# 実験物理と計算機「LabVIEW による装置制御と解析」

授業用テキスト　Ver.1.2（ 2007 / 12 ）

第〇回　12 月 06 日　ガイダンス　＆　実験室の席決めおよび PC の割り当て
第一回　12 月 13 日　コンピュータおよび LabVIEW についての概説
第二回　12 月 20 日　プログラミング解説１
第三回　 1 月 10 日　プログラミング解説 2
第四回　 1 月 17 日　プログラミング解説 3
第五回　 1 月 24 日　計測器を用いた信号の入出力実験
第六回　 1 月 31 日　データ解析手法
第七回　 2 月 07 日　計測器を用いた実験１
第八回　 2 月 14 日　計測器を用いた実験２
第九回　 2 月 21 日　レポート作成または予備日

## 授業の概要

基礎研究や開発の現場において、最近良く用いられているプログラミング言語「LabVIEW」を学習し、これを用いて PC に接続された測定機器の制御を行い、簡単な測定および解析までを行う。

通常のプログラミング言語（C 言語や FORTRAN 等）は、決められた命令文と文法に従い文章を書いて必要とする機能を実現している。これに対し LabVIEW 言語は、各種機能を持ったアイコンを順番に並べ、それらを配線で接続して機能を実現するので、簡単で直感的にプログラムが記述できる。加えて充実したユーザーインターフェースや高度な解析機能を有していること、そして Windows や MacOS、Unix 等、複数の OS に対応しており、プログラムの移植が容易であるなど、計測制御を行うプログラム作成に非常に適した高級言語である。

授業では、各個人割り当てのパーソナルコンピューターを使用し、まずプログラミング言語一般に共通して用いられる基礎的な命令群を習得することから始める。その際には、初めてプログラミングを行う者でも理解できるように、LabVIEW 言語と他の言語を対比させて解説する。その後過渡現象を教材とし、LabVIEW 言語を用いて、USB ポートを経由した計測機器の制御および信号取り込みを実践し、結果の解析まで行う予定である。

## 目標

課題の実習を通してプログラミング一般の知識を身に付けること、特に LabVIEW 言語で書かれたプログラムを読み取れるようになり、またプログラムを作成して簡単な計測機器の制御を自分で行えるようになること。

※本テキスト中に問題点や誤植等を見つけられましたら、担当まで随時お知らせ下さい。

# 第一回　コンピュータおよびLabVIEWに関する概説

## 1）授業で使用するコンピュータシステムの構成

| | |
|---|---|
| ■ＣＰＵ | インテル(R) CeleronDプロセッサ 341 （ 2.93 GHz ） |
| ■メモリ | 256 ＭＢ |
| ■オペレーティングシステム | Windows ＸＰ |
| ■ディスク容量 | 80 GB, 7200 rpm |

## 2）配線と接続

今回、外部計測装置とコンピュータの間の通信（入出力）に使用するポート（接続口)は、USBポート（GPIBポート）である。各自コンピュータの背面を見て確認しておくこと（下図参照)。

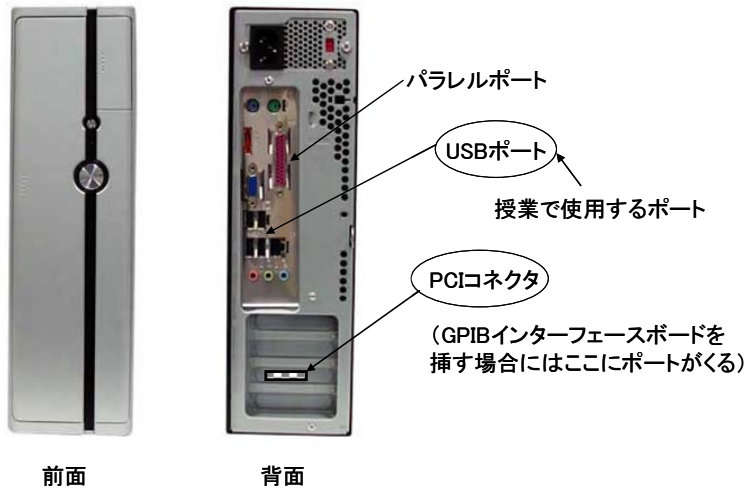



図１：コンピュータの外観.

## 3）コンピュータの起動と終了

**起動**：電源をONし、オペレーティングシステムを起動する。各自ユーザー名として、「User」を割り当ててあるので、この名のアイコンをマウスで左クリックし、ログインする。

**終了**：画面左下「スタート」ボタンをマウスで左クリックし、「終了オプション」を選択して現れる「電源を切る」ボタンを左クリックして選択する。これでコンピュータの電源が自動的に切れる。


ログイン画面


終了画面

図２：起動／終了画面.

## ４）データ保存領域の確認

既定のデータ保存領域は、「マイ　ドキュメント」フォルダ内に設定してある。


図３：授業で使用する格納用フォルダ.

ここ（実際には、C:¥Documents and Settings¥User¥My Documents　になる）の中に、各自のフォルダ（例えば　User_01　等）を作成して、今後作成する各種ファイルを保存せよ。授業で使用するサンプルプログラムは、「Learning Directory」フォルダに保存してある。

> ※　注意　※
> サンプルプログラムを開いて実行した後、プログラムを終了する際（フロントパネルの「ファイル」メニューの「終了」を選択するか、フロントパネル右上の×印をクリックする）に、「プログラムの変更を保存しますか」と問われるが、ここは「**保存しない**」を選択すること。

## ５）拡張子の扱い

コンピュータで扱われるファイルは通常、それぞれ内容・用途別に拡張子を付けて区別される。拡張子とは、ファイル名の後に付ける付加的な文字列のことである。代表的なものとして以下のような例が挙げられる。

| | |
|---|---|
| AAA.**txt** | テキストファイル |
| BBB.**exe** | 実行形式のファイル |
| CCC.**lib** | ライブラリのファイル。 |
| DDD.**f** | FORTRAN プログラムファイル |
| EEE.**c** | C プログラムファイル |

拡張子が付加され、対象ファイルの属性が直ぐに判別できるようになっていることで、ファイル操作やプログラム作成時の効率が著しく改善される。LabVIEW 言語で作成・実行されるプログラムファイルは、「sample.**vi**」のように、拡張子「vi」を付けて表され、LabVIEW プログラムがインストールされたコンピュータでは、この拡張子が自動的に認識されるようになっており、「xxxx.**vi**」ファイルをダブルクリックするだけで、LabVIEW プログラムが起動する。

## LabVIEW 言語と他のプログラミング言語の違い

これまで計測用途に一般に用いられてきたのは、主として**書き下し型**のプログラム言語（BASIC,FORTRAN,C 言語,VisualBasic,…)であった。これらの言語によるプログラムは、基本的に文字列で記述されており、個別の命令を組み合わせて文章を作成する要領で作られる。プログラム中で演算するデータや計測器等から取得した信号は、変数に入れられて文章内で操作されることになる。我々は、テキストエディタ(Unix 系なら vi や emacs、Windows なら notepad や WordPad 等)で、変数を基にして構成されたプログラムの文章を作成し、最終的にそれを「コンパイル」することによって、初めて実行可能なファイルを得ることになる。

```
C          LEGENDRE POLYNOMIALS COEFFICIONS2
C          INITIAL SETTING
      INTEGER H,I,J,K,L,M,N
      REAL X,P,Q
      PARAMETER (H=100)
      INTEGER PRECO(0:H),BUNBO
C          LEGENDRE PARAMETER N,M & INSTITUTION VALUE X(DEGREE)
      READ(*,100) N,M
      READ(*,150) X
C          INSTITUTION 1
 100  FORMAT(I7)
 150  FORMAT(E15.7)
      DO 10 I=0,N
         J=N-I
         PRECO(I)=KAIJO(N)*((-1)**J)/KAIJO(I)/KAIJO(J)
 10   CONTINUE
C          INSTITUTION 2
      L=0
      DO 20 I=0,N
         J=I*2-(N+M)
           IF (J.LT.0) THEN
            PRECO(I)=0
            L=J+2
            GOTO 20
         ENDIF
         DO 30 K=0,N+M-1
            PRECO(I)=PRECO(I)*(2*I-K)
 30      CONTINUE
 20   CONTINUE
      BUNBO=(2**N)*KAIJO(N)
C           CALCULATION
      P=0.0
      DO 40 I=(L+N+M)/2,N
         J=2*I-(N+M)
         P=P+PRECO(I)*((COS(X))**J)
 40   CONTINUE
```

図４：FORTRAN プログラムの例.

これらに対し、LabVIEW 言語は、「**グラフィカルプログラミング言語**」と呼ばれている。LabVIEW 言語においては、言語を構成する個々の命令それぞれに対応した「**アイコン**」が存在しており、演算するデータや取得した信号の流れに沿ってこれらを「**配線**」で結ぶことでプログラムが作られる。つまり、LabVIEW では基本的に、データの流れを可視化してプログラムするのである。ここが書き下し型のプログラム言語との大きな違いとなっている。

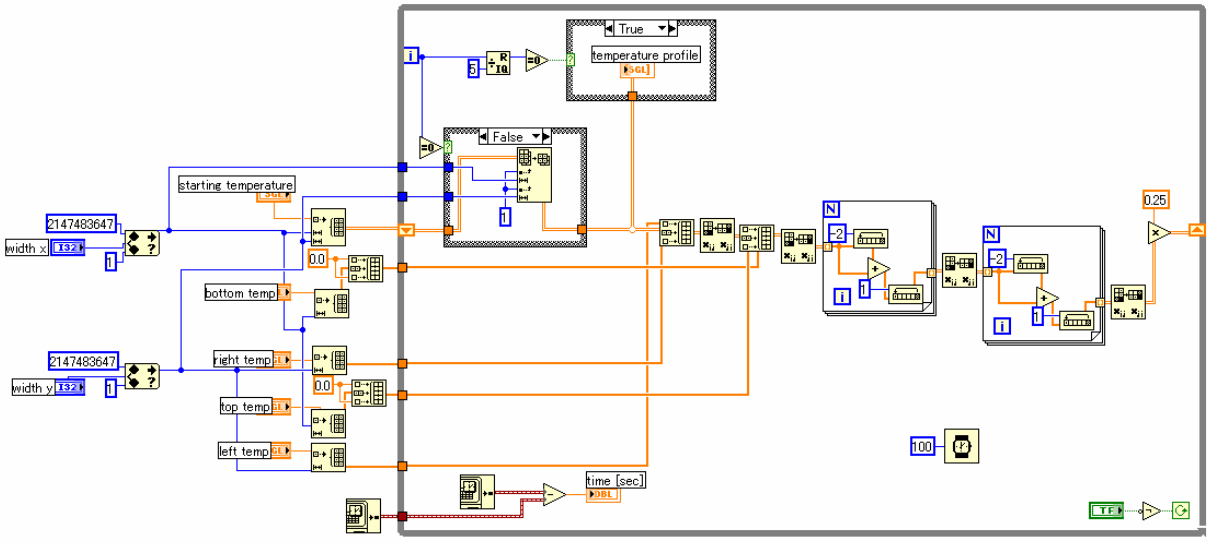



図５：LabVIEW プログラムの例．信号の流れが配線で表されており、各アイコンの中で、またはその組み合わせによって信号が逐次処理されている．

## 「フロントパネル」と「ブロックダイアグラム」

作成したプログラムを実行して実際の測定やシミュレーションを行う際には、その途中経過や最終結果を確認するために、何らかの方法でそれをＰＣ画面に表示させなければならない。例えば横軸を時間や濃度、縦軸を吸収や発光の強度として、測定結果の変化を二次元(または三次元)グラフで、刻一刻と表示したい場合などが良くある。グラフィカル言語 LabVIEW 以外のプログラミング言語においては、大体以下のようにしてこの要求を満たしている。

１）実行プログラムは数値出力のみを与え、その表示は他のプログラムで行う。

※　例えば FORTRAN や C、PASCAL 言語など．

この場合、ＰＣ画面にはプログラム実行結果の数値が単にテキスト表示されるだけとなり、得られた数値データは、配列データとしてテキストファイル等へ記録される。したがって、表示または解析を行うためには、別のソフトウェアが必要となる。

２）結果の出力表示も同じプログラムで行う。

※　例えば BASIC(QuickBasic や VisualBasic など)や Visual C++、Delphi 言語など.

これらには、結果をグラフィカルに表示するための命令がそれぞれ含まれており、画面表示をそのプログラム言語自体で行うことになる。測定やシミュレーション中、およびその結果を即座に表示して、全体像をすばやく見渡せる利点がある。

LabVIEW における結果の表示は、後者２）の形式になっている。さらに、プログラム本体とその結果を出力するインターフェースが、プログラミングの過程で既に一体のものとして存在しているのが、LabVIEW 言語の特徴である。これらはそれぞれ、「ブロックダイアグラム」と「フロントパネル」と呼ばれている。図 6 にプログラムの一例を示す。

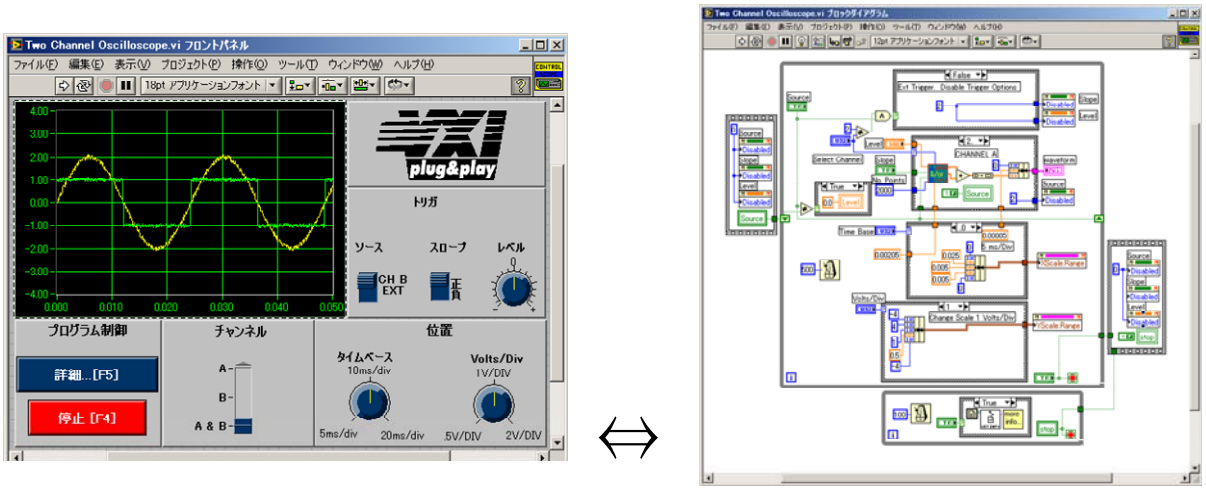



図６：「フロントパネル」(左)と「ブロックダイアグラム」(右).

●フロントパネル　--- プログラムを実行し、パラメータの変更や結果の表示及び確認を行う.
●ブロックダイアグラム　---プログラム本体を記述する.

フロントパネル上で、表示部品(グラフや数値制御器、テキスト表示器、ボタン等)を設置すると、ブロックダイアグラム上に、それに対応したプログラム命令ブロックが自動的に生成される。ブロックダイアグラム内で行われる演算結果などを、これらの表示部品に配線で繋いでやれば、結果はフロントパネル上の表示に反映されるようになり、結果を視覚的に確認することが出来る。

## ■　企業等での使用例　■　http://www.ni.com/labview/ja/　参照

１）Microsoft 社による NI LabVIEW と PXI モジュール式計測器を使用した Xbox 360 コントローラ用製造テストシステムの開発

Microsoft Windows XP、Microsoft SQL サーバ、NI LabVIEW、NI PXI モジュール式計測器に基づいた柔軟性の高い自動テストシステムを使用して、有線式と無線式の両方の Xbox 360 コントローラの機能的性能のテストを行う。

２）Key Energy 社: 自動監視による油田の安全性の改善

NI 製品である CompactFieldPoint と LabVIEW Real-Time Module を採用することにより、グラフィカルなオペレータ端末で、掘削装置の位置（GPS）や掘削装置の動作、作業ステータスに関する運用データを収集し、ブロックの位置、フックの重量、ロッドのトルク、油圧／通気装置、エンジンの性能などの物理パラメータを監視する。

３）1997年、NASAによるSojourner Rover(火星探査機)を使った火星探査において、着陸船に対するこの探査機の位置、地面に対する向き、および探査機全般の物理的な状態などをモニタするために使用された。

http://www.findarticles.com/p/articles/mi_m0EIN/is_1997_July_18/ai_19593795

等など。

## ■　LabVIEW ソフトウェアの入手　■

http://www.ni.com/labview/ja/　ここから、「学生版」は安価に購入出来る（2006/12 現在）。

□ 製品名称：LabVIEW 8.20 Student Edition
□ 価格：4,200 円（消費税込み）
□ 対象者：学生のみ（購入の際、在学証明書（学生証のコピー）の提出が必要になる）
□ 動作環境　：Windows 2000/XP インストール済みの PC　/　100MB 以上のドライブ空き容量

【推奨ハードウェア】 DAQ 信号アクセサリ、ケーブル、GPIB インターフェース等

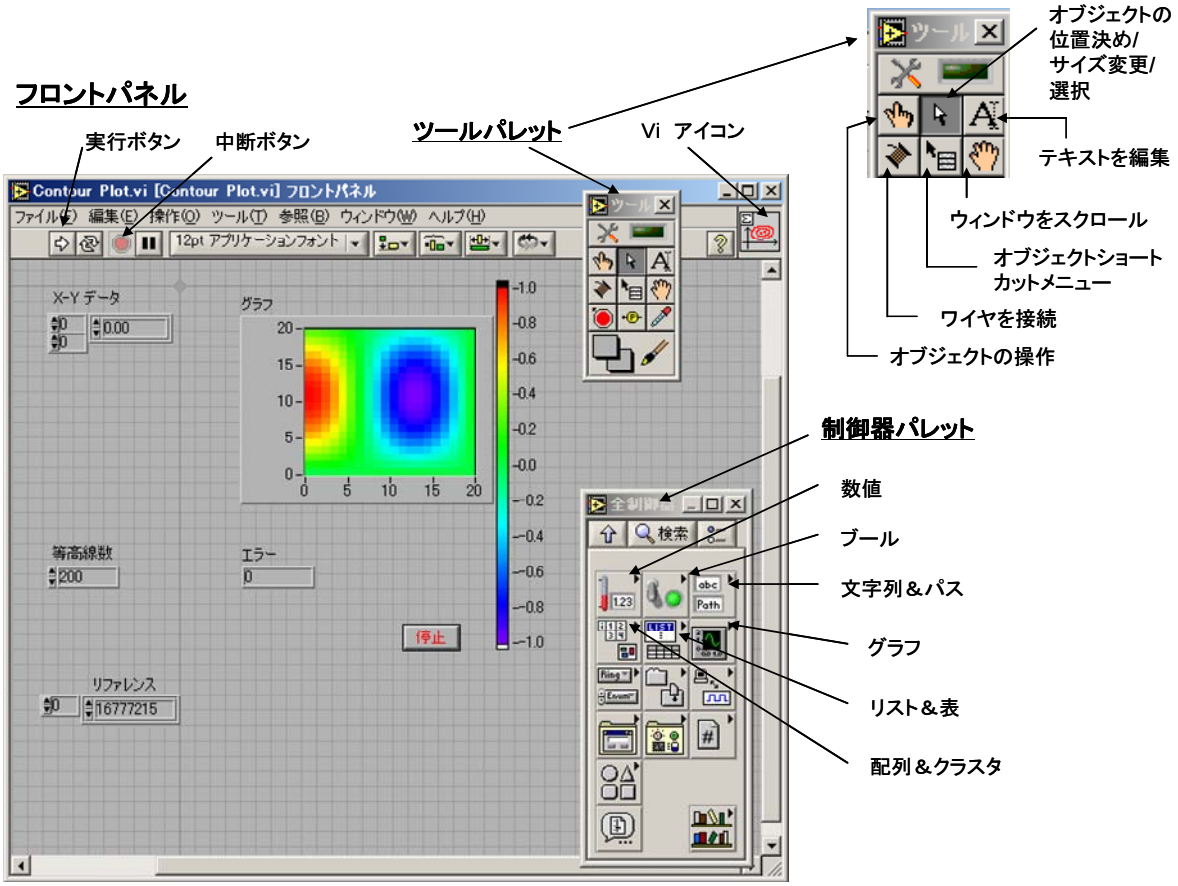
【オプション】 メモ帳、ワードパッドなどのワープロ用ソフトウェア

## ■　LabVIEW 言語の参考書　■

LabVIEW プログラミングガイド グラフィカル言語による PC ベース計測とデータ解析
尾花健一郎 訳、日本ナショナルインスツルメンツ株式会社 監訳
形態：B5 変（568 ページ）ISBN：4-7561-4585-X

## フロントパネルとブロックダイアグラム画面の各部名称と意味

フロントパネルの編集では「制御器パレット」が、ブロックダイアグラムの編集では「関数パレット」がプログラムの部品を供給する。部品の移動や配線は「ツールパレット」でツールを選択して行う。プログラム作成において主に使用するボタン類を図 7 に示す。



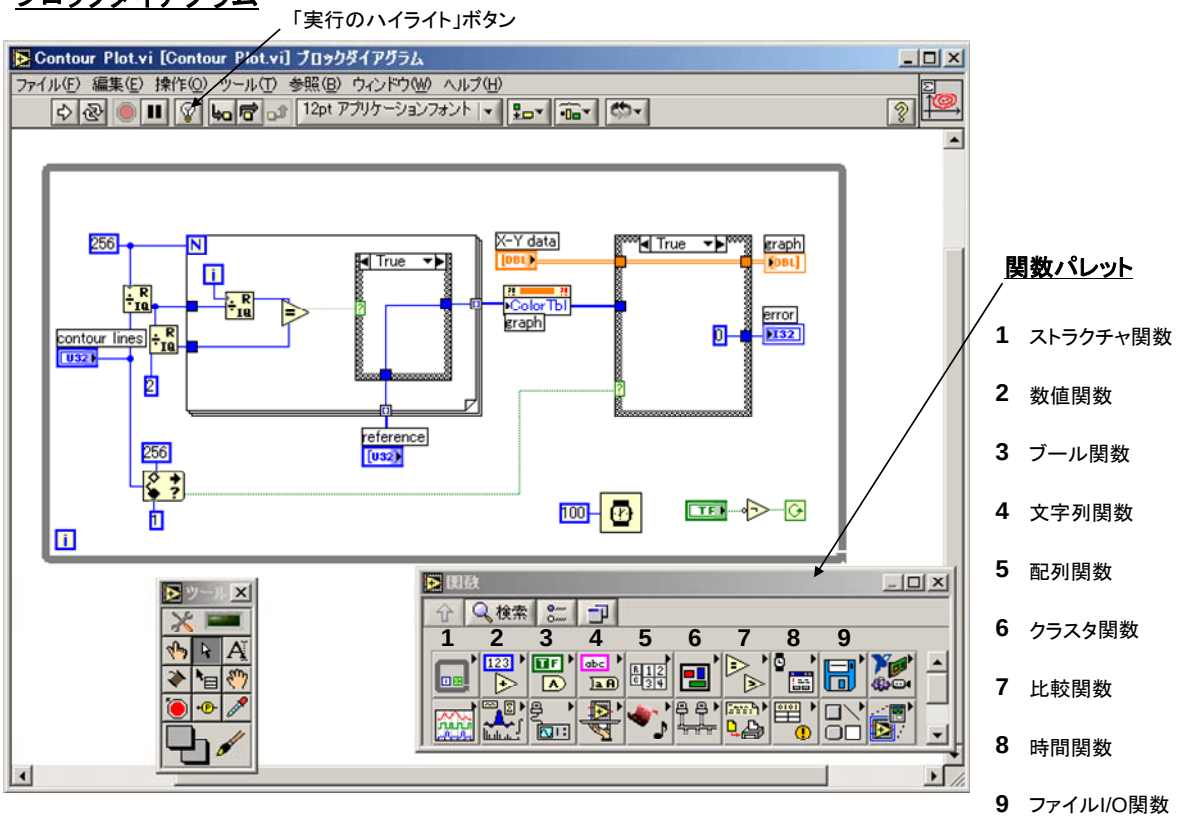




図 7 ：フロントパネルとブロックダイアグラムの各部名称.

※　　部品を配置するための「**ツールパレット**」、プログラム部品を選択する「**制御器パレット**」および「**関数パレット**」は、フロントパネルとブロックダイアグラムにおいて、「ウィンドウ」メニューから「ツールパレットの表示」／「制御器パレットの表示」／「関数パレットの表示」を選択して表示させる。

## ■　制御器パレットの部品について

制御器パレットには、数値や文字列をそのまま、またはチャートやグラフの形で表示するための「表示器」や、プログラム実行者が値を書き込んだりスイッチを押せたりする「制御器」などが、一覧表示されている。このパレットから適宜選択して、制御器または表示器等をフロントパネル上に配置してゆくことにより、プログラム使用者向けのインターフェースが完成し、計測や解析の作業を対話的に行えるようになる。

フロントパネル上に制御器や表示器を配置すると、前述のようにブロックダイアグラム上にこれらに対応する部品が現れる。LabVIEW においては、ブロックダイアグラム上のプログラムの進行は、横書きの文章のようにおおむね画面の左側から右側へ向かって進行する。従ってこの流れに沿って配線が行えるように、ブロックダイアグラム上の各種部品を配置するのがよい。ブロックダイアグラム上で、制御器の**配線端子**が右側にあり、表示器の**配線端子**が左側にあるのは、このプログラムの流れを意識したものである。なお、ブロックダイアグラム上で制御器は一般に**太枠**で表示され、表示器は**細枠**で表示されるが、これらはそのアイコンを右クリックして現れるメニューで相互に変換できる。

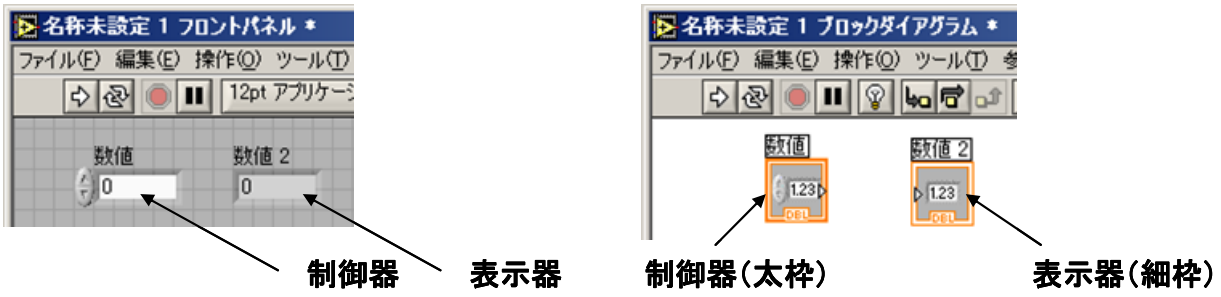



図 8：数値制御器と数値表示器の例.

## ■　ツールパレットについて

1）ツールパレット上の「オブジェクトの操作」ボタン  は、フロントパネルに配置したボタンを押す、数値制御器において数値を上げ下げするなど、制御器をプログラム実行前もしくは実行中に操作する際に使用する。

2）「オブジェクトの位置決め/サイズ変更/選択」ボタン  は、制御器を制御器パレットからフロントパネル上へ、または各種関数を関数パレットからブロックダイアグラムへ持ってきたり、これらのサイズを変更したり、それぞれの場所で移動させたりする際に使用する。

3）「テキストを編集」ボタン  は、数値制御器や文字列制御器内に、文字を書き込む際に使用する。

4）「ワイヤを接続」ボタン  は、ブロックダイアグラムにおいて部品間の配線を行う際に使用する。配線の色は、接続する部品に応じて変化する。配線を流れるデータが実数値なら**橙色**、整数値なら**青色**、文字列なら**紫色**、真偽の値なら**緑色**、ファイルのパスなら**青緑色**となるので、プログラム内でどのようなデータが扱われているかが一目瞭然となる。

## ■ ブロックダイアグラム上の関数パレットの部品について

ブロックダイアグラム上で、関数パレットから選択して配置された各関数は、ブロックダイアグラム内のみでその機能を果たし、フロントパネル上に対応する部品は現れない。LabVIEW プログラミングで我々が行うのは、フロントパネル上に配置された制御器や表示器の値を、この関数パレット内の関数で操作・加工して、その結果をまた制御器や表示器の値に反映させることが殆ど全てである。測定装置からのデータの取得や装置の制御といった外部機器の制御、計算や解析結果のファイルへの書き込みや読み込み、といったデータの入出力を行うのも、関数パレット上に置かれた各種の入出力関数を用いて行われる。

## ■ ヘルプについて

各部品について、その働きと接続する配線の種類等の情報を見るためには、フロントパネルかブロックダイアグラムにおいて、「ヘルプ」→「詳細ヘルプを表示」を実行する。すると、詳細ヘルプの別窓が常時開いた状態になり、カーソルを制御器や表示器および関数上に置いた際に、これらの簡易的な情報が見られるようになる。更に詳細な情報を得るためには、詳細ヘルプ内にある「ヘルプを表示するには、ここをクリック」の箇所をクリックすればよい。

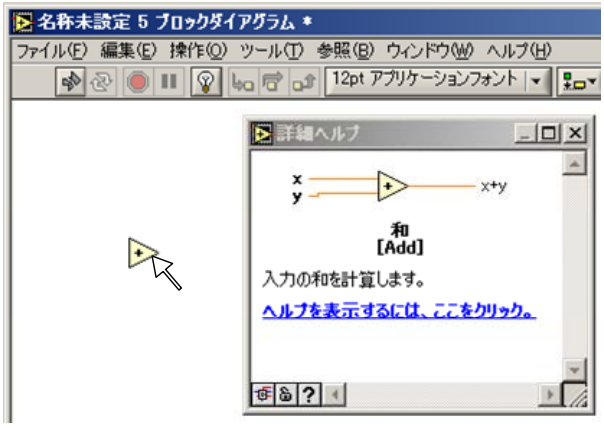



図 9：詳細ヘルプを使用している場面.

## ■　実際に部品を配置して、簡単なプログラムを作成してみる

**数値を二つ入力して、同じなら LED ランプ点灯し、また和と積を表示させるプログラム**

**［　使用する部品類　］**

フロントパネル：　数値制御器×２、数値表示器×２、　LED 表示器×１

ブロックダイアグラム：　上記部品に対応する各ブロック、数値関数（和・積）、比較関数

１）LabVIEW 起動後の初期画面で、「新規」→「ブランク VI」を選択して、ブランク(空)の VI フロントパネル(とブロックダイヤグラム)を作成する。

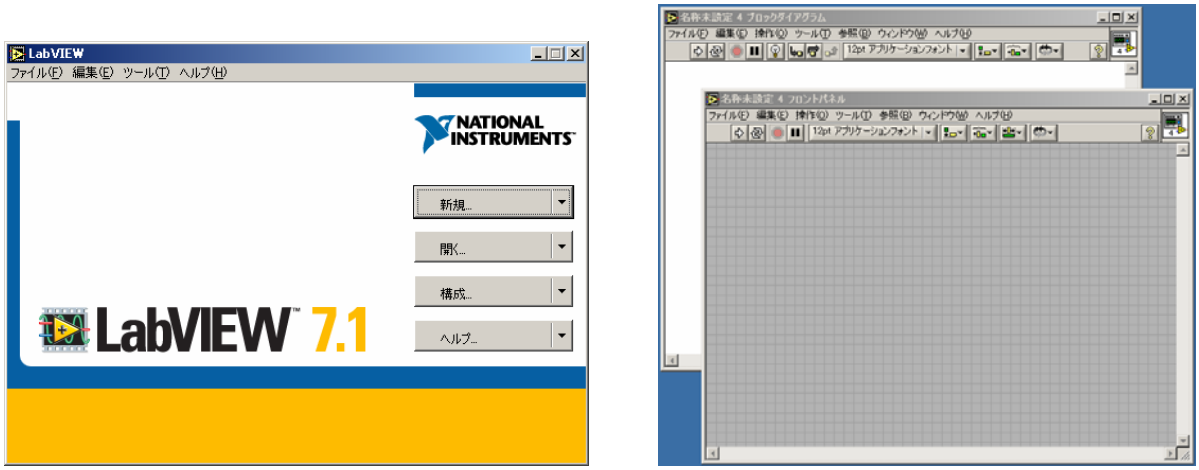



図 10：起動後の初期画面（左）と、ブランク VI（右）.

