9. ディジタルフィルタ

1. 目的

本実験の目的は, ディジタルフィルタを設計し, ディジタルシグナルプロセッサを用いて実現することで, ディジタル信号処理に関する理解を深めることである。

2. 原理

ディジタルフィルタとは, 観測信号から目的とする信号成分を取り出す機能を持つディジタル信号処理システムである。本実験では, 連続時間信号を処理するディジタルフィルタを設計する問題を取り扱う。

2.1 連続時間信号のディジタル信号処理

図 9.1 に連続時間信号のディジタル信号の流れを示す。図 9.1 において, A/D 変換器とは連続時間信号 $x(t)$

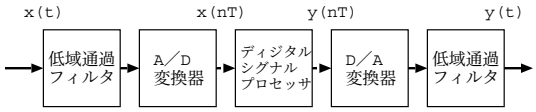

図 9.1 連続時間信号のディジタル信号処理

を一定時間 T ごとにサンプリングして離散化しとびとびの値を取る数列 $\{x(0), x(T), \ldots\}$ に変換するという処理をおこなう装置であり, D/A 変換器とはディジタルシグナルプロセッサによる処理の結果得られた数列 $\{y(0), y(T), \ldots\}$ を連続時間信号 $y(t)$ に変換する処理をおこなう装置である。A/D 変換の処理の流れを図 9.2 に, D/A 変換器の処理の流れを図 9.3 に示す。

以下では, 観測信号は $T = 1/f_s$ 秒ごとにサンプリングされているものとする。 このとき, f_s をサンプリング周波数と呼ぶ。また, サンプリング周波数の半分の値 $(f_s/2)$ をナイキスト周波数と呼ぶ。また, $x(kT)$, $y(kT)$ などをより簡単に x_k, y_k などと略記する。

2.2 FIR フィルタ

FIR フィルタ (Finite Impulse Response Filter) とは, インパルスを入力したときの出力信号が有限時間で 0 に収束するフィルタである。FIR フィルタは, 多くの場合, 式 (9.1) のような非再帰型の差分方程式で実現される。

$$y_k = h_0 x_k + h_1 x_{k-1} + \cdots + h_n x_{k-n} \qquad (9.1)$$

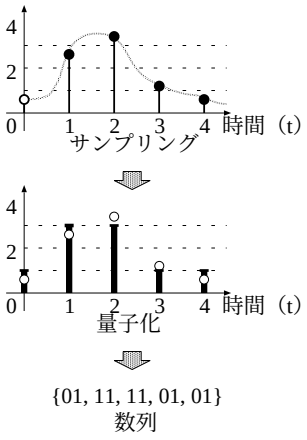

図 9.2 A/D 変換器の処理の流れ

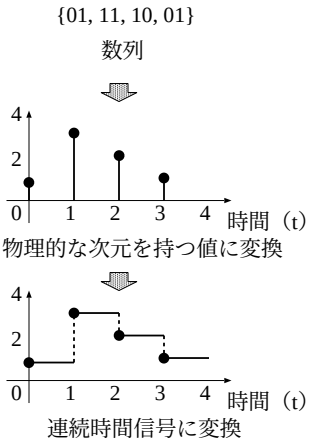

図 9.3 D/A 変換器の処理の流れ

式 (9.1) の意味は, 「時刻 kT における出力信号 y_k は, 現在および過去の入力信号値 $x_k, x_{k-1}, \ldots, x_{k-n}$ の重み付き平均であらわされる」ということである。重み付き平均を取るときの重み h_i が FIR フィルタの係数である。この係数のことをタップ係数と呼ぶこともある。特に断らない限り, h_i は実数であるものとする。また, 式 (9.1) において, n をフィルタの次数, $n+1$ をフィルタ長と呼ぶ。

2.3 FIR フィルタの周波数応答

フィルタの入力 $x(t)$ が周波数 f[Hz] の正弦波である場合を考える。

$$x_k = e^{j2\pi fkT} \qquad (9.2)$$

とする。 式 (9.2) を式 (9.1) に代入すると,

$$y_k = \left(h_0 + h_1 e^{-j2\pi fT} + \cdots + h_n e^{-j2\pi fnT}\right) e^{j2\pi fkT} \qquad (9.3)$$

が得られる。

$$G(f) = h_0 + h_1 e^{-j2\pi fT} + \cdots + h_n e^{-j2\pi fnT} \qquad (9.4)$$

と定義する。 式 (9.3) から, フィルタの出力信号 y_k は, 入力信号 x_k に複素数 $G(f)$ を乗じた値となることがわかる。 よって, このフィルタの周波数応答は式 (9.3) によって与えられる。

以下では, フィルタの周波数応答の絶対値をゲインと呼ぶ。

FIR フィルタのタップ係数 h_i を適当に設定することで, 低域通過フィルタ, 高域通過フィルタ, 帯域通過フィルタあるいは帯域阻止フィルタなどといったいろいろなフィルタを設計することができる。ただし, 有限次元の因果的なフィルタでは, 通過域の信号を完全に歪みなく透過させ阻止域の信号は完全に遮断する理想的な特性を持つフィルタは実現できないことが知られている。また, フィルタの性能を理想的なものに近付ければ近付けるほど, 高性能なハードウェアが必要になり, 必然的にコストも増大する。このため, 実際にフィルタを設計する際には, 性能とコストとのトレードオフなどのさまざまな要素を考慮しなければならない。

