ノートパソコンによる計測と制御
－ HT(HP)-BASIC への誘い－

富永雅樹*

Measurement and Control by Note PC
- Introduction to HT(HP)-BASIC -

TOMINAGA, Masaki

Advanced Technology Research Group,
National Research Institute for Earth Science and Disaster Prevention, Japan

Abstract

Measurement and control systems by use of computer are becoming popular. Most of them that we can use in daily experiments are offered in the form of pre-fixed systems. The pre-fixed systems are designed to remove complexity in the setup of the instrumentation and control systems. It is true that the hardware is well combined with the software in those systems, but it is difficult or impossible for us to modify them for our own use. On the other hand the published software that have flexibility in use are so well designed that researcher cannot see what is going on in the systems or the programs. In the present paper HT-BASIC is introduced for the use of instrumentation and control by use of note PC. The HP-BASIC was a well acknowledged software to control measurement systems by PC. Researchers could manipulate them for their own use. Since discontinuance of the PC specified for the use of the HP-BASIC the software became unpopular. But new version of the HP-BASIC those name was changed to “HT-BASIC for WINDOWS” is designed to work well in WINDOWS environment. We can make personal systems for instrument and control by the “HT-BASIC for WINDOWS” without much difficulty.

Key words : Note PC, BASIC, Field instrumentation, Instrumentation, User oriented instrumentation system

1. はじめに

インターネットによる情報検索や電子メールの交換・文書作成のための万能事務端末として，コンピュータは個人で持つものになってきている．機能も向上し，Cray 社が 1976 年に発売した 27 億円もするスーパーコンピュータ Cray1 の速度が 160 メガフロップス，メモリが 8 メガバイトだったのに比べると，現在普通のパソコンでクロック 2 ギガヘルツ，メモリ 500 メガバイトだから，当時のスパコン以上の道具が机上にある時代になった [8]．計測方面でも利用されていて，汎用のオペレーティング・システム WINDOWS 上で動作する計測ソフトウェア（VEE，LabView など）も普及している．しかし，これらのソフトは WINDOWS 独特の GUI（Graphic User Interface）を利用して作られていて，簡単ではあるが操作の中身がよく見えないという性質をもっている．研究・開発現場ではプログラムの中で明示的に何を行うかという作業を指示して計測システムの制御を行いデータを収集しないことにはその後の処理を行えないから，市販のソフトでは隔靴掻痒（かっかそうよう）の感を免れないのである [5, 6]．他方，移植性の高さで利用されている C 言語も，操作者・計測されたデータ・環境条件・計測システムの制御などの間で相互作用を行わせるといったインタラクティブ（interactive）なシステムを作るには向いていない．作った本人以外にはプログラムの流れを理解しにくく，修正のたびにコメントを書き込んでおかなければ，数年たつと作成した本人にもわからなくなってしまう．同じ作業でも多様な手法がありわかりにくいこと，作業を関数として表現しなければならないために作成者によってアルゴリズムがまちまちであること，時間がたてば細かな指定を何のために行ったのか忘れてしまうことなどがその理由である．新人の研究員や学生諸君が多大のエネルギーをつぎ込みながらも，自

* 独立行政法人　防災科学技術研究所　防災基盤科学技術研究部門

分の体の一部のように動作する計測システムを持ち得ない状況を見ていると，グラフィカルなプログラム全盛の時代にはどうしても影が薄くなってしまった HT(HP)-BASIC という言語を再評価せざるを得ない．本報告は計測システムを作成する場合に上記のような壁や悩みに直面されたことのある，すべての学生および技術者諸君のために，当研究所での経験を踏まえ，HT(HP)-BASIC のアウトラインを紹介するものである．
(なお記載されている製品名，会社名は各社の登録商標または商標である．)

2. HT(HP)-BASIC の歴史

計測システムの過去 30 年の歴史を見ると，1970 年代初頭では計測のシステム化は原子力などの大規模なプロジェクトで使用されていた CAMAC などを除けば，まだ夢だった．アナログ計測器が単体で使用され，ペン書き記録計やデータレコーダで経時変化が記録できればよい方だった．しかし，信号処理の分野ではダイナミックな信号理論の研究から，目的とする情報を得るには測定された信号一つ一つの精度も高めねばならないが，それらの信号が測定された時刻の精度も重要であった．複数の測器を組み合わせた任意の計測システムを実現するには，測定器のデジタル化が必要だったが，まだ一般的ではなかった．その後，集積回路技術の発展で演算増幅器（Operational Amplifier）と Texas Instruments 社の論理 IC が普及するようになって測定器のデジタル化が一気に進んだ．製造業やプラント現場では「計装」と呼ばれるように，計測は計測単独で存在するものではない．かならず，測定された情報をもとにしてプラントや製造ラインなどの制御が伴うものである．研究においても，測定されたデータに基づいて実験条件や入力信号を変更するなどの操作が行われる．それまでアナログ信号にもとづきスイッチや弁の開閉のためのアクチュエータを駆動するには個別に工夫された機構が必要だったが，それらをユーザの要望に応じた汎用性のあるシステムに組み上げるには，コンピュータ制御にもとづく計測システムの実現が望まれていた．Hewlett-Packard（HP）社が社内規格として計測システムのデジタル制御方式（HPIB, Hewlett-Packard Interface Bus）を実現したのは 1973 年である．これは，計測器メーカとしてのノウハウと卓上型計算機（desktop computer）の技術を持った HP 社でなければ成しえない快挙だった．HP 社はその後自社の規格を国際規格として提案し，1975 年に IEC 規格として承認された．それ以来，規格の向上も含めて公式には GPIB（General Purpose Interface Bus，HP 社内では HPIB と呼ばれている）として普及している．

当初の HP 社の社内規格段階でも，プリンタおよびデータ保存用のテープカートリッジドライブを内蔵した卓上計算機（パーソナルコンピュータと呼ばれていた）を中心として，デジタル電圧計（A/D コンバータ），スキャナ（リレーによるスイッチング回路，現在の画像入力機器とは異なる），デジタルクロック（時計），D/A コンバータ，シンセサイザ（信号発生），カウンタ（周波数・時間間隔測定）などがそろっていた．これだけあれば一通りの計測・制御システムを構成できるようになっていた．

1977 年に HP 社は HP9825A という歴史に残る卓上計算機を発売した．これは GPIB が国際規格になったことを受けてその完全な制御を行うために設計されたもので，これ以後システムの中心となる計算機はコントローラと呼ばれるようになった．ディスプレイは 32 文字表示の一行のみ，250kbyte のテープによるデータカートリッジ，16 桁のサーマルプリンタ，ディスクなどの不揮発性のメモリはなく，標準装備での RAM は 6.8k バイトという，今では考えられない小さな容量しかない HP9825A だったが，優れたオペレーティングシステムと HPL（Hewlett-Packard Language）という独特の言語の使用で，高度な計測・制御システムを組み上げることができた．その後，今ではあたりまえの，CRT ディスプレイを内蔵した卓上計算機とともに HP-BASIC 言語（Rocky Mountain Basic とも呼ばれている）が登場した．この言語はその後 HP 社の計測・制御用計算機（コントローラ）の中核となった言語で，外部機器やデータの入出力を制御するための命令だけではなく，数値演算機能，グラフィックス出力機能，ファイルの作成やデータ転送機能などを含んだ言語である．このころから LA（Laboratory Automation：「実験室の自動化」）あるいは FA（Factory Automation：「工場の自動化」）という概念が浸透していった．その後，GUI のさきがけといえる HP-BASIC Plus が付加された包括的な言語になった．

1986 年に Toshiba USA が持ち運び可能なラップトップコンピュータ T-3100 を発売した．この衝撃は大きく，その後のパーソナルコンピュータの形状と流れを決定付けた．これを受けて HP-BASIC のユーザは持ち運び可能なコントローラの出現を強く望んだ．そのようなコントローラがあれば，室内にとどまらない計測システムを組み立てることができるからである．IBM PC が世界標準として普及するとともにそれを支える Intel 社の CPU チップも標準となった．Motorola 社も CPU チップを生産していたが，Intel とは異なる 68 系とよばれるアーキテクチャだったので将来の販路拡大を望めず生産を中止した．それにともない，Motorola 社の CPU を採用していた HP 社は 1990 年代半ばにコントローラの生産中止を発表した．このニュースに全世界のユーザが驚き，現行の計測・制御システムの保守用品として HP のコントローラの買いだめ現象が起きた．またユーザからの生産続行の要望を受け入れ，生産中止が数年先延ばしにされたりした．結局 HP-BASIC 対応のオペレーティングシステムを持つ HP9000/300 シリーズのパソコンは 1996 年 10 月に公式に販売中止になった．HP 社による HP-BASIC はバージョン 6.2 で終了した．しかし，製造中止になり，保守も受けられない以上，手持ちのコントローラが壊れたらどうにもならない．そこで，IBM PC の標準 OS である WINDOWS 上で動く HP-BASIC 出現の要望が高まった．TransEra 社は 1988 年から IBM PC 上で動く HP-BASIC を Rocky Mountain Basic という名前で発売していた．HP 社は 1996 年初頭から 2000 年 11 月にかけて HP-BASIC for WINDOWS という名前で TransEra 社の OEM 製品を発売していた．

これで表向きには HP-BASIC のユーザはコントローラの故障を心配しなくてもよくなったのだが，状況は混乱してしまった．これからの計測の形が見えなくなったことが原因である．いくつか理由を挙げてみると，まず，ユーザの，従来の HP-BASIC への絶大なる信頼があったことが理由のひとつである．当時は各社各様の CPU とそのマニュアルが市販されていてコンピュータを手作りすることがはやった時代である．Motorola 社の 68 系の CPU のアーキテクチャが Intel とは異なるということは，当時の開発技術者にとっては当たり前の知識だった．システムが異なる以上，古いプログラムを完全に移行させることはできないが，HP-BASIC for WINDOWS の英文マニュアルには新旧の対比がされていて，できないことを細かに書いてあった. 開発技術者としては当然のことをしたまでなのだろうが, ユーザはこれはとても使えないと思ってしまった．また，HP 社は VEE の開発に力を注ぎ，TransEra 社の HP-BASIC for WINDOWS を販売はしたが，積極的なバックアップはしなかった．特に，日本 HP 社の日本語マニュアルは有料の講習会参加者を前提としたもので，関心がある研究者・学生にもマニュアルは手に入らなかった．また，それを読んでみても通常のプログラムの書き方が説明してあるだけで，計測制御が簡単に出来るという HP-BASIC の優れた特徴が理解できるようには書かれていなかった．さらに WINDOWS の登場も影響している．初期段階で，どんなことでもできる夢のシステムのような宣伝をされたため，移植性の高い C でプログラムを組んでみようという人もいた．そのための勉強や，勉強するためのマニュアル探しなど，時間を浪費したベテラン技術者もいたと思う．C についてはさまざまな解説書が出ているということ自体が, C のわかりにくさを証明している．計測・制御システムを白紙の状態から組み上げるには C 言語は不向きである．WINDOWS では視覚に訴えるビジュアルなドキュメントが簡単に作れるようになり，HP 社の VEE とか National Instruments（NI）社の LabVIEW とか，計測・制御の分野でもビジュアルな「プログラム」が組めるというソフトが登場した．NI 社は将来のユーザを見込んで学生向けにこれらのソフトの講習会を開いたり，各国語のマニュアルを出しているが，HP 社は熱心とはいえない．他方，PC の普及につれて，両社以外にも各社各様の計測制御用インターフェイスボードが市販されるようになった．これらのボードには専用のソフトが付属している．初めて PC で計測制御を行う学生諸君は当然としても，他のシステムを使ってきた人でも，マニュアルを読めばデータ収録が出来るので普及している．しかし反面，これらのユーザはマニュアルどおりにアイコンをクリックしてデータを収集はするが，いったいどのようにしてデータが得られているのかがわからない，といった不満・不安も持っているのである．このように，データはデータとして取り込み，その後の計算やグラフ化などの処理は，それ専用のソフトでおこなうというのが現状である．

本報告で述べようとする HP-BASIC は, 上述したような機能が全部含まれた, オールインワン（all in one）のソフトである．TransEra 社はその後 HT-BASIC for WINDOWS という名前で継承して，編集画面での使いやすさなどに改良を加えて現在にいたっている．

HPIB 登場以来の計測システム制御の流れを見てきたが，計測器の制御というのは，形こそ違え今日でもなお GPIB のプロトコルが基本であり，各社が出している GPIB や GPIO（General Purpose Input and Output，汎用のデジタル信号 I/O バス）の PC カードは HT(HP)-BASIC で制御できるのである．現在 40 代後半から 50 代になる技術者・研究者の間で圧倒的に支持され世界標準といってもよかった HT(HP)-BASIC が，今日学生の間であまり普及していない理由は知名度が低いためである．当時 HP 社の測定器およびコントローラ（計算機）は高価で，大学にはなかなか普及しなかった．日本語マニュアルもあったので，資金に恵まれた公的研究所や民間企業では普及したが，1 ドル 300 円前後でかつ為替レートが大きく変動する時代には，大学で購入を計画することさえ難しかった．その結果大学の研究室において HP-BASIC は普及せず，学生の間にソフトが伝承されることもなかった．現状としては，膨大な英文マニュアル（それもオンラインマニュアル）のみで，全体像もよくわからないシステムが，これから学生諸君に普及していくとは思えない．しかし，導入のための価格が下がった今こそノートパソコンで自由に計測・制御システムを組める時代がやってきているのだ．このような現状を踏まえ，一読されれば HT(HP)-BASIC のアウトラインがわかり，その可能性に気付いてもらえるような紹介をするのが本報告の目的である．

国立防災科学技術センター（現独立行政法人　防災科学技術研究所）大型降雨実験施設が完成したのは 1974 年の 3 月で，計測設備は未整備だった．多種多様な実験を行うには自由に構成が変更できるコンピュータを中心にした計測・制御システムが最適だったが実例はなかった．ちょうどそのころ HP 社が社内規格として HPIB を提案したので調べてみると，ひととおりのシステムが組めるようになっていた. その段階では HPIB は公的規格ではなかったが, HP 社が発売している（上述した）製品のみでも閉じた計測システムが組めるので HP 社のシステムを導入した．その後は国際規格（GPIB）として，かつデジタル技術の進歩とともに GPIB コネクタを装備していない計測器はないぐらいに普及し，コントローラも各社から発売された．自由に計測・制御システムを組み上げ実験を行うという立場からは HT(HP)-BASIC で GPIB を制御することの優位性は変わっていない．HP-BASIC を継承した TransEra 社は HT-BASIC という呼び名で改良を続け，HP 社の最終版に比べると，その優れた制御機能を残したまま，編集機能，数学関数，サブルーチン，ビジュアルなディスプレイ機能などを拡張，統合させている．時代の流れに沿って, WINDOWS 上での GUI（Graphic User Interface）を利用した使いやすいプログラムになっている．

3. HT-BASIC のアウトライン [1,2)

当初 HP 社で開発された HP-BASIC は，現在は TransEra 社が引き継ぎ，HT-BASIC という名前で開発を続けている．現在市販されているのは HT-BASIC バージョン 9.2 である．本章以下では HT-BASIC と呼んで，その特徴を概観する．

HT-BASIC は，GPIB バスラインにつながる外部測器等の制御，任意のデジタル信号入出力制御，コンピュータの CRT（液晶表示画面も含む，以下同じ）出力画面制御，ディスク上のファイルの作成とデータの入出力制御および一般の数値計算を含むプログラミングを行う言語である．その全体構成を**図 1** に示す．デスクトップコンピュータについては，GPIB ボード，および汎用のビット入出力を行う GPIO ボードが製品化されている．ノートパソコンについては GPIB 用およびビット入出力用の PC（PCMCIA）カードが市販されている．

