１序論

1-1　　はじめに

近年、環境問題が深刻となっている。中でも温暖化、大気汚染、酸性雨、オゾン層の破壊など、気体の影響が大きく関与しているものが多い、その原因の多くは大気中の微量成分によるものである。これらの気体を測定することは、将来の地球環境の動向を予測していく上で非常に重要である。気体の成分を測定する方法の一つとして、波長変調分光法がある。本研究では波長変調分光法による気体の成分の測定のためのロックインアンプの製作を目的とする。しかし、アナログ方式の回路では、ロックインアンプの主要部である乗算器では直流ドリフトが発生する。また、RC ローパスフィルタは漏れ電流や素子の大きさの関係上、時定数は一定以上までしか大きくならない。そのため、直流ドリフトや時定数の影響が少ないディジタル信号処理によるロックインアンプの製作を目的とする。

1-2 目的

本研究では DSP(Digital Signal Processor)を使用し、プログラムによりロックインアンプの作成を行い、動作を確認した。

1-3 概要

ロックインアンプとは周波数変換技術を利用した測定装置である。図 1-1 にその構成を示す。入力信号と同じ周波数を持つ参照信号を乗算しローパスフィルタに通すことで入力信号の振幅が得られる。

本研究では、DSP(Digital Signal Processor)を使用しロックインアンプのプログラムを作成した。また、作成したロックインアンプを用いて雑音に埋もれた入力信号でも測定できるか動作を確認した。このロックインアンプの参照信号の周波数はすべて 400Hz で実験を行った。DSP には、TexasInstruments 社製の TMS320C6711 を使用し、同社製の開発環境 Code Composer Studio 上にて C 言語を用いてプログラムを行った。

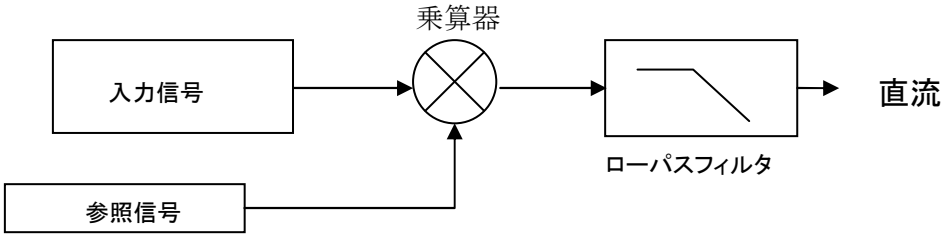


図 1-1　ロックインアンプの基本構成

2　DSP(Digital Signal Processor)

2-1 DSP とは？

ディジタル信号処理専用のマイクロプロセッサである。DSP は一般にリアルタイム・システムで使われるため、高い演算能力が要求される。そのため、特にディジタル信号処理でよく現れる連続した積と和の計算、$a_1x_1 + a_2x_{21} + \cdots + a_Nx_N$ という形式の計算を高速に実行できるように設計されている。

1970 年代後半から開発が始まり、1980 年にベル研究所が試作品を、同年 NEC が製品としてμPD7720 を発表した。1983 年にはテキサス・インスツルメンツが高性能な TMS32010 を発売、その後アナログ・デバイセズ社などがこの市場に参入し、広く使われるようになった。今日ではマルチメディア処理向けの組み込みプロセッサにおいて、DSP コアや DSP の機能を備えるものも登場している。

2-2　DSP の特徴

1）演算回路として高速の乗算器（並列乗算器）と ALU（算術論理演算ユニット）を内蔵している。

2）複数化された内部のデータバスを用いることにより、命令の実行と同時に次の命令の読み出しを行う高速処理（パイプライン処理）を可能としている。

3）内蔵されているメモリとして、プログラムを記憶するプログラム ROM と記憶するデータ ROM と、RAM がそれぞれ明確に分離されている。

4）外部の A/D、D/A 変換器と直接接続を可能とするための入出力インターフェースが内蔵されている。

これらの特徴より汎用のマイクロプロセッサでは実現できなかった高速のディジタル信号処理を可能としている。

2-3　DSP(TMS320C6711DSP)の構成

図 2-1 は本実験で使用した DSP の構成図で、CPU コア、内部メモリ、ペリフェラルに分けられる。CPU コアはデータ処理を実行するセンターであり、命令フェッチなどの命令を行う部分、データ処理の中心となる２組のデータパスから構成されている。内部メモリは２レベルのキャッシュメモリになっていて１次キャッシュは更にデータ用とプログラム用がある。２次キャッシュメモリはデータとプログラム共用である。２次キャッシュメモリはリセット時には通常メモリとして設定されているので、キャッシュとして用いるときにはモードの切り替えが必要である。

２次キャッシュとペリフェラルとのデータのやりとりはエンハンスト DMA(EDMA)コントローラを介して行う。外部メモリとのデータの授受は外部メモリのインターフェース(EMIF)を通して行う。マルチチャネル・バッファード・ポート(McBSP)は同期式のシリアルポートであり、128 チャネルまで扱うことができる。ホスト・ポート・インターフェース(HPI)は 16 ビット幅で、ホストコンピュータとのデータ授受にタイマは 32 ビット幅で 2 組あり、外部イベントや内部クロックとして使われる。

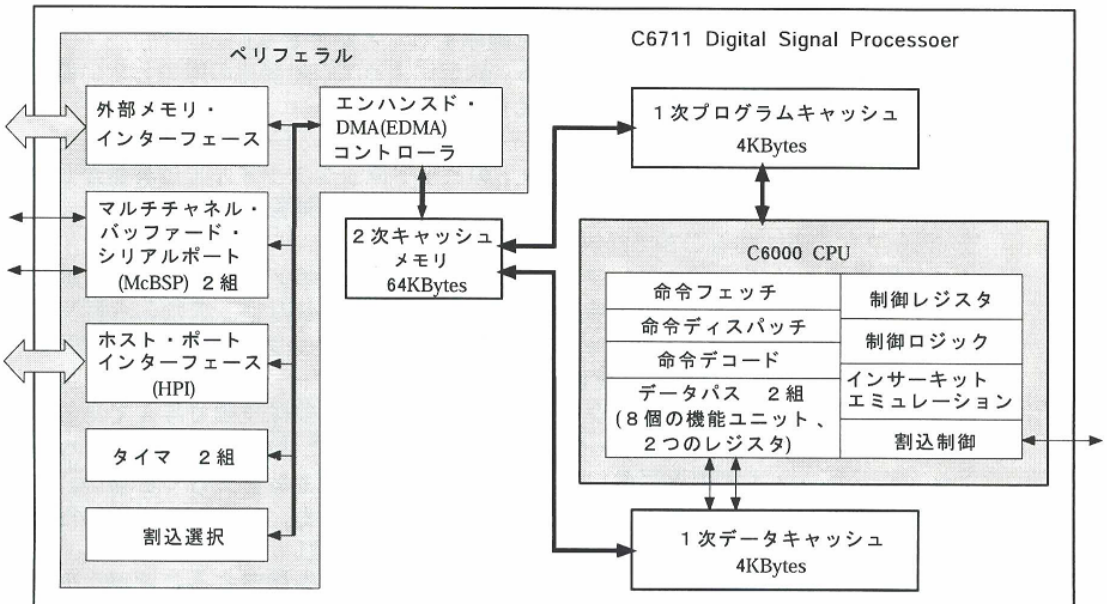


図 2-1　　TMS320C6711DSP の構成

2-4　DSP ボード(TMS320C6711DSK)の構成

実験で使用した DSK 基板の上面を図 2-2 に示す。

本実験は浮動小数点演算 DSP(TI(テキサス・インスツルメンツ)社製 TMS320C6711)に SD-RAM、フラッシュメモリ、AD/DA 変換器などを搭載した DSP スターターキット TMS320C6711DSK(以後 DSK と称する)を用いる。この DSK はコード開発ツール(オブジェクトサイズ制限つき C コンパイラ、アセンブリ・オプティマイザ、アセンブラ、リンカ)及びデバッグ・ツール(DSK 用コード・コンポーザ・スタジオ：以後 CCS と称す)が付属している。図 2-2 の右端にある 25 ピン端子(DB-25P)は、ホストコンピュータであるパソコンのプリンタポートと接続し、DB-25P の上部にある端子は専用の 5[V]電源用に接続できる。上部にはアナログ信号の入力端子と出力端子があり、本実験ではそのアナログ入出力端子を利用した。搭載されている A/D 変換器のサンプリング周波数は 8[kHz]に固定されている。すなわち、サンプリング周期は 0.125[ms]である。エリアシングを回避するため、扱える信号の最高周波数が 3.8[kHz]に制限されている。

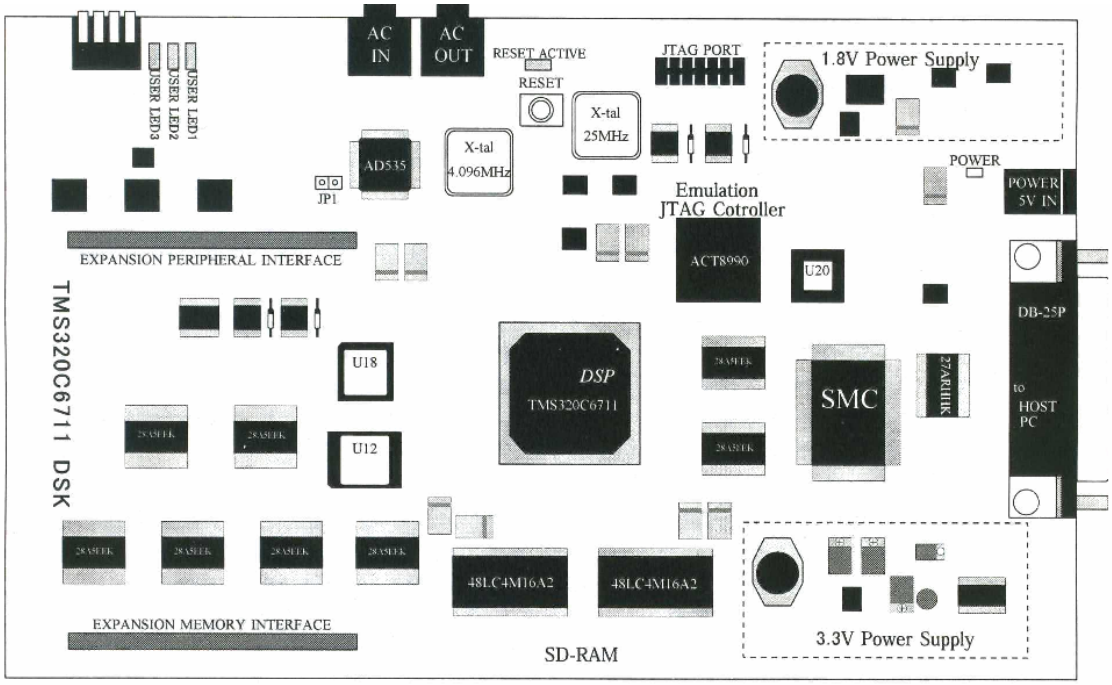


図 2-2　TMS320C6711DSK の構成図

3 ロックインアンプ

ロックインアンプは、増幅（アンプ）機能とロックイン機能を合わせ持ったものである。ロックインアンプで直接測定できるのは、正弦波（交流）の電圧または電流の実効値と位相である。この直流出力は、入力された交流信号を増幅し（アンプ機能)、交流入力信号中の特定の周波数成分のみロックする（ロックイン機能）ので、他の成分には影響されない。

ロックインアンプの大きな特長は雑音に埋もれて普通の電圧計では計れない微少な信号を測定できることである。

3-1 ロックインアンプの原理

図 3-1 にロックインアンプの原理を示す。図 3-1 はスペクトルで表されている。

雑音を取り除いて目的の信号を取り出したいとき一般には測定したい正弦波信号の周波数だけを通すバンドパスフィルタを使って、信号と周波数が異なる雑音を取り除く。フィルタの帯域幅を狭くするほど雑音が減衰する。しかし、通常のアナログフィルタでは部品の性能に限界があるため帯域幅をあまり狭くできない。また、部品の特性や信号周波数が変動するため中心周波数を常に調整する必要がある。さらに、信号の周波数が大きく変化するときは特性を維持することが困難である。このように、一般的なフィルタでは十分に帯域幅を狭めることができない。これに対してロックインアンプは雑音を取り除くための実効的な帯域幅がきわめて狭いバンドパスフィルタとして働く。目的の周波数と同じ周波数をもつ正弦波信号を目的の入力信号に乗算すると直流成分と交流成分 2f に変換できる。この直流成分のみをローパスフィルタで取り出すことで信号を検出できる。このとき f 周辺の雑音は直流周辺および 2fc 周辺に変換されている。遮断周波数が fc のローパスフィルタは、等価的に中心周波数が f で帯域が 2fc のバンドパスフィルタとして働く。参照信号と周波数が異なる雑音成分は直流にならないのでローパスフィルタで取り除くことができる。つまり、ロックインアンプは参照信号をもらって信号の周波数に追従しローパスフィルタ