２）ツールパレットの「オブジェクトの位置決め/サイズ変更/選択」ボタン  を用いて、制御器パレットから数値制御器、数値表示器、LED ランプの部品をそれぞれ選択してフロントパネル上に配置する。ラベル名もツールパレットの「テキストを編集」ボタン  を利用してそれぞれ変更せよ。

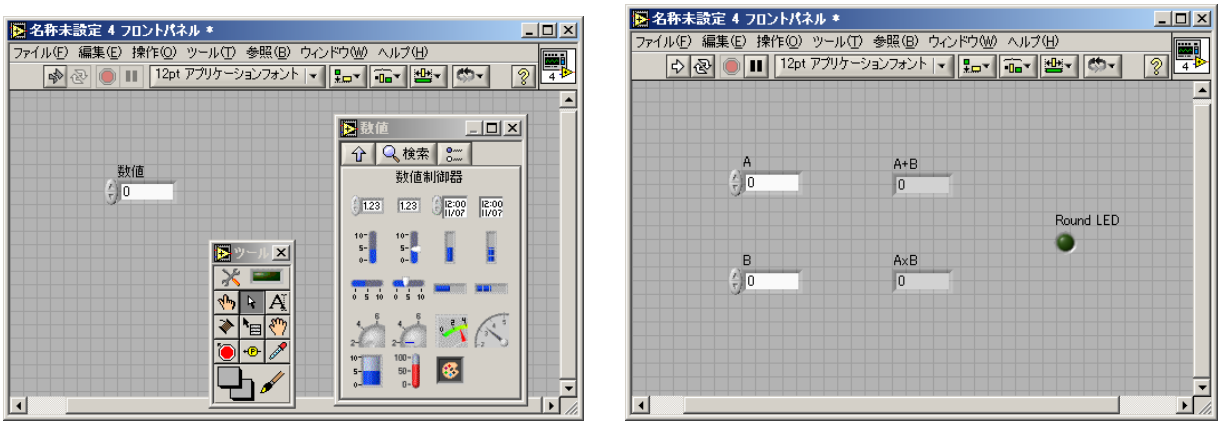



図 11：数値制御器の配置過程（左）と、その他部品を含めた配置完了図（右）.

３）ブロックダイアグラム上で、ツールパレットの配線ツール  を用い、各部品間に配線する。和と積の演算関数と比較を行う関数は、関数パレット上の「数値関数」と「比較関数」からそれ

ぞれ取得できる。

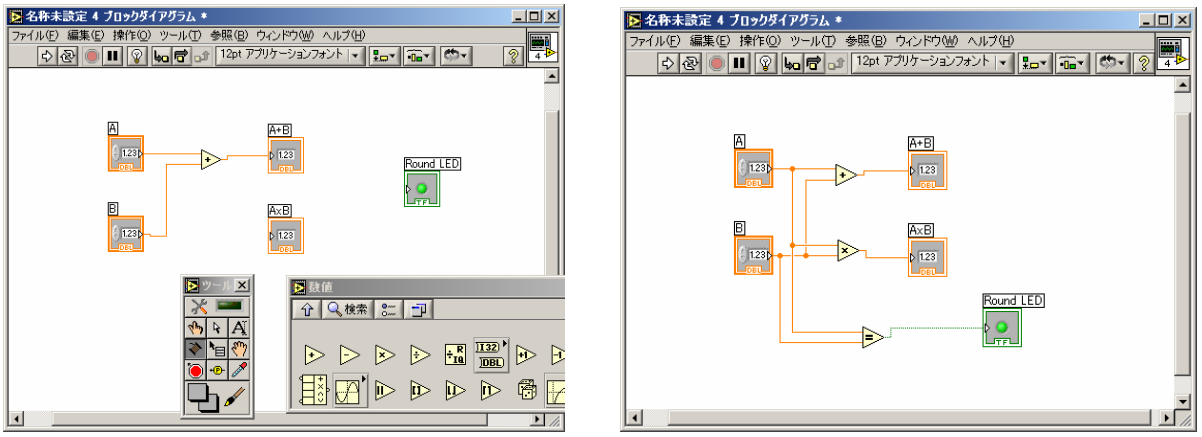



図 12：数値関数の配置とワイヤ接続（左図）．右は配線が完了した状態．

接続を間違えると配線は繋がらず、この場合配線は実線ではなく破線で示されるので、配線上で右クリックして現れるメニューのうち、「分岐ワイヤを削除」を選んでその配線を削除し、配線をやりなおすこと。

4）フロントパネル（またはブロックダイアグラム）上で、実行ボタン  を押し、作成したプログラムを実行する。この際、ブロックダイアグラム上で「実行のハイライト」ボタン  を押した後にプログラムを実行させると、配線上のデータの流れが可視化されるようになり、プログラムの問題箇所の発見や修正が容易になる。


図 13：「実行」ボタンと「実行のハイライト」ボタン．

配線が不正確であったり、部品の配置が間違っていたりすると、実行ボタンが  のように表示され、プログラムの実行は出来ない。この場合、  のアイコンをクリックすれば、**エラーリスト**が表示されるので、それを見て誤った箇所を訂正し、プログラムが実行できるようにする。

**問題１－１：**

数値 A と B の値を、ツールパレットの「オブジェクトの操作」もしくは「テキストを編集」ボタンを押して得られるモードで変更し、和と積の値がそれぞれ正しく表示されるか、また A=B のときに LED が点灯するかを確かめてみよ。

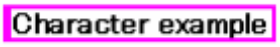
# 第二回　プログラミング解説１　----- プログラミングに必要な基本的知識 -----

## A)　変数およびその型式

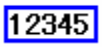
一般にプログラム言語においては、**数値変数**と**文字変数**は区別して扱う。LabVIEW においても同様であり、またこの他にも真偽を表す**ブール変数**やファイルのパスを示す**パス変数**がある。フロントパネルにおいて配置される制御器や表示器が、それ自体こうした変数になっている。

プログラムの中でどのような名前の変数を用いるか、またその**データの型**(数値変数なら整数、実数、単精度、倍精度など)は何であるかを明確に示すことを変数の**宣言**というが、LabVIEW では C や FORTRAN 言語などのプログラミング言語とは異なり、改めて変数を宣言する必要は無く、必要なときに部品としてブロックダイアグラム上に配置してゆけばよい。型に関しては、制御器(表示器)のプロパティ(右クリックで現れる)中の「表記法」メニューで選択することで、変更が可能である。四則演算や比較、条件判断などの**変数間の演算操作**は、関数パレット上にある各関数をブロックダイアグラム上に配置し、各変数からの配線を行うことで行われる。

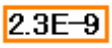
変数に代入される数字・文字を代表する「定数」については、以下のような表記になる。

**１．文字定数：**　“Character example”　のような文字列（紫色）　 


**２．数値定数**

**整数型：**　-1, 0, 12345, 23,　　のような整数値（青色）　 


**実数型：**　0.235, 52300, -0.023542,　　のような実数値（橙色）　 

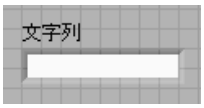

**指数型：**　2.3e-1, 5.23e+4, -2.3542e-2　のような指数表記の値（橙色）　 


※2.3e-1 は、実定数の１つの表し方であり、e-1 というのは $10^{-1}$のことである。すなわち、2.3e-1 = $2.3\times10^{-1}$ = 0.23 を意味する。

**３．論理定数：**　「真」の値　または「偽」の値（緑色）　  , 

「変数」に関しては上記と同様の色で、それぞれフロントパネルとブロックダイアグラム上に、

１．文字変数：





２．数値変数


整数型：


／ 実数型または指数型：


３．論理変数：



このような形で表される。プログラム中で同じ変数を多用する際に、配線が煩雑になる場合には、関数パレットの「ストラクチャ」欄にある「ローカル変数」 

をブロックダイアグラム上に配置し、そのプロパティ内で項目を選択すれば、これらの変数の分身を作成することができる（例えば、上記の実数型変数なら 

など）。この分身を用いれば、随意に各場所で変数へ配線したり配線を取り出したり出来る。

なお、数値変数に関しては、アイコンを右クリックして現れるメニューのうち、「表記法」において数値変数の型（実数，整数，複素数）および精度（単精度，倍精度）を指定して、属性を変更することができる。また、メニューのうち「プロパティ」を指定すれば、表記法および初期値や有効数字の桁数などの詳細な属性の変更もできる。

※ 数値変数に限らず、大抵の制御器および関数は、アイコンを右クリックして現れるメニュー内にある「プロパティ」項目で、その特性を確認および変更することが出来る。各アイコンのプロパティ項目を大体把握しておくことが、円滑に LabVIEW プログラミングを行うために重要である。

また、文字列関数と数値関数は、関数パレット上の「文字列/数値変換」項目にある各変換関数を用いて、互いに変換できる。これら関数を使用することにより、例えば文字列データファイルから読み込んだデータを数値に変換して、プログラム内で演算処理することなどが可能となる。

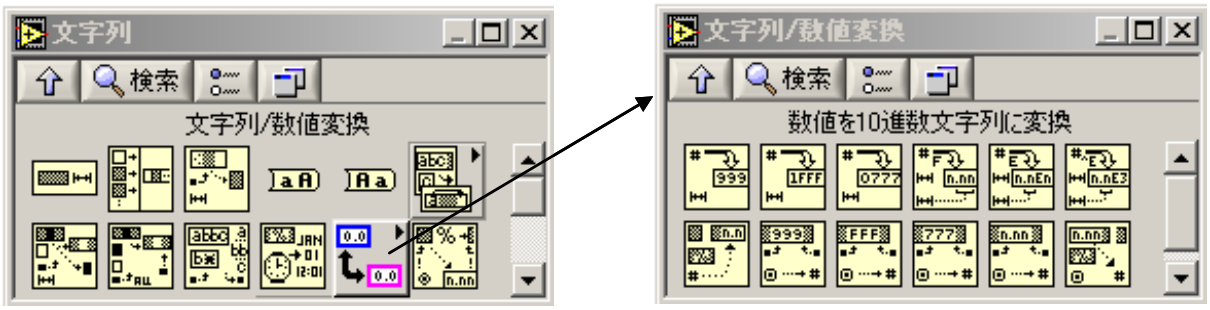



図 14：関数パレット上の「文字列/数値変換」項目、およびその中の各関数.

**問題２－１：**

関数パレット上の「数値」関数から数値定数を、「文字列」関数から文字定数を、「ブール」関数から「TRUE 定数」か「FALSE 定数」を選んでブロックダイアグラム上に配置し、適当な値をこれらに書き込め。次に、制御器パレット上の「数値」欄から「数値表示器」を、「文字列&パス」欄から「文字列表示器」を、「ブール」欄から「LED ランプ」を選んで、フロントパネル上にそれぞれ配置せよ。各種定数から各種表示器へ配線を接続してみて、どのような型同士なら接続できるかを確認せよ。

**問題２－２：**

問題２－１で配線できなかった文字列と数値の間に、文字列/数値変換の関数を介在させて配線をもう一度行ってみよ。例えば、「123456789.123456789」の数値定数を、10 進文字列、指数文字列、工学形式文字列で表示させてみよ。また、数値表示器の「プロパティ」メニューで「形式と精度」を選択し、有効数字を変えて実際に表示が変化することを確認せよ。

## B)　繰り返し処理　　----- 同じ手順で繰り返し作業を行う場合に用いられる命令 -----

### ●　FOR ~ NEXT　　※LabVIEW では、「For ループ」という名前で登録されている.

ブロック内の処理を、指定した回数だけ繰り返して行う。回数を指定する N 端子 [N] には、処理を行う繰り返しの回数を入力する(今回は数値関数の中から「数値定数」を選んで数値を与えている)。**反復端子** [i] には、現在実行している処理が何回目であるかという値が返されている（指標 i は 0 から始まっていることに注意)。数値表示器をこの反復端子に配線して、実際に処理が繰り返し行われていることを確認せよ。


図 15：For ループの例.

### ●　DO ~ WHILE　/　WHILE ~ WEND

※LabVIEW では、「While ループ」という名前登録されている.

WHILE 文の条件式で設定した**条件**が満たされるまで、ブロック内の処理を繰り返し行う。**条件端子**には、ブール値[真(true)か偽(False)か)]を配線で接続して While ループの継続と停止を選択させる。反復端子には、For ループ同様、現在実行している処理が何回目であるかという値が返されている。ブール制御器からスイッチと LED 表示器等を取り出して下記のようにフロントパネルに配置し、配線を行って実際にスイッチ操作でループを止められることを確認せよ。

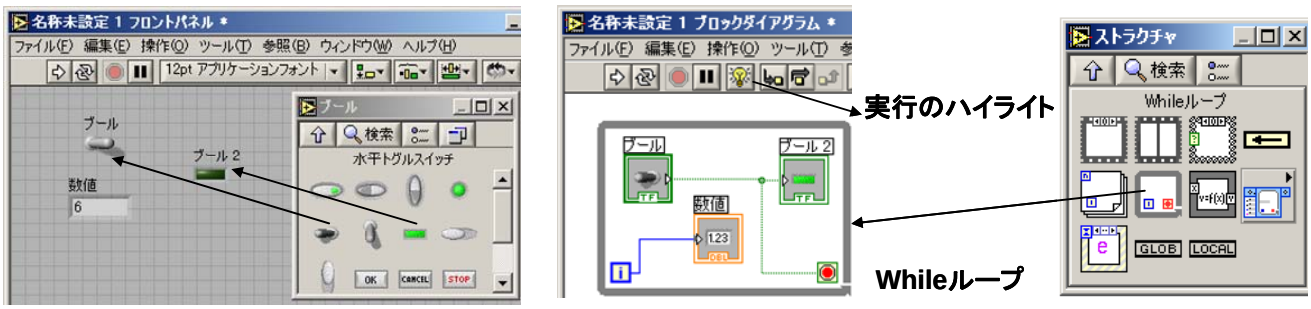



図 16：While ループの例.

図 16 の例は、条件が「真」の場合にループが止まるが、条件端子を右クリックして現れるメニューで「True の場合、継続」を選択すれば、これを条件が「偽」の場合にループを止めるように変更することも出来る。また、関数パレット内にあるブール関数のうち「Not」関数を、条件端子に繋いでも同様の変更が出来るので、試してみよ。

For ループや While ループにおいては、ループ内の命令が反復して実行されるが、ある回の実行結果をのちの回でも使用できるようにするための仕組みとして、「シフトレジスタ」が用意されている。各ループの左端か右端の輪郭線上から右クリックでショートカットメニューを出して、「シフトレジスタを追加」を選択すれば、左右両側にセットでシフトレジスタが追加される。

1）左側の端子  ：初期値の代入、および前の回の値を格納.
2）右側の端子  ：ループ内の計算結果の代入先、およびループ完了後の結果の取り出し.

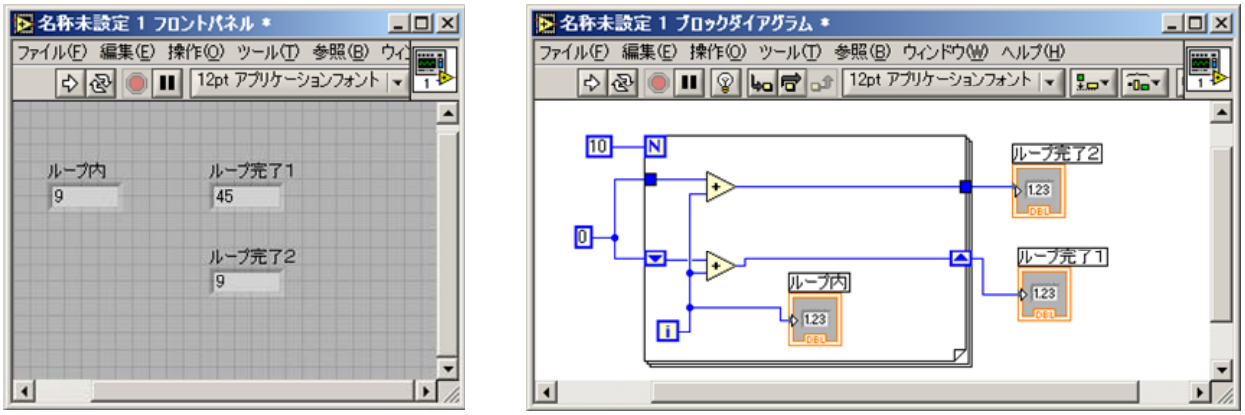



図 17：For ループにおけるシフトレジスタの使用例.

図 17 の例は、0 から 9 までの数字の総和を結果として表示するものである（答えは 45）。初期値を 0 として、0 から始まり N-1（この例では 10-1＝9）に至る反復端子  の値を、毎ループごとに加算してゆく際に、シフトレジスタを一時的な保管場所として使用している。シフトレジスタを使わず、単純に端子を接続した場合の結果（ループ完了 2）と比較してみよ。プログラム実行の際に、「実行のハイライトボタン」  を押しておくと、数値がどのように代入され移動してゆくかが可視化さ

れる。各自プログラムの推移を確認してみよ。

**問題２－３：**

For ループを用いて、初項 3、公差 5 の等差数列　3, 8, 13, 18, 23,…　の 100 番目の値を求めよ。

**問題２－４：**

For ループとシフトレジスタを用いて、1 から 999 までの奇数を全て足した値を求めよ。

**問題２－５：**

フィボナッチ数列の第 n 番目の項がどのような数字であるかを求めて表示するプログラムを書け。すなわち、n を数値制御器で与えて、結果を数値表示器で表すことになる。

フィボナッチ数列とはとは、最初の２項が $f_1 = 1$, $f_2 = 1$ で始まり、以降は順次、反復公式 $f_{n+2} = f_{n+1} + f_n$ を用いて定められる数列のことである。
※数列の最初の部分は、以下のようになる。
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597...

ヒント　シフトレジスタの左端子を右クリックして現れるメニューのうち、「要素を追加」を選択して、端子を一つ増やす。増やした端子には、前回の格納データではなく前々回のデータが格納されるので、この２つの端子を用いればよい。

**問題２－６：**

上記の問題で作成したプログラムで、隣り合うフィボナッチ数の比の値を表示させるようにせよ。n の値を増してゆくと、比の値は 1.618 に収束することを確認せよ。

$\lim_{n\to\infty} \frac{f_{n+1}}{f_n} = x$ とすると、

$$x = \lim_{n\to\infty} \frac{f_n + f_{n-1}}{f_n} = \lim_{n\to\infty} \left(1 + \frac{f_{n-1}}{f_n}\right) = \lim_{n\to\infty} \left(1 + \frac{1}{f_n / f_{n-1}}\right) = 1 + \frac{1}{x}$$

すなわち $x^2 - x - 1 = 0$ 。求める値 $x$ はこの２次方程式の解となる。この比の値は「黄金比」と呼ばれており、バランスが取れていると感じられる故、身近に多くの適用例がある。

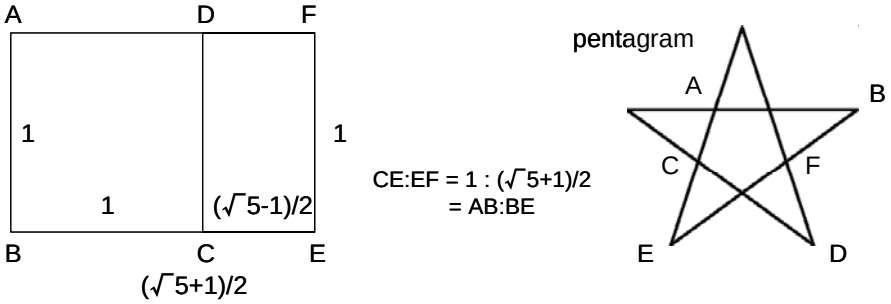



図 18：黄金比を示す幾何学図形の例.

## C)　条件分岐

- **IF ~ THEN**
- **CASE**　　※LabVIEW では、「ケーススストラクチャ」で登録されている.

プログラムが順番に実行されてゆく中で、時に、ある着目した変数の値に応じてそれぞれ異なった処理を行いたい場合がよくある(例えば変数「A」の値が正の場合には電球を点灯し、負の場合には電球を消す、など. )。こうした**条件に応じた分岐**が必要な場合に用いられるのが、一般に「IF ~ THEN」または「CASE」で知られている構文である。LabVIEW 言語では、「ケーススストラクチャ」で判別を行う。

先に例で示した和と積を表示させるプログラム(＊＊参照)を少し変更して、トグルスイッチにより和と差の結果を切り替えて表示させるプログラムを書くと、図 19 のようになる。フロントパネル上に配置したスイッチによって与えられる真偽値を、ケーススストラクチャのセレクタ端子  に配線し、各条件(True, False)における動作をそれぞれのページに記述することで、条件分岐処理を行っている。実際にこのプログラムを書いてみて、動作を確認せよ。

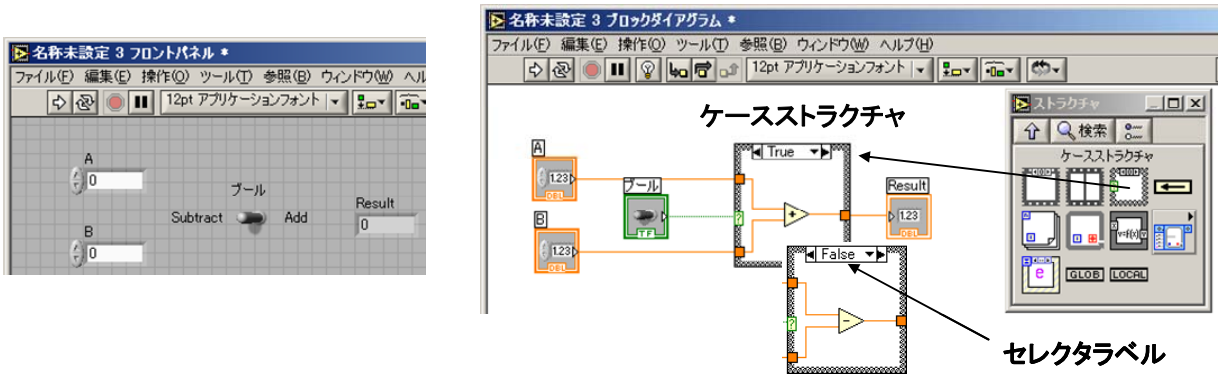



図 19：ケーススストラクチャの例.

ケーススストラクチャへの配線は、ブール値を与えるもの(スイッチやブール関数)に限らず、数値でもよい。その場合、入力される数値に応じた各場合における動作を、それぞれのケースに記述することになる。この選択に用いる値と各ページとの対応は、ケーススストラクチャの「セレクタラベル」で表されている。テキスト編集ツール  を用いてここに値を直接入力したり、すでにある値を書き換えたりして、選択条件とケーススストラクチャの対応を修正することが出来る。

**問題２－７：**

図 19 のトグルスイッチ（ブール代数）部分を、0, 1, 2, 3 の 4 つの整数値のみを入力する数値制御器に変え、この制御器の値が 0 ならば和、1 ならば差、2 ならば積、3 ならば商を答えとして文字列表示器「Result」に出すようなプログラムを新たに書け。

**問題２－８：**

３つの数字 a, b, c を読み込み、a, b, c を三角形の辺の長さとみなしたとき、これで三角形が出来るかどうかを判別するプログラムを書け（例えば出来るなら LED が点灯する、出来ないなら消灯のまま、など)。また、三角形が書ける場合にはその面積を計算して出力せよ。

ヒント　$a > b,c$ なら、$a < b + c$ のとき三角形ができる。また、$s = (a+b+c)/2$　とした場合、三角形の面積は、ヘロンの公式 $S = \sqrt{s(s-a)(s-b)(s-c)}$ で与えられる。

## D)　配列変数の扱い　　----- 複数のデータ要素をひとまとめにして扱う -----

時系列データの一時格納および保存のためには、番号で指標づけした変数を一般的に用いる。例えば、下記のような測定結果

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 指標 | 0 | 1 | 2 | ..... | 98 | 99 | 100 |
| 時間 | 0.15 | 0.16 | 0.17 | ..... | 0.248 | 0.249 | 0.250 |
| 光強度 | 5.325 | 5.310 | 5.250 | ..... | 3.210 | 3.201 | 3.188 |

は、時間と強度のデータそれぞれを、**指標**を用いて Time(i), Intensity(i)のようにまとめて表すようにしておけば、後で参照するときに非常に便利である。配列は 1 次元に限らず、2 次元、3 次元というように次元を増やすことができるので、例えば温度、電流強度、磁場強度のように更に測定するものが増えても、共通する指標によってそれぞれのデータ系列を時間と対応付けられる。

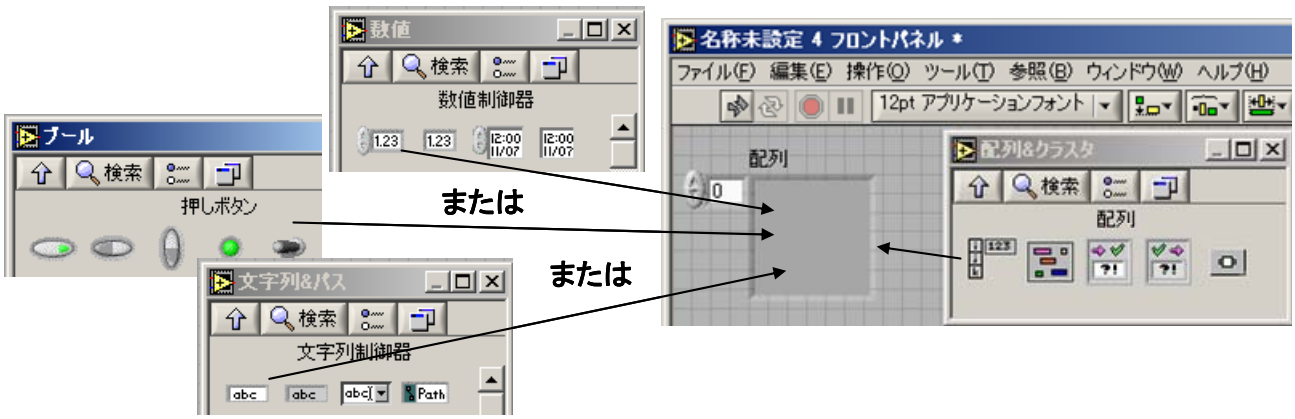



図 20：配列変数の作成方法.

配列の要素には、上記 A）の様々な変数が使用される。制御器パレットの「配列&クラスタ」欄から**配列シェル**  を選択してフロントパネル上に配置し、その中にこれら変数の制御器(表示器)を挿入することで、配列変数は完成する。これにより**配列変数の型**も決定するので、配線を接続して

プログラム中で扱えるようになる。配列の個々の要素へのアクセスは**指標**を通じて行われ、その値は０から始まり最大は $2^{31}$ まで原理的に可能である。関数パレット内の**配列関数**をブロックダイアグラム上に配置し、配列関数と配線で接続することにより、配列変数をまとまった形で処理することが、また個々の内部要素を操作することが出来る。

代表的な配列関数を幾つか以下に挙げる。

１）「部分配列」関数  :

指定した指標から始まり、指定した長さ分の要素数を含む配列の一部を返す。
サンプルプログラム「Array Subset Demo.vi」※

２）「指標配列」関数  :

入力する n 次元配列の要素または部分配列を返します。
サンプルプログラム「2D Array Index Demo.vi」※

３）「配列連結追加」関数  :

複数の配列を連結、または n 次元配列に要素を追加します。
サンプルプログラム「Practice with Arrays Done.vi」※

※これらのサンプルプログラムは、「C:¥Documents and Settings¥User¥My Documents¥Learning Directory¥Chapter 6」のフォルダに在る。

各自サンプルプログラムを開いて配列の値を一覧した後に、プログラムを実行させて、各配列関数の働きを確認せよ。

**問題２－９：**

2 行 2 列の数値制御器を埋め込んだ配列変数をフロントパネルに配置し、それぞれの数値を要素 a，b，c，d とする行列 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ を考える。この配列から要素を抽出して計算することにより、この行列の逆行列を求めて、数値表示器を埋め込んだ配列変数に表示するプログラムを作成せよ。

ヒント　正方行列 A において、$AB = BA = E$ （$E$：単位行列）を満たす行列 B を A の**逆行列**といい、これは $A^{-1}$ で表わされる。2 行 2 列の正方行列 $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ の逆行列は，$ad - bc \neq 0$ のとき存在し、$A^{-1} = \dfrac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$ で与えられる。

例：$A = \begin{pmatrix} -1 & -2 \\ 2 & 5 \end{pmatrix} \quad \Rightarrow \quad A^{-1} = \begin{pmatrix} -5 & -2 \\ 2 & 1 \end{pmatrix}$

## E)　クラスタ変数の扱い

上記のように、配列変数を使用すれば、複数のデータ要素ひとまとめにして扱うことが出来るが、これらデータ要素の形式は統一されている必要があった。これに対し、**異種のデータ要素**をまとめてグループ化できるようにしたものが、**クラスタ変数**と呼ばれるものである。

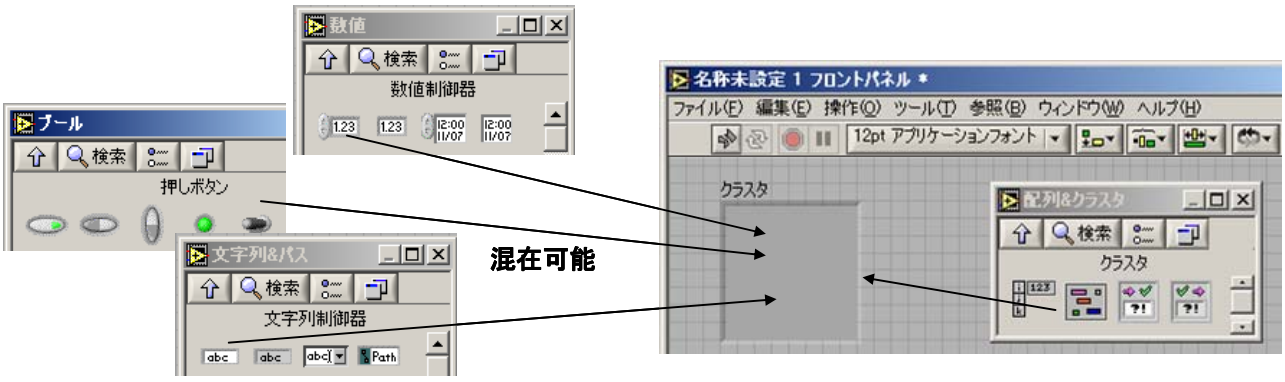



図 21：クラスタ変数の作成方法.