3. 実験

3.1 実験課題

教官が音楽に正弦波の雑音が加算された信号の記録されたミュージックテープを渡す。音楽の波形をスペクトルアナライザを用いて観測せよ。続いて, 正弦波の雑音を消去するフィルタ (帯域阻止フィルタ) を設計せよ。さらに, 設計されたフィルタをディジタルシグナルプロセッサで動かし, 特性を測定せよ。

3.2 実験手順

この実験では, Scilab を用いて FIR フィルタを設計し, テキサス・インスツルメンツ社の TMS320C6711 という DSP ボードおよびそれに付属する開発環境である Code Composer Studio を利用して DSP 上で動作するプログラムを開発し実行してから, その特性を測定する。DSP ボードの構成を図 9.4 に示す。なお, この DSP ボードのサンプリング周波数は 8kHz である。

以下では, マウス左ボタンを 1 回押して離すことを「クリックする」とよび, マウス左ボタンを 2 回連続して押して離すことを「ダブルクリックする」とよぶ。また, マウス右ボタンを 1 回押して離すことを「右クリックする」とよぶ。さらに, アイコンにマウスカーソルを合わせてクリックすることを「アイコンをクリックする」という。また, 「メニューを選択する」とは, メニューが表示されたウインドウ内で色が青くなっている部分を選択したいものの名前に合わせてクリックすることをいう。また, Windows98 ではウインドウ右上の×印をクリックするとそのウインドウのプログラムを終了させることができることを覚えておくこと。

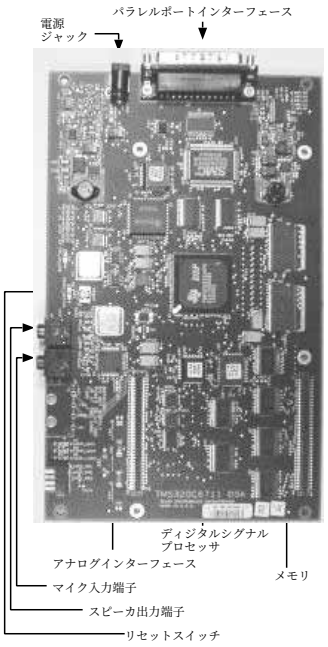

図 9.4 DSP ボード TMS320C6711 の構成

3.2.1 準備

コンピューターの電源を投入する前にパラレルポートと DSP ボード を接続し, 続いて DSP ボード と電源ユニットを接続して, DSP ボード 上の LED が点灯することを確認すること。

パソコンが起動した後に, フロッピーディスクドライブにフロッピーディスクを挿入せよ。実験に必要なファイルはすべてあらかじめフロッピーディスク上に用意されている。

3.2.2 雑音周波数の測定

この実験では, 消去すべき雑音の周波数がグループによって異なる。そこで, フィルタ設計に先立って, ミュージックテープから再生される音楽を観測し, 雑音の周波数を確定しておく必要がある。

このために, パソコン上で動作するソフトウェアスペクトルアナライザである WaveSpectra[1] (フリーウェアである) を用いる[2]。

WaveSpectra を起動するには, デスクトップの左端にある WS というアイコンをダブルクリックする。すると, 図 9.5 のようなウィンドウがあらわれる。

[1] http://www.ne.jp/asahi/fa/efu/index.html

[2] パソコンに搭載されたサウンドカードを用いて測定をおこなう場合, 何らかの方法でマイク入力端子のレベル較正をおこなった場合を除き, 測定された信号の値は相対的にしか意味を持たない。また, 一般にサウンドカードのノイズ対策は極めて不十分なので, 測定値の信頼性も低い。

続いて, パソコンのマイク入力端子とカセットテープレコーダのオーディオ出力端子を接続する。

カセットテープレコーダの再生ボタンを押してからWaveSpectra の録音ボタン (図 9.5 の説明を参照) を押すと, 図 9.7 のように, マイク入力端子から取得された信号の時間波形とスペクトルが表示される。

図 9.7 において観測されている信号には周波数 2kHz の雑音が印加されているのであるが, 表示されたスペクトルが 2kHz の部分にピークを持つことが図から読み取れる。

なお, スペクトルのピークが複数ある場合にも, 雑音が周波数と振幅が一定の正弦波のときには正弦波雑音に対応するスペクトルが時間か経過しても変化しないのに対し, それ以外の信号に大層するスペクトルは時間とともに変動するので, スペクトルの時間変化を観察すれば, 正弦波雑音とそうでない部分を区別することができる。