で決まる帯域幅を持つバンドパスフィルタと等価な働きをする。

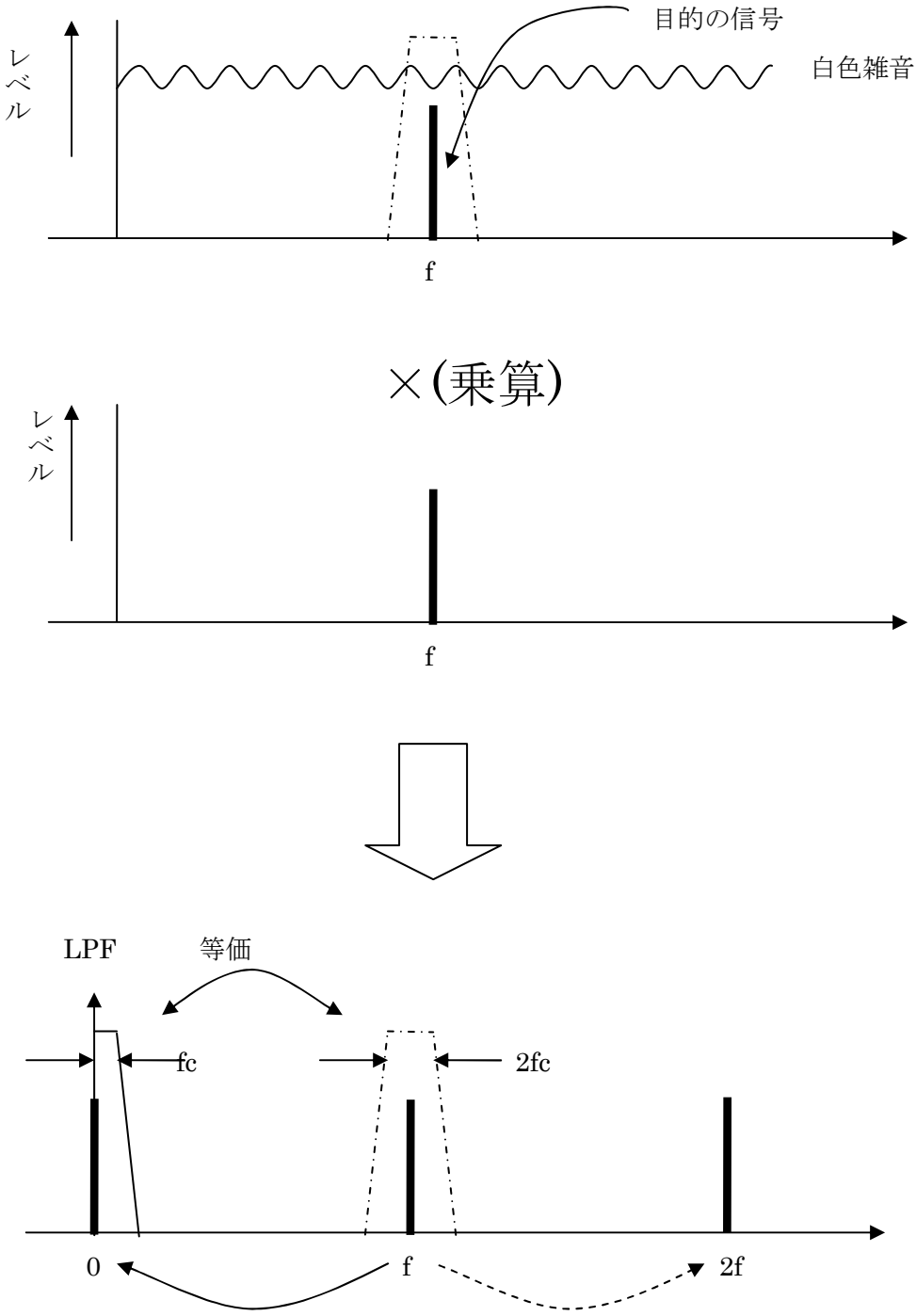


図 3-1　ロックインアンプ原理

3-3　2位相ロックインアンプ(Two Phase Lock-In Amplifier)

　ロックインアンプで信号の振幅を計測するときには、入力信号と参照信号との位相調整を行わなければいけない。しかし、位相調整を行うのは非常に難しい。

そのため、位相調整を不要にしたのが２位相ロックインアンプである。図 3-2　は 2 位相ロックインアンプの構成図である。図 3-3 はベクトル演算のベクトル図である。

90°位相差のある参照波を２つ作成し、入力信号とそれぞれ乗算すると極座標上のＸ成分とＹ成分とに検出できる。したがって、得られた信号 X と Y とをベクトル演算すると位相調整なしに入力信号の振幅が求まる。

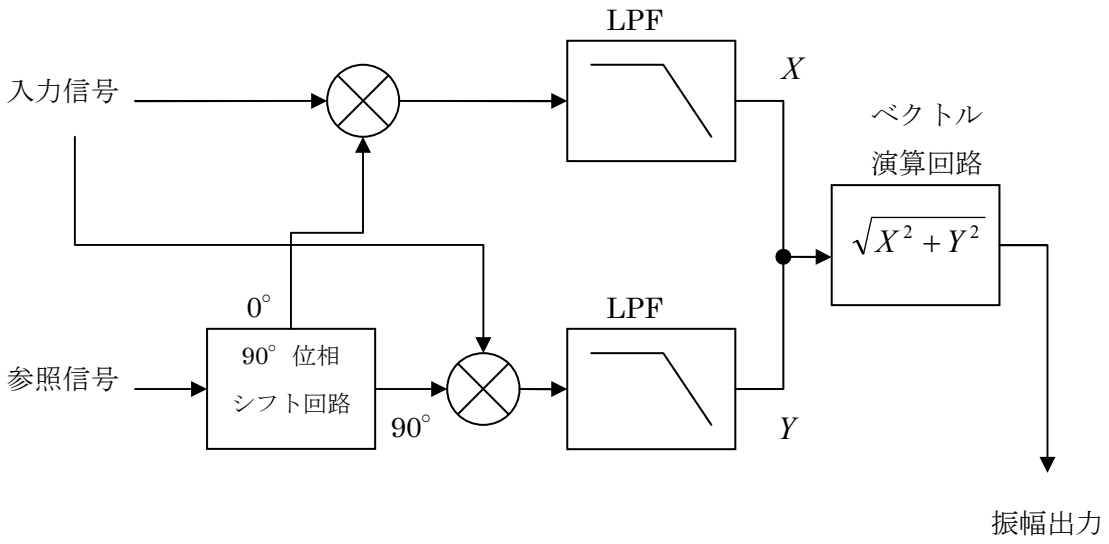


図 3-2　2 位相ロックインアンプ

X 成分

$$\overline{\sin(\omega t)\cdot\{A\sin(\omega t+\phi)+n(t)\}} = \frac{1}{2}A\cdot\cos(\phi) \tag{4-2}$$

Y 成分

$$\overline{\cos(\omega t)\cdot\{A\sin(\omega t+\phi)+n(t)\}} = \frac{1}{2}A\cdot\sin(\phi) \tag{4-3}$$

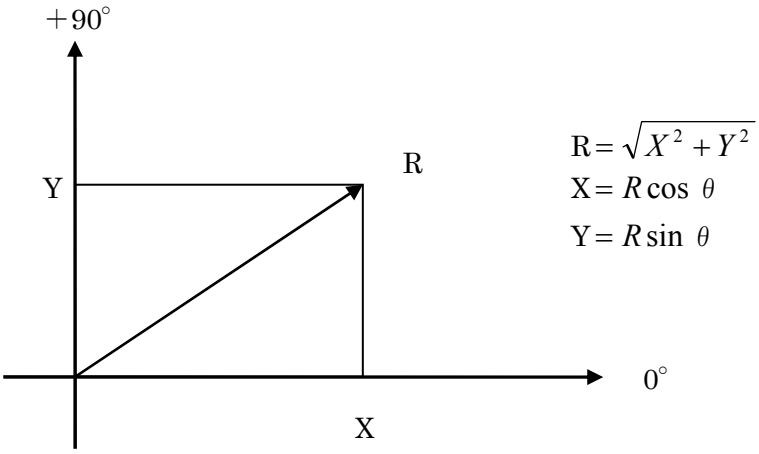


図 3-3　ベクトル演算

4　フィルタについて

4-1 フィルタとは？

本研究で使うフィルタの役割は、不要な周波数の雑音を除去し目的の信号を選択することである。

4-2 フィルタの種類

フィルタには種類が様々な、抽出したい信号によりフィルタは異なる。フィルタを通過域と阻止域の範囲で分類すると次の 4 種類に分けられる。

１．低域通過フィルタ(lowpass filter:LPF)

２．高域通過フィルタ(highpass filter:HPF)

３．帯域通過フィルタ(bandpass filter:BPF)

４．帯域除去フィルタ(band reject filter:BRF)(帯域阻止フィルタ(bandstop filter)と呼ぶ場合もある)

各フィルタが理想フィルタである場合、その振幅に関する周波数特性は図 4-1 のようになる。理想フィルタとは、全帯域が通過域と阻止域のどちらかに分かれるフィルタのことである。通過域と阻止域の境目は遮断周波数(cut-off frequency)と呼ばれる。通過域の振幅特性の値は 1(0dB)で、出力信号の振幅は入力信号の振幅と等しい。阻止域の振幅特性の値は 0(-∞dB)で、出力には信号が現れない。

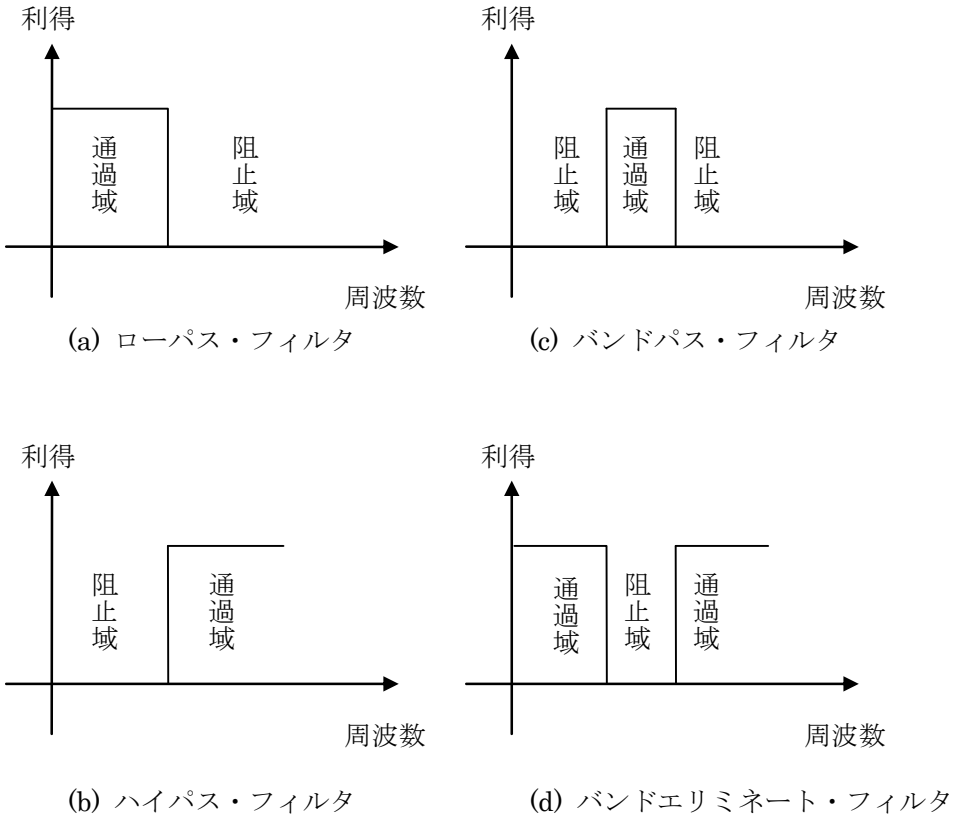
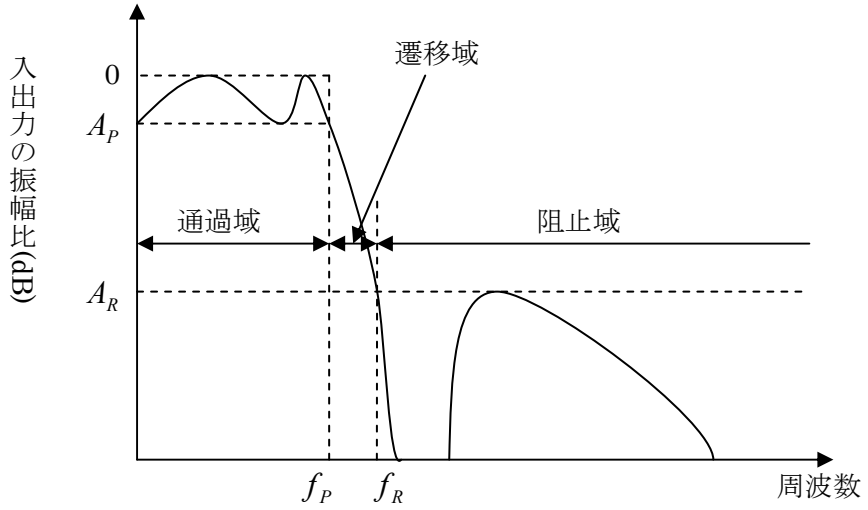


図 4-1　各理想フィルタの周波数特性

しかし、このような特性を持つフィルタは実際に実現できない。

実際の低域通過フィルタの例を図 4-2 に示す。現実のフィルタは、通過域と阻止域の間に、遷移域(transition band)と呼ばれる部分が必ず存在する。



f_P : 通過域端の周波数
f_R : 阻止域端の周波数
A_P : 通過域の振幅特性における偏差の許容値
A_R : 阻止域における最小減衰量

図 4-2　現実に見られる低域通過フィルタの振幅特性の例

フィルタを通過域と阻止域の形状により分類すると低域通過フィルタを例にとると図 4-3 のようになる。通過域と阻止域のそれぞれにリップルが存在する場合としない場合の組み合わせで、4 種類に分かれる。リップルとは、さざなみのように小さく波打つ様子のこと。

１．バタワース(Butterworth)特性

２．チェビシェフ(Chebyshev)特性

３．逆チェビシェフ(inverse Chebyshev)特性

４．連立チェビシェフ(simultaneous Chebyshev)特性

バタワース特性は、通過域および阻止域のいずれにもリップルが存在しない。チェビシェフ特性は、通過域にリップルが存在する。逆チェビシェフ特性は、阻止域にリップルが存在する。連立チェビシェフ特性は、通過域および阻止域のいずれにもリップルが存在する。遮断特性に注目すると、もっとも急峻なのは連立チェビシェフ特性である。