3.1 HT-BASIC の特徴

(1) インタープリタ（interpreter）言語：

HT-BASIC は，基本的には，プログラムの命令ラインを一行づつ読み込んでは解釈し実行するというインタープリタ（interpreter）言語である．C あるいは Fortran のようにコンパイラ言語ではないので，大規模な数値計算等を行うには向いていない．しかし，極端な高速化が要求されない数値計算・操作者との対話型プログラム・バスラインで結ばれたコンピュータ周辺機器（プリンタ，画像取り込み用スキャナー，ハードディスクなど）以外の機器（各種の測定機器，プロッタ，デジタイザ，デジタル信号の入出力，あるいは自作のリレー回路など）は，それぞれ勝手な速度で動いているから，各機器の応答を待って次の作業を実行するような場合にはインタープリタ方式の言語が便利である．

さらに，インタプリタ言語の最大の特徴ともいえるのは，その呼び名の由来でもある，RUN キーを押せば即実行するという性格である．実験室でのプログラム開発では，まず自分がやりたい処理の大枠を決めて骨格となるプログラムを書く．その後，いろいろな機能を付け加えて使い勝手のよいプログラムに仕上げていくというのが普通のやりかたである．そのときに，少しの変更であっても，ファイルの保存 → コンパイル → 実行を繰り返さなければならない C 言語などでは，湧き出すアイデアの確認に手間取るだけである（あとがき参照）．また，実験や計測の現場では以前の実験のプログラムを元にして改良を加えるというような場合がよくある．このような場合にも C 言語だと変数のみならずプログラムの流れに細心の注意を払わないとうまくいかない．BASIC 言語ならば行を追っていくことでプログラムの流れを確認することは簡単である．元になるプログラムが他人の手になるものであれば，C 言語の場合改変は不可能といってもいいが，BASIC であれば可能であり，先輩の研究を引きつぐ研究室の資産としてのプログラムの価値が増すだろう．

翻って考えれば，これらの特徴は特に列挙するまでもないことである．考えたとおりの作業が行えるというのがプログラムの使命である．それが見えなくなってきていることが本報告執筆の動機でもある．

(2) コンパイラ機能：

HT-BASIC は基本的にはインタープリタ言語であるが，数値計算などを行う場合には，付属のコンパイラ機能が利用できる．

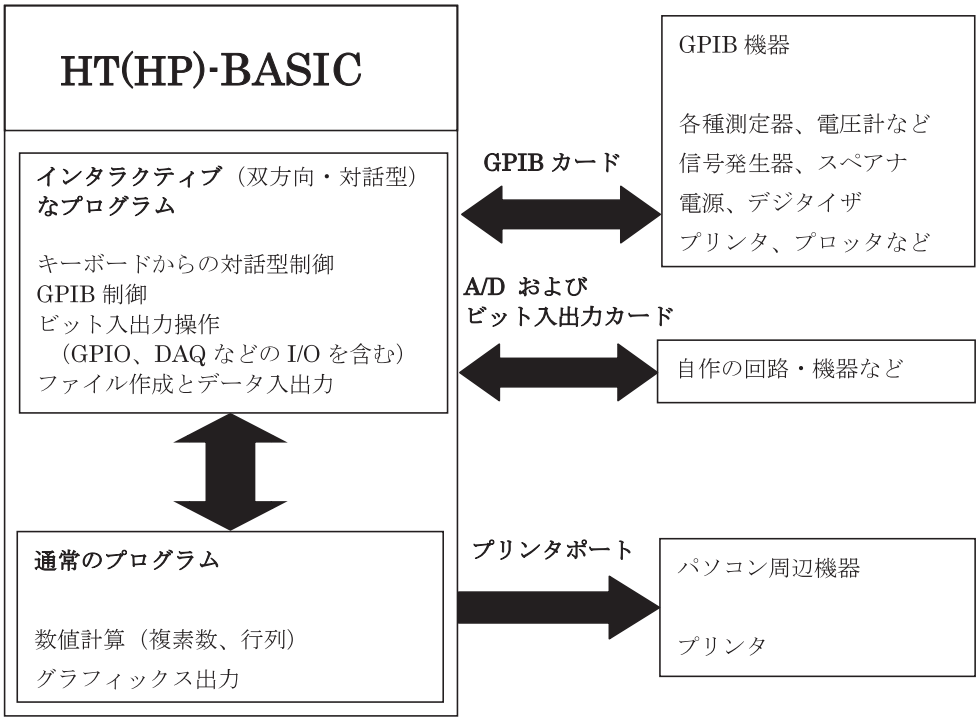

図 1　HT (HP) -BASIC のアウトライン

Fig. 1　HT (HP) -BASIC and its environment.

(3) BASIC 言語：
名前のとおり BASIC 言語だから，BASIC 言語に共通するプログラムの構造をもっていて，論理がわかりやすい．作成者や他人が数年後に読んでも理解でき，改良も容易である．プログラム言語は，言語である以上わかりやすい表記が望まれる．C 言語の場合，ベテランのプログラム開発者であっても初歩的ミスをおかしやすいということ自体[7]，その言語のもつ本質的欠陥といえる．

(4) サブルーチン：
通常のサブルーチンのみならず，コンパイルされた数学関数やサブルーチンのライブラリがあり，プログラム中に自由に呼び出して使用できる．HT-BASIC では数学関数に関してはほとんどすべてが CSUB 化されているが，C 言語などを利用して CSUB も自作できる．

(5) ファイル入出力：
プログラム中で任意の名前のファイルをディスク上に作成し，収録データなどを書き込むことができる．長期にわたる計測を行うときなどに威力を発揮する．

(6) GPIB:
HT-BASIC は単独でも強力な言語だが，GPIB バスラインを介して計測機器等の制御を行うときに真価が発揮される．ノートパソコンの場合，PC カードのスロットに GPIB のカードを差込み，他端の GPIB 専用のコネクタを計測器の GPIB ソケットに接続すればよい．あるいは USB ケーブルの他端が GPIB のコネクタになっているものもある．

(7) ビット入出力（GPIO）：
ビット I/O の制御も GPIB 同様に，PC カードのスロットにビット I/O のカードを差込み，他端のケーブルを自作のデジタル信号変換回路に接続して使用する．GPIO というのは HP 社が当初発売した，パラレルで独立した夫々 16 ビットづつの入力ラインと出力ラインを制御できる I/O 機器の名称である. 現在 GPIO はデスクトップ用の PCI ボードしか市販されていないが，ビット I/O の制御ができる PC カードとして, NI 社や INES 社が DAQ（Data Acquisition）と呼ばれるカードを販売している（5 章で詳述する）．このなかには，A/D コンバータを内蔵していて単独でも電圧等の測定がおこなえると同時に，8 ～ 24 ビットの入出力制御が行えるカードがある．AD コンバータを含む PC カードがあればそれだけで電圧信号に変換できる信号の計測システムを作ることが出来る．

(8) HT-BASIC Plus：
HT-BASIC Plus という拡張言語を利用して，計測データをメータやグラフ表示することができる．現在のパソコンで主流になっている GUI のさきがけとなった機能である．しかし，計測・制御という目的からは付随的な機能であること，プログラムの本筋とは無関係な，表示のためのパラメータの細かい設定が多く煩雑であること，中核をなす HT-BASIC の機能で代替できることなどから，HP-BASIC 以来の古いユーザでも HT-BASIC Plus の機能を使ったことがない人が多い．

(9) グラフィック出力：
グラフィックスは直接 CRT 上に描くことができる．GPIB ラインに HPGL というグラフィック言語を読み取るプロッタが接続されていればプロッタに出力できる. CRT 上の画像をそのまま画像ファイルとして保存する命令は HT-BASIC にはないが，Windows 内臓の「ペイント」あるいは市販の画像キャプチャソフトを利用すれば，ビットマップなどの画像ファイルとして保存できる．

3.2 HT-BASIC のインストール

(1) プログラム：
HP-BASIC の最新版は HP-BASIC を継承した TransEra 社の HT-BASIC for WINDOWS で，現在バージョン 9.2 が出ている．これを手順に従ってインストールすればよいが，レガシーバージョンではなく HT-BASIC を選択し，カスタムインストールを選んですべてのコンポーネントをインストールする．全体でも約 40M バイトを占有するだけである．インストール後のディレクトリは，特に指定しなかった場合，

```
C:\Program Files\HTBwin92
```

となっている．

(2) GPIB カード：
各社から発売されているが，以下の例では NI 社の GPIB ＋カードを使用する．カードスロットにカードを差込み，指示どおりにドライバをインストールする．

(3) デジタル I/O カード：
各社から発売されているが，以下の例では INES 社の DAQ カード i250 を使用する．INES 社からは数種類のカードが販売されているが，i250 はこれまで故障もなく最も多く出荷されている汎用の計測カードといえるものである．指示どおりにドライバをインストールする．準備はこれだけである．

以上，大体の特徴を述べたが，次章からプログラムを作成する手順を示す．

4. HT-BASIC のプログラミング 1（双方向・対話型制御を行なうプログラム）

HT-BASIC はバスラインで接続された機器の制御をするときに優れた特徴をもっているが，数値計算やグラフィックス出力などを行うための通常のプログラムも簡単に作成できる．4・5 章で対話型のプログラム作成手法，6・7 章で通常のプログラム作成手法とその長所を解説する．

4.1 対話型プログラム

本節ではインタープリタ（interpreter）言語としての HT-BASIC の特徴がよくわかるプログラム例を示す．プログラムのわかりやすさ，簡単さ，テストして変更を加えたりする手順の簡便さを C や FORTRAN と比較してもらいたい．以下では，プログラム文中で使用される命令を「命令」，一連の命令を「命令文」と呼ぶことにする．また，＜ENTER＞のように括弧で囲んだ表現はキーボード上のキーの名前，および混乱しない範囲で，キーボードからの入力や選択すべきメニュー，あるいは強調等をあらわすことにする．HT-BASIC では，「命令」は大文字のアルファベットで書かれる．これらの「命令」はプログラム中でもキーボードからでも使用できるものが多いので，混乱しない限り断らない．「命令」の解説や文法的に正しい使用例はヘルプマニュアルに詳しく解説してある．さらに，各「命令」の具体的な使用例が example ファイルの中に個別のプログラムとして取り上げてあるので参考にされるとよい．HT-BASIC for WINDOWS Ver. 9.2 のエディタ画面では「命令」，変数，文字列などごとに色を変えて一目してわかるように表示されている．また，命令ならば自動的に大文字に変換したり, 文法ミスを指摘するなどの機能が加わって使いやすいものになっている．example プログラムの中でわからない「命令」に出会ったときには，その命令をマウスの左ボタンで選択し，右ボタンでメニューを表示すると最下行に Help という項目が出る．それを左クリックで選択すると，その命令の説明項目が表示される．

（1）EDIT 画面：

ディレクトリ

```
C:\Program Files\HTBwin92
```

のなかにある HTBwin.exe ファイルを実行する．黒色の背景画面が表示されるから，ツールバーの File→New を選び，Edit →Edit Mode を選ぶと，背景が白色のプログラム編集画面に変わる．プログラムの編集は，任意のテキストエディタで作成したものを，カットアンドペーストで白背景のプログラム編集（エディタ）画面に貼り付けてもよい．

（2）キーボードからの入力（プログラムの中断と変数の変更，任意行からの再開）：

EDIT 画面で次のようにプログラムを入力する．

```
10  INPUT A      ！キーボードからの入力
20  PRINT 3*A    ！ディスプレイへの出力
30  PAUSE        ！プログラムの中断
40  END          ！プログラムの終了
```

プログラムラインを入力し＜ENTER＞キーを押せば，次行にカーソルが移動して自動的に行番号がつく．＜！＞より右はコメント文である．行頭に＜！＞をおくとその行全体がコメント文になる．キーボードから

```
RUN  ＜ENTER＞
```

と入力するか，ツールバーの中の Run（右方向を向いた緑色の三角形の記号）を選ぶとプログラムが実行される．画面左下に

```
?
```

が表示されるが，これはキーボードからの入力待ちであることを示している．数字 5 を入力し＜ENTER＞キーを押せば 15 が画面上部左隅に表示され，30 行で停止する．このとき変数 A には 5 が代入されている．それを確認するために，キーボードから

```
A  ＜ENTER＞
```

と入力すると 5 が，今度は画面の左下に表示される．

プログラムは停止したままなので，キーボードから，

```
A=123  ＜ENTER＞
```

と入力し，再び，A　＜ENTER＞と入力すると，画面の左下に 123 と表示され，A に 123 が代入されたことがわかる．キーボードから，

```
CONT  20  ＜ENTER＞
```

を実行すれば，プログラムはライン 20 行から再開し，3*123 の演算結果 369 を表示して停止する．

ライン番号をを指定せず，

```
CONT  ＜ENTER＞
```

を入力すれば，プログラムは END ラインで終了する．
このようにインタプリタ言語では途中でプログラムを停止させ，変数を変更したり，任意の行から再開させることができる．計測中に周辺機器が異常な応答をしてプログラムが停止するような場合にも，変数をチェックして，プログラムを中断させずに再開できる特徴がある．
プログラムを保存させるには，EDIT 画面に戻り，ツールバーの＜ファイル → 名前をつけて保存 →SAVE　AS ＞を選び，適当な名前をつけて保存する．＜ SAVE ＞はプログラムをテキストとして保存することを意味している．プログラムを読み出すときはツールバーの＜開く＞から，ファイル名をマウスで左クリックするという通常の方法でよい．「"SAVE" されたテキストファイルは "GET" で読み出す」ということを意識せずに読み出すことが出来る．

(3) キーボードからの文字列入力（文字変数の変更，任意行からの再開）：

EDIT 画面に戻り，前節とよく似た下記のプログラムを入力して実行する．

```
10  INPUT A$              ! キーボードからの文字列入力
20  PRINT "Abc"&A$        ! 文字列の連結と出力
30  PAUSE                 ! プログラムの中断
40  END                   ! プログラムの終了
```

10 行の A$ は文字列の変数である．任意の変数名のあとに＜ $ ＞をつけるとそれは文字列と判断される．画面左下に ? が表示されるので，

　　def　＜ ENTER ＞

を入力すると，画面左上に「連結」された文字列 "Abcdef" が表示され，30 行の PAUSE でプログラムは停止する．20 行の＜ & ＞は文字列を連結する命令で，＜"　"＞で囲まれた文字列 "Abc" と入力された文字列 A$ を連結している．

　　A$　＜ ENTER ＞

と入力すれば，画面左下に "def" が表示される．文字列は＜"　"＞で囲めばよいので

　　A$ ＝ "xyz"　＜ ENTER ＞

として A$ に新たな文字列をあたえ

　　CONT　20　＜ ENTER ＞

を実行すれば，画面左上にあたらしく連結された文字列 "Abcxyz" が表示される．
プログラムは 30 行で停止しているので

　　CONT　＜ ENTER ＞

を入力すれば，プログラムは END ラインで終了する．
このような，文字列データの操作の柔軟さは，C 言語とは比較にならない．

(4) プログラムの保存と呼び出し

HT-BASIC ではプログラムを＜ STORE ＞および＜ SAVE ＞の 2 種の状態で保存できる．一つは即実行可能なファイルとして＜ STORE ＞命令で保存する．STORE で保存されたプログラムは＜ LOAD ＞命令で呼び出されるが，LOAD 命令に行番号を追加しておけば，LOAD したあとに直ちに指定された行から実行できる．もうひとつは，ASCII ファイルとしてテキストのみを＜ SAVE ＞命令で保存する．SAVE で保存されたファイルは＜ GET ＞で呼び出される．GET で呼び出されたプログラムは，LOAD のように即実行はできないが，プログラムを ASCII フォーマットで保存できるから，ほかのシステムにコピーするときなどには便利である．HT-BASIC ではファイルのタイプを自動的に判断してプログラムが呼び出されるので，LOAD 命令などで特に指示をしない場合は，LOAD か GET かということを意識しないで，プログラムを呼び出すことができる．

(5) ソフトキーと，プログラムの停止，続行，中止

プログラムの実行中はディスプレイ画面下部にソフトキーのメニューが表示される．ファンクションキー＜ F9 ＞を押すたびにメニューの表示／非表示が繰り返される．
＜ Shift + F9 ＞を押すたびにソフトキーのメニューが

```
┌─► System → User1 → User2 → User3 ─┐
└──────────────────────────────────┘
```