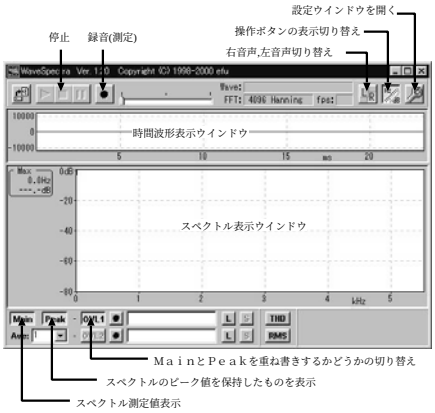

図 9.5 WaveSpectra の起動

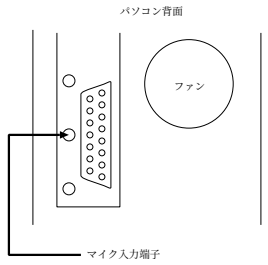

図 9.6 マイク入力端子

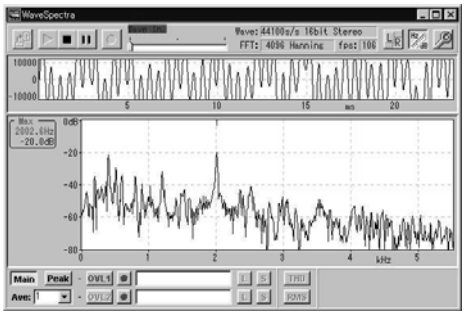

図 9.7 スペクトルが表示されたところ

3.2.3 FIR フィルタの設計

フィルタの設計には Scilab を用いる。
Scilab を起動するには,

スタート→プログラム (P)
→ Scilab 2.6 → Scilab2.6

を選択する (図 9.8)。

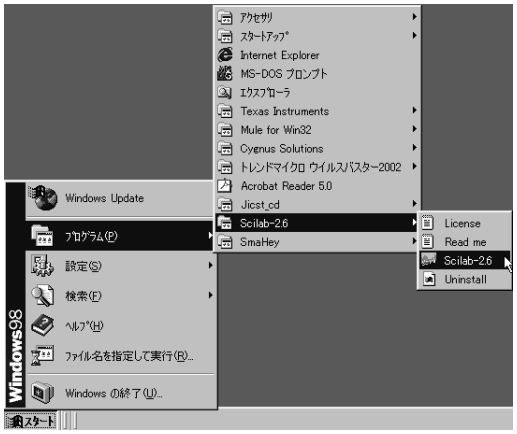

図 9.8 Scilab の起動

フィルタの設計にあたって, 諸君のフロッピーディスクのフォルダ fir にあるサンプルスクリプト design.sci を利用する。実験ではこのファイルを修正する必要があるのだが, 実際に修正作業をおこなう前に, まずこのスクリプトを実行してみることにする。

このためには, Scilab のウィンドウ内で

```
exec('a:\fir\design.sci');
```

と入力する (図 9.9)。すると, フィルタの設計が実行さ

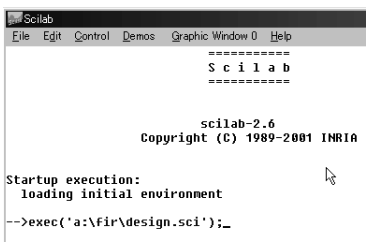

図 9.9 design.sci の実行

れ。画面に設計されたフィルタの周波数応答が表示された後に, 設計されたフィルタのタップ係数がフロッピーディスクのフォルダ fir に tap.txt という名前で書き出される。

なお, フロッピーディスク上にすでに同名のファイルがある場合には, そのファイルは強制的に削除される。古い tap.txt を保存しておく必要があるときにはあらかじめ名前を変更しておくこと。

ファイル design.sci の簡単な説明を図 9.10 に示す。

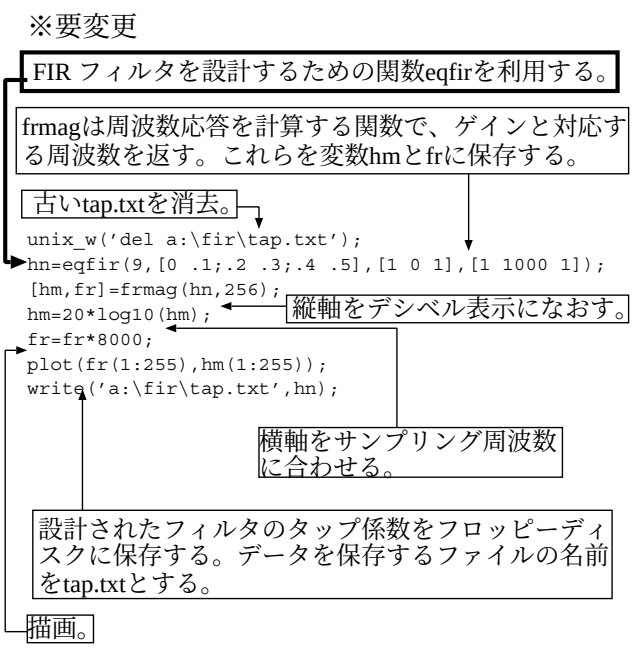

図 9.10 FIR フィルタ設計プログラム (design.sci) の説明

続いて, design.sci を修正して, 各自に配布されたテープに含まれる雑音が消えるようにする。design.sci を開くには,

```
A:\fir\design.sci
```

をダブルクリックすればよい (図 9.11)。ファイルをど

図 9.11 design.sci をエディタで開く

のように修正すれば良いかについては, 後で述べる。

ファイルの修正が終わったら必ずウインドウ上部のメニューから

ファイル→上書き保存 (S)

を実行して変更後の内容を保存しておくこと。

ファイルの保存に成功したら, 再び design.sci を実行する。画面に表示されたフィルタの性能が満足すべきものだったら次に進む。性能に問題があると思われる場合は前に戻る。

ここで, ファイル design.sci のどこを変更する必要があるかについて説明しておく。

変更を要するのは最初の行のみである。この行では関数 eqfir を使って帯域阻止フィルタを設計しているのだが, この関数の第 1 引数, 第 2 引数, 第 4 引数を設計したいフィルタの特性に対応したものに書き換えなければならない。

関数 eqfir の引数の意味と注意事項について以下で説明する。

第 1 引数 フィルタの次数を指定する部分である。この数値は整数でなければならない。一般に, フィルタの次数が高いほど理論的にはフィルタは高性能になる。しかし, 実際には, 次数を高くしすぎると, 計算の遅延時間が長くなる, メモリを大量に消費する, 桁溢れや桁落ちの影響を受けやすくなる, といった問題が生じる。一方, 次数が低すぎる場合には, フィルタの設計上の性能がかなり低くなってしまう。一般的に言えば, フィルタの性能が同等であるならば, 次数は低ければ低いほどよい。各自でどの程度の次数が適切であるか試してみること。なお, フィルタの次数が偶数と奇数の場合で挙動が大幅に異なることを注意しておく。

