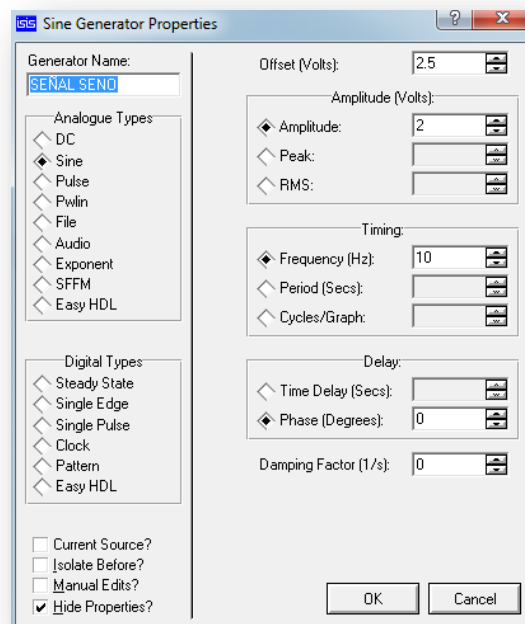
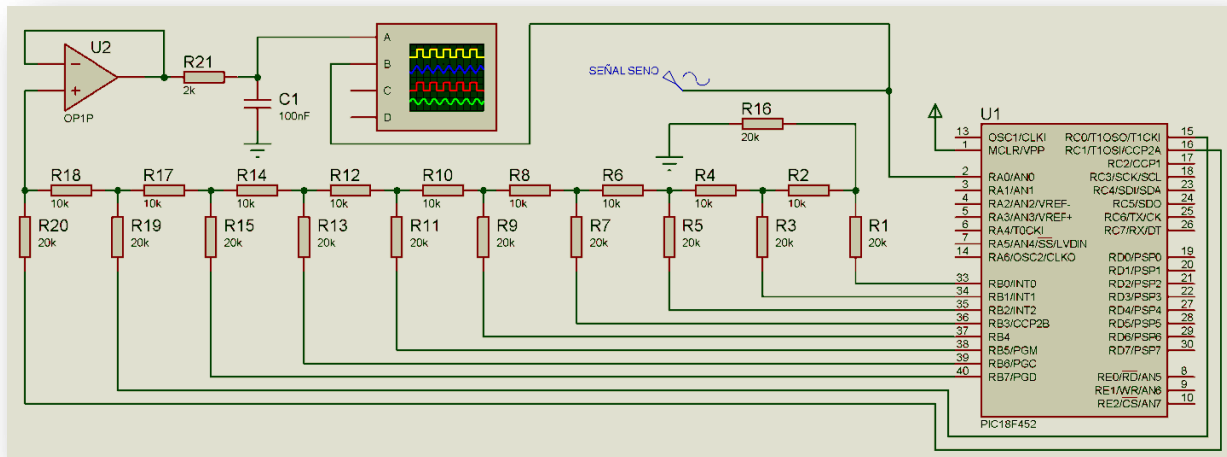


Figura 14-19

Indiferente al ejemplo que se simule en esta sección, se puede implementar el siguiente circuito electrónico en ISIS:



Circuito 14-1

Para la reconstrucción de la señal procesada, se configuran 10 bits de salida, para hacer un convertidor DA, por medio de un arreglo R-2R.

Para todos los ejemplos usados en este apartado, se usará la interrupción por TIMER 0, para crear el periodo de muestreo, y por defecto la frecuencia de muestreo.

El siguiente código fuente muestra un ejemplo del muestreo de la señal por medio del TIMER 0:

```
//Declaración de variables.
float x0, y0;
unsigned int YY;
//Declaración de la función de interrupciones.
void interrupt ( void )
{
    if( INTCON.F2 )
    {
        TMR0L=135;
        //Timer0 con periodo de 774,4u segundo.
        //Fs = 1291,32 Hz.
        //Adquisición de una muestra de 10 bits en, x[0].
        x0 = (float)(ADC_Read(0)-512.0);
        //
        //Espacio para procesar la señal.
        //
        //Reconstrucción de la señal: y en 10 bits.
        YY = (unsigned int)(x0+512);
        PORTC = (YY>8)&3;
        PORTB = YY&255;
    }
}
```

```

    INTCON.F2=0;
}
}

void main( void )
{
    //Inicio del puerto B como salida.
    TRISB = 0;
    PORTB = 0;
    TRISC = 0;
    PORTC = 0;
    //Se configura el TIMER 0, su interrupción.
    INTCON = 0b10100000;
    T0CON = 0b11000101;
    while(1)//Bucle infinito.
    {
    }
}

```

14.5.1 Ejemplo de diseño para filtro pasa bajas

$F_s = 1291,32\text{KHz}$.

