9. ディジタルフィルタ

1. 目的

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。

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 $x(t)$

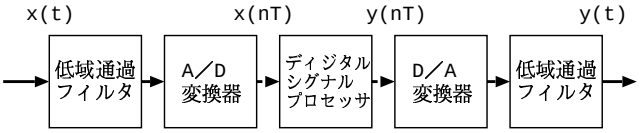

図 9.1 連続時間信号のディジタル信号処理

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 $\{x(0), x(T), \dots\}$ 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 $\{y(0), y(T), \dots\}$ 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。

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

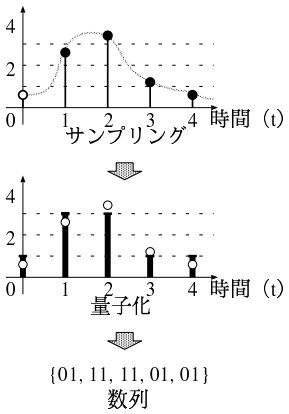

図 9.2 A/D 変換器の処理の流れ

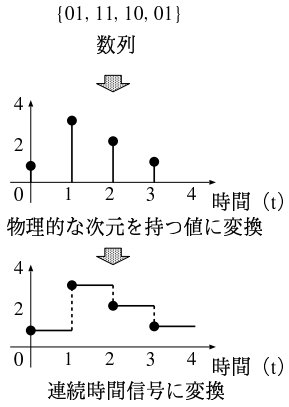

図 9.3 D/A 変換器の処理の流れ

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 $x_k, x_{k-1}, \dots, x_{k-n}$ 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。

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。

$$x_k = e^{j2\pi fkT} \tag{9.2}$$

とする。 式 (9.2) を式 (9.1) に代入すると,

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

が得られる。

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

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。

2.4 固定小数点演算と q フォーマット

固定小数点演算とは, プログラマが, 整数のデータ型を表現するための n ビットのパターンのうち, 第 k 番目のビットと第 $k+1$ 番目のビットのあいだに小数点があると「思って」プログラムを組む方法である。このような方法を取ることで, 整数に対応するデータ型しか持たないプロセッサで実数の演算を近似することができる。

数を 2 進数で表現したとき, 上記の仮想的な小数点以下の数が n ビット分あるデータのフォーマットのことを qn フォーマットという。

qn フォーマットを使って計算をおこなうときには, 加算については整数の演算がそのまま使えるが, 乗算については整数の乗算をおこなったあとで固定小数点の位置に応じた桁数分だけ結果をシフトしなければならない。また, プログラマは桁落ちや桁溢れに注意する必要がある。詳細は文献 [3] を参照すること。

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 TMS320C6211 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 1.23 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。

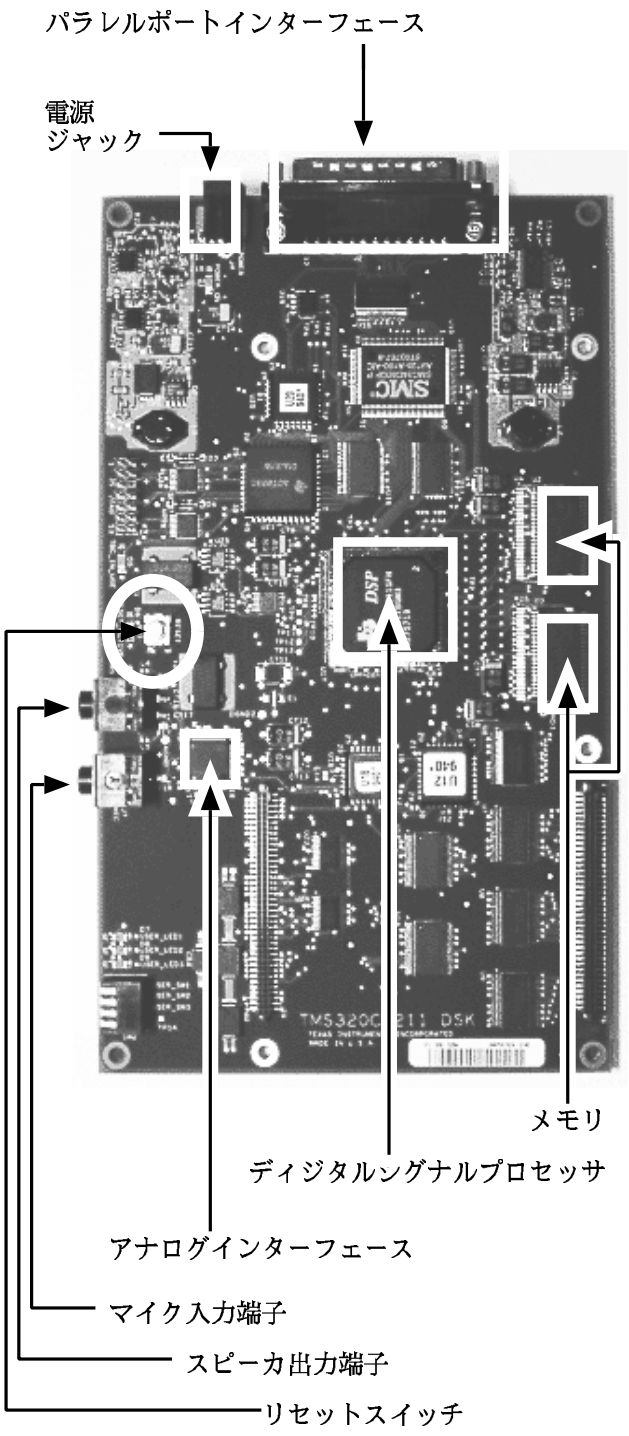

図 9.4 DSP ボード TMS320C6211 の構成

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。

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 を接続しておくこと。続いてコンピュータを起動し, さらに 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。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スペクトルアナライザを使って観測し, 雑音の周波数を確認しておくこと。