クラスタの要素には、上記 A）の様々な変数を混在できる。制御器パレットの「配列&クラスタ」欄から**「クラスタ」**  を選択してフロントパネル上に配置し、その中にこれら変数の制御器(表示器)を挿入することで、クラスタ変数は完成する。また、関数パレットのクラスタ関数の中から「バンドル」関数の VI  を選択してブロックダイアグラム上に配置し、各種変数からの配線を接続すれば、クラスタ変数出力が得られる。クラスタ関数の中の「バンドル解除」の VI  を選択してブロックダイアグラム上に配置し、これらクラスタ変数からの配線を接続すると、配線に含まれるこれら各種変数を、それぞれ分離して取り出すことが出来る。以下のサンプルプログラムを参照して、各自クラスタ変数の概念を確認せよ。

1）「バンドル」関数のサンプルプログラム
「Cluster Bundle Demo.vi」,「Cluster Element Replace.vi」※

2）「バンドル解除」関数のサンプルプログラム
「Cluster Unbundle Demo.vi」※

※これらのプログラムは、配列関数の場合と同様、「C:¥Documents and Settings¥User¥My Documents¥Learning Directory¥Chapter 6」のフォルダに在る。

## F)　サブルーチン

短いプログラムは別にして、通常測定に用いられるような長いプログラムは、まとまった一つの処理をするサブユニットが幾つか組み合わさって作られている。こうしたプログラム上のサブユニットのことを、「**サブルーチン**」と呼び、LabVIEW 言語においてもこの構文が使用できる。　LabVIEW においては、演算やファイル操作等を行う各関数に対応した部品を、ブロックダイアグラム上に配置することによってプログラムが構築されるが、各個人が作成して入出力端子を持つ一つの部品として保存しておいた VI プログラムを、これら既存の関数部品と同様に配線して使用することができるのである。他のプログラムから呼び出して使用することが出来るこのプログラム部品は、「**サブ VI**」と呼ばれている。ここではサブ VI の一例として、模擬温度信号を実数値で出力するプログラム「Digital Thermometer.vi」※を挙げる。

※VI「Digital Thermometer.vi」の本体は、「C:¥Documents and Settings¥User¥My Documents¥Learning Directory¥activity」のフォルダに在る。

「Digital Thermometer.vi」では、フロントパネルのスイッチにより、華氏か摂氏かどちらかに選択された単位で、模擬温度が生成され、その結果がバー表示される(図 22 参照)。この VI をサブ VI として使用する際に表示されるアイコンの図柄(ここでは Temp の文字と温度計)は、フロントパネル右上のアイコン部分を右クリックして現れるメニュー「アイコンを編集」を選択し、表示される編集画面で任意に変更できる。

このプログラムのブロックダイアグラムを見ると、ここでは「Volt Read」アイコンから出力される模擬的な華氏温度の値を、スイッチの切り替えに対応した Case ストラクチャで華氏のまま、もしくは摂氏に変えて表示している※。

※　華氏 F 度と摂氏 C 度の間の変換公式は、$C = \frac{5}{9}(F - 32)$ である)。

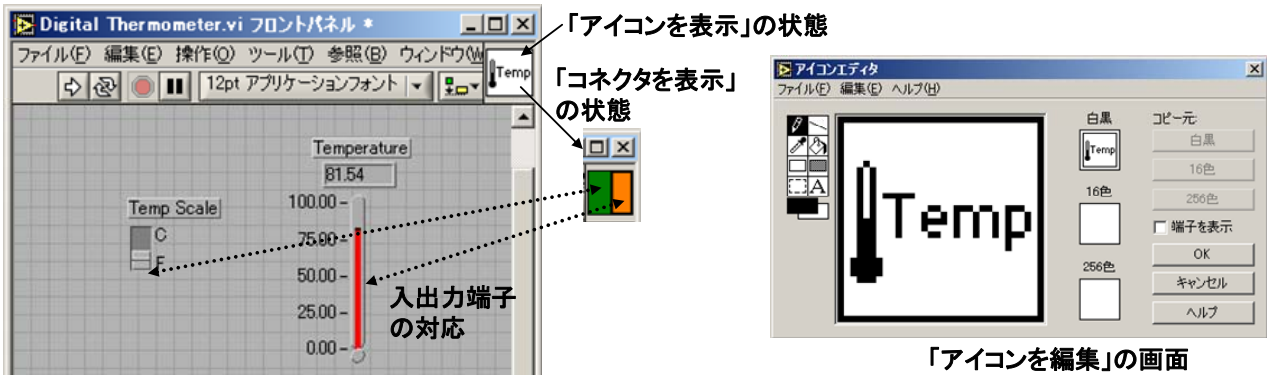



図 22：「Digital Thermometer.vi」のフロントパネルとコネクタ表示およびアイコン編集画面.

このVIを別のVIから部品として呼び出せるようにするためには、入力(華氏と摂氏の切り替えスイッチ)と出力(模擬温度の値)の端子を、それぞれこのVIの入力／出力値として登録する必要がある。これは、フロントパネル右上のアイコン上で現れるメニュー「コネクタを表示」を選択し、現れたコネクタとこれらの制御器(または表示器)端子とを配線で接続することによって完了する。実際に端子の接続状態を確認してみよ。なお、端子の色は配線と同様に接続されている部品の持つ属性に応じて変化する。

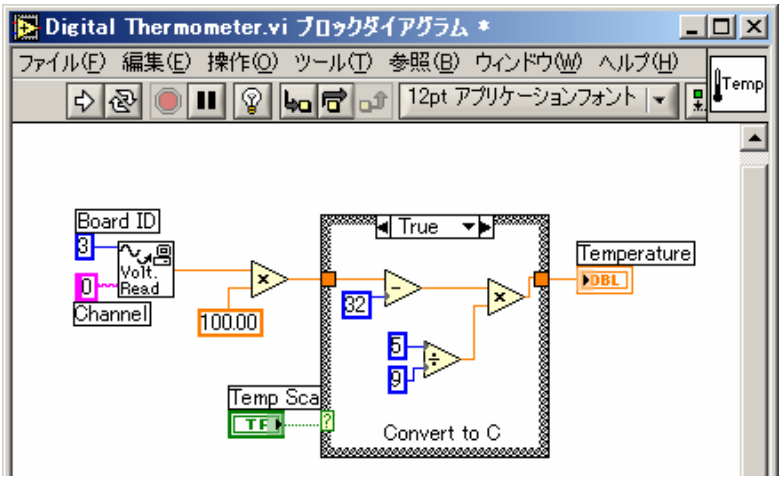



図 23：「Digital Thermometer.vi」のブロックダイアグラム.

**問題２－１０：**

実はこのブロックダイアグラム上の「Volt Read」アイコンは、別のサブVI「Demo Voltage Read.vi」に対応しており、内容が確認できる。このアイコンをダブルクリックしてフロントパネルおよびブロックダイアグラムを開き、どのようにして模擬温度の値が出力されているかを確かめよ。

## G)　グラフ・チャートの作成　（結果の表示）

測定中および測定後のデータを、視覚的に把握したい場合に用いられるのが、「グラフ」や「チャート」である。LabVIEW には、2 次元および 3 次元データの表示を行う様々なグラフ・チャートが用意されているが、時系列データに代表される 2 次元の計測データの表示には、以下の 3 種のチャート・グラフが使用される。

**1）波形チャート**：　送られてくるデータを次々とプロットしてゆく表示器。
**2）波形グラフ**：　等間隔に分布している一連の点について、一価関数の値をプロットするときなどに使用する。
**3）XY グラフ**：　等間隔でないデータポイントをプロットしたり、２つの独立変数の関係をプロットしたりするのに用いられる。

ここでは、まず先程の**サブ VI**「Digital Thermometer.vi」を用い、ここから出力される模擬信号を「波形チャート」にてフロントパネルに表示するサンプルプログラム「Single and Multiple Charts.vi」を例として挙げる。

### 波形チャート

「Single and Multiple Charts.vi」のブロックダイアグラムは、図 24 のようになっている。While ループ内にあるサブ VI「Digital Thermometer.vi」から出力される模擬温度信号の実数値が、波形チャートへの入力として用いられている。

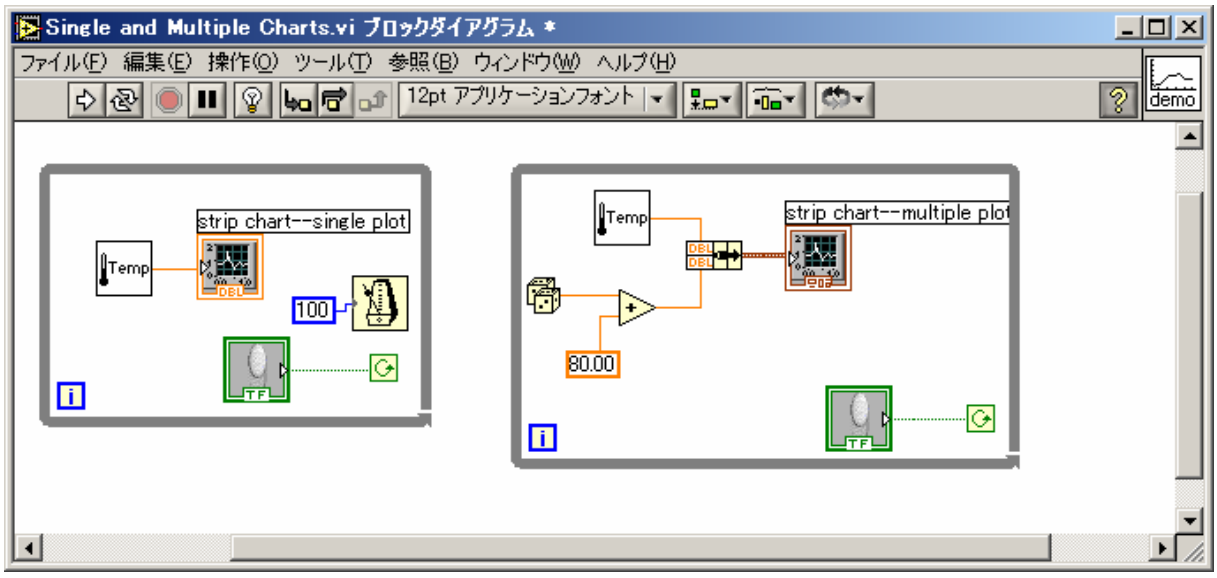



図 24：「Single and Multiple Charts.vi」のブロックダイアグラム.

対応する図 25 のフロントパネルにおいて、左側のチャートが、スイッチが押されるまで単一の温度計測値をプロットするもの、右側のチャートが、それに加えて乱数値を入力元とした 2 系列のデータをプロットするものである。右側のチャートでは、関数パレット内の「クラスタ」欄に在る「バンドル」関数を使用し、２つのデータをまとめて入力に与えることで、2 つの波形の同時表示を実現している。

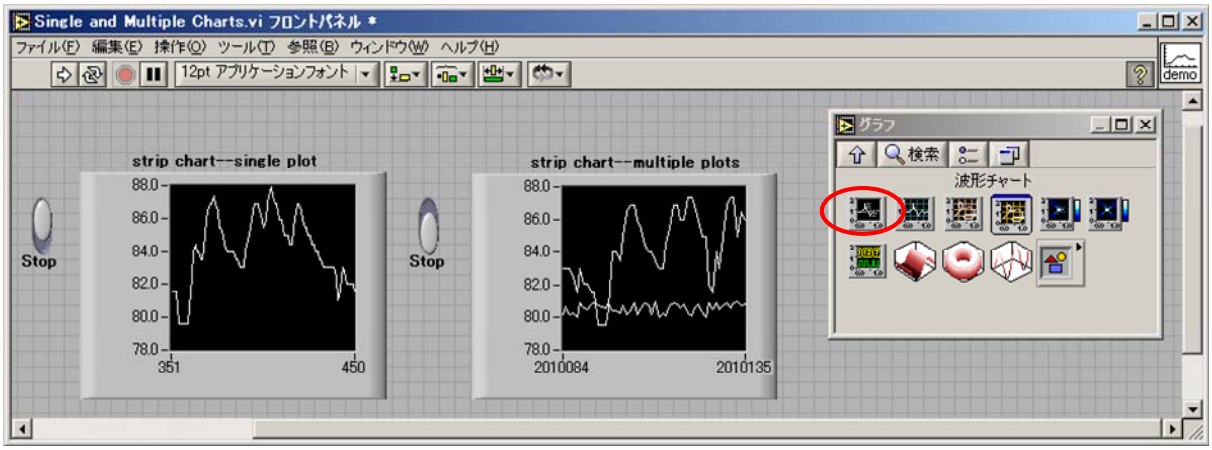



図 25：「Single and Multiple Charts.vi」のフロントパネル.

※ここで使用するプログラム「Single and Multiple Charts.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 7」に置いてある。

## 波形グラフ

一方「波形グラフ」は、既に取得済みのデータを表わすための表示器である。取得済みのデータは通常、配列変数に格納されており、これを配線で接続することによってデータが波形グラフにプロットされる。これまで同様、サブ VI「Digital Thermometer.vi」を用いて、波形グラフを用いた結果の表示を試みよう。

「Digital Thermometer.vi」から For ループを用いて模擬温度値を 10 回出力させ、これを配列の形で波形グラフへ描画させるのが、図 26 の左側のプログラムである。単純に接続しただけの場合には、図 27 の実行結果に見られるように、横軸の値には自動的に 0 から始まる整数値が割り当てられる。これに対し、図 26 の右側に示したように、横軸の初期値および増分値を明示し、これらを配列データと共にクラスタ変数の形で波形グラフに対して入力させれば、実行結果（図 27）に見られるように、横軸の表示を思うように変えることが出来る。

波形グラフには､複数のグラフを同時に表示させることが出来る｡図 16 の右側のプログラムでは、2 系列のデータ（模擬温度出力と乱数の値）を、関数パレットの「配列」欄に在る「配列連結追加」関数を用いてクラスタの 2 次元配列とし、これを波形グラフに配線して同時表示を実現している。

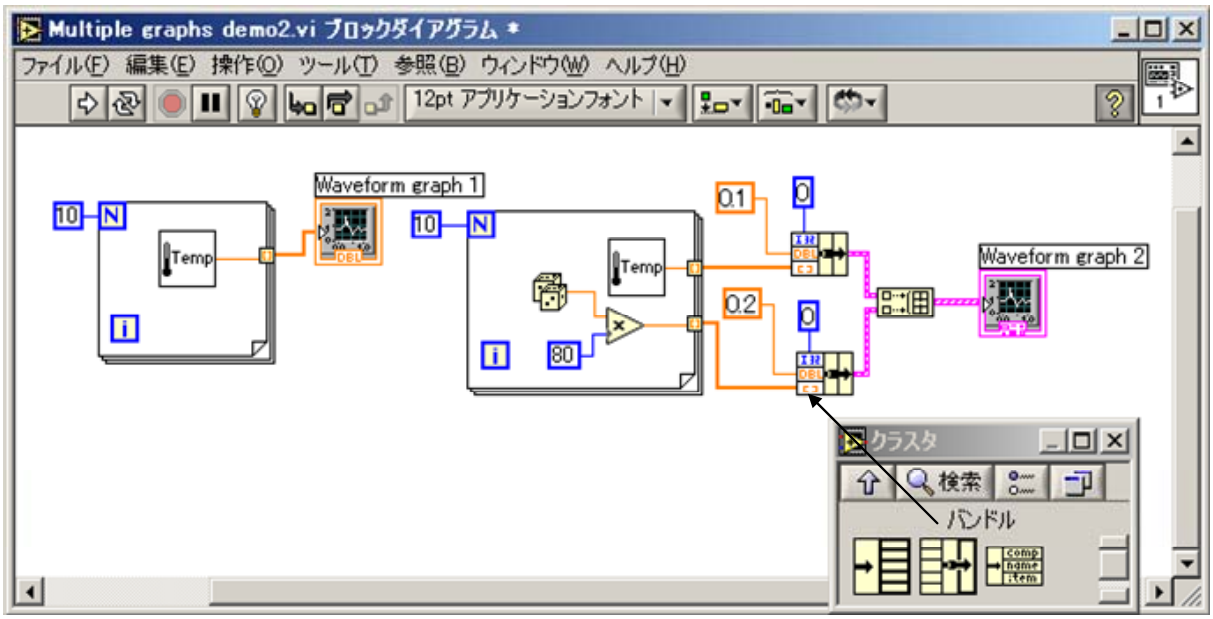



図 26：波形グラフ描画のプログラム例（ブロックダイアグラム）.

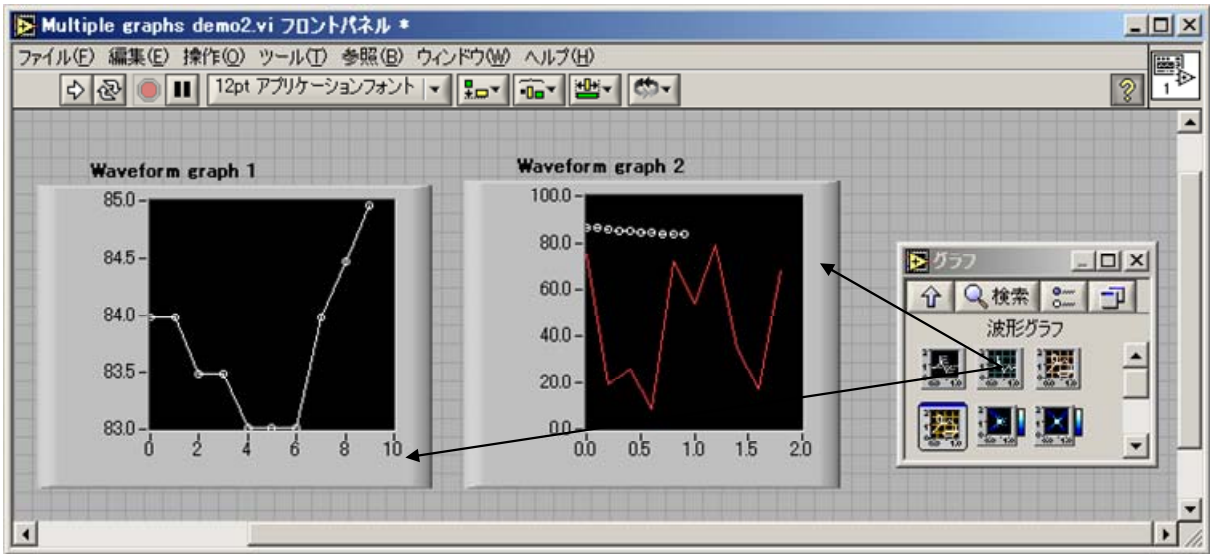



図 27：波形グラフ描画のプログラム例（フロントパネル）.

※ここで書いたプログラム「Multiple graphs demo2.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 7」に置いてある。

## XY グラフ

「XY グラフ」も、「波形グラフ」同様に、取得済みのデータを主として表わすための表示器である。等間隔でない抽出で得られたデータをプロットしたり、２つの独立変数（例えば x, y）の関係をプロットしたりするには、このグラフが向いている。XY グラフに受け渡すデータとしては、X,Y の各１次元データを「バンドル」関数でまとめたものを用意することになる。

図 28 に XY グラフの描画例を示す。For ループを用いて $x$ の値を作り、これを代入した $\sin(x)$ の値を Y 軸の値として、最終的にこれらをバンドルしたものを波形グラフへ描画させている（フロントパネルおよび図 29 のブロックダイアグラムの左側の For ループ）。

※ここでは同時に、２次元配列に x および y の値を格納している。

配列変数に入った２次元データを XY グラフに表示させるという、実験や解析でよく用いられる XY グラフの使用例を示したのが図 28 のフロントパネルおよび図 29 のブロックダイアグラムの右側の For ループの部分である。ローカル変数を使用して、左側の For ループで作成した配列変数の分身を作り、そこから各行・各列の要素を For ループによって抽出し、これを XY グラフへ表示している。

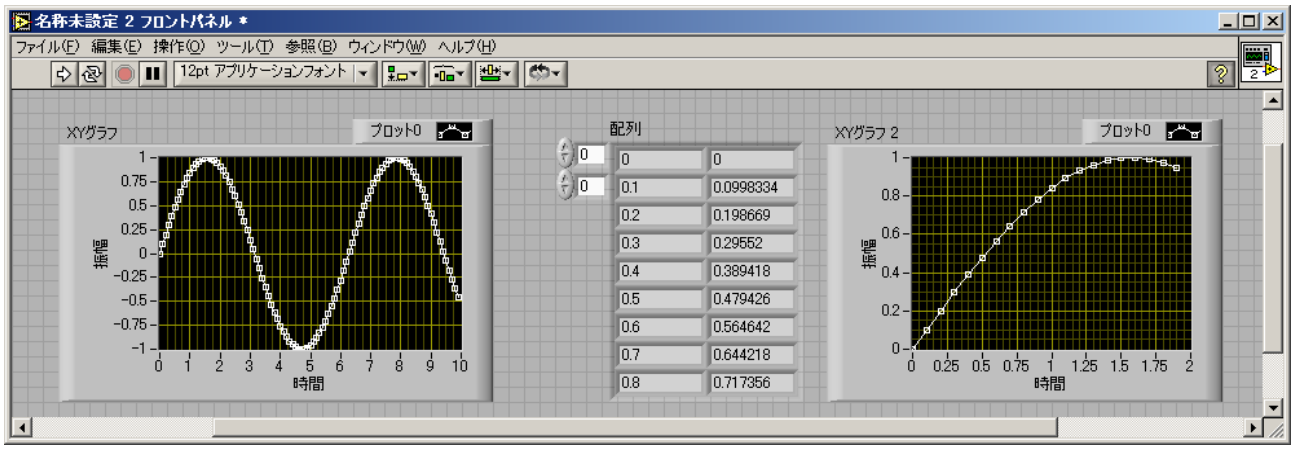



図 28：XY グラフ描画のプログラム例（フロントパネル）.

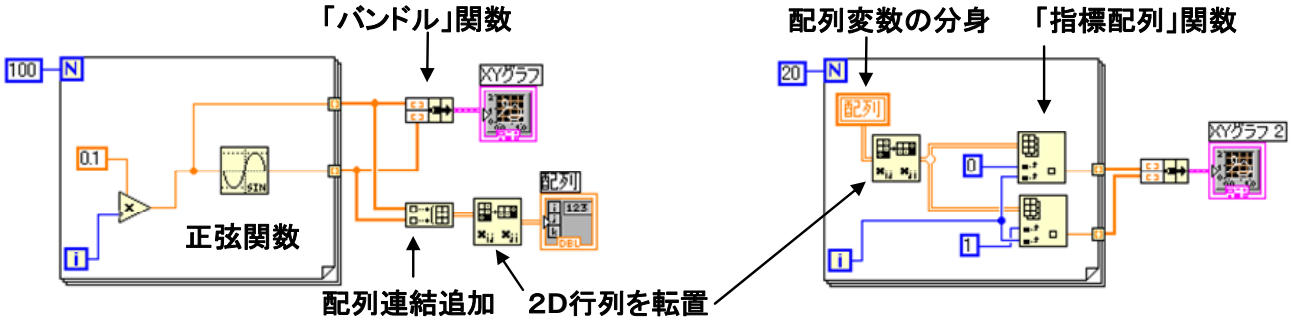



図 29：XY グラフ描画のプログラム例（ブロックダイアグラム）

上記のような制御器パレットの「グラフ」表示器欄にある「XY グラフ」項目を用いたグラフ作成法とは別に、「Express XY グラフ」項目を用いて XY グラフを作成する方法もある。サンプルプログラム「XY Graphs Demo.vi」や「Time Value of Money.vi」を例として、「Express XY グラフ」を用いた XY グラフの表示のさせ方を各自確認せよ。

※サンプルプログラム「XY Graphs Demo.vi」および「Time Value of Money.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 7」に置いてある。

## H)　ファイルの読み込みと書き出し　※LabVIEW では、各種「ファイル I/O」関数を用いる.

測定結果を後で解析・表示出来るようにするには、測定後に配列変数に記録されているデータを、ファイルに保存しておく必要がある。ディスク上のファイルにこのデータを書き込んだり、ディスク上のファイルからデータを読み取ったりする操作を、**ファイル I/O**(Input/Output)と呼ぶ。通常のプログラム言語においては、それぞれのプログラム言語に固有な外部出力命令において出力ファイル名および出力形式を指定し、そこを対象として記録したい変数を出力することによってデータのファイルへの記録を行うようになっている。LabVIEW 言語においても、ファイル I/O 用の関数がいくつか用意されており、状況に応じて適切な入出力形式を選択できるようになっている。

### 1）データをファイルに書き込む

PC と測定装置との間の通信やシミュレーションを通じて、LabVIEW プログラムから出力されるデータを、後から参照するもしくは他のプログラムで使用する際には、データを文字列としてファイルに書き込むか、エクセル等の表計算ソフトで取り込める**スプレッドシート形式**でファイルに書き込むかの二つの方法が主として用いられる。特に、電圧対電流の関係や時系列データのような、相関する二組以上のデータを記録したい場合には、スプレッドシート形式での書き込みが適している。

では、実際にファイルにデータを書き込むプログラムを書いてみる。ここでは記録するデータとして、模擬的に温度出力を行う**サブ VI プログラム**として先程の「Digital Thermometer.vi」を用い、得られた結果を一つの**文字ファイル**に保存するものとする。

※ここで使用するプログラム「Write Temperature to File.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 8」に置いてある。

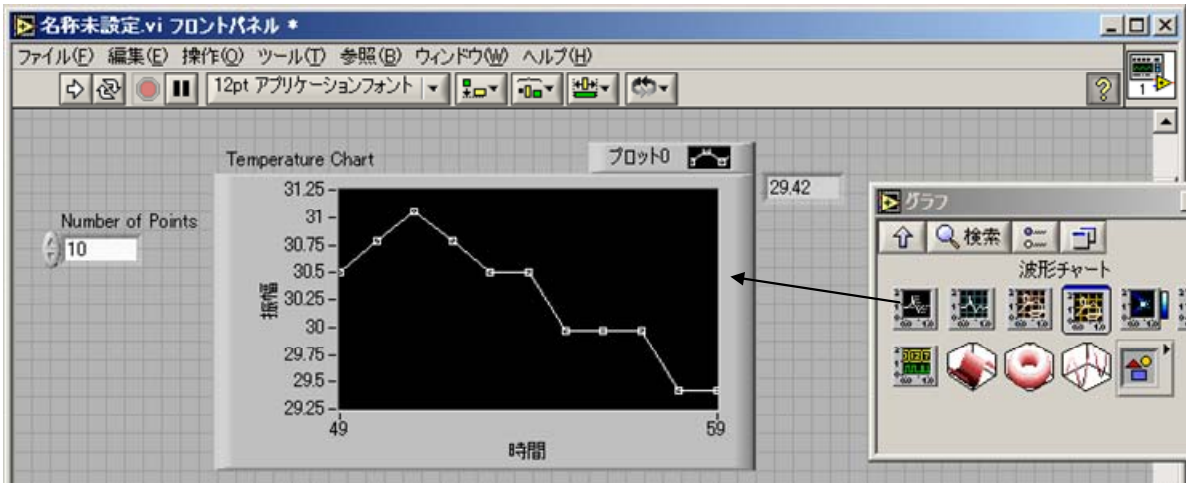



図 30：温度計出力からのデータをファイルに書き込むプログラムの例(フロントパネル).

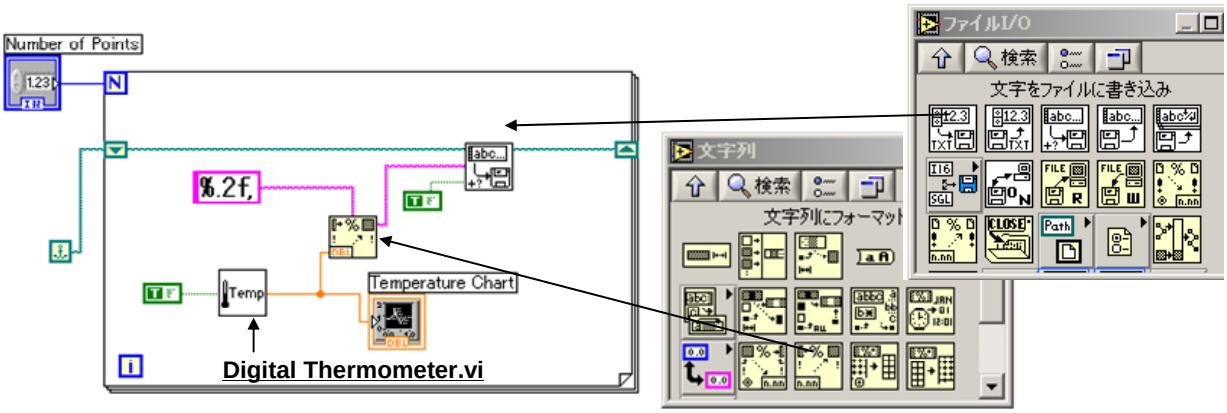



図 31：図 30 の対応するブロックダイアグラム.

図 30 がフロントパネル、図 31 が対応するブロックダイアグラムである。このプログラムは、「Digital Thermometer.vi」から温度数値として与えられる模擬信号を、波形チャートに表示しつつ文字列に変換し、この文字列を一つのファイルへ上書きしながら保存する構成になっている。ここでは、ファイルへの書き込みは、関数パレット内の「ファイル I/O」欄に在る「文字をファイルに書き込み」の VI が使用されており、ファイルの保存場所および上書きモードの ON/OFF 指定(今回は ON)が、文字列データと共にこの VI へそれぞれ配線されている。また、記録するデータの形式(数値の桁数やその小数点の位置)は統一されていることが望ましいため、関数パレット内の「文字列」欄に在る「文字列にフォーマット」の VI を用いて、[浮動小数点表記で小数点以下 2 桁まで表示]という形式にしている(フォーマットの詳細は、ヘルプの「**形式指定子**」項目を参照のこと)。

プログラムをスタートさせると、ファイル保存場所を聞いてくるので、「C:¥Documents and Settings¥User¥My Documents」やデスクトップなど適当な保存場所とファイル名を指定せよ。取得データ数は、数値制御器「Number of Points」であらかじめ指定しておく。実行後に保存されたファイルの内容を、エディタ(NotePad や WordPad など)で開いて確認してみよ。コンマで区切られた一連の数値データが記録されているはずである。

次に、測定温度の時系列に沿った変化を記録するために、スプレッドシート形式での書き込みを行ってみる。ここでは、関数パレット内の「時間&ダイアログ」欄にある「経過時間」VI を使用して測定開始時からの時間を取得し、これと温度の値をまとめた 2 列のデータをファイルに上書き保存を行う。「Write Temperature to File.vi」（図 30）を基にしてプログラムを作成すると、そのブロックダイアグラムは図 32 のようになる。元から変更された点は、以下の 3 点である。

1）「経過時間」VI を使用し、測定開始からの経過時間を秒単位で出力させる。
2）関数パレットの「配列」欄にある「配列連結追加」VI を使用して 2 つのデータをまとめる。
3）1 秒単位で取り込むために「ミリ秒待機」VI に整数値「1000」を入力し、For ループの進行を遅くする。

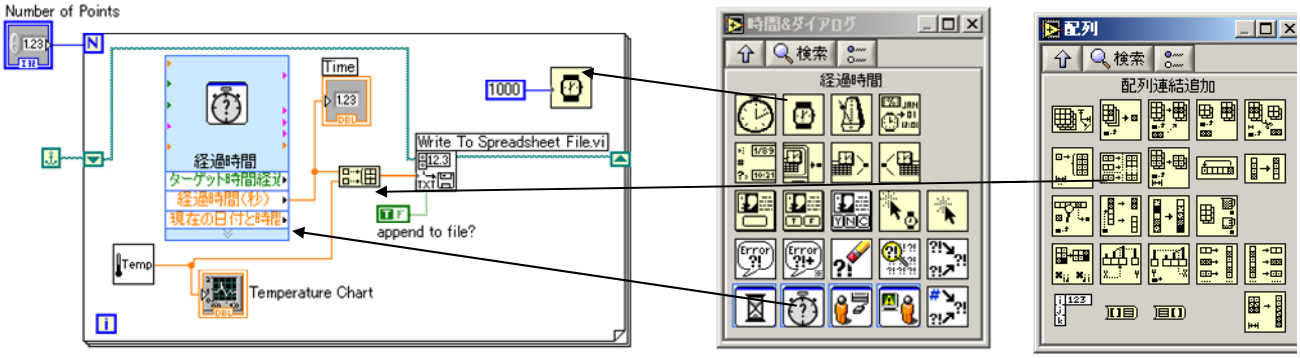



図 32：スプレッドシート形式でのファイルへのデータ書き込みプログラム.