$F_c = 150\text{Hz}$.

Ventana Rectangular.

Se determina la frecuencia de corte digital:

$$W_c = 2 \pi F_c / F_s = 2 \pi 150 / 1291,32 = 0,72985.$$

Usando la formula (14.7), se designan los coeficientes de la función $h(n)$:

```

h(-8)=-0.0171035387965417
h(-7)=-0.0419431579233366
h(-6)=-0.0501329294124475
h(-5)=-0.0309497847516785
h(-4)=0.0175345019583181
h(-3)=0.0864308262744764
h(-2)=0.158173992108178
h(-1)=0.212237065988464
h(0)=0.232320416318186
h(1)=0.212237065988464
h(2)=0.158173992108178
h(3)=0.0864308262744764
h(4)=0.0175345019583181
h(5)=-0.0309497847516785
h(6)=-0.0501329294124475
h(7)=-0.0419431579233366
h(8)=-0.0171035387965417

```

Se debe tener presente que para fines de programación los valores de n no pueden ser negativos, por esta razón se debe iniciar el vector en 0, y para usarlo en el código en lenguaje C, se implementa de la siguiente manera:

```
const float h[]=
{
-0.0171035387965417, //h(0)
-0.0419431579233366, //h(1)
-0.0501329294124475, //h(2)
-0.0309497847516785, //h(3)
0.0175345019583181, //h(4)
0.0864308262744764, //h(5)
0.158173992108178, //h(6)
0.212237065988464, //h(7)
0.232320416318186, //h(8)
0.212237065988464, //h(9)
0.158173992108178, //h(10)
0.0864308262744764, //h(11)
0.0175345019583181, //h(12)
-0.0309497847516785, //h(13)
-0.0501329294124475, //h(14)
-0.0419431579233366, //h(15)
-0.0171035387965417 //h(16)
};
```

El programa definitivo con la función de transferencia $h(n)$, será de la siguiente forma:

```
#define M 17
//Función de transferencia h[n]
const float h[]=
{
-0.0171035387965417, //h(0)
-0.0419431579233366, //h(1)
-0.0501329294124475, //h(2)
-0.0309497847516785, //h(3)
0.0175345019583181, //h(4)
0.0864308262744764, //h(5)
0.158173992108178, //h(6)
0.212237065988464, //h(7)
0.232320416318186, //h(8)
0.212237065988464, //h(9)
0.158173992108178, //h(10)
0.0864308262744764, //h(11)
0.0175345019583181, //h(12)
-0.0309497847516785, //h(13)
-0.0501329294124475, //h(14)
-0.0419431579233366, //h(15)
-0.0171035387965417 //h(16)
};
```



```

//Declaración de variables.
float x0, y0;
float x[M];
unsigned int YY;
unsigned short i;
//Declaración de la función de interrupciones.
void interrupt ( void )
{
    if( INTCON.F2 )
    {
        TMR0L=135;
        PORTC.F7=1;
        //Timer0 con periodo de 774,4u segundo.
        // Fs = 1291,32 Hz.
        //Corrimiento continuo de la señal x[n]
        for( i=M-1; i!=0; i-- )x[i]=x[i-1];
        //Adquisición de una muestra de 10 bits en, x[0].
        x[0] = (float)(ADC_Read(0)-512.0);
        //Convolución continua.
        y0 = 0.0; for( i=0; i<M; i++ ) y0 += h[i]*x[i];
        //Reconstrucción de la señal: y en 10 bits.
        YY = (unsigned int)(y0+512.0);
        PORTC = (YY>>8)&3;
        PORTB = YY&255;
        PORTC.F7=0;
        INTCON.F2=0;
    }
}

void main( void )
{
    //Inicio del puerto B como salida.
    TRISB = 0;
    PORTB = 0;
    TRISC = 0;
    PORTC = 0;
    //Se configura el TIMER 0, su interrupción.
    INTCON = 0b10100000;
    T0CON = 0b11000101;
    while(1)//Bucle infinito.
    {
    }
}

```

Por limitaciones de velocidad en este microcontrolador, y teniendo presente la frecuencia de muestreo de 1291,32Hz no es posible usar un orden del filtro superior a 17. Sin embargo es posible usar órdenes menores como; 15, 13, 11, 9, 7, 5, o 3. Para fines prácticos y didácticos en los próximos ejemplos se implementarán filtros de orden 17 igual a este.