3.2.2 Code Composer Studio の練習

本節では, 用意されたソースプログラムをそのままコンパイルして実行してみる。

3.2.2.1 サンプルファイルのコピー 実験用のサンプルファイルはすべて `C:\ti\myprojects` にあるフォルダ `sample` に格納されている。これをフロッピーディスクにコピーする。Windows98 の操作に不慣れな読者のために, このための手順を以下に説明しておく。必要のない者は読み飛ばしてよい。

1. 3.5 インチ FD(A:) を開く。このためには, 画面左上の「マイコンピュータ」アイコンをダブルクリックし, 続いて, 現れたウインドウ内で「3.5 インチ FD(A:)」をダブルクリックする。
2. `C:\ti\myprojects` を開く。このためには, 画面左上の「マイコンピュータ」アイコンをダブルクリックし, 続いて, 現れたウインドウ内で `(C:)`, `ti`, `myprojects` というアイコンをダブルクリックしてゆく。
3. `C:\ti\myprojects` のウインドウ内で `sample` アイコンを右クリックする。開いたウインドウ内で「コピー (C)」を選択する。このためには, ウインドウ内でマウスカーソルを上下に動かして色が青くなっている部分を「コピー (C)」に合わせ, マウス左ボタンを 1 回押して離す。
4. 3.5 インチ FD(A:) のウインドウに移動して右クリックし, 現れたメニューから「貼り付け (P)」を選択する。

3.2.2.2 Code Composer Studio の起動

スタート→プログラム (P) →
6211-6711 DSK Development Tools
→ CCStudio DSK 'C6000 1.23
を選択する。

上記では意味のわからない読者のために, より詳しい手順を以下に説明しておく。 必要のない者は 読み飛ばしてよい。

1. まず画面左下にある「スタート」アイコンをクリックする。
2. 現れたウインドウ内でマウスカーソルを上下に移動させ, 色が青くなっている部分を「プログラム (P)」に合わせ, 続いてマウスカーソルを右に移動させる。すると新たなウインドウが開くので, 再びマウスカーソルを上下に移動させ, 色が青くなっている部分を「6211-6711 DSK Development Tools」に合わせ, マウスカーソル右に移動させる。新しく開いたウインドウの中でマウスカーソルを上下に移動させ, 色が青くなっている部分を「CCStudio DSK 'C6000 1.23」に合わせてマウス左ボタンをクリックする。

上記の手順によって Code Composer Studio が起動する (図 9.5)。なお, 図で表示されているメニューの内容はコンピュータによって異なる場合があるので注意すること。

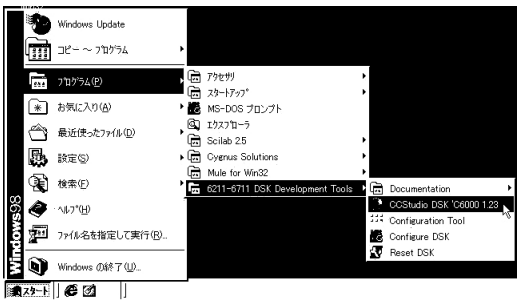

図 9.5 Code Composer Studio の起動

Code Composer Studio が起動すると, 画面に図 9.6 のようなウィンドウがあらわれる。

3.2.2.3 新しいプロジェクトの作成 プロジェクトとは, 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 としておく。

プロジェクト作成の手順は以下の通りである。

1. Code Composer Studio の Project メニューから New を選択する (図 9.7)。
2. フロッピーディスク内の sample というフォルダに移動する。このためには, 図 9.8 中に表示されたアイコンを適宜クリックしてゆけばよい。
3. ファイル名を fir にして「保存」をクリックする (図 9.9)。

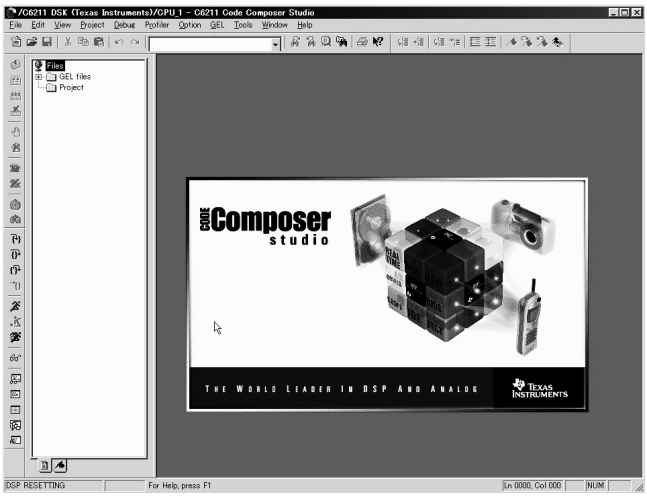

図 9.6 Code Composer Studio が起動したところ

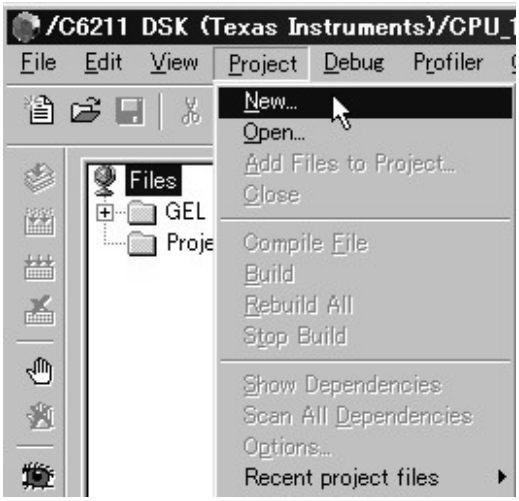

図 9.7 新規プロジェクトの作成 (1)