5 ディジタル・フィルタ

5-1 ディジタル・フィルタの特徴

フィルタとは、周波数選択性がある伝送回路のことである。アナログ素子（コイル)、(コンデンサ)、(抵抗）の物理学特性を生かしてフィルタの特性を実現する回路をアナログ・フィルタと呼ぶ。これに対してディジタル・フィルタでは、アナログ信号を一度ディジタル信号に変換(A/D 変換)してディジタル処理を行い、適当な演算を行った後再びアナログ信号に変換(D/A 変換)してフィルタの特性を実現している。

アナログ・フィルタと比べてディジタル・フィルタが優れている点は２つあり、係数メモリと呼ばれるメモリのデータによってフィルタ特性が決定されるため、この係数メモリのデータを変えることによって容易にフィルタ特性を変えることができる点である。また、抵抗などのアナログ素子を用いないので温度変化、経年変化、構成素子の特性のばらつきによる影響をまったく受けず安定した特性を得ることができる点である。

欠点としては、アナログ・フィルタと比べて使用周波数帯域の高周波上限が低いことが挙げられる。

5-2 ディジタル・フィルタの分類

一般的にディジタル･フィルタには FIR(finite impulse response)フィルタと IIR(infinite impulse response)フィルタがある。それぞれの特徴を表 5-1 に示す。

この 2 つの名前の由来は、インパルス応答の継続時間による。インパルス応答の継続が有限という意味は、入力に一瞬だけ信号を与えた場合に、有限の時間が経過した後に出力が 0 になるということである。差分方程式で両者を比較すると、式の右辺に、過去に計算された出力信号が存在するかしないかが両者の違いである。M は有限の値である。つまり、IIR フィルタでは、現在の出力信号を計算するために、過去に計算された。出力結果も使っていることになる。これは、一種のフィードバックと考えることができる。

伝達関数で比較すると FIR フィルタ Z^{-1} に関する多項式になっているのに対して IIR フィルタは Z^{-1} に関する有理関数になっている。したがって、$Z=0$ の点を除くと FIR フィルタは極をもたず IIR フィルタは極をもつことになる。このことは表の項目にある安定性と密接な関係がある。

過去に計算した値を再び使うか使わないかということで非再帰形と再帰形に分類される。IIR フィルタは過去に計算された値を使うので必ず再帰形になる。一方、FIR フィルタは非再帰形を利用する。FIR フィルタは通常非再帰形で構成するので、その場合は必ず安定になる。

完全に正確な直線位相特性が実現できるのは FIR フィルタだけで IIR フィルタでは近似は可能なものの完全に正確な直線位相特性が実現することはできない。直線位相というのは、フィルタによる位相遅延が周波数に比例しているということで、言い換えれば、入力に対して一定の時間遅延（群遅延と呼びます）が与えられるだけで、周波数の違いによる相対的な位相ずれが生じないということである。

演算誤差の影響で比較すると IIR フィルタは FIR フィルタに比べて影響が大きく現れる場合がある。その大きな原因は、IIR フィルタの場合フィードバックがあるので過去の演算誤差の影響が蓄積されることにある。

	FIR フィルタ	IIR フィルタ
インパルス応答の継続時間	有限	無限
差分方程式	$y[n]=\sum_{m=0}^{M} h_m y[n-m]$	$y[n]=\sum_{m=1}^{M} a_m y[n-m]+\sum_{k=0}^{K} b_k x[n-k]$
伝達関数	$H(z)=\sum_{m=0}^{M} h_m z^{-m}$	$H(z)=\frac{\sum_{k=0}^{K} b_k z^{-k}}{1-\sum_{m=1}^{M} a_m z^{-m}}$
構成方法	非再帰形	再帰形
安定性	常に安定	伝達関数の極がz平面の単位円内に存在するときは安定
直線位相特性の実現性	完全に正確なものが可能	不可能
演算誤差の影響	余り大きく現れない	大きく現れる場合がある
急峻な遮断特性の実現	高次のフィルタが必要	比較的低次のフィルタで十分

表 5-1　FIR フィルタと IIR フィルタの比較

5-3　IIR フィルタによる参照波の作成

入力信号に乗算するための参照波を IIR フィルタにより作成した。次に示すのは、IIR フィルタによる参照波の理論式の導出方法である。

5-3-1Sin 波

ある離散時間システムで、入力信号 x[n]の z 変換を $X(z)$、出力信号 y[n]の z 変換を $Y(z)$ とすると、その離散時間システムの伝達関数 H(z)は、次式で定義される。

$$H(z)=\frac{Y(z)}{X(z)} \tag{5.1}$$

ただし、n<0 に対して x[n]=0,y[n]=0 とする。

(5.1)を変形すると

$$Y(z) = H(z) \cdot X(Z) \tag{5.2}$$

と表すことができる。

したがって、伝達関数 *H(z)*が sin の *z* 変換になるようなシステムを作成し、このシステムに X(z)=1 である信号を入力すれば出力信号の z 変換は sin の z 変換に等しくなる。

X(z)= 1 である信号を x[n]とすると、それは単位インパルス信号 δ [n]で、次のように表すことができる。

$$x[n] = \delta[n] = \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases} \tag{5.3}$$

一方角周波数 ω =(2 π F)の正弦波を標本化間隔 T で標本化した離散的信号を sin[n ω T]とする。この離散的正弦波の *z* 変換は次のようになる。

$$\frac{(\sin \omega T) z^{-1}}{1 - 2(\cos \omega T) z^{-1} + z^{-2}} \tag{5.4}$$

したがって、正弦波発生システムの入出力関係は次のように表すことができる。

$$Y(z) = \frac{(\sin \omega T) z^{-1}}{1 - 2(\cos \omega T) z^{-1} + z^{-2}} \cdot X(z) \tag{5.5}$$

上式は次のように書き換えができる。

$$Y(z) - 2(\cos \omega T) Y(z) z^{-1} + Y(z) z^{-2} = (\sin \omega T) X(z) z^{-1} \tag{5.6}$$

これを考慮すると上式は次のような時間領域の表現に変換できる。

$$y[n] - 2(\cos \omega T) y[n-1] + y[n-2] = (\sin \omega T) x[n-1] \tag{5.7}$$

上式を整理すると sin 波発生システムの入出力の関係を表す差分方程を導出できる。

$$y[n] = 2(\cos \omega T) y[n-1] - y[n-2] + (\sin \omega T) x[n-1] \tag{5.8}$$

5-3-2Cos 波

角周波数ω=(2πF)の正弦波を標本化間隔 T で標本化した離散的信号を cos[nωT]とする。

この離散的正弦波の z 変換は次のようになる。

$$\frac{1-(\cos\omega T)z^{-1}}{1-2(\cos\omega T)z^{-1}+z^{-2}} \tag{5.9}$$

したがって、正弦波発生システムの入出力関係は次のように表すことができる。

$$Y(z)=\frac{1-(\cos\omega T)z^{-1}}{1-2(\cos\omega T)z^{-1}+z^{-2}}\cdot X(z) \tag{5.10}$$

上式は次のように表すことができる。

$$Y(z)-2(\cos\omega T)Y(z)z^{-1}+Y(z)z^{-2}=X(z)-(\cos\omega T)X(z)z^{-1} \tag{5.11}$$

これを考慮すると

$$y[n]-2(\cos\omega T)y[n-1]+y[n-2]=x[n]-(\cos\omega T)x[n-1] \tag{5.12}$$

上式を整理すると cos 波発生システムの入出力の関係を表す差分方程を導出できる。

$$y[n]=2(\cos\omega T)y[n-1]-y[n-2]+x[n]-(\cos\omega T)x[n-1] \tag{5.13}$$

5-4　FIR ローパスフィルタによる移動平均

移動平均のブロック図を図 5-1 に示した。図 5-1 より非再帰形だとわかる。つまり、この移動平均は FIR フィルタである。

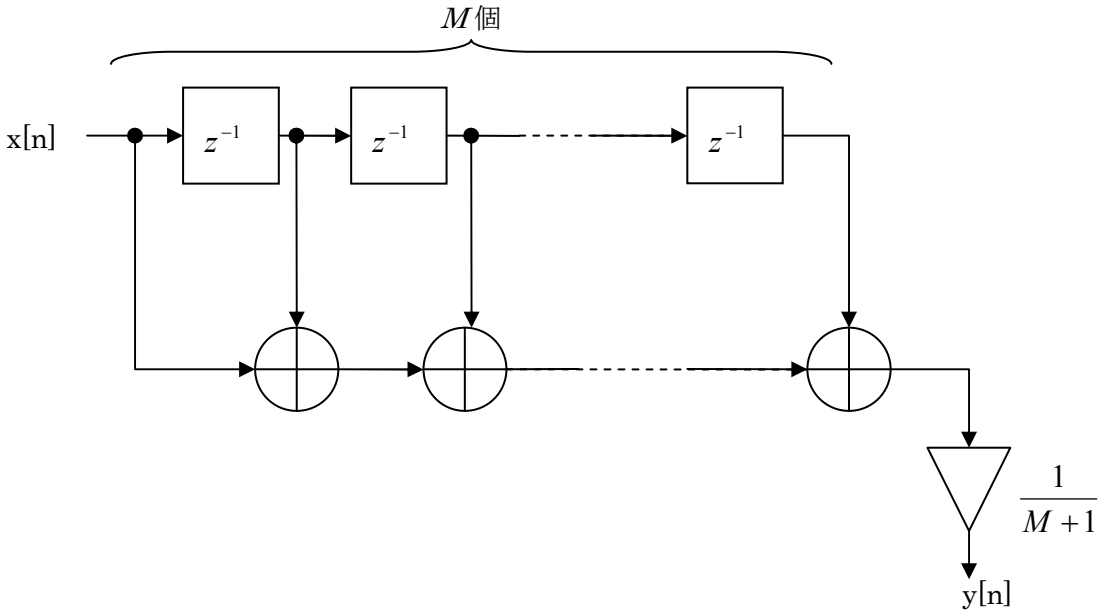


図 5-1　　移動平均のブロック図

このブロック図の M 点移動平均のプログラムを作成した。そして、この M 点移動平均プログラムの周波数特性を測定し、理論値と比較を行った。

測定方法は、波形発振器の正弦波信号を DSP ボードに入力し、DSP 内部で M 点の移動平均の計算が行われ、正弦波が出力される。DSK から出力された信号の振幅をオシロスコープで測定した。また、波形発振器はオシロスコープとも接続されており、入力信号の振幅も同時に測定した。A/D 変換器が扱える最高周波数は 3.8kHz である。波形発振器の周波数は 10Hz から始め、徐々に周波数を上げていき、4000Hz まで周波数を上げた。測定には、1、5、10 点の移動平均のプログラムを使用した。

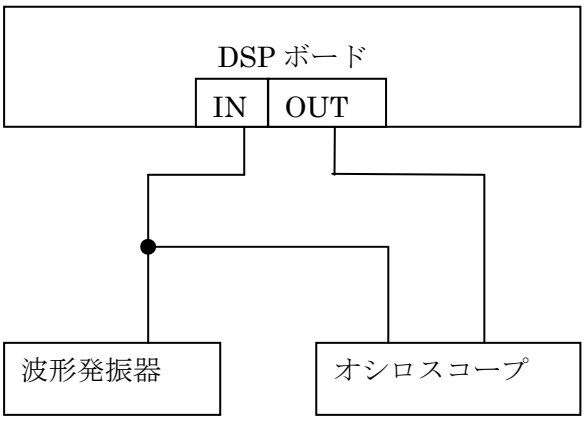


図 5-2　実験構成

測定した出力信号の振幅と入力信号の振幅の比で周波数特性を求めた。

理論値の導出方法は次の通りである。伝達関数は入力信号の z 変換に対する出力信号の比で定義される。つまり、H(z)を伝達関数、x(z)を入力信号の z 変換、Y(z)を出力信号の z 変換とすると式(1)のように表すことができる。

$$H(z)=\frac{Y(z)}{X(z)} \tag{5.14}$$

式(5.14)を変形する。

$$Y(z)=H(z)\cdot X(Z) \tag{5.15}$$

平均するデータ数が N の場合の重み付き移動平均の差分方程式は式(5.16)で表される。

$$y[n]=\frac{1}{M+1}\sum_{k=0}^{N-1}x[n-k] \tag{5.16}$$

この差分方程式から伝達関数を求めることができる。最初に、x[n]の x 変換を X(z)、y[n]の z 変換を Y(z)と仮定する。次に、z 変換の性質を使うと式(5.3)の両辺の z 変換は式(1)で表すことができる。

$$Y(z)=\frac{1}{M+1}X(z)\sum_{m=0}^{M}Z^{-m} \tag{5.17}$$

したがって、式(3.3)の差分方程式に対応する伝達関数は次の式で表せる。

$$H(z)=\frac{1}{M+1}\sum_{m=0}^{M}Z^{-m} \tag{5.18}$$

伝達関数の変数zを次のように置き換えると周波数特性を表す関数になる。

$$z=e^{j\omega T} \tag{5.19}$$

この式でωは角周波数、Tは標本化間隔を表しています。

そこで、周波数特性を$H(\omega)$とすると、式(3.3)に対応する周波数特性は次のようになる。

$$H(e^{jwT})=\frac{1}{M+1}\sum_{m=0}^{M}e^{-j\omega T} \tag{5.20}$$

オイラーの公式より　$e^{-j\omega T}=\cos m\omega T-j\sin nm\omega T$ (5.21)

よって式(5.7)は次のように表される。

$$=\frac{1}{M+1}\cdot\frac{1-\exp(-j(M+1)\omega t)}{1-\exp(-j\omega T)} \tag{5.22}$$

この式を変形して周波数特性の式は

$$\left|H(\omega)\right|=\frac{1}{M+1}\cdot\left|\frac{\sin\frac{(M+1)\omega T}{2}}{\sin\frac{\omega T}{2}}\right| \tag{5.23}$$

