노트 PC 에 의한 계측과 제어
－HP(HP)-BASIC 에의 권유－

富永雅樹*(토미나가 마사키:TOMINAGA, Masaki)
Advanced Technology Research Group
National Research Institute 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 from 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 impassible 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 I going on in the systems or the programs. In the present paper HT-BASIC is introduced for the use for note PC. The HP-BASIC was a well acknowledged software to control measurement systems by PC. Researcher 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.

Keywords; Note PC, BASIC、Field instrumentation、Instrumentation、User oriented instrumentation system

1. 서론

인터넷에 의한 정보검색이나 전자메일의 교환.문서작성을 위한 만능 사무 단말로서 컴퓨터는 개인이 소유하게 되었다.기능도 향상이 돼 Cray 사가 1976 년에 발표한 27 억엔이나 하는 슈퍼 컴퓨터 Cray1 의 속도가 160MFLOPS, 메모리가 8 메가바이트였던 것에 비교하면 현재 보통의 PC 에서 클록이 2 기가헬츠, 메모리가 500 메가바이트이니까, 당시의 슈퍼 컴퓨터이상의 도구가 책상 위에 있는 시대가 되었다.계측방면에서도 이용되고 있고 범용의 오퍼레이션 시스템 WIND OWS 상에서 동작을 하는 계측 소프트웨어(VEE, LabView 등) 도 보급이 되어 있다. 그렇지만, 이러한 소프트는 WINDOWS 특유의 GUI(Graphic User Interface)를 이용해서 만들어졌기 때문에 간단은 하지만 조작내용이 잘 보이지 않는다는 성질을 갖고 있다. 연구, 개발의 현장에서는 프로그램중에 명시적으로 무엇을 행할 것인지에 대해 작업을 지시해, 계측시스템의 제어를 해서 데이터를 수집하지 않으면, 그 후의 처리를 할 수가 없기 때문에 시판되고 있는 소프트웨어로는

뭔가 모자라는 느낌을 피할 수가 없다(Reference 5,6). 한편으로, 이식성이 높기 때문에 이용이 되고 있는 C 언어도 조작자, 계측된 데이터,환경조건,계측시스템의 제어 등의 사이에서 상호작용을 시킨다는 인터랙티브 (interactive)한 시스템을 만들기에는 적합하지 않다.작성한 본인이외에는 프로그램의 흐름을 이해하기가 어려우며 수정을 할 때마다 코멘트를 써 놓지 않으면 몇 년이 지나면 작성을 한 본인마저도 잘 모르게 된다 같은 작업에도 여러 종류의 수법이 있어서 이해하기가 어려운 점, 작업관수 로서 표현을 하지 않으면 안되기 때문에, 작성자에 따라 알고리즘이 제각기 틀린 점,시간이 지나면 세세한 지정을 무엇을 위해 해놓았는지를 잊어버리는 것 등이 그 이유이다. 신참의 연구원이나 학생 등이 많은 에너지를 쏟아 부어 넣고서도 자기 몸의 일부분처럼 작동을 하는 계측시스템을 가질 수 없다는 상황을 보고 있으면, 그래피컬한 프로그램의 전성시대에는 그림자가 흐려져 버린 HT(HP)-BASIC 이라는 언어를 재평가하지 않을 수가 없다. 이 리포트는 계측시스템을 작성할 경우, 상기와 같은 장벽이나 문제점에 직면한 경험이 있는 모든 학생 및 기술자 여러분을 위하여 당 연구소에서 경험을 토대로 HT(HP)-BASIC 의 Outline 을 소개하는 것이다 .

(그리고 여기에 소개되어 있는 제품명, 회사명은 각 사의 등록상표 또는 상표이다)

2. HT(HP)-BASIC 의 역사

계측시스템의 과거 30 년의 역사를 보면, 1970 대초반에는 계측의 시스템화는 원자력 등의 대규모의 프로젝트에서 사용되고 있던 CAMAC 등을 제외하면 아직 꿈과 같은 일이었다. 애널로그가 개체로 사용돼 펜으로 쓰는 기록계나, 데이터 레코드로 시간의 경과에 따른 변화를 기록할 수 있으면 괜찮은 편이었다.

그러나 신호처리의 분야에서는, 다이나믹한 신호이론의 연구에서 목적으로 하는 정보를 얻기 위해서는, 측정된 신호 하나 하나의 정밀도를 높이지 않으면 안되지만, 그러한 신호가 측정된 시각의 정밀도도 중요했다.복수의 계측으로 구성된 임의의 계측시스템을 실현시키기 위해서는 측정기의 디지털화가 필요했지만 아직 일반적이 아니었다. 그 후, 집적회로기술의 발전으로 연산증폭기(Operation Amplifier) 와 Texas instruments 사의 논리 IC 가 보급이 되기 시작하면서부터 계측기의 디지털화가 단숨에 진행이 되었다.

제조업이나 프랜트 현장에서는 [계측 장비] 라고 불리고 있는 것처럼 계측은 계측 단독으로 존재하고 있는 것은 아니다. 반드시 측정된 정보를 바탕으로 해서 프랜트나 제조 라인 등의 제어가 동반되는 것이다. 연구에 있어서도 측정된 데이터를 기준으로 실험조건이나 입력신호를 변경하는 등의 조작이 이루어진다 그 때까지는 애널로그 신호 기준으로 스위치나 변의 개폐를 위한 액츄에터(actuator)를 구동시키기 위해서는 개별적으로 고안된 기구가 필요했었지만, 그러한 사용자의 요망에 맞는 범용성이 있는 시스템을 만들기 위해서는 컴퓨터제어에 의거한 계측기 시스템의 실현이기대되고 있었다. Hewlett-Packard(HP)사가 사내 규격으로 계측기 시스템의 디지털 제어방식(HPIB, Hewlett-Packard Interface Bus)을 실현 한 것은 1973 년이었다. 이것은 계측기 회사로서의 노하우와 탁상형 계산기(desktop computer)의 기술을 갖고 있는 HP 사가 아니면 이룰 수가 없는 쾌거였다. HP 사는 자기회사의 규격을 국제규격으로서 제안해 1975 년에 IEC 규격으로서 승인이 되었다. 그 이후로 규격의 향상도 포함해 공식적으로는 GPIB(General

Purpose Interface Bus, HP 사 내부에서는 HPIB 로 불리고 있다)로서 보급이 되고 있다.

당시의 HP 사의 사내 규격의 단계에서도, 프린터 및 데이터 보존용의 테이프 커트리지 드라이브를 내장한 탁상형 계산기(퍼스널 컴퓨터라고 불리고 있었다)를 중심으로 디지털 전압계(A/D 컨버터), 스캐너 (릴레이에 의한 스위치 회로,현재의 화상입력기기와는 다른,) 디지털시계, D/A 컨버터. 신디사이저(신호발생), 카운터(주파수.시간간격 측정) 등이 갖춰져 있었다. 이 정도 있으면 일정의 계측,제어시스템을 구성하는 것이 가능하게 되어 있었다.

1977 년에 HP 사는 HP9825A 라는 역사에 남는 탁상계산기를 발표하였다.이것은 GPIB 가 국제규격이 된 것을 받아 들여 그 완전한 제어를 실행하기 위해서 설계된 것으로, 그 이후 시스템의 중심이 되는 계산기는 컨트롤러라고 불리게 되었다. 디스플레이는 32 문자표시의 단 한 줄 뿐으로 250byte 의 테이프에 의한 데이터 커트리지(data cartridge), 16 자릿수의 서멀 프린터(thermal printer), 디스크 등의 불휘발성의 메모리는 없었으며 표준장치로서의 RAM 은 6.8k 바이트라는 지금으로서는 상상도 할 수 없는 작은 용량밖에 없는 HP9825A 였었지만 뛰어난 오퍼레이팅 시스템과 HPL(Hewlett-Packard Language)이라는 독특한 언어의 사용으로 고도의 계측, 제어시스템을 편성할 수가 있었다.

그 후, 현재에는 당연하게 되었지만 CRT 디스플레이를 내장한 탁상계산기와 함께 HP-BASIC 언어 (Rocky Mountain Basic 이라고 불리고 있다)가 등장했다. 이 언어는 그 후 HP 사의 계측, 제어용 계산기(컨트롤러) 의 핵심이 된 언어로, 외부기기나 데이터의 입출력을 제어하기 위한 명령 뿐만이 아니라 수치연산기능, 그래픽스 출력기능, 파일의 작성이나 데이터의 전송기능 등을 포함한 언어이다 이 무렵부터 LA(Laboratory Automation: 실험실의 자동화) 또는 FA(Factory Automation: 공장의 자동화)라는 개념이 침투하게 되었다. 그 후, GUI 의 선구라고 할 수 있는 HP-BASIC Plus 가 추가돼 포괄적인 언어가 되었다.

1986 년에 Toshiba USA 가 운반이 가능한 랩톱 컴퓨터(lap-top computer) T-3100 을 발매했다. 그에 대한 충격은 대단히 컸으며 그 후의 퍼스널 컴퓨터의 형상과 흐름을 결정지었다. 그것을 받아 들여 HP-BASIC 의 사용자는 운반이 가능한 컨트롤러의 출현을 기다리게 되었다. 그러한 컨트롤러가 있다면 실내에서 뿐만이 아니라 계측기를 조립할 수가 있기 때문이다. IBM PC 가 세계표준으로서 보급됨과 동시에 그것을 뒷받침하는 Intel 사의 CPU 칩도 표준이 되었다. Motorola 사도 CPU 칩을 생산하고 있었지만 Intel 사와는 다른 68 계라고 불리는 아키텍쳐(architecture)였기 때문에 장래의 판매 확대를 기대할 수가 없어 생산을 중지했다. 그에 따라 Motorola사의 CPU를 채용하고 있던 HP 사는 1990년대 중반에 컨트롤러의 생산중지를 발표했다. 이 뉴스에 전 세계의 사용자(유저)는 깜짝 놀라 현행의 계측, 제어시스템의 보존용품으로서 HP를 대량으로 사서 재어두는 현상이 일어났다. 그러나 사용자들의 생산을 계속 해 달라는 요청을 받아들여 생산중지가 몇 년 뒤로 연기되기도 했다. 결국 HP-BASIC 대응의 오퍼레이팅 시스템을 갖고 있는 HP9000/300 시리즈의 PC 는 1996 년 10 월에 공식으로 판매중지가 되었다. HP 사에 의한 HP-BASIC 은 버전 6.2 로 종료가 되었다. 그러나 판매중지가 되어 보수도 받을 수 없게 된 이상, 현재 보유하고 있는 컨트롤러가 고장이 나면 어떻게 할 방법이 없었다. 이에 IBM PC 의 표준 OS 인 WINDOWS 상에서 움직이는 HP-BASIC 의 출현에 대한 기대가 높아졌다. TransEra 사는 1988 년부터 IBM PC 상에서 움직이는 HP-BASIC 을 Rocky Mountain Basic 이라는 이름으로

발매하고 있었다. HP 사는 1996 년 초반부터 2000 년 11 월에 걸쳐 HP-BASIC for WINDOWS 라는 이름으로 TransEra 사의 OEM 제품을 발매하고 있었다.
이것으로 표면적으로는 HP-BASIC 의 사용자는 컨트롤러의 고장을 걱정하지 않아도 되게 되었지만, 상황은 곤란해졌다. 지금까지의 계측의 형태가 보이지 않게 된 것이 그 원인이었다. 그 원인의 몇 갠가를 예로 들면 우선 사용자의 종래의 HP-BASIC 에 대한 절대적인 신뢰가 그 이유중의 하나였다. 당시는 각각의 회사가 제각기 다른 CPU 와 매뉴얼을 판매하고 있어서 컴퓨터를 직접 만드는 것이 유행하던 시절이었다. Motorola 사의 68 계의 CPU 의 아키텍쳐가 Intel 과는 다르다는 것은 당시의 기술개발자에 있어서는 지극히 당연한 지식이었다. 시스템이 다른 이상, 낡은 오래된 프로그램을 완전히 이행시키는 것은 불가능하지만 HP-BASIC for WINDOWS 의 영문 매뉴얼에는 신구의 대비가 되어 있어서 안 되는 일이 자세하게 적혀 있었다. 기술개발자로서는 당연한 일을 했다고 할 수 있지만 사용자들은 그렇다면 이것은 사용을 할 수가 없다고 생각을 하게 되어 버렸다. 또한 HP 사에서는 VEE 의 개발에 힘을 쏟아, TransEra 사의 HP-BASIC for WINDOWS 는 판매는 하기는 했지만 적극적인 백업을 하지 않았다. 특히, 일본 HP 사의 일본어 매뉴얼은 유료의 강습회 참가자를 전제로 한 것으로서 관심이 있는 연구자, 학생도 매뉴얼을 손에 넣을 수가 없었다.
또한 그것을 읽어 보아도 통상의 프로그램의 작성법이 설명이 되어 있을 뿐 계측, 제어를 간단하게 할 수 있다는 HP-BASIC 의 뛰어난 특징이 이해하기 쉽게 적혀 있지 않았다. 그 위에 WINDOWS 의 등장도 영향이 있었다. 초기단계에, 어떤 일이라도 가능하다는 꿈같은 시스템인 것처럼 선전을 했기 때문에 이식성이 높은 C 에서 프로그램을 작성해보려고 하는 사람마저 나타났다. 그러기 위해서 공부를 하고 또 공부를 위해서 매뉴얼을 찾는 등 시간을 낭비한 베테랑 기술자들도 많이 있었으리라 생각한다. C 에 관해서 많은 종류의 해설서가 나와 있다는 것 자체가 C 가 이해하기 어렵다는 것을 증명하고 있다. 계측 ,제어시스템을 백지의 상태에서부터 조립하기 위해서는 C 언어는 적합하지 않다. WINDOWS 에서는 시각에 호소하는 비주얼한 다큐멘트가 간단하게 작성할 수 있게 되어 HP 사의 VEE 라든지 National Instrument(NI)사의 LabView 라든지 계측, 제어의 분야에서도 비주얼한 [프로그램] 을 작성할 수 있다는 소프트가 등장했다.
NI 사는 장래의 사용자를 예상해 이러한 소프트의 강습회를 열거나 각국의 언어로 된 매뉴얼을 내놓고 있지만, HP 사는 그렇게 적극적이라고는 할 수가 없다. 한 편, PC 가 보급되면서 두 회사 이외에도 여러 회사가 제각기 다른 계측, 제어용 인터페이스 보드를 시판하게 되었다. 이러한 보드에는 전용의 소프트웨어가 부속이 되어 있다. 처음으로 PC 에서 계측, 제어를 실행하는 학생들은 물론 다른 시스템을 사용해 왔던 사람들도 매뉴얼을 읽으면 데이터 수록이 가능하기 때문에 보급이 되어 있다. 그러나 한편으로는, 이러한 사용자들은 매뉴얼에 써 있는 대로 아이콘을 클릭해 데이터는 수집하지만 도대체 어떻게 해서 데이터가 수집되는지에 관해서는 알 수가 없다라는 불만과 불안도 갖고 있다. 이처럼 데이터는 데이터로서 따로, 그 후의 계산이나 그래프화 등의 처리는 또 따로 전용의 소프트로 행한다는 것이 현재의 상황이다.
이 리포트에서 기술하려고 하는 HP-BASIC 은 상술 한 바와 같이 기능이 전부 포함 된 올인원(all in one)소프트이다. TransEra 사는 그 후, HP-BASIC for WINDOWS 라는 이름을 계승해 편집화면에서의 사용을 간편하게 하는 등의 개량을 거듭해 현재에 이르고 있다.