ここで「経過時間」VI は、Ver.7 以降の LabVIEW に付加された新しい VI 形式（**Express VI**）を用いている。Express VI は、測定作業に使用する VI 群をひとまとめにしたものであり、ダブルクリックして現れる別窓内で対話的に構成の変更が行えるので、これを用いることで一般的な測定プログラムは、配線作業を少なくして手早く作成できるようになる。

**問題３－１：**

「配列連結追加」VI で二次元配列になった「時間」と「温度」のデータは、1 秒毎に 2 列の数値データ形式でファイルに保存される。実際にプログラムを動作させ、温度の時間変化をデータとして保存せよ。また、保存されたデータが 2 列の数値で構成されていることを確認し、XY グラフに表示させてみよ。

※ここで書いたプログラム「Write Temperature to file2.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 8」に置いてある。

**２）ファイルからデータを読み込む**

記録したデータを参照する、もしくは他のプログラムで使用する際に必要なデータファイルの読み取り手続きを、１）で記録した文字列およびスプレッドシートのファイルからのデータを読み出すことで、実践し確認してみよう。

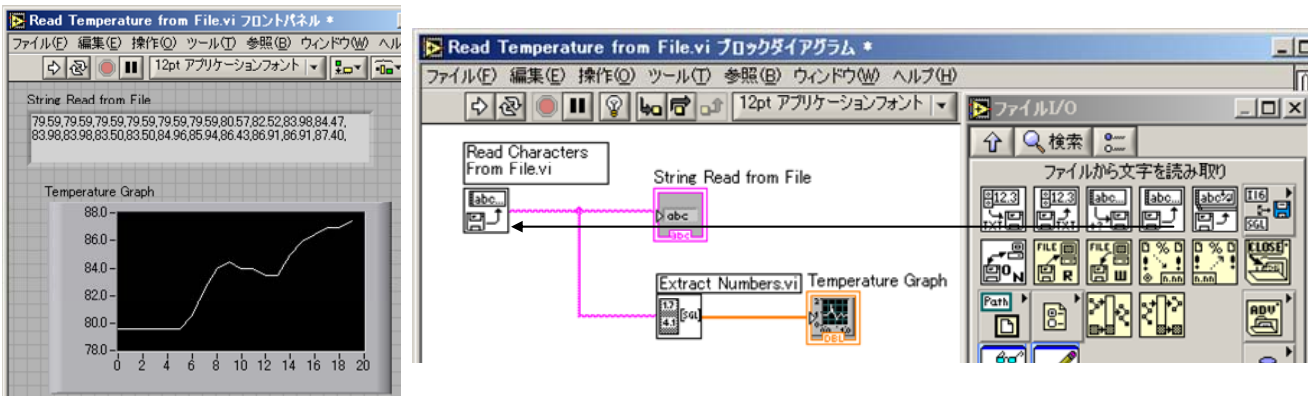



図 33：文字列ファイルからのデータの読み込み.

字を読み取り」VI を用いる。得られた文字列形式のデータを「Extract Numbers.vi」なる Vi で数値配列に変換し、これを波形チャートに表示させるプログラムを描いたのが、図 33 である。
各自、先程保存した文字列データファイルに対しプログラムを実行させて、文字列表示器と波形チャートに、それぞれデータファイルから読み取られた結果が表示されることを確認せよ。

※ここで使用するプログラム「Read Temperature from File.vi」および「Extract Numbers.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 8」に置いてある。

一方、スプレッドシートファイルからのデータの取り込みは、関数パレットの「ファイル I/O」欄に在る「スプレッドシートファイルから読み取り」VI を使用する。関数パレットの「配列」欄に在る「配列から削除」VI を使用して、得られた 2 次元配列データから時間と温度の 1 次元データを取り出し、それぞれ配列に格納すると同時に XY グラフで描画するプログラム「Read Spreadsheet2.vi」の実行画面およびブロックダイアグラムを、図 34, 35 に示す。

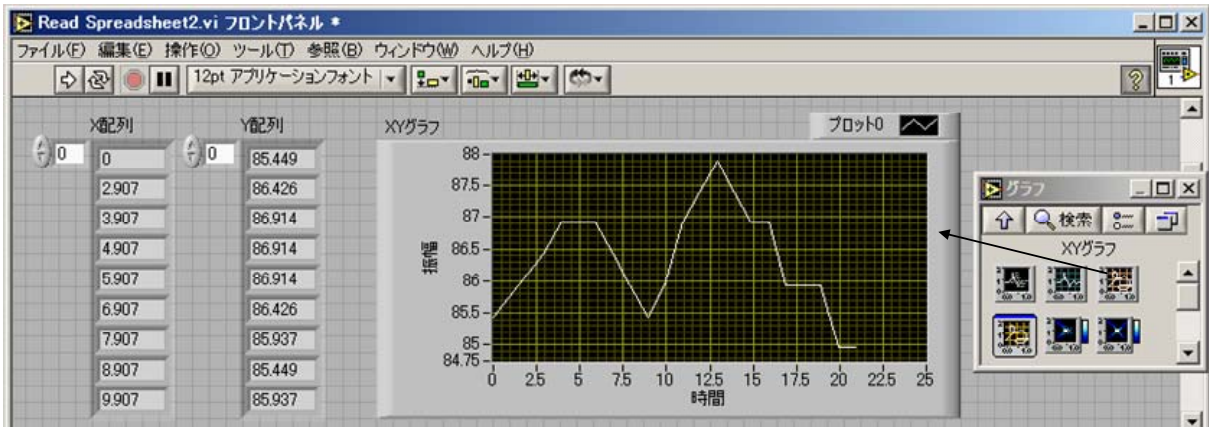



図 34：スプレッドシートファイルからの読み込みプログラムのフロントパネル.

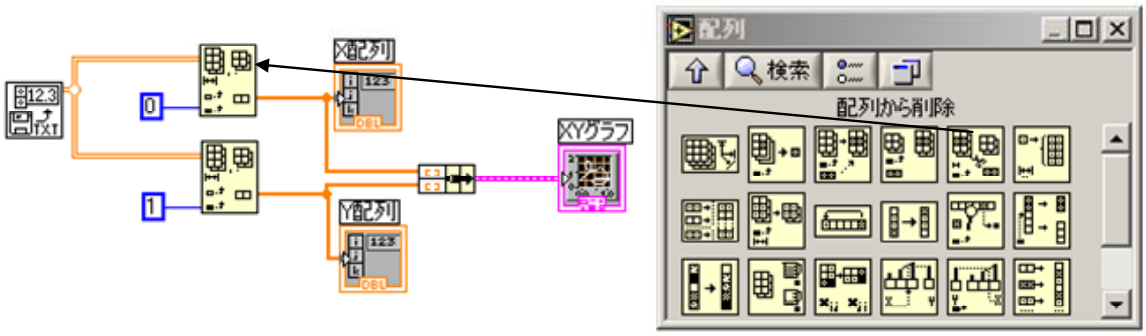



図 35：スプレッドシートファイルからの読み込みプログラムのブロックダイアグラム.

※ここで使用するプログラム「Read Spreadsheet2.vi」は、「C:¥Documents and Settings¥User¥My Documents ¥Learning Directory¥Chapter 8」に置いてある。

**問題３－２：**

各自、先程保存したスプレッドシートデータファイルに対して上記プログラムを実行させ、X 配列および Y 配列にそれぞれデータファイルから読み取られた時間と温度のデータが格納されていることを確認せよ。

## Ⅰ) 移動

● GOTO　　※LabVIEW では、該当命令なし.

プログラムは、そこに書かれている命令を、最初から順に実行していくことで進行する。しかし時には、その流れを無視して、プログラム中のある命令部分に直接飛んで、そこで処理を行いたい場合がある。この際に用いられる命令が、「移動」命令である。Basic や Fortran では、各命令文の文頭に行番号を設け、その番号を指標として「Goto （行番号)」命令を行うことで移動が為される。単純な命令を行う文へ移動するのみならず、一連の処理をまとめたブロックとして、または引数を与える関数として設置されたサブルーチンへと処理を移す際にも、この移動命令は使用される。

これに対し LabVIEW では、信号の流れが配線によって明示的に現されており、この配線上の信号の流れがそのままプログラムの進行に対応している。従って、プログラム上の配線自体がこの移動命令をも与えていることになる。

# 第四回　プログラミング解説 3　----- 追加項目と、これまでの内容の復習 -----

## その他、LabVIEW プログラミングで用いる命令群

### J) 実行タイミングの制御

LabVIEW 言語には、VI プログラムの実行タイミングを制御したり、実行時間を測定したりするための専用の関数や Express VI が用意されている。

こうしたタイミング制御で用いる代表的な関数を以下に挙げる。これらの関数は、関数パレットの「時間&ダイアログ」欄内にあるので、ヘルプを参照しつつ各自確認せよ。

1）「ティックカウント(ms)」関数  :

LabVIEW プログラムは、PC の内部クロックに対応した現在の時刻を、ある時刻を基準として、ミリ秒単位でモニタしている。この関数はその現在のミリ秒の値を返す。

2）「待機(ms)」関数  :

入力端子で指定したミリ秒の間、プログラムの実行を止める。

3）「次のミリ秒倍数まで待機」関数  :

入力端子で指定したミリ秒の値の倍数が経過するまで、プログラムの実行を止める。これにより、プログラムがモニタしている時間を基準にして、プログラムの実行タイミングをある程度同期させることが出来る。

計測器を制御するプログラムにおいては、これらの関数をブロックダイアグラム内に配置して、計測機器との通信や各種命令を実行する際の前後関係、およびタイミングの調整をすることが、よく行われる。

図 36 の左側に、上に挙げた関数を用いたプログラムの例を示す。ここで、For ループによって繰り返されるプログラムの進行を律速するのは、500 ms の時間単位で待機をかける「次のミリ秒倍数まで待機」関数である。このプログラムを実行した結果が図の右側であるが、実際に LabVIEW プログラムがカウントしている現在のミリ秒の値が、「ティックカウント」および「次のミリ秒倍数まで待機」関数から出力されていること、そしてその値が指定したミリ秒「500 ms」の倍数になっていることが分かる。すなわち、この For ループ内に何かプロセスを行う関数を配置すれば、そのプロセスは 500 ms の時間刻みに従って実行されることになる。

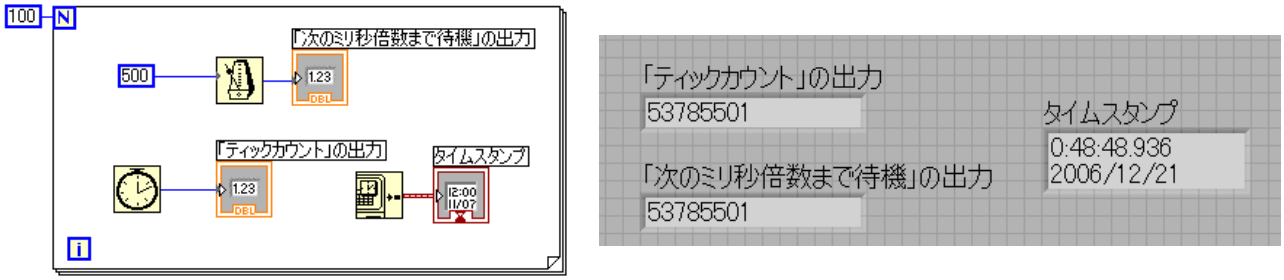



図 36：「次のミリ秒倍数まで待機」関数でループの実行速度を制限したプログラムとその実行結果.

**問題４－１：**

このプログラムの For ループ内に、ループ内の反復端子を利用した計算（ 例えば exp(i+1) ）を入れ、その結果を反復端子の値を指標とした 1 次元配列 Y に代入する。また同様に、ティックカウントの値を X 配列に代入する。これら２つの配列の数値をスプレッドシートファイルに保存するプログラムを書け。そして、このデータを読み出して XY グラフに表示させるプログラムも作成せよ。

## K) シーケンスストラクチャ

FORTRAN や C 言語など、書き下し型のプログラミング言語の殆どでは、プログラムに記述されている命令文の順序によって、基本的な実行の流れが決められている（これを制御フロー型という）。

一方、LabVIEW 言語においては、プログラムの要素（各制御器や演算子のアイコン）は、入力配線からのデータ入力がすべて揃った時点ではじめて実行されることになっている（これをデータフロー型という）。データを処理し終えてまた、次の処理ブロックへと次々にデータを受け渡してゆくこの流れが、命令の実行順序を与えることになる。したがって、プログラムによっては、併置した複数の命令群を順に実行させたくても、データの流れの順番が相前後してしまい、順序を守れなくなることが起こる。このような場合には、プログラム進行の順序を強制的に決定することのできる「**シーケンスストラクチャ**」が用いられる。シーケンスストラクチャは、For ループや While ループと同じく、関数パレットの「ストラクチャ」欄内にある。

シーケンスストラクチャは、ブロックダイアグラムにおいて、内部に各種制御器や関数を配置できる枠組み（フレーム）として配置される。順番に実行したい幾つかのプログラム部分を、その順番どおりに各フレームに入れてゆくことで、プログラムの流れは決定される。では、この実行順序が問題になる例を、図 37 に示す。図 37 のプログラムは、まず左の For ループ内で、各瞬間の時刻を表示させ、ループが終わった後に 500 ms の待機時間を経て、その時点での時刻を表示させることを意図している。（※　実際の測定プログラムにおいて、このように実行順序を守らせたい場合は多々ある。）

このプログラムを実行させると、For ループ実行後の時刻(タイムスタンプ)と、500 ms 後に表示させたかった時刻(タイムスタンプ２)の両方が表示されるが、図 37 の実行結果は、意図に反して両方の時刻が同じになってしまう。

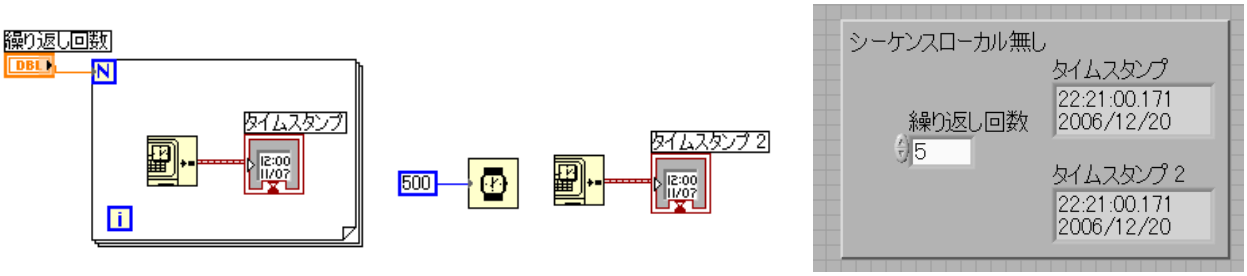



図 37：シーケンスストラクチャを用いないプログラム例と、その実行結果.

これに対し、シーケンスストラクチャを用いて、意図したとおりの実行順序にしたのが、図 38 のプログラムである。実行後のタイムスタンプ 1 と 2 を見ると、予定通りの時間差が記録されていることが分かる。

**問題４－３：**

各自これらのプログラムを描いてみて、実際に実行時間のずれを確認せよ。

※「実行のハイライト」ボタンを押して、プログラム進行の様子を目視してみよ。

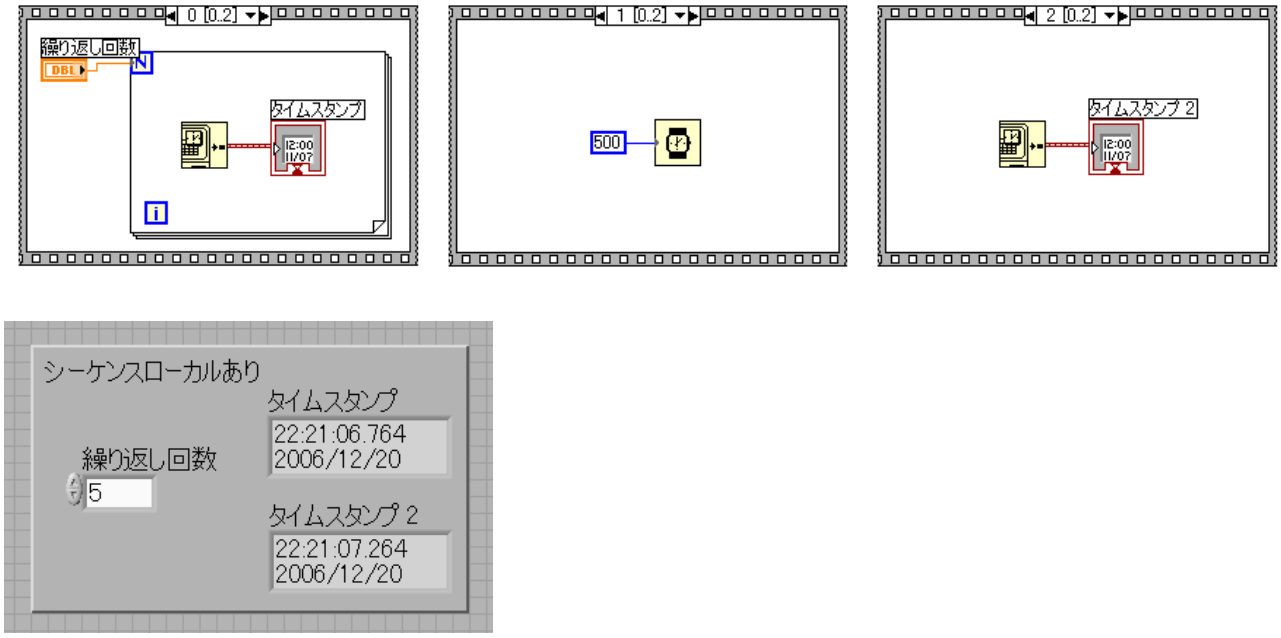



図 38：シーケンスストラクチャを用いて図 37 のプログラムを書き換えたものと、その実行結果.

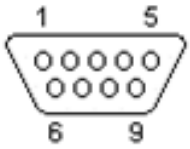
# 第五回　計測器を用いた信号の入出力実験　---計測用インターフェースの形成---

コンピュータと外部機器の間で通信に用いられている規格は、RS232C, GPIB, USB, IEEE1394, イーサネット(LAN), SCSI, パラレル接続(プリンタポート)など、多岐にわたっている。中でも計測機器との通信で、現在広く用いられているのは、**RS232C**, **GPIB**, **USB** と呼ばれる各通信規格である。

## RS232C

シリアル伝送の通信規格。米国の EIA（The Electronic Industries Alliance）により通信用として規格化され、ここでは、テレタイプライタやパソコンとモデム等の間を接続してデータ伝送を行うための電気的・機械的な特性が定められている。これまでの装置制御における主要な通信方式であり、現在でも多くの機器がこの通信ポートを有している。しかしその仕様の古さから、現在では他の通信形態にその役割を取って代わられつつある。

図 39：RS232 通信用の D-Sub25 ピン／9 ピン規格のコネクタケーブル




**DSub 9Pin**

| Pin番号 | 信号名 | 信号 | 内容 |
|---|---|---|---|
| 1 | CD | Carrier Detect | キャリア検出 |
| 2 | RXD | Receive Data | 受信データ |
| 3 | TXD | Transmit Data | 送信データ |
| 4 | DTR | Data Terminal Ready | データ端末レディ |
| 5 | GND | Ground | 信号グランド |
| 6 | DSR | Data Set Ready | データセットレディ |
| 7 | RTS | Request to Send | 送信リクエスト |
| 8 | CTS | Clear to Send | 送信可 |
| 9 | RI | Ring Indicate | リング表示 |

図 40：RS232 通信用 9 ピンコネクタの各端子に対する信号割り当て一覧.

※　信号の伝送には、基本的に GND と RXD と TXD の 3 本の信号線が使用される。それ以外の信号線は RS232C がもともとモデムとの通信用に開発された際の名残であると言ってよい。RS232C の信号は、1 バイトごとに、スタートビット→データ→(パリティー)→ストップビットの順番で送られ、これにより繋がれた双方の機器が、データの区切りをそれぞれ認識することになる。

**特徴**

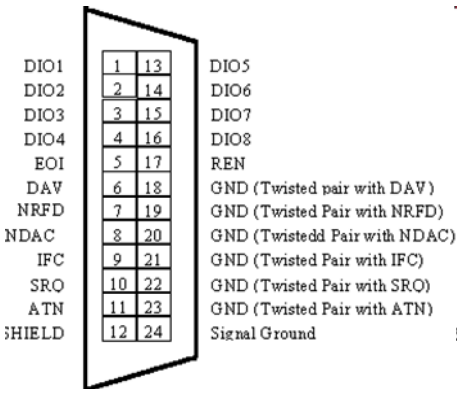
1）RS232 端子は、通常コンピュータに標準的に付属しており、これ用のインターフェースを別途購入しなくても外部機器の制御が可能である。ただし、通常付属しているのは 2 個程度迄であり、それ以上の数の機器を制御するには、拡張ポートを用いて RS232C のポートを増やす等の工夫が必要である。また、最近のデスクトップ PC やノート PC では、RS232C 端子が元々付いてないものが増えてきている。

2）通信速度が比較的遅く(最高通信速度は 115.2kb/s)、ケーブルの最大長は約 15m までとなっている。

## GPIB ----- General Purpose Interface Bus -----

パラレル伝送の通信規格。ヒューレットパッカード社の開発した通信規格である、HP Interface Bus (HP-IB) を基に、他メーカーも採用する規格として制定された。現在このバスは、IEEE (The Institute of Electrical and Electronics Engineers, Inc. 電気電子学会)によって、「IEEE-488」として形式化されている。市販されている各種計測器には GP-IB 端子を持っている物が多く、各種 GP-IB インターフェースを付けたパソコンを用いて、これらの制御や計測データの取り込みが行える。

図 50：GPIB 通信用の専用規格ケーブル




| No。 | 信号･記号 | 機 能 | |
|---|---|---|---|
| 1 | DIO 1 | データ | |
| 2 | DIO2 | データ | |
| 3 | DIO3 | データ | |
| 4 | DIO4 | データ | |
| 5 | EOI | End or Identify | データ転送の終了 |
| 6 | DAV | Data valid | データバス上のデータ有効 |
| 7 | NRFD | Not ready for data | データの受信準備完了 |
| 8 | NDAC | Not data accepted | データを受け取った |
| 9 | IFC | Interface clear | GPIB機器のインターフェス初期化 |
| 10 | SRQ | Service request | コントローラへのサービス要求 |
| 11 | ATN | Attention | コマンドとデータの識別 |
| 12 | シールド | ケーブルシールド | |
| 13 | DIO5 | データ | |
| 14 | DIO6 | データ | |
| 15 | DIO7 | データ | |
| 16 | DIO8 | データ | |
| 17 | REN | Remote enable | リモート制御とローカル制御の識別 |
| 18 | グラウンド | DAV用 | |
| 19 | グラウンド | NRFD用 | |
| 20 | グラウンド | NDAC用 | |
| 21 | グラウンド | IFC用 | |
| 22 | グラウンド | SRQ用 | |
| 23 | グラウンド | ATN用 | |
| 24 | グラウンド | 論理信号共通 | |

図 51：GPIB 通信用コネクタの各端子に対する信号割り当て一覧.

GPIB 通信用ケーブルは、双方向データ通信用の 8 本の信号線からなる「**データ線**」、3 本の信号線からなる「**データ伝送制御線**」、そして 5 本の信号線からなる「**バス管理線**」で構成される。 (1)データ線(No. 1～4, 13～16)は、データを ASCII コードとしてパラレルに転送するのに使用する。また、データ線はこうした計測器の測定結果や条件設定だけではなく、機器を指定するための情報伝達にも使用される。 (2)データ伝送制御線(No. 6～8)は、データ線上の信号の確実な受渡しを行なうための制御(ハンドシェークという)を行なうラインである。(3)バス管理線(No. 5, 9～11, 17)はバス上のデータの流れを制御するために使用する。

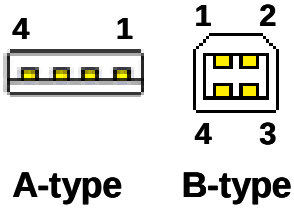
**特徴**

1）最大 15 台までの機器を並列に接続し、それぞれ個別に制御することが出来る。
2）転送速度が比較的速い(約 1Mb/s. 最近では約 8 Mb/s まで高速化されている)。
3）ケーブル長は、各機器間で 2m 以内、全長が 20m までに制限されている。

## USB ----- Universal Serial Bus -----

シリアル伝送の通信規格。従来からのポートに代わる新たな汎用バス・インターフェースとして、Compaq・Intel・Microsoft・NEC により策定された。今や PC において最もポピュラーな通信規格となっており、ほぼ全ての PC にこの端子が存在する。特に USB1.1 から USB2.0 仕様への改善で、転送速度に大幅な向上が見られ、このことにより急速に普及が進み適用機器の範囲が広がっている。

図 52：USB 通信用ケーブル(A タイプおよび B タイプコネクタ).




**USB 4Pin**

| Pin番号 | 信号名 | 信号 | 内容 |
|---|---|---|---|
| 1 | VCC | +5V VDC | キャリア検出 |
| 2 | D+ | Data − | データ + |
| 3 | D− | Data + | データ − |
| 4 | GND | Ground | データ端末レディ |

図 53：USB コネクタの各端子に対する信号割り当て一覧.

※ USB 規格ではデータ線以外の信号線が全て無くなっており、通信の制御は、データ線上を流れる信号に載せられたプロトコル(通信規約)で行われている。

**特徴**

1）PC の電源を切らなくても端子の抜き差しが可能である(ホットプラグ)。

2）ハブ(HUB)を用いれば、最大 127 台の機器まで多段接続が可能。また小電力な機器なら USB 端子自体から給電することが出来る。

3）伝送速度が高速(USB1.0 で最大 12Mb/s　⇒　USB2.0 では最大 480Mb/s)。

4）一本のケーブルの長さは 5m、総延長は 25m まで(USB ハブを使用した場合も然り)。

## オシロスコープの使い方

本講義では今後、信号電圧の値および時間発展の確認のためにオシロスコープを使用する。ゆえにまずここで、オシロスコープの基本的な使用方法について簡単に述べておく。オシロスコープとは、時間の経過と共に電気信号（電圧）が変化していく様子をリアルタイムでブラウン管に描かせ、目では追えない高速な電気信号の変化していくさまを観測できるようにした、電圧波形測定器である。

ブラウン管テレビとオシロスコープは、真空中で電子銃から出た電子線が画面の裏側の蛍光体に当たり、これが蛍光面上の輝点として現れるという点では、表示原理が共通している。ブラウン管テレビでは、上下左右に配置された偏向板からの周期的な電界変化によって、左から右、端まで動けば一段下に移って再び左から右へと、この電子線の方向がディスプレイ面上を走査するように動かされ、最終的に 1 つの画面を形成している。オシロスコープにおいては、横方向の電子線走査が時間に、縦方向の走査が入力電圧値に対応するように偏向板の電界強度が変調されている。従って、ディスプレイ上の輝点の動きの速さや振れの大きさを測ることで、間接的に電気信号の電圧の時間的変化を簡単に可視化することができる。　図 54 は、この静電偏向方式で構成されるブラウン管オシロスコープの模式図である。

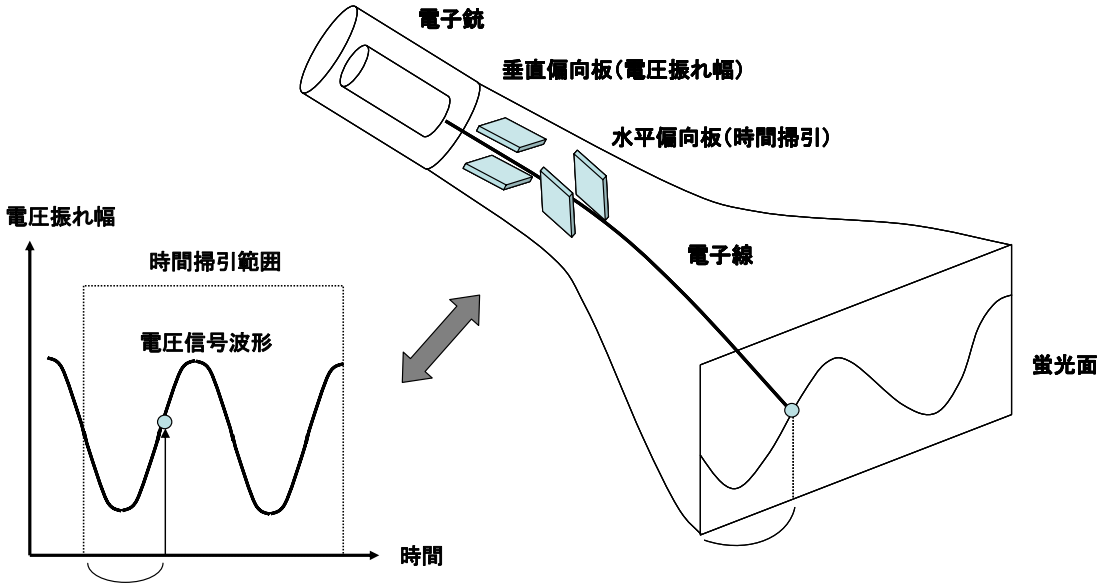



図 54：静電偏向形ブラウン管オシロスコープの模式図．入力信号の電圧振れ幅を縦、時間掃引を横方向とした偏向板の電界により、電子線を蛍光面上で走査させて信号電圧の時間変化を可視化する．

信号が無い場合には、オシロスコープの蛍光面上には時間掃引に伴って**トレース**（軌跡）と呼ばれる水平線が画面の左から右に繰り返し引かれる。オシロスコープの時間軸調整つまみは、この線を引く時間を設定するものである。入力電圧が 0 から離れていたら、トレースはそれによって基準位置より上側または下側に外れた位置にくる。垂直軸調整つまみは、この垂直方向の変位量の拡大または縮小表示設定を行うものである。

もし信号が周期的であれば、時間軸を入力信号の周波数に合わせて設定することで、ほぼ固定したトレースが得られる。オシロスコープの時間掃引設定が完全に正確ではない、あるいは入力信号の周波数が完全に固定でなければ、トレースは浮動して測定するのが難しくなる。そこで、周期信号において固定したトレースを得るために、オシロスコープには**トリガ**と呼ばれる機能がある。このトリガ設定を行うと、画面の右端に行った後、左端に戻るまで特定のイベント（トリガの種類によって異なる)が有るまで待機し､その後に次のトレースを描画するようになり､安定したトレースが得られる。

<u>トリガの種類</u>

1）外部トリガ　：外部ソースからのパルスを、専用の入力端子に入れる。
2）エッジトリガ：入力信号が、決められた方向から決められたしきい電圧を横切った時、エッジ検出器がパルスを生成する。
3）遅延トリガ　：エッジトリガから掃引を開始するまで、調節した特定の時間待機する。

等々

## デジタルストレージオシロスコープ

普通のオシロスコープでは､繰り返しの信号波形を目の残像現象を利用して見えるようにしているが、単発現象は減衰が速すぎてこれでは見えない。アナログストレージ(蓄積)オシロスコープでは、通常のオシロの表示画面に特殊な処理を施し､掃引された残像が長時間表示されるようになっており、これを用いることで稀に起こる現象を観測することが出来る。これを発展させたデジタルストレージオシロスコープ(DSO)では、ストレージの手段をデジタルメモリで置き換え、データを劣化なく好きなだけ保持することができるようになった。

DSO では、垂直入力は、垂直偏向板を駆動する代わりにアナログ-デジタル変換回路でデジタル化され、高速なデジタル信号処理によってデータ列としてメモリに保存されるとともに、表示器に送られる。また、単発現象にトリガをかけ波形を止めたり、トリガ条件の前におこった現象を確認したりといった､アナログオシロでは不可能なこともできる。また､通常たくさんの有用な信号解析機能（例えば立ち上がり時間、パルス幅、振幅、周波数スペクトル、ヒストグラムや統計等）が付加されており、電気通信の技術者や研究開発に携わる者にわかり易く測定結果を表示させることができる。すなわち、DSO にはアナログオシロよりも遥かに広帯域な信号測定や複雑な信号処理も行える利点がある。今回使用するのは、<u>菊水電子工業株式会社のデジタルストレージオシロスコープ（COM7061）</u>である。