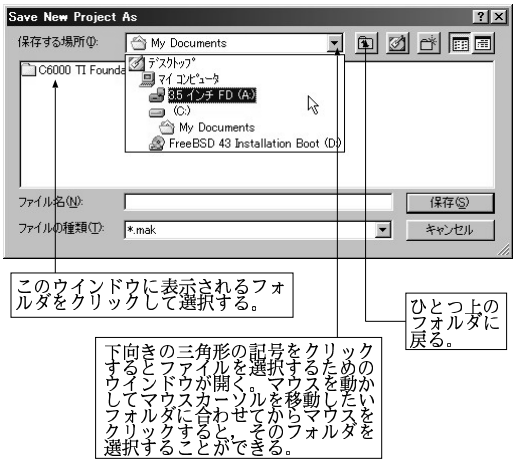

図 9.8 新規プロジェクトの作成 (2)

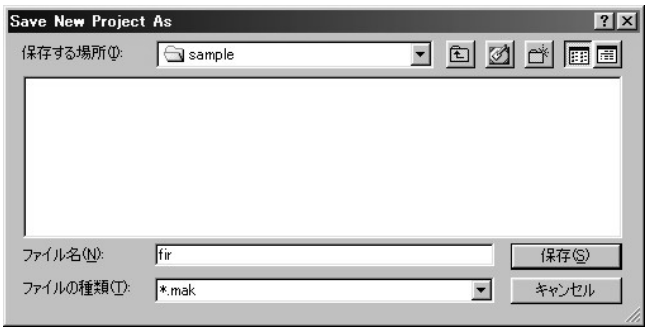

図 9.9 新規プロジェクトの作成 (3)

3.2.2.4　プロジェクトが作成されていることを確認する　Code Composer Studio の左側のウインドウ内にある Project というアイコンの左側の+記号をクリックし, 続いて fir.mak というアイコンの左側の+記号をクリックする。図 9.10 のようになっていたらプロジェクトは正常に作成されている。

図 9.10 プロジェクトの作成が終了した状態

ここで作成された fir.mak というファイルは, いくつかのソースプログラムとライブラリファイルおよびヘッダファイルなどに基づいて実行可能プログラムを生成するために必要な規則が記載された, makefile と呼ばれるファイルである。

3.2.2.5　必要なファイル群を makefile に追加する　Code Composer Studio はユーザが選択したファイルから正しく実行可能プログラムが生成されるように自動的に makefile を変更する。以下ではそのための手順を実行してみる。

rts6201.lib の追加　Project メニューから Add files to Project... を選択する (図 9.11)。ファイル選択のた

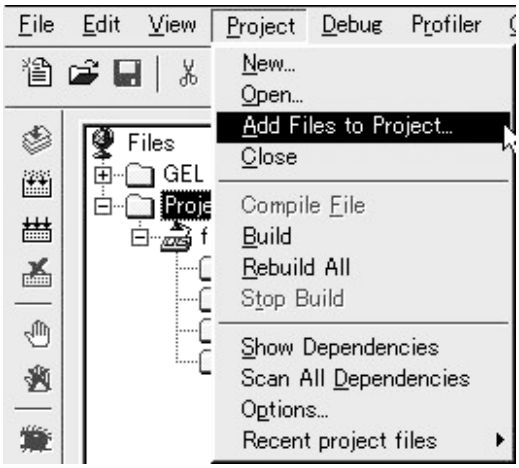

図 9.11 ファイル追加メニュー (1)

めのウインドウが現れるので, 3.5 インチ FD(A) の中の sample というフォルダに移動し, 続いて画面最下部の「ファイルの種類」の部分の右側の下向き三角印をクリックし, 「Object and Library Files(*.o*,*.lib)」を選択する。最後にその左側のウィンドウで rts6201.lib を選択 (クリック) してから画面右下の「開く」をクリックする (図 9.12)。

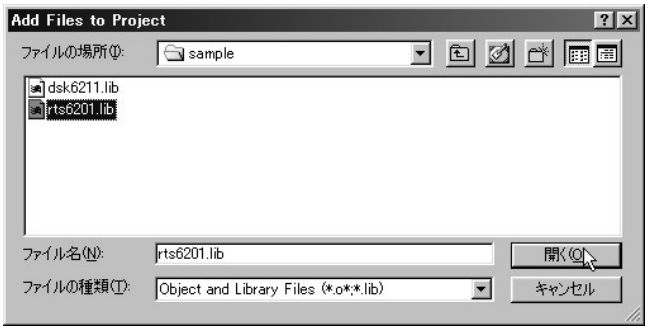

図 9.12 ファイル追加メニュー (2)

dsk6211.lib の追加 Project メニューから Add files to Project... を選択する。必要なら 3.5 インチ FD(A) の中の sample というフォルダに移動し (以下同じ), 続いて画面最下部の「ファイルの種類」の部分の右側の下向き三角印をクリックしてから, その左側のウィンドウで「Object and Library Files(*.o*;*.lib)」を選択 (クリック) する。最後に dsk6211.lib をクリックしてから画面右下の「開く」をクリックする。

fir.cmd の追加 Project メニューから Add files to Project... を選択する。続いて画面最下部の「ファイルの種類」の部分の右側の下向き三角印をクリックし, 「Linker Command File(*.cmd)」を選択する。最後に fir.cmd をクリックしてから画面右下の「開く」をクリックする。