第 2 引数 フィルタの形状を指定する部分である。この部分では, Scilab の行列の記法を使って, どの周波数帯でフィルタの性能を指定するかを決める。

周波数帯を指定する際には, 周波数軸をいくつかのたがいに重ならない領域に区切り, 各行の成分がこれらの領域の左端および右端の周波数になっている 2 列の行列を作成する。この実験で作成したいフィルタは帯域阻止フィルタで, 性能を指定したい領域は

- 低周波側の信号を通過させる領域
- 信号を遮断したい領域
- 高周波側の信号を通過させる領域

の 3 個である。よって, 第 2 引数は 3 行 2 列の行列になる。

実験課題を達成するためには, 第 2 引数の行列の内容を諸君が取り扱わなければならない帯域阻止フィルタに対応したものに書き換える必要がある。

なお, 周波数軸は正規化されていて, 設計上の周波数 1 が実際のフィルタにおけるサンプリング周波数 (この実験では 8kHz) に対応する。また, ディジタルフィルタの周波数特性はナイキスト周波数 (サンプリング周波数の半分) を境にして対称になるので, フィルタを設計する際に周波数として 0.5 より大きい値を指定しても意味がない。

また, FIR フィルタは急峻な特性を持つフィルタを設計するには不向きであるので, 指定する領域や特性を指定している領域にはさまれた「遷移領域」の幅が狭すぎると設計がうまくいかないことがある。

第 3 引数 第 2 引数において指定した領域において信号を通過させるか遮断するかを指定する。値 1 を指定すると信号通過, 値 0 を指定すると信号遮断を指定したことになる。この部分は変更しなくてよい。

第 4 引数 この部分は「重み」と呼ばれるもので, 第 2 引数で指定した各領域がどのくらい重要であるかを指定する部分である。この部分に指定される数値は整数である。大きい重みが与えられたは理想的な値に近い特性を持つようになるが, その場合, 小さい重みが与えられた領域の性能が犠牲になる。

図 9.10 の第 1 行目でどのようなフィルタの特性が指定されているかを図 9.12 に示す.

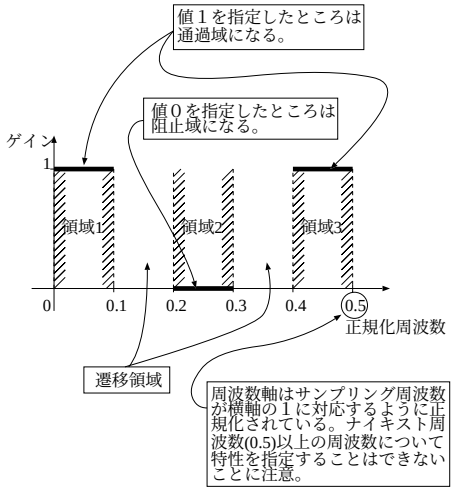

図 9.12 関数 eqfir におけるフィルタの特性の指定

希望する特性を持つフィルタが得られるまで, 何度でも design.sci を変更して Scilab で実行すること。なお, design.sci を実行すると以前作成した tap.txt は消えてしまうので, 必要な場合は各自で名前を変えて保存しておくこと。

なお, ファイルの編集には, Mule, メモ帳など, どのようなエディタを使ってもよい。

なお, Scilab のグラフィックスウィンドウに描画されたグラフを印刷するには, グラフィックスウィンドウ上部の File メニューから Print を選択すること。ここには Print(Scilab) と Print(Windows) の 2 種類の印刷用メニューがある。どちらを使ってもよい。また, グラフをファイルに保存しておくには, File メニューから Export を選択する。

3.2.4 Code Composer Studio を用いた実行可能プログラムの作成

本節では, Code Composer Studio を用いて C6711 で実行可能なプログラムを作成する方法について述べる。

3.2.4.1 Code Composer Studio の起動 Code Composer Studio を起動するには,

スタート→プログラム (P) → Ti → CCStudio

を選択する (図 9.13)。図 9.13 のメニューが実行され

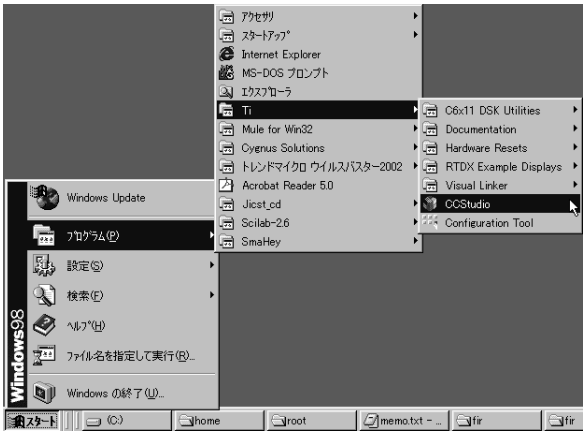

図 9.13 Code Composer Studio の起動

ると, 画面に図 9.14 のようなロゴがあらわれ, 続いて DSP ボードが認識されたことを確認するウィンドウが開く (図 9.15)。OK ボタンを押すと図 9.16 のように Code Composer Studio が起動する。

3.2.4.2 プロジェクトファイルを開く プロジェクトファイルとは, プログラムを完成させるために必要となる一連の手続きを記述したファイルのことである。ここでは, プロジェクト名を fir とする。この実験ではフロッピーディスクにあらかじめ用意されたプロジェクトファイルを利用する。