別紙Bに示すこのオシロスコープの前面パネル上で、主として操作する部分は以下の通りである。

| ノブやスイッチの名称 | 機能 |
|---|---|

■　画面設定

| | |
|---|---|
| ① POWER | 電源スイッチ |
| ② INTEN | 波形の輝度を調節するノブ |
| ③ TRACE ROTA | スクリーンの輝線の傾きを修正するノブ |
| ④ FOCUS | 波形の焦点を調節するノブ |
| ⑤ B INT SCAL READ OUT | 目盛照明の明るさを調節するノブ |

■ CH1入力設定

| | |
|---|---|
| ⑥ VOLTS/DIV | 垂直感度を切り換えるスイッチ |
| ⑧ CH1 INPUT | CH1信号を入力する端子 |
| ⑨ COUPLING AC-GND-DC | 入力信号の周波数成分を選択するスイッチ |
| ㊴ VERTICAL MODE | 垂直入力信号を選択するスイッチ |
| ㊵ VERTICAL POSITION | 波形の垂直位置を調節するノブ |

■　時間掃引設定

| | |
|---|---|
| ⑰ TIME/DIV | 掃引時間を切り換えるスイッチ |
| ⑳ HORIZONTAL POSITION | 波形の水平位置を調節するノブ |

■ トリガ設定

| | |
|---|---|
| ㉓ TRIGGER MODE | トリガモードを選択するスイッチ |
| ㉔ TRIGGER SOURCE | トリガソース（信号元）を選択するスイッチ |
| ㉘ TRIGGER SLOP | トリガスロープを選択するスイッチ |
| ㉚ TRIGGER | トリガレベルの調節ノブ |

## オシロスコープの使用：正弦波の観察

波形発生器(ファンクションジェネレータ)からの出力を直接オシロスコープのチャンネル１(CH1)に接続する。CH1の入力はまずDC Couplingにする。ボルトレンジは一番低い感度にする（大きな電圧側）。波形発生装置の出力は 数 kHz で適当な振幅の正弦波とする。トリガはソースをCH1、Normalとする。この状態で入力の波形が見えるようにトリガレベルを調整する。信号が止まって見える状態でトリガレベルを動かすと、画面上の正弦波の位相がずれていくのが見えるはずである。これは、時間掃引の基準位置がトリガレベルに応じて変わることに拠る。また、トリガスロープを反転させると、画面左端の信号の高さは変わらずに、正弦波の位相が反転するのが確認できるはずである。これは、180度位相のずれた時間位置にトリガを設定したことになる。

0 レベルの設定とレンジの調整

トリガをAuto にしてCH1のカップリングをGND（Ground） にすると、水平な輝線が1本現れるはずである。この線（GND）はCH1の入力信号の基準電位を表しているので、この状態でオシロスコープの信号振れ幅の縦軸方向の原点合わせを、VERTICAL POSITIONノブを用いて行う。通常は、画面の中央の水平グリッド線にこの輝線の位置を合わせておけばよい。トリガをAuto にするのは、トリガ設定を抜きで信号のスキャンが行われるようにするためである。原点合わせが終わったら、CH1のカップリングをDC に戻してトリガをNormal にする。信号が見られない場合や、信号が静止しない場合は、トリガレベルを調整してこれを見えるようにする。続いて、入力感度を調整して信号が画面内部でなるべく大きく見られるようにする。

時間掃引の設定

正弦波が画面できれいに見えるようになったら、時間掃引設定を行う。これは画面の 1 目盛りがどの程度の時間になるかを定めるものである。時間掃引幅を小さくすると、より短時間の領域を拡大して表示できる。実際の計測時には、信号に応じた最適な時間領域を見出し、適宜時間掃引幅の設定を行うこと。

## LabVIEW 言語を使用して USB 計測器の制御を行う

講義において使用する計測器は、**ナショナルインスツルメンツ社製　Data Acquisition (DAQ) Device USB-6008 (USB 接続)** である。これは、デジタル信号および 12bit までのアナログ信号の入出力が可能な、USB ポートを使用する汎用型の計測・制御機器である(図 55)。

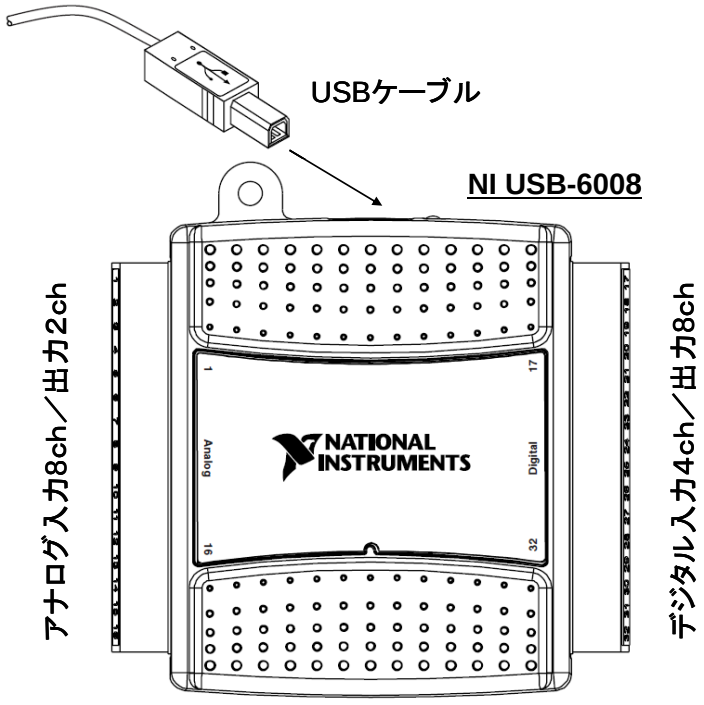



図 55：汎用 USB 計測器 USB-6008 の外観.

各自、USB ケーブルを用いてこの測定器を PC と接続してみよ。USB-6008 の USB コネクタ付近にある黄色の LED ランプが点滅を開始すれば、接続は正常に完了している。PC 上から接続状況を確認するには、デスクトップ上にある「Measurement & Automation」アイコンをダブルクリックする（図 56)。


図 56：「Measurement & Automation」アイコン.

「Measurement & Automation Explorer」の画面が現れたら、ここで「構成」ツリーの「デバイスとインターフェース」項目を見ると、「NI－DAQmx デバイス」の項があるのが分かる。ここに、先程接続した「NI USB-6008」が、“Dev1”として登録されていることを確認できるはずである（図 57)。

この Measurement & Automation Explorer（MAX）アプリケーションは、コンピュータに接続されている計測器の自動検出やドライバのインストール等のハードウェアの管理と、ドライバを介した制御ソフトウェアの構成などを行うためのユーティリティである。また、USB 測定機器だけではなく、GPIB や RS232C 等のポートで接続された外部測定機器に関しても、そのデバイスおよびインターフェースの物理プロパティ(リソースなど)や属性(アドレス番号など)の管理・確認を行うことが出来る。

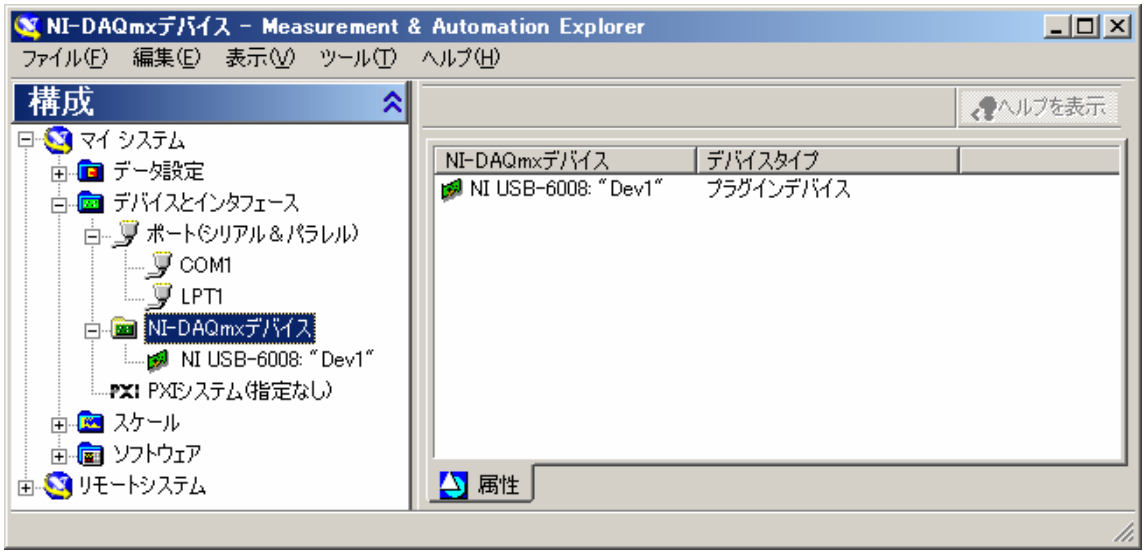



図 57：「Measurement & Automation Explorer」の画面.

MAX の構成ツリー上に現れた USB-6008 ”Dev1”の項目を右クリックすると、「テストパネル」というメニューが現れるが、これを用いて簡易的に、USB-6008 に装備された各アナログおよびデジタル入出力端子のモニタおよび制御を行うことも出来る。これを実際に試してみよう。

テストパネルの画面が現れたら、その中で「電圧出力」のタブを選択してみる。すると図 58 のような画面になるはずである。チャンネル名「Dev1/ao0」は、接続された機器“Dev1”および電圧値を出力する端子“ao0”を表している。端子“ao0”は、別紙 A の端子一覧表に示しているように、2 つあるアナログ出力端子の内の片方で、ピン番号 14 にある端子（AO0 と表記）である。この端子と、対になる GND 端子である 13 番目の端子に、BNC コネクタと繋がった赤色及び白色の配線をそれぞれ接続してみよ。

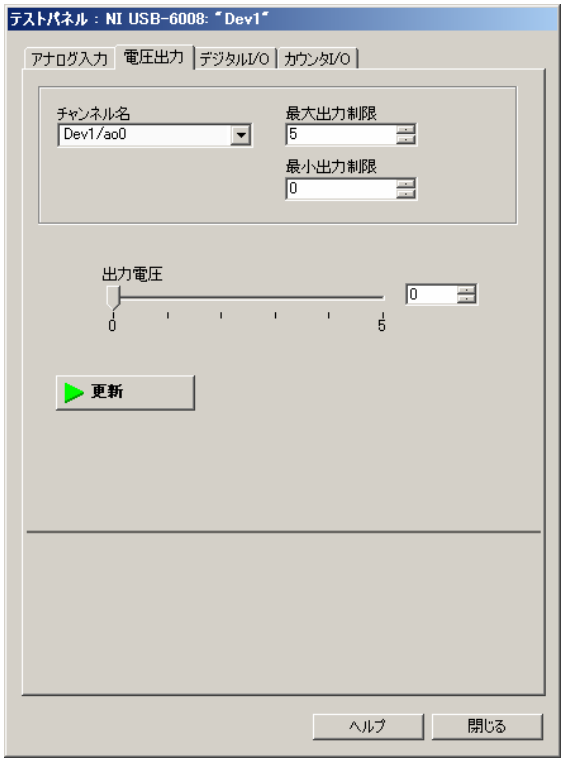



図 58：MAX プログラムが認識した「NI－DAQmx デバイス」の
「USB-6008 ” Dev1”」におけるテストパネル（電圧出力に関するもの）.

この状態で、「出力電圧」のスライダを動かして適当な電圧値を選び、「更新」のボタンを押すと、その電圧値が端子 14（AO0）と端子 13（GND）の間に印加されることになる。

※　但し、DAQ（USB-6008）の電圧出力の最大値は 5V なので(別紙参照)、これ以上には上げないよう注意せよ。

**問題５－１：**

各自、この出力値をオシロスコープで確認してみよ。0 レベルの設定とレンジの調整を行った後、静的なこの電圧出力を、トリガモードを Auto として観察せよ。MAX プログラム上で電圧値を変化させてみて、このオシロスコープ上の信号に反映されることを確認せよ。

## 1）DAQ（USB-6008）のアナログ出力を使用するプログラムを作成する

では、DAQ からアナログ信号を出力するプログラムを書いてみよう。まず、正弦波を出力させることとし、ここでは、ブロックダイアグラムの関数パレット内の「Express」欄に含まれる「入力」項にある、「信号シミュレーション」関数を使用する。この Express 欄内の関数は「**Express VI**」と呼ばれており、一連の作業に使用する VI 群がひとまとめにされ、ダブルクリックして現れるプロパティ内で対話的にその構成の変更が行えるものである。

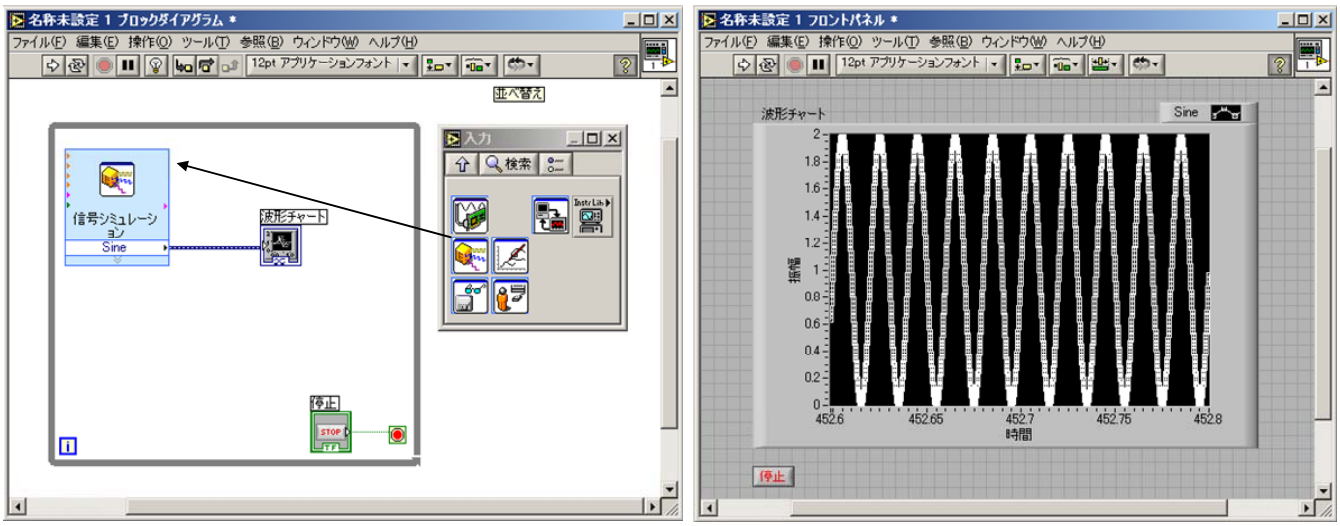



図 59：Express VI「信号シミュレーション」を用いて正弦波を出力するプログラム.

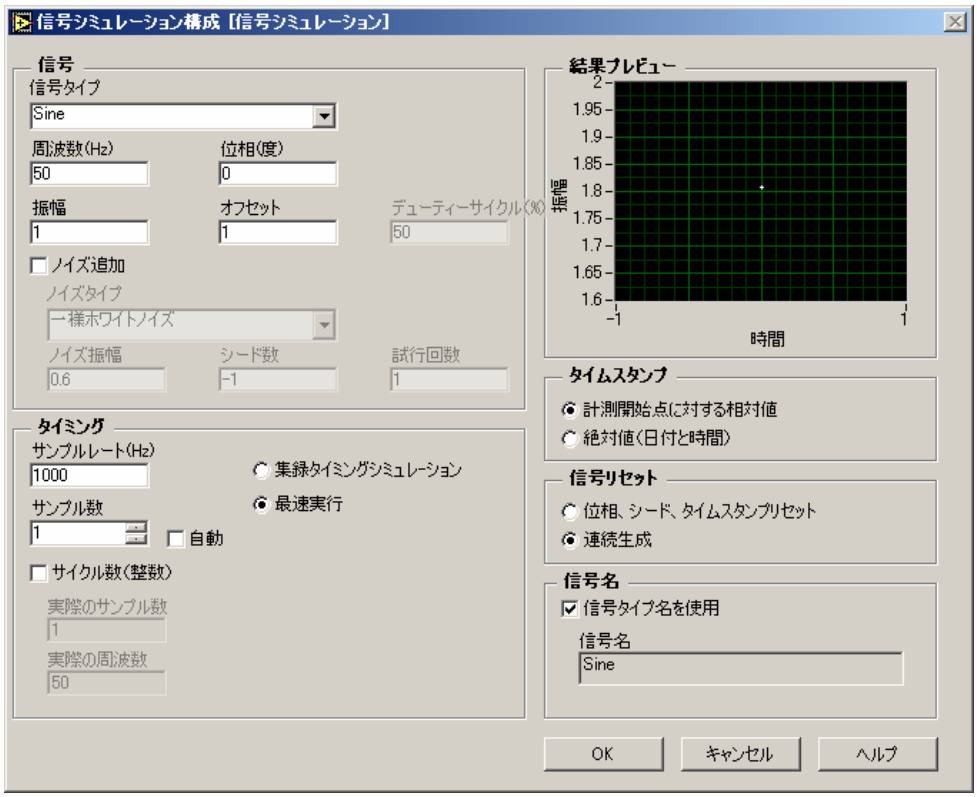



図 60：Express VI「信号シミュレーション」のプロパティ設定画面.

図 59 にプログラム例を示す。フロントパネル上には波形チャートを配置し、ブロックダイアグラム上では、信号シミュレーションの Express VI を while ループで囲み、信号を連続的に出力させるようにする。信号シミュレーション VI を配置した後、これをダブルクリックすると、図 60 のような設定画面が現れる。正弦波の出力値として、信号タイプを「Sine」、周波数 50 Hz、振幅 1 およびオフセット 1 を設定し、タイミングレートは 10 kHz、サンプル数は 1000 で、最速実行を選択する。そうすると、図 60 プロパティ設定画面の結果プレビューに、出力される波形の概形が表示される。最後に OK ボタンを押して設定を反映させると、信号シミュレーション VI は、このパラメータで正弦波を出力することになる。

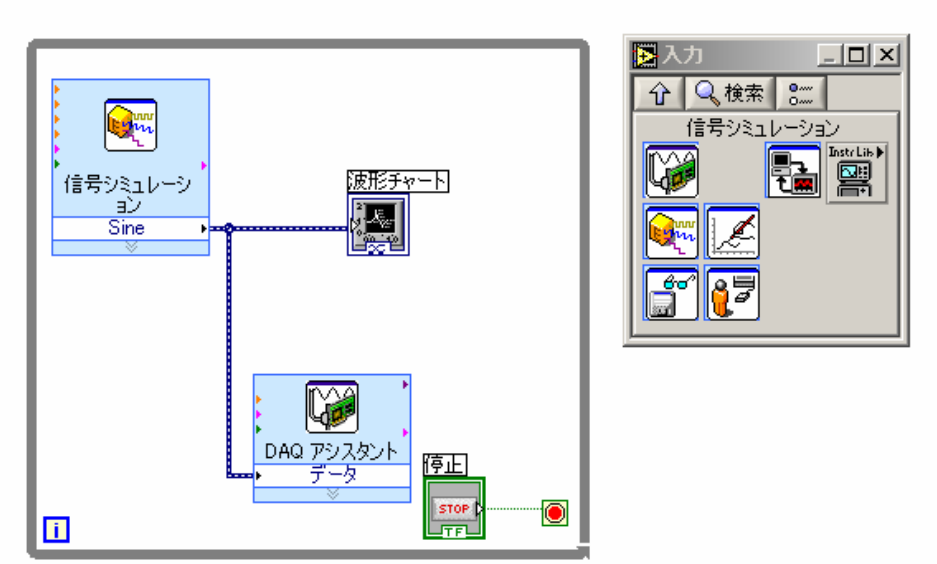



図 61：正弦波を DAQ のアナログ出力端子から出力するプログラム.

信号シミュレーション VI の出力を元に、DAQ アシスタント VI を使用してアナログ出力ポート(ao0)から電圧出力させるプログラムは、図 61 のようになる。信号シミュレーションの場合と同様に、この DAQ アシスタント上でも、ダブルクリックすると、DAQ の設定画面が現れる。設定画面の「アナログ出力」の「電圧」項目において、出力先を「ao0」とすると、図 62 の画面になる。タスクタイミングは、1 サンプル（オンデマンド）を選択し、最後に OK ボタンを押してこの設定を反映させ、信号シミュレーションと DAQ アシスタントの VI を配線して完了である。

※　図 62 で示される DAQ の信号出力範囲は 0～5 V となっており、これは別紙 A で挙げたこの DAQ 計測器（USB-6008）の出力定格値に沿っている。従ってここは変えないこと。

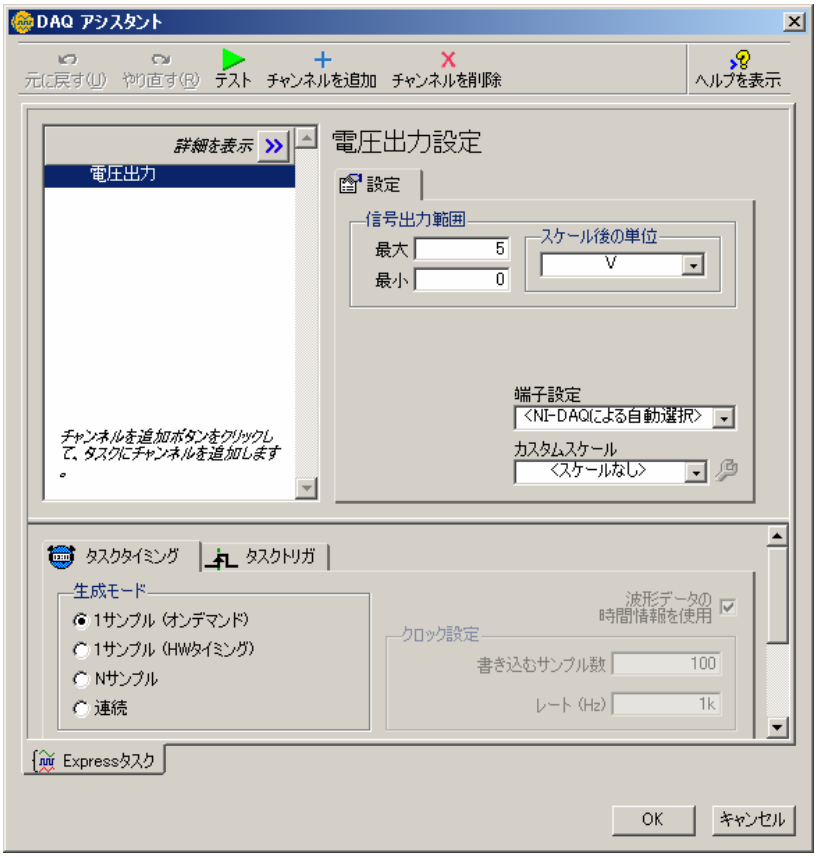



図 62：Express VI「DAQ アシスタント」の電圧出力設定画面.

では、既に DAQ（USB-6008）の各端子（AO0, GND）には BNC コネクタからの配線が繋がっているはずなので、これと接続したオシロスコープで実際の電圧出力波形を確認し、オシロスコープの画面とフロントパネル上のチャート出力を比較してみよ。

**問題５－２：**

ブロックダイアグラム上で信号シミュレーション VI のアイコンの縦幅を広げると、「エラー出力」という項目が増える。この項目上で右クリックして現れるメニューの中から「入出力を選択」および「周波数」を選択すると、この項目名が周波数となって周波数の値を VI の外から指定できるようになる。フロントパネル上に数値制御器の１つであるスライドバーを配置し、ブロックダイアグラム上でこの値を周波数項目に配線することで、プログラム実行中に任意に周波数を変えられるようにせよ。これを実際にオシロスコープ上で確認せよ。

※以上の LabVIEW プログラムと DAQ により、簡易的な波形発生器が作成できたことになる。

## 2）USB-6008 のアナログ入力を使用するプログラムを作成する

次に、外部からのアナログ信号を、DAQ で取得するプログラムを書いてみよう。ここでも、Express VI の DAQ アシスタント VI を用いる。フロントパネル上に波形チャートを配置し、ブロックダイアグラム上では DAQ アシスタント VI を while ループで囲み、入力信号を連続的にモニタするようにする（図 63 左）。

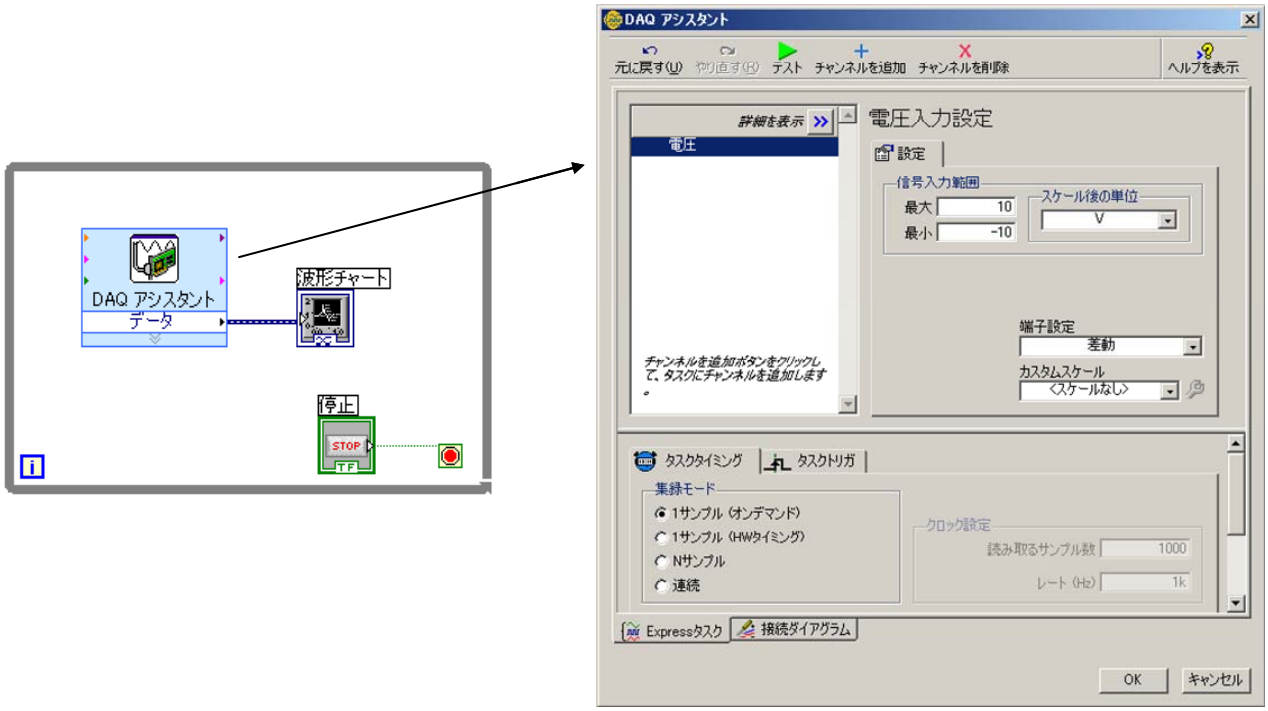



図 63：Express VI「DAQ アシスタント」の電圧入力設定画面.

まず、波形発生器（ファンクションジェネレータ）からの正弦波出力をオシロスコープに接続し、画面を見ながら電圧振れ幅が 2 V、周波数が 10Hz 程度になるように、波形発生器の出力レンジを変更する。その後に、BNC コネクタに繋がった 2 本の配線を、DAQ（USB-6008）の 2 番の端子（AI0+）と 3 番の端子（AI0-）に接続せよ。図 63 右のように DAQ アシスタントの電圧入力設定を行った後、プログラムを実行させると、波形発生器からの正弦波信号が取得され、それがフロントパネルの波形チャート上に時々刻々と表示されるはずである。

※　DAQ の信号入力範囲は、別紙 A にあるように±10 V であるので、波形発生器からの電圧出力はこれを超えないようにしなければならない。

波形発生器の周波数を上げてゆくと、波形チャート上の 1 つの正弦波を形成する点の数が減少し、正弦波を表現できなくなってくる。これは、本プログラムを用いた DAQ によるデータの抽出（サンプリング）が、信号に対して追いつかなくなることに拠る。

**問題５－３：**

図 63 のプログラムを元にして、横軸を時間、縦軸を DAQ からのアナログ入力値とするデータを 200 点取得し、そのデータをスプレッドシート形式のファイルに保存するプログラムを書け。その際に、数値表示器は浮動小数点形式、有効数字は 9 桁、表記法は倍精度（DBL）を使用せよ。また、波形チャートの表示領域も適当に変えて見やすいようにせよ。

**ヒント**

1）For ループを使用し、「初期化配列」関数で作成した空配列をシフトレジスタで毎ループ使えるようにする。

2）配列関数としては、「初期化配列」「部分配列置換」「配列連結追加」を使い、各測定時刻は「ティックカウント」から取得する。

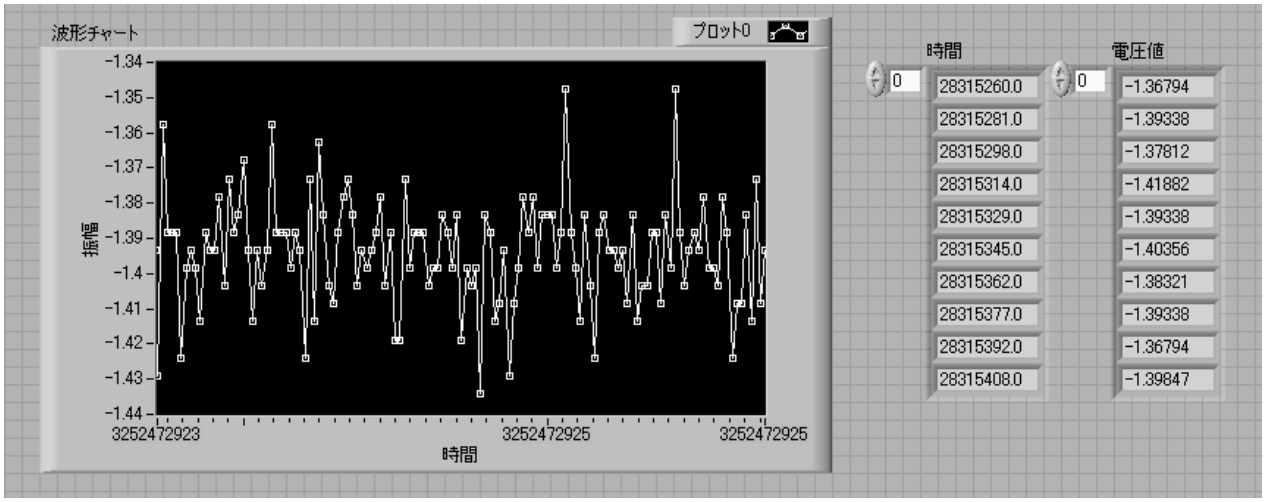



図 64：DAQ 入力端子から電圧データを取得し、それをスプレッドシートファイルに保存するプログラムのフロントパネル.