vectors.asm の追加 Project メニューから Add files to Project... を選択する。画面最下部の「ファイルの種類」の部分の右側の下向き三角印をクリックし, 「Asm Source Files(*.a*;*.s*)」を選択する。最後に vectors.asm をクリックしてから画面右下の「開く」をクリックする。

dsk6211.c の追加 Project メニューから Add files to Project... を選択する。画面最下部の「ファイルの種類」の部分の右側の下向き三角印をクリックし, 「C/C++ Source Files(*.cpp;*.cc;*.cxx;*.c;*.sa)」を選択する。最後に dsk6211.c をクリックしてから画面右下の「開く」をクリックする。

main.c の追加 dsk6211.c と同様の手順にしたがって main.c を追加する。

上記のファイル群のうち, `*.lib` という拡張子の付いたファイルは, 実行可能なプログラムの部品が含まれるライブラリファイルと呼ばれるものである。 `*.cmd` という拡張子が付いたファイルは, コマンドファイルとファイルと呼ばれる, メモリにどのようにデータやプログラムを配置するかが指示されたファイルである。vectors.asm というファイルは, 割り込みと呼ばれる処理が定義されているアセンブリ言語で書かれたプログラムである。dsk6211.c はパラレルポートとの通信に必要な関数などが定義された C 言語のソースプログラムである。 また, main.c は FIR フィルタの本体である。 本実験の課題を実行するためには, 上記のファイルがすべて必要である。

3.2.2.6 コンパイル Project メニュの Rebuild All を選択する。すると, プログラムのコンパイルが始まる。画面下部のウインドウに 0 Errors, 0 Warnings と表示されたら成功である (図 9.13)。コンパイルに失敗したときは, このウインドウの右端のスクロールバーを使ってエラーメッセージを探し, ソースプログラム中の誤りを訂正する必要がある。

3.2.2.7 実行可能プログラムのロード File メニューから Load Program を選択する (図 9.14)。すると Load Program ウインドウが現れる (図 9.15)。フォルダ `A:\sample` を選択し, 続いて fir.out を選択して「開く」をクリックする。ここに, fir.out というのは今生

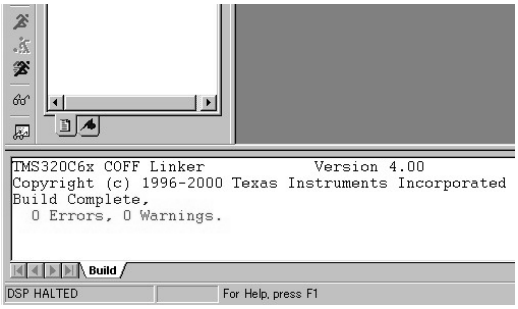

図 9.13 プログラムのコンパイル

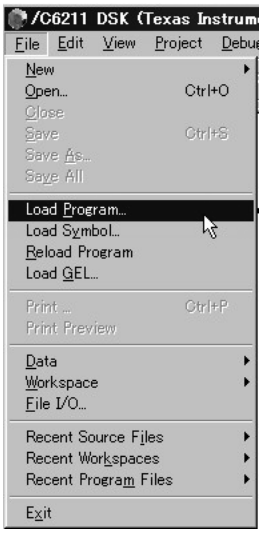

図 9.14 プログラムのロード (1)

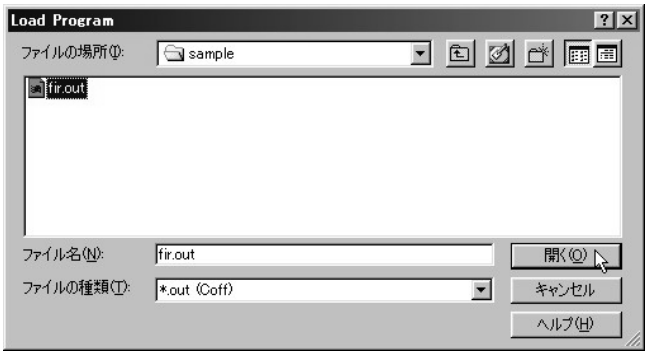

図 9.15 プログラムのロード (2)

成された実行可能プログラムの名称である。実行可能プログラムの名称はプロジェクトの名称と同一になる。

3.2.2.8　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 9.16)。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。