と，順に変わる．各表示のソフトキーを押すか，マウスで左クリックすると，キーボード上からその命令を実行したのと同じ効果を持つ．＜ System ＞ソフトキーを表示させ，＜ Run ＞（F3）をマウスで左クリックするとプログラムが実行される．＜ Pause ＞（F4）をクリックするとプログラムの実行が停止する．＜ Continue ＞（F2）をクリックすると停止していたプログラムが再び続行される．＜ Stop ＞（F4）をクリックするとプログラムの実行が中止され，システムは＜ idle ＞状態に戻る．ソフトキーには，表示されているメニューの名前のもとに実行させるべき内容がキーに定義付けられている．つま

り，定義づけられている内容には各種の制御文字を含んでいる．したがって，キーによっては，表示されているとおりにキーボードから入力しても，ソフトキーと同じ効果をもつわけではない．3 個のユーザーキーメニューはユーザーが自由に定義できる．自分で定義するためにはデフォルトの状態でのキーの定義が参考になる．キーに定義された内容は＜ list key + ENTER ＞で表示される．＜ LIST KEY #10 + ENTER ＞でプリントアウトされる．＜ #10 ＞はデフォルトのプリンタの識別番号である（後述 4.3 (2)）．

4.2　外部機器の制御

(1) GPIB による機器制御（アドレス指定，機能設定，データ入力）

PC カード用のスロットに GPIB のカードを挿入し，そのカードのドライバをインストールする．他端の GPIB コネクタを GPIB 機器に接続する．本例では NI 社の GPIB ＋カードを使用し，GPIB 機器として HP 社のデジタルマルチメータ 3457A に接続した．**図 2** に NI 社の PC カードと GPIB ケーブルを示す．プログラム編集画面に戻り，デジタルマルチメータからデータを読み込むために，以下のプログラムラインを入力する．

```
10  ASSIGN @hp3457 TO 722              !I/O パスを開く
20  OUTPUT @hp3457;"NPLC .0005"        !hp3457 への設定データ
30  OUTPUT @hp3457;"DCV AUTO,.033"     !hp3457 への設定データ
40  WAIT 1                             ! 約 1 秒待つ
50  ENTER @hp3457;B$                   !hp3457 から文字列データを読み込む
60  PRINT B$,VAL(B$)                   ! データの文字列と数値の表示
70  CLEAR 722                          !I/O パスを閉じる
80  END
```

10 行の＜ ASSIGN ＞は I/O ポート 7 番を通して GPIB のアドレス 22 番の機器への I/O パスを開くことを意味する．I/O ポート 7 番は通常，GPIB カードを示している．22 番というのは GPIB ラインで接続された機器が個別に持っているアドレス番号で，機器それぞれに 0 ～ 32 までの番号を設定できるが，同じ GPIB ラインで接続されている機器の間で同じ番号を重複して設定することはできない．ASSIGN 文中の＜ @hp3457 ＞は，＜ 722 ＞という番号の I/O パスに付けた名前である．ASSIGN 文で名前＜ @hp3457 ＞とパス＜ 722 ＞を対応させておけば，その後の作業は名前を指示するだけでパスの指定が行える．

20，30 行は＜ @hp3457 ＞という名前で指示された I/O ポート＜ 722 ＞へ設定データを送り出す命令であり，必ず文字列によるデータである．記号や数字の意味は各機器ごとに異なるので，それぞれのマニュアルに説明されている．HP3457A というデジタルマルチメータの場合，20 行は AD 変換の積分時間を最小値にする設定であり，30 行はオートレンジによる DC 電圧を指示された精度で測定させる設定である．

40 行はプログラムの実行を約 1 秒だけ遅らせている．測定対象が過渡応答を示す場合など，設定された計測系が安定するまでの時間である．遅らせる必要のない場合もある．計測器の機能だけを見れば短時間での計測が可能であっても，測定対象を含む計測系が安定するまでには時間がかかることがある．何を測定しようとしているのかを理解しておくことはプログラム作成以前の問題である．

50 行では @hp3457 という名前で指示された 722 という I/O ポートからデータを読み込んでいる．データは文字列として出力されているから，文字列データとして読み込んでいる．

60 行の＜ VAL(B$) ＞は文字列 B$ を数値に変換する命令である．文字列として読み込まれた数字はそのままでは数値変数として使用できない．60 行では読み込んだ文字列とそれを数値に変換した値を画面に出力している．

70 行で @hp3457 という I/O ポートを閉じている．

GPIB バスラインの制御はほとんどすべての機器に対して，このような手順で行われる．

このプログラムを実行する場合には，前もって GPIB カードを制御させるためのバイナリプログラムを読み込んでおかねばならない．それは各社の GPIB カードごとに異なるので，HT-BASIC には各社の GPIB カードに対応するバイナリプログラムが用意されている．NI 社の GPIB カードの場合には，GPIB 制御を含むプログラムを実行する前に，

```
LOAD BIN "GPIBNI"
```

を実行しておかねばならない．＜ BIN ＞はバイナリプログラムの意味である．"GPIBNI" ファイルはディレクトリ

```
C:\Program Files\HTBwin92
```

のなかにある．デフォルトのディレクトリが上記と異なる場合には

図 2　GPIB カードとケーブル
Fig. 2　GPIB card and cable（national instruments Co.）.

```
LOAD BIN "C:\Program Files\HTBwin92\GPIBNI"
```

を実行する．これはキーボードからも実行できる．＜LOAD BIN "GPIBNI"＞は上書きはできないので，GPIB の制御を含む個別のプログラムの先頭にコマンドとして書き込んでおいて，プログラム実行のたびに読み込ませるということは出来ない．事前に必ず一回だけロードするようにする．第 5 章で説明するビット入出力カードの場合には，カードを制御するための専用コマンドを CSUB（コンパイル済みのサブルーチン）として，プログラム実行のたびに読み込むようになっている．この場合は，プログラムの最初にこれらの CSUB を読み込むコマンドを書き込んでおけば忘れることはない．取り扱い方が異なるのは，HP-BASIC の生まれた背景から来ている．HP-BASIC にとって，GPIB の制御をおこなうことは数値の四則演算をおこなうことと同じように基本的な機能だった．HP 社製の GPIB 制御ボードしか利用できなかった時代にはシステムプログラムの中にインストールしておけばよかった．しかし WINDOWS 搭載のノートパソコンでは各社から PC カードという形で GPIB 制御カードが市販されだしたので，各社の PC カードに対応するコマンドを全部システムに書き込んでおくわけにはいかない. だからプログラム実行の前にロードしておくという取り扱い方になっているのである. したがって, たとえば NI 社の GPIB カードしか使用しないというのであれば, 後述するオートスタート機能（4.2（3）節）を使って, ＜LOAD BIN "GPIBNI"＞を実行しておけばよい．このようにしておけば GPIB を利用するときもしないときも，気にせずにプログラムを作成し実行することが出来る．**表 1** に HT-BASIC Ver.9.2 内臓の GPIB ドライバと，それに対応する各社の PC カードの一覧を示す．対応する WINDOWS OS については確認することが必要である．

表 1　HT-BASIC 9.2 対応の GPIB カード
（対応する WINDOWS OS は夫々確認すること）
Table 1　GPIB supported by the HT-BASIC 9.2.

HT-BASIC 9.2 ドライバ名	製造者名	PCIバス	PCカード	USBバス	ISA/EISAバス
HPIBS.DW6	HP/Agilent(USA)	82350		82357	82341
	ines(Germany)	GPIB-PCI-NT+	GPIB-PCM-NT+		GPIB-ISA-NT+
	TAMS(USA)	60488			
GPIBNI.DW6	NI(USA)	PCI-GPIB	PCMCIA-GPIB	GPIB-USB	
	ines(Germany)	GPIB-PCI-XL	GPIB-PCM-XL		
GPIB900.DW6	TransEra(USA)				HM900

(2) ビット入出力（GPIO，ビット操作）

現在は，ほとんどの計測器が GPIB インターフェイスを装備しているので，GPIB バスラインによる制御をおこなう場合に困ることはないが，状況によっては，リレー回路や弁の開閉など自作の機器をふくめて制御したい場合もある．HP 社の HP9000/300 シリーズのコントローラには，GPIB のほかに GPIO というインターフェイスがあった．これは，完全にパラレルな 16 ビットのデジタル信号の入力および出力ラインを制御するインターフェイスで，HP-BASIC 言語で自由に制御することができた．現在でもデスクトップ PC には PCI バスを介した GPIO カードが市販されていて，HT-BASIC で制御できる．ノートパソコン用には，PC カードの制約から，完全にパラレルな 16 ビットのデジタル入出力を制御できるものはないが，デジタル入出力と A/D コンバータを組み合わせた PC カードが市販されている．これからの利用を考えると，データを A/D コンバータを通して計測するだけの場合も多いと思われる．そのような場合にはわざわざ GPIB を介して電圧計を接続するまでもなく，PC カード一枚で，A/D 変換も行え，デジタル入出力で外部機器の制御も行えたほうが便利である．したがって，ビット入出力に関しては次の第 5 章で詳しく解説する．

(3) オートスタート

HT-BASIC を起動させると，プログラムは同じディレクトリ内の＜ AUTOST ＞という名前のファイルを検索する．もしその名前のファイルがあれば，プログラムは自動的に＜ AUTOST ＞ファイルを＜ LOAD ＞して実行する．この機能が HP 社のコンピュータに現れたのは卓上計算機 HP9825A で，実行中のシステムの状態を一括して記憶するという機能とともに使うようになっていた．プログラム制御による計測中に，定期的にシステムの状態をテープカートリッジ上の AUTOST というファイルに記憶させておく．そうしておけば，たとえば，計測中に停電が発生し，その後復帰したような場合にも，コンピュータはテープカートリッジ上の AUTOST ファイルに書き込まれているシステムの状態を元にシステムを再度立ち上げ，計測を続行させることができた．現在のコンピュータにはシステムの状態を記憶させる命令はないので，このような使い方はできない．現在のシステムでは，実行時に必要となるバイナリプログラムを，前もって読み込ませておくために使用されている．具体的に例を示す．AUTOST プログラムは

```
C:\Program Files\HTBwin92
```

のディレクトリの中にある．これをエディタ画面上で "LOAD" 命令で開く．あるいは，HT-BASIC を起動すれば，必ず AUTOST を実行するから，起動後の黒い画面が表示された状態で，ツールバーの＜ Edit → Edit Mode ＞をクリックすれば AUTOST のプログラムが表示される．このプログラムには各種のバイナリプログラムやドライバを設定するための命令がコメント文として記述されている．解説文（英語）を読み，ロードすべき命令文の先頭の＜！＞を削除すれば，そのまま命令文になる．必要な命令文を選択し終えたら＜ RE-STORE ＞命令で，同じディレクトリにストアしておく．＜ LOAD ＞命令でプログラムをロードし，＜ RE-STORE ＞命令でプログラムを保存しなければならないのは，AUTOST ファイルと CSUB 命令がプログラムの後尾に追記されているプログラムのみである．CSUB を含まないプログラムは＜ (LOAD) / RE-STORE ＞でも＜ (GET) / SAVE ＞でもかまわない．

プログラム中で CSUB を利用する場合には，AUTOST ファイルのなかで指示をせずに，たとえば，プログラムの最初の行で，

```
LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
```

という命令を使って，CSUB をプログラムの最後に追加して実行させる．保存する場合には追加された CSUB を消去すれば "SAVE" 命令で保存することができる．CSUB を残したまま "SAVE" すると，次に "GET" 命令で呼び出したとき（実際には＜ファイルを開く＞という作業）にエラーが発生する．しかしこのエラーは Edit 画面で追加されている CSUB を消去すれば，もとのテキストだけで書かれたプログラムに戻るので，＜ RUN ＞で実行できる．

4.3 ファイル操作

HT-BASIC では実験中に測定したデータをファイルに保存したり，プログラム中で，作成したファイル中のデータにアクセスすることが大変簡単におこなえる．

(1) アスキーファイルの作成，データの書き込み，呼び出し

プログラムの中でファイルを作成し，そのファイルにデータを保存したり呼び出したりすることも HT-BASIC なら簡単にできる．次のプログラムを入力する．

```
10   MASS STORAGE IS "d:\hp\temp"            ! ディレクトリ設定
20   CREATE "test.txt",0                     ! ファイル生成
30   ASSIGN @File TO "test.txt"; FORMAT ON   ! ファイルへの I/O パスを開く
40   FOR I=1 TO 5                            ! ループ
50      INPUT "numeric, characters",A,B$     ! キーボードから数値と文字の入力
60      PRINT A,B$
70      OUTPUT @File;A,B$                    ! ファイルへの書き込み
80   NEXT I
90   RESET @File                             ! ファイルポインタのリセット
100  PRINT "------"
110  FOR I=1 TO 5
120     ENTER @File;C,D$                     ! ファイルからの読み出し
130     PRINT C,D$
140  NEXT I
150  ASSIGN @File TO *                       !I/O パスを閉じる
160  END
```

10 行は新しくファイルを作成するディレクトリを示す命令である．通常，キーボードから入力するときは＜Mass Storage Is＞の省略形で＜MSI "d:\hp\temp"＞と入力してよい．＜Mass Storage Is＞で宣言したディレクトリは，そのプログラムの中では，明示的に変更しない限り，暗黙に指定されたディレクトリとなっている．
20 行は 10 行で指定されたディレクトリに＜test.txt＞という名前のファイルを作成している．HP-BASIC では＜.txt＞という拡張子は認識しないので，＜test.txt＞という名前のファイルにデータを書き込んでも，自動的に ASCII フォーマットとして書き込まれるわけではない．この場合の拡張子は，WINDOWS 上のソフト（たとえば表計算ソフトの EXCEL など）でこのファイルのデータを利用する場合のために便宜的につけたものと理解すべきである．コンマのあとの数字はファイルに記録されるデータ数をレコード数で指示したものである．WINDOWS ではファイルに保存されるデータの数に制限はないから，ここでは＜0＞という数字を使用している．
30 行では＜@File＞という名前で＜test.txt＞というファイルへの I/O ポートを開いている．＜FORMAT ON＞はデータが ASCII 表現であることを示している．もし＜FORMAT OFF＞になっていればバイナリデータであることを示す．ASCII ファイルであれば OS が変わってもデータの互換性があるから，ASSIGN 命令を使用するときは常に＜FORMAT ON＞と明示したほうがよい．
40 ～ 80 行はキーボードからのデータ読み込み，表示，ファイルへの書き込みを行うための＜FOR ～ NEXT＞ループである．
50 行は上記(2)節の INPUT 命令の書き方とは少し違っている．50 行では＜" "＞内の文字列を表示して，数値と文字のデータの入力を待っている．入力するには，数値と文字のデータをコンマ＜，＞で区切って＜ENTER＞キーを押す．
60 行で入力したデータが表示され，70 行でファイルへの書き込みが行われる．ファイルに書き込む作業は 80 行で終わりである．
90 行からはファイルに書き込んだデータを読み出す作業である．90 行はファイルのポインタをリセットしている．
100 行は＜"　"＞で囲まれた文字列を画面に出力している．
110 ～ 140 行は書き込んだデータをファイルから実数変数 C，および文字列変数 D$ へ読み出している．
150 行では(4)節とは異なる形式で I/O ポートを閉じている．本来 ASSIGN は I/O ポートを開く命令だが，同時に複数のポートを開くことはできない．したがって，不特定のパス＜*＞に対して I/O ポートを開く命令ではどのポートも開けないから，結果としてすべての I/O ポートを閉じることになる．このプログラムを実行すると，データは書き込まれた順序で読み出されていることがわかる．このファイルは ASCII 文字でデータが保存されているので，WINDOWS 上のソフトであるマイクロソフト社の EXCEL でも読み出すことができる．