このプロジェクトを開くには, Code Composer Studio の画面上端のメニューから

図 9.14 Code Composer Studio の起動ロゴ

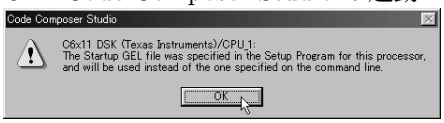

図 9.15 DSP ボードの確認ウィンドウ

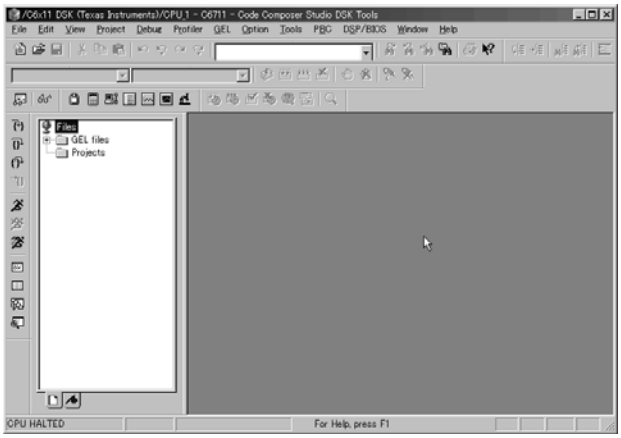

図 9.16 Code Composer Studio が起動したところ

Project → Open...

を選択する (図 9.17)。すると図 9.18 にようなウィンドウがあらわれるので, そのウィンドウに表示された fir.pjt をダブルクリックして選択する。

以上でプロジェクトを開く手順は終了する。

3.2.4.3　ソースプログラムの編集　先に Scilab を用いて設計されたフィルタのタップ係数を実際のプログラムに反映させるためには, ソースプログラムにこの数値を書き込む必要がある。この実験で用いるソースプログラムの名前は fir.c である。このファイルはフロッピーディスクのフォルダ fir に保存されている。

fir.c を編集するためには, まずこれを開く必要がある。このファイルを開くには, Code Composer Studio の画面上端のメニューから

File → Open...

を選択する (図 9.19)。続いて, 画面にあらわれたウィンドウでファイル fir.c をダブルクリックして選択する (図 9.20)。すると, 図 9.21 のように, ファイル fir.c が表示されたウィンドウが開く。このウィンドウを使ってファイルの編集作業をおこなうことができる。

続いて, ソースプログラムのうち変更を要する箇所について説明する。変更を要するのは 5 行目, 10 行目の 2 箇所だけである。

ソースプログラムの 5 行目では, LENGTH という定数を使ってフィルタ長が宣言されている。この数値を諸君が設計したフィルタのフィルタ長に変更する必要がある。

続いて, 配列 `fir_tap` の値 ({} の中) に, 設計された FIR フィルタのタップ係数を書き込む。このためには, まず設計済みのフィルタのタップ係数が保存されたファイル

```
A:\fir\tap.txt
```

をダブルクリックして開く。続いてウインドウ左上の端にマウスカーソルを合わせてからマウス左ボタンを押し下げた状態にする。そして, そのままの状態でマウスカーソルを下に移動させると, ファイルに書き込まれた数値の部分がすべて反転表示されるようになる (図 9.22)。この状態でマウスを右クリックし, あらわれたメニューの上から 3 番目のコピー (C) にマウスカーソルを合わせてマウスをクリックする (図 9.23)。上述の操作が終わったら, CodeComposerStudio 内の fir.c が開いているウインドウをクリックし, 9 行目の配列 `fir_tap` の値 ({} の中の数値) をすべて消してから {} の内部にカーソルを合わせて (マウスカーソルを {} 内に入れてクリック) マウスを右クリックする。すると

図 9.17 プロジェクトを開く (1)

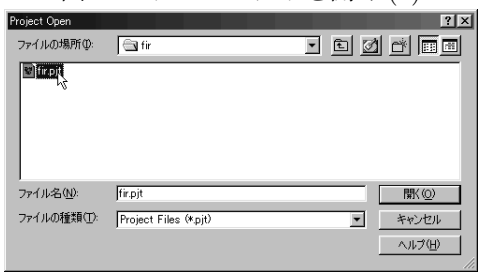

図 9.18 プロジェクトを開く (2)

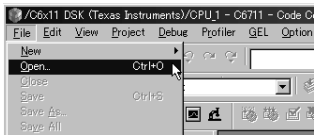

図 9.19 ソースプログラム fir.c を開く (1)

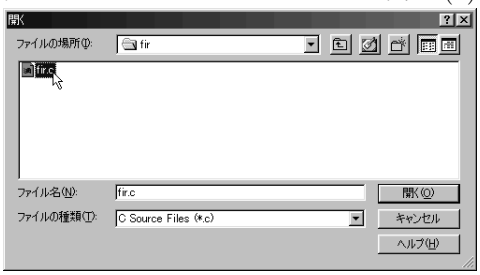

図 9.20 ソースプログラム fir.c を開く (2)

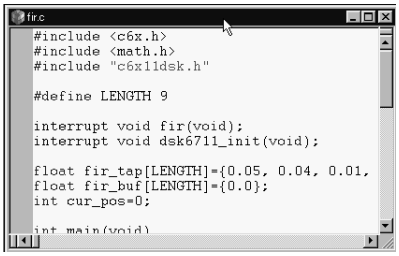

図 9.21 ソースプログラム fir.c を開く (3)

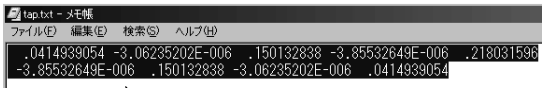

図 9.22 tap.txt の内容をコピー (1)