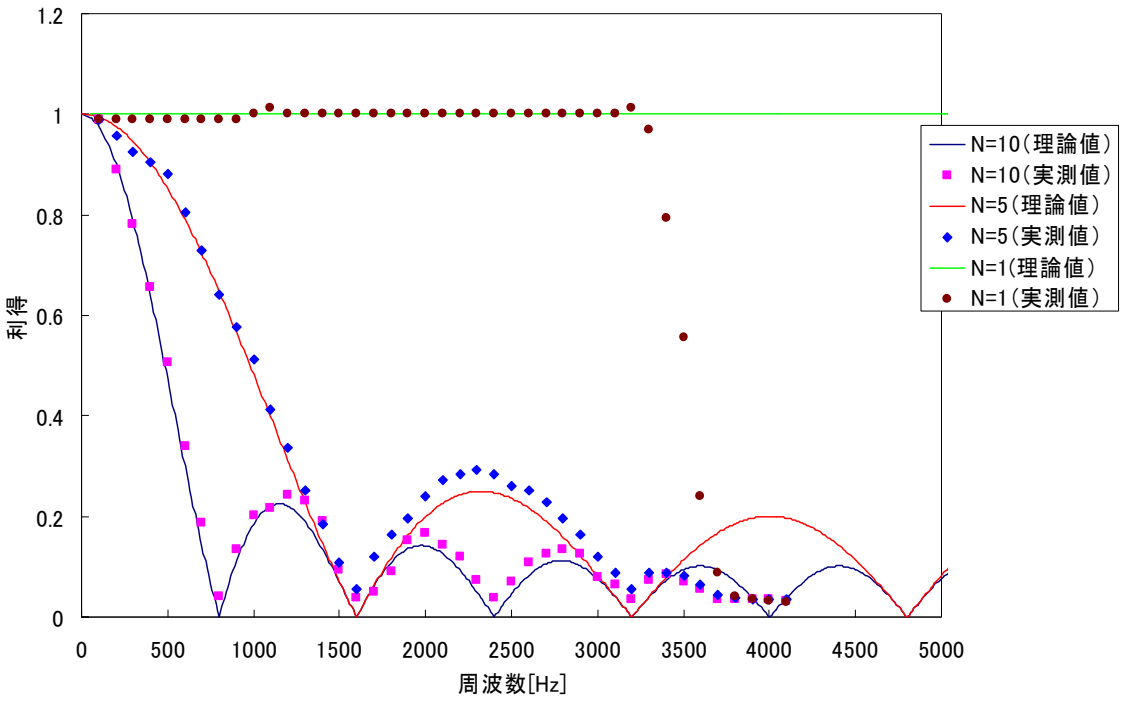


図 5-3　各移動平均の測定結果

グラフより、移動平均する点の数を増やすことでより急峻な遮断周波数特性が得られることが分かった。理論値から遮断周波数を計算すると、10 点の移動平均の場合は 355.9Hz、5 点の場合は 720.3Hz、1 点の場合は発散した。10 点と 5 点の遮断周波数の関係から点数を倍に増やすと遮断周波数は 1/2 倍になり、反比例していることが分かった。次に遮断周波数より時定数を求めた。10 点の移動平均の場合は 902.2μs、5 点の場合は 447.0μs、10 点と 5 点の時定数の関係から点数を倍に増やすと時定数は 2 倍になり、比例していることが分かった。グラフで実測値が 3500Hz 辺りで利得が下がっているのは内蔵されているローパスフィルタによるものである。

5-4-1　移動平均と実行時間

移動平均の帯域幅を大きくすると遮断特性は大きくなるが、その分移動平均にかかる時間は大きくなる。しかし、DSK の A/D 変換器のサンプリング周波数が 8000Hz であるため 125μs ごとでしか信号を読み取ることができない。そのため、125μs 以内にプログラムを実行し続けなければならないため 125μs 以内の移動平均で行わなければならない。そこで、移動平均にかかる時間を測定することにした。

図 4-4 に、プログラムの実行時間と移動平均の点数の関係を示す。図 4-4 より実行時間と移動平均の点数は比例していることが分かる。この比例の関係より移動平均の点数を変えたとき移動平均にかかる実行時間が分かるようになった。

図 4-4 より移動平均の点数は 125μs 以内の 200 が適当と思われる。

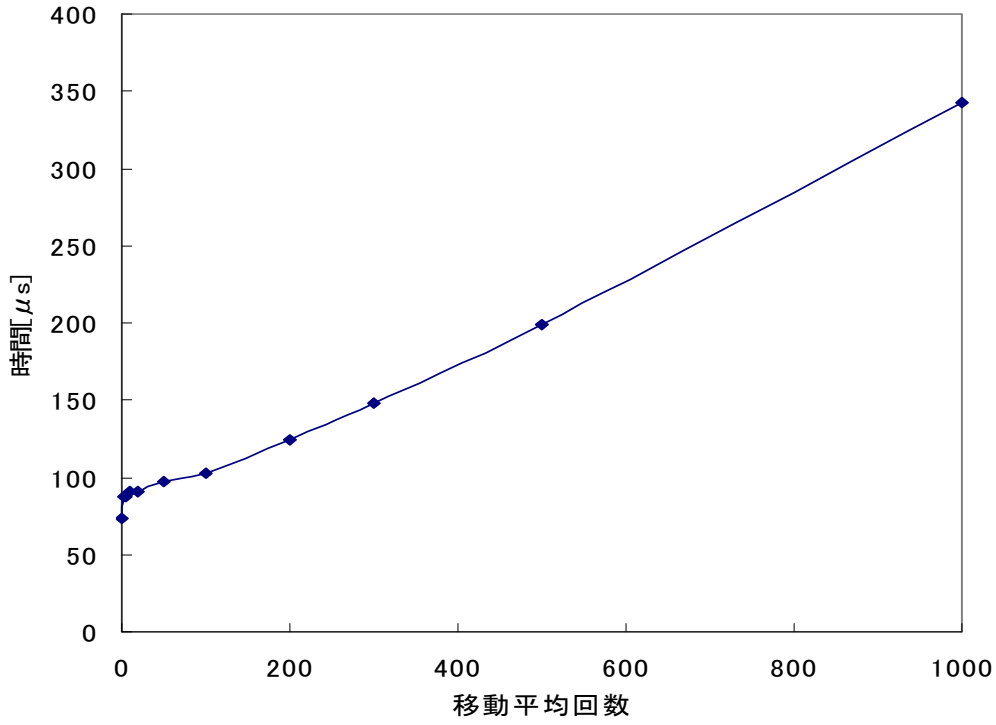


図 5-4　移動平均の点数と実行時間の関係

6 白色雑音発生器

6-1 ホワイトノイズ

ホワイトノイズ（White noise）とは、不規則に上下に振動する波のこと。また、白色雑音とも呼ぶ。音声などに混入するノイズの中で、全ての周波数帯域においてエネルギーが均一に混入した雑音のことである。

音の周波数をグラフで表現すると、ホワイトノイズでは雑音が均一に混入しているため平坦に近いグラフとなる。このグラフを光の波長に置き換えると、白色を示す周波数の特徴に一致することから、ホワイトノイズと呼ばれている。

なお、ホワイトノイズに対して、周波数の高い音ほどエネルギーが弱くなるようなノイズはピンクノイズと呼ばれる。これは周波数を光に置き換えた場合に、波長の長い赤い光ほど赤く見えることに由来している。

6-2 雑音発生のしくみ

ツェナーダイオードに電流制限抵抗を直列に接続してツェナー接続してツェナーダイオード電圧以上の電源電圧を加える。このときツェナーダイオード両端の電圧は温度によって多少変動はするが、ほぼツェナー電圧に等しくなる。ツェナーダイオードにツェナー電圧以上の電圧が加わるとツェナーダイオードが ON して電流が流れ出す。電流が流れると抵抗によって電圧が降下する。ツェナー電圧がツェナー電圧以下まで降下すると OFF する。すると電流が遮断されるので再び電圧が上昇する。ツェナー電圧を超えると、また ON になるという動作を高速に繰り返す。ON 電圧はスイッチングごとに微妙にばらつくので発生する雑音のスペクトルはとても幅広い帯域に分布する。Vout 端子で観測されるのはこのスイッチング電圧が平均化された電圧、つまりツェナー電圧になる。

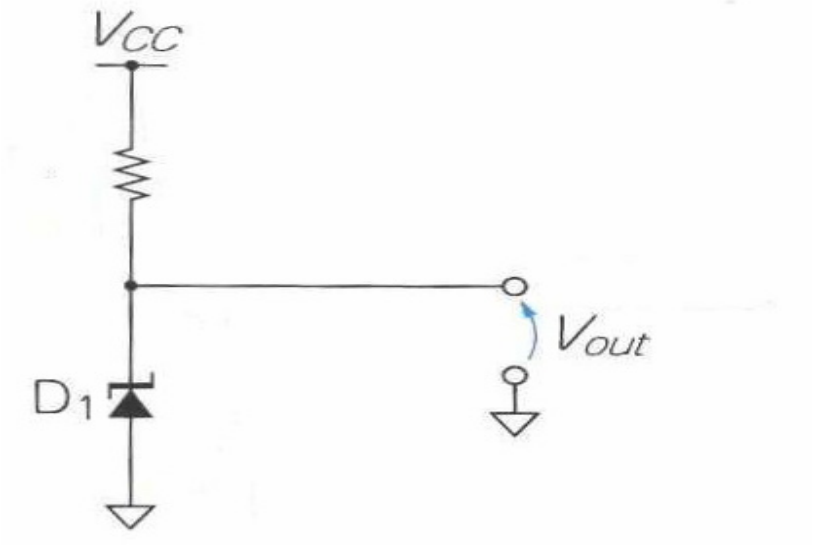


図 6-1　白色雑音発生源

6-3　白色雑音発生器

ツェナーダイオードを用いてホワイトノイズジェネレータの製作を行った。今回は、ロックインアンプの雑音除去動作の確認を行うために製作を行った。図 4-4 にノイズジェネレータの回路図を示す。

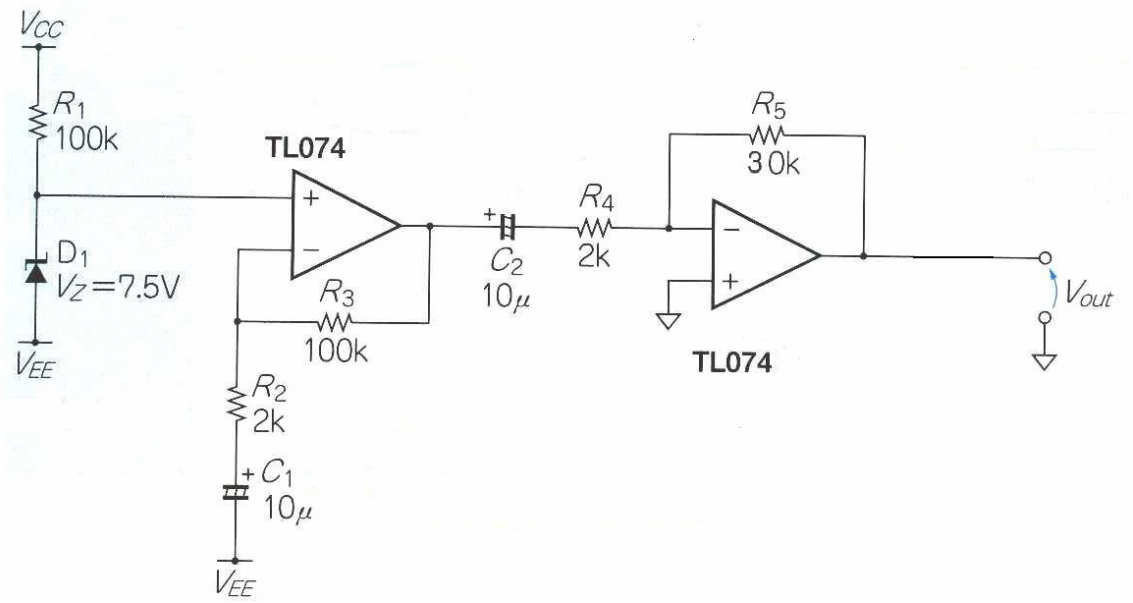


図 6-2　雑音発生器回路図

7 ロックインアンプの動作特性

7-1　ディジタル値と振幅の関係

ロックインアンプの特性を評価するためプログラムで作成した参照波を DSK から出し DSK に戻すということを行った。この際、正弦波が DSK から出力コード分岐させ一つを DSK にもう一つをオシロスコープにつないだ。図 7-1 は実験の構成図である。オシロスコープにつないだのはDSKから出力されているか確認するためと正弦波の振幅を調べるためである。オシロスコープに表示される振幅と正弦波のプログラム上の振幅の関係を図 7-2 に示す。

また、振幅と出力結果Rの関係を図 7-3 に示す。この図 7-3 より出力結果Rが表示されたとき入力信号の振幅がわかるようになった。縦軸の R とは DSP ボードから Excel にデータが送られベクトル演算を行ったディジタル値である。