プログラムを実行すると分かるが、データの取得スピードには限度がある。どの程度の周波数の信号まで取り込めるかを、１つの正弦波を構成する点の数および正弦波の振れ幅を見ることによって、各自確認してみよ。それと別紙 A に表記されている DAQ のスペック上の性能限界とを比較して、考察を行ってみると良い。

**問題５－４：　デジタル入出力の確認**

電気的に構成された論理回路においては、通常各点での論理の真偽の判別を、その箇所での電圧値が 0 V（Low）であるか 5 V（High）であるかによって区別している。この間の閾値（どこで High と Low を区別するかの線引きの値）の設定は、場合にもよるが大体 0～2 V を Low、4～5 V を High とするようである。実際、今回使用する USB 計測器 USB-6008 においては、-0.3～0.8 V を Low、2.0～5.8 V を High としている。デジタル入出力は、この High と Low の 2 つの電圧値を用いることにより、計測器および PC に対して真か偽かの二値の入出力を与えるものである。

デジタル出力ポートへ BNC コネクタの配線を繋ぎ、各自それぞれデジタル出力のプログラムを作成した後、アナログ入出力の場合と同様にデジタル出力の電圧値をオシロスコープで確認せよ。

# 第六回　データ解析手法

## 過渡現象

過渡現象とは、ある事象が初期状態から定常状態に達するまでの過渡的な変化を指す言葉であり、この現象は、物質の運動から電磁波に至るまで、あらゆるところで我々の生活に深く関わっている。

1）雨が雲から落下し、地上付近で一定の速度になるまでの過程
2）夜光塗料からの発光が減衰してゆく過程
3）加熱した金属が冷えてゆく過程

等、数え上げれば枚挙の暇も無い。とりわけ、電気回路における過渡的な電流の流れは、発電機から電気を生む過程や、電磁波を発生させる過程、高速な繰り返しパルスを用いた演算過程などの場合において根幹を成すものである。身近にある電気製品は、すべてこの過渡的な電流を制御することによって動作していると言っても過言ではない。したがって電気回路は、過渡現象の理解を進めるにおいて、非常に良い題材となる。

そこで、本講義では、電気回路におけるスイッチ開閉操作や外部信号入力の際に、回路各部の電流や電圧の値に現れる過渡現象について学ぶ。電気回路の過渡現象の起こる理由を理解し、数学的手法用いてそれを解く方法を修得し、さらに、過渡現象を計算機でシミュレーションすることにより、その解析結果を体感することを目標とする。

## LCR 回路

それでは、図 65 のようなコイル（L)、コンデンサ（C)、抵抗（R）から構成される回路（LCR 回路）を流れる電流について考えてみよう。スイッチを ON にし、回路に時間変化する電流 $I$ が流れたとき、の両端にかかる電圧をそれぞれ $V_C$，$V_R$，$V_L$ とおくと、これらは以下のように表される。

$$\text{C}: V_c = \frac{1}{C}\int_0^t Idt + V_{C0} \qquad \text{R}: V_R = RI \qquad \text{L}: V_L = L\frac{dI}{dt} \qquad (1)$$

※$V_{C0}$ はあらかじめ電荷が蓄えられている場合のコンデンサの電圧.

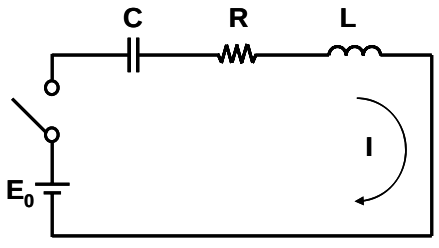



図 65：LCR 回路（直流電圧入力の場合）.

ここでは、キルヒホッフの第二法則より、回路電圧の釣り合いの式、

$$L\frac{dI}{dt}+RI+\frac{1}{C}\int_0^t Idt+V_{C0}=E_0 \qquad (2)$$

が成立している。ある時刻$t$における電流 $I$ の値は、式（1）の両辺を微分して得られる以下の 2 階常微分方程式

$$L\frac{d^2I}{dt^2}+R\frac{dI}{dt}+\frac{1}{C}I=0 \qquad (3)$$

を出発点とし、この微分方程式の解を得ることによって求めることが出来る。電圧源 $V$が一定値ではなく時間変化する電圧$e(t)$を与えるものであるならば、(2) 式は、

$$L\frac{d^2I}{dt^2}+R\frac{dI}{dt}+\frac{1}{C}I=\frac{de(t)}{dt} \qquad (3')$$

という形になる。2 階常微分方程式（3）は、式（3’）の基本解$I_t$を与えており、微分方程式（3’）の一般解$I$は、この基本解$I_t$に特解$I_s$を加えたものとなる。そこでまずこの基本解を導き、次いで特解を求めることにする。

**基本解**

式（3）の微分演算子の部分を、 $d/dt=m$ および $d^2/dt^2=m^2$ と置き換えて、

$$Lm^2I+RmI+\frac{1}{C}I=\left(Lm^2+Rm+1/C\right)I=0 \qquad (4)$$

と表現しなおすと、$Lm^2+Rm+1/C=0$ が解を持つための条件となる。この、$m$ に関する二次方程式の解$m_1$と$m_2$は、それぞれ解の公式により簡単に求められ、

$$m_1=\frac{-R}{2L}+\frac{\sqrt{R^2-4L/C}}{2L}=-\alpha+\beta \qquad (5)$$

$$m_2=\frac{-R}{2L}-\frac{\sqrt{R^2-4L/C}}{2L}=-\alpha-\beta \qquad (6)$$

このようになる。但し、簡単のために $\alpha=R/2L$， $\beta=\left(\sqrt{R^2-4L/C}\right)/2L$ と置いた。

以上のことから、方程式（3）は、$\beta$の値が実数、０および虚数である３つの場合について、以下のようにそれぞれ異なる解を持つ。

（ｉ）$\beta$が実数（　$R^2 > 4L/C$　）のとき、(4) 式の解は　$m_1 = -\alpha + \beta,\quad -\alpha - \beta$ であり、

式 (3) の一般解は、

$$I_t = K_1 e^{m_1 t} + K_2 e^{m_2 t} = e^{-\alpha t}\left(K_1 e^{\beta t} + K_2 e^{-\beta t}\right) \qquad (6)$$

（ii）$\beta = 0$ （　$R^2 = 4L/C$　）のとき、(4) 式の解は重根 $m = -\alpha$ であり、 式(3)の一般解は、

$$I_t = e^{-\alpha t}\left(K_1 + K_2 t\right) \qquad (7)$$

（iii）$\beta$が虚数（　$R^2 < 4L/C$　）のとき、$\gamma = \left(\sqrt{4L/C - R^2}\right)/2L$　と置くと、(4) 式の解は

$m_1 = -\alpha + j\gamma,\quad -\alpha - j\gamma$　となり、方程式 (3) の一般解は、

$$I_t = e^{-\alpha t}\left\{K_1 \sin(\gamma t) + K_2 \cos(\gamma t)\right\} \qquad (8)$$

と表される。

図 65 のような直流電源回路で、時刻 $t = 0$ に回路のスイッチが入れられた場合を想定すると、電流 $I_t(t)$ の初期条件は、$I_t(0) = 0$　かつ　$L\dfrac{dI_t(0)}{dt} = E_0$　であり、これによって未定係数 $K_1$ と $K_2$ が求まる。

（ｉ）$R^2 > 4L/C$　のとき

$$K_1 = \frac{E_0}{2\beta L} \qquad (9) \qquad , \qquad K_2 = -\frac{E_0}{2\beta L} \qquad (10)$$

$$I_t = \frac{E_0}{2\beta L} e^{-\alpha t}\left(e^{\beta t} - e^{-\beta t}\right) \qquad (11)$$

（ii）$R^2 = 4L/C$　のとき

$$K_1 = 0 \qquad (12) \qquad , \qquad K_2 = \frac{E_0}{L} \qquad (13)$$

$$I_t = \frac{E_0}{L} t e^{-\alpha t} \qquad (14)$$

（iii）$R^2 < 4L/C$　のとき

$$K_1 = \frac{E_0}{\gamma L} \qquad (15) \qquad , \qquad K_2 = 0 \qquad (16)$$

$$I_t = \frac{E_0}{\gamma L} e^{-\alpha t} \sin(\gamma t) \qquad (17)$$

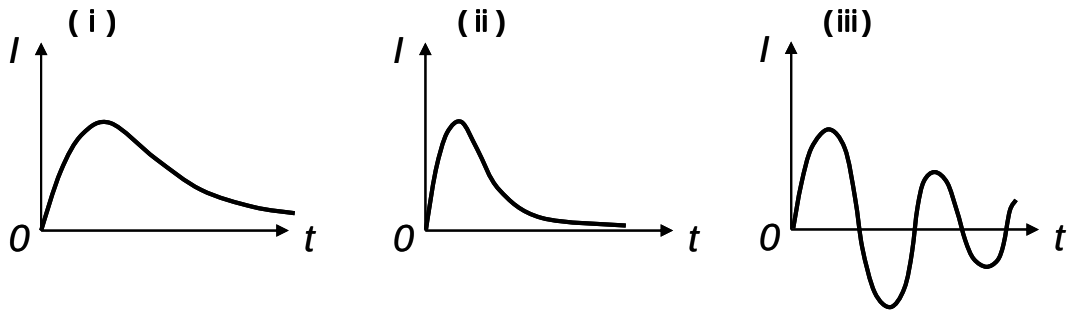



図 66：（ｉ），（ii），（iii）の各場合における過渡電流の時間変化.

**特解**

電圧源が $e(t) = E_0 \sin(\omega t)$ で与えられる振動型であるとすると、特解 $I_s$ も $\sin(\omega t)$ や $\cos(\omega t)$ を含む三角関数となる。そこで、$I_s = I_0 \sin(\omega t + \phi)$ とおいて、未定係数 $I_0$ と $\phi$ を求める方法を取る。

$$L\frac{d^2 I}{dt^2} + R\frac{dI}{dt} + \frac{1}{C}I = \frac{de(t)}{dt} = \omega E_0 \cos(\omega t) \qquad (18)$$

これに、$I_s$ の関数を代入すると、

$$\left(\frac{1}{\omega C} - L\omega\right) I_0 \sin(\omega t + \phi) + R I_0 \cos(\omega t + \phi) = E_0 \left\{\cos(\omega t + \phi)\cos\phi + \sin(\omega t + \phi)\sin\phi\right\} \qquad (19)$$

これが任意の時刻 $t$ で成り立つためには、両辺の $\sin(\omega t + \phi)$ 項と $\cos(\omega t + \phi)$ 項の係数が等しくなければならない。この条件から、

$$\left(\frac{1}{\omega C} - L\omega\right) I_0 = E_0 \sin\phi \qquad (20) \qquad , \qquad R I_0 = E_0 \cos\phi \qquad (21)$$

の関係が得られる。すなわち、未定定数 $I_0$ と $\phi$ は、それぞれ

$$I_0 = \frac{E_0}{\sqrt{\left(\frac{1}{\omega C} - L\omega\right)^2 + R^2}} \qquad (22) \qquad , \qquad \tan\phi = \frac{\frac{1}{\omega C} - L\omega}{R} \qquad (23)$$

と求まり、特解 $I_s = I_0 \sin(\omega t + \phi)$ が得られる。

コンデンサ C、抵抗 R、コイル L に発生する電圧$V_C$，$V_R$，$V_L$は、式（1）に対しこの特解を代入すると分かるように、これらは互いに 90 度位相がずれる。すなわち、図 67 に示すように、$V_R$に対し$V_L$は位相が 90 度遅れており、$V_R$に対し$V_C$は位相が 90 度進んでいる。位相平面上でこれらの電圧ベクトルを合成することにより、そのベクトルの持つ位相$\phi$と回路のインピーダンス$Z$が求められる。このことを式（22）と式（23）は表現している。

LCR 回路のインピーダンス　　$$Z=\sqrt{\left(\frac{1}{\omega C}-L\omega\right)^2+R^2} \qquad (24)$$

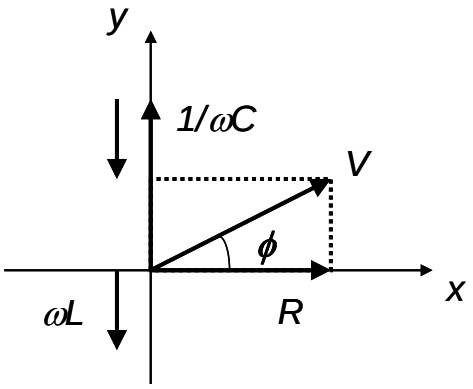



図 67：$V_C$，$V_R$，$V_L$の各電圧ベクトルの位相関係と，
合成電圧ベクトル$V$の大きさ及び位相$\phi$の意味.

なお、特解$I_s$は、入力電圧源の振動数と同じ周波数で振動する電流の形になっている。したがってこれは、インピーダンス$Z$であるこの回路が、定常状態に落ち着いた時点での定常電流であると言える。

$$I_s=\frac{E_0}{Z}\sin(\omega t+\phi) \qquad (25)$$

以上のことから、LCR 回路の式（3’）の一般解（$I=I_t+I_s$）は、電圧源が正弦波　$e(t)=E_0\sin(\omega t)$ のとき、式（6），（7），（8）と式（25）とから、

（ⅰ）$R^2>4L/C$　のとき

$$I=\frac{E_0}{Z}\sin(\omega t+\phi)+e^{-\alpha t}\left(K_1e^{\beta t}+K_2e^{-\beta t}\right) \qquad (26)$$

（ⅱ）$R^2=4L/C$　のとき、

$$I=\frac{E_0}{Z}\sin(\omega t+\phi)+e^{-\alpha t}\left(K_1+K_2t\right) \qquad (27)$$

（iii） $R^2 < 4L/C$　のとき、

$$I = \frac{E_0}{Z}\sin(\omega t + \phi) + e^{-\alpha t}\left\{K_1 \sin(\gamma t) + K_2 \cos(\gamma t)\right\} \qquad (28)$$

このように与えられる。式中の未定係数 $K_1$ と $K_2$ の値は、時刻 $t = 0$ における回路の初期条件、

$I_t(0) = 0$　，　$\dfrac{dI_t(0)}{dt} = \dfrac{E_0}{L}$　を、各場合の式に代入して決めることが出来る。

以上、L, C, R の直列回路における過渡電流の一般的な表式を求めてきた。本講義では、これを単純化し、図 68 のような直流(交流)RC 回路における過渡現象を実験の対象とする。この場合の過渡電流 $I_t(t)$ の値は、式（3）を元にして以下のようにまとめられる。

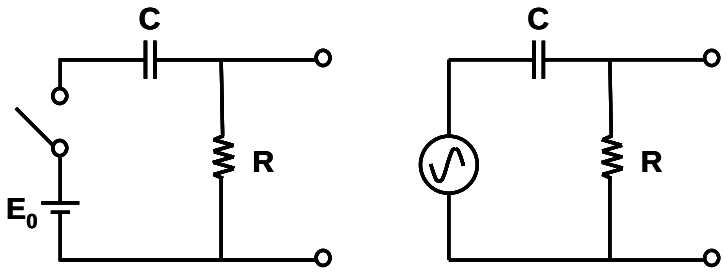



図 68：直流 RC 回路（左）と交流 RC 回路（右）.

■　直流 RC 回路　　----- 直流電圧源は $E_0$ とする -----

$$R\frac{dI}{dt} + \frac{1}{C}I = 0 \qquad (29)$$

この微分方程式から、コンデンサ C があらかじめ充電されていない初期状態の条件 $I_t(0) = \dfrac{E_0}{R}$　により、流れる電流は

$$I = \frac{E_0}{R}e^{\frac{-1}{RC}t} \qquad (30)$$

という式で表される。

■　交流 RC 回路　　----- 交流電圧源は $e(t) = E_0 \sin(\omega t)$　とする -----

$$R\frac{dI}{dt} + \frac{1}{C}I = \frac{de(t)}{dt} = \omega E_0 \cos(\omega t) \qquad (31)$$

この微分方程式から、初期条件　$I_t(0) = \dfrac{E_0}{R}$　により、流れる電流の値は

$$I = \frac{E_0}{Z}\left\{\sin(\omega t + \phi) - \sin\phi\, e^{\frac{-1}{RC}t}\right\} \qquad (32)$$

と表される。ただしここで、 $Z = \sqrt{\left(\frac{1}{\omega C}\right)^2 + R^2}$ , $\tan\phi = \frac{1}{\omega CR}$ である。

これらの解は、いずれも過渡電流(の一部)が減衰することを示している。RC 回路におけるこの減衰の時間は、指数関数内にある値 RC によって特性づけられ、これを**緩和時間（Decay Time）**または**時定数**と呼ぶ。これは、指数関数項の大きさが、最初の値の$1/e$倍になるまでの時間を表している。

※ RC は時間次元を持っている。これを各自確認せよ。

**問題６－１：**

講義において用意する抵抗 R、コンデンサ C は、以下の種類の値を持つものである。

１）抵抗 R（オーム） ： 100, 1k, 10k, 100k, 470k, 1MΩ
２）コンデンサ C（ファラッド）： 0.01μ, 0.1μ, 0.47μ, 1μF

※ 単位は SI 系．また，μ（マイクロ）は $10^{-6}$, k（キロ）は $10^{3}$, M（メガ）は $10^{6}$ である．パラメータの値を SI 単位系で統一しておけば、これらの値を式中でそのまま使用して良い。

これらの値を用いて、実際に回路を作成した場合にどのような電圧波形が観測されるかを確認しておく必要がある。そのために、上記の直流(交流)RC 回路の過渡電流の値$I_t(t)$を用いて、コンデンサ C および抵抗 R の両端にかかる電圧$V_C$, $V_R$の時間発展の様子を、XY グラフにプロットするプログラムをそれぞれ作成せよ。なお、C,R の値は、フロントパネル上で数値制御器を用いて入力できるようにせよ。

※ 講義で使用する計測器（DAQ：USB-6008）のサンプリング速度は、別紙 A にあるように、10 kS/s［ 1 秒間に 10000 回（0.1 ms に 1 点）データを取得できる ］と比較的遅いために、あまり速い過渡現象は測定できない。従って、時定数は 1 ms 程度以上になるように調整せよ。

## LabVIEW 上で LCR 回路の微分方程式の数値解をシミュレートする

次に、微分方程式を元にして、LCR 回路内を流れる電流 $I$ の時間変化を、LabVIEW プログラム上でシミュレートすることを試みる。

一般にｎ階常微分方程式は、ｎ個の 1 階常微分方程式（Ordinary Differential Equations: ODE）に書き換えることが出来る。例えば、(18) 式の形の 2 階常微分方程式

$$a\frac{d^2y(t)}{dt^2}+b\frac{dy(t)}{dt}+cy(t)=g(t) \qquad (33)$$

は、 $x_1=y$, $x_2=\dot{y}$, $u=g(t)$ とし、$x_1$ と $x_2$ を時間微分すると、

$$\dot{x}_1=x_2 \qquad (34)$$

$$\dot{x}_2=-\frac{b}{a}x_2-\frac{c}{a}x_1+\frac{1}{a}u(t) \qquad (35)$$

すなわち、

$$\dot{\mathbf{x}}(t)=\mathbf{A}\mathbf{x}(t)+\mathbf{B}u(t)=\begin{bmatrix}0 & 1\\ -c/a & -b/a\end{bmatrix}\mathbf{x}(t)+\begin{bmatrix}0\\ 1/a\end{bmatrix}u(t) \qquad (36)$$

のように、2 つの 1 階常微分方程式の組に変換することが出来る。LCR 回路の一般式（18）の微分方程式は、これと同様にして、以下のような 2 つの 1 階常微分方程式の組に分けることが出来る。すなわち、

$x_1=I$, $x_2=\dot{I}$, $u=E_0\omega\cos(\omega t)$ とすることにより、

$$\dot{x}_1=x_2 \qquad (37)$$

$$\dot{x}_2=-\frac{R}{L}x_2-\frac{1}{CL}x_1+\frac{\omega E_0}{L}\cos(\omega t) \qquad (38)$$

である。こうした導関数の関係式で書かれる常微分方程式の組から、近似の数値解を求めるための一連の方法としては、「オイラー法」や「ルンゲクッタ法」などがあり、C 言語や FORTRAN 言語には、これらの機能を果たすサブルーチンが多数用意されている。

LabVIEW 言語にも、線形や非線形の常微分方程式を解くための微分方程式 VI が用意されおり、関数パレットから「関数→全関数→解析→数学→微分と積分→微分方程式」と辿ってゆくと、これらを見出すことが出来る。

微分方程式 VI には、図 69 に示すように 1 階微分方程式用のものが 5 種類、高階用が 2 種類の計 7 種類があり、対象とする微分方程式と計算時間に応じて、適切な VI を選択できるようになっている。

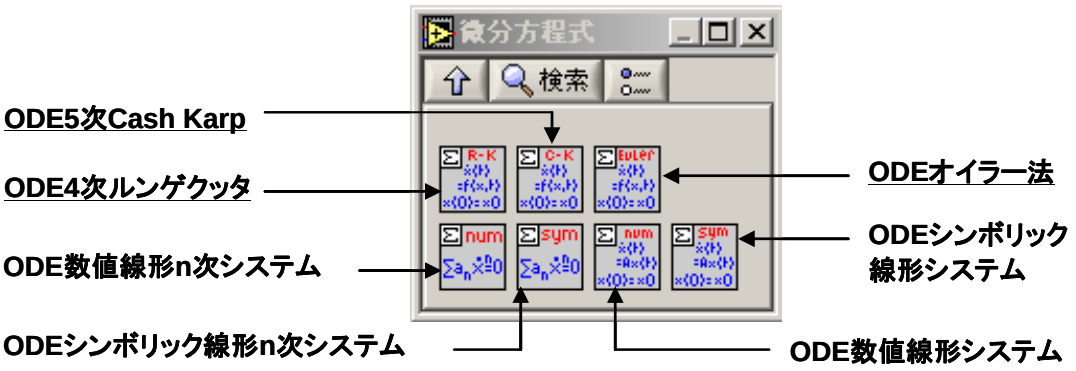



図 69：LabVIEW の微分方程式 VI 一覧.

これらはそれぞれ以下のような特徴を有している。

1）**ODE5 次 Cash Karp**： ステップサイズを調整して数値誤差を低減する可変ステップ式の求解法。
2）**ODE4 次 ルンゲクッタ**： ルンゲクッタ法を用いた固定ステップ式の数値積分法により求解する。
3）**ODE オイラー法**： オイラー法を用いた固定ステップ式の数値積分法により求解する。
4）**ODE 数値線形システム**： 同次線形連立微分方程式の数値解を出力する。
5）**ODE シンボリック線形システム**： 同次線形連立微分方程式のシンボリック解を出力する。
6）**ODE 数値線形 n 次システム**： n 階線形連立微分方程式の数値解を出力する。
7）**ODE シンボリック線形 n 次システム**： n 階線形連立微分方程式のシンボリック解を出力する。

式（20）のような方程式の数値解を求める際には、通常、1）、2）、3）の 3 種類の VI を使用する。その選択の基準としては、一般に

■ 「ODE オイラー法」は、ごく単純な常微分方程式にのみ使用する。
■ 特別な場合以外は、「ODE4 次 ルンゲクッタ」か「ODE5 次 Cash Karp」を使用する。
　□ 等間隔で求解を行う場合には、「ODE4 次 ルンゲクッタ」を使用する。
　□ 大局解と高速処理を求める場合は、「ODE5 次 Cash Karp」を使用する。
■ 定数係数を持つ斉次線形微分方程式に対しては、「ODE 数値線形システム」を使用できる。

このようにすればよい。

※ 以降で提示する全てのプログラムは、「C:¥Documents and Settings¥User ¥My Documents ¥Learning Directory」に置いてある。

## 問題６－２：

常微分方程式の数値解を求めるための「オイラー法」、「ルンゲクッタ法」および「Cash Karp 法」について、その具体的に値を求める手順と、その各々の特徴（長所，短所）を調べよ。

それでは、実際に式（20）に基づいて LCR 回路の過渡電流の値を求めてみよう。各自、ブランク VI をあらためて用意し、図 70, 71 に示すプログラム「LRCRunge.vi」を作成してみよ。

このプログラム構成の特徴としては、以下のような点が挙げられる。

1）数値解を得るために ODE4 次 ルンゲクッタ法  を用いる。

2）パラメータの記号とその値を格納する文字列配列「Substitution Rules」は、まず制御器パレットから「配列」を取り出してフロントダイアグラム上に配置し、その中に「クラスタ」を入れ、さらにその中に文字列関数を 2 つ入れて作成されている。これによってクラスタの配列が形成され、パラメータの値が変数置換の関数  に使用できるようになる。

3）時刻 $t=0$ における回路の初期条件、$I_t(0)=0$ ， $\dfrac{dI_t(0)}{dt}=\dfrac{E_0}{L}$ は、入力パラメータから自動的に代入される。

プログラムの内容を確認したら、各パラメータ[ C, R, L, V($E_0$ のこと), m($\omega$ のこと)]の値を代入して、直流(交流)RC 回路において電流値がどのように推移するかを観察せよ。その際のパラメータの値は、問題６－１の中で挙げた一覧のものを使用せよ。また、コイルの値に関しては、

● コイル L（単位はヘンリー）： 0.1m, 1m, 10m, 33mH

の系列を使用せよ。

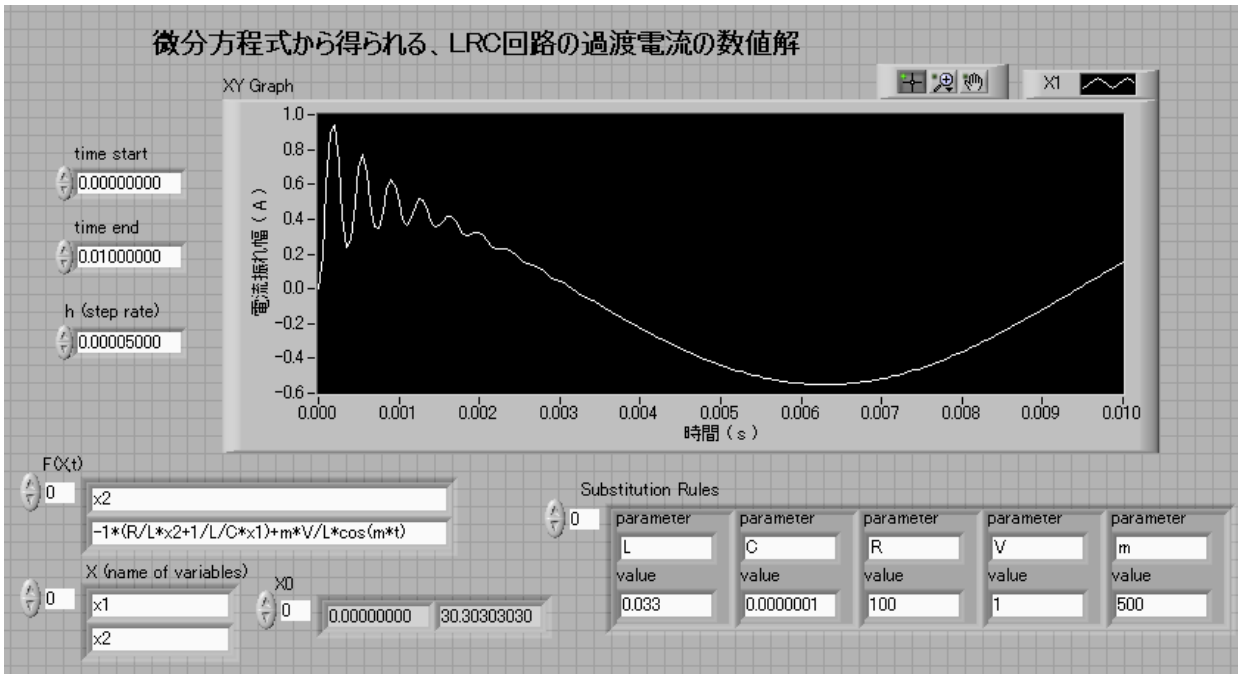



図 70：CR 回路を流れる電流のシミュレーションを行うプログラム「LRCRunge.vi」のフロントパネル.

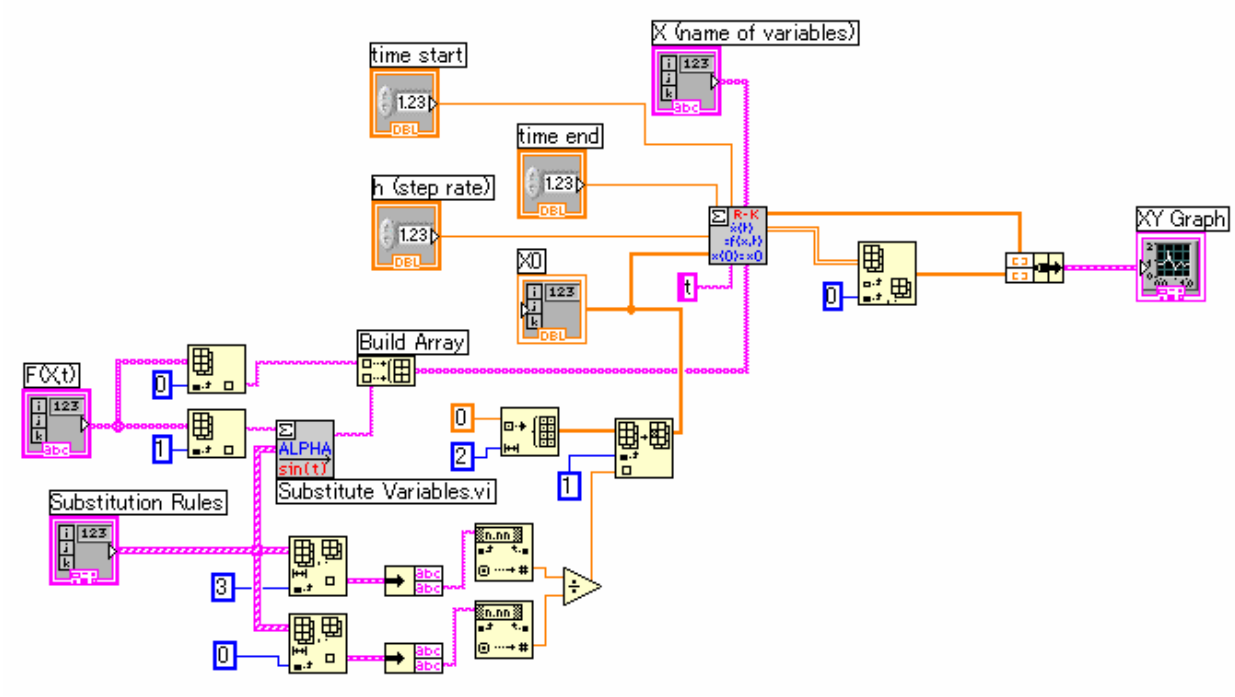



図 71：図 70 に対応するブロックダイアグラム.

## RC 回路の微分方程式の数値解を求める

直流 RC 回路（ 直流電圧源は $E_0$ とする ）の過渡電流 $I$ を表す 1 階常微分方程式は、

$$R\frac{dI}{dt}+\frac{1}{C}I=0 \qquad (39)$$

であり、 $x_1=I$ として、 $\dot{x}_1=-\frac{1}{RC}x_1$ と表される。

これにより、$I$ の数値解を得るプログラムを作成してみよう。参考として、図 72 と 73 に示すようなプログラム「RC 数値線形.vi」を挙げておく。

プログラム構成の特徴としては、以下のような点が挙げられる。

1）関数パレットの「関数→全関数→解析→数学→微分と積分→微分方程式」にある、「ODE 数値線形システム」の VI を使用する。

2）時刻 $t=0$ における回路の初期条件 $I_t(0)=\frac{E_0}{R}$ は、入力パラメータから自動的に代入される。

3）実行した結果得られる数値解 $I$ の時系列データを、スプレッドシート形式のファイルに保存できるようにする。

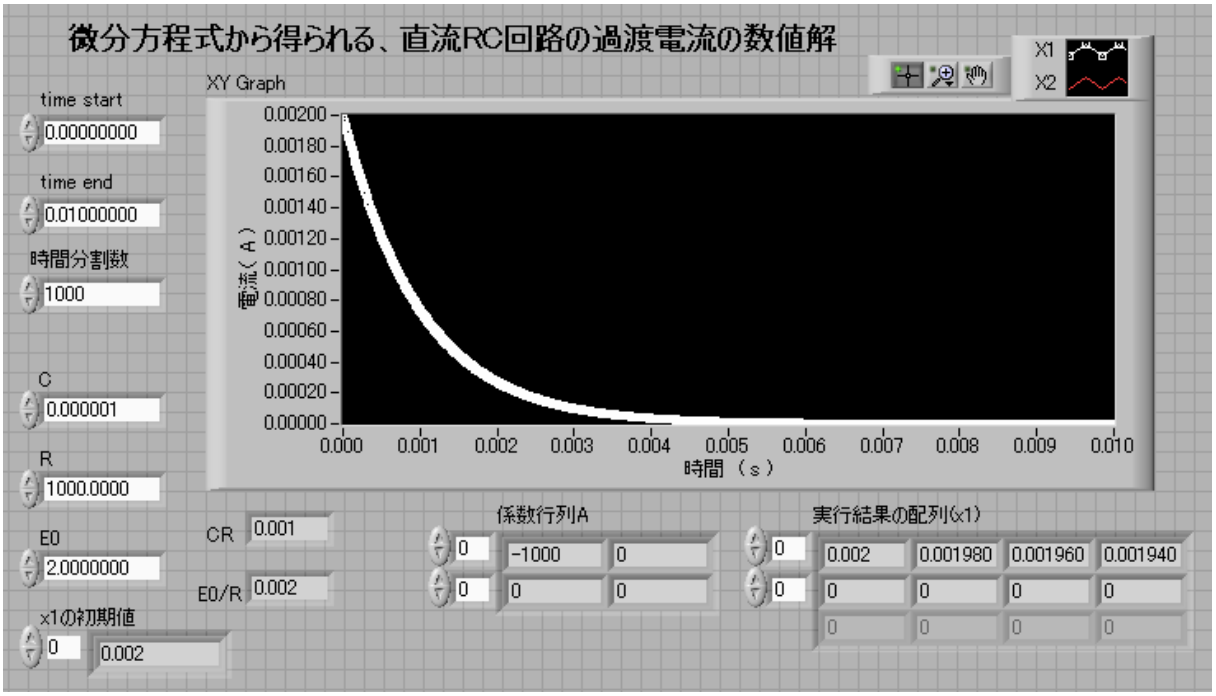



図 72：直流 RC 回路の微分方程式(39)から過渡電流の数値解を得る
プログラム「RC 数値線形.vi」のフロントパネル.