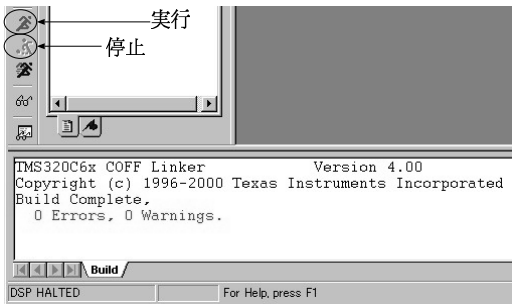

図 9.16 プログラムの実行と停止

3.2.2.9　終了　File メニューから Exit を選択すると Code Composer Studio は終了する。

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。

フィルタの設計にあたって, 諸君のフロッピーディスクにすでにコピーされてい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:\sample\design.sci');↩
```

と入力する。すると, フィルタの設計が実行され。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にある hn.txt というファイルに書き出される。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 hn.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 9.17 に示す。

続いて, 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する。

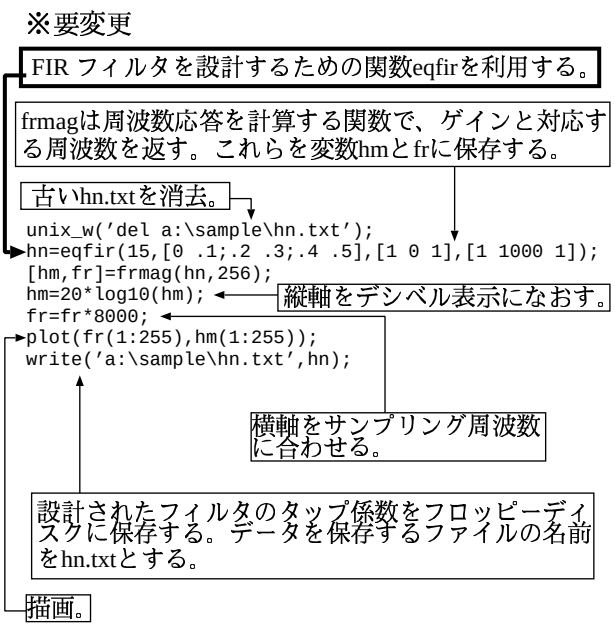

図 9.17 FIR フィルタ設計プログラム (design.sci) の説明

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。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 9.17 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 9.18 に示す.

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 hn.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。

Mule を起動するには,

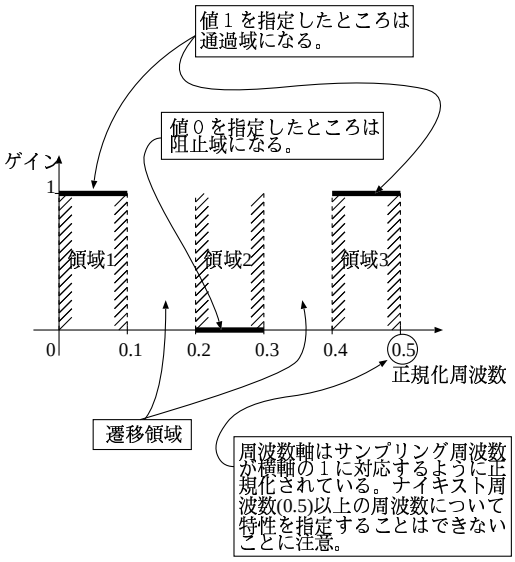

図 9.18 関数 eqfir におけるフィルタの特性の指定

スタート→プログラム (P)
→ Mule for Win32 → Mule for Win32

を選択するか, 画面左端の Mulw for Win32 というアイコンをダブルクリックする。また, メモ帳を起動するには,

スタート→プログラム (P)
→アクセサリ →メモ帳

を選択する。

諸君はすでに Mule には慣れているはずなので, Mule の操作法については特に説明しない。

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 フィルタのタップ係数を q14 フォーマットに変換する

Scilab が設計したフィルタのタップ係数が保存されたファイルには, 数値が小数で記録されている。ところで, この実験で使う DSP には浮動小数点演算機能はないので, このデータをプログラムに書き込むことに先立って, ファイルの内容を固定小数点 (見掛け上は整数と同じ) に変換しておかなければならない。この変換をおこなうためのプログラム q14.exe が用意され, すでに フォルダ `a:\sample` に複写されている。

このプログラムを実行するときには, まず bash を起動し,

`q14` 入力ファイル 出力ファイル⏎

という操作で変換をおこなう。

bash を起動するには

スタート→プログラム (P)
→ Cygnus Solutions → Cygwin B20

を選択する。

例えば, フロッピーディスクのフォルダ `a:sample\`直下にある hn.txt というファイルを q14 フォーマットに変換して fir-q14.txt というファイルを作るには, bash のウィンドウの中で