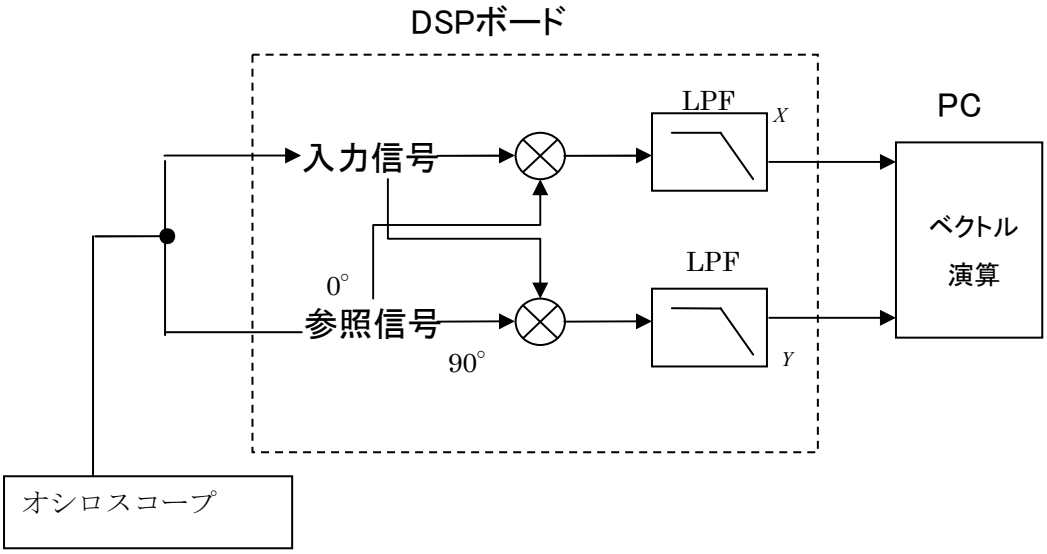


図 7-1　実験構成

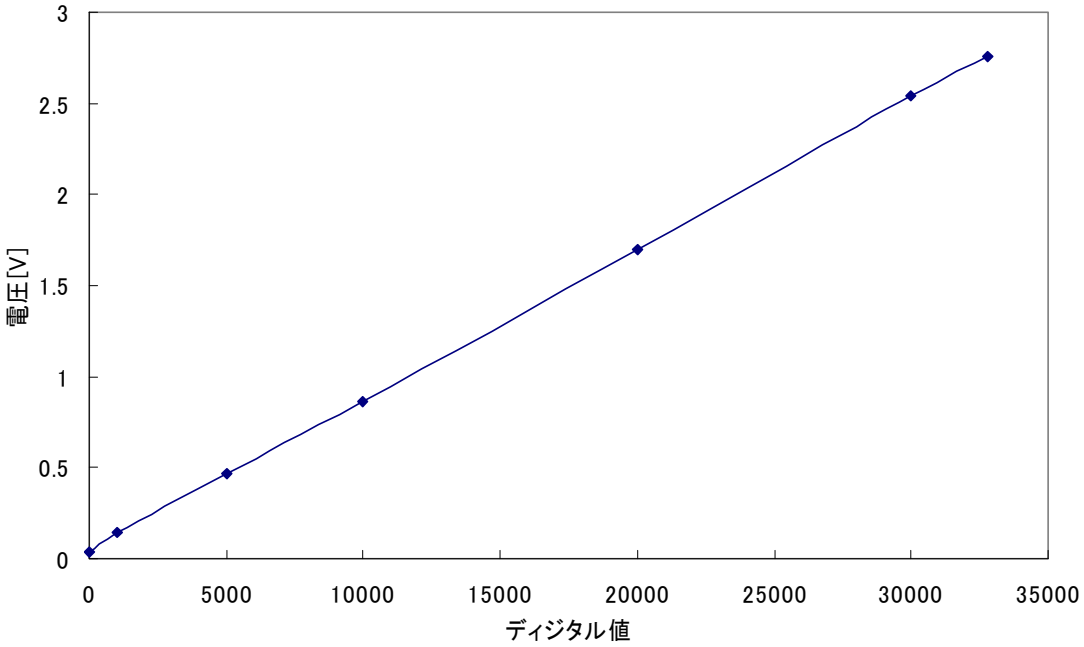


図 7-2　入力電圧とプログラムの振幅の関係

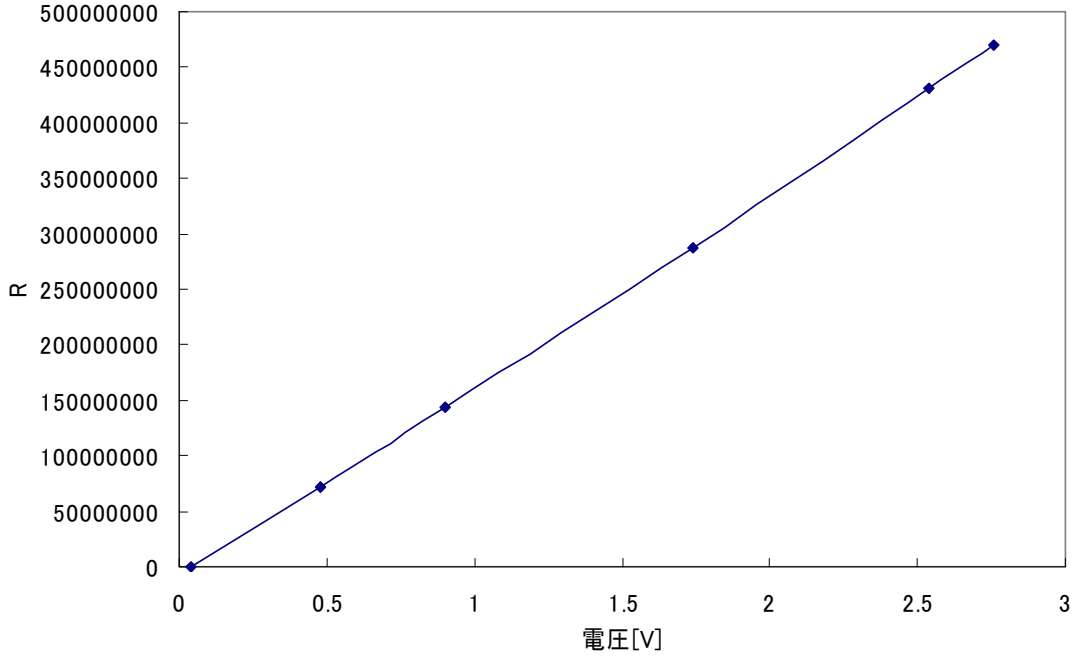


図 7-3　Ｒと入力電圧の関係

7-2 ロックインアンプの周波数特性

ロックインアンプの実験の構成図を図 7-4 に示した。白色雑音発生器からの白色雑音と波形発振器からの正弦波を加算したものを入力信号とした。FIR ローパスフィルタの移動平均の帯域を変えたときのロックインアンプの出力を図 7-5 に示した。波形発振器からの信号の周波数は 400Hz で振幅は 1.06V である。移動平均の点数を大きくすると遮断周波数が小さくなるが測定時間が長くなってしまう。

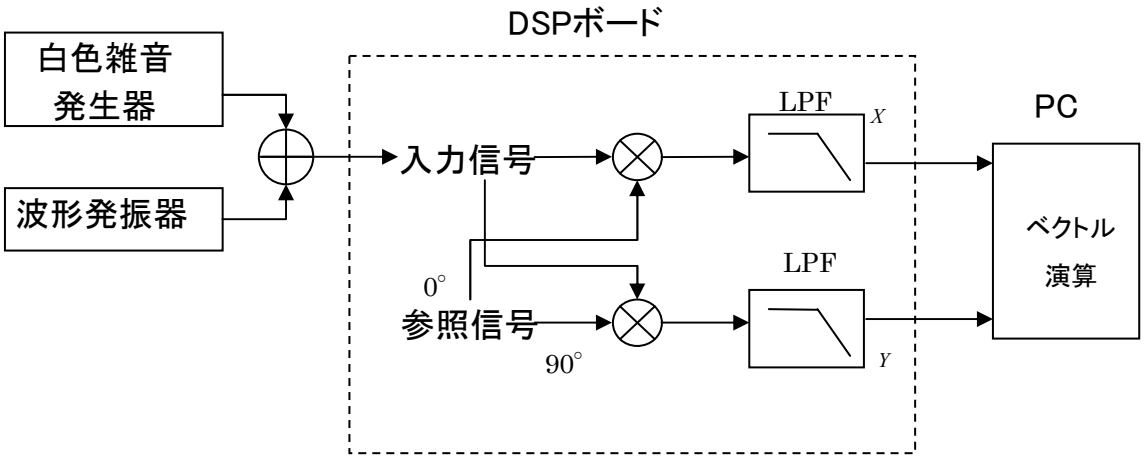
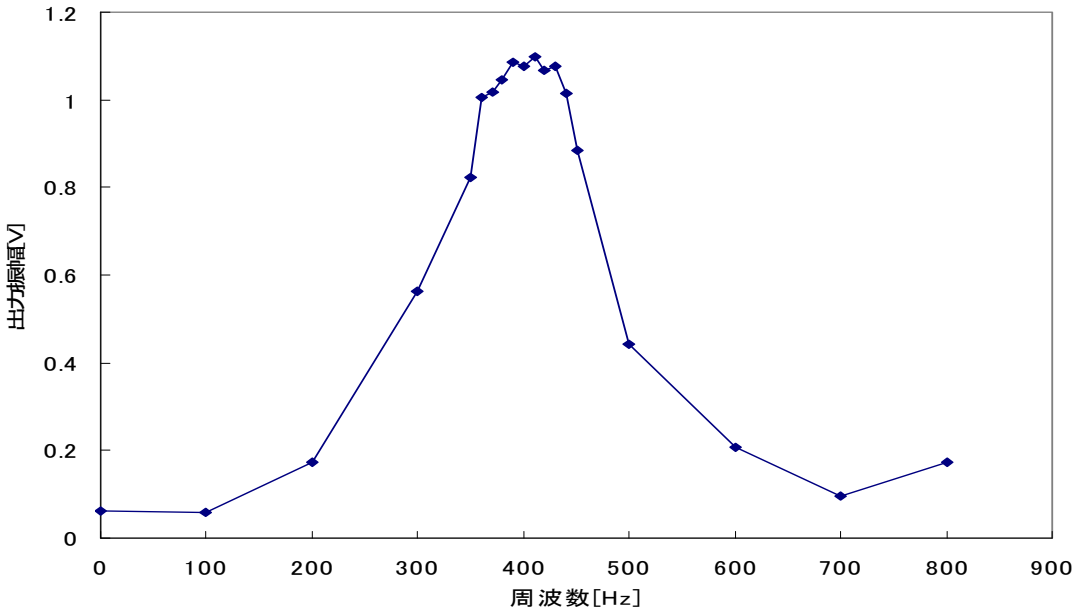
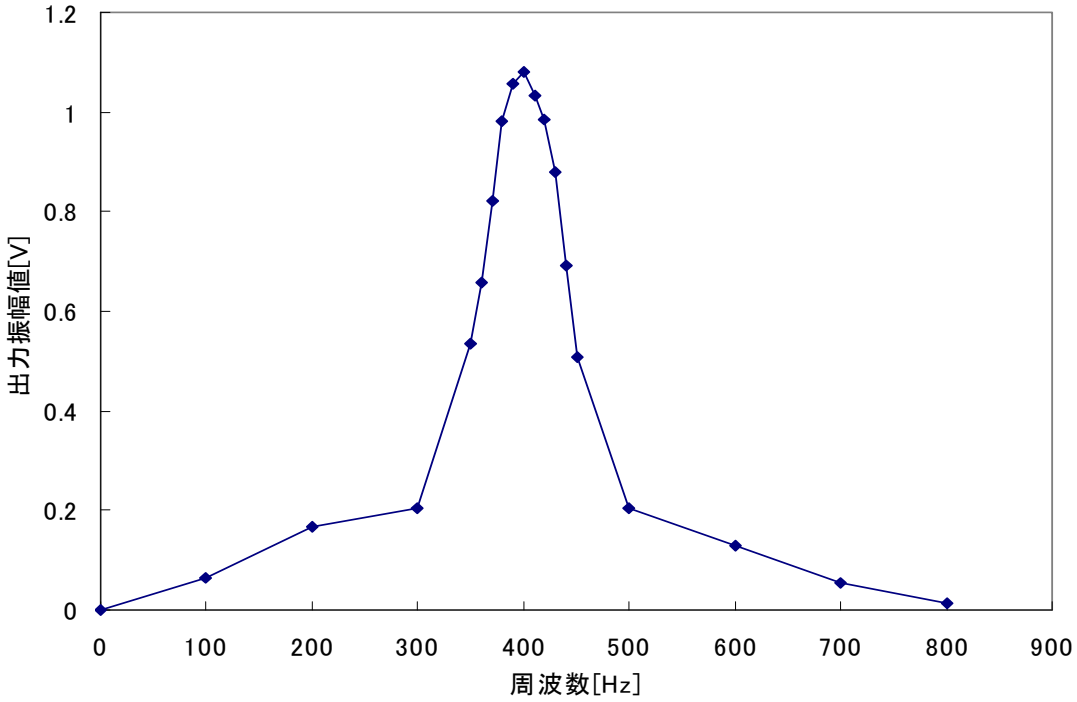