14.5.2 Ejemplo de diseño para filtro pasa altas

$F_s = 1291,32\text{KHz}$.

$F_c = 200\text{Hz}$.

Ventana Hamming.

Se determina la frecuencia de corte digital:

$$W_c = 2 \pi F_c / F_s = 2 \pi 200 / 1291,32 = 0,97314.$$

Usando la formula (14.8), se definen los coeficientes de la función $h(n)$ multiplicados por la ventana Hamming $w(n)$ de la ecuación (14.12), dando como resultado la siguiente función $h(n)$:

$h(-8)=0$
 $h(-7)=-0.000774581987451374$
 $h(-6)=0.00297591407324525$
 $h(-5)=0.0174357425540551$
 $h(-4)=0.0246447931670207$
 $h(-3)=-0.0148886974624375$
 $h(-2)=-0.118648252092459$
 $h(-1)=-0.243426834227261$
 $h(0)=0.684363125797732$
 $h(1)=-0.260893089082499$
 $h(2)=-0.136977589378571$
 $h(3)=-0.0187342667800403$
 $h(4)=0.0345797805794857$
 $h(5)=0.028555061237806$
 $h(6)=0.00631989035694063$
 $h(7)=-0.00299371636029323$
 $h(8)=-0.00134023946307802$

La implementación del código fuente es igual al filtro pasa bajas, sustituyendo el arreglo $h[]$, por el siguiente:

```
const float h[]=  
{  
    0, //h(0)  
    -0.000774581987451374, //h(1)  
    0.00297591407324525, //h(2)  
    0.0174357425540551, //h(3)  
    0.0246447931670207, //h(4)  
    -0.0148886974624375, //h(5)  
    -0.118648252092459, //h(6)  
    -0.243426834227261, //h(7)  
    0.684363125797732, //h(8)  
    -0.260893089082499, //h(9)  
    -0.136977589378571, //h(10)  
    -0.0187342667800403, //h(11)  
    0.0345797805794857, //h(12)  
    0.028555061237806, //h(13)  
}
```

```

0.00631989035694063, //h(14)
-0.00299371636029323, //h(15)
-0.00134023946307802 //h(16)
};

```

14.5.3 Ejemplo de diseño para filtro pasa banda

$F_s = 1291,32\text{KHz}$.
 $F_{c1} = 150\text{Hz}$.
 $F_{c2} = 400\text{Hz}$.
 Ventana Hanning.

Se determinan las frecuencias de corte digital:

$$W_{c1} = 2 \pi F_c / F_s = 2 \pi 150 / 1291,32 = 0,72985.$$

$$W_{c2} = 2 \pi F_c / F_s = 2 \pi 400 / 1291,32 = 1,94628.$$

Usando la formula (14.9), se definen los coeficientes de la función $h(n)$ multiplicados por la ventana Hanning $w(n)$ de la ecuación (14.13), dando como resultado la siguiente función $h(n)$:

```

h(-8)=0.0018052104538582
h(-7)=0.00905810215732946
h(-6)=0.00179096961917994
h(-5)=0.00393014193876244
h(-4)=0.0307760790492056
h(-3)=-0.0879236273273945
h(-2)=-0.218010454414228
h(-1)=0.078115497545518
h(0)=0.384167993159943
h(1)=0.0832388331308223
h(2)=-0.248392736747743
h(3)=-0.107904878578323
h(4)=0.0411879382357919
h(5)=0.00583790841302624
h(6)=0.00299868100666665
h(7)=0.0163162389068139
h(8)=0.00250614601893693

```

La función $h[]$, para implementar en el código fuente en lenguaje C es:

```

const float h[]=
{
0.0018052104538582, //h(0)
0.00905810215732946, //h(1)
0.00179096961917994, //h(2)
0.00393014193876244, //h(3)
0.0307760790492056, //h(4)
-0.0879236273273945, //h(5)
-0.218010454414228, //h(6)
0.078115497545518, //h(7)

```

```

0.384167993159943, //h(8)
0.0832388331308223, //h(9)
-0.248392736747743, //h(10)
-0.107904878578323, //h(11)
0.0411879382357919, //h(12)
0.00583790841302624, //h(13)
0.00299868100666665, //h(14)
0.0163162389068139, //h(15)
0.00250614601893693 //h(16)
};

```

14.5.4 Ejemplo de diseño para filtro rechaza banda

$F_s = 1291,32\text{KHz}$.

$F_{c1} = 150\text{Hz}$.

$F_{c2} = 400\text{Hz}$.