(2) ファイル名の自動生成と配列入出力

前節ではデータをひとつづつ書き込んだり読み出したりしたが，配列全体をそのまま記録したり，読み出したりもできる．また作成するファイル名も禁則文字以外は任意である．したがって，実行中の停電などの事故を想定して次のようなプログラムが組める．

```
10   OPTION BASE 1                           ! 配列指示子の最小値を指定
20   DIM P$[11],Q$[8]                        ! 文字列配列定義 (number)[length]
30   DIM A$(3)[18],B$(1:5)[20],C$(1:5)[22]   !(from:end)[length]
```

```
40   MASS STORAGE IS "d:\hp\temp\"            ! 作業ディレクトリの指定
50   FOR I=1 TO 3                            ! 繰り返しループ指定，I は配列指示子
60      LOOP
70         P$=DATE$(TIMEDATE)                !dd Mmm yyyy，日付の読み取り
80         Q$=TIME$(TIMEDATE)                !hh:mm:ss    ，時刻の読み取り
90         EXIT IF Q$[8;1]="0"               ![from;lenth]，副文字列の指定
100     END LOOP
110     A$(I)="data("&Q$[1,2]&"-"&Q$[4,5]&"-"&Q$[7,8]&").txt"
                                             ![from,end]，副文字列の指定
120     CREATE A$(I),0                       ! ファイル作成
130     FOR J=1 TO 5
140        B$(J)=P$&" "&TIME$(TIMEDATE)
150        WAIT 1                            ! 約 1 秒待つ
160     NEXT J
170     ASSIGN @File TO A$(I);FORMAT ON      !I/O パスを開く
180     OUTPUT @File;B$(*)                   ! 配列をファイルへ出力する
190  NEXT I
200  CAT "d:\hp\temp"                        ! ファイルリスト表示
210  FOR I=1 TO 3
220     PRINT "------"
230     PRINT A$(I)
240     ASSIGN @File TO A$(I);FORMAT ON
250     ENTER @File;C$(*)                    ! ファイルから配列を読み込む
260     FOR J=1 TO 5                         ! 読み込んだ配列を表示するループ
270        PRINT C$(J)
280     NEXT J
290  NEXT I
300  ASSIGN @File TO *                       !I/O パスを閉じる
310  END
```

HP-BASIC では DIM 命令で配列を定義する際，配列指示子の下限と上限を任意に指定できる．10 行はその範囲が指定されていない場合に，その下限を 1 とする命令である．明示的に＜ OPTION BASE 1 ＞を指定しない場合には，下限は＜ 0 ＞（＜ OPTION BASE 0 ＞を指定した場合に相当）になる．

20 行は文字列変数の文字長を定義している．配列ではない文字列変数はプログラム中で自由に使用でき，デフォルトの文字長は 18 文字である．だから，20 行はわざわざ指定する必要のない命令だが，70，80 行で P$ および Q$ の文字長の限度まで任意の文字を代入できることを示すために定義した．通常の使用では余裕を持って指定しておくほうがよい．

30 行の＜ A$(3)[18] ＞は，最大 18 文字からなる文字列 3 個の配列を定義している．＜ B$(1:5)[20] ＞は配列指示子を 1 から 5 までと指定して定義した 20 文字からなる配列 5 個である．＜ DIM ＞ではこのように配列指示子の範囲を指定できる．

40 行は作業ディレクトリの指定．

50 ～ 190 行は＜ FOR ～ NEXT ＞構文の繰り返しループ．

60 ～ 100 行は繰り返し作業のループである．90 行でループ終了の判断をしている．

70 行は日付の読み取り．＜ TIMEDATE ＞は紀元前約 48 世紀から始まっているという秒数で，＜ P$=DATE$(TIMEDATE) ＞と呼ぶことにより P$ には，たとえば＜ 11 Apr 2004 ＞という，スペースで区切られた文字列が代入される．文字列は左詰で代入されるので，この場合はスペースも含め 11 文字である．

80 行の＜ Q$=TIME$(TIMEDATE) ＞ではたとえば＜ 12:54:32 ＞という＜時 : 分 : 秒＞の時刻を表す 8 文字の文字列が代入される．

90 行は 60 ～ 100 行のループを抜け出す判断である．式＜ Q$[8;1]="0" ＞は文字列 Q$ の 8 文字目から始まる 1 文字が "0" のとき＜ 1 ＞を返し，作業の流れがループを抜け出す．Q$ の意味から，10 秒ごとにループを抜け出すことになる．

110 行では文字列を作成し，配列 A$(I) に代入している．

```
A$(I)="data("&Q$[1,2]&"-"&Q$[4,5]&"-"&Q$[7,8]&").txt"
```

の先頭部分の＜"data("&Q$[1,2]＞は＜"data("＞という文字列と＜Q$[1,2]＞という文字列を＜&＞で連結した文字列である．ここで，Q$[1,2]は文字列Q$の第1文字から第2文字までの文字列のことである．したがって，110行が実行されるとA$(I)には最終的に18文字からなる＜"data(hh-mm-ss).txt"＞という，時刻を表す文字列が代入されることになる．このように文字列の中の一部を取り出すには，90行と110行の2つの方法がある．取り出された文字列の一部をHT-BASICでは「副文字列」と呼んでいる．
120行ではA$(I)の文字列を名前とするファイルを作成している．つまり，時刻を名前の一部に持つファイルを作成している．
130～160行はファイルに書き込むデータを作成するループである．
140行では文字列P$にスペース" "をはさんでTIME$(TIMEDATE)を連結している．
150行で約1秒だけ待つから，

```
B$(J)=P$&" "&TIME$(TIMEDATE)
```

には，70行で読み込んだP$の日付と新しく読み込んだ時刻が，たとえば＜"11 Apr 2004 12:54:33"＞のような連結文字列として代入される．
170行でA$(I)の名前を持つファイルへのI/Oパスを開き，ASCIIフォーマットでのデータ出力を指示している．
180行で配列B$(*)のデータをファイルへ一括して書き込んでいる．(*)は配列全体を指定するワイルドカードの指示子である．
200行のCATは，ファイルが正しく作成されているかどうかを確認するために，ディレクトリのファイルリストを表示する命令である．ファイルが作成されていることを確認できる．CAT 命令には

```
CAT Directory$ TO #10
```

というオプションをつけることで，文字列Directory$で指示されたディレクトリのファイルのリストをプリンタに出力できる．WINDOWSでは#10でデフォルトのプリンタを示している．WINDOWSにはディレクトリのファイルリストをプリントアウトする命令がないから（DOSにはある），この命令は便利である．プリントアウトの文字列の並びがそろわないようならば，HT-BASICの編集画面の上部にあるツールバーでプロポーショナルでないフォントを選ぶとよい．
＜Tool→Device Setup→Win-Print→Properties→Select Font→MS 明朝＞など．
210～290行は，上述した作業を確認するために，作成した3個のファイルから，それぞれデータを一括して読み込み，表示するループである．
220行は区切りの罫線を表示している．
230行はファイル名を表示している．
240行はA$(I)の名前を持つファイルへのI/Oパスを開き，ASCIIフォーマットでのデータ入力指示している．
250行でデータを文字列配列C$(*)へ一括して読み出している．
260～280行で読み出したデータを表示している．
300行ではI/Oポートを閉じている．
上記のプログラムの130～160行の部分を，4.2(1)節で説明したGPIB制御のプログラムに書き換えると測定器からのデータを収録することができる．長期間の実験中には，不慮の停電とか周辺機器の故障などで一連の測定が終了できないことも考えられる．上記プログラムのように一回の測定ごとにデータファイルを作成しディスクに保存しておけば，少なくとも測定が中断するまでのデータは読み出すことができる．

5. HT-BASICのプログラミング2（汎用PCカード（InesDAQ i250）による制御[3)]）

ここではElan社（英）製のAD132というPCカードについて使用方法を解説する．このカードは16チャンネルのA/Dコンバータと8ビットのデジタルI/Oポートを持っている．A/D変換の精度は12ビットで速度は最高250kHzである．このカードを取り上げたのは，通常の計測はこのスペックで満足できると思うことと，Elan社のA/Dコンバータのシリーズの中でこれまでに故障が報告されていないからである．このカードはハードウェアの製造はElan社であるが，ドライバソフトはInes（独）社が作成していて，Ines社による型番はInesDAQ i250となっている．**図3**はInesDAQ i250 PCカードとケーブルである．ソフトウェアのプログラム中ではi250という型番が出てくるので，以下では(InesDAQ) i250と呼ぶことにする．A/Dが16チャンネルというのは，A/D変換ポートを16チャンネル持っているということではなく，1個のA/D変換ポートに対して16チャンネルのアナログ入力信号をスキャンできるという意味である．

5.1 InesDAQ i250 PCカードの概要

上述したように，A/D変換の精度は12ビットで，入力ポート数は16チャンネル，A/D変換のサンプルレートは305.1Hz

図 3　InesDAQ i250 カードとケーブル
Fig. 3　InesDAQ i250 and cable（Ines Co.）.

～250kHz である. 標準のモードでは実数データを読み込むが, 高速読み込みを行うときのために整数モードがある. 整数モードを使用する場合はあとで整数データを実数データに変換する必要がある．入力が 1 チャンネルだけの場合には実数・整数モードともに 250 kHz までのサンプルレートが指定できるが，複数のチャンネルを使用するときは 90 kHz が推奨されている（250 kHz を指定することはできる）．アナログ入力電圧範囲は最大 -10 ～ +10 ボルトで，両極性で 16 通り，単極性（正電圧）で 8 通りの指定ができる．入力チャンネルは 1 チャンネルづつ指定できるが，複数チャンネルを使用するときは連続したチャンネルしか指定できない．入力モードは片側共通接地で 16 チャンネル，入力ポートを 2 本使った差動入力の場合 8 チャンネルになる.

デジタル入出力は TTL レベル（0V ＜ Low ＜ 0.8V, 2V ＜ High ＜ 5V）のポートを 8 ビット持っている．入力と出力の選択は 1,2,4,8 ビット単位で指定できるが，その組み合わせには後述（**表 3**）するように制約がある.
カードからの出力ケーブルには 37 極の D-sub ／ plug タイプコネクタが接続されている．デジタル I/O には統一規格はないので，**表 2** のピン番号にしたがって外部回路を作成するとよい.

5.2　A/D 変換のプログラム

InesDAQ i250 の A/D 変換機能は，チャンネルの選択，変換速度の指定，データ数，トリガ設定など細かな指定ができるようになっている．以下では 1 個づつデータを取る場合と，高速でデータを収録する場合の例を示す.

（1）1 チャンネルで 1 個だけデータを読みこむ

InesDAQ i250 用のドライバをインストールして，以下のプログラムを入力する．一般的に PC カードを操作するときは，いくつかの CSUB（コンパイルされたサブルーチン）を，順を追って呼び出すことが必要になる．次のプログラムは i250 の A/D 機能を使うときの骨格だけを示す.

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   DIM A(1)                                          ! データ配列
40   INTEGER Dummy, Hadc,Cadc
50   DIM L$[256]                                       !256 文字長の文字列変数を定義
60   CALL Daqhtb_ioctl(Dummy,"( fctn init ) ")         !PC カードの初期化
70   CALL Daqhtb_open(Hadc,"i250 ADC",0)               !A/D 機能を実数モード（0）で開く
80   L$="( ioa ( timeout 10000 fsmpl 100000 chan 1 iomode single trigmode off
 samples 1 range [ -10 10 ] ) )"                       !A/D モードの動作定義文
90   CALL Daqhtb_ioctl(Hadc,L$)                        !PC カードへ A/D モードの動作定義文を書き込む
100  CALL Daqhtb_read(Cadc,Hadc,A(*))                  ! データを（1 個だけ）読み込む
110  PRINT A(1)                                        ! 読み込んだデータをプリントする
120  CALL Daqhtb_close(Hadc)                           !PC カードの動作を終了させる
130  END
```

10 行では HT-BASIC で InesDAQ カードを操作するための CSUB（コンパイルされたサブルーチン）を読み込んでいる．この命令が実行されると InesDAQ を使用するのに必要な CSUB がプログラムの最後に追加される.

表 2　InesDAQ PC カードのピン配置
Table 2　Pin assignment of InesDAQ i250.

ピン番号	機能	差動入力	ピン番号	機能
1	A/D入力1	1+	19	DI/O 0
2	A/D入力5	1−	20	DI/O 1
3	A/D入力2	2+	21	DI/O 2
4	A/D入力6	2−	22	DI/O 3
5	A/D入力3	3+	23	DI/O 4
6	A/D入力7	3−	24	DI/O 5
7	A/D入力4	4+	25	DI/O 6
8	A/D入力8	4−	26	DI/O 7
9	A/D入力9	5+	27	+5V
10	A/D入力13	5−	28	+15V
11	A/D入力10	6+	29	−15V
12	A/D入力14	6−	30	接地
13	A/D入力11	7+	31	トリガ入力
14	A/D入力15	7−	32	接地
15	A/D入力12	8+	33	接地
16	A/D入力16	8−	34	接地
17	アナログ接地		35	接地
18	デジタル接地		36	接地
			37	接地

20 行は配列の引数が 1 から始まることを指定している．
30 行はデータを収録するための実数配列を宣言している．配列 A の要素数が 1 個であることに注意していただきたい．
40 行は整数変数の定義である．＜ Dummy ＞，＜ Hadc ＞，＜ Cadc ＞は InesDAQ カードを操作するときに使用される戻り変数であり，名前は任意である．
50 行は 256 文字の文字長を持つ文字列変数を定義している．
60 行では PC カードを初期化している．

```
CALL Daqhtb_ioctl(Dummy,"(fctn init)")
```

引数の中の Dummy は InesDAQ カードを初期化するときに付加する変数でプログラムでは使用しない．2 番目の引数である文字列＜ "(fctn init)" ＞は PC カードの初期化を行う動作定義文であり，このとおりに書かなければならない．初期化では FIFO バッファのサイズ（デフォルトで 1M バイト）と IRQ（interrupt request）発生までのサンプル数（デフォルトで 8192，内部バッファの半分）を指定できる．
70 行は PC カードの A/D 変換機能を呼び出している．引数の中の＜ "i250 ADC" ＞は i250 という名前のカードの ADC（A/D 変換）機能を使用するという意味である．整数変数＜ Hadc ＞へは＜ "i250 ADC" ＞で特定されるハンドル（PC カードの機能を認識する番号）の数値が戻ってくる．これ以降，動作が終了するまで CSUB を呼び出すたびに，必ず変数＜ Hadc ＞が引数として使用される．数値＜ 0 ＞は，動作を実数値で行うことを指示している．高速のデータ収録を行う場合など整数値で読み込む時には＜ 1 ＞を指示する（後述）．
80 行は A/D 変換動作の内容を定義する文字列である．プログラム例では文字列が長いので 2 行に分けて表示してあるが，入力するときには途中で改行＜ ENTER> を入れてはいけない．後でプログラムを修正するときのために，2 行以上に明示的に分けてわかりやすい表示をするには，いくつかの文字列に分割すればよい．80 行の場合には，50 行の＜ DIM ＞文中で文字列変数＜ P$[128],Q$[128] ＞を定義しておけば，以下のように書き直すことが出来る．各指示項目の間はスペースで区切ることに注意すること．

```
50  DIM L$[256],P$[128],Q$[128]
・・・・・・
75  P$="( ioa ( timeout 10000 fsmpl 100000 chan 1 iomode single"
76  Q$="trigmode off samples 1 range [ -10 10 ] ) )"
80  L$=P$&" "&Q$
```

＜ ioa ＞はアナログ信号の I/O 動作（この場合は A/D 変換）を行うことを示し，続く＜（ ）＞のなかに細かく動作を指示している．