```
cd //a/sample⏎
q14.exe hn.txt hn-q14.txt⏎
```

と入力する。

3.2.5 main 関数の編集

続いて, フォルダ `a:\sample` にあるファイル main.c を諸君が作成したフィルタに対応するものに書き換える。

main.c の内容および簡単な説明を図 9.19 に示す。

図 9.19 のプログラム中で変更の必要があるのは,「※要変更」という印がつけられた定数 `LENGTH` と配列 `h` だけである。定数 `LENGTH` の部分については, 数値を諸君が設計したフィルタの長さに変更する。配列 `h` については, 初期化に使う数値を諸君が設計したフィルタのタップ係数 (上の例におけるファイル hn-q14.txt の内容) で置き換える。

ファイルの編集には Mule やメモ帳などが使えるほか, Code Composer Studie にもファイルの編集機能がある。これらの操作法については特に説明しないが, Mule においてあるファイルに他のファイルの内容を挿入するコマンドが

Ctrl+x i

(まず Ctrl キーを押しながら x キーを押し, 続いてキー i を押す) であることだけは復習しておく。画面上端の File メニューの Insert File を選択することでも, ファイルの挿入が実行できる。

3.2.6 コンパイルおよび実行

3.2.6.1 コンパイル プログラムをコンパイルして実行するためには, Code Composer Studio を起動す

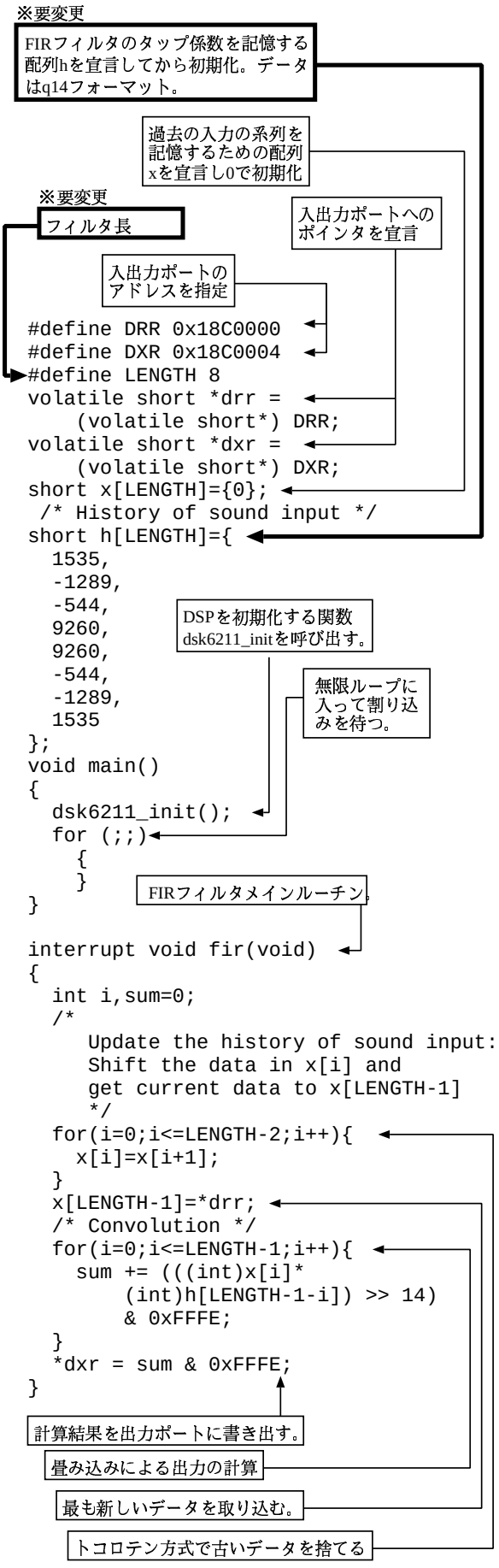

図 9.19 FIR フィルタ本体 (main.c) の説明

る。続いて第 3.2.2.6 節で説明されているコンパイルの操作をおこなう。コンパイルエラーが出たときには第 3.2.5 節に戻って main.c を訂正する。

3.2.6.2　実行　続いて DSP ボード をリセットし, プログラムをロードして実行する。このための手順は第 3.2.2.7 節以降の部分と同じである。DSP ボード に異常があるときには, DSP ボード のリセットボタンを押すか, DSP ボード の電源を一旦抜いてから再度電源を入れる。それでも正常に動作しない場合にはコンピュータを再起動する。

3.2.7　フィルタの動作確認

フィルタの動作確認をするには, 図 9.20 のようにフィルタの入力側にカセットテープレコーダからの信号を入力し, 出力側にアンプ内蔵スピーカを接続して音楽を再生すること。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。

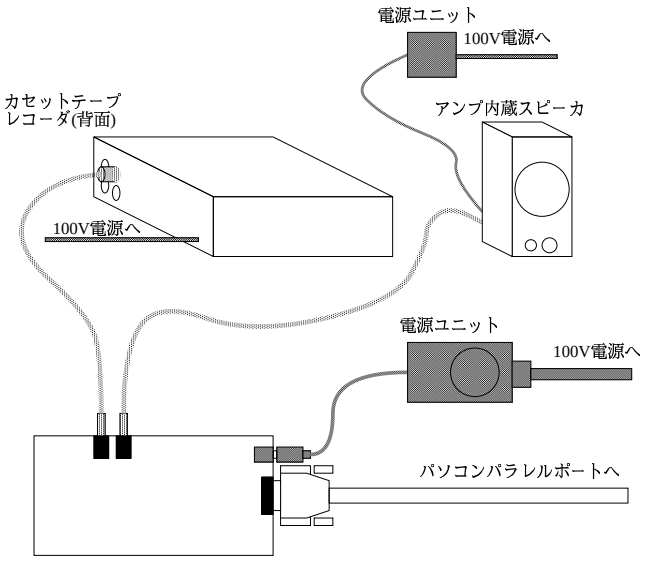

図 9.20 雑音が消えているかどうかの確認

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 9.20 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。

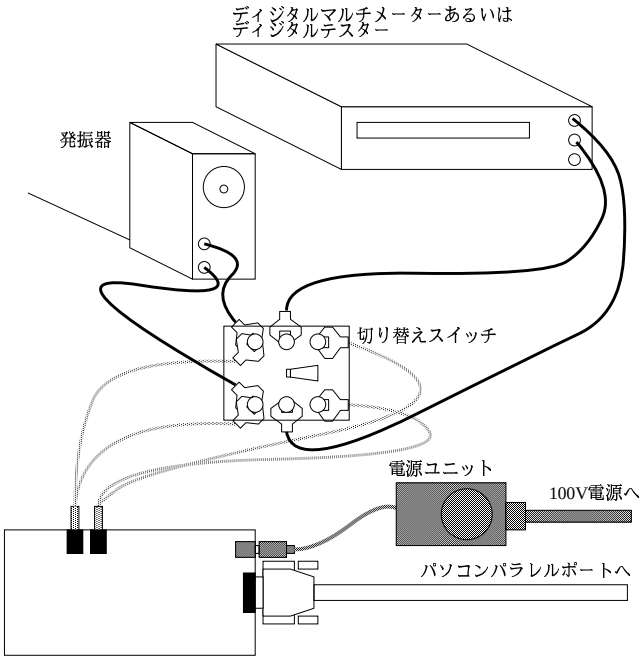

図 9.21 周波数特性の測定

3.2.9　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波形を観測するときには, 図 9.22 のように機材を接続する。

波形観測をするときには, フィルタの入力端子にカセットテープレコーダから再生される音楽を入力する。

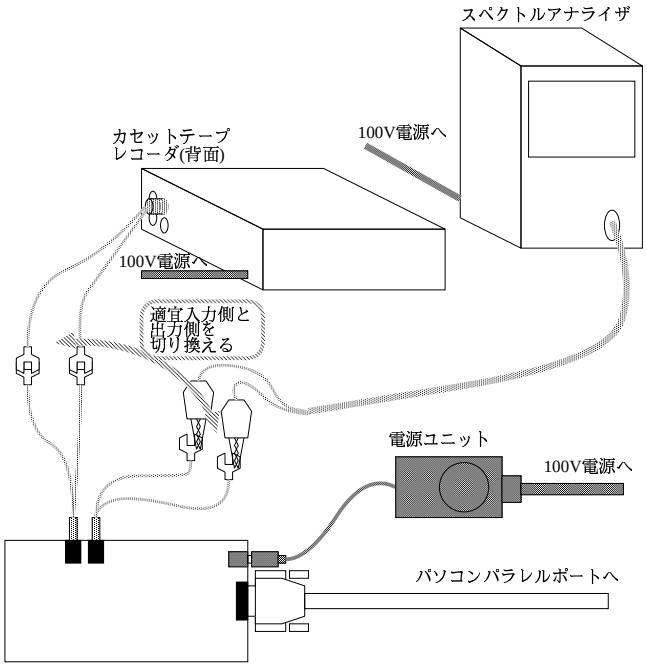

図 9.22 スペクトルアナライザを用いた波形の観測

そして, フィルタによる処理前の波形と処理後の双方を観測し, これらをフロッピーディスクに保存する。これらの操作の手順は**参考**の節に記載されているので参照すること。

カセットテープレコーダに付属しているカウンタを使って音楽の再生位置を指定し, フィルタによる処理前のデータを取るときと処理後のデータを取るときで入力信号が同一になるように工夫すること。

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

スペクトルアナライザの使い方