Ventana Blackman.

Se determinan las frecuencias de corte digital:

$W_{c1} = 2 \pi F_c / F_s = 2 \pi 150 / 1291,32 = 0,72985$.

$W_{c2} = 2 \pi F_c / F_s = 2 \pi 400 / 1291,32 = 1,94628$.

Usando la formula (14.10), se definen los coeficientes de la función $h(n)$ multiplicados por la ventana Blackman $w(n)$ de la ecuación (14.14), dando como resultado la siguiente función $h(n)$:

```

h(-8)=3.13154095338657E-19
h(-7)=-0.00105084724932717
h(-6)=-0.000518135020348656
h(-5)=-0.00174730211401576
h(-4)=-0.0182611591144528
h(-3)=0.0645431455464317
h(-2)=0.186587834540544
h(-1)=-0.0738928265716166
h(0)=0.604271792109402
h(1)=-0.0827284691705043
h(2)=0.234965429330525
h(3)=0.0923521657912548
h(4)=-0.0302353209611255
h(5)=-0.00346395569934133
h(6)=-0.00133318382487159
h(7)=-0.00472035632958784
h(8)=-0.000290742652784183

```

La función $h[]$, para implementar en el código fuente en lenguaje C es:

```

const float h[]=
{
    3.13154095338657E-19, //h(0)
    -0.00105084724932717, //h(1)

```

```

-0.000518135020348656, //h(2)
-0.00174730211401576, //h(3)
-0.0182611591144528, //h(4)
0.0645431455464317, //h(5)
0.186587834540544, //h(6)
-0.0738928265716166, //h(7)
0.604271792109402, //h(8)
-0.0827284691705043, //h(9)
0.234965429330525, //h(10)
0.0923521657912548, //h(11)
-0.0302353209611255, //h(12)
-0.00346395569934133, //h(13)
-0.00133318382487159, //h(14)
-0.00472035632958784, //h(15)
-0.000290742652784183 //h(16)
};

```

14.5.5 Ejemplo de diseño para filtro multi banda

$F_s = 1291,32\text{KHz}$.

$F_{c1} = 161\text{Hz}$.

$F_{c2} = 322\text{Hz}$.

$F_{c3} = 484\text{Hz}$.

$F_{c4} = 645.66\text{Hz}$.

$A_1 = 1$.

$A_2 = 0,5$.

$A_3 = 0,2$.

$A_4 = 1$.

Ventana Rectangular.

Se determinan las frecuencias de corte digital:

$W_{c1} = 2 \pi F_{c1} / F_s = 2 \pi 161 / 1291,32 = 0,78337$.

$W_{c2} = 2 \pi F_{c2} / F_s = 2 \pi 322 / 1291,32 = 1,56675$.

$W_{c3} = 2 \pi F_{c3} / F_s = 2 \pi 484 / 1291,32 = 2,355$.

$W_{c4} = 2 \pi F_{c4} / F_s = 2 \pi 645,66 / 1291,32 = \pi$.

Usando la formula (14.11), se definen los coeficientes de la función $h(n)$:

$h(-8)=-0.000403386658426763$

$h(-7)=-0.00443133542115946$

$h(-6)=-0.0685784954060509$

$h(-5)=0.0330418473673128$

$h(-4)=-0.000367826230314909$

$h(-3)=-0.0538949660023408$

$h(-2)=0.207286062892996$

$h(-1)=0.0275264614524777$

$h(0)=0.674596536876994$

$h(1)=0.0275264614524777$

$h(2)=0.207286062892996$

h(3)=-0.0538949660023408
h(4)=-0.000367826230314909
h(5)=0.0330418473673128
h(6)=-0.0685784954060509
h(7)=-0.00443133542115946
h(8)=-0.000403386658426763

La función h[], para implementar en el código fuente en lenguaje C es la siguiente:

```
const float h[]=  
{  
-0.000403386658426763, //h(0)  
-0.00443133542115946, //h(1)  
-0.0685784954060509, //h(2)  
0.0330418473673128, //h(3)  
-0.000367826230314909, //h(4)  
-0.0538949660023408, //h(5)  
0.207286062892996, //h(6)  
0.0275264614524777, //h(7)  
0.674596536876994, //h(8)  
0.0275264614524777, //h(9)  
0.207286062892996, //h(10)  
-0.0538949660023408, //h(11)  
-0.000367826230314909, //h(12)  
0.0330418473673128, //h(13)  
-0.0685784954060509, //h(14)  
-0.00443133542115946, //h(15)  
-0.000403386658426763, //h(16)  
};
```