```
timeout 10000        ! 動作のタイムアウト時間が 10000 ミリ秒
fsmpl 100000         ! サンプリング周波数が 100 kHz
chan 1               !1 番のチャンネルの電圧信号を A/D 変換する
iomode single        ! 電圧信号の入力モードはシングルエンド（片側接地）
trigmode off         ! トリガモードは使用しない
samples 1            ! 読み取るデータは 1 個のみ
range [-10 10]       ! 入力電圧範囲は -10 ～ +10 ボルト
```

各文字や数値の間にはスペースを入れる．この他にも機能が指示できるが，指示しない機能にはデフォルトの値が与えられる．電圧信号の入力モードは 2 種類ある．シングルエンドの場合には 16 チャンネルが使用でき，2 個のチャンネルを使う差動入力モードの場合には，対になるチャンネル番号は決まっていて，**表 2** のように 8 チャンネルが使用できる．トリガモードは細かく指定できる．入力電圧範囲は既定で，正負にわたるもの（16 種）と正のみの範囲のもの（8 種）を指定できる．
90 行では 80 行の動作定義文をハンドル番号＜ Hadc ＞で指定された機能に書き込んでいる．
100 行で配列＜ A(*) ＞に指定された個数だけデータを読み込んでいる．本例ではデータを 1 個だけ読み込むので配列 A(*) の要素の数は 1 個にしたが，1 以上であればよい．戻り変数＜ Cadc ＞には読み込んだデータの数が入っている．
110 行でデータをディスプレイにプリントアウトしている．
120 行では PC カードの動作を終了させている．InesDAQ ではいろいろな動作を指定して操作することができるが，新しい動作の指定を行うときには必ずその前の動作を終了しておかなければならない．
130 行はプログラムの終了文である．
この例のように InesDAQ は初期化から終了まで 5 種類の一連の CSUB を呼び出している．それらを処理の順に並べると：

① PC カードの初期化を行う．
② A/D 変換入力，あるいはデジタル I/O 等の機能を選ぶ．
③ 動作内容の細かな設定を指示する．
④ データの入力あるいは出力を行う．
④’ 整数データを実数値に変換する．
⑤ PC カードの動作を終了させる．

となる．④’は高速にデータを読み込む場合に必要な処理で，(4) ②「整数モードの入力」の項で説明する．

(2) 複数のチャンネルで 1 個づつデータを読み込む

複数のチャンネルで 1 個づつデータを読み込む場合のプログラムは以下のようになる．プログラムの流れは 1 チャンネルのときと変わらない．

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"       !CSUB の読み込み
20   OPTION BASE 1
30   DIM A(2),L$[256]
40   INTEGER Dummy,Hadc,Cadc
50   CALL Daqhtb_ioctl(Dummy,"( fctn init) ")
60   CALL Daqhtb_open(Hadc,"i250 ADC",0)
70   L$="( ioa ( timeout 10000 fsmpl 100000 chan 102 iomode single
 trigmode off samples 2 range [ -10 10 ] ) )"
80   CALL Daqhtb_ioctl(Hadc,L$)
90   CALL Daqhtb_read(Cadc,Hadc,A(*))
100  PRINT A(1),A(2)
110  CALL Daqhtb_close(Hadc)
120  END
```

30 行では 2 個の要素をもつデータ読み込み用の配列 A(*) を定義している．
50 行は PC カードの初期化．

60 行は実数モード＜ 0 ＞による A/D 変換機能を open させている.
70 行では動作を指示している．プログラム例では文字列が長いので 2 行に分けて表示してあるが，入力するときには途中に改行＜ ENTER> を入れてはいけない.

```
fsmpl 100000       ! サンプリング周波数が 100 kHz
chan 102           ! チャンネル 1 からチャンネル 2 までの信号を A/D 変換する
samples 2          ! 読み取るデータ数は 2 個
trigmode off       ! トリガモードは OFF
```

複数の A/D 入力チャンネルを使用する場合は，InesDAQ ではハード上の制約があって測定するチャンネルをランダムに選ぶことはできない．必ず連続した複数のチャンネルを指定しなければならない．最後のチャンネルの番号が 1 桁の場合には番号の前に 0 を入れる．複数チャンネルのデータ収録を行う場合にトリガモードを指定すると，どのチャンネルからデータの読み込みが始まったかわからなくなるので，推奨されていない．ただし，指定した複数のチャンネルのうち最初の A/D 入力チャンネルをトリガ専用に使用して，入力電圧レンジを超えたレベルでトリガさせるようにすると，次のデータは 2 番目のチャンネルに入力されたことになる．データを取り終わったあとで, 先頭チャンネルのデータを廃棄するようにすれば, トリガによる複数チャンネルの A/D 入力が実現できる.サンプリング周波数は 2 チャンネル合計で 100kHz という意味である.
80 行で，PC カードの番号 Hadc で指示された機能に動作定義文 L$ を書き込んでいる.
90 行では 1,2 チャンネルからデータを 1 個づつ，計 2 個，読み込んでいる．1 チャンネルのデータは A(1) へ，2 チャンネルのデータは A(2) へ読み込まれている.
100 行はプリントアウト.
110 行は PC カードの動作の終了.
一定の時間間隔を置いて, 連続してデータを読み込む場合には（1）あるいは（2）で示した動作を, ループ（6 章参照）の中に入れて繰り返せばよい．その際，4.3 (1), (2) 節で説明したようにファイルの中にデータを書き込むようにすれば，ハードディスクの容量まで，ほとんど無制限にデータを記録することができる.

(3) 1 チャンネルで高速にデータを取り込む

通常は決まった時間間隔でデータを 1 個づつ測定していても，あるイベントが発生したときには高速でデータを収録し，その変化を観測したいときがある．そのような場合のプログラム例を次に示す．1 チャンネルの場合 250 kHz までのサンプリングが可能である.

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   DIM A(1024) ,L$[256]
40   INTEGER Dummy, Hadc,Cadc
50   CALL Daqhtb_ioctl(Dummy,"( fctn init) ")
60   CALL Daqhtb_open(Hadc,"i250 ADC",0)
70   L$="( ioa (timeout 10000 fsmpl 100000 chan 1 iomode single
 trigmode level.gt triglevel -10.0 pretrig 0 samples 1024 range [ -10 10 ] ) )"
80   CALL Daqhtb_ioctl(Hadc,L$)
90   CALL Daqhtb_read(Cadc,Hadc,A(*))              !64(2^6) < A(*)=2^n < 16348(2^14)
100  ------------------------------------
110  CALL Daqhtb_close(Hadc)
120  END
```

30 行でデータバッファとなる配列 A(*) を定義している.
60 行では実数値に変換することを指定（0）.
70 行では動作を指定している．プログラム例では文字列が長いので 2 行に分けて表示してあるが，入力するときには途中に改行＜ ENTER> を入れてはいけない.

```
fsmpl 100000           ! サンプル周波数は 100 kHz
trigmode level.gt      ! トリガモードは“信号値が指定レベル以上のとき”
triglevel -10.0        ! トリガレベルは - 10 ボルト
pretrig 0              ! プレトリガデータは読み込まない
```

```
samples 1024            ! 読み込みデータ数は 1024 個
```

読み込みデータの長さは 64(2^6) から 16348(2^14) の範囲で，2^n の数値で指定する．トリガモードを指定（＜ trigmode off ＞でない場合）すると，常に A/D 変換が行われるから，トリガがかかった瞬間から前のデータも測定されている．そのデータをどこまで読み込むのかを＜ pretrig ＞で指定する．
80 行で A/D 変換モードの動作定義文を書き込む．
90 行で配列 A(*) に読込んでいる．バッファがいっぱいになると読込みを中止する．
本プログラム例では，1 番のチャンネルからの電圧を監視してトリガ信号を出している．トリガ電圧として 70 行で指示したレンジ範囲を超える電圧（-10V 以下，あるいは +10V 以上）を指示する場合には，**表 2** に示されているトリガ入力ピン 31 番にトリガ信号を印加する．

(4) 複数のチャンネルで高速にデータを取り込む

複数のチャンネルで高速にデータを取り込む場合には，実数配列にデータを読み込む場合と整数配列に読み込む場合とがある．実数配列に読み込む場合は A/D の結果を実数に変換するのに時間がかかる．変換を行わずに整数配列に読み込めば高速のデータ収録が行えるが，後で整数データを実数に変換しなければならない．複数チャンネルの場合には 90 kHz までのサンプリングが推奨されている．

① 実数モードでのデータ収録

2 個のチャンネルの A/D ポートからデータを入力するプログラムを以下に示す．

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   DIM A(1024),Dat(512,2),L$[256]          ! 配列定義
40   INTEGER Dummy,Hadc,Cadc
50   CALL Daqhtb_ioctl(Dummy,"(fctn init)")
60   CALL Daqhtb_open(Hadc,"i250 ADC",0)
70   L$="( ioa ( timeout 10000 fsmpl 90000 chan 102 iomode single
 trigmode off samples 1024 ) )"
80   CALL Daqhtb_ioctl(Hadc,L$)
90   CALL Daqhtb_read(Cadc,Hadc,A(*))        ! 配列 A(*) への読み込み
100  CALL Daqhtb_close(Hadc)
110  FOR J=1 TO Cadc/2.                      ! データを各チャンネルに振り分ける
120     Dat(J,1)=A(2*J-1)
130     Dat(J,2)=A(2*J)
140  NEXT J
150  END
```

30 行では読み込むデータの配列を宣言している．2 チャンネルを使い，配列 A(*) に 1024 個のデータを収録する．そのあとで各チャンネルごとに 512 個のデータを配列 Dat(*) に振り分ける．
50 行は PC カードの初期化．
60 行では，PC カードの A/D 変換機能を実数モード＜ 0 ＞で開く．
70 行は動作を指定している．プログラム例では文字列が長いので 2 行に分けて表示してあるが，入力するときには途中に改行＜ ENTER> を入れてはいけない．

```
fsmpl 90000        ! サンプルレートは 90 kHz
chan 102           ! 連続するチャンネル 1 から 2 までの A/D 入力をスキャンする
iomode single      ! 片側接地入力
trigmode off       ! トリガモードは使わない
samples 1024       !2 チャンネル合計で 1024 個のデータを読み込む
```

A/D 入力に使用するチャンネルは連続したものでなければならない．片側接地の場合には 16 チャンネル，差動入力の場合には 8 チャンネルを指定できる．入力ケーブルの接続先は**表 2** に示してある．
80 行で動作内容を PC カードに送っている．
90 行でデータを読み込んでいる．A/D コンバータは指示されたサンプルレートでチャンネルを切り替えながらデータを配

列に送り込むので，A(*) には 2 チャンネル交互に，合計 1024 個のデータが読み込まれる．戻り変数 Cadc には読み込んだデータ数が入っている．
100 行でカードの動作を終了している．
110 ～ 140 行は，A(*) に読み込まれたデータをチャンネル 1 と 2 に振り分けるループである．各チャンネルのデータ数は，＜ Cadc/2 ＞である．

②整数モードでのデータ収録

実数モードの測定では，測定のたびに実数への変換時間がかかる．そこで，高速の現象を観測するために実数への変換を行う前に，2 値信号のままデータを収録し，測定が終了してから実数に変換することが行われる．整数モードの場合，PC カードの整数データが 32 ビット（4 バイト）なのに対し，HT-BASIC の整数データは 16 ビット（2 バイト）であるから，読み込みたいデータ数の 2 倍の要素をもつ配列を用意しておく必要がある．以下にそのプログラム例を示す．測定するデータ数は，例では各チャンネルごとに 256 個であるが，2^n で任意に指定できる．

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   INTEGER A(1024)                          ! 整数データ配列の宣言
40   DIM R(512),Dat(256,2)                    ! 実数データ配列の宣言
50   INTEGER Dummy,Hadc,Cadc                  ! 戻り整数変数の宣言
60   DIM L$[256]                              ! 動作指示のための文字列変数の宣言
70   CALL Daqhtb_ioctl(Dummy,"(fctn init)")   !PC カードの初期化
80   CALL Daqhtb_open(Hadc,"i250 ADC",1)      ! 整数モード (1) で読み込む
90   L$="( ioa ( timeout 10000 fsmpl 100000 chan 102 iomode single
 trigmode off samples 1024 ) )"
100  CALL Daqhtb_ioctl(Hadc,L$)
110  CALL Daqhtb_readi(Cadc,Hadc,A(*))        ! 整数配列へのデータ読み込み
120  CALL Daqhtb_demangle(R(*),A(*),L$,0)     ! 整数入力の実数値への変換
130  CALL Daqhtb_close(Hadc)
140  FOR J=1 TO Cadc/2.                       ! 実数データの各チャンネルへの振り分け
150     Dat(J,1)=R(2*J-1)
160     Dat(J,2)=R(2*J)
170  NEXT J
180  END
```

30 行は A/D 変換した binary データをそのまま整数値として読み込むための配列を宣言している．PC カードでは 32 ビットの整数値を出力するが，HT-BASIC では整数値は 2 バイト（16 ビット）で扱われる．1 個の整数データを読み込むには 2 個の整数配列要素が必要である．A(1024) というのは，整数データ 512 個分の配列である．
40 行は実数配列の宣言である．R(512) は A(*) に読み込まれた 1024 個のデータを実数値に直したデータを保存する配列，Dat(256,2) は各チャンネルに振り分けられた 256 個のデータを収納する配列である．
80 行では PC カードの A/D 変換機能を，整数変数でおこなうように指示＜ 1 ＞している．
90 行は動作内容の指示である．プログラム例では文字列が長いので 2 行に分けて表示してあるが，入力するときには途中に改行＜ ENTER> を入れてはいけない．：

```
fsmpl 100000      ! サンプルレート 100 kHz
chan 102          ! 連続するチャンネル 1 から 2 までの A/D 入力をスキャンする
iomode single     ! 片側接地入力
trigmode off      ! トリガモードは使わない
samples 1024      !2 チャンネル合計で 1024 個のデータを読み込む
```

110 行で配列 A(*) に読み込むわけだが，上述したように，A の要素 2 個で PC カードの整数データ 1 個に相当する．したがって samples 1024 では実際には 512 個の整数データが配列 A の要素に読み込まれることになる．
100 行では PC カードに動作内容を書き込んでいる．
110 行でデータを配列 A(*) に読み込んでいる．実数モードの時は＜ Daqhtb_read ＞だったが，整数モードでは整数（integer）を示す＜ Daqhtb_readi ＞に変わっている．

120 行は，整数モードのときに必要な機能で，整数データを実数データに変換している．配列 A(*) のデータ数は 1024 個であるが，実際に読み込まれたデータ数は 512 個なので，実数配列 R(*) の要素は 512 個である．
130 行で動作を終了させている．
140 行からのループで各チャンネルごとにデータを振り分けている．

5.3　ビット入出力のプログラム

InesDAQ i250 のデジタル I/O 機能は，合計 8 個あるビットポートの機能（入力か出力か）を指定し，出力するデータの書き込み，および入力データの読み込みができるようになっている．各ビットポートの指定の仕方を**表 3** に示す．カードの大きさの制約から，各ポートの入力･出力の指定を任意におこなうことは出来ず，**表 3** に示すように 1，2，4，8 個づつまとめて指定するようになっている．実用上は不都合はないと思われる．

以下ではビット出力の場合と，ビット入力の場合の例を示す．

(1) ビットデータの出力

次のプログラムを入力する．CSUB（コンパイルされたサブルーチン）の呼び出し順序は A/D 変換のときと変わらない．

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   DIM D(1)
40   INTEGER Dummy,Hdio,Cdio
50   CALL Daqhtb_ioctl(Dummy,"(fctn init)")          ! カードの初期化
60   CALL Daqhtb_open(Hdio,"i250 DIO",1)             ! デジタル I/O 機能を指定
70   CALL Daqhtb_ioctl(Hdio,"( iod ( dir 255 ) )")   ! デジタル I/O ポートの指定
80   INPUT P                                         ! キーボードからの 10 進数の入力
90   D(1)=BINAND(P,255)                              !2 進数表現の下位 8 ビットのみを残す
100  CALL Daqhtb_write(1,Hdio,D(*))
110  CALL Daqhtb_close(Hdio)
120  END
```

10 行では PC カードを制御するための CSUB を一括して読み込んでいる．
20 行は，配列の指示子を（0 からではなく）1 から始まるように指定している．
30 行はビット出力値を与えるための要素数 1 個からなる配列 D(*) である．
60 行では InesDAQ i250 のデジタル I/O 機能を整数モードで開いている．
70 行では表 2 にしたがい，デジタル I/O のポートを 8 ビットとも出力用に指定している．
80 行ではキーボードから数値を入力している．
90 行では入力された数値の下位 8 ビットだけを取り出し，配列 D(1) に代入している．BINAND(A,B) は 2 個の数値 A と B の 16 ビット表現のビットごとの AND を取り出す関数である．
100 行では 1 個のデータを書き込んでいる．
110 行で動作を終了．