図 9.23 にスペクトルアナライザの正面図を示す。説明の便宜のために, 実験で実際に操作する必要があるボタンには数字が付けてある。

波形の観測

波形を観測するときには, 以下の手順にしたがう。

1. スペクトルアナライザ背面にある電源スイッチを投入する。
2. フロッピーディスクをスペクトルアナライザ正面下部のフロッピーディスクドライブ **(10)** に挿入する。
3. **(9)** を 2 回押し, グラフの横軸の最大値を 10kHz に合わせる。
4. ロータリースイッチ **(8)** を回し, カーソル (画面で光っている部分) を **(3)** に合わせて SELECT ボタン **(5)** を押す。
5. ロータリースイッチ **(8)** を回し, 数字を −100 に合わせて SELECT ボタン **(5)** を押す。
6. ロータリースイッチ **(8)** を回し, カーソルを POWER SUM**(1)** に合わせて SELECT ボタン **(5)** を押す。
7. ロータリースイッチ **(8)** を回し, 数字を 32[1]に合わせて SELECT ボタン **(5)** を押す。
8. START ボタン **(7)** を押す。
9. テープを開始位置に合わせて再生を開始するのと同時に, AVG STAR ボタン T**(6)** を押し, POWER SUM**(1)** の読みが 32/32 [2]になるまで待つ。

データのフロッピーディスクへの保存

以下のような操作をすると, スペクトルアナライザの画面に表示された波形データをフロッピーディスクに保存することができる。

1. ロータリースイッチ **(8)** を回し, カーソルを DISK**(2)** に合わせて SELECT ボタン **(5)** を押す。するとディスク操作メニュー (図 9.24) が現れる。

[1]ここは平均を取る回数を指定する部分である。実際には回数は任意でよい。
[2]上記で指定した回数に準ずる。

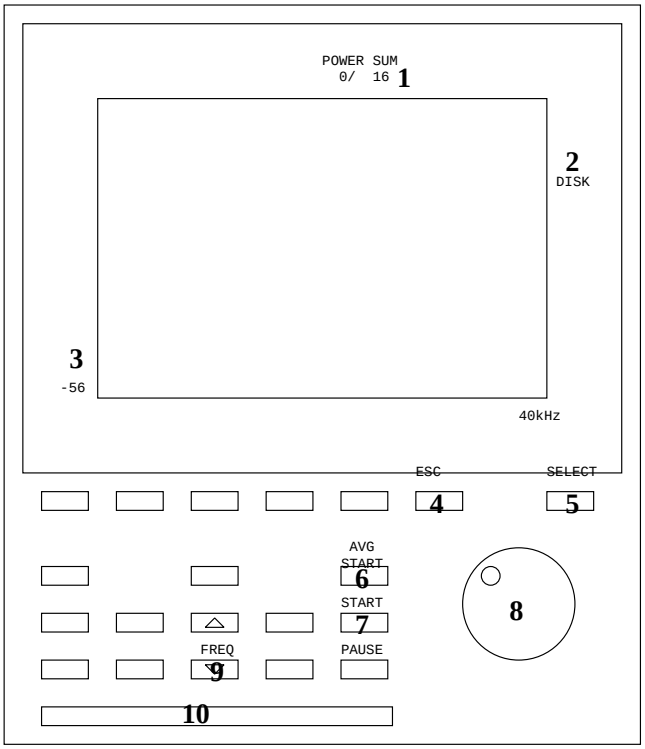

図 9.23 スペクトルアナライザ正面図

2. ロータリースイッチ **(8)** を回し, カーソルを LOAD**(11)** に合わせて SELECT ボタン **(5)** を押す。
3. ロータリースイッチ **(8)** を回し, カーソルを STORE に合わせて SELECT ボタン **(5)** を押す。
4. ファイル名を変更する必要があるときには, ロータリースイッチ **(8)** を回し, カーソルを 001**(12)** に合わせて SELECT ボタン **(5)** を押す。 続いてロータリースイッチ **(8)** を回し, 数値を適当に変更する。
5. ロータリースイッチ **(8)** を回し, カーソルを DAT**(12)** に合わせて SELECT ボタン **(5)** を押す。
6. ロータリースイッチ **(8)** を回し, カーソルを GRP に合わせて SELECT ボタン **(5)** を押す。
7. ロータリースイッチ **(8)** を回し, カーソルを EXECUTE**(13)** に合わせて SELECT ボタン **(5)** を押す。20 秒ほどでデータの保存が終了し, ディスクメニューが画面から消える。

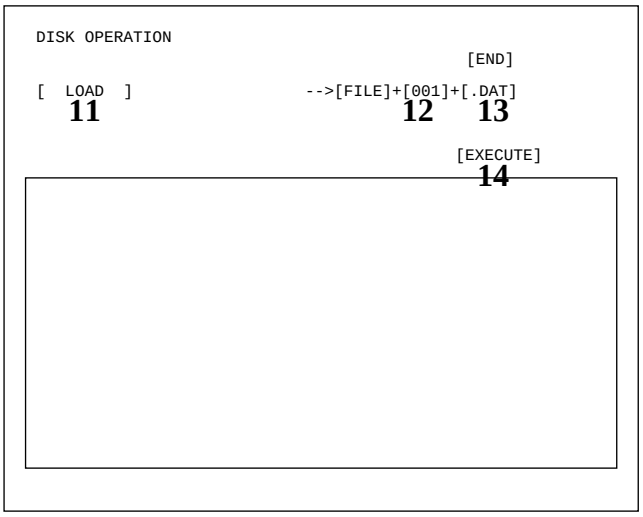

図 9.24 ディスク操作画面

フロッピーディスクに保存されたグラフを Scilab で描画する方法

以下では, file001.grp というファイルに保存されたデータを data1.sci というファイルにコピーしてから Scilab が取り扱える形に加工し, 実際に画面に描画するまでに手順を示す。 ファイル名や変数名などは必要に応じて変更すること。

1. file001.grp というファイルを data1.sci というファイルにコピーする。

2. Mule for Win32 で data1.sci を開く。 画面下部に

   File exists, but cannot be read.

   というメッセージが出る場合には,

   Ctrl+x Ctrl+q

   と入力する。

3. ファイル冒頭の