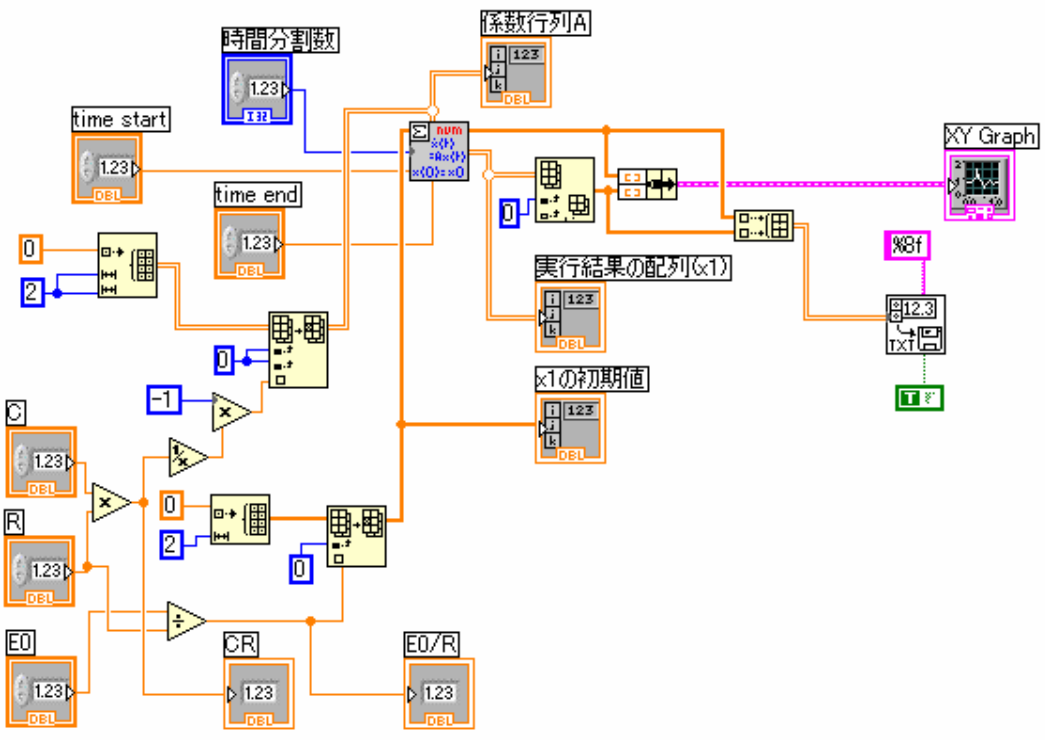



図 73：図 72 のプログラムのブロックダイアグラム.

実際にプログラムを動作させ、得られた過渡電流の数値解データをスプレッドシート形式のファイルに保存せよ。また、問題６－１で、微分方程式の解析解を用いた波形の表示を行ったが、この解析解に同じ C, R のパラメータを代入して実行し、その結果をこの数値解の結果と比較してみよ。

## 波形データのフィッティング（最小２乗法）

電気回路を作成し、その特性を確認する際には、そこで測定し記録した実際の過渡電流の値の時系列データに対して、妥当なモデル関数によるフィッティングを行う必要がある。LabVIEW プログラムは、このような実験で得られる計測データやプログラム内部で発生させた信号データに対してフィッティングを行う機能を持った関数 VI を幾つか有している。ここでは、その中でも設定が容易に行える Express VI の関数を使用して、読み込んだデータファイルに対しフィッティングを行うことにする。

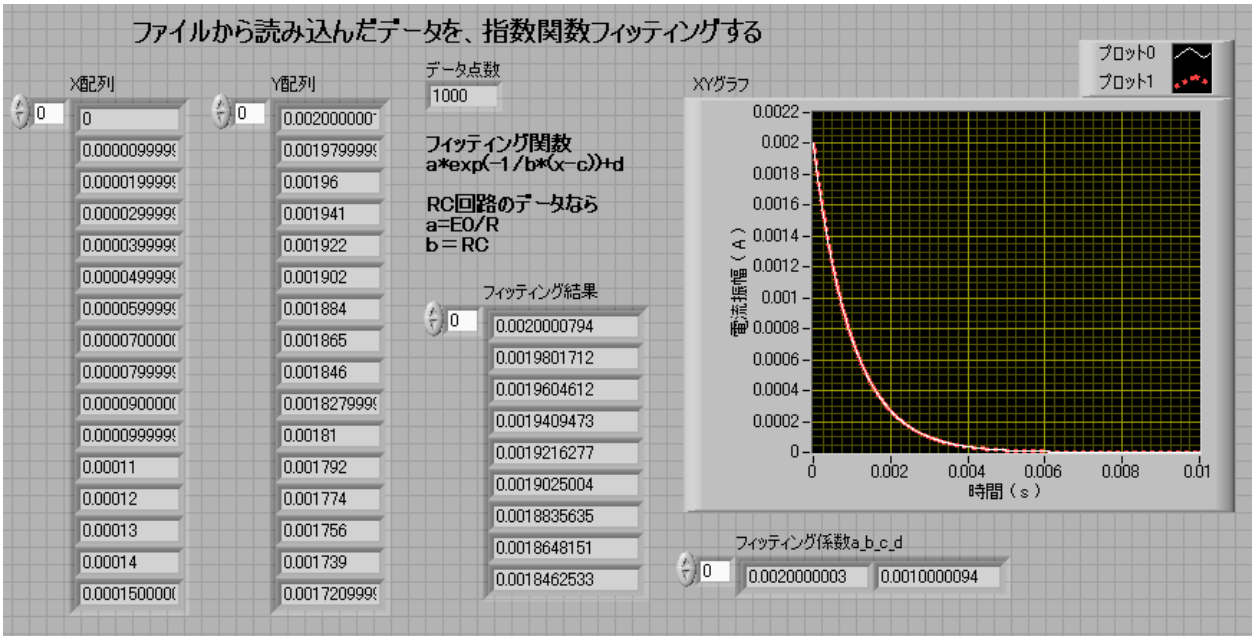



図 74：データファイルを読み込み，それを指数関数でフィッティングするプログラム「FittingRC0.vi」のフロントパネル.

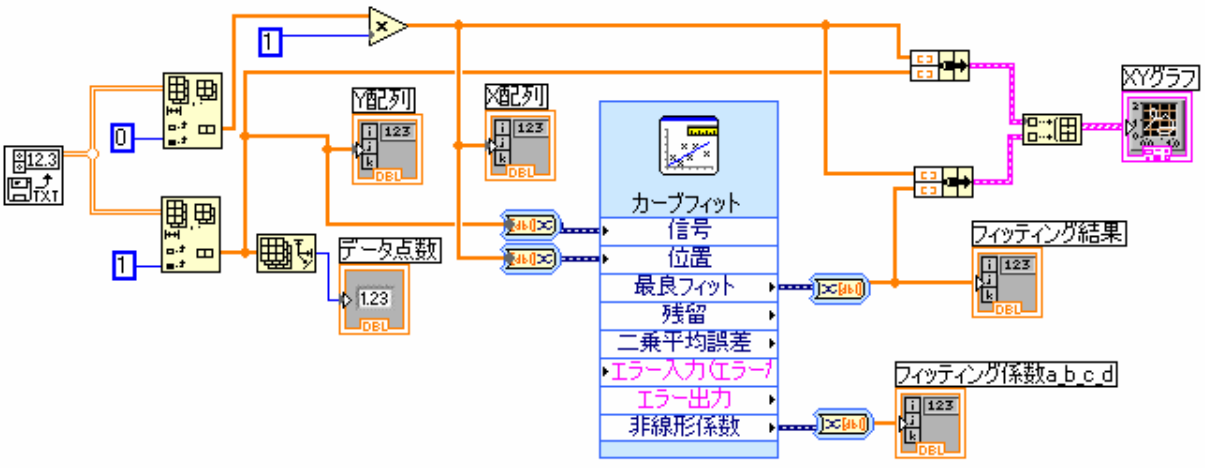



図 75：図 74 のプログラムのブロックダイアグラム.

このプログラム構成の特徴としては、以下のような点が挙げられる。

1）関数パレットの「関数→全関数→Express→解析」にある、「カーブフィット」の VI を使用する。
※　これは、最小二乗法により、データに対して指定した関数のフィッティングを行って関数のパラメータを決定するための VI である。
2）スプレッドシート形式のファイルから時系列の数値データを読み込む。
3）「カーブフィット」VI のプロパティを開くと、図 76 のような画面が現れる。今、データに対してフィッティングしたい関数は、指数関数であるので、モデルタイプは「非線形」を選択し、「非線形モデル」の欄にパラメータを含んだ形の関数を記入する。ここでは、指数関数の大きさを a、時定数を b で現している。また、フィッティングに際しては、適切なパラメータの初期値を与えることが必要であり、これは「初期推定値」の欄に記入する。

※　パラメータの初期値が著しく実態と異なっていると、フィッティングの結果が収束しないので注意せよ。あらかじめ波形を確認しておき、見当の付くパラメータに関しては、出来るだけ正確に値を見積もっておく必要がある。

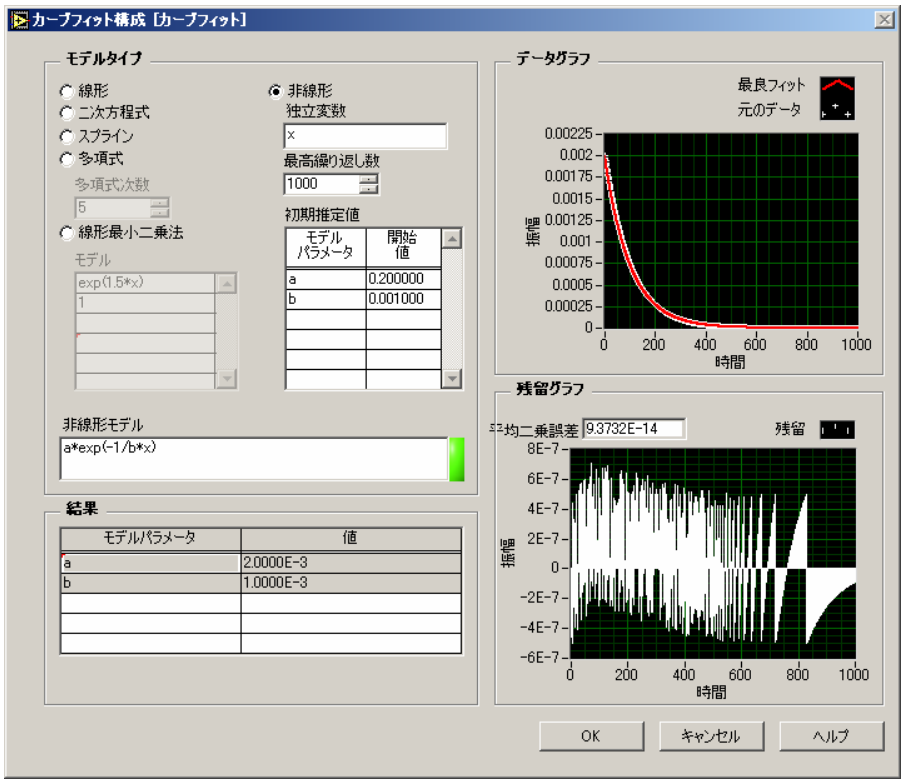



図 76：Express VI「カーブフィット」のプロパティ画面.

図 72 のプログラムで与えたパラメータ C, R, $E_0$ の値と、パラメータ a，b の値の間には、(30)式より、$a=E_0/R$，$b=CR$ の関係がある。図 72 のプログラムを実行してデータファイルを作成し、それをこのプログラム「FittingRC0.vi」で読み込んで a, b の値のフィッティング結果を見てみよ。そして実際に式(30)が正しい値を与えているか、パラメータ C, R, $E_0$ を幾つか変えてみて、それぞれ確認せよ。

# 第七回　計測器を用いた実験１

## ブレッドボード上の回路形成

本講義で行う LCR 回路の実験では、図 77 のようなブレッドボード（サンハヤト：SAD-01, 10）を使用する。これにより、標準の 2.54mm ピッチのパーツ(IC, LSI, LED, 抵抗, コンデンサなど)を半田付けなしに取り付けた、簡単な電子回路を組み立てることが出来る。

CR 回路を形成するためには、まず波形発生器(Function Generator)からの信号をバナナピンジャック端子に接続し、抵抗及びコンデンサを図 76 の様に接続する。端子間の配線はジャンパ線を使用して行い、信号出力はジャンパ線を用いて空いているバナナピンジャック端子に接続し、そこから USB-6008 に接続して電圧値を PC で取り込めるようにする、もしくはオシロスコープでその値を読み取れるようにする。

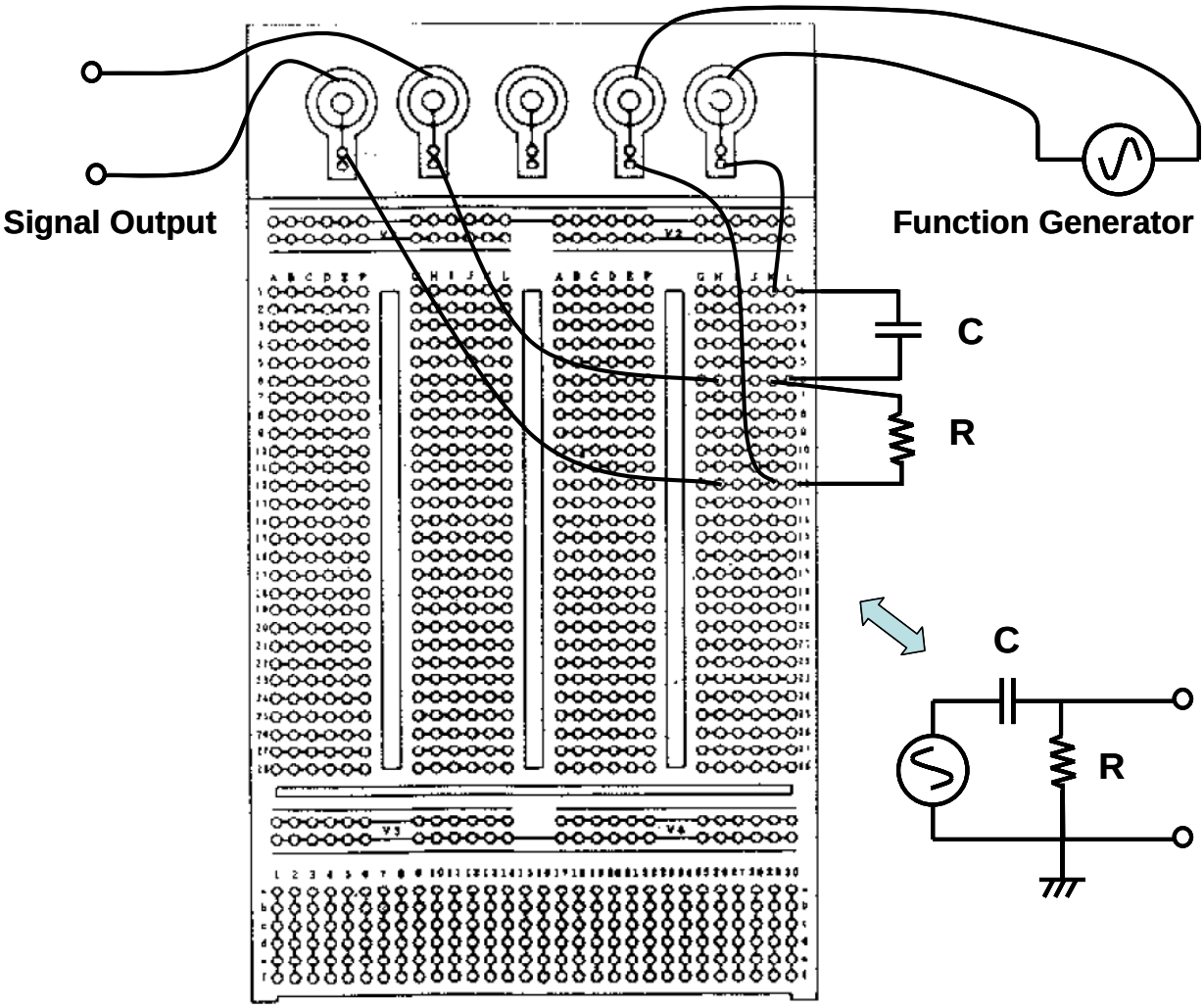
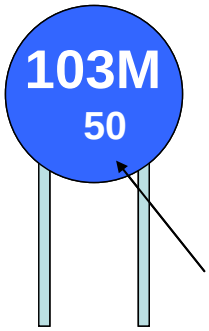



図 77：講義で使用するブレッドボードの外形．実線で結ばれている穴同士（例えば 1 の A-B-C-D-E-F 等）は電気的に接続されているので、それを利用して配線を行う．

## コンデンサの表示値の読み方




※ｐＦが基本

103
=10×10³
=10000　pF

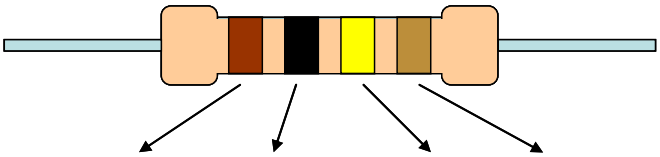
耐圧　50V

| 表示 | 変換値 | 単位 |
|---|---|---|
| 101 | 100 | ｐF |
| 102 | 1000 | ｐF |
| 103 | 0.01 | μF |
| 104 | 0.1 | μF |
| 474 | 0.47 | μF |
| 105 | 1.0 | μF |

誤差の表示　J：　5％以内
K：　10％以内
M：　20％以内

図 78：表示記号とコンデンサの値の各桁の数字および誤差範囲の対応.

## 抵抗器のカラーコードの読み方

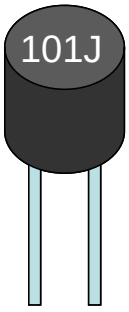


第1色帯　第2色帯　第3色帯　第4色帯

| 色 | 数値 | 数値 | 乗数 | 誤差 |
|---|---|---|---|---|
| 黒 | 0 | 0 | ×1 | ― |
| 茶 | 1 | 1 | ×10¹ | ± 1% |
| 赤 | 2 | 2 | ×10² | ± 2% |
| 橙 | 3 | 3 | ×10³ | ― |
| 黄 | 4 | 4 | ×10⁴ | ― |
| 緑 | 5 | 5 | ×10⁵ | ― |
| 青 | 6 | 6 | ×10⁶ | ― |
| 紫 | 7 | 7 | ×10⁷ | ― |
| 灰 | 8 | 8 | ×10⁸ | ― |
| 白 | 9 | 9 | ×10⁹ | ― |
| 金 | ― | ― | ×10⁻¹ | ± 5% |
| 銀 | ― | ― | ×10⁻² | ±10% |
| 無着色 | ― | ― | ― | ±20% |

図 79：カラーコードの色と抵抗値の各桁の数字および誤差範囲の対応.

## コイル（インダクタ）の表示値の読み方




※μＨが基本

101
=10×10¹
=100　μＨ

| 表示 | 変換値 | 単位 |
|---|---|---|
| 101 | 100 | μＨ |
| 102 | 1.0 | ｍH |
| 103 | 10.0 | ｍH |
| 333 | 33.0 | ｍH |

誤差の表示　J：　5％以内
K：　10％以内
M：　20％以内

図 80：表示記号とインダクタの値の各桁の数字および誤差範囲の対応.

実際の配線に先立ち、使用する電子部品であるコンデンサ、抵抗、およびインダクタ(コイル)の形と、それぞれの部品に記載されている表記の意味を、図 78～80 にまとめておく。1 桁目と 2 桁目が有効数字で、3 桁目が乗数である点はどれも共通しているので、後は基準の単位(コンデンサではｐF、コイルではμH、抵抗はΩ)さえ覚えておけばよい。抵抗のカラーコードは数字になっていないが、これも色と番号の対応を覚えておくと、今後何かと便利である。抵抗の種類によっては、カラーコードが 5 本のものもあるが、その場合には、有効数字が 2 桁から 3 桁に増えたものとして考えればよい。

最初に、オシロスコープで発振器自体の出力の確認を行うことにする。発振器の出力は、正弦波と矩形波のどちらかを選べるようになっているが、今回は矩形波を選択する。これにより、ステップ関数状の電圧波形を回路の入力に与えることができる。これは、時間 0 でスイッチをいれて直流電源から電流が流れ始める図 65 のような状況を、発振器の周波数で繰り返し生成させることに他ならない。

実際に図 77 に倣い、下記の項目に留意して発振器とブレッドボード、およびオシロスコープ間の配線を各自行ってみよ。

1）発振器の矩形波出力は、電圧振れ幅を ±1 V に、周波数を 70 Hz に設定。
2）発振器からの信号を、CR 回路の入力信号として、BNC 端子経由で取り込む。
3）抵抗 R の両端の電圧を、BNC 端子を使用して取り出し、これをオシロスコープに接続して確認できるようにする。
4）抵抗値は 100 kΩ、コンデンサは 0.01 μF とする。

矩形波の周波数で繰り返される、コンデンサ C と抵抗 R の両端電圧$V_C$と $V_R$の値の変化を確認せよ。その際、基準電位からの値を見られるように、C と R の位置を交換してそれぞれの電圧値をモニタすること。

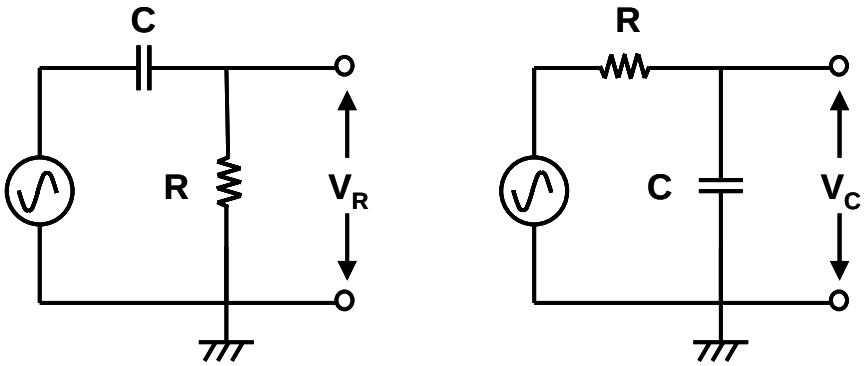



図 81：コンデンサと抵抗の端子間電圧$V_C$，$V_R$の測定方法.

オシロスコープの画面において観察される$V_C$と $V_R$の波形は、式(1)と式(30)に基づきそれぞれ図 82 のような形になるはずである。

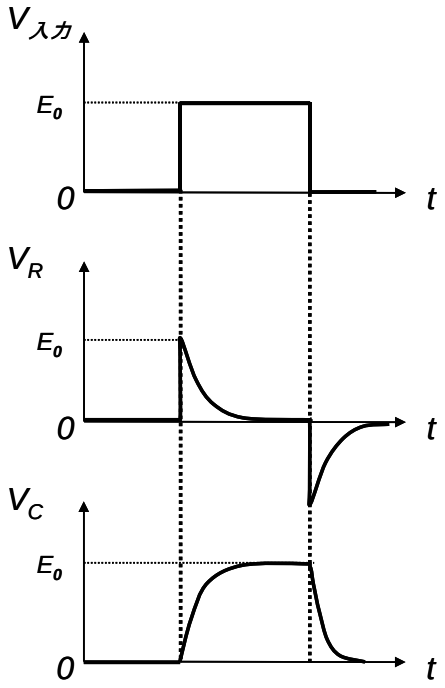



図 82：矩形波入力を与えた場合の、抵抗およびコンデンサ両端の電圧変化.

**課題７－１：**

コンデンサ C と抵抗 R の両端にかかる電圧$V_C$と $V_R$の値が図 82 のように変化する理由を、矩形波電圧の立ち上がりおよび立ち下がり部分それぞれについて考察せよ。

これまでの講義では、計測器 USB-6008 を用いて信号波形を時々刻々取り込むプログラムの作成を行ってきた。しかしながら、別紙 A にあるように、この計測器を用いたデータ取得の際のサンプリングレートは、あまり速いものではない。従って、この装置の性能限界まで高速にデータを取得するために、今後波形の取り込みには、これまでとは違った高速波形取り込みのためのプログラム、「アナログ入力過渡用.vi」を使用する（図 83 参照)。

このプログラムは、計測器側で一定の数のデータを一括して取得し、そのデータブロックを PC 側に転送するものである。ここでは、信号の入力端子としては、＋の ai0 端子（端子番号２）と GND（端子番号１）を用いる。ブレッドボード上の出力端子（バナナピンジャック端子）と、USB-6008 上のこれらの端子を互いに接続して、信号を取り込めるようにせよ。

また、このプログラム内の DAQ アシスタント VI の設定では、最初に以下のようにオプションを選択・設定せよ。

１）集録モードは、「N サンプル」を選択
２）クロック設定は、「読み取るサンプル数」を「100」、「レート」を「10k」Hz とする。
３）「端子設定」を「基準化シングルエンド」とする。

こうすると、DAQ アシスタント VI は、0.1 ms 刻みの時系列電圧データを取得して、それを出力できるようになる（今回は 100 点のデータ取得）。

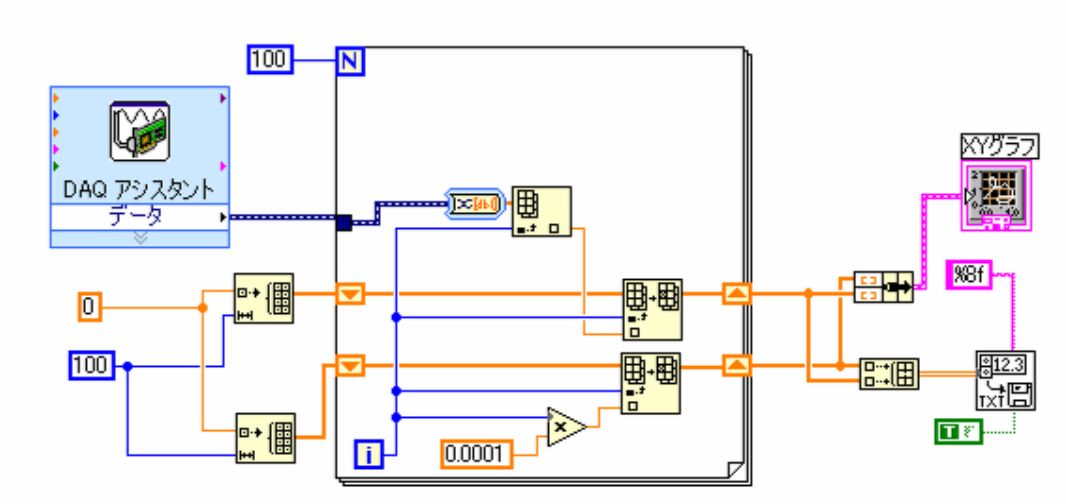



図 83：高速波形取り込みのためのプログラム「アナログ入力過渡用.vi」のブロックダイアグラム.
DAQ アシスタントで 0.1 ms 刻みの時系列データを取得した後、それを表示すると共に
スプレッドシートファイルに保存する.

■　CR 回路の時定数が、$CR$ = 0.01 μF × 100 kΩ = 0.001 s　の場合の例　■

0.1 ms 単位で 100 点取得し、データ全体の時間範囲が 10 ms である現在の取り込み設定において、図 81 のような電圧データ波形を得るためには、発振器の繰り返し周波数を、1/(0.010～0.020)=100～50 Hz の間にすると良い。こうすれば、一つの波形構造（立ち上がり、または立ち下がり部分）のみがデータ取得範囲に入り、得られたデータの解析が行いやすくなる。このように、回路の時定数に応じて、適宜測定時間領域および繰り返し周波数の設定を行う必要がある。

※下記の図 83～86 は、発振器の繰り返し周波数を 70 Hz にして取得したものである。

なお、オシロスコープで波形をモニタする際には、信号波形の変化する部分の電圧値に基準を定め、時間基準を設定することで、毎回同じ波形を画面上で見ることが出来る。これに対し、USB-6008 で今回使用するプログラム「アナログ入力過渡用.vi」の場合には、このようなタイミング基準の信号を用いることが出来ないので、毎回のデータ取得開始位置が異なってくる。したがって、データの取得に際しては、適当な波形が得られるまでプログラムを何回か実行せよ。

最初に、上記のように設定した場合の抵抗電圧$V_R$の立ち上がり部分を、プログラム「アナログ入力過渡用.vi」を用いて取得せよ。すると図 84 のような波形が得られるはずである。

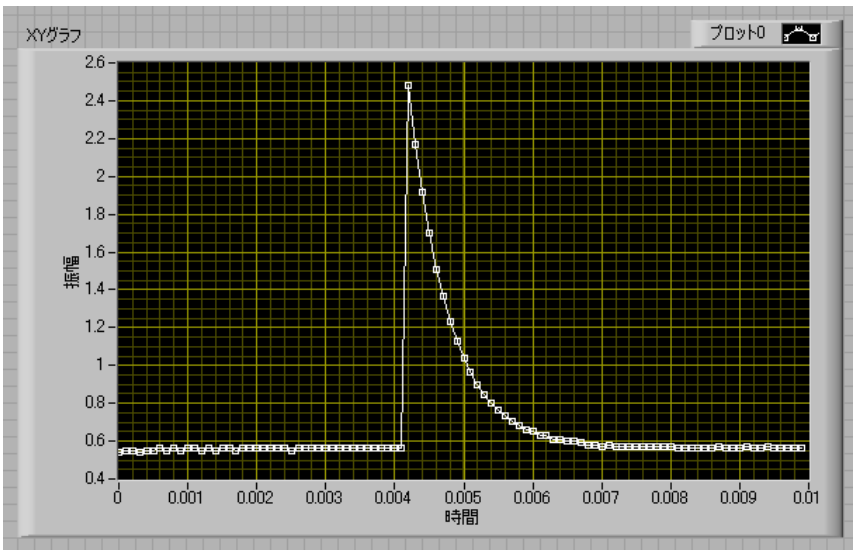



図 84：CR 回路における、$V_R$の測定データ（$R$=100 kΩ, $C$=0.01μF）.