HPIB 의 등장 이후의 계측시스템의 흐름을 봐 왔지만, 계측기의 제어라는 것은 비록 형태는 틀리지만 오늘날에도 역시 GPIB 프로트콜(protocol)이 기본이며 각 회사가 내 놓고 있는 GPIB 나 GPIO(General Purpose Input and Output, 범용의 디지털신호 I/O 버스)의 PC 카드는 HT(HP)-BASIC 로 제어를 할 수가 있는 것이다. 현재 40 대후반에서 50 대의 기술자,연구자들 사이에서 압도적으로 지지를 받고 있어 세계표준이라고 할 수 있는 HT(HP)-BASIC 이 현재의 학생들 사이에는 그다지 보급이 되지 않고 있는 이유는 지명도가 낮기 때문이다. 당시 HP 사의 측정기 및 컨트롤러(계산기)는 가격이 비싸서 대학에서는 좀처럼 보급이 되질 않았다. 일본어 매뉴얼도 있었기 때문에 자금이 풍족한 공공연구소나 민간기업에서는 보급이 되었지만 1 달러에 300 엔 전후로, 그 위에 환율의 변동이 심한 시대에는 대학에서 구입을 검토한 다는 것 자체가 매우 어려웠다. 그 결과, 대학의 연구실에 있어서 HP-BASIC 은 보급이 되질 않고 학생들 사이에 소프트가 전승되는 일도 없었다. 현재의 상태는 방대한 영문 매뉴얼(그것도 온라인 매뉴얼) 뿐으로 전체상도 잘 모르는 시스템이 지금부터 학생들에게 보급이 되어 간다고는 생각할 수도 없다. 그러나. 도입 가격이 내려간 지금이야말로 노트 PC 로 자유롭게 계측, 제어시스템을 짤 수 있는 시대가 되었다고 할 수 있다. 이러한 현재의 상태를 참작한 다음, 이 리포트를 한 번 읽어 보면 HT(HP)-BASIC 의 아우트라인을 파악하게 되고 그 가능성에 대해 소개를 하는 것이 이 보고서의 목적이라 말 할 수 있다.

국제 방재 과학기술연구센터(현 독립행정법인 방재과학기술연구소)의 대형 강우 실험시설이 완성된 것이 1974 년 3 월로, 계측 설비는 아직 완성이 되지 않은 상태였다. 다종 다양한 실험을 하기 위해서는 자유롭게 구성의 변경이 가능한 컴퓨터를 중심으로 한 계측, 제어시스템이 최적이라고 여겨졌지만, 실제의 예가 없었다. 때마침 그 무렵. HP 사가 사내 규격으로서 HPIB 를 제안하고 있어서 조사를 해 보니 일정의 시스템을 구성할 수가 있다는 것이 판명이 되었다. 아직 그 단계에서는 HPIB 는 공적인 규격은 아니었지만, HP 사가 발매하고 있는 (전술한) 제품만으로도 닫혀진 계측시스템을 짤 수가 있었기 때문에 HP 사의 시스템을 도입했다. 그 후에 국제규격(GPIB)으로서, 그 위에 또한 디지털기술의 진보와 함께 GPIB 커넥터를 장비하지 않은 계측기는 없을 정도로 보급이 되고, 컨트롤러도 각사에서 판매를 하기에 이르렀다. 자유롭게 계측,제어 시스템을 조립해 실험을 한다는 입장에서는, HT(HP)-BASIC 으로 GPIB 를 제어할 때의 우위성은 변함이 없다. HP-BASIC 을 계승한 TransEra 사는 HT-BASIC 이라는 이름으로 개량을 거 듭해, HP 사의 최종 버전에 비교해 그 뛰어난 제어기능은 남긴 채로 편집기능, 수학관수, 서브 루틴(sub-ro utine), 비주얼한 디스플레이 기능 등을 확장, 통합하고 있다. 현재는 시대의 흐름에 맞춰 WINDOWS 상에서의 GUI(Graphic Interface)를 이용한 사용하기에 편리한 프로그램이 되었다.

3. HT-BASIC 의 outline (Reference 1, 2)

원래 HP 사에서 개발된 HP-BASIC 을 지금은 TransEra 사에서 계승 받아 HT-BASIC 이라는 이름으로 개발을 거듭하고 있다. 현재 시판이 되고 있는 것은 HT-BASIC 9.x 버전이다. 이 장의 이후에서는 HT-BASIC 이라는 명칭으로 그 특징을 전체적으로 살펴보기로 한다.

HT-BASIC GPIB bus line 에 연결되는 외부 계측 등의 제어와 임의의 디지털신호의 입출력제어, 컴퓨터의 CRT(액정표시화면도 포함함,이하 같음) 출력화면의 제어, 디스크상의 파일의 작성과 데이터의 입출력 제어 및 일반의 수치계산을 포함한 프로그래밍 언어이다. 그 전체 구성은 **그림**

1 과 같다. 데스크탑 컴퓨터에 관해서는, GPIB 보드 및 범용의 비트 (bit)입출력을 행하는 GPIO 보드가 상품화 되어 있다. 노트 PC 에 관해서는,GPIB 용 및 비트 입출력용의 PC(PCMCIA)카드가 시판이 되고 있다.

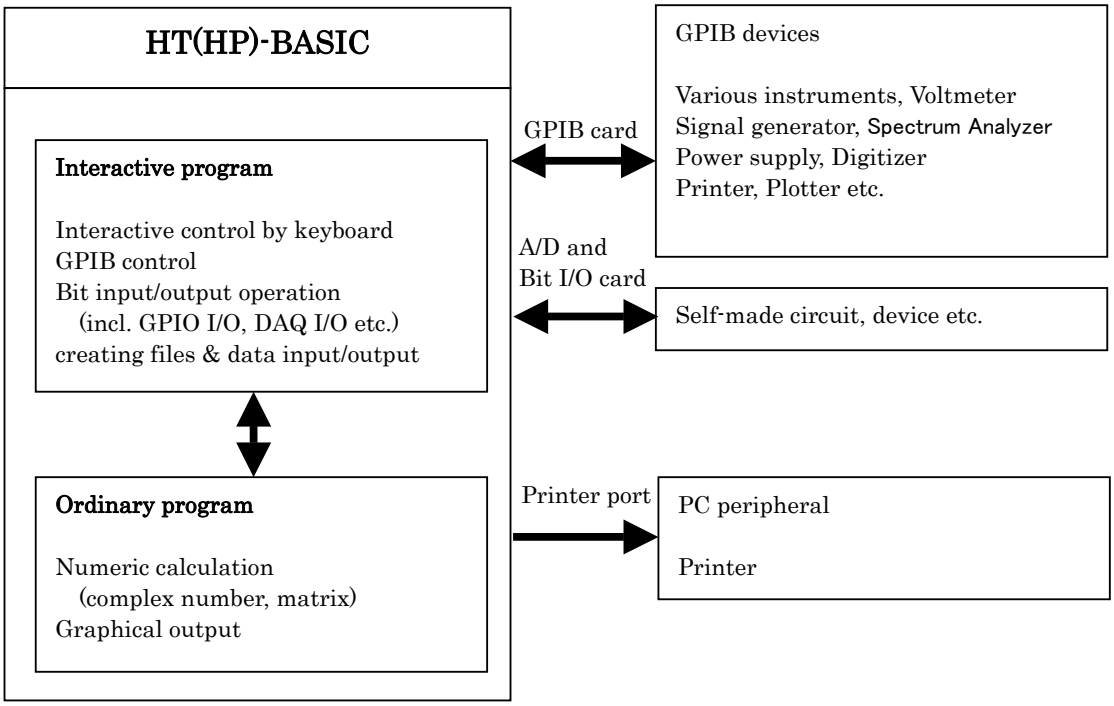


그림 1 HT(HP)BASIC and its environment

3-1. HT-BASIC 의 특징

(1) 인터프리터(interpreter) 언어

HT-BASIC 은 기본적으로는 프로그램의 명령 라인을 한 줄씩 읽은 다음에, 해석을 하는 인터프리터(interpreter)언어이다. C 또는 Fortran 과 같은 컴파일러(compiler)언어는 아니기 때문에 대규모의 수치 계산 등을 하기 위해서는 적합하지 않다.

그러나 극단적인 고속화를 요구하지 않는 수치계산, 조작자와의 대화형 프로그램/버스라인으로 결합된 컴퓨터 주변기기(프린터, 화상 입력용 스캐너, 하드디스크 등)이외의 기기(각종의 측정기기, 프로터 (plotter) , 디지타이저 (digitizer) , 디지털신호 입출력, 또는 자기가 직접 만든 릴레이회로 등)는 각각 제 각기의 속도로 움직이고 있으므로, 각 기기의 응답을 기다려 다음의 작업을 실행하는 경우에는 인터프리터 방식의 언어가 편리하다.

그 위에 인터프리터 언어의 최대의 특징이라 할 수 있는 것은, 그 이름의 유래이기도 한 RUN 키를 누르면 즉시 실행을 한다는 성격이다. 실험실에서 프로그램을 개발할 경우에는 우선 자기가 하고 싶은 처리의 커다란 틀을 정한 다음 골격이 되는 프로그램을 작성한다. 그 후에 여러 가지 기능을 추가해서 사용하기에 편리한 프로그램으로 완성을 해 나가는 것이 보통의 방식이다. 그 때에 약간의 변경이 있어도 파일의 보존→컨파일→실행을 반복하지 않으면 안 되는 C 언어 등에서는 넘쳐나는 아이디어의 확인에 시간이 걸릴 뿐이다.(후기 참조) 또한 실험이나

계측의 현장에서는 이전 했던 실험의 프로그램을 원래의 상태로 돌아가서 개량을 거듭하는 경우가 많이 있다. 그런 경우에도 C 언어라면 변수 뿐만이 아니라 프로그램의 흐름에 세심한 주의를 하지 않으면 안 된다. 그러나 BASIC 언어라면 줄(행)을 따라 가는 것으로 프로그램의 흐름을 확인하는 것은 간단하다. 바탕이 되는 프로그램을 타인이 작성했을 경우, C 언어라면 개편은 불가능에 가깝지만, BASIC 언어라면 개편이 가능하며 선배의 연구를 이어 받아 연구실의 자산으로서의 프로그램의 가치도 높아질 것이다.

바꾸어서 생각하면 이러한 특징은 특별히 일일이 열거할 필요도 없다. 생각한대로의 작업을 행할 수 있다는 것이 프로그램의 사명이다. 그러나 그 자체가 보이지 않게 되었다는 것이 이 리포트를 집필하게 된 동기이기도 하다.

(2) 컴파일러 기능

HT-BASIC 은 기본적으로 인터프리터 언어이지만, 수치계산 등을 할 때에는 부속의 컴파일러 기능을 이용할 수가 있다.