(2) ビットデータの入力

70 行でビットデータの出力機能を指定しているほかは，ビットデータ出力の場合と同じ．

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   DIM D(1)                                        ! ビットデータを読みこむための配列
40   INTEGER Dummy,Hdio,Cdio
50   CALL Daqhtb_ioctl(Dummy,"(fctn init)")          ! カードの初期化
60   CALL Daqhtb_open(Hdio,"i250 DIO",1)             ! 整数モードでのデジタル I/O 機能
70   CALL Daqhtb_ioctl(Hdio,"( iod ( dir 0 ) )")     !8 ビットのポートを入力に指定
80   ----------
90   CALL Daqhtb_read(Cdio,Hdio,D(*))                ! 配列 D にデータを読み込む
100  CALL Daqhtb_close(Hdio)                         ! 動作終了
110  END
```

70 行では 8 ビットのポート全部を入力ポートに指定している．
各ビットの入力は TTL レベル（0V ＜ Low ＜ 0.8V, 2V ＜ High ＜ 5V）で与えなければならない．**図 4** は GPIO（16 ビットの入出力を並列で行うビット制御バス）ラインで使用する外部回路の例を示す．このボードでは GPIO ラインにより，16 個の外部リレーの駆動，2 チャンネルの 16 ビットデジタルカウンタからの入力，交流 100 ボルトのリレーの駆動，およびランダム信号の発生制御が行えるようになっている．**図 5** は InesDAQ i250 のデジタル I/O 機能のテスト回路である．16 チャンネルのアナログ入力，8 ビットのデジタル入出力機能がテストできる．

6. HT-BASIC のプログラミング 3（プログラムの構造（ループと分岐，計算））

前節では，C 言語などでは簡単にはできないインタラクティブなプログラムに焦点を当てて特徴を説明した．本章では通常のプログラミングの手法について概観する．

(1) ループと分岐（FOR ～ NEXT，IF ～ THEN，LOOP，WHILE，REPEAT，SELECT ～ CASE）

① FOR ～ NEXT

```
10   FOR I=1 TO 100
20      FOR J=2*PI TO 0 STEP  － PI/100
        -----
80      NEXT J
90   NEXT I
```

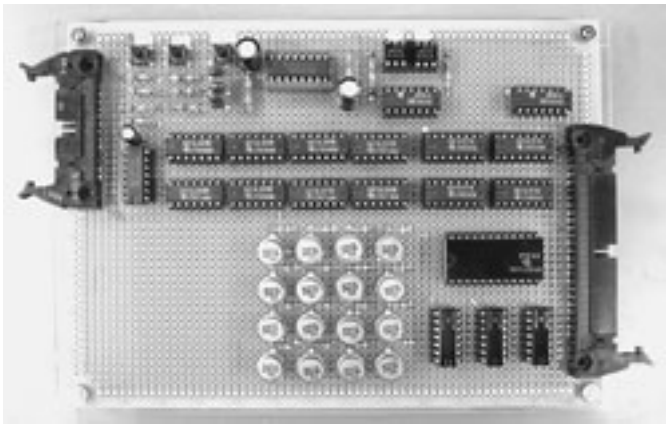

図 4　GPIO 制御ボード（16 ビット並列入出力）用の外部回路の例
Fig. 4　An external actuator of GPIO output.

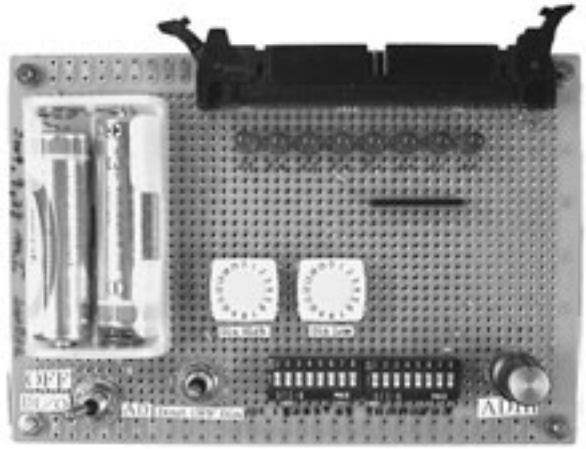

図 5　Ines DAQ i250 カードの入出力テスト回路の例
Fig. 5　An external test board of Ines DAQ i250.

表 3　ビットポートの指定方法（I：入力，O：出力）
Table 3　Assignment code of InesDAQ digital Port（I : input, O : output）.

16進表示	0x0	0x1	0x2	0x3	0xC	0xD	0xE	0xF	0xF0	0xF1	0xF2	0xF3	0xFC	0xFD	0xFE	0xFF
10進表示	0	1	2	3	12	13	14	15	240	241	242	243	252	253	254	255
DIO 0	I	O	I	O	I	O	I	O	I	O	I	O	I	O	I	O
DIO 1	I	I	O	O	I	I	O	O	I	I	O	O	I	I	O	O
DIO 2	I	I	I	I	O	O	O	O	I	I	I	I	O	O	O	O
DIO 3	I	I	I	I	O	O	O	O	I	I	I	I	O	O	O	O
DIO 4	I	I	I	I	I	I	I	I	O	O	O	O	O	O	O	O
DIO 5	I	I	I	I	I	I	I	I	O	O	O	O	O	O	O	O
DIO 6	I	I	I	I	I	I	I	I	O	O	O	O	O	O	O	O
DIO 7	I	I	I	I	I	I	I	I	O	O	O	O	O	O	O	O

20 行のように 2π から 0 まで -π/100 づつ変化させるように STEP で増（減）分を指定できる．指定しない場合は 1 とみなされる．

② IF ～ THEN（～ ELSE ～ END IF）

条件式や論理式はその評価結果が true の場合に＜ 1 ＞（正）を返す．

```
10   IF J2=K THEN 90
        -----
30   IF X=Y THEN Y=Z*Z
        -----
40   IF A<0 THEN
50      GOTO End
60   ELSE
        --------
80   END IF
90 End:END
```

10 行は＜ J2=K ＞という論理式が true の場合に 90 行へプログラムの流れが分岐する．

30 行では条件式が true の場合 Y=Z*Z を実行する．

40 ～ 80 行では A<0 が true の場合に End という名前を持つ 90 行へ分岐する．そうでない（false）場合は ELSE 以下のプログラムを実行する．各行には自由に名前をつけることができる．行の名前のあとはコロン＜ : ＞で区切る．

③ LOOP ～ EXIT IF ～ END LOOP

```
100  LOOP
        ----
170  EXIT IF J=5 OR A$>B$
        ----
190  END LOOP
```

100 ～ 190 行を繰り返し実行し，170 行の論理式が true の場合に LOOP を抜け出す．ループの途中で条件判断を行う．

④ WHILE ～ END WHILE

```
100  WHILE X<1000
        -----
200  END WHILE
```

100 行の論理式が真（true）の場合に 200 行までのプログラムを繰り返す．ループの先頭で条件判断を行う．

⑤ REPEAT ～ UNTIL

```
60   REPEAT
        ----
80   UNTIL X=10
```

80 行の論理式が真（true）のなるまで REPEAT に戻り処理をくり返す．ループの最後で条件判断を行う．

⑥ SELECT ～ CASE ～ END SELECT

```
10   SELECT Option$
20   CASE "B"
30      A=1
40   CASE "0" TO "9","y","n"
50      A=2
60   CASE ELSE
70      A=0
```

```
80    END SELECT
```

10 行で Option$ という文字列変数を選び，20 行および 40 行の CASE 文でその内容を比較している．論理式＜ Option$="B" ＞が真（true）ならば 30 行を実行し，どの CASE 文とも一致しない場合には 70 行を実行する．数値変数を SELECT した場合には数値による論理式を使う．

(2) サブプログラムと関数 1（SUB，COM，FN）

① 主プログラムと変数を共有しない場合

```
10    CALL Mysub              ! サブプログラムを呼ぶ
20    END                     ! 主プログラムの終了
30    SUB Mysub               ! サブプログラムの定義
40       PRINT "In My SUB"
50    SUBEND
```

10 ～ 20 行が主プログラムで，Mysub というサブプログラムを呼んでいる．サブプログラムは主プログラムの終わりを示す＜ END ＞文の後に記述する．

② 主プログラムと変数を共有する場合（COM）

```
10    COM /Test/A$(2)[20],P                    ! 名前をつけた共通変数の宣言
20    A$(1)="He"
30    A$(2)="llo"
40    P=7
50    Subit                                    ! サブプログラムを呼ぶ
60    END                                      ! 主プログラムの終了
70    SUB Subit                                ! サブプログラムの定義
80    COM /Test/C$(*),R                        ! 共通変数の宣言
90       PRINT "A$(*)=";C$(1)&C$(2),"P=";R     ! 文字を連結して表示
100   SUBEND
```

10 行の＜ COM ＞文で共通変数を定義している．＜ /Test/ ＞は定義された変数をひとつのブロックとしてつけた名前である．

50 行でサブプログラムを呼んでいる．通常は CALL 文を使って＜ CALL Subit ＞のように呼び出さねばならないが，サブプログラムを呼ぶだけの行ならば CALL を省略できる．

80 行では，例として主プログラムと異なる変数名を使用しているが，理由のない限り変数名は主プログラムと同じにしたほうが混乱しない．

③ 引数がある場合

```
10    DIM A$(2)[20]
20    A$(1)="He"
30    A$(2)="llo"
40    P=7
50    Subit(A$(*),P,Q)                         ! 引数によるデータの送付
60    PRINT Q
70    END
80    SUB Subit(C$(*),R,S)                     ! 引数によるデータの受領
90       PRINT "A$(*)=";C$(1)&C$(2),"P=";R
100      S=R*R                                 ! 演算結果を引数で戻す
110   SUBEND
```

80 行では変数の並び順を 50 行と同じ順にそろえること．SUB プログラムを作成する場合には主プログラムラインの最後に追加しなければならない．80 行の引数 C$(*) を，たとえば 81 行で，10 行のように定義しようとすればエラーになる．

(3) サブプログラムと関数 2（GOSUB，FN，CSUB）

① 主文脈中のサブルーチン（GOSUB）

HP-BASIC では主プログラム中にサブルーチンをおくこともできる．この場合，GOSUB 命令で作業の流れが指定された行に分岐し RETURN 文で GOSUB 命令の次の行に戻るので作業の流れに注意しなければならない．

```
10   Y=3
20   Z=4
30   GOSUB Calc_x
40   PRINT "X = ";X
50   STOP
60 Calc_x:X=Y*45/Z
70   RETURN
80   END
```

30 行で 60 行のサブルーチンに飛び，70 行の RETURN 文で戻るが，50 行の STOP 命令がなければ再度 60 ～ 70 行を実行することになる．＜ STOP ＞命令では＜ END ＞と同じく，プログラムは idle 状態に戻る．＜ CONTINUE ＞（続行）は出来ない．

② 関数（DEF FN）

任意の関数を定義して，プログラム中で使用できる．

```
10   PRINT "5 + 8 =";FNAdd(5,8)   ! 関数の引用
20   END                          ! 主プログラムの終了
30   DEF FNAdd(A,B)               ! 関数の定義
40      RETURN A+B
50   FNEND
```

30 ～ 50 行が関数の定義である．主プログラム文のあとに定義を付加する．関数名は FN から始まること．この場合 CALL 文は必要ない．

③ コンパイル済みサブプログラム（CSUB）

HT-BASIC には多くの数学関数がコンパイル済みのサブプログラムとして用意されている．その使用例を以下に示す．

```
10   LOADSUB ALL FROM "C:\Program Files\HTBwin92\mathlib\Mathlib.hts"
        -----
100  A=FNErf(X)
        -----
300  END
```

100 行では X という引数の関数値 FNErf(X) を A に代入している．関数 FNErf(X) は誤差関数である．コンパイル済みのサブプログラムを利用するには，サブプログラムファイルを前もってロードしておかねばならない．数学関数のコンパイル済みサブプログラムはデフォルトのディレクトリの中のファイル，

```
C:\Program Files\HTBwin92\mathlib\Mathlib.hlp
```

に，一覧リスト，関数の解説，および使用するに際しロードしておくべきサブルーチンファイル名などが記してある．特に上記プログラム例の 10 行に記した，

```
C:\Program Files\HTBwin92\mathlib\Mathlib.hts
```

にはすべての数学関数の CSUB プログラムを含んでいるので，これをロードしておけば，準備されたすべての数学関数が使用できる．ただし，CSUB サブルーチンが末尾に付加されたプログラムは ASCII ファイルとしては保存できない．RE-STORE 命令で保存し，LOAD 命令で呼び出さなければならない．

(4) 行列計算（MAT）

行列およびベクトル相互の演算は MAT 演算子により簡単に行うことができる．以下のプログラムでいくつかの例を示す．

```
10   OPTION BASE 1                      ! 配列指示子の最小値を指定
20   DIM A(3,3),B(3,3),C(3,3)           !3 行 3 列のマトリクス（配列）を定義
30   DIM V1(3),V2(3)                    !1 行 3 列のベクトルを定義
40   DATA 1,3,2,-1,-2,-1,2,4,3          ! マトリクス要素のデータ
50   RESTORE 40                         ! 次の READ 文で読み取る DATA を指定
60   READ A(*)                          !A(*) に DATA を読み込む
70   CALL Subdisp1("A(*) = ",A(*))      ! サブプログラムによる A(*) の要素の表示
80   MAT B=INV(A)                       !A(*) の逆行列を B(*) に代入
90   Subdisp1("INV A(*) = ",B(*))       !B(*) の要素の表示
100  MAT C=B*A                          !B(*) と A(*) との積を C(*) に代入
110  Subdisp1("INV(A)*A = ",C(*))       !C(*) の要素の表示
120  MAT V1=B(1,*)                      !B(1,*) の要素をベクトル V1(*) に代入
130  Subdisp2("INV(1) = ",V1(*))        ! サブプログラムによる V1(*) の要素の表示
140  MAT V2=V1*A                        ! ベクトル V1(*) と A(*) との積を V2(*) に代入
150  Subdisp2("INV(1)*A = ",V2(*))      !V2(*) の要素の表示
160  END                                ! 主プログラムの終了
170  SUB Subdisp1(P$,Q(*))              ! サブプログラム 1 の定義
180     PRINT ""                        ! 空白行の表示（印字）
190     PRINT P$
200     FOR I=1 TO 3
210        PRINT USING "3(10D.DD)";Q(I,1),Q(I,2),Q(I,3)       ! 書式付表示（印字）
220     NEXT I
230  SUBEND
240  SUB Subdisp2(P$,Q(*))              ! サブプログラム 2 の定義
250     PRINT ""
260     PRINT P$
270     PRINT USING "3(10D.DD)";Q(*)                          ! 書式付表示（印字）
280  SUBEND
```

40 行はプログラム中でデータを与える DATA 文である．データは＜,＞で区切る．
50 行は次に出てくる READ 文で読み取るべき DATA 文を行番号で指定している．
60 行で行列（配列）A(*) に DATA 文の内容が読み込まれる．DATA 文から読み込まれる順序は，次元を左側から特定し，最後の列に対して順番に読み込まれる．たとえば，P(2,3,2) という配列には，

```
P(1,1,1), P(1,1,2), P(1,2,*), P(1,3,*), P(2,1,*), ・・・P(2,3,*)
```

という順序にデータが読み込まれる．
70 行は行列 A(*) の内容を表示するためのサブプログラムを呼び出している．
80 行は A(*) の逆行列を B(*) に代入し，100 行は B(*) と A(*) との積を C(*) に代入している．通常の代入文の前に MAT を付加すると行列の代入文になる．
90 行，110 行は 70 行と同じサブプログラムを呼んでいる．サブプログラムを呼ぶだけの行ならば CALL を省略できる．
120 行は B(1,*) の要素をベクトル（一次元配列）V1(*) に代入している．配列から配列への代入は，要素の数が対応するように注意が必要．
130 行，150 行はベクトルの要素を表示するサブプログラムを呼んでいる．
140 行はベクトル V1(*) と行列 A(*) との積を V2(*) に代入している．一次元配列 V1(*) は左側から掛けるときには行ベクトルとして，右側から掛けるときには列ベクトルとして処理される．