図 9.23 tap.txt の内容をコピー (2)

図 9.24 のようなメニューがあらわれるので, この中から paste を選択する。すると, フィルタのタップ係数が {} 内に書き込まれる。さいごに, 個々の数値のあいだにコンマ (,) を入れる。ただし, { の直後と } の直前にはコンマを入れないことと。また, 数値と数値のあいだに改行が入っている場合にもコンマが 1 個必要であることに注意すること。変更後のファイルは, たとえば図 9.25 のようになる。変更がおわったら, 画面上端

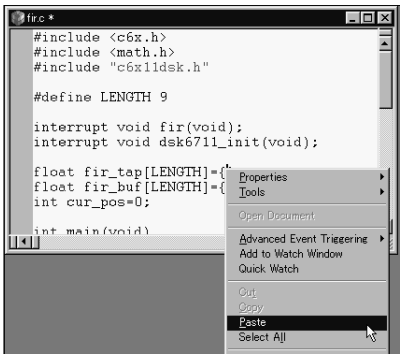

図 9.24 fir.c へのタップ係数の書き込み (1)

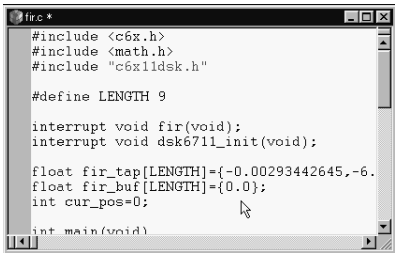

図 9.25 fir.c へのタップ係数の書き込み (2)

のメニューから

File → Save

を選択して, 編集後のファイルを保存する (図 9.26)。

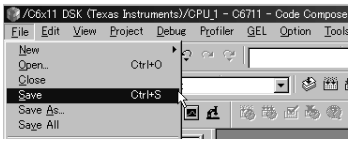

図 9.26 変更後の fir.c の保存

3.2.4.4　コンパイル　ソースプログラムの変更が終わったら, プログラムをコンパイルして DSP に送るためのデータを作る。このためには, Code Composer Studio の画面上端のメニューから