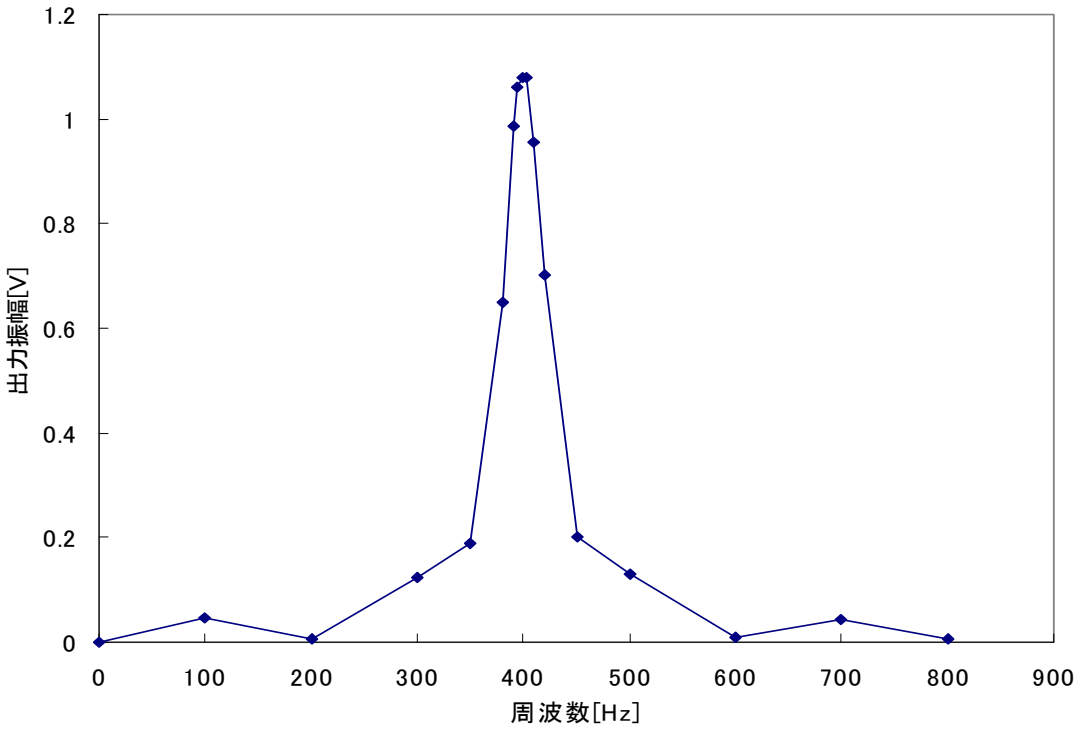
図 6-6　ロックインアンプ実験構成



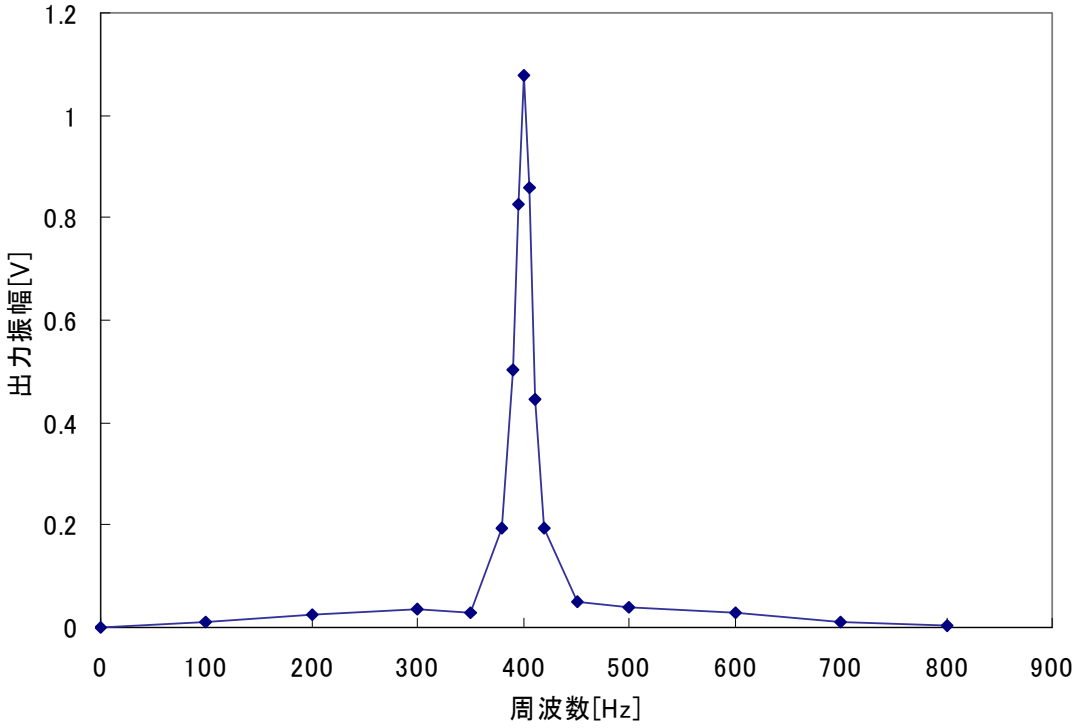
(a)移動平均 50



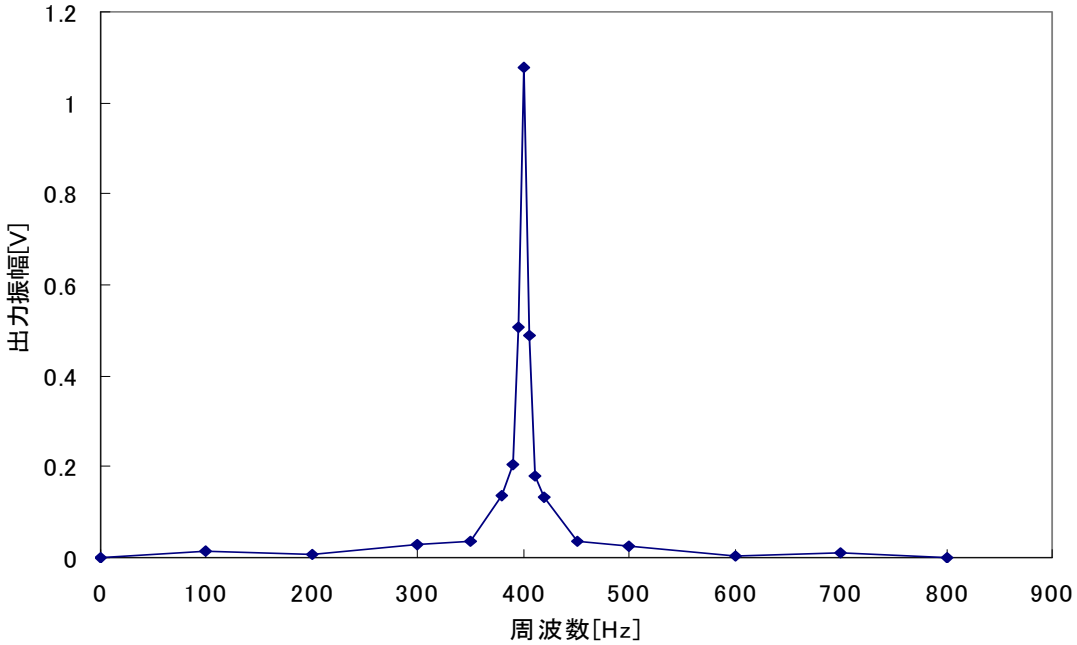
(b)移動平均 100



(c)移動平均 200



(d)移動平均 500



(e)移動平均 1000

図 7-5　ロックインアンプの周波数特性

7-3 フィルタの遮断周波数特性

図 7-5 より各移動平均の時の遮断周波数を測定しその時定数との関係を図 7-6 に示した。理論値はアナログ方式の RC ローパスフィルタの理論式から求めた。実測値がほぼ理論値通りになっているのが分かる。

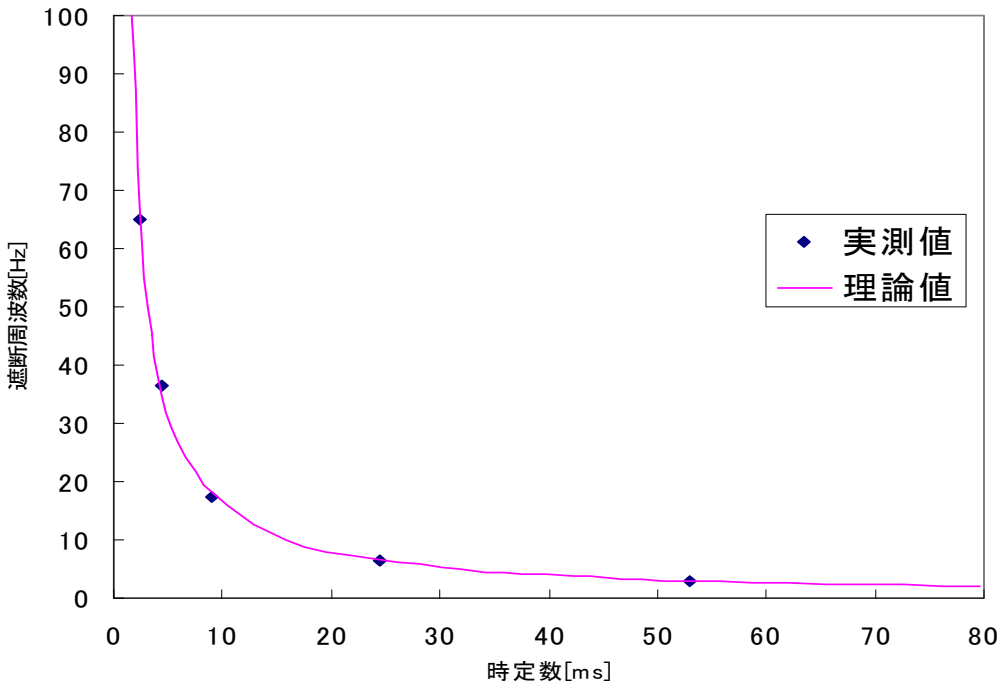


図 6-8　フィルタの遮断周波数特性

7-4 ロックインアンプ入出力特性

　ロックインアンプの精度を検証するためロックインアンプの入出力特性を測定した。図 6-9 に入出力特性図を示す。今回測定したデータはほぼ理論値を取っていることがわかる。

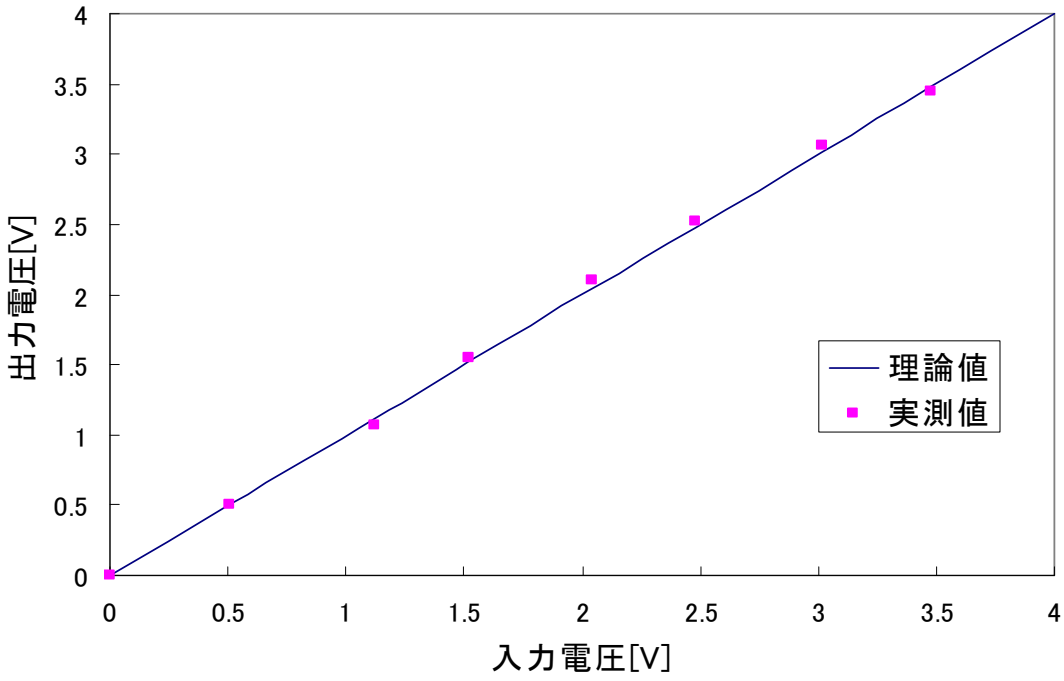


図 6-9　ロックインアンプ入出力特性

7 まとめ

7-1 まとめ

DSP による FIR フィルタを使ったロックインアンプを作成した。移動平均に伴って遮断周波数が低くなり、動作の確認が行われた。今後の課題として、ロックインアンプの試作に留まったがダイナミックリザーブ（雑音の大きさに対してどのくらいまで目的の信号を計れる）の測定を行い完成させたい。現在は C6711DSP ボードを使用しているが C6713DSP ボードに移行する。

7-2 謝辞

本研究を行うにあたり、懇切親身且つ具体的なご指導と御支援さらに本論文作成について多くの貴重な御意見を賜りました本校電子制御工学科由井四海教員に心から感謝いたします。

また研究実験および論文作成に当って貴重な御意見を多数下さいました制御情報システム工学専攻科 2 年中谷健一氏、制御情報システム工学専攻科 1 年鹿熊航太氏にも深く感謝の意を表したいと思います。

ここに心より感謝の意を表明することで、皆様方への謝辞とさせて頂きます。

参考文献

三上直樹：「C 言語によるディジタル信号処理入門」,CQ 出版

三上直樹：「はじめて学ぶディジタル・フィルタと高速フーリエ変換」, CQ 出版

三上直樹：「ディジタル信号処理と DSP」, CQ 出版

瀬谷啓介：「DSP C プログラミング入門」,技術評論社

浅川毅・北川宗行：「絵ときディジタル信号処理入門」,オーム社

遠坂俊昭：「計測のためのフィルタ回路設計」, CQ 出版

山口晶大：「はじめての DSP 活用大全」, CQ 出版

呉尚謙：「近赤外半導体レーザによるガス中微量水分の高感度検出法」

付録

DSK ボードの接続と CCS(Code Composer Studio)の起動

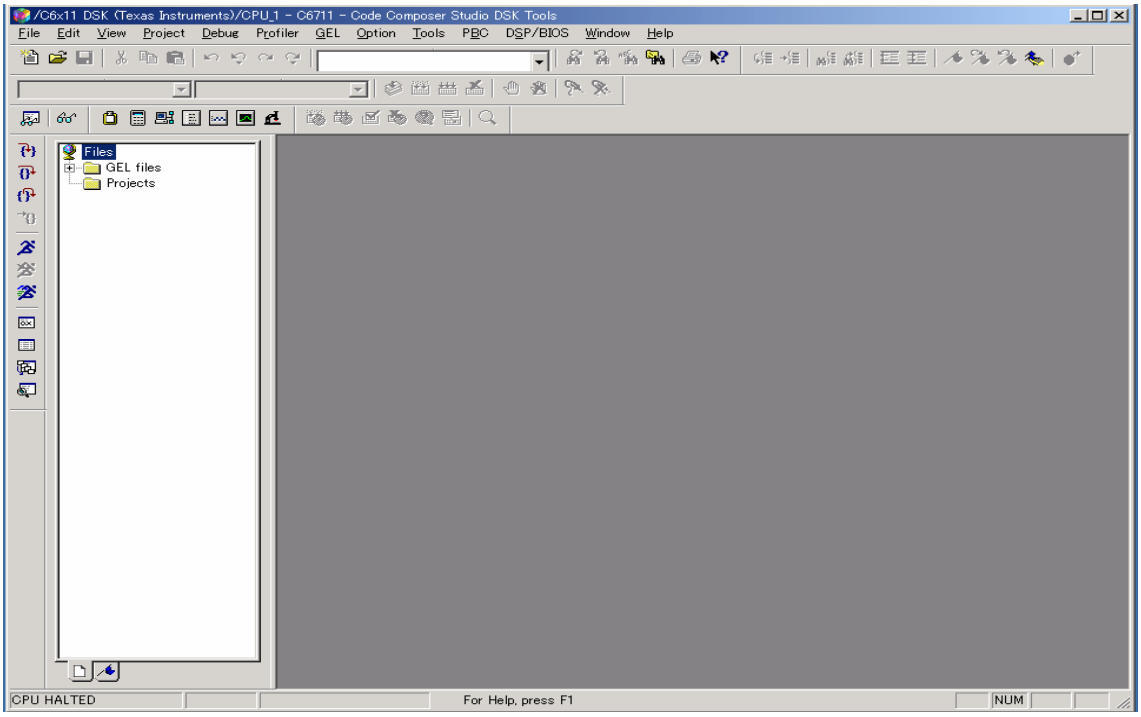
［開始手順］

１．DSK ボードの電源を入れる。

DSK 用電源ケーブルのプラグをコンセントに差し込んだ後、電源コネクタを DSK ボード上の電源コネクタに接続する。電源投入と同時に Power On Self Test(POST)が実行され、LED 0-2 が点滅する。POST が完了し、DSK ボードが正常に機能していることが確認されると、すべての LED が同時に点灯した状態になる。