(3) BASIC 언어

이름처럼 BASIC 언어이니까 BASIC 언어에 공통된 프로그램 구조를 갖고 있어 논리가 이해하기 쉽다. 작성자 본인이나 타인이 몇 년 후에 읽어 보아도 이해할 수가 있고 또한 개량도 간단하다. 프로그램 언어는 언어인 이상 이해하기 쉬운 표기가 바람직하다. C 언어의 경우, 베테랑의 프로그램 개발자라고 하여도 초보적인 미스를 하기가 쉽다는 자체[7]가 그 언어가 갖고 있는 본질적인 결함이라고 할 수 있다.

(4) 서브루틴(subroutine)

보통의 서브루틴뿐만이 아니라 컴파일(compile) 된 수학 함수나 서브루틴의 라이브러리가 있어 프로그램 중에 자유롭게 불러 내어 사용을 할 수 있는 HT-BASIC 에서는, 수학 함수에 관해서는 거의 모두가 CSUB 화 되어 있지만 C 언어 등을 이용해서 CSUB 를 직접 만들 수도 있다.

(5) 파일의 입출력

프로그램 중에 임의의 이름의 파일을 디스크상에서 작성해, 수록데이터 등을 작성하는 일이 가능하다. 장기에 걸친 계측을 실시할 경우 등에 위력을 발휘한다.

(6) GPIB

HT-BASIC 은 단독으로도 강력한 언어이지만 GPIB 를 거쳐 계측기기 등의 제어를 실행할 때에 그 진가가 발휘된다. 노트 PC 의 경우에는 PC 카드의 슬롯에 GPIB 카드를 집어 넣고, 다른 한 쪽의 GPIB 전용의 커넥터를 계측기의 GPIB 소켓에 접속하면 된다. 아니면 USB 케이블의 다른 한 쪽이 GPIB 의 커넥터로 되어 있는 것도 있다.

(7) 비트(bit) 입출력(GPIO)

비트 I/O 제어도 GPIB 와 마찬가지로 PC 카드 슬롯에 비트 I/O 카드를 집어 넣고, 다른 한 쪽의 케이블은 직접 만든 디지털 신호변환 회로에 접속하여 사용한다. GPIO 란 HP 사가 당초 발매한, 파라렐(parallel) 에서 독립한 각각 16 비트 씩의 입력라인과 출력라인을 제어할 수 는 I/O 기기의 명칭이다. 현재 GPIO 는 데스크탑용의 PCI 보드밖에 시판이 되고 있지 않지만 비트 I/O 를 제어할 수 있는 PC 카드로서는, NI 사나 ines 사가 DAQ (Data Acquisition) 라고 불리는 카드를 판매하고 있다.(5 장에서 자세하게 설명) 이 중에는 A/D 컨버터를 내장하고 있어 단독으로도 전압 등의

측정을 할 수 있음과 동시에 8~24 비트의 입출력을 제어를 할 수 있는 카드도 있다. AD 컨버터를 포함한 PC 카드가 있으면 그것만으로 전압신호로 변환시킬 수 있는 신호의 계측시스템을 만드는 일이 가능하다.

(8) HT-BASIC Plus

HT-BASIC Plus 라는 확장언어를 이용해서 계측데이터를 미터나 그래프로 표시할 수가 있다. 현재의 PC 의 주류가 되어 있는 GUI 의 선구가 된 기능이다. 그러나 계측, 제어라는 목적으로부터는 부수적인 기능이라는 점, 프로그램의 본 줄기와는 관계없는, 표시를 위한 파라미터의 세세한 설정이 번잡한 점, 중핵을 이루는 HT-BASIC 의 기능으로 대체가 되는 점등으로, 오래된 HP-BASIC 버전에서부터 계속 사용을 해 온사용자들 중에도 HT-BASIC Plus 기능을 사용해 본 적이 있는 사람은 극소수이다.

(9) 그래픽 출력

그래픽스는 직접 CRT 상에 그리는 일이 가능하다. GPIB 라인에 HPGL 이라는 그래픽 언어를 읽는 프로터 (Plotter)가 접속이 되어 있다면 프로터에 출력을 할 수가 있다. CRT 상의 화상을 그대로의 화상파일로서 저장하는 명령은 HT-BASIC 에는 없지만, Windows 에 내장되어 있는 「페인트」 또는 시판이 되고 있는 캡쳐 소프트를 이용하면 비트맵(bit map)등의 화상파일로서 저장할 수가 있다.

3-2 HT-BASIC 의 인스톨

(1) 프로그램

HT-BASIC 의 최신 버전은 HT-BASIC 을 계승한 TransEra 사의 HT-BASIC for Windows 로 현재 버전 9.x 가 나와 있다. 이것은 순서에 따라 인스톨을 하면 되지만, 레거시(legacy) 버전이 아니라 HT-BASIC 을 선택해, 커스톰 인스톨을 선택하여 모든 컴퍼넨트(component)를 인스톨 한다. 전체에서도 약 40M 바이트를 차지할 뿐이다. 인스톨 후의 디렉토리는, 특별히 지정을 하지 않았을 경우,

C:\Program Files\HTBwin

notes; Correctly, although it is HTBwinXY (X:Version and Y:Release), below, it is marked as HTBwin.

로 되어 있다.

(2) GPIB 카드

각사에서 발매를 하고 있지만 이하의 예에서는 NI 사의 GPIB+ 카드를 사용한다. 카드 슬롯에 카드를 집어 넣어 지시에 따라 드라이버를 인스톨을 한다.

(3) 디지털 I/O 카드

각사에서 발매를 하고 있지만 이하에서는 ines 사의 DAQ 카드 i250 를 사용한다. ines 사에서는 수 종류의 카드를 판매하고 있지만 i250 은 지금까지 고장도 없고 가장 많이 출하되고 있는 범용의 계측카드라고 말할 수 있다. 지시에 따라 드라이버를 인스톨을 한다. 준비는 이것뿐이다.

이상으로 대강의 특징을 기술했지만, 다음장부터 프로그램을 작성하는 순서를 나타낸다.

4. HT-BASIC 의 프로그래밍 1 (쌍방향・대화형 제어를 실행하는 프로그램)

HT-BASIC 은 버스라인으로 접속이 된 기기를 제어할 때에 뛰어난 특징을 갖고 있지만 수치계산이나 그래픽스 출력 등을 행하기 위한 보통의 프로그램도 간단하게 작성할 수 있다. 그러면, 4・5 장에서는 대화형 의 프로그램의 작성법, 6・7 장에서는 보통의 프로그램의 작성법과 장점에 관해 설명을 하겠다.

4-1.대화형 프로그램

이 절에서는 인터프린터(interpreter)언어로서의 HT-BASIC 의 특징을 잘 알 수 있는 프로그램의 예를 들어 보겠다. 프로그램의 알기 쉬움, 간단함, 테스트로서 변경을 할 경우의 순서의 간편함 등에 관해 C 나 FORTRAN 과 비교를 해 주었으면 한다. 이하에서는 프로그램의 문장 중에 사용되는 명령을「명령」, 일련의 명령을「명령문」이라고 부르기로 한다. 또한 <ENTER>와 같이 괄호로 둘러싸인 표현은 키보드상의 키의 이름 및 혼란이 되지 않는 범위에서, 키보드에서의 입력이나 선택해야 할 메뉴 혹은 강조 등을 나타내는 것을 의미한다.

HT-BASIC 은 「명령」은 대문자의 알파벳으로 쓰여진다. 이러한「명령」은 프로그램중에서도, 키보드로부터도 사용을 할 수 있는 것이 많기 때문에 혼란이 일어날 염려가 없는 한 별도로 양해를 구하지 않겠다. 「명령」의 해설이나 문법적으로 올바른 사용례는 헬프 매뉴얼에 상세하게 해설이 되어 있다. 그 위에 각「명령」의 구체적인 사용례가 example 파일안에 별도의 프로그램으로서 들어있으므로 참고를 하면 좋다. HT-Basic for WINDOWS ver9.x 의 에디터 화면은「명령」, 변수, 문자열 등마다 제각기 색상을 바꿔서 한눈에 알기 쉽게 표시가 되어 있다. 또 명령이라면 자동적으로 대문자로 변환이 된다든지 문법의 미스를 지적한다든지 등의 기능이 덧붙여져 사용하기에 편리하게 되어 있다. example 프로그램중에 잘 모르는「명령」에 부딪쳤을 경우에는 그 명령을 마우스의 왼쪽 단추로 선택해, 오른쪽 단추로 메뉴를 표시하면 제일 밑 줄에 Help 라는 항목이 나온다. 그 항목을 클릭해서 선택을 하면 그 명령 을 설명하는 항목이 표시가 된다.

(1) EDIT 화면

디렉토리

```
C:\Program Files\HTBwin
```

의 안에 있는 HTBwin.exe 파일을 실행한다. 검은 색의 배경화면이 표시가 되니까, 툴바의 File→New 를 선택해, Edit→Edit Mode 를 선택하면, 배경이 흰색의 프로그램 편집화면으로 변한다. 프로그램 편집은, 임의의 텍스트 에디터로 작성한 것을 컷 앤 페이스트(cut & paste)로, 흰 배경의 프로그램 편집(에디터) 화면에 붙여넣기를 하면 좋다.

(2) 키보드에서의 입력(프로그램의 중단과 변수의 변경, 임의의 줄에서부터의 재개)

화면에서 다음과 같이 프로그램을 입력한다.

```
1 !01_KBD input numerical data.bas
2 !
10 INPUT A
20 PRINT 3*A
30 PAUSE
40 END
```

프로그램 라인을 입력하고 <ENTER>키를 누르면 다음 줄로 커서(cursor)가 이동해서 자동적으로 줄 번호가 매겨진다. <!>보다 오른쪽은 코멘트 문장이다. 줄의 처음에 <!>를 놓으면 그 줄 전체가 코멘트 문장이 된다. 키보드에서,

RUN <ENTER>

라고 입력을 하든가, 툴바 중에 Run(오른쪽을 향한 초록색의 삼각형 기호)을 선택하면 프로그램이 실행된다. 화면의 왼쪽 밑에,

?

가 표시되지만 이것은 키보드로부터의 입력을 기다린다는 것을 나타내는 것이다. 숫자5를 입력해 <ENTER>를 누르면, 화면의 왼쪽 구석에 15 라는 숫자가 표시돼 30 에서 정지한다. 이 때에 변수 A 에는 5 가 대신 들어가게 된다. 그것을 확인하기 위해서 키보드에서

A<ENTER>

를 입력하면 5 가 이번에는 화면의 왼쪽 밑에 나타난다.
프로그램은 정치한 상태이므로 키보드에서

A=123 <ENTER>

를 입력해 다시 한 번 , A<ENTER>라고 입력을 하면, 화면의 왼쪽 밑에 123 이라고 표시가 되어, A 에 대신 123 이 들어갔다는 것을 알 수 있다. 키보드에서,

CONT 20 <ENTER>

를 실행하면 프로그램은 라인 20 줄부터 재개돼 3*123 의 연산결과 369 가 표시되고 정지한다.
라인의 번호를 지정하지 않고,

CONT <ENTER>

를 입력하면, 프로그램은 END 라인에서 종료 한다.
이렇게 인터프린터 언어는 도중에 프로그램을 정지시켜 변수를 변경시킨다든지, 임의의 줄에서부터 재개 시키는 것도 가능하다. 계측 중에 주변기기가 이상한 응답을 해서 프로그램이 정지가 됐을 경우에도 변수를 체크해 프로그램을 중단시키지 않고 재개가 가능한 것이 특징이다.
프로그램을 저장하기 위해서는 EDIT 화면으로 돌아가 툴바의 <파일 → 다른 이름으로 저장 → SAVE AS>를 선택해 적당한 이름을 붙여서 저장한다. <SAVE> 는 프로그램을 텍스트로서 저장한다는 것을 의미한다. 프로그램을 읽을 경우에는 <열기>에서, 파일명을 마우스 왼쪽을 클릭한다는 보통의 방법을 사용한다.
「*"SAVE"* 된 텍스트 파일은 *"GET"* 으로 읽기 시작. 」이라는 것을 의식하지 않고 읽기 시작할 수 있다.

(3) 키보드에서 문자열을 입력(문자 변수의 변경,임의의 줄(행)부터 재개)

EDIT 화면으로 돌아가 전 절에서 설명한 것과 비슷한 아래의 프로그램을 입력해 실행한다.

```
1 ! 02_KBD input characters.bas
2 !
10   INPUT A$
20   PRINT "Abc"&A$
30   PAUSE
40   END
```

10 행의 A$는 문자열의 변수이다. 임의의 변수명의 다음에 <$>를 덧붙이면 그것은 변수명으로 판단된다.화면의 왼쪽 밑에 ? 가 표시되므로,

def <ENTER>

를 입력하면 화면의 왼쪽 위에 「연결」된 문자열 “Abcdef” 가 표시돼 30 행의 PAUSE 에서 프로그램은 정지한다. 20 행의 <&>는 문자열을 연결하는 명령으로 <” “>로 둘러싸인 문자열 ”Abc”라고 입력된 문자열 A$를 연결하고 있다.

A$ <ENTER>

라고 입력을 하면,화면의 왼쪽 밑에“def”가 표시된다. 문자열은<” “>로 둘러싸면 되는데

A$= “xyz” <ENTER>

라고 해서 A$에 새로운 문자열을 부여해서

CONT 20 <ENTER>

를 실행하면 화면의 왼쪽 위에 새롭게 연결된 문자열 “Abcxyz”가 표시된다.