```
"ONO SOKKI CF-4220 PERSONAL FFT ANALYZER"
"CF-4200","XA11..A411~"
"61-08-15 12:18","AB11..B411~"
"   SPECTRUM    ","TX"
401,"OTFONO SOKKI CF-4220 PERSONAL
"Hz ","TS   SPECTRUM    ~"
"dBV","TXHz ~"
"  MAG","TY  MAG dBV~"
"","QV "
""
```

   という部分 (紙面の都合で 1 部略) を削除する。

4. データは例えば

```
0.0000E+00,-2.8658E+01
..(中略)...
0.0000E+00,-2.8658E+01
```

   のようになっているが, これを

```
d1=[0.0000E+00,-2.8658E+01
..(中略)...
0.0000E+00, -2.8658E+01];
```

   のように変更する。 すなわち, データの最初に “d1=[” という部分を付け加え, データの最後に “];” を付け加える。

   ここまでで, Scilab でグラフに対応するデータを d1 という変数に読み込む準備ができたことになる (変数名は自由に変更してよい)。

5. Scilab を起動して File メニューの Exec を選択し, 表示されたウィンドウの中で data.sci を選択し, 「開く (O)」をクリックする。 続いて,

```
plot(d1(:,1),d1(:,2))\rt
```

   と入力する。

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 9.25 に同軸ケーブルの構造を示す。

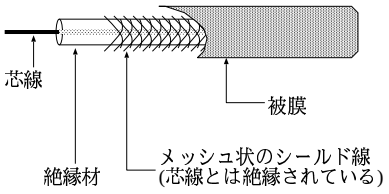

図 9.25 同軸ケーブルの構造

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 9.26 に示す。図 9.26 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。

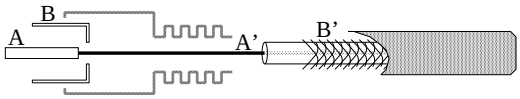

図 9.26 ピンプラグの構造

ミニプラグの構造を図 9.27 に示す。図 9.27 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。

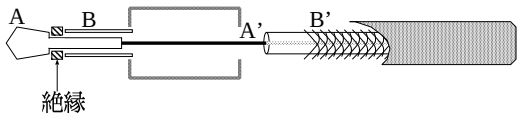

図 9.27 ミニプラグの構造

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 9.28 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 9.28 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。

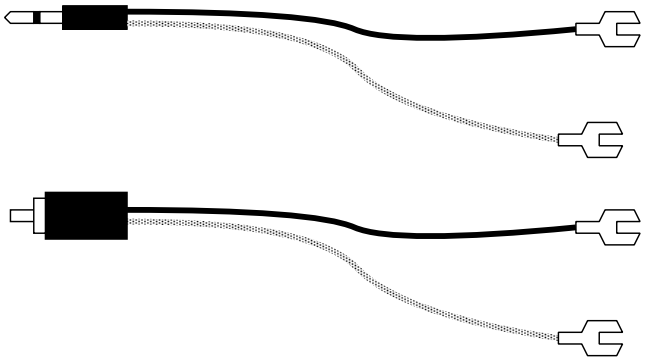

図 9.28 オーディオ機器に接続する測定用ケーブル

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)