２．CCS(Code Composer Studio)を起動する。

ホスト PC のデスクトップにあるアイコン C6711DSKCCS をダブルクリックする。



起動直後の画面

［終了手順］

1．CCS の終了

File>Exit とクリックし、CCS を終了する。

2．DSK ボードの電源プラグをはずす。

［取り扱い上の注意事項］

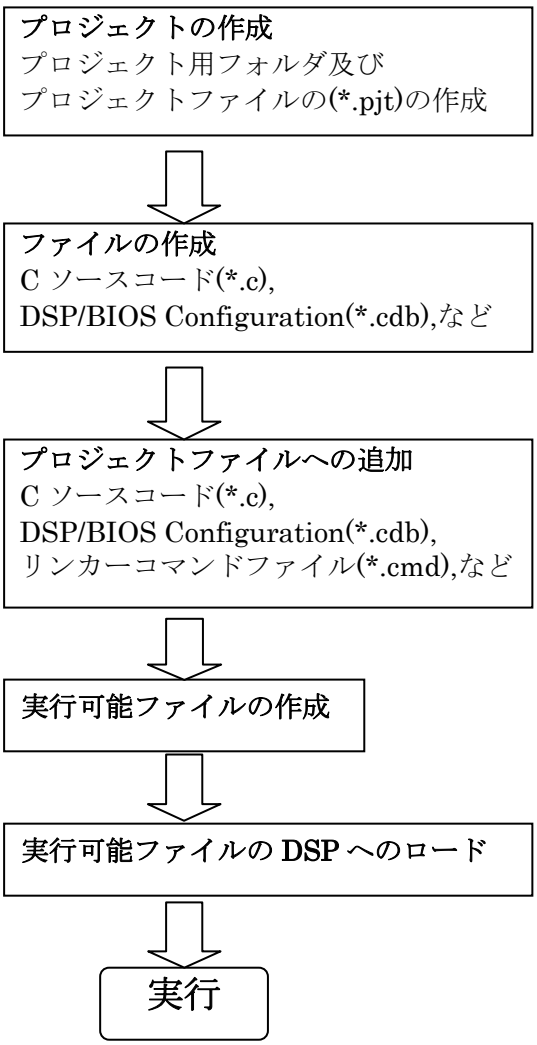
1．DSK ボード上の回路にプラグ等の金属部分を接触させないように注意する。

2．実験前に GND に触れるなど、静電気の除去に留意する。

3．接触不良防止のため、妄りにコネクタ部には触らない。

4．物理的な破損を起こさないよう、電源プラグ、オーディオプラグ等の抜き差しは慎重に行う。

［プログラムの開発］



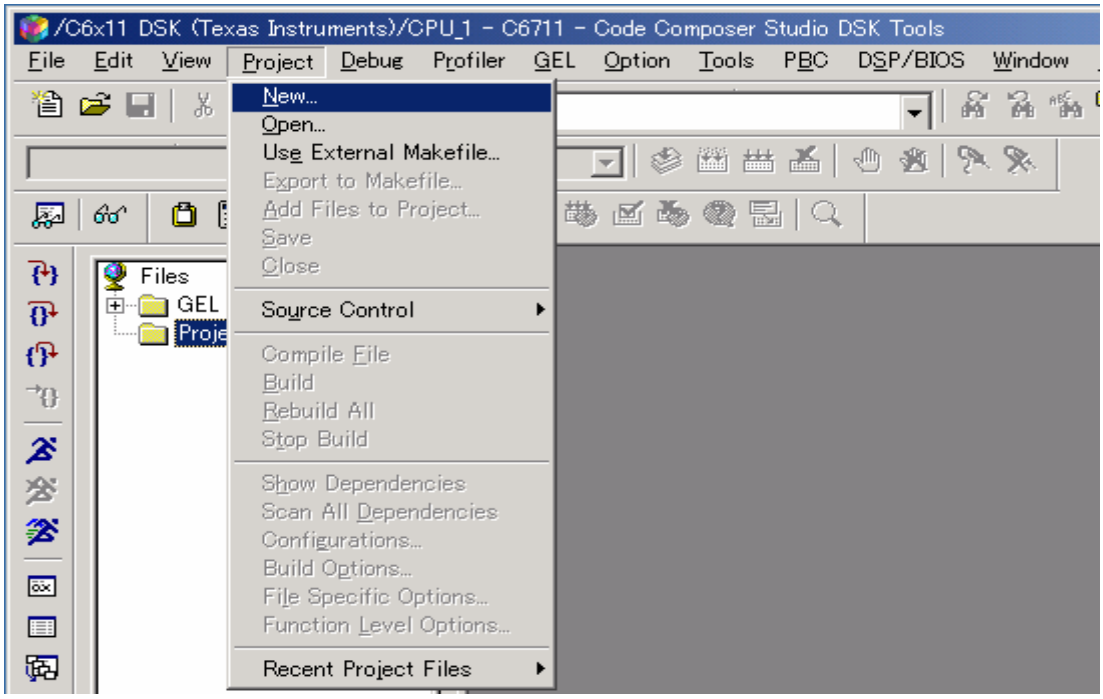
プログラム作成の流れ図

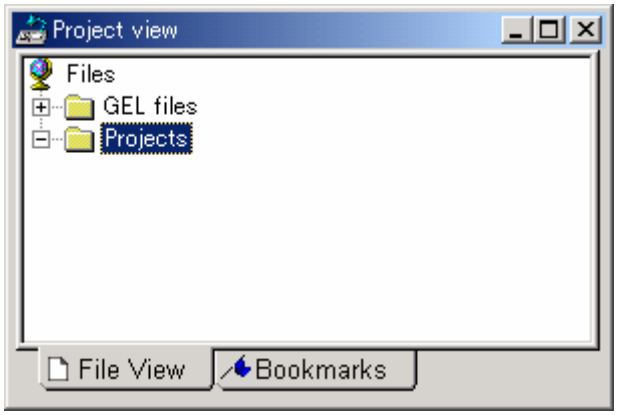
１．新しいプロジェクトの作成

(1) Project から New を選択する。

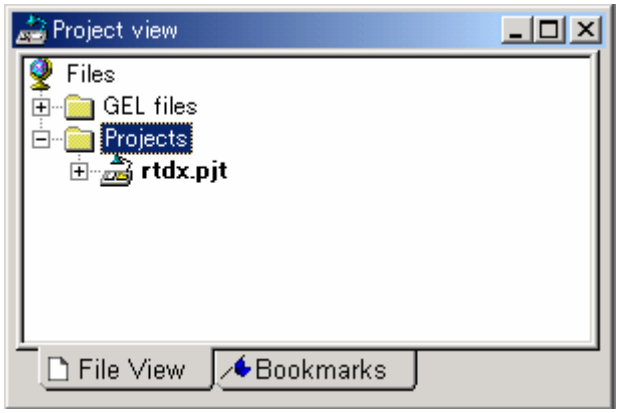
(2) Save New Project As のウィンドウで、作成した作業用のフォルダを選択する。ファイル名を rtdx として保存します。

CCS は rtdx というプロジェクトファイルを生成します。このファイルには、作成したプロジェクトで使用する様々なファイルのプロジェクトに関する設定及びリファレンスが保存されます。



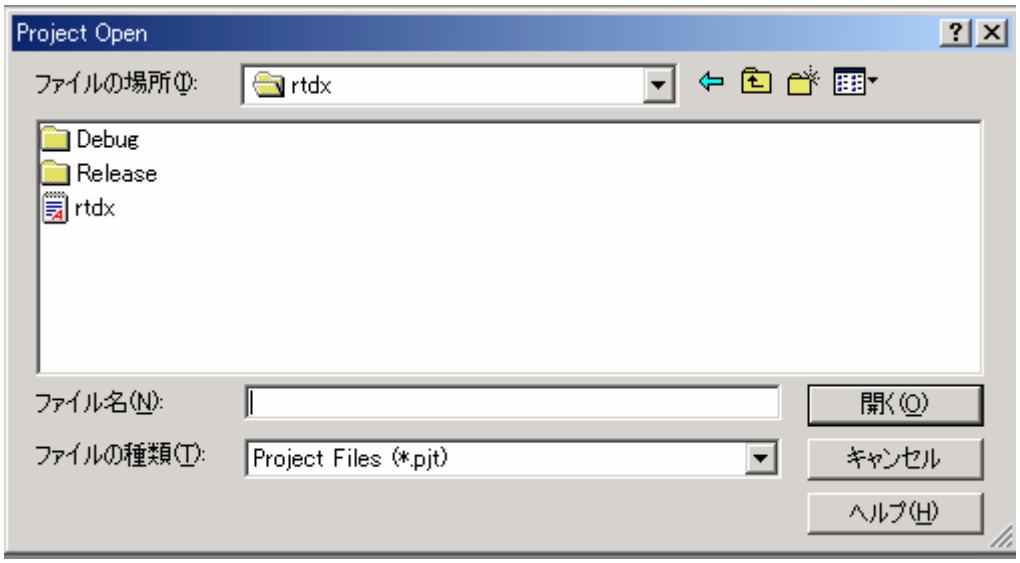


プロジェクトを作成する前



新規作成のプロジェクトに”rtdx”という名前をつけた状態

すでに作られているプロジェクトを再び開く場合はメニューバーで Project→Open を選択する。すると、”Project Open”ダイアログが現れるのでプロジェクト名を選択し、[開く(O)]ボタンをクリックするとプロジェクトが開く。すでにファイルが作成されているプロジェクトを開く場合は２．ファイル作成の手順は必要がない。ここでは、すでに作成されている rtdx のフォルダに入っている rtdx を開く。



２．ファイルの作成

１．新しいプロジェクトの作成で

CCS でソフトウェアを開発する際には最低限 3 種類のファイルを作成する必要がある。CCS のエディタでファイルを作成する場合はメニューバーで File→New→Source File を選択するとエディタが立ち上がる。作成したファイルはプロジェクトに対応するフォルダに格納する必要がある。

① C 言語ソースファイル

② リセットや割り込みに対するベクタ

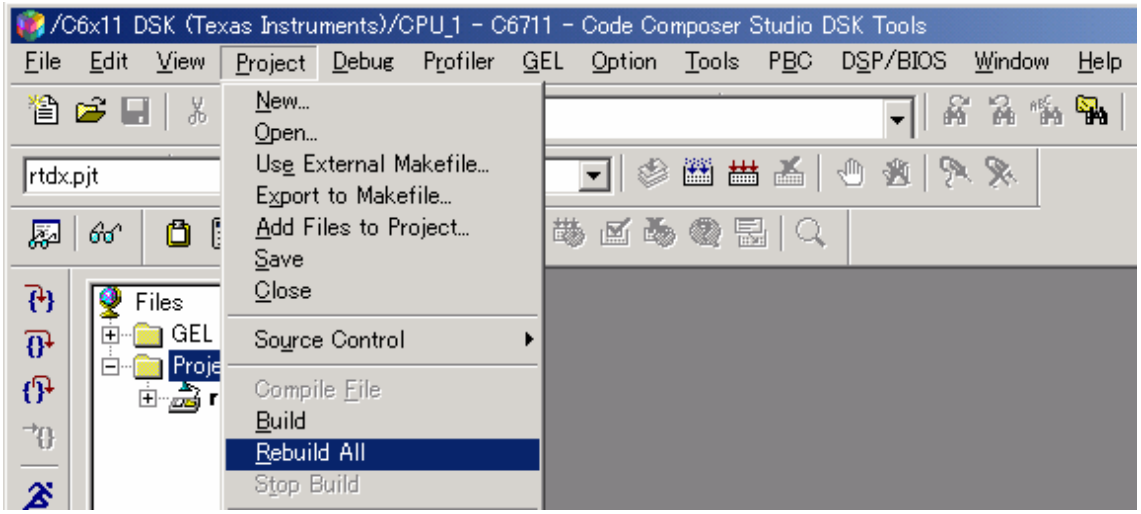
このファイルは TMS320C6x のアセンブリ言語で書く必要がある。

③ リンカ・コマンド・ファイル

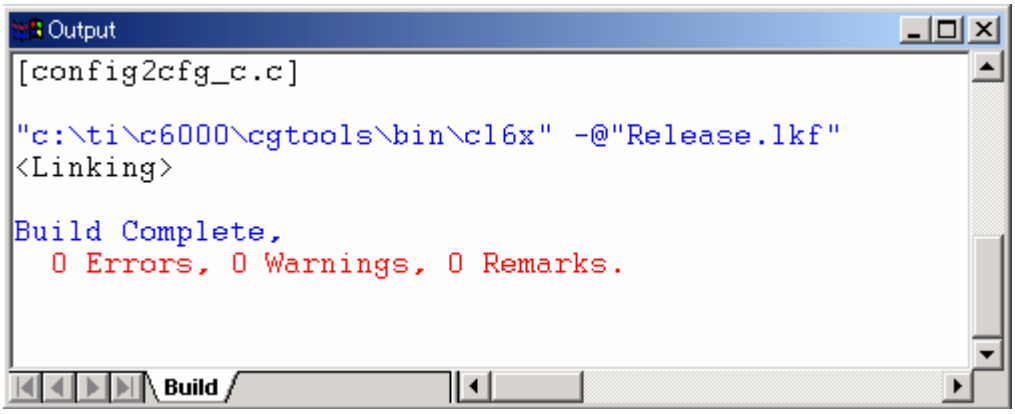
リンカがターゲットのメモリにセクションを配置する際に参照するファイル

プログラムのビルドと実行

(1) Project→Rebluilt All の順に選択する。



これにより、CCS は、プロジェクトに含まれるすべてのファイルに対するコンパイル、アセンブル、リンクをやり直します。このプロセスについてのメッセージは、ウィンドウの一番下のフレームに表示されます。Errors がない場合は Build Complete が表示されます。