프로그램은 30 행에서 정지가 되어 있으니까

CONT <ENTER>

를 입력하면 프로그램은 END 라인에서 종료된다.

이와 같이 문자열 데이터 조작의 유연함은 C 언어와는 비교가 되지 않는다.

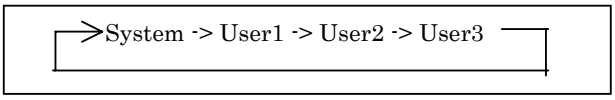
(4) 프로그램의 저장과 불러냄

HT-Basic 에서는 프로그램을 <STORE> 및 <SAVE>의 2 종류의 상태로 보존할 수가 있다. 그 중 하나는 즉시 실행이 가능한 파일로서 <STORE>명령으로 저장한다. STORE 로 저장된 프로그램은 <LOAD> 명령으로 불러내지만, LOAD 명령에 줄(행)번호를 추가시켜 놓으면 LOAD 를 한 다음에 바로 지정된 줄부터 실행이 가능하다. 다른 하나는 ASCII 파일로서 텍스트만은 <SAVE>명령으로 저장한다. SAVE 로 저장된 파일은 <GET>으로 불러 내진다. GET 으로 불러 내진 프로그램은 LOAD 처럼 즉시 실행을 할 수는 없지만, 프로그램을 ASCII 포맷으로 저장을 할 수가 있으므로 다른 시스템에 복사를 할 경우등에 편리하다. HT-Basic 에서는 파일의 타입을 자동적으로 판단해서 프로그램을 불러내기 때문에 LOAD 명령 등으로 특별히 지시를 하지 않은 경우에는 LOAD 인지 GET 인지를 의식하지 않고 프로그램을 불러낼 수가 있다.

(5) 소프트 키와 프로그램의 정지, 속행, 중지

프로그램의 실행 중에는 디스플레이의 화면 밑쪽에 소프트 키의 메뉴가 표시된다. 펑션 키<F9>를 누를 때마다 메뉴의 표시/비표시가 반복된다.

<shift+ F9>를 누를 때마다 소프트 키의 메뉴가



로 순서대로 바뀐다. 각 표시의 소프트 키를 누르던가, 마우스로 왼쪽 클릭을 하면, 키보드상에서 그 명령을 실행한 것과 마찬가지 효과를 갖는다. <System>소프트 키를 표시 시켜 <Run>(F3)을 마우스로 왼쪽 클릭하면 프로그램이 실행된다. <PAUSE>(F4)를 클릭하면 프로그램의 실행이 정지된다. <Continue>(F2)를 클릭하면 정지돼 있던 프로그램이 재차 속행된다. <Stop>(F4)를

클릭하면 프로그램의 실행이 정지돼 시스템은 <idle>상태로 되돌아간다. 소프트 키는, 표시되어 있는 메뉴의 이름하에서 실행을 해야만 할 내용이 키에 정의가 되어 있다. 다시 말하자면 정의가 되어 있는 내용에는 각종의 제어문자를 포함하고 있다.그렇기 때문에 키에 따라서는 표시가 되어 있는 대로 키보드에서 입력을 해도 소프트 키와 같은 효과를 갖고 있는 것은 아니다. 3 개의 (사용자)유저 키 메뉴는 유저가 자유롭게 정의 할 수가 있다. 자기 자신이 정의를 하기 위해서는 디폴트 상태에서의 키의 정의가 참고가 된다. 키에 정의된 내용은 <list key + ENTER>로 표시된다. <LIST KEY #10+ENTER> 로 프린트아웃이 된다. <#10>은 디폴트의 식별 번호이다. (뒤에서 설명 4.3 (2))

4.2 외부 기기의 제어

(1) GPIB 에 의한 기기의 제어 (주소 지정, 기능 설정, 데이터 입력)

그림 2 GPIB Card and Cable (National Instruments Co.)

PC 카드용의 슬롯에 GPIB 카드를 집어넣어 그 카드의 드라이버를 인스톨한다. 다른 한 쪽의 GPIB 커넥터는 GPIB 기기에 접속한다. 여기의 예에서는 NI 사의 GPIB+ 카드를 사용해, GPIB 기기로서 HP 사의 디지털 멀티미터 3457A 에 접속했다. **그림 2** 는 NI 사의 PC 카드와 GPIB 케이블을 나타낸다. 프로그램의 편집화면에 돌아가 디지털 멀티미터에서 데이터를 읽기 위해서는, 이하와 같은 프로그램 라인을 입력한다.

```
1 !03_GPIB control, HP3457A.bas
2 !
10    ASSIGN @Hp3457 TO 722            ! open I/O pass
20    OUTPUT @Hp3457;"NPLC .0005"      ! data to hp3457
30    OUTPUT @Hp3457;"DCV AUTO, .033"  ! data to hp3457
40    WAIT 1   ! wait one second
50    ENTER @Hp3457;B$  ! read character data from hp3457
60    PRINT B$,VAL(B$)  ! display character and value
70    CLEAR 722                        ! close I/O pass
80    END
```

10 행(줄)의 <ASSIGN>은, I/O 보드 7 번을 통해서, GPIB 의 어드레스 22 번 기기로의 I/O path 를 여는 것을 의미한다. I/O 포트 7 번은 보통 GPIB 를 나타낸다. 22 번이라는 것은 GPIB 라인으로 접속 되어 있는 기기가 개별적으로 갖고 있는 주소 번호로, 기기에 각각 0~32 까지의 번호를

설정할 수 있지만, 같은 GPIB 라인으로 접속이 되어 있는 기기 사이에는 같은 번호를 중복해서 설정할 수는 없다. ASSIGN 문중에 <@hp3457>은, <722>라는 번호의 I/O path 에 붙인 이름이다. ASSIGN 문에서 이름<@hp3457> 과 path <722>를 대응시켜 놓으면, 그 뒤의 작업은 이름을 지시하는 것만으로 path 의 지정을 할 수가 있다.

20,30 줄(행)은, <@hp3457>라는 이름으로 지정된 I/O 포트<722>에 데이터를 보내는 명령이며, 반드시 문자열에 의한 데이터이다. 기호나 숫자의 의미는 각 기기마다 다르기 때문에 각각의 매뉴얼에 설명이 되어 있다. HP3457A 라는 디지털 멀티미터의 경우, 20 행은 A/D 변환의 적분시간을 최소치로 하는 설정이며, 30 행은 오토랜지에 의한 DC 전압을 지정된 정밀도로 측정시키는 설정이다.

40 행은 프로그램의 실행을 약 1 초만 느리게 한다 측정 대상이 과도한 응답을 나타내는 경우 등에 측정된 계측계가 안정되기까지의 시간이다. 느리게 하지 않아도 될 경우가 있다. 계측기의 기능만을 보면 단시간내에 측정이 가능하다 하더라도 측정 대상을 포함한 계측계가 안정될 때까지 시간이 걸리는 경우가 있다. 무엇을 측정하려고 하는지를 이해해 놓는 것은 프로그램 작성 이전의 문제이다.

50 행에서는 @hp3457 이라는 이름으로 지정된 722 라는 I/O 포트로부터 데이터를 읽고 있다. 데이터는 문자열로서 출력이 되고 있기 때문에 문자열로서 읽고 있다.

60 행의 <VAL (B$)> 는 문자열 B$ 를 수치로 변환시키는 명령이다. 문자열로서 읽혀진 숫자는 그 상태 로는 수치변수로서 사용할 수가 없다. 60 행에서는 읽혀진 문자열과 그것을 수치로 변환시킨 값을 화면에 출력하고 있다.

70 행은 @hp3457 이라는 I/O 포트를 닫고 있다.

GPIB 버스 라인의 제어는 거의 모든 기기에 대해 이와 같은 순서로 이루어진다.

이 프로그램을 실행하는 경우에는 사전에 GPIB 를 제어시키기 위한 바이너리 프로그램을 읽어 놓지 않으면 안 된다. 그것은 각사의 GPIB 카드마다 다르기 때문에, HT-BASIC 에는 각사의 GPIB 카드에 대응하는 바이너리 프로그램이 준비되어 있다. NI 사의 GPIB 의 경우에는, GPIB 제어를 포함한 프로그램을 실행하기 전에,

```
LOAD BIN "GPIBNI"
```

를 실행하지 않으면 안 된다. <BIN>은 바이너리 프로그램이라는 의미이다.

"GPIBNI"파일은 디렉토리

```
C:\Program Files\HTBwin
```

의 안에 있다. 디폴트의 디렉토리가 위에 기술한 내용과 다를 경우에는,

```
LOAD BIN "C:¥Program Files¥HTBwin¥GIBNI"
```

를 실행한다. 이것은 키보드에서도 실행할 수가 있다. <LOAD BIN "GPIBNI">는 덧쓰기는 안되기 때문에, GPIB 의 제어를 포함한 개별의 프로그램의 선두에 커맨드로서 써 놓고서, 프로그램을 실행할 때마다 읽게 한다는 것을 불가능하다. 사전에 반드시 한 번만 LOAD 를 할 수 있게 한다 제 5 장에서 설명하는 비트의 입출력 카드의 경우에는 카드를 제어하기 위한 전용의 커맨드를 CSUB(컴파일이 끝난 서브루틴) 로서 프로그램을 실행할 때마다 읽게 되어 있다. 이 경우, 프로그램의 처음에 그러한 CSUB 를 읽는 커맨드를 써 놓으면 잊어 버리는 일도 없다. 취급방법이

다른 것은 HP-BASIC 이 탄생한 배경에 기인한다. HP-BASIC 에 있어서 GPIB 의 제어를 한다는 것은 수치의 사칙연산을 실행하는 것과 마찬가지로 기본적인 기능이었다. HP 사 제품의 GPIB 제어 보드밖에 이용할 수 없던 시대에는 시스템의 프로그램에 인스톨을 해 놓으면 좋았다. 그러나 WINDOWS 탑재의 노트 PC 에서는, 각 사로부터 PC 카드라는 형태로 GPIB 제어가 시판되기 시작했기 때문에, 각사의 PC 카드에 대응하는 커맨드를 전부 시스템에 써 놓을 수는 없었다. 그래서 프로그램을 실행하기 전에 LOAD 를 해 놓는다는 취급 방법이 되었던 것이다. 그렇기 때문에 예를 들어, NI 사의 GPIB 카드밖에 사용할 수 없는 경우라면, 나중에 설명하는 오토 스타트 기능 (4.2 (3)절) 을 사용해서 <LOAD BIN "GPIBNI">을 실행시켜 놓으면 된다.

이렇게 해 놓으면 GPIB 를 사용할 때에도 사용하지 않을 때에도 신경을 쓰지 않고 프로그램을 작성해 실행할 수가 있다. **표 1** 에 HT-BASIC Ver9.x 내장의 GPIB 드라이버와 그것에 대응하는 각사의 PC 카드의 일람을 적어 놓았다. 대응하는 WINDOWS OS 에 관해서는 따로 확인을 할 필요가 있다.

HT-BASIC 9.x Name of driver	Manufacturer	PCI bus	PC card	USB bus	ISA/ EISA bus
HPIBS.DW6	HP/Agilent(USA)	82350		82357	82341
	ines(Germany)	GPIB-PCI-NT+	GPIB-PCM-NT+		GPIB-ISA-NT+
	TAMS(USA)	61488		63488	
GPIBNI.DW6	NI(USA)	PCI-GPIB	PCMCIA-GPIB	GPIB-USB	
	ines(Germany)	GPIB-PCI-XL GPIB-LPCI-XL GPIB-cPCI-XL	GPIB-PCM-XL	GPIB-USB-XL	
GPIB900.DW6	TransEra(USA)				HM900

표 1 GPIB supported by the HT-BASIC 9.x

(대응하는 WINDOWS OS 는 각각 확인할 것)

(2) 비트 입출력(GPIO,비트의 조작)

현재는 거의 모든 계측기가 GPIB 인터페이스를 장비하고 있기 때문에, GPIB 버스라인에 의한 제어를 할 경우에도 곤란한 경우는 없지만 상황에 따라서는 릴레이 회로나 변의 개폐등의 자신이 직접 만든 기기를 포함해서 조작을 하고 싶을 경우가 있다. HP 사의 HP9000/300 시리즈의 컨트롤러에는 GPIB 이외에도 GPIO 라는 인터페이스가 있었다. 이것은 완전히 파라렐의 16 비트의 디지털 신호의 입력 및 출력 라인을 제어하는 인터페이스로서 HT-BASIC 언어로 자유롭게 제어할 수가 있었다. 현재에도 데스크탑 PC 에는 PCI 버스를 개재시키는 GPIO 카드가 시판되고 있어서 HT-BASIC 으로 제어할 수가 있다. 노트 PC 용으로는 PC 카드의 제약이 있기 때문에 완전한 파라렐의 16 비트의 디지털 입출력을 제어할 수 있는 것은 없지만 디지털 입출력과 A/D 컨버터로 구성된 PC 카드가 시판되고 있다. 앞으로의 이용을 생각하면 데이터를 A/D 컨버터를 통해서 계측만을 하는 경우도 많으리라 생각한다. 그럴 경우에는 일부러 GPIB 를 개재시켜 전압계를 접속할 필요도 없이, PC 카드 한 장으로 A/D 변환도 하고 디지털 입출력으로 외부기기의

제어도 하는 것이 편리하다. 따라서 비트 입출력에 관해서는 다음의 제 5 장에서 자세하게 해설한다.