```
MAT V2=V1*A    ! V1(*) は行ベクトル，結果は行の一次元配列
MAT V2=A*V1    ! V1(*) は列ベクトル，結果は列の一次元配列
```

170 ～ 230 行は行列（配列）を表示するサブプログラム，240 ～ 280 行はベクトル（一次元配列）を表示するサブプログラ

ムである．両方とも（P$,Q(*)）で，文字列と配列データを引数として呼び出されている．
210 行，270 行は書式付の表示文である．

```
PRINT USING "3(10D.DD)";Q(I,1),Q(I,2),Q(I,3)
```

＜"3(10D.DD)"＞は＜整数部 10 桁，小数点，小数部 2 桁＞による表示を 3 回繰り返すことを指示している．表示するデータは＜；＞のあとに並べる．270 行では Q(*) でデータ列を一括指示している．HP-BASIC ではデータのやり取りの際に，細かな指示をすることができる．書式についてはレファレンスマニュアルの＜ IMAGE ＞の項に詳しい説明がある．また，example フォルダには各命令を使用するプログラム例が示してある．

(5) 論理式

論理式の判断は，true の場合 1 を返し，false の場合 0 を返す．したがって，論理式を含む数式を作ることができる．たとえば，

```
PRINT ((A=B) OR (C>D))*PI
```

は (A=B) および (C>D) が両方ともに false の場合には 0，それ以外の場合には PI（円周率）を出力（表示・印字）する．

(6) ビット操作

16 ビット数で表現された数値のビット操作に関しては以下の命令がある．

```
BINAND(A,B)     ! 二つの 16 ビット数のビットごとの AND をとる.
BINCMP(A)       !16 ビット数の各ビットを反転する.
BINEOR(A,B)     ! 二つの 16 ビット数の各ビットごとの排他的論理和をとる.
BINIOR(A,B)     ! 二つの 16 ビット数のビットごとの OR をとる.
BIT(A,5)        !16 ビット数の指定されたビットを戻す.
ROTATE(A,2)     !16 ビット数のビット列を右方向あるいは左方向に回転させる.
SHIFT(A,2)      !16 ビット数のビット列を右方向あるいは左方向に移動させる.
```

BIT 命令中の 5 はビット位置を示している，ビット位置は，＜ OPTION BASE ＞の指定にかかわらず，最下位（右端）ビットを 0, 最上位（左端）ビットを 15 として指示する．ROTATE 命令中の 2 は正ならば右（下位）方向に数値の大きさだけ回転させる．端からはみ出たビットは他の端に戻る．SHIFT 命令中の 2 は正ならば右（下位）方向に数値の大きさだけ移動させる．はみ出たビットは消え，他端には 0 が入る．

(7) 複素数

複素数は 2 個の実数（8 バイト）の組（16 バイト）として取り扱われる．

```
COMPLEX A,B          ! 複素変数の宣言文
A=CMPLX(2,1)         !A=2+i
B=CMPLX(5,1/5)*A     !B=(5+i1/5)*(2+i)=9.8+i5.4
```

7. HT-BASIC のプログラミング 4（グラフィックス）

HP-BASIC は多様なグラフィックス命令を持っている．本章ではグラフィックスに関するプログラミングの例を示す．

7.1 グラフ出力

(1) 関数のグラフを書く

本節では，ＸＹ座標平面で関数のグラフ（**図 6**）を書く例をとりあげ，グラフィックスを出力する作業の流れを示す．

```
10   GINIT                          ! 描画出力のためのパラメータを初期化する
20   PLOTTER IS CRT,"INTERNAL"      ! 描画の出力先をディスプレイに指定
30   Xlow=-1                        ! 出力するグラフのＸ軸の最小値（左側の限界）
40   Xhigh=1                        ! 出力するグラフのＸ軸の最大値（右側の限界）
50   Xlength=Xhigh-Xlow             ! 出力するグラフのＸ軸の値の長さ
60   Ylow=-1                        ! 出力するグラフのＹ軸の最小値（下側の限界）
70   Yhigh=1                        ! 出力するグラフのＹ軸の最大値（上側の限界）
```

```
80   Ylength=Yhigh-Ylow                ！出力するグラフのＹ軸の値の長さ
90   VIEWPORT 40,120,20,90             ！出力デバイス上の描画範囲を指定
100  WINDOW Xlow,Xhigh,Ylow,Yhigh      ！描画範囲にユーザ指定の値を定義
110  FRAME                             ！描画範囲を実線で囲む
120  AXES Xlength/20,Ylength/20,0,0,5,5,2     ！座標軸を描く
130  CLIP OFF                          ！描画範囲の制限を解除する
140  F$="F(X)=X*X*X"                   ！文字列の代入
150  MOVE 0,Yhigh                      ！描画ペンを指定された座標に移動する
160  LORG 4                            ！座標に対応させるラベルの位置を指定する
170  LABEL F$                          ！文字列ラベルを出力する
180  CLIP ON                           ！描画範囲を制限する
190  PEN 6                             ！ペンの選択
200  FOR X=Xlow TO Xhigh STEP Xlength/20      ！線を描画するためのループ
210     PLOT X,X*X*X                   ！線を描画する
220  NEXT X
230  CSIZE 3                           ！文字のサイズを指定
240  LORG 1                            ！座標に対応させるラベルの位置を指定する
250  PEN 2                             ！ペンの選択
260  FOR X=Xlow TO Xhigh STEP .5       ！Ｘ座標軸の目盛値を描くループ
270     MOVE X,0                       ！描画ペンを指定された座標に移動する
280     LABEL X                        ！数字ラベルを出力する
290  NEXT X
300  FOR Y=Ylow TO Yhigh STEP .5       !Y 座標軸の目盛値を描くループ
310     MOVE 0,Y                       ！描画ペンを指定された座標に移動する
320     LABEL USING "DD.D";Y           ！数字ラベルを出力する
330  NEXT Y
340  END
```

10 行は描画出力のためのパラメータを初期化している．初期化されるパラメータは，描画ペンの種類・LABEL で描く文字の大きさ・座標に対する文字の配置などが含まれている．
20 行は描画出力をディスプレイに出力するように指定している．この文の代わりに，

```
PLOTTER IS 712,"HPGL"
```

という文にすれば，描画の出力先は GPIB ケーブルで接続された，12 番というアドレスを持つプロッタに出力されることになる．"HPGL"（Hewlett Packard Graphic Language）は HP 社が開発したグラフィックス出力のための言語である．
30 行～ 80 行ではグラフを描くための，X と Y の値域などを任意の変数に代入している．
90 行ではパソコン画面などの描画デバイスの中で，グラフィックスを出力する範囲を指定している．すなわち，VIEWPORT を実行すると，その後の描画命令文による出力は，VIEWPORT で指定した範囲に限られてしまう．範囲の指定は，描画デバイスの全画面の Y 軸（縦）方向を 100 目盛の長さに取り，X 軸（横）方向はそれぞれのディスプレイに依存する長さにとったときの，目盛値で行う．この目盛のことを HT-BASIC では GDU（Graphic Display Unit）と呼んでいる．たとえばノートパソコンの全画面は横長の画面であるから，Y 軸方向が 100 であり，X 軸方向は一般に 100 より大きい値で指定される．それを確認するには，

```
RATIO
```

という命令を実行すればよい．＜RATIO＞は描画できる範囲の X 軸方法の長さと Y 軸方向の長さとの比（たとえば 1.4 程度の値）を返す．もし，ノートパソコンの画面の RATIO の値が 1.4 ならば描画できる範囲は Y 軸方向を 100 目盛，X 軸方向を 140 目盛とする範囲であり：

```
VIEWPORT 横方向の左限，横方向の右限，縦方向の下限，縦方向の上限
```

として指定する．したがって，90 行は画面全体の中で横方向の全目盛数（大体 140 ぐらい）のうちの 40 ～ 120 の範囲，縦方向の全目盛数 100 のうちの 20 ～ 90 の範囲とする長方形の範囲に描画せよ，という意味になる．前もってキーボードから RATIO 命令を実行して確認しておけばよい．プログラム中で RATIO を使用して，

```
Xscale=100*RATIO
VIEWPORT 0.3*Xscale,0.7*Xscale,20,80
```

とすれば，描画範囲は常に X 方向の 30% から 70% の範囲，Y 方向の 20 から 80 の範囲になる．
100 行では，VIEWPORT で指定した範囲に，描画で使用する座標値を定義している．この場合，30 ～ 80 行で定義した変数により，

```
WINDOW Xlow,Xhigh,Ylow,Yhigh
```

は，左端が −1，右端が 1，下端が −1，上端が 1 と定義されたことになる．
110 行では描画範囲をフレーム（線）で囲んでいる．
120 行は座標軸を描く命令である．パラメータの指定は，

```
AXES Xt, Yt, Ox, Oy, Cx, Cy, S
  ただし，Xt：X 軸方向の座標目盛の間隔，
          Yt：X 軸方向の座標目盛の間隔，
          Ox：Y 軸と交差する X 座標値，
          Oy：X 軸と交差する Y 座標値，
          Cx：X 軸の主要目盛の間隔，
          Cy：Y 軸の主要目盛の間隔，
          S ：VIEWPORT で使用した単位で表現した主要目盛マークの長さ，
```

となる．120 行は，

```
AXES Xlength/20,Ylength/20,0,0,5,5,2
```

となっているから，

```
＜Xlength/20,Ylength/20,＞で，X 軸方向には，X の表示域を 20 等分した目盛を，
                              Y 軸方向にも同様の目盛を描かせている．
＜0,0,＞だから，XY 軸は原点で交差する．
＜5,5,＞だから，XY 軸ともに 5 個おきに主要目盛が現れる．
＜2＞だから，2 という長さの主要目盛マークを描くことになる．
```

130 ～ 180 行は描画オブジェクトの名前をラベルとして描く作業である．前章までの例では文字や数値を出力するために PRINT 命令を使用した．しかし，PRINT はワープロで文字を出力するようなもので，罫線に沿った決まった場所にしか文字を出力できない．描画オブジェクトに合わせて任意の場所に文字・数字を出力するには＜ LABEL ＞命令を使用しなければならない．
130 行では 90 行の＜ VIEWPORT ＞命令で指定された描画範囲の制限を＜ CLIP OFF ＞命令で解除している．
140 行は文字列の代入 .
150 行ではペンの位置を指定した座標へ移動している．この場合 X 座標が 0，Y 座標が Yhigh となっている．Yhigh という座標は，100 行で指定したように描画範囲の上端にあたる．その周辺で文字を書いたら，VIEWPORT 命令で指定された範囲については文字を描けるが，その範囲を超えてしまうと描くことができない．そこで 130 行によって，一時的に描画範囲を解除したのである．
160 行では，150 行で移動したペン位置が，ラベルとして書く文字列のどの位置にあたるかを指定している．数字は 1 ～ 9 の値をとり，それぞれ次のような意味である．

```
3：(文字列の) 左上，6：(文字列の) 中上，9：(文字列の) 右上
2：(文字列の) 左中，5：(文字列の) 中中，8：(文字列の) 右中
1：(文字列の) 左下，4：(文字列の) 中下，7：(文字列の) 右下
```

したがって，160 行の＜ LORG 4 ＞というのは，ラベル全体の文字列の中央下端を，150 行で指定した座標値に一致させることになる．
170 行ではラベルを出力している．
180 行はラベルを書くために描画範囲の制限を解除していたので，再び＜ VIEWPORT ＞命令の範囲に制限するものである．
190 行は描画するためのペンの色を指定している．数字は 0 ～ 15 の値をとり，その色は：

```
0(black),1(white),2(red),3(yellow),4(green),5(cyan),
6(blue),7(magenta),8(black),9(olive green),10(aqua),
11(royal blue),12(maroon),13(brick red),14(orange),15(brown)
```

となっている．描画の際には色を確認したほうがよい．
200 ～ 220 行は PLOT 命令により折れ線グラフを描くループである．
210 行は指示された座標値 (X,X*X*X) まで線を描く命令である．
230 ～ 290 行は X 軸に目盛値を描くループである．
230 行では LABEL 命令で描く文字の大きさを GDU 単位（80 行の説明を参照）で指示したものである．
＜ CSIZE ＞命令は 2 個のパラメータを持ち，

```
CSIZE 文字の高さ , 文字の幅／文字の高さ
```

となっている．文字の高さを GDU で指示し，文字の高さに対する文字の幅の比を与えることができる．文字高のみを指示した場合には，デフォルトの文字幅の比 0.6 が使用される．
240 行は，150 行で説明したように，ペンの移動先の座標が文字列の左下にくるように指示している．
250 行ではペンの色を 2(red) に指示している．
260 ～ 290 行ではペンを 0.5 ステップで移動させながら X 軸に目盛値を描いている．
300 ～ 330 行ではペンを 0.5 ステップで移動させながら Y 軸に目盛値を描いている．320 行の書式付出力については，6 章 (4) 節，210 行の説明を参照されたい．
図 6 はグラフィック出力の例である．

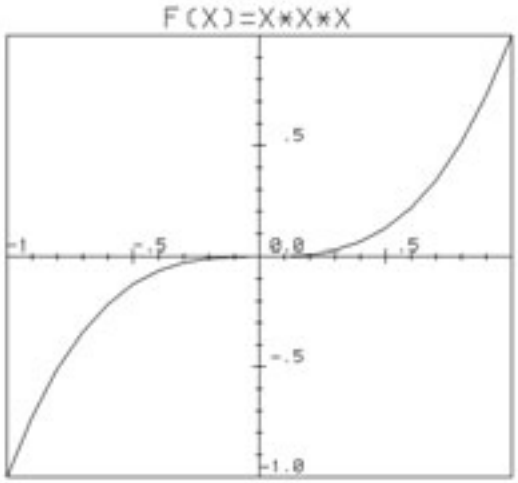

図 6　グラフィックスの出力の例
Fig. 6　A graphic output.

(2) オンライン入力データをグラフで表示する
InesDAQ i250 用のドライバをインストールして，以下のプログラムを入力する．このプログラムには測定されたデータをグラフ表示するプログラムが含まれている．

```
10   LOADSUB ALL FROM "c:\InesDAQ\htbasic\daqhtb.csb"
20   OPTION BASE 1
30   DIM A(1),B(1024) ! Data arrays
40   INTEGER Dummy, Hadc,Cadc
50   DIM L$[256]                                ! Label for configuring ADC card
60   CALL Daqhtb_ioctl(Dummy,"(fctn init)")
70   CALL Daqhtb_open(Hadc,"i250 ADC",0)
80   L$="(ioa(timeout 10000 fsmpl 100000 chan 1 iomode single trigmode off
 samples 1 range [-10 10]))"
90   PEN 3
100  AXES 5,5,0,50,2,2,2
110  PEN 1
120  MOVE 0,50
130  FOR I=1 TO 101
140     CALL Daqhtb_ioctl(Hadc,L$)
150     CALL Daqhtb_read(Cadc,Hadc,A(*))
160     DRAW I,A(1)*4+50
170     B(I)=A(1)
180     WAIT .1
190  NEXT I
200  FOR I=1 TO 101 STEP 10
210     PRINT I,B(I)
220  NEXT I
230  CALL Daqhtb_close(Hadc)
240  END
```

80 行ではデータ入力の動作を指示している．プログラム例では文字列が長いので 2 行に分けて表示してあるが，入力するときには途中に改行＜ ENTER> を入れてはいけない．
90 ～ 120 行は座標軸描画など，データを表示させるための準備．
130 ～ 190 行は InesDAQ i250 によるアナログデータの読み込みとグラフ表示．
200 ～ 220 行は読み込まれたデータの数値表示である．

7.2　グラフィックスの出力と保存

前節でディスプレイに出力したグラフィックス画像を保存するには，3 通りの方法がある．
①第一の方法は，GPIB インターフェイスを介して，プロッタを接続し，直接プロッタに描かせるものである．前節のプログラム例で 20 行の＜ PLOTTER IS CRT,"INTERNAL" ＞という命令文を