Project → Rebuild All

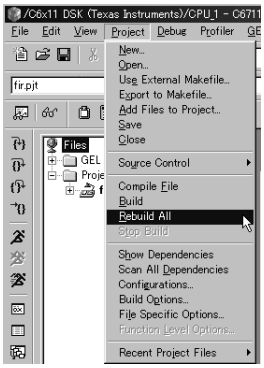

図 9.27 プログラムのコンパイル

を選択する (図 9.27). コンパイルに成功すると, 画面下部のウインドウに

```
Build Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

というメッセージが出る。この状態になったら, 作成したプログラムを DSP で実行できる (図 9.28)。

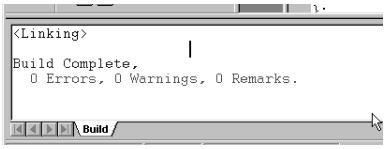

図 9.28 コンパイルに成功した画面

コンパイル時にエラーや警告が出た場合には, ソースプログラム fir.c に問題があるので, ソースプログラムを修正してから再度 Rebuild All を実行すること。

3.2.5 実行可能プログラムのロードと実行

コンパイルが成功して実行可能プログラムが完成した後には, いつでもこのプログラムをロードして実行することができる。だから, ソースプログラムを変更した場合を除き, 1 回コンパイルが成功すれば, それ以降は再コンパイルは不要である。

3.2.5.1 実行可能プログラムのロード 実行可能プログラムをロードするには, Code Composer Studio の画面上部のメニューから

File → Load Program...

を選択する (図 9.29)。すると図 9.30 のようなウィンドウがあらわれるので, フォルダ Debug をダブルクリックする。すると, ウィンドウの内容は図 9.31 のように変わる。このウインドウ内で fir.out をダブルクリックすると, プログラムのロードが終了する。

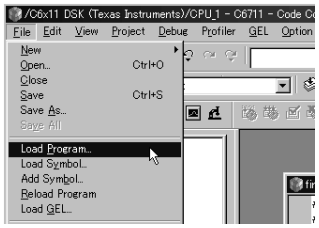

図 9.29 実行可能プログラムのロード (1)

図 9.30 実行可能プログラムのロード (2)

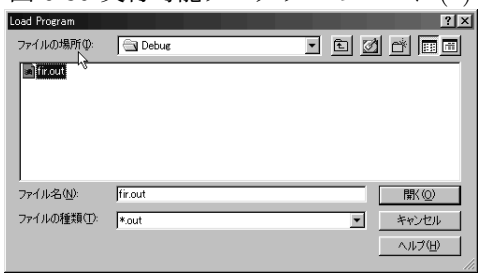

図 9.31 実行可能プログラムのロード (3)

3.2.5.2 実行と停止 Code Composer Studio 左端のアイコンの中で, 人が走っているように見えるものがプログラムの実行のアイコンである。ここをクリックするとプログラムの実行が始まる (図 9.32)。プログラム実行中は, DSP ボード のオーディオ入力端子から入力された信号が DSP によってリアルタイムで処理され, 結果がオーディオ出力端子に出力される。人が止まっているように見える絵をクリックするとプログラムが停止する。

プログラムを実行した後は, 停止することを忘れないこと。

3.2.6 フィルタの動作確認

フィルタの動作確認をするには, カセットテープレコーダのオーディオ出力端子を DSP ボードのマイク入力端子に接続し, DSP ボードのオーディオ出力端子をスピーカを接続して音楽を再生する。続いて,

図 9.32 プログラムの実行と停止

1. フロッピーディスクドライブにフロッピーディスクを挿入する
2. Code Composer Studio を起動する
3. Code Composer の File → Load Program... メニューで fir.out を読み込む
4. Code Composer の 実行 ボタンを押す

という手順をおこなう。すると, 作成したフィルタによって処理された音ををスピーカで聞くことができる。

雑音がある程度消えていることが確認できれば測定に進む。雑音が消えていない, あるいはスピーカから音がまったく出ないときには,

- ケーブルの接続が正しいことを確認する
- プログラムを見直す
- DSP ボード をリセットする
- DSP ボード の電源を再投入する
- パソコンを再起動する

という順で確認作業および機材の再起動をおこない, なお状況が好転しない場合は担当教官に申し出ること。

なお, これに続いてスペクトルアナライザを用いた波形観測による波形観測をおこなう場合には, 再生音が割れて聞こえないようにカセットテープレコーダのボリュームを調整するとよい (再生信号の電圧が過大である場合がある)。また, 続いて波形観測をおこなう場合には, 次節に述べる操作手順のうち, File → Load Program の部分と 実行 の部分は無視してよい。続いて波形観測をおこなわない場合には, Code Composer Studio の停止ボタンを押して DSP ボードを停止してから Code Composer Studio を終了する。

3.2.7 スペクトルアナライザを用いた波形観測

スペクトルアナライザを用いて DSP によって処理された後のスペクトルを観測するには, WaveSpectra および CodeComposerStudio が起動されている状態で,

1. Code Composer の File → Load Program... メニューで fir.out を読み込む
2. Code Composer の 実行 ボタンを押す
3. テープをあらかじめ決めた位置まで巻き戻す
4. カセットテープレコーダのオーディオ出力端子を DSP ボードのマイク入力端子に繋ぐ
5. DSP ボードのオーディオ出力端子をパソコンのマイク入力端子に繋ぐ
6. WaveSpectra の録音ボタンを押す
7. 一定時間が経過したら WaveSpectra の停止ボタンを押す (カセットテープレコーダのカウンタを利用するとよい)

という一連の作業をおこなう。フィルタの特性を見るためには, スペクトルのピーク値のみを表示するとよい。

測定値のグラフをファイルに保存するためには, スクリーンキャプチャソフトウェアである WinShot [3] (フリーウェアである) を使う[4]。

WinShot を起動するには, デスクトップ左端にある WinShot というアイコンをダブルクリックする。すると, タスクバー右端に WinShot のアイコンがあらわれる。

続いて WaveSpector のウィンドウをキャプチャするわけであるが, このためには,

1. WaveSpectra のウィンドウをアクティブにする (ウィンドウのどこかでマウス左ボタンを押す)