(3) 오토 스타트(Auto start)

HT-BASIC 을 기동시키면 프로그램은 같은 디렉토리 안의 <AUTOST>라는 이름의 파일을 검색한다. 만약에 그 이름의 파일이 있다면 프로그램은 자동적으로 <AUTOST>파일을 <LOAD> 해서 실행한다. 이 기능이 HP 사의 컴퓨터에 나타난 것은 탁상계산기 HP9825A 로, 실행중의 시스템의 기능을 일괄해서 기억하는 기능과 함께 사용을 하게 되어 있었다. 프로그램의 제어에 의한 계측 중에, 정기적으로 시스템의 상태를 테이프 커트리지상의 AUTOST 라는 파일에 기억시켜 놓는다. 그렇게 해 놓으면 예를 들어, 계측 중에 정전이 일어나 그 후 복구한 경우에도, 컴퓨터는, 테이프 커트리지상의 AUTOST 파일에 쓰여 있는 시스템의 상태를 바탕으로 시스템을 재차 기동시켜, 계측을 속행시킬 수가 있었다. 현재의 컴퓨터에는 시스템의 상태를 기억시키는 명령은 없기 때문에, 이런 사용법은 없다. 현재의 시스템에서는 실행시에 필요로 하는 바이너리 프로그램을, 사전에 읽혀 놓기 위해서 사용되고 있다. 구체적인 예를 들어 보겠다. AUTOST 프로그램은,

```
C:\Program Files\HTBwin
```

의 디렉토리에 있다. 이것을 에디터 화면상에서 “LOAD”명령으로 연다. 아니면 HT-BASIC 을 기동시키면, 반드시 AUTOST 를 실행하므로 기동 후의 검은 화면이 표시된 상태에서 툴바의 <Edit → Edit Mode>를 클릭하면, AUTOST 의 프로그램이 표시된다. 이 프로그램에는 각종의 바이너리 프로그램과 드라이버를 설정하기 위한 명령이 커맨드문으로서 기술되어 있다. 해설문(영문)을 읽은 다음 LOAD 해야 할 명령문의 선두의 <!>를 삭제하면 그 상태로 명령문이 된다. 필요한 명령문의 선택이 끝나면 <RE-STORE>명령으로 같은 디렉토리에 저장을 해 놓는다. <LOAD>명령으로 프로그램을 로드해, <RE-STORE>명령으로 프로그램을 저장시키지 않으면 안 되는 것은, AUTOST 파일과 CSUB 명령이 프로그램의 제일 뒤에 추기가 되어 있는 프로그램 뿐이다. CSUB 를 포함하지 않는 프로그램은 <(LOAD)/RE-STORE> 라도 <(GET)/ SAVE> 라도 상관없다.

프로그램중에 CSUB 를 이용하는 경우에는 AUTOST 파일 안에서 지시를 하지 말고 예를 들어 프로그램의 제일 첫번째 줄(행)에서,

```
LOADSUB ALL FROM ”c:\inesDAQ\htbasic\daghtb.csb”
```

라는 명령을 사용해서 CSUB 를 프로그램의 제일 마지막에 추가시켜 실행시킨다. 저장을 하는 경우에는 추가된 CSUB 를 제거하면 “SAVE”명령으로 저장할 수가 있다. CSUB 를 남겨 놓은 채로 “SAVE”를 하면 다음에 “GET” 명령으로 불러 냈을 때에 (실제로는 <파일을 연다>라는 작업) 에러가 발생한다. 그러나, 이 에러는 Edit 화면에 추가되어 있는 CSUB 를 제거하면 원래의 텍스트만으로 쓰여진 프로그램으로 돌아오므로, <RUN>으로 실행을 할 수 있다.

4.3 파일 작업

HT-BASIC 에서는 실험 중에 측정한 데이터를 파일에 저장하거나, 프로그램중에 작성한 파일 안의 데이터에 접속하는 일이 매우 간단하다.

(1) 아스키 (ASCII)파일의 작성, 데이터를 씀, 불러냄

프로그램중에 파일을 작성해 그 파일의 데이터를 저장하거나, 불러내는 일도 HT-BASIC 이라면 간단하게 할 수 있다. 다음과 같은 프로그램을 입력한다.

```
1 !04_store data to ASCII file.bas
2 ! read data from file
10   MASS STORAGE IS "d:¥hp¥temp"              ! set up directory
20   CREATE "test.txt",0                       ! create file
30   ASSIGN @File TO "test.txt", FORMAT ON     ! open I/O pass to file
40   FOR I=1 TO 5                              ! loop
50   INPUT "numeric, characters",A,B$    ! input numeric and character by keyboard
60   PRINT A, B$
70   OUTPUT  @File;A,B$                 ! write to file
80   NEXT I
90   RESET  @File                       ! reset file pointer
100  PRINT  "------"
110  FOR I=1  TO  5
120  ENTER  @File;C,D$                  ! read from file
130  PRINT  C,D$
140  NEXT I
150  ASSIGN  @File TO *                 ! close I/O pass
160  END
```

10 줄(행)은 새롭게 파일을 작성하는 디렉토리를 나타내는 명령이다. 보통 키보드에서 입력을 할 때에는 <Mass Storage Is> 의 생략형으로 <MSI "d:\hp\temp" > 라고 입력하면 된다. <MASS STRAGE IS>라고 선언한 디렉토리는, 그 프로그램에서는 명시적으로 변경을 하지 않는 한 암묵적으로 지정이 된 디렉토리로 되어 있다.

20 줄(행)은 10 행에서 지정된 디렉토리에 <test.txt>라는 이름의 파일을 작성하고 있다. HP-BASIC 에서는 <.txt>라는 확장자는 인식을 하지 않기 때문에 <test.txt>라는 이름의 파일에 데이터를 써 넣어도 자동적으로 ASCII 포맷으로서 쓰여지는 것은 아니다. 이 경우의 확장자는, WINDOWS 상의 소프트(예를 들어 표계산 소프트의 EXCEL 등)에서 이 파일의 데이터를 이용할 경우를 위해서 편의적으로 붙여진 것이라고 이해를 해야만 한다. 콤마의 뒤의 숫자는 파일에 기록되는 데이터수를 레코드수로 지정한 것이다. WINDOWS 에서는 파일에 저장되는 데이터의 숫자에 제한이 없기 때문에 여기에서는 <0>이라는 숫자를 사용하고 있다.

30 행에서는 <@File>이라는 이름으로 <test.txt>라는 파일로의 I/O 포트를 열고 있다. <FORMAT ON>은 데이터가 ASCII 표현이라는 것을 나타내고 있다. 만약에 <FORMAT OFF>가 되어 있으면 그것은 바이너리 데이터라는 것을 나타낸다. ASCII 파일이라면 OS 가 바뀌어도 데이터의 호환성이 있기 때문에, ASSIGN 명령을 사용할 때에는 언제나 <FORMAT ON>이라고 명시하는 것이 좋다.

40~80 행은 키보드로부터의 데이터를 읽어서, 표시, 파일에 쓰기 등을 실행하기 위한 루프(loop)이다.

50 행은 위의 (2)절의 INPUT 명령을 쓰는 방법과는 약간 다르다. 50 행에서는 <" "> 내의 문자열을 표시하며, 수치와 문자의 데이터의 입력을 기다리고 있다. 입력을 하려면, 수치와 문자의 데이터를 <,>로 단락을 맺은 다음에 <ENTER>키를 누른다.

60 행에서는 입력한 데이터가 표시되고, 70 행은 파일에 쓰는 것이 실행된다. 파일에 쓰는 작업은

80 행에서 종료된다.
90 행부터는 파일에 쓴 데이터를 읽는 작업이다. 90 행은 파일의 포인터를 리세트하고 있다.
100 행은 <" ">로 둘러싸인 문자열을 화면에 출력하고 있다.
110~140 행에서는 쓴 데이터를 파일에서 실수변수 C 및 문자열 변수 D$ 로 읽어 내고 있다.
150 행에서는 (4)절과는 다른 형식으로 I/O 포트를 닫고 있다. 본래 ASSIGN 은 I/O 포트를 여는 명령이지만, 동시에 복수의 포트를 열 수는 없다. 그렇기 때문에, 불특정의 패스(Path)<*>에 대해 I/O 포트를 여는 명령으로는 어느 포트도 열 수가 없기 때문에 ,결과적으로는 모든 포트를 닫게 된다. 이프로그램을 실행하면, 데이터가 쓰여진 순서대로 읽혀지고 있다는 것을 알 수 있다. 이 파일은ASCII 문자로 데이터가 저장되어 있기 때문에, WINDOWS상의 소프트인 마이크로소프트사의 EXCEL 에서도 읽을 수가 있다.

(1) 파일명의 자동생성과 배열 입출력

앞의 절에서는 데이터를 하나씩 일일이 쓰거나 읽거나 했지만, 배열전체를 그 상태대로 기록하거나 또한 읽을 수가 있다. 또한 작성하는 파일명도 금지된 문자이외에는 임의로 사용할 수가 있다.

```
1 !05_create ASCII file name automatically.bas
2 !
10      OPTION BASE 1                       ! minimum number of descriptor
20      DIM P$[11],Q$[8]                    ! character array (number)[length]
30      DIM A$(3)[18],B$(1:5)[20],C$(1:5)[22]  ! character array (from:end)[length]
40      MASS STORAGE IS "d:¥hp¥temp¥"         ! working drive
50      FOR I=1 TO 3                       !
60         LOOP
70            P$=DATE$(TIMEDATE)           ! read date dd Mmm yyyy
80            Q$=TIME$(TIMEDATE)           ! hh:mm:ss
90         EXIT IF Q$[8;1]="0"             ! [from;lenth]
100         END LOOP
110         A$(I)="data("&Q$[1,2]&"-"&Q$[4,5]&"-"&Q$[7,8]&").txt" ! [from,end]
120         CREATE A$(I),0                  ! create file
130         FOR J=1 TO 5
140            B$(J)=P$&" "&TIME$(TIMEDATE)
150            WAIT 1                      !
160         NEXT J
170         ASSIGN @File TO A$(I);FORMAT ON    ! open I/O pass
180         OUTPUT @File;B$(*)               ! write data to file
190      NEXT I
200      CAT "d:¥hp¥temp"                    ! list files
210      FOR I=1 TO 3
220         PRINT "------"
230         PRINT A$(I)
240         ASSIGN @File TO A$(I);FORMAT ON
250         ENTER @File;C$(*)               ! read data array
260         FOR J=1 TO 5                   !
270            PRINT C$(J)
280         NEXT J
290      NEXT I
300      ASSIGN @File TO *                   ! close I/O pass
310      END
```

그렇기때문에, 실행중의 정전 등을 예측해서 다음과 같은 프로그램을 구성할 수가 있다.

HP-BASIC 에서는 DIM 명령으로 배열을 정의할 때, 배열 지시자의 하한과 상한을 임의로 지정할 수 있다. 10 행은 그 범위가 지정되어 있지 않은 경우에, 그 하한을 1 로 하는 명령이다. 명시적으로 <OPTION BAS E 1>을 지정하지 않을 경우에는. 하한은 <0> (<OPTION BASE 0>을 지정한 경우에 상당)한다.
20 행은 문자열 변수의 문자의 길이를 정의하고 있다. 배열이 아닌 문자열 변수는 프로그램중에 자유롭게 사용할 수가 있으며, 디폴트의 문자의 길이는 18 문자이다. 그렇기때문에 20 행은 일부러 지정할 필요가 없는 명령이지만, 70,80 행에서 P$ 및 Q$ 의 문자의 길이의 한도까지 임의의 문자를 대신 집어 넣을 수 있다는 것을 나타내기 위해서 정의했다. 통상, 사용할 경우에는 여유를 갖고 지정해 놓는 것이 좋다.
30 행의 <A$(3) [18]>은 최대 18 문자열 3 개의 배열을 정의하고 있다. <B$(1:5) [20]>은 배열 지시자를 1 에서 5 까지라고 지정을 해서 정의한 20 문자로 된 배열 5 개이다. <DIM>에서는 이와 같이 배열 지시자의 범위를 지정할 수가 있다.
40 행은 작업 디렉토리의 지정
50~190 행은 <FOR~NEXT>구문의 반복의 루프
60~100 행은 반복작업의 루프이다. 90 행에서 작업 종료의 판단을 한다.
70 행은 날짜를 읽는다. <TIMEDATE>는 기원전 약 48 세기부터 시작하고 있다는 초수로, <P$=DATE$ TIMEDATE)> 라고 부르는 것에 의해 P$ 에는, 예를 들어 <11 Apr 2004> 라는, 스페이스로 단락이 지어진 문자열이 대입 된다. 문자열은 왼쪽 끝부터 차례로 대입이 되기 때문에, 이 경우는 스페이스도 포함해 11 문자이다.
80 행의 Q$=TIME$(TIMEDTE)>에서는 예를 들어<12:54:12>라는 <시,분,초>의 시각을 나타내는 8 문자의 문자열이 대입이 된다.
90 행은 60~100 행의 루프를 벗어나는 판단이다. 식 <Q$[8;1]="0">은 문자열 Q$ 의 8 문자째부터 시작되는 1 문자가 <0>일 경우 1 을 돌려줘 작업의 흐름이 루프로부터 벗어난다. Q$ 의 의미에서 10 초 마다 루프를 벗어나는 일이 된다.
110 행에서는 문자열을 작성해, 배열 A$(I)에 대입을 하고 있다.