```
PLOTTER IS 712,"HPGL"
```

と書き直せば，GPIB で接続された，12 番のアドレスを持つプロッタに "HPGL" という命令コードで出力される．"HPGL" は HP 社がグラフィックス用に開発した言語で，使用するプロッタは "HPGL" 言語機能を持ったものに限られる．現状ではプロッタがあまり使われなくなったが，HP 社のプロッタはこの機能を持っていた．販売中の製品としては，グラフテック社の MYPLOT シリーズが "HPGL" 機能を持っている．
②第二の方法は汎用のプリンタに出力する方法である．この場合もプリンタは "HPGL" 言語機能を持ったものに限られる．
③第三の方法はディスプレイに出力された描画をビットマップ画像ファイルとして取り込む方法である．プロッタで出力する場合に比べ，ディスプレイ画像の質は落ちるが，画像ファイルとして保存してしまえば，WINDOWS 上ではコピーや書類への貼り付けなどの操作が簡単におこなえる．最も簡単な方法は：

```
＜Alt ＋ Print Screen＞
→ ＜→スタートメニュー→プログラム→アクセサリ→ペイント＞
→ ＜→新規作成→編集→貼り付け＞
```

＜Alt＞キーを押しながらキーボードの右上のほうに配置されている＜Print Screen＞キーを押すと，そのときのアクティブウィンドウ画面がコピーされる．次に，＜スタートメニュー → プログラム → アクセサリ → ペイント＞で WINDOWS 付属の＜ペイント＞アプリケーションを開き，＜ → 新規作成 → 編集 → 貼り付け＞を選ぶと，新規作成画面が自動的に調整され，そのときのアクティブウィンドウ画面が画像として表示される．この画像はファイルとして保存できるので，その後 jpg ファイルとして処理したり，ワード画面に貼り付けたり出来る．それ以外には Adobe 社の画像ソフトなどで＜ファイル → 新規作成＞で開いた白紙画面に貼りつけるとか，あるいは，ソースネクスト社のスグレモ 3 というソフトには撮画ツールがあって，同様の作業が行える．＜Print Screen＞ キーを起動するには，必ずしも＜Alt＞キーで行うとは限らないので機種ごとに試してみるとよい．プロッタやプリンタに出力しても，そのままではその画像を書類に貼り付けることはできないから，③の方法が便利と思われる．

7.3 HT-BASIC Plus

HT-BASIC には各種の GUI が用意されていて，プログラムで定義できる．もともとは HP-BASIC Plus という名前で追加された命令群だったが，HT-BASIC for WINDOWS Ver.9.2 には標準で含まれている．たとえば，プログラム実行中に，変数の状態をグラフなどで確認したいとき，キーボードから変数に値を与えたり，幾つかの処理の中から適当な処理を選びたいとき，エラー発生時の処理を指示したいときなどに利用できる．しかし，GUI の命令がなくても HT-BASIC 固有の命令で同様な作業を行えるので，プログラム実行上不都合はない．HP-BASIC Plus が出現する以前からのユーザで，HP-BASIC Plus を使用したことがない技術者は多い．以下はエラー発生時の作業を選ぶダイアログ（問い合わせ画面）を表示するプログラム例である．

```
10   MASS STORAGE IS "C:\Program Files\HTBwin92\"
20   LOAD BIN "BPLUS"            !HP-BASIC Plus 用のバイナリプログラム
30   DIM S1$(0:1)[10],P$[20]
40   INTEGER Btn
50   S1$(0)="Abort"
60   S1$(1)="Continue"
70   P$="Error is caused"
80   DIALOG "ERROR",P$,Btn;SET("DIALOG BUTTONS":S1$(*)),TIMEOUT 5
90   SELECT Btn
100  CASE =-1                    !Btn=-1 はタイムアウト
110     PRINT Btn
120     DISP "Timeout"
130  CASE =0
140     PRINT Btn
150     DISP S1$(Btn)
160  CASE =1
170     PRINT Btn
180     DISP S1$(Btn)
190  END SELECT
200  END
```

HP-BASIC Plus を使用する場合は，10 ～ 20 行で示したように，必ず

```
LOAD BIN "BPLUS"
```

を実行し，バイナリプログラムを読み込んでおかねばならない．
80 行で "ERROR" というダイアログを設定している．

```
DIALOG "ERROR",P$,Btn;SET ("DIALOG BUTTONS":S1$(*)),TIMEOUT 5
```

パラメータの中で，"ERROR" はダイアログの種類を示している．P$ はこのダイアログが起動されたときに示されるメッセージで，70 行で定義した文字変数で指示されている．Btn はどのボタンが選択されたかによって 0，1，2‥の整数が代入される．何も入力されないときのタイムアウト（制限時間）が指定されていれば，制限時間が終了した時点で -1 が代入される．SET の（ ）の中では表示するボタン名を文字列配列で指示している．TIMEOUT 5 はタイムアウト時間が 5 秒の意味である．

このプログラムを実行するとダイアログ画面の中に，"Abort" および "Continue" という 2 個のボタンが表示される．どちらかを選ぶと，左側のボタンなら 0，右側なら 1 の数字とボタンの文字が表示される．ボタンは 3 個以上定義してもよい．何も入力せず，5 秒経つと -1 および "Timeout" という文字が表示される．

HT-BASIC Plus では 10 種のダイアグラム（diagrams;問い合わせ画面）および, 30 種のウィゼット（widgets;メータ, メニュ, グラフなどの表示道具）が準備されている．HT-BASIC Plus で使用できる命令はヘルプファイルに詳述されていて，"plus examples" という名前のフォルダのなかに，プログラム例が収録されている．ただ，上記したように，ウィゼットの中には設定項目を多く含むものがあるのので，7 章 (1) 節で取り上げたグラフ表示のような，通常の HT-BASIC の範囲で描いたほうが，プログラムとしては簡単になる．

8.　変数の定義

①変数名のつけ方
　15文字までの英数字および＜ _ ＞，先頭はアルファベット大文字,
　文字列変数には名前の最後に＜$＞をつける.
②変数の種類
・数値変数　　：数値（例：Data，Name）
　実数精度：8バイト，±有効数字15桁±308乗,
　　上限と下限は命令＜MAXREAL＞，＜MINREAL＞で確認できる.
　整数精度：2バイト，－32768～＋32767
・文字列変数：1バイト/1文字，デフォルトで18文字長まで代入できる，任意の文字長はDIMで定義する.
　　例） Data$　　：デフォルトで18文字まで
・単純変数　：次元が定義されていない変数
・配列変数　：配列の次元を定義した変数
③変数の定義の仕方
・数値変数：
　実数：　REAL X,A(5)　：実数（REAL）の定義は省略できる.
　　例）INPUT X　　：Xは実数として定義される.
　　DIM A(5)　　：5個の実数値配列
　　DIM Data(10:14)　：添え字10 ~ 14 で呼び出される5個の実数値配列,
　整数：　INTEGER Y,B(5)　：整数単純変数と整数配列変数
　複素数は2個の実数（8バイト）の組（16バイト）として取り扱われる.
　　COMPLEX A,B　　：複素変数の宣言文
・文字列変数：
　　例）DIM Name$[20]　：20文字までの文字列単純変数.
　　INPUT B$　　：B$ は18文字までの文字列変数として定義される.
　　DIM Name$(5)　：5個の文字列配列，18文字まで
　　DIM Name$(10:14)[20]：添え字10～14で呼び出される20文字の配列.
④配列の定義
　次元数は最大6次元まで,
　各次元の要素数は最大32767個まで.
　配列の添え字：添え字の範囲を指定していない場合には,
　　DIM A(5)　　：5個の実数値配列
　　OPTION 0　を指定した場合には 0～4 で呼び出す（デフォルト）.
　　OPTION 1　を指定した場合には 1～5 で呼び出す.
⑤数値演算子と優先順位
　　()　：かっこ
　　FN　：関数
　　^　　：べき乗
　　* ，/ ，MOD ，DIV　：積，商，MOD（割り算の余り），DIV（割り算の商）
　　+ ，－　　：和，差
　　= ，＜ ，＞ ，＞= ，＜= ，＜＞ ：関係演算子
　　NOT　　：論理演算子

```
      AND          ：論理演算子
      OR , EXOR  ：論理演算子
+ , － , * , MOD , DIV は両方の演算対象が整数のときには，整数演算を行う．
実数値の関係演算は
      DROUND(A,12)=DROUND(B,12)   ：12 桁で四捨五入
      ABS(A － B)<=1E-10              ：差の絶対値が誤差範囲以下
```

9. あとがき

現在のノートパソコンは，**1**）電子文房具あるいは情報端末といった利用の仕方がほとんどである．本来の目的だった **2**）計算機能でさえ既成のプログラムに任せることが多くなってきている．本論で述べたような，**3**）計測･制御への利用というのは，これら，情報端末・計算機能とはまったく別の次元での計算機の利用方法である．情報端末としての利用に習熟している人が計算機能を利用しようとすると，利用するハードウェアとしてはコンピュータ本体のみであっても概念や用語の違いなどの壁にぶつかってしまう．さらに，情報端末や計算機能に習熟している人であっても計測・制御に利用しようとすると，コンピュータ本体の操作だけではなく，センサ，測定器，アクチュエータなどのハードウェアがケーブルを通して接続されるので，それらの使用法に習熟していなければ大きな壁になる．市販の AD コンバータボードが，このような中身の面倒くささには一切触れなくても決められたとおりに接続すれば，専用のソフトでデータ収録が出来るように設計されているのにも理由がある．このような傾向は今後ますます盛んになっていくと思われる．しかし，研究者・技術者・学生のレベルでは自分の発想のままに自由にデータを得て処理が出来るというシステムが不可欠のものである．与えられたシステムだけでは自由な発想は生かせないし，外注したとしても，受注した技術者にその研究分野のノウハウがなければユーザが希望するシステムは作れない．

本報告で述べた HT-BASIC はこれら 3 種の利用の方法を相互に結びつける all-in-one タイプの言語システムである．すなわち，①計測システムを組上げてデータをファイルに収録する．②収録されたデータをファイルから呼び出し，HT-BASIC あるいは市販のソフトウェアを利用して数値計算やグラフ化などの処理をおこなう．③結果の数値をそのまま利用するか，あるいはグラフィックス画面を画像ファイルとして保存し，プレゼンテーションに利用する．このような一連の作業を，鉛筆を持って裏紙に書き付けるような感覚で，命令ラインを書き込めるのである．これらの作業はもちろん既存のプログラム言語でもできる．しかし，がんばれば「作れる可能性がある」とか，特殊な訓練を受けた人だけが「作れる」ということと，一般の研究者・技術者・学生が実際に出来るということとはまったく意味が違う．普段は現地調査・実験装置の組み立て・文献の調査などに追われていて，プログラムや計測システムの専門家でもない研究者や学生が，急に利用したい場合にも普通の論理をたどっていくだけで，無理なく使える言語システムは HT-BASIC 以外には見当たらない．

本報告を執筆する過程で多くの大学・企業・研究所の方々と過去・現在・未来の計測手法について議論をした．いろいろな議論があったが，共通していることは，WINDOWS の登場以来コンピュータの概念が変わってしまって，それぞれが自分独自の計測手法を探し，創り出しておられるということだった．もちろん HP-BASIC のこともご存知の世代である．しかし，圧倒的に C 言語が普及している時代に BASIC とは？という意見も多かった．過去の歴史を知り経験がある年配の方の中には，中身はわからずとも LabVIEW で十分データが取れ，処理ソフトも付いていて便利だとおっしゃる方もいた．他方，若い人たちの間では，LabVIEW 等の既成のソフトばかりの環境の中で育ってきて，たとえ C 言語であっても，自分で実際にコマンドラインを書き込んでデータを収録した経験はないのです，とおっしゃる方もいた．学生諸君の世代に共通するのは，ソフトつきの市販の計測ボードでデータを収録したら，その処理や表示は，インターネットなどで好みのソフトを探し出して利用するというものである．3. 1 節でも述べたように，試行錯誤の中でパラメータを変更して RUN キーを押せば即実行するというシステムでなければ，実験や研究現場での自由な発想を活かすことは出来ない．1977 年 8 月 23 日早朝，Dr. Paul B. MacCready 等は人力飛行機 Gossamer Condor による人類初の 8 の字飛行を成功させた．翼の設計は HP 社のデスクトップコンピュータ HP9820A を駆使した Dr. Peter Lissaman によって行われた．彼は古い理論と（彼等にとって）古い計算機だったにもかかわらず成功した理由として，改造しやすい機体を設計したことと HP9820A によって思いついたヒントを直ちに確かめることが出来たことをあげている [4]．この飛行実験の記録映像は 1978 年度のドキュメンタリー部門のアカデミー賞を受けている．彼等は 2 年後の 1979 年に人力飛行機 Gossamer Albatross によって人類初の英仏海峡横断も成功させている．HT-BASIC は計測現場からの要望に十分に答えられるだけではなく，コンパイラを持ち，コンパイルされたサブルーチンと現在のパソコンの性能向上もあって，数値計算の面でもかなりのことが出来るはずである．また大学の研究室においては先輩のプログラムを改良して活かすことも出来る．

本報告では，このままでは，少なくとも日本国内では確実に忘れ去られてしまうと思われる HT-BASIC という言語のアウトラインを示した．実際には数値計算や論理判断以外の，GPIB システム監視に関する多くの命令，入出力制御命令，エラーメッセージなどがあって，HT-BASIC をかなり使った人でも，その全貌がわかる人は少ないだろう．しかし，通常の使い方であれば，本文中の例が手がかりになるはずである．

本文中では InesDAQ i250 について詳しく動作を説明したが，工夫すれば，GPIB を使わなくてもこのカード一枚でかなりの程度のデータ収録プログラムが作成できるからである．

前書きにも述べたとおり，これらすべての解説書が現在では，英文のオンラインマニュアルとしてしか提供されていない．さらに，これらのマニュアルの中にも，全体像を解説したものはないのである．本報告が幾らかでもその役を果たすことを期待する．

（原稿受理：2004 年 10 月 28 日）

参考文献

1）TransEra（2002）: TransEra HTBasic Installing and Using Manual.
2）TransEra（2002）: TransEra HTBasic のオンラインマニュアル.
3）Ines（2001）: InesDAQ のオンラインマニュアル.
4）Blair, John（1979）: Keyboard, Jul-Aug, pp.14/16, Hewlett-Packard.
5）CQ 出版（1996）: パソコンによる計測・制御入門．トラ技スペシャル，No.**53**.
6）CQ 出版（1999）: Windows PC による計測・制御入門．トラ技スペシャル，No.**68**.
7）藤原博文（1994）: C プログラミング専門課程，技術評論社.
8）岩波書店（1981）: 科学特集号，**51**-10.

要　旨

HT-BASIC を使用したノートパソコンによる計測・制御システムのプログラミングについて詳述している．現在，パソコンを利用した計測・制御システムとしては，市販のアナログ信号変換ボードに付属しているソフトを使用するか，GUI を多用した汎用の計測・制御ソフトを使用するかに分かれる．前者はユーザ独自の変更のための自由度が少なく，後者は簡便さを狙ったあまり，GUI の便利さに隠れて実際の計測の流れが見えないという性質がある．また C 言語を基礎におくシステムでは，しばらくたつと設計者でもプログラムの流れが見えなくなるという本質的な欠点がある．HT-BASIC は前身を HP 社の HP-BASIC に置く計測・制御システム構築のためのソフトウェアで，作業の流れが大変にわかりやすい言語構造をもっている．とくに最近のバージョンは WINDOWS に対応して使いよいソフトに改良されている．実験装置組み立て・実験の遂行とデータ整理・現地調査・文献調査など，日ごろプログラムだけに集中できない学生・研究者・技術者にとって，HT-BASIC は，ノート PC レベルで計測・制御システムを自由に組み立てるための必需品といえる．

キーワード：BASIC，ノート PC，GPIB，ユーザ指向，計測