(2) File→Load Program の順に選択する。リビルドしたプログラムである rtdx を選択し、開くをクリックします。CCS は、このプログラムをターゲット DSP にロードして、プログラムを構成する命令をディスアセンブルして表示する Dis-Assembly ウィンドウを表示します。

(3) プログラムの実行

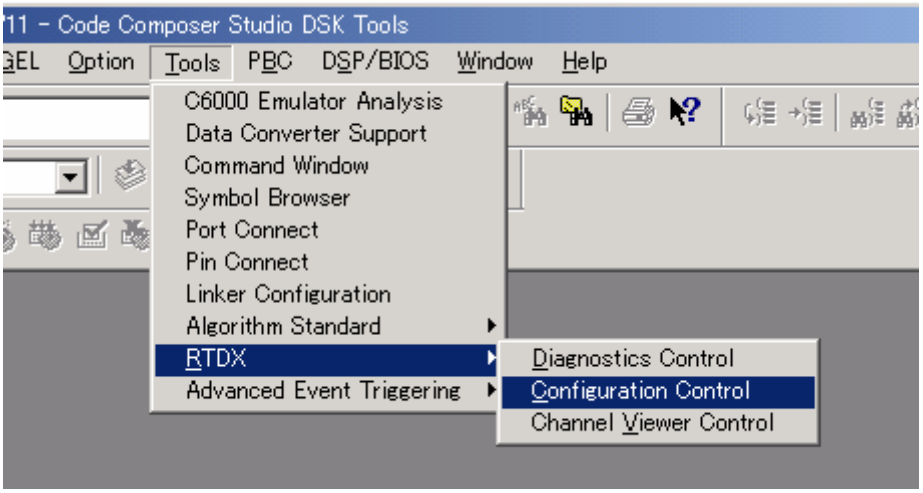
Debug→Run の順に選択する。

RTDX(Real-time data exchange)

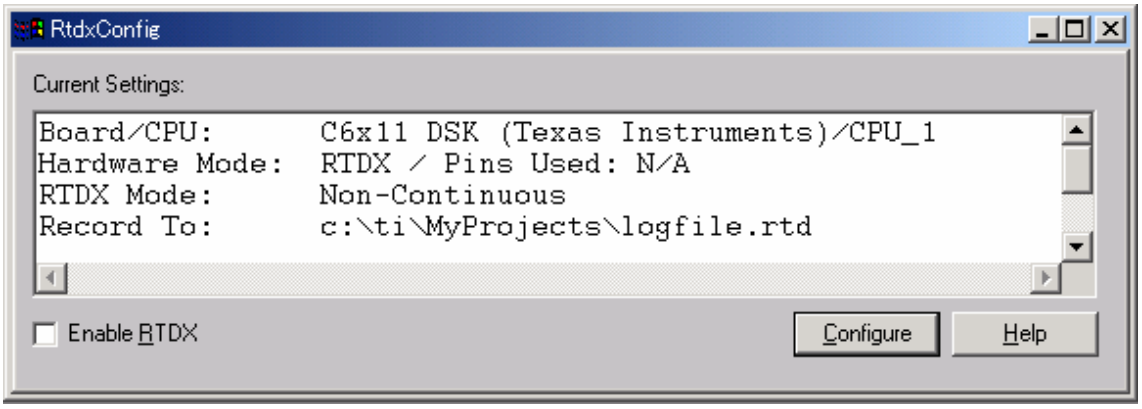
RTDX は CCS に搭載されたホストとターゲット間のリアルタイムデータ通信機能である。RTDX は CCS のリアルタイムデバッグを可能にする基幹技術である。

プログラムの演算結果を Excel にデータを送るため RTDX の機能を使用する。

(1)メニューバーの Tools→RTDX→Configuration Control の順に選択する。

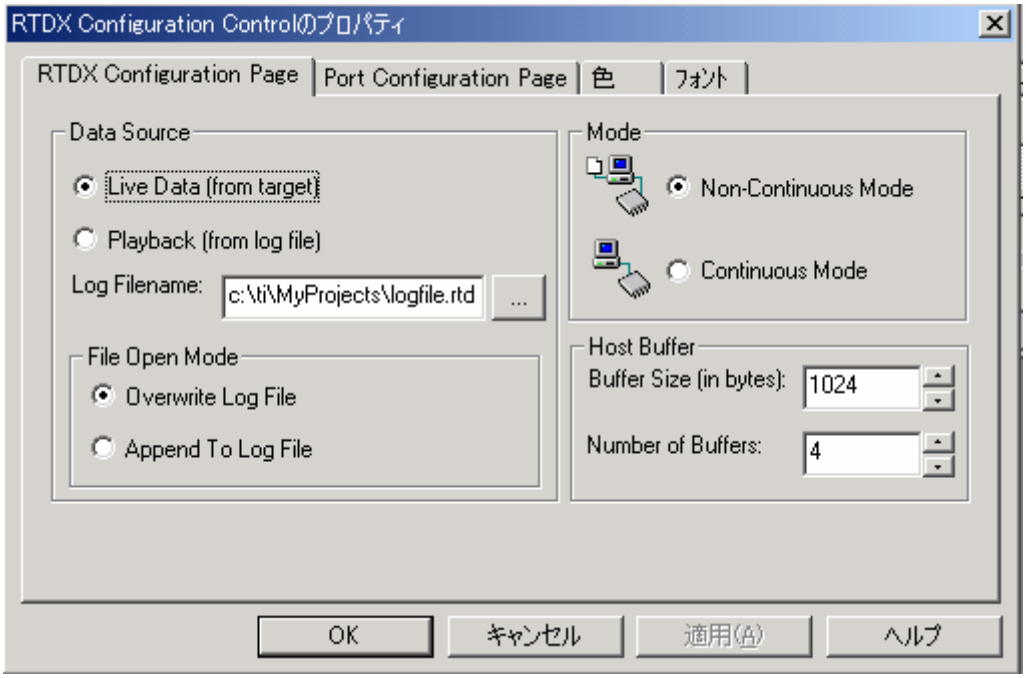


(2)RtdxConfig の画面が表示される。Configure をクリック



(3)RTDX Configuration Control のプロパティが表示されるので、Mode の Non-Continuous Mode に選択されていたら Continuous Mode に設定し、OK をクリック。

(4) RtdxConfig の画面に戻り EnableRTDX にチェックをつける。



(4) 演算結果は Excel の lab13 に 100 個ずつデータが送られるように設定されている。送るデータは 2 位相ロックインアンプの X 成分と Y 成分である。

ブレーク・ポイントとウォッチ・ウインドウ

プログラムの開発とテストにおいては、しばしば、プログラムの実行中における変数の値をチェックする必要があります。そのときにはブレーク・ポイントとウォッチ・ウインドウを使ってその値を表示させます。

(1) File→Reload Program の順に選択します。

(2) Project View でファイル rtdx をダブルクリックします。

(3) 止めたいプログラムの行にカーソルを移動します。

(4) ツールバーの Toggle Breakpoint ボタンをクリックするか、F9 を押します。その行が赤で強調表示されます。(表示の色を変えるには、Option→Color から選択)

(5) View→Watch Window の順に選択します。Code Composer Studio のウインドウの右下の隅に独立した領域が現れます。プログラムの実行時には、この領域に対象とされる変

数の値が表示されます。

(6) Watch ウインドウの領域で右クリックしてポップアップ・リストから Insert New Expression を選択します。

実行時間測定

実行時間は、以下の手順で測定する。

(1)実行コードをロードする。

(2)メニューバーで[DSP/BIOS]を選択し、[statistics]をクリックし、”statistics view”ウィンドウを開く。このウィンドウの大きさを調整し見やすいようにする。

(3)メニューバーで[Debug]を選択し、[Run]をクリックし、プログラムを実行させると”statistics view”ウィンドウに実行に要したサイクル数が表示される。

(4)この DSK の TMS320C6711 のサイクルタイムは 26.67ns なので、実行時間は表示されたサイクル数に 26.67ns を乗算すればよい。

ロックインアンププログラム

```
/* Include files */
#include "config2cfg.h"
#include <c6x11dsk.h>
#include <c6x.h>
#include "DSK_polling.c"
#include "DSK_intr.c"
#include <math.h>

/* Defines */
#define ORDER 1000
#define S 400.0
#define pi 3.14159265358979323846264338327 95

/* Prototypes */
void LEDout();
void FIR_BPF();
void ISR_Rcv();

/* Variables */

float scoeffa1 = 0.0;
float coscoeffa1 = 0.0;
short datin[ORDER+1];
short datsin[ORDER+1];
float kakesin[ORDER+1];
float kakecos[ORDER+1];
float datcos[ORDER+1];
float nyusin[1];
float nyucos[1];
float nyusyutu[1];

float h=1.0;
float tmpadd;
float costmpadd;
int acount;
float sintmpadder[100];
float costmpadder[100];

unsigned int tmp;

extern far LOG_Obj LOG0;
extern far SWI_Obj SWI_FIR_BPF;
extern far STS_Obj STS0;

/* main */
void main()
{
      acount=0;
```

```
    McBSP0Init();                   /* Initialization of McBSP0          */
    CodecInit();                    /* Initialization of CODEC           */
    CacheSet(cach_bk4);             /* cashe size: 64KBytes of L2        */

    //for (k=0; k<=100; k++) xn[k] = 0.0;
    tmp = 0x07000000;
    *(u_v_int *)IO_PORT = tmp;      /* trun off LEDs                     */

/* Default assignment of McBSP0 Receive using DSP_BIOS is INT11 */
    IER |= 0x2 | (1<<11);           /* INT11 enable                      */

        return;
}

/* Software interrupt service routine for PRD_swi:  Count-up LEDs */
void LEDout()
{
    LOG_printf(&LOG0,"%d",sintmpadder[0]);

     tmp = tmp - 0x01000000;
     *(u_v_int *)IO_PORT = tmp;
}

/* Software interrupt service routine for SWI_FIR_BPF:  FIR filter */
void FIR_BPF()
{
    int k;
        float tmpin;//tmpcos,tmpsin;

        STS_set(&STS0,CLK_gethtime());

    tmpin = McBSP0Read();       /* input */
    nyusin[0]=32767.0*sin(2.0*pi*S*h/8000.0);
    nyucos[0]=32767.0*cos(2.0*pi*S*h/8000.0);
    nyusyutu[0]=32767.0*sin(2.0*pi*S*h/8000.0);
    if(h>=8000){
    h=0.0;
    }
    h=h+1.0;

    McBSP0Write((short)nyusyutu[0]  &  0xFFFE);                    /*  output
*/

#pragma UNROLL(1);
    for (k=ORDER; k>0; k--) {       /* shift input samples    */
        datin[k] = datin[k-1];
        datsin[k] = datsin[k-1];
        datcos[k] = datcos[k-1];
        }

        datin[0]=tmpin;
        datsin[0]=nyusin[0];
```

```
        datcos[0]=nyucos[0];
        tmpadd=0;
        costmpadd=0;
    for (k=ORDER; k>-1; k--){           /* shift input samples        */
        kakesin[k] = (datin[k] * datsin[k]);
        kakecos[k] = (datin[k] * datcos[k]);
        tmpadd = tmpadd+kakesin[k];
        costmpadd = costmpadd+kakecos[k];

            }

        sintmpadder[acount]=tmpadd/(ORDER+1);
        costmpadder[acount]=costmpadd/(ORDER+1);
        acount++;

        if(acount==100)
        {
        acount=0;
        }

        STS_delta(&STS0,CLK_gethtime());
}

/* Interrupt service routine for HWI_INT11 (McBSP0 receive) */
void ISR_Rcv()
{
    SWI_post(&SWI_FIR_BPF);             /* post SWI_FIR_BPF              */
}
```

RTDX プログラム

```
#include <rtdx.h>
extern float costmpadder[100];
extern float sintmpadder[100];
//extern short fft_power[];       /* FFT power result                                    */
int status = 0;                   /* RTDX write status: [0]failure [!0]success */
RTDX_CreateOutputChannel(fft_channelx);   /* create output channel to Host     */

/* fft_rtdx */
void fft_rtdx( void )
{
        float b[2];

    RTDX_enableOutput(&fft_channelx);
```

```
            b[0]=sintmpadder[0];
            b[1]=costmpadder[0];

        status = RTDX_write(&fft_channelx, b, sizeof(b));

        RTDX_disableOutput(&fft_channelx);
}
```