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

의 선두부분의 <"data("&Q$[1,2]>는 <"data("> 라는 문자열과 <Q$[1,2]>라는 문자열을 <&>로 연결한 문자열이다. 여기에서 Q$ [1,2]는 문자열 Q$ 의 제 1 문자에서 제 2 문자까지의 문자열이다. 그렇기 때문에, 110 행이 실행이 되면 A$(I)에는 최종적으로 18 문자로 이루어진 <"data(hh-mm-ss).txt">라는 시각을 나타내는 문자열이 대입이 된다. 이와 같이 문자열 중의 일부를 빼내기 위해서는 90 행과 110 행의 두 가지 방법이 있다. 빼어진 문자열의 일부를 HT-BASIC 에서는 [부 문자열]이라고 부르고 있다.
120 행에서 A$(I)의 문자열을 이름으로 하는 파일을 작성하고 있다. 말하자면 시각(時刻)을 이름의 일부로 갖는 파일을 작성하고 있다.
130~160 행은 파일에 쓰는 데이터를 작성하는 루프이다.

140 행에서는 문자열 P$에 스페이스" "를 사이에 끼고 TIME$(TIMEDATA)를 연결하고 있다.
150 행에서 약 1 초만 기다리기 때문에

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

에는, 70 행에서 읽은 P$날짜와 새롭게 읽은 시간이, 예를 들어 <"11 Apr 2004 12:54:33">과 같이 연결 문자열로서 대입이 되게 된다.
170 행에서는 A$(I)의 이름을 갖는 파일로의 I/O 패스(path)를 열어, ASCII 포맷에서의 데이터의 입출력을 지시하고 있다.
180 행에서는 배열 B$(*)의 데이터를 파일에 일괄적으로 쓰고 있다. (*)는 배열 전체를 지정하는 와일드 카드의 지시자이다.
200 행의 CAT 는, 파일이 맞게 작성이 되었는지를 확인하기 위해서, 디렉토리의 파일 리스트를 표시하는 명령이다. 파일이 작성 되어 있는지를 확인을 할 수가 있다. CAT 명령에는,

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

라는 옵션을 붙이는 것으로 문자열 Directory$ 로 지시된 디렉토리의 파일의 리스트를 프린터에 출력시킬 수가 있다. WINDOWS 에서는 #10 으로 디폴트의 프린터를 나타내고 있다. WINDOWS 에는 디렉토리의 파일 리스트를 프린트 아웃하는 명령이 없으므로(DOS 에는 있다), 이 명령은 편리하다. 프린트 아웃의 문자열의 줄이 맞지 않을 경우에는, HT-BASIC 의 편집화면의 상부에 있는 툴바에서 프로포셔널이 아닌 폰트를 선택하면 된다.
<Tool→Device Setup→Win-Print→Properties→Select Font→MS 명조>등
210~290 행은,위에서 기술한 작업을 확인하기 위해서, 3 개의 파일에서, 각각 데이터를 일괄해서 읽고 표시하는 루프이다.
220 행을 단락을 짓는 괘선을 표시하고 있다.
230 행은, 파일명을 표시하고 있다.
240 행은 A$(I)의 이름을 갖는 파일로의 I/O 파스(path)를 열어, ASCII 포맷에서의 데이터의 입출력을 지시하고 있다.
250 행에서는 데이터의 문자열 배열 C$(*)로 일괄해서 읽어 내고 있다.
260~280 행은 읽은 데이터를 표시하고 있다.
300 행에서는 I/O 포트를 닫고 있다.
위에서 상술한 프로그램의 130~160 행의 부분을 , 4.2(1)절에서 설명한 GPIB 제어의 프로그램에 바꿔 쓰면 측정기로부터의 데이터를 수록할 수가 있다. 장기간의 실험 중에는 불의의 정전이라든지 주변기기의 고장 등으로 일련의 측정을 마칠 수 없는 경우도 예상할 수가 있다. 상기의 프로그램과 같이 한 번씩 측정 할 때마다 데이터 파일을 작성해 디스크에 저장을 해 놓으면, 적어도 측정이 중단될 때까지의 데이터는 읽을 수 가 있다.

5. HT-BASIC 의 프로그램 2 (범용 PC (Ines DAQ i250)에 의한 제어) (Reference 3)

여기에서는 Elan 사(영국)제의 AD132 라는 PC 카드의 사용법에 관해 설명을 하겠다. 이 카드는 16 채널의 A/D 컨버터와 8 비트의 디지털 I/O 포트를 갖고 있다. A/D 변환의 정밀도는 12 비트로 속도는 최고 250 kHz 이다. 이 카드를 예로 들은 것은, 통상의 계측은 이 스펙으로 만족할 수 있다는 것, Élan 사의 A/D 컨버터의 시리즈 중에서 지금까지 고장이 났다는 등의 보고가 전혀 없기 때문이다. 이 카드는, 하드웨어의 제조는 Élan 사이지만, 드라이버 소프트는 ines(독일)사가 작성을 하고 있고, ines 사에 의한 모델번호는 ines DAQ i250 이다. **그림 3** 은 ines DAQ i250 PC 카드와 케이블이다. 소프트웨어의 프로그램 중에는 i250 이라는 모델 번호가 나오므로, 이하에서는(ines DAQ) i250 이라고 부르기로 한다. A/D 가 16 채널이라는 것은, A/D 변환 포트를 16 채널 갖고 있다는 것이 아니라, 1 개의 A/D 변환 포트에 대해 16 채널의 애널로그 입력신호를 스캔 할 수 있다는 의미이다.

그림 3 inesDAQ i250 and cable (ines Co.)

5.1 ines DAQ i250 PC 카드의 개요

위에서 설명한 바와 같이, A/D 변환의 정밀도는 12 비트로, 입력 포트 수는 16 채널, A/D 변환의 샘플 레이트는 305.1Hz ~250kHz 이다. 표준의 모드에서는 실수의 데이터를 읽지만, 고속으로 읽기를 할 때에는 정수 모드가 있다. 정수모드를 사용할 경우에는 나중에 정수 데이터를 실수 데이터로 변환시킬 필요가 있다.

입력이 1 채널만의 경우에는 실수, 정수모드 양 쪽 다 250kHz 까지의 샘플 레이트의 지정을 할 수 있지만, 복수의 채널을 사용할 경우에는 90kHz 가 권장되고 있다.(250kHz 을 지정할 수도 있다.) 애널로그 입력의 전압 범위는 최대 −10~+10 볼트로, 양극성에서 16 가지 방법을, 단극성(정전압)에서 8 가지 방법을 지정할 수 있다. 입력채널은 1 채널 씩 지정할 수 있지만, 복수 채널을 사용할 경우에는 연속된 채널을 지정할 수 없다. 입력 모드는 한 쪽 편 공통접지에서 16 채널, 입력모드를 2 개 사용한 차동 입력의 경우에는 8 채널이 된다.

디지털 입출력은 TTL 레벨(0V<Low<0.8V, 2V<High<5V)의 포트를 8 비트 갖고 있다. 입력과 출력의 선택은 1,2,4,8 비트 단위로 지정을 할 수 있지만 그 구성(조립)에는 나중에 설명하겠지만 (**표 2**) 제약이 있다.

카드로부터의 출력 케이블에는 37 극의 D-sub/plug 타입의 커넥터가 접속되어 있다. 디지털 I/O 에는 통일 규격이 없기 때문에, 표 2 의 핀 번호에 따라서 외부회로를 작성하면 좋다.

표 2 Pin assignment of ines DAQ i250

Pin No.	Function	Differential Input	Pin No.	Function
1	A/D input 1	1+	19	DI/O 0
2	A/D input 5	1-	20	DI/O 1
3	A/D input 2	2+	21	DI/O 2
4	A/D input 6	2-	22	DI/O 3
5	A/D input 3	3+	23	DI/O 4
6	A/D input 7	3-	24	DI/O 5
7	A/D input 4	4+	25	DI/O 6
8	A/D input 8	4-	26	DI/O 7
9	A/D input 9	5+	27	+5V
10	A/D input 13	5-	28	+15V
11	A/D input 10	6+	29	-15V
12	A/D input 14	6-	30	grounding
13	A/D input 11	7+	31	trigger input
14	A/D input 15	7-	32	grounding
15	A/D input 12	8+	33	grounding
16	A/D input 16	8-	34	grounding
17	Analog grounding		35	grounding
18	Digital grounding		36	grounding
			37	grounding

5.2 A/D 변환의 프로그램

ines DAQ i250 의 A/D 변환기능은, 채널의 선택, 변환속도의 지정, 데이터 수, 트리거 설정 등 세세한 지정을 할 수 있게 되어 있다. 이하에서는 1 개씩 데이터를 수록하는 경우와, 고속으로 데이터를 수록하는 경우의 예를 나타낸다.

(1) 1 채널로 단 한 개만의 데이터를 읽는다

ines DAQ i250 용의 드라이버를 인스톨해서 이하의 프로그램을 입력한다. 일반적으로 PC 카드를 조작할 때에는, 몇 갠가의 CSUB(컴파일 된 서브 루틴)를, 순서에 따라 불러낼 필요가 있다. 다음의 프로그램은 i250 의 A/D 기능을 사용할 경우의 골격을 나타낸다.

```
1 !06_Ines DAQ - ADC_singlechannel_real_single_input.bas
2 !
10      LOADSUB ALL FROM "c:¥inesDAQ¥htbasic¥daqhtb.csb"
20      OPTION BASE 1
30      DIM A(1)                                ! data array
40      INTEGER Dummy,Hadc,Cadc
50      DIM L$[256]                              ! string variable of 256 characters
60      CALL Daqhtb_ioctl(Dummy,"( fctn init ) ")   ! initialization of PC card
70      CALL Daqhtb_open(Hadc,"i250 ADC",0)    ! designate A/D with real mode (0)
80      L$="( ioa ( timeout 10000 fsmpl 100000 chan 1 iomode single trigmode off samples
1 range [ -10 10 ] ) )"               ! describe A/D function
90      CALL Daqhtb_ioctl(Hadc,L$)             ! describe A/D function to PC card
100      CALL Daqhtb_read(Cadc,Hadc,A(*))          ! read one datum
110      PRINT A(1)                              ! print datum
120      CALL Daqhtb_close(Hadc)                 ! close PC function
130      END
```

10 행에서는 HT-BASIC 에서 Ines DAQ 카드를 조작하기 위한 CSUB(컴파일 된 서브 루틴)을 읽고 있다. 이 명령이 실행되면 Ines DAQ 를 사용하는 데에 필요한 CSUB 가 프로그램의 제일 마지막에 추가가 된다.
20 행은 배열의 인수가 1 부터 시작되는 것을 지정하고 있다.
30 행은 데이터를 수록하기 위한 실수배열을 선언하고 있다. 배열 A 의 요소수가 1 개인 것에 주의를 해 주기 바란다.
40 행는 변수 정수의 정의이다. <Dummy>,<Hadc>,<Cadc>는 ines DAQ 를 조작할 때에 사용되는 되돌아가기 변수이며,이름은 임의로 정할 수 있다.
60 행에서는 PC 카드를 초기화하고 있다.

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

인수 중에 Dummy 는 inesDAQ 카드를 초기화 할 때에 부가하는 변수로 프로그램에서는 사용을 하지 않는다. 2 번째의 인수인 문자열<"(fctn init)">은 PC 카드의 초기화를 할 때의 동작 정의문이며, 그대로 쓰지 않으면 안 된다. 초기화에서는 FIFO 버퍼(buffer)의 사이즈(디폴트의 1 메가 바이트)와 IRQ(interrupt request)발생까지의 샘플수(디폴트 8192, 내부 버퍼의 반)를 지정할 수 있다.
70 행은 PC 카드의 A/D 변환기능을 불러내고 있다. 인수중의 <"i250 AD">는 i250 이라는 이름의 카드의 ADC(A/D 변환)기능을 사용한다는 의미이다. 정수변수<Hadc>에는 <"i250 ADC">에서 특정된 핸들 (PC 카드의 기능을 인식하는 번호)의 수치가 되돌아온다. 이 이후에 동작이 종료될 때까지 CSUB 를 불러 낼 때마다, 반드시 변수<Hadc>가 인수로서 사용된다. 수치<0>은, 동작을 실수치로 실행하는 것을 의미한다. 고속으로 데이터 수록을 할 경우 등 정수치로 읽을 경우에는 <1>을 지시한다.(이후에 설명)
80 행은 A/D 변환동작의 내용을 정의하는 문자열이다. 프로그램의 예에서는 문자열이 길기 때문에 2 줄로 나누어서 표시하고 있지만, 입력을 할 때에는 도중에 개행<ENTER>를 집어 넣어서는 안 된다. 나중에 프로그램을 수정할 경우를 생각해서, 2 줄(행)이상으로 명시적으로 나누어서 알기 쉽게 표시를 하기 위해서는, 몇 갠가의 문자열로 분할을 하면 좋다. 80 행의 경우에는, 50 행의 <DIM>의 문장 중에 문자열 변수 <P$[128],Q$[128]>를 정의해 놓으면, 이하와 같이 다시 고쳐 쓸 수가 있다. 각 지시항목의 사이는, 스페이스로 단락을 짓는다는 점에 주의 할 것.

```
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>는 애널로그 신호의 I/O 동작(이 경우는 A/D 변환)을 실행하는 것을 나타내며, 계속되는 <()>의 안에 세세한 동작을 지시하고 있다.