この波形に対してフィッティングを行い、パラメータを評価することを試みる。ここで使用するフィッティングプログラム「FittingRCVr.vi」は、指数関数を用いて$V_R$の立ち上がり部分のフィッティングを行うためのプログラムである。指数関数だけでは、時間原点前の信号に対しフィッティングがうまく行かないので、プログラム中では、**階段関数**を掛けて時間原点前まで使えるようにした以下の改良式(41), (42)を、フィッティング関数として用いている。各自、このプログラムのブロックダイアグラムを開き、「カーブフィット」VI のプロパティを見て、これを確認せよ。

階段関数　（$t<0$ で 0、$t\geq 0$ で 1）

$$f(t)=\frac{(|t|+t)}{2|t|} \tag{40}$$

抵抗電圧$V_R$の立ち上がり部分

$$V_R(t)=E_0 f(t)e^{\frac{-1}{RC}t} \tag{41}$$

なお､取得データにおける信号の立ち上がりの時間原点は､上述のように測定毎にずれているので､時間原点補正のフィッティングパラメータ（プログラム中では c の値）を調整する必要がある。

プログラム「アナログ入力過渡用.vi」で取得したデータに対し、実際にこのプログラムを用いて波形のフィッティングを行った結果を図 85 に示す。

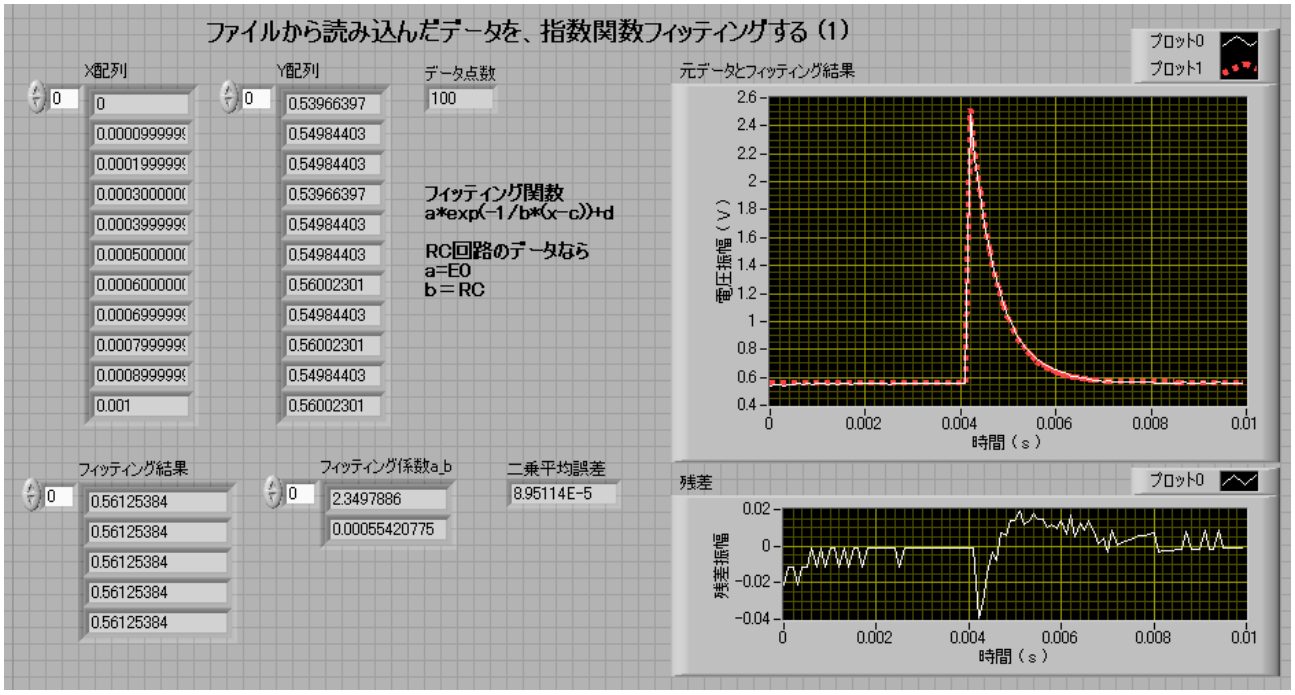



図 85：CR 回路における、$V_R$ の測定データに対してフィッティングを
行うプログラム「FittingRCVr.vi」と、その実行結果.

各自、最小二乗法による二乗平均誤差の大きさが、図 85 にあるように $10^{-4}$ から $10^{-5}$ 程度まで減少するまで、初期パラメータ値 a，b，c，d を変えてフィッティングを行い、この $V_R$ のデータから得られる時定数の値を確認せよ。

次に、コンデンサ C と抵抗 R の位置を入れ替えて抵抗電圧 $V_C$ を測定できるようにし、その立ち上がり部分を、プログラム「アナログ入力過渡用.vi」を用いて取得せよ。すると図 86 のような波形が得られるはずである。

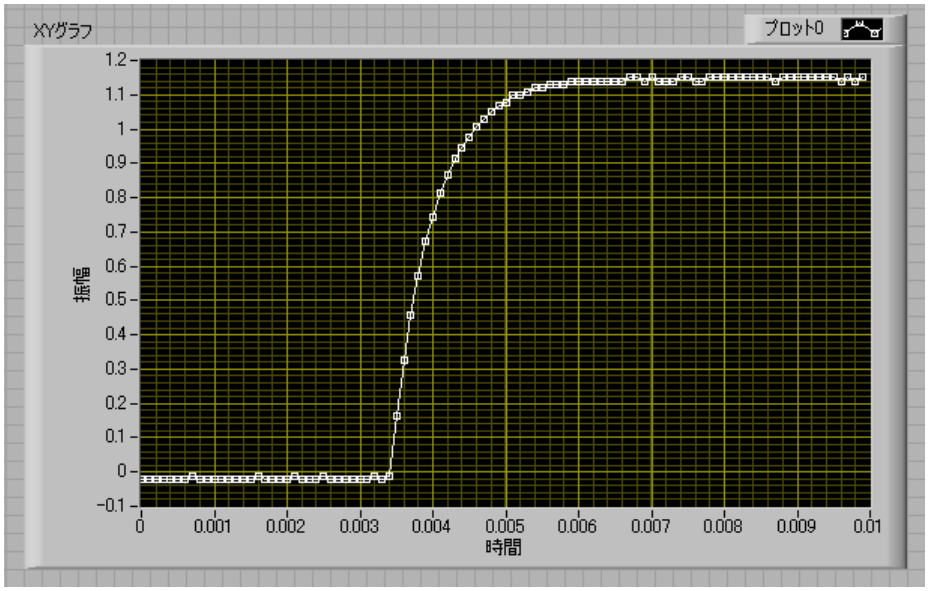



図 86：CR 回路における、$V_C$ の測定データ（$R$ =100 kΩ，$C$ =0.01 μF）.

この波形に対してもフィッティングを行い、各パラメータを評価することを試みる。ここで使用する　フィッティングプログラム「FittingRCVc.vi」は、指数関数を用いて$V_C$の立ち上がり部分のフィッティングを行うためのプログラムである。ここでも、指数関数だけでは時間原点前の信号に対しフィッティングがうまく行かないので、階段関数を掛けて時間原点前まで使えるようにした以下の改良式(42)を、データのフィッティング関数として用いる。

コンデンサ電圧$V_C$の立ち上がり部分　　$$V_C(t) = E_0 f(t)(1 - e^{\frac{-1}{RC}t}) \qquad (42)$$

プログラム「アナログ入力過渡用.vi」で取得したデータに対し、このプログラムを用いて波形フィッティングを行った結果を図 87 に示す。

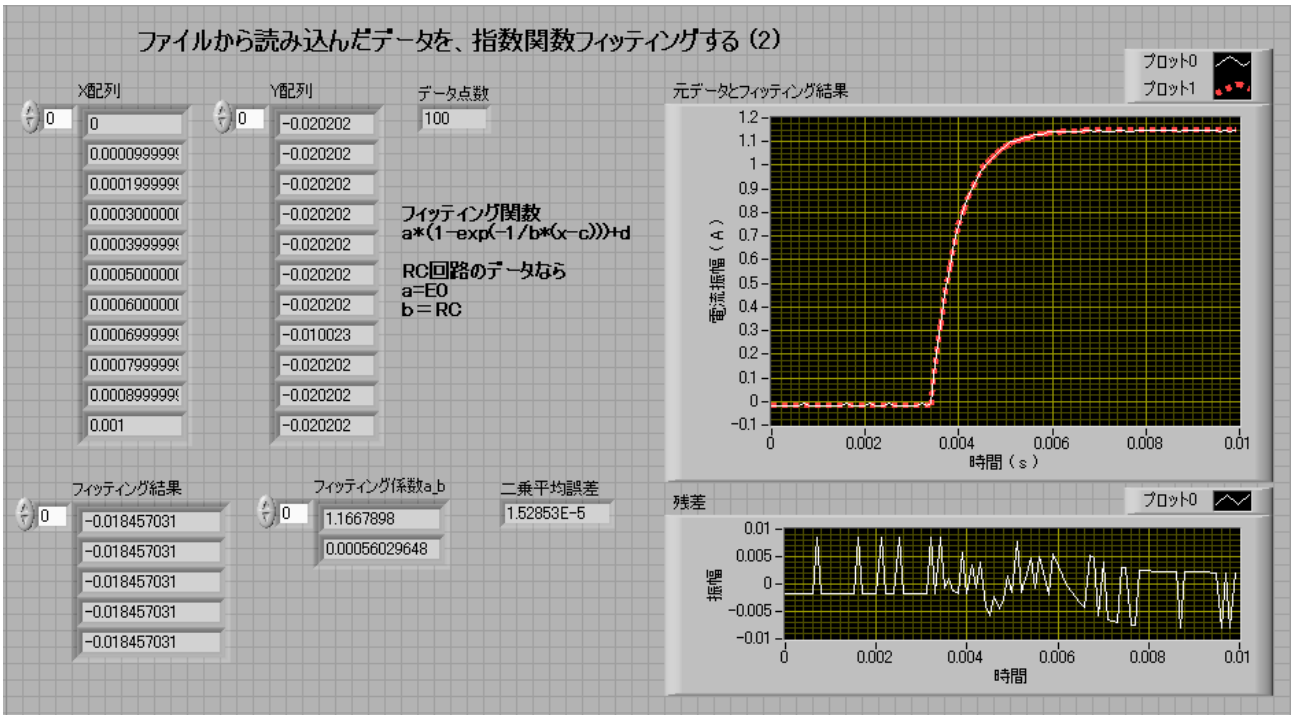



図 87：CR 回路における、$V_C$の測定データに対してフィッティング
を行うプログラム「FittingRCVc.vi」のフロントパネル

## 課題７－２：

各自、最小二乗法による二乗平均誤差の大きさが、$10^{-4}$から$10^{-5}$の程度になるまで、初期パラメータ値 a，b，c，d を変えてフィッティングを行い、この$V_C$のデータから得られる時定数の値を確認せよ。

**課題７－３：**

この回路の$V_C$と$V_R$のデータに対するフィッティングから、それぞれの時定数を求められたら、引き続き下記の場合の時定数の実測値をそれぞれ求めて、ノートに記録せよ。その際に、「アナログ入力過渡用.vi」の DAQ アシスタント VI 内でのデータ取得時間範囲の設定、および発振器の繰り返し周波数の調整を適宜行うこと。

１）時定数が、$CR$ = 0.01 μF × 100 kΩ = 1 ms の場合
２）時定数が、$CR$ = 0.01 μF × 470 kΩ = 4.7 ms の場合
３）時定数が、$CR$ = 0.01 μF × 1 MΩ = 10.0 ms の場合

■ フィッティング結果の考察 ■

各$C$と$R$の値に対し、DAQ を用いて測定した$V_R$と$V_C$のデータにフィッティングを行った結果得られた時定数の値は、そのときの$C$と$R$の積$CR$から予想される時定数の値とはおそらく異なっているはずである。これは何故であろうか？

図 88 に示すように、この計測器を接続する前と計測器を接続した後で、実効的な抵抗の値が異なる。 回路内の抵抗値を$R$、計測器の内部抵抗を$R_{in}$と置いたとき、実効的な抵抗は、これら 2 つの抵抗を並列接続したもの $R_X$ になる。

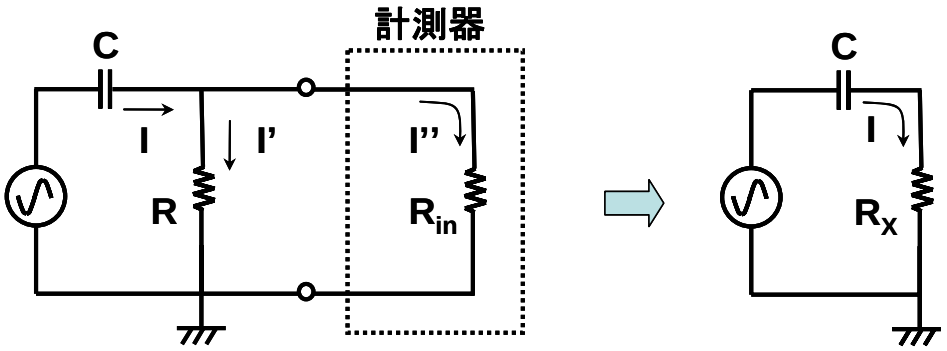



図 88：計測器の等価回路図（左）と計測器が無い場合の回路図(右).

この場合、$R_X$ の値は、

$$R_X = \frac{RR_{in}}{R + R_{in}} \qquad (43)$$

で与えられる。この式から分かるように、計測器において、伝達する電圧レベルに影響を与えない理想的な入力抵抗は、$R_{in} = \infty$ ということになる。

別紙 A に挙げた USB-6008 の仕様書を見ると、この計測器の入力抵抗の値が 140 kΩ であることが分かる。すると、例えば$C$ = 0.01 μF , $R$ = 100 kΩ （ $CR$ = 0.001 s ） の場合には、$R_X$ = 59 kΩ となり、回路の時定数は $CR_X$ = 0.59 ms となる。

※　入力抵抗が 1 MΩ の場合には $R_X$ = 91 kΩ 、$CR_X$ = 0.91 ms となって、$CR$ = 0.001 s の値に近くなる。

通常の計測器は、この理想値に近づけるために、十分に高い入力抵抗の値（数 MΩ）を持っているので、回路内部の抵抗が低い場合には、ほぼ回路内の実際の電圧値と同じ値が計測される。このように、計測器の入力抵抗は、測定対象内を伝わる信号の電圧レベルに影響を与えるので、電圧の測定を行う際に、計測器の入力抵抗を意識することは重要なことである。

**課題 7－４：**

課題７－２と７－３で得た、$C$ と $R$ の各組み合わせに対する $V_R$ の測定において、フィッティングにより得た時定数の実測値が、それぞれ妥当なものになっているかどうか、入力抵抗を考慮した計算を行って確かめてみよ。

**課題 7－５：**

コンデンサ $C$ と抵抗 $R$ の位置を入れ替え、$V_C$ の値を測定する配置にした場合の、入力抵抗の回路に与える影響について考察せよ。その際、入力抵抗 $R_{in}$ を流れる電流を $I_1$、コンデンサ $C$ に流れる電流 $I_2$ として、それぞれによる電圧降下を考え、抵抗 $R$ を流れる電流 $I$ が、$I = I_1 + I_2$ であることを利用して微分方程式を立てよ。コンデンサ $C$ と抵抗 $R$ の値として、例えば課題７－３のような場合を考えたとき、この回路の時定数はどのようになるか？実際に時定数を計算してみて、その値と課題７－３で得られた結果とを比較してみよ。

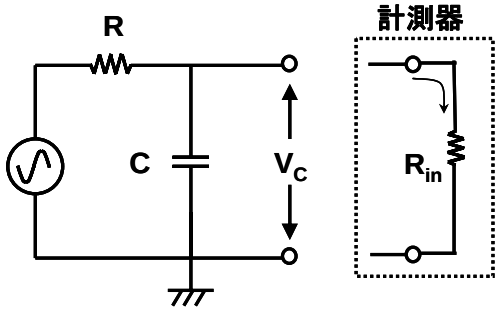



図 89：　CR 回路のコンデンサ電圧 $V_C$ を内部抵抗 $R_{in}$ の計測器で測定.

# 第八回　計測器を用いた実験２

第六回の講義では、直列 LCR 回路を流れる電流に関する微分方程式(3)の一般解は、$\beta=\sqrt{R^2-4L/C}/2L$ の値が（ｉ）実数、（ⅱ）0、および（ⅲ）虚数の各場合に応じて、それぞれ式(11)、(14)、(17)で与えられることを示した。これらの各条件は、LCR 回路の特性値$Q$を

$$Q=\frac{1}{R}\sqrt{\frac{L}{C}} \qquad (44)$$

このように定義すれば、$Q<1/2$ 、$Q=1/2$ および $Q>1/2$ と表現し直すことが出来る。すなわち、$Q=1/2$を境に過渡電流の挙動が、図 66 のように様変わりすることになる。振動することなく指数関数的に信号強度が減衰する（ｉ）$Q<1/2$の場合の緩和過程は、「過減衰」、また信号が振動しながら減衰する（ⅲ）$Q>1/2$のような場合は「減衰振動」、そして両者の中間である（ⅱ）$Q=1/2$の場合は「臨界減衰」と呼ばれている。この中で、過渡的緩和が「臨界制動」の状態にあるような電子回路は、我々の身近に数多く用いられており、

1）自動ドアの開閉動作：　自動ドアは、なるべく迅速かつスムーズに、かつバタバタと振動しないように閉まらなければならない。開閉用のモーターを、臨界減衰状態の電流で制御すれば、最適な動作を行える。
2）蓄電器への充放電：　コンデンサをはじめとする蓄電器への急速な充電および放電を行う際には、回路内部の抵抗およびインダクタンスが無視できない。このような場合に、適当な内部抵抗の調整を行って臨界制動の状態にすれば、回路電流の振動を無くして迅速な充電や放電が行える。

こうした緩和過程を積極的に制御する応用用途に対し、非常に重要なものとなっている。

さて、講義で用意している$R$，$C$，$L$の各素子の種類は

1）抵抗$R$（オーム）：　100，1k，10k，100k，470k，1MΩ
2）コンデンサ$C$（ファラッド）：　0.01μ，0.1μ，0.47μ，1μF
3）コイル$L$（単位はヘンリー）：　0.1m，1m，10m，33mH

以上のようになっている。ここで、例えばインダクタ（コイル）の値を 33 mH、コンデンサ容量$C$の値を 0.01μF に固定して、抵抗$R$の値を変化させたとき、どのような $R$ の値で臨界制動の状態が得られるかを確認せよ。その値の近辺で$R$の値を幾つか変え、対応する $\tau_L(=L/R)$，$\tau_C(=CR)$，$\tau_0(=\sqrt{LC})$，$Q$の値をそれぞれ一覧表にせよ。一覧表が出来たら、それらの値を図 70 のシミュレーションプログラム「LRCRunge.vi」に入れ、実際に LCR 回路を流れる電流をシミュレートしてみよ。

※　このとき、シミュレートする電流波形の時間幅と時間刻み、およびグラフの縦と横軸のスケール等は、適宜調整して見やすくすること。

## 課題８－１：

直流 LCR 回路においては、臨界制動の条件が満たされた場合に、信号が最も早く減衰すると言われている。このことを、臨界制動を与える（$R$，$C$，$L$）組の近傍で、$R$ の値だけを少し変えたシミュレーション計算を何組か行い、実際に確認せよ。そのための手順は、

１）プログラム「LRCRunge.vi」を改良し、計算結果をスプレッドシートファイルに保存できるようにする。このプログラムに新たに名前を付け（例えば LRCRunge01.vi)、保存せよ。
２）$R$ の値だけを変えて、過減衰、振動減衰、および臨界制動の各３パターンの結果をファイルに記録する。
３）これら３つのデータを読み込んで、１つの XY グラフに重ね書きするプログラムを作成する。
４）プログラムを実行して実際にファイルを読み込み、重ね書きした画面を表示させる。
５）表示画面を最前面にしておき、「Alt」キーと「PrintScreen」キーを押して Windows のバッファにこの画面を取り込ませる。その後、WindowsXP の左下にある「スタート」ボタンから、「全てのプログラム」→「アクセサリ」→「ペイント」と進んでこのプログラムを起動し、そこで、新規ファイルにこの内容を貼り付ける（「編集」の「貼り付け」で可能)。このファイルを「Problem8_1Answer.bmp」の名で保存せよ。

このようになる。

## 補題：

$R$，$C$，$L$ の素子の値として、講義で用意している上記のものを使用する場合、計測器 USB-6008 でこの臨界制動の様子を観測することは可能か、それとも不可能か？具体的な根拠を示してこの可能性を論じよ。

## 課題８－２：

課題８－１の前提である、「過減衰、臨界制動、減衰振動のうち、固有振動数 $\omega = 1/\sqrt{LC}$ が同じなら臨界制動が一番速く減衰する」という命題が正しいかどうかを、微分方程式(3)の解(6)～(8)(または(11)，(14)，(17))から解析的に考察せよ。

※　参考図書：
１）過渡現象の基礎：　(著)吉岡 芳夫，作道 訓之，森北出版 (2004/09)
ISBN-13: 978-4627735514
２）常微分方程式と過渡現象の解析：(著)田中 久四郎，電気書院 (2000/08)
ISBN-13: 978-4485429280
等、「過渡現象」のキーワードで検索した図書を参考に考察を行うと良い。

**課題８－３：**

これまでの各章の内容において、判り難かった点や不明だった点をそれぞれ理由も含めて具体的に列挙せよ。また、初めてLabVIEW言語を用いてプログラム作成を行おうとするとき、その言語学習の過程で障害になると考えられる事柄は何か？重要度の順に３つ列挙して、その根拠も述べよ。今後の参考のため、講義内容として更に希望する項目があれば、それらを動機と共に挙げて述べよ。

**追記**

１）　電気回路においては、振動電流の有するエネルギーは、やがて抵抗加熱等の他のエネルギーに変化し散逸してゆく。一方、振動する運動系の有するエネルギーは、時間の経過と共に摩擦力等のエネルギーに変化し、消費されて散逸してゆく。通常、このエネルギー消費は、振動の速度の関数として与えられる抵抗力によって生じているものと考えてよい。一般に、速度に比例した抵抗力がある場合には、質量$m$でばね定数$K$を持つ振動子の平衡位置近傍での微小変位の運動方程式は、抵抗の係数を$J$として、

$$m\frac{d^2x}{dt^2} = -Kx - J\frac{dx}{dt} \qquad (45)$$

と書ける。この式は $m = L$， $K = 1/C$， $J = R$ と考えれば、LCR 回路を流れる電流に関する微分方程式(3)と全く同一の形になっている。すなわち、これまで述べてきたような、過渡的な回路電流に関する数学的な取り扱いは、そのまま振動子の運動系にも適用できる。なお、この振動系の抵抗力の無い場合の固有振動数$\omega$は、$\omega = \sqrt{K/m}$で与えられるが、これは電子回路においては、LC 回路の固有振動数$\omega_{LC} = 1/\sqrt{LC} = 1/\tau_0$ に対応している。

※　なお、LR 回路の時定数$\tau_L (= L/R)$，CR 回路の時定数$\tau_C (= CR)$, および LC 回路の時定数$\tau_0 (= \sqrt{LC})$の間には、$\tau_0 = \sqrt{\tau_L \tau_C}$ の関係が成り立っている。

２）　これまで、図 65 のような回路を流れる電流$I(t)$の値は、式(3)のような微分方程式をそのまま解いて求めていた。これに対し、関数$I(t)$に

$$I(s) = \int_0^\infty I(t)e^{-st}dt \qquad (46)$$

なる変換を施して得られる関数$I(s)$をもとに、$d^2I/dt^2$や$dI/dt$といった各項を変換した式

$$LsI(s) + RI(s) + \frac{1}{Cs}I(s) = 0 \qquad (47)$$

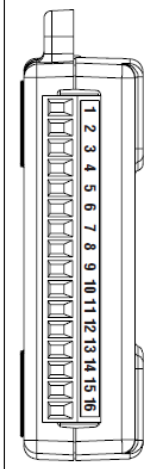
を解き、その結果得られた$I(s)$に式（46）の逆変換の式

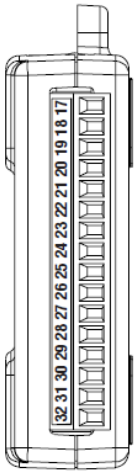
$$I(t) = \frac{1}{2\pi i}\int_{r-i\infty}^{r+i\infty} I(s)e^{st}ds \qquad (48)$$

を施して、$I(t)$の元の関数形を求めるという手法がある。これは**ラプラス変換法**と呼ばれており、微分・積分を変換によりそれぞれ掛け算・割り算にするものである。ラプラス変換は、このように微分方程式を代数方程式の問題とし、微分方程式の解法を容易にすることができる便利な手法である。各自、CR 回路、LCR 回路について、この手法で解を導いてみよ。

# 別紙 A

Data Acquisition (DAQ) Device NI USB-6008 の端子図および入出力部のスペック一覧。

| Module | Terminal | Signal, Single-Ended Mode | Signal, Differential Mode |
|---|---|---|---|
|  | 1 | GND | GND |
| | 2 | AI 0 | AI 0+ |
| | 3 | AI 4 | AI 0– |
| | 4 | GND | GND |
| | 5 | AI 1 | AI 1+ |
| | 6 | AI 5 | AI 1– |
| | 7 | GND | GND |
| | 8 | AI 2 | AI 2+ |
| | 9 | AI 6 | AI 2– |
| | 10 | GND | GND |
| | 11 | AI 3 | AI 3+ |
| | 12 | AI 7 | AI 3– |
| | 13 | GND | GND |
| | 14 | AO 0 | AO 0 |
| | 15 | AO 1 | AO 1 |
| | 16 | GND | GND |

| Module | Terminal | Signal |
|---|---|---|
|  | 17 | P0.0 |
| | 18 | P0.1 |
| | 19 | P0.2 |
| | 20 | P0.3 |
| | 21 | P0.4 |
| | 22 | P0.5 |
| | 23 | P0 6 |
| | 24 | P0.7 |
| | 25 | P1.0 |
| | 26 | P1.1 |
| | 27 | P1.2 |
| | 28 | P1.3 |
| | 29 | PFI 0 |
| | 30 | +2.5 V |
| | 31 | +5 V |
| | 32 | GND |

## Analog Input

Analog inputs ..... 8 single-ended

Input resolution ..... 12 bits differential,
11 bits single-ended

Max sampling rate ... 10 kS/s

Input impedance….. 144 kΩ

Timing resolution..... 41.67 ns (24 MHz time base)

Input range ..... ±10 V

Working voltage..... ±10 V

## Analog Output

Analog outputs..... 2

Output resolution ..... 12 bits

Maximum update rate... 150 Hz, software-timed

Output range ..... 0 to +5 V

Output impedance..... 50 Ω

Output current drive.....5 mA

Slew rate..... 1 V/µs

Absolute accuracy (no load) .....7 mV typical,
36.4 mV maximum at full scale

## Digital I/O

Digital I/O P0.<0..7> .... 8 lines / P1.<0..3> .... 4 lines

Output driver type..... Open collector (open-drain)

Compatibility..... TTL, LVTTL, CMOS

Absolute maximum voltage range ..... – 0.5 to 5.8 V with respect to GND

Pull-up resistor ..... 4.7 kΩ to 5 V

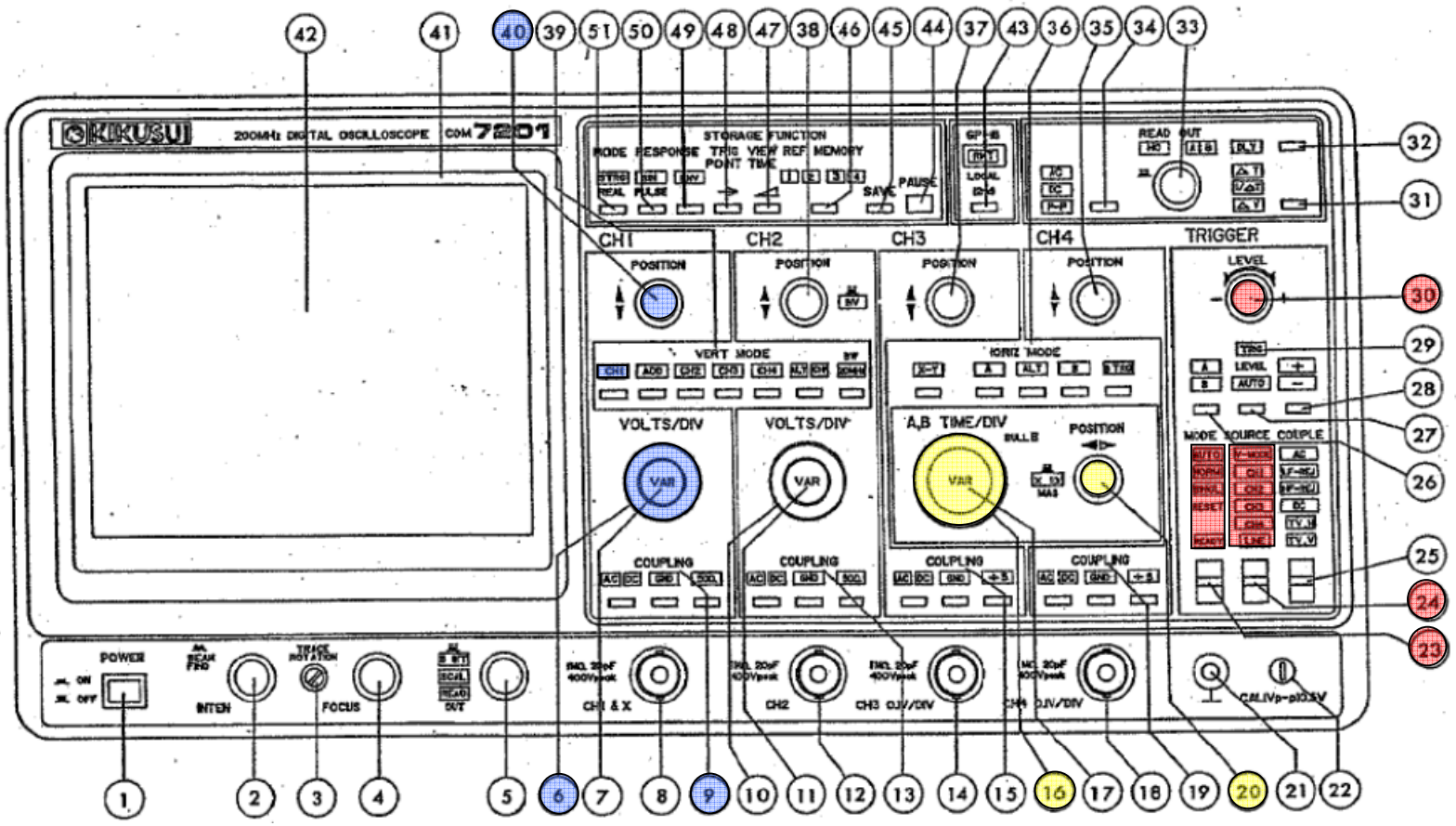
Digital logic levels

| Level | Min | Max | Units |
|---|---|---|---|
| Input low voltage | –0.3 | 0.8 | V |
| Input high voltage | 2.0 | 5.8 | V |
| Input leakage current | — | 50 | µA |
| Output low voltage (I = 8.5 mA) | — | 0.8 | V |
| Output high voltage | | | |
| Active drive (push-pull), I = –8.5 mA | 2.0 | 3.5 | V |
| Open collector (open-drain), I = –0.6 mA, nominal | 2.0 | 5.0 | V |
| Open collector (open-drain), I = –8.5 mA, with external pull-up resistor | 2.0 | — | V |

# 別紙B　　菊水電子　デジタルオシロスコープ　COM7061　のフロントパネル配置

SCREEN
表示画面

40：VERTICAL　POSITION
電圧基準レベルの移動




30：LEVEL
トリガレベル
の移動

24：SOURCE
トリガソース
の選択

23：MODE
トリガモード
の選択

POWER
電源スイッチ

6：VOLTS/DIV
電圧レンジ選択

9：COUPLING
電圧入力形式の選択

16：TIME/DIV
表示時間レンジ選択

20：HORIZONTAL　POSITION
表示時間位置の移動