2. タスクバー右端の WinShot のアイコンを右クリックする
3. JPEG 保存→アクティブウィンドウを選択する

という手順をおこなう (図 9.33) 。すると, デスクトップに画像データが保存される。ファイル名は ws00.jpg から始まり, 番号が自動的に増えてゆく。

なお, フィルタの特性を見るためには, DSP ボードによって処理される前のスペクトルデータも必要である。このデータを取るためには, カセットテープレコーダのオーディオ出力端子をパソコンのマイク入力端子に繋ぎ換えてから, 上記と同様の手順をおこなう。

[3] http://www.woodybells.com/

[4] WaveSpectra にもグラフを保存する機能があるが, データを bitmap 形式でしか保存できないという制限があり, bitmap 形式のデータはファイルサイズが大きくなるので, ここでは使用しない。

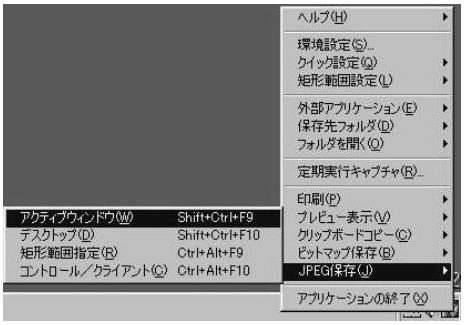

図 9.33 WinShot による画像の保存

3.2.8 周波数特性の測定

フィルタの周波数特性を測定するには, 図 9.34 のようにフィルタの入力側に発振器を接続し, 周波数を変えながら入力電圧と出力電圧をスイッチで切り換えながらディジタルマルチメータあるいはディジタルテスターで測定すること。発振器の出力電圧は 1V 程度にする。

ある周波数におけるフィルタのゲインは,

$$\text{ゲイン} = 20\log_{10}\frac{\text{出力電圧}}{\text{入力電圧}} \qquad (9.5)$$

によって計算される。ゲインの単位は dB(デシベル) である。位相特性は測らなくてよい。

この実験で使用する DSP ボード は直流電流を遮断する設計になっているので, あまり低い周波数についてデータを取ることは無意味である。また, ナイキスト周波数 (4kHz) 以上の周波数の信号は DSP ボード に付属している低域通過フィルタで減衰してしまうので, あまり高い周波数についてもデータを取ることは無意味である。

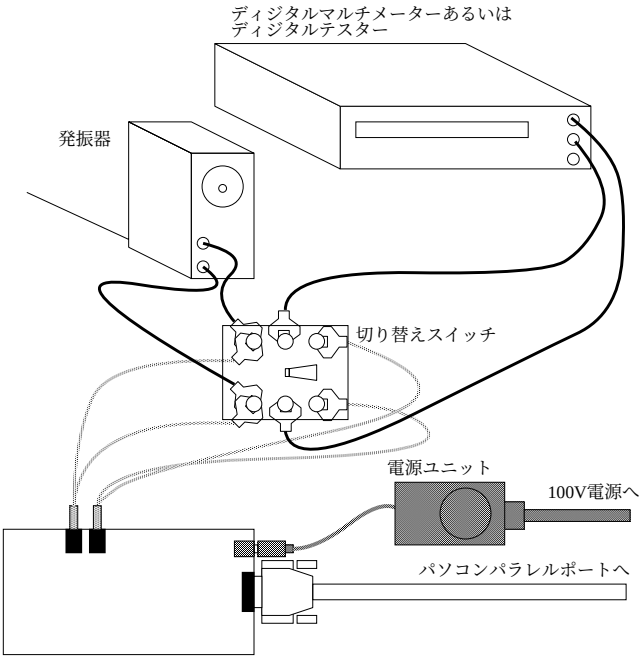

図 9.34 周波数特性の測定

4. データ解析

- フィルタの周波数特性の設計値と実測値を比較し, 特性にずれがある場合にはずれの要因について検討せよ。
- 実測されたフィルタの周波数特性とスペクトルアナライザで観測された波形における雑音の減衰量を比較せよ。

参考

同軸ケーブル, ピンプラグ, ミニプラグの構造

オーディオ機器の出力端子どうしを接続するために使われているケーブルの多くは, 同軸ケーブルと呼ばれるものである。このケーブルは, 外見上は 1 本だけの線なのだが, 内部には芯線とグランド線の 2 本の線が通っている。芯線とグランド線のあいだは絶縁材で絶縁されている。図 9.35 に同軸ケーブルの構造を示す。

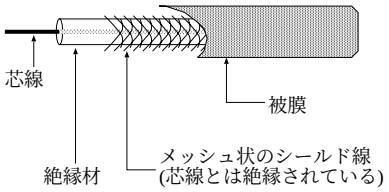

図 9.35 同軸ケーブルの構造

機器同士を接続して電気回路を閉じるためには線が 2 本必要である。これに対応して, オーディオ機器の出力端子 (ピンジャックやミニジャックなど) にも 2 本の線に対応する部分があり, それらに挿入されるピンプラグやミニプラグも 2 本の線に対応する部分を持っている。

ピンプラグの構造を図 9.36 に示す。図 9.36 において, オーディオ機器との接続部の中央のピン A と外縁部の円筒形の金属部品 B は絶縁材によって絶縁されている。部品 A は同軸ケーブルの芯線 A' と接続され, 部品 B は同軸ケーブルのシールド線 B' と接続される。

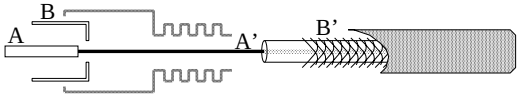

図 9.36 ピンプラグの構造

ミニプラグの構造を図 9.37 に示す。図 9.37 において, オーディオ機器との接続部のピンは絶縁材で先端の部品 A と根本の部品 B に区切られている。部品 A は同軸ケーブルの芯線 A' と接続され, 部品 B は同軸ケーブルのシールド線 B' と接続される。

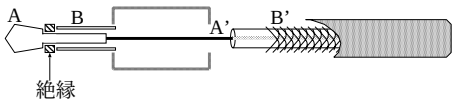

図 9.37 ミニプラグの構造

これにより, オーディオ機器の出力信号をディジタルマルチメータやスペクトルアナライザで測定したいときには, 図 9.38 のようなケーブルを作製しておけばよいことがわかる。図 9.38 のようなケーブルは実験室にあらかじめ用意されている。

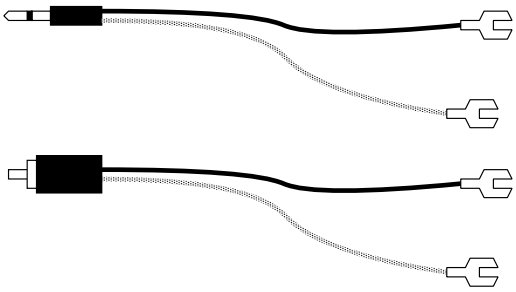

図 9.38 オーディオ機器に接続する測定用ケーブル

スペクトルアナライザの縦軸の表示について

観測された時間信号を $x(t)$, そのフーリエ変換を $X(w)$ としたとき, フーリエ変換の定義

$$X(w) = \frac{1}{2\pi} \int x(t) \exp[-jwt] dt$$

により, 信号 $X(w)$ の単位は, 信号 $x(t)$ の単位に時間を乗じたものになる。たとえば, 信号 x の単位が電圧 [V] で, 時間の単位が秒 [s] であるときには, X の単位は [Vs] となる。ところで, スペクトルを表示するときには縦軸を対数軸に取ることが多いのであるが, 対数を取るためには, その対象となる変数が無次元量になっていないと都合が悪い。そこで, スペクトルアナライザでは, 形式的に, 測定信号が 1Vs で規格化されているとみなして, 機械的に $X(w)$ の対数を取り, 慣習的に, 縦軸の単位を [dBV] と表示することが多い。

一方, 縦軸の単位を [dB] で表示するものもある。本実験で利用している WaveSpectra は後者にあたる。ただし, 縦軸の単位を [dB] と表示することは, 単位 [dB] の意味を考えると, 必ずしも適切とはいえない。

文献

[1] 樋口 龍雄: 「ディジタル信号処理の基礎」, 昭晃堂 (1986)

[2] 電子情報通信学会 (編): 「ディジタル信号処理ハンドブック」, オーム社 (1993)

[3] 瀬谷: 「DSP C プログラミング入門」, 技術評論社 (2000)