```
timeout 10000          ! timeout of operation is 10000msec.
fsmpl 100000           ! sampling frequency is 100kHz
chan 1                 ! A/D conversion of voltage signal at channel #1
iomode single          ! Input mode of voltage signal is single-end grounding
trigmode off           ! no trigger mode used
sample 1               ! read only single data
range  [-10 10]        ! range of input voltage is -10~+10V
```

각 문자나 수치의 사이에는 스페이스를 집어 넣는다. 이 외에도 기능은 지정할 수 있지만, 지정하지 않은 기능에는 디폴트의 값이 주어진다. 전압신호의 입력모드는 2 종류가 있다. 싱글 엔드의 경우에는 16 채널을 사용할 수 있고, 2 개의 채널을 사용하는 차동 모드의 경우에는, 짝(쌍)이 되는 채널번호는 정해져 있어서 표 2 와 같이 8 채널을 사용할 수 있다. 트리거 모드는 세세하게 지정할 수가 있다. 입력전압범위는 이미 정해져 있어서, 양음 (플러스,마이너스)에 걸친 것(16 종류)와 양(플러스)만의 범위의 것(8 종)을 지정할 수 있다.
90 행에서는 80 행의 동작 정의문을 핸들번호<Hadc>로 지정된 기능에 써 넣고 있다.
100 행의 배열<A(*)>에 지정된 개수만큼 데이터를 읽고 있다. 여기의 예에서는 데이터를 한 개만 읽기 때문에 배열 A(*)의 요소의 수는 1 개로 했지만, 1 이상이라면 괜찮다. 되돌아오는 변수(Cadc>에는 읽은 데이터의 수가 들어가 있다.
110 행에서는 데이터를 디스플레이에 프린트 아웃한다.
120 행에서는 PC 카드의 동작을 종료시키고 있다. inesDAQ 에서는 여러 가지의 동작을 지정해서 조작할 수가 있지만, 새로운 동작의 지정을 할 때에는 반드시 그 전에 동작을 종료시켜 놓지 않으면 안 된다.
130 행은 프로그램의 종료문이다.
이 예와 같이 inesDAQ는 초기화에서 종료까지 5 종류의 일련의 CSUB 를 불러내고 있다. 그것들을 처리의 순서대로 열거하면:

1.	initialize PC card
2.	input A/D conversion or setting up digital I/O function
3.	specify the contents of operation in detail
4.	input or output of data
4'.	convert integer data to real value
5.	end operation of PC card

가 된다. 4'는 고속으로 데이터를 읽는 경우에 필요한 처리로, (4) (ii)「정수 모드의 입력」의 항목에서 설명한다.

(2) 복수의 채널로 1 개씩 데이터를 읽는다

복수의 채널로 한 개씩 데이터를 읽는 경우의 프로그램은 이하와 같다. 프로그램의 흐름은 1 채널의 경우와 다르지 않다.

```
1 !07_Ines DAQ - ADC_multichannel_real_single_input.bas
2 !
10      LOADSUB ALL FROM "c:¥inesDAQ¥htbasic¥daqhtb.csb"    ! load CSUBs
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 행에서는 2 개의 요소를 갖는 데이터 읽기용의 배열 A(*)을 정의하고 있다.

50 행은 PC 카드의 초기화.

60 행은 실수 모드<0>에 의한 A/D 변환을 open 시킨다.

70 행은 동작을 지시하고 있다. 프로그램의 예에서는 문자열이 길기 때문에 2 줄(행)으로 나눠서 표시하고 있지만, 입력을 할 때에는 도중에 개행<ENTER>를 집어 넣어서는 안된다.

```
fsmp1 100000            ! sampling frequency is 100kHz
chan 102                ! A/D conversion of signals from channel 1 to channel 2
samples 2               ! two data to be read
trigmode off            ! trigger mode off
```

복수의 A/D 채널을 사용하는 경우는, ines DAQ 에서는 하드상의 제약이 있어서 측정하는 채널을 랜덤으로 선택할 수는 없다. 반드시 연속된 복수의 채널을 지정하지 않으면 안된다. 제일 마지막의 채널의 번호가 한 자릿수의 경우에는 번호의 앞에 0 을 집어 넣는다. 복수 채널의 데이터 수록을 할 경우에 트리거모드를 지정하면, 어느 채널부터 데이터의 읽기가 시작됐는지 모르게 되기 때문에, 권장할 수가 없다. 단, 지정된 복수의 채널 중에 최초의 채널을 트리거 전용으로 사용하고, 입력전압 렌지를 넘는 레벨로 트리거를 시키려고 하면, 다음의 데이터는 2 번째의 채널에 입력이 되게 된다. 데이터를 전부 수록한 다음에, 선두 채널의 데이터를 폐기하게 하면, 트리거에 의한 복수 채널의 A/D 입력을 실현시킬 수 있게 된다. 샘플링 주파수는 2 채널 합계가 100kHz 라는 의미이다.

80 행에서는 PC 카드의 번호 Hadc 에서 지정된 기능에 동작 정의문 L$를 써 넣고 있다.

90 행에서는 1,2 채널에서 데이터를 한 개씩, 계 2 개, 써 넣고 있다. 1 채널의 데이터는 A(1)에, 2 채널의 데이터는 A(2)에 읽혀지고 있다.

100 행은 프린트 아웃

일정의 시간 간격을 두고, 연속된 데이터를 읽는 경우에는 (1) 또는 (2)에서 나타낸 동작을, 루프(6 장 참조)의 안에 집어 넣어 반복을 하면 된다. 그 때, 4.3(1),(2)절에서 설명을 한 것과 같이, 파일 안에 데이터를 써 넣게 한다면, 하드 디스크의 용량까지, 거의 무제한으로 데이터를 기록할 수가 있다.

(3) 1 채널로 고속으로 데이터를 수록

통상은 정해진 시간 간격으로 데이터를 한 개씩 측정하고 있어도, 어떤 이벤트가 발생했을 때에는 고속으로 데이터를 수록해, 그 변화를 관측하고 싶을 경우가 있다. 그러한 경우의 프로그램의 예를 다음에 들어 보기로 하겠다. 1 채널의 경우, 250kHz 까지의 샘플링이 가능하다.

```
1 !08_Ines DAQ - ADC_singlechannel_real_streaml_input.bas
2 !
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 행에서는 데이터 버퍼(buffer)가 되는 배열 A(*)을 정의하고 있다.
60 행에서는 실수치로 변환하는 것을 지정(0)
70 행에서는 동작을 지정하고 있다. 프로그램의 예에서는 문자열이 길기 때문에 2 줄(행)로 나눠서 표시 하고 있지만, 입력을 할 때에는 도중에 개행 <ENTER>를 집어 넣어서는 안 된다.

```
fsmp1 100000            ! sampling frequency is 100kHz
trigmode level.gt       ! trigger mode “when signal value is more than specified level”
triglevel –10.0         ! trigger level is –10V
pretrig 0               ! pre-trigger is not to be loaded
samples 1024            ! 1024 data to be loaded
```

읽는 데이터의 길이는 64(2^ 6)에서 16348(2^ 14)의 범위로, 2^ n 의 수치로 지정한다. 트리거 모드를 지정(<trigmode off>가 아닌 경우)하면, 언제나 A/D 변환이 이루어지기 때문에. 트리거가 걸린 순간부터 이전의 데이터도 측정이 되고 있다. 그 데이터를 어디까지 읽을 것인지를 <pretrig>로 지정을 한다.
80 행에서 A/D 변환 모드의 동작 정의문을 써 넣는다.
90 행에서 배열 A(*)에 읽혀진, 버퍼가 가득 차게 되면 읽는 것을 중지한다. 이 프로그램의 예에서는 1 번 채널로부터의 전압을 감시해 트리거 신호를 내보내고 있다. 트리거 전압으로서 70 행에서 지시한 렌지의 범위를 넘는 전압(-10V 이하.또는 +10V 이상)을 지시할 경우에는, **표 2** 에 표시되어 있는 트리거 입력 핀 31 번에 트리거 신호를 인가한다.

(4) 복수의 채널로 고속으로 데이터를 수록

복수의 채널로 고속으로 데이터를 수록하는 경우에는, 실수배열에 데이터를 수록하는 경우와 정수배열에 수록하는 경우가 있다. 실수배열에 수록하는 경우는 A/D 의 결과를 실수로 변환하는

데 시간이 걸린다. 변환을 실시하지 않고 정수배열에 수록하면 고속의 데이터 수록을 할 수 있지만, 나중에 정수 데이터를 실수 데이터로 변환시키지 않으면 안된다. 복수 채널의 경우에는 90 KHz 까지의 샘플링이 권장되고 있다.

(i) 실수 모드로의 데이터 수록

2 개의 채널의 A/D 포트로부터 데이터를 입력하는 프로그램을 이하에 나타낸다.

```
1 !09_Ines DAQ - ADC_multichannel_real_streaml_input.bas
2 !
10      LOADSUB ALL FROM "c:¥inesDAQ¥htbasic¥daqhtb.csb"
20      OPTION BASE 1
30      DIM A(1024),Dat(512,2),L$[256]         ! definition of data array
40      INTEGER Dummy,Hadc,Cadc
50      CALL Daqhtb_ioctl(Dummy,"(fctn init)")
60      CALL Daqhtb_open(Hadc,"i250 ADC",0)
70      L$="( ioa ( timeout 10000 fsmpl 90000 chan 102 iomode single trigmode off
samples 1024 ) )"
80      CALL Daqhtb_ioctl(Hadc,L$)             ! describe function
90      CALL Daqhtb_read(Cadc,Hadc,A(*))      ! read data to A(*)
100      CALL Daqhtb_close(Hadc)               ! close function of PC card
110      FOR J=1 TO Cadc/2.                    ! devide data to each channel
120         Dat(J,1)=A(2*J-1)
130         Dat(J,2)=A(2*J)
140      NEXT J
150      END
```

30 행에서는 수록하는 데이터의 배열을 선언하고 있다. 2 채널을 사용해, 배열 A(*)에 1024 개의 데이터를 수록한다. 그 다음에 각 채널마다 512 개의 데이터를 배열 Dat(*)에 할당한다.

50 행은 PC 카드의 초기화

60 행에서는, PC 카드의 A/D 변환 기능을 실수 모드(0)에서 연다.

70 행은 동작을 지정하고 있다. 프로그램의 예에서는 문자열이 길기 때문에 2 줄(행)로 나눠서 표시하고 있지만, 입력을 할 때에는 도중에 개행 <ENTER>를 집어 넣어서는 안된다.

```
fsmp1 90000           ! sampling frequency is 90kHz
chan 102              ! scan A/D input of successive channel 1 to channel 2
iomode single         ! single-end grounding input
triggmode off         ! trigger mode off
samples 1024          ! total 1024 data to be loaded by two channels
```

A/D 입력에 사용하는 채널은 연속된 것이 아니면 안된다. 한쪽편 접지의 채널은 16 채널, 차동입력의 경우에는 8 채널을 지정할 수 있다. 입력 케이블의 접속처는 표 2 와 같다.

80 행은 동작 내용을 PC 카드에 보내고 있다.

90 행은 데이터를 읽고 있다. A/D 컨버터는 지정된 샘플 레이트로 채널을 바꿔가면서 데이터를 배열에 보내기 때문에, A(*)에는 2 채널 교대로, 합계 1024 개의 데이터가 읽혀진다. 되돌아가기 변수 Cadc 에는 읽은 데이터수가 들어 있다.

100 행에서 카드의 동작이 종료되고 있다.

110~140 행은, A(*)에 읽혀진 데이터를 채널 1 과 채널 2 에 할당하는 루프이다. 각 채널의 데이터

수는, <Cadc/2>이다.

② 정수모드로의 데이터 수록

실수 모드의 측정에서는, 측정을 할 때마다 실수로의 변환하는 시간이 걸린다. 그래서, 고속의 현상을 관측하기 위해서 실수로의 변환을 실행하기 전에, 2 치(값)신호의 상태로 데이터를 수록해, 측정이 끝나고 나서부터 실수로의 변환이 이루어진다. 정수 모드의 경우, PC 카드의 정수 데이터가 32 비트(4 바이트)인 것에 대해, HT-BASIC 의 정수 데이터는 16 비트(2 바이트)이기 때문에, 수록하고 싶은 데이터의 2 배의 요소를 갖는 배열을 준비해 놓을 필요가 있다. 이하에서 그 프로그램의 예를 들어 보겠다. 측정하는 데이터수는, 예에서는 각 채널마다 256 개 있지만, 2^ n 으로 임의로 지정할 수가 있다.

```
1 !10_Ines DAQ - ADC_multichannel_integer_streaml_input.bas
2 !
10      LOADSUB ALL FROM "c:¥inesDAQ¥htbasic¥daqhtb.csb"
20      OPTION BASE 1
30      INTEGER A(1024)                ! define integer array
40      DIM R(512),Dat(256,2)          ! define real array
50      INTEGER Dummy,Hadc,Cadc        ! return variables
60      DIM L$[256]                    ! string array of 256 characters
70      CALL Daqhtb_ioctl(Dummy,"(fctn init)") ! initialization of PC card
80      CALL Daqhtb_open(Hadc,"i250 ADC",1)    ! read data with integer mode (1)
90      L$="( ioa ( timeout 10000 fsmpl 100000 chan 102 iomode single trigmode off samples
1024 ) )"
100      CALL Daqhtb_ioctl(Hadc,L$)
110      CALL Daqhtb_readi(Cadc,Hadc,A(*))     ! read data to integer array
120      CALL Daqhtb_demangle(R(*),A(*),L$,0)  ! conversion integer to real
130      CALL Daqhtb_close(Hadc)               ! close function of PC card
140      FOR J=1 TO Cadc/2.                    ! divide data
150         Dat(J,1)=R(2*J-1)
160         Dat(J,2)=R(2*J)
170      NEXT J
180      END
```

30 행은 A/D 변환한 binary 데이터를 그대로의 상태의 정수치로서 읽기 위한 배열을 선언하고 있다. PC 카드 에는 32 비트의 정수치를 출력하지만, HT-BASIC 에서는 정수치는 2 바이트(16 비트)로 취급된다. 1 개의 정수 데이터를 읽기 위해서는 2 개의 정수배열 요소가 필요하다. A(1024)라는 것은,정수 데이터 512 개분의 배열이다.

40 행은 실수배열의 선언이다. R(512)는 A(*)에 읽혀진 1024 개의 데이터를 실수치로 고친 데이터를 보존하는 배열이며,

80 행에서는 PC 카드의 A/D 변환 기능을, 정수변수로 실행하도록 지시<1>하고 있다.

90 행은 동작 내용의 지시이다. 프로그램의 예에서는 문자열이 길기 때문에 2 줄(행)로 나눠서 표시하고 있지만, 입력을 할 때에는 도중에 개행 <ENTER>를 집어 넣어서는 안 된다.

```
fsmp1 100000          ! sampling frequency is 100kHz
chan 102              ! scan A/D input of successive channel 1 to channel 2
iomode single         ! single-end grounding input
triggmode off         ! trigger mode off
samples 1024          ! total 1024 data to be loaded by two channels
```

110 행에서 배열 A(*)에 수록되는 것이지만, 위에서 설명한 것과 같이 A 의 요소 2 개로 PC 카드의 정수데이터 1 개에 상당한다. 그렇기 때문에 samples 1024 에서는 실제로는 512 개의 정수 데이터가 배열 A 의요소에 읽히게 된다.
110 행에서는 PC 카드에 동작 내용을 써 넣고 있다.
실수 모드의 경우에는 <Daqhtb_read>였지만, 정수 모드 에서는 정수(integer)를 나타내는 <Daqhtb_readi>로 변해 있다.
120 행은, 정수모드의 경우에 필요한 기능으로,정수 데이터를 실수 데이터로 변환시키고 있다. 배열 A(*)의 데이터수는 1024 개이지만, 실제로 읽혀진 데이터수는 512 개이기 때문에 실제 배열 R(*)의 요소는 512 개 이다.
130 행에서 동작을 종료시키고 있다.
140 행부터의 루프로 각 채널별로 데이터를 할당하고 있다.

5.3 비트 입출력의 프로그램

inesDAQ 의 디지털 I/O 기능은, 합계 8 개 있는 비트 포트의 기능(입력 or 출력)을 지정해, 출력하는 데이터의 쓰기 및 입력 데이터를 읽을 수 있도록 되어 있다. 각 비트 포트를 지정하는 방법을 표 3 에 나타냈다. 카드의 크기에 제약때문에, 각 포트의 입력. 출력의 지정을 임의로 행할 수가 없기 때문에 , 표 3 에 나타낸 것과 같이 1,2,4,8 개씩 한꺼번에 지정하도록 되어 있다. 실용상으로는 그렇게 불편한 점은 없으리라 생각한다. 이하에서는 비트 출력의 경우와 비트 입력의 경우의 예를 나타낸다.

(1) 비트 데이터의 출력

다음의 프로그램을 입력한다. CSUB(컴파일 된 서브 루틴)를 불러내는 순서는 A/D 변환의 경우와 다르지 않다.

```
1 !11_Ines DAQ - DIO_output_manualtest.bas
2 !
10      LOADSUB ALL FROM "c:¥inesDAQ¥htbasic¥daqhtb.csb" ! Load CSUB
20      OPTION BASE 1
30      DIM D(1)
40      INTEGER Dummy,Hdio,Cdio
50      CALL Daqhtb_ioctl(Dummy,"(fctn init)")        ! initialization of card
60      CALL Daqhtb_open(Hdio,"i250 DIO",1)           ! choose DIO function
70      CALL Daqhtb_ioctl(Hdio,"( iod ( dir 255 ) )")! specify digital output(255)
80      INPUT P                                  ! input decimal data from keyboard
90      D(1)=BINAND(P,255)                       ! lower 8 bit of binary expression
100      CALL Daqhtb_write(1,Hdio,D(*))
110      CALL Daqhtb_close(Hdio)
120      END
```

10 행에서는 PC 카드를 제어하기 위한 CSUB 를 일괄해서 읽고 있다.
20 행은, 배열의 지시자를 (0 부터가 아니고)1 부터 시작하게 지정하고 있다.
30 행은 비트 출력치를 부여하기 위한, 1 개로 이루어진 요소수의 배열 D(*)이다.
60 행에서는 InesDAQi250 의 디지털 I/O 기능을 정수 모드로 열고 있다.
70 행에서는 표 2 에 따라, 디지털 I/O 포트를 8 비트도 출력용으로 지정하고 있다.
80 행에서는 키보드에서 수치를 입력하고 있다.

90 행에서는 입력된 수치의 하위 8 비트만을 꺼내서 배열 D(1)에 대입하고 있다. BINAND(A,B)는 2 개의 수치 A 와 B 의 16 비트 표현의 비트마다의 AND 를 끄집어 내는 함수이다.

100 행에서는 1 개의 데이터를 쓰고 있다.

110 행은 동작의 종료.

(2) 비트 데이터의 입력

70 행에서 비트데이터의 출력 기능을 지정하고 있는 것 이외에는, 비트 출력의 경우와 동일

```
1 !12_Ines DAQ - DIO_intput_manual_test.bas
2 !
10      LOADSUB ALL FROM "c:¥inesDAQ¥htbasic¥daqhtb.csb"
20      OPTION BASE 1
30      DIM D(1)                                  ! data array for bit input
40      INTEGER Dummy,Hdio,Cdio
50      CALL Daqhtb_ioctl(Dummy,"(fctn init)") ! initialization of card
60      CALL Daqhtb_open(Hdio,"i250 DIO",1)            ! DIO function
70      CALL Daqhtb_ioctl(Hdio,"( iod ( dir 0 ) )")   ! specify bit data input (0)
80 !    ----------
90      CALL Daqhtb_read(Cdio,Hdio,D(*))       ! input binary data
100      CALL Daqhtb_close(Hdio)               ! close function
110      END
```

70 행에서는 8 비트의 포트 전부를, 입력 포트로 지정하고 있다.

각 비트의 입력은, TTL 레벨 (0V<Low<0.8V,2V<High<5V)에서 부여하지 않으면 안 된다.

표 4 는 GPIO (16 비트의 입출력을 병렬로 실행하는 비트 제어 버스)라인에서 사용하는 외부회로의 예를 나타낸다. 이 보드에서는 GPIO 라인에 의해. 16 개의 외부 릴레이의 구동, 2 채널의 16 비트 디지털 카운터로부터 입력, 교류 100 볼트의 릴레이 구동 및 랜덤 신호의 발생 제어가 실행할 수 있게 되어 있다. **표 5** 는 inesDAQi250 의 디지털 I/O 기능의 테스트 회로이다. 16 채널의 애널로그 입력, 8 비트의 디지털 입출력 기능을 테스트할 수 있다.

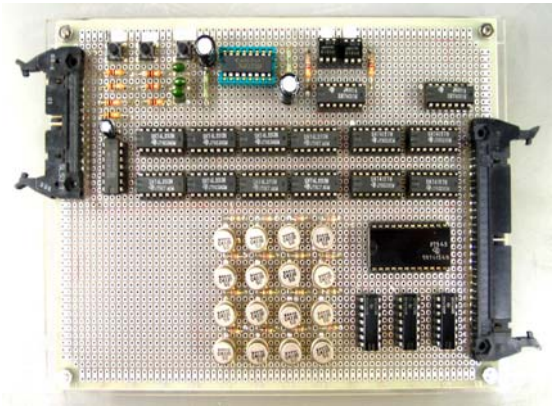


그림 4 GPIO 제어 보드(16 비트의 병렬 입출력)

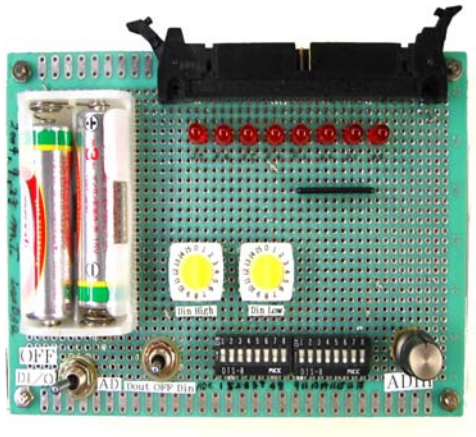


그림 5 ines DAQ i250 카드의 입출력 테스트 회로의 예

표 3 비트 포트의 지정 방법(I : 입력, O : 출력)

hexadecimal format	0x0	0x1	0x2	0x3	0xC	0xD	0xE	0xF	0xF0	0xF1	0xF2	0xF3	0xFC	0xFD	0xFE	0xFF
Decimal format	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

6. HT-BASIC 의 프로그래밍 3 (프로그램의 구조(루프와 분기[갈라짐],계산)

이전의 절에서는, C 언어 등에서는 간단하게 할 수 없는 대화형 프로그램에 초점을 맞춰서 특징을 설명했다. 이번 장에서는 통상의 프로그래밍의 방법에 관해 전체적으로 살펴보기로 한다.

(1) 루프와 분기(갈라짐) (FOR~NEXT, IF~THEN, LOOP, WHITE, REPEAT, SELECT~CASE)

(i) FOR~NEXT

```
1 !13_for-next loop.bas
2 !
10      FOR I=1 TO 100
20         FOR J=2*PI TO 0 STEP -PI/100
30 !---------------------------------------
80         NEXT J
90      NEXT I
100 END
```

20 행과 같이 2n 부터 0 까지 -n/100 씩 변화될 수 있게 step 에서 증가(감소)한 분을 지정할 수 있다.지정을 하지 않은 경우에는 1 로 간주한다.

(ii) IF~THEN(~ELSE~END IF)

조건식이나 논리식은 그 평가의 결과가 true 의 경우에 <1>(정)을 돌려준다.

```
1 ! 14_if-then-else branching.bas
2 !
10      IF J2=K THEN 110
20  !      --------
30      IF X=Y THEN Y=Z*Z
35  !      --------
40      IF A<0 THEN
50         GOTO End
60       ELSE
70  !      --------
80      END IF
85  !      --------
100  End: END
```

10 행은 <J2=K>라는 논리식이 true 의 경우에 90 행에 프로그램의 흐름이 갈라진다.

30 행에서는 조건식이 true 의 경우, Y=Z*Z 를 실행한다.

40~80 행에서는 A<0 이 true 의 경우에 End 라는 이름을 갖는 90 행으로 갈라진다. 그렇지 않은(false)의 경우는 ELSE 이하의 프로그램을 실행한다. 각 행에는 자유롭게 이름을 붙일 수가 있다. 행의 이름 다음 에는 콜론<:>으로 단락을 짓는다.

(iii) LOOP~NEXT IF~END LOOP

```
1 !15_loop_loop decision in loop.bas
2 !
100      LOOP
110  !      ----
170      EXIT IF J=5 OR A$>B$
175  !      ----
190      END LOOP
200 END
```

100~190 행은 반복해서 실행해, 170 행의 논리식이 true 의 경우에 LOOP 를 빠져 나온다. 루프의 도중에 조건 판단을 한다.

(iv) WHILE~END WHILE

```
1 !16_while_loop decision at beginning.bas
2 !
100      WHILE X<1000
150 !      ----------
200      END WHILE
250 !      ----------
300 END
```

100 행의 논리식이 진(true)의 경우에 200 행까지의 프로그램을 반복한다. 루프의 선두에서 조건판단을 한다.

(v) REPEAT~UNTIL

```
1 !17_repeat-until loop.bas
2 !
60      REPEAT
65 !       -----
80      UNTIL X=10
100 !       -----
150 END
```

80 행의 논리식이 진(true)이 될 때까지 REPEAT 에 돌아가 처리를 반복한다. 루프의 제일 마지막에서 조건판단을 한다.

(vi) SELECT~CASE~END SELECT

```
1 !18_select-case branching
2 !
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
90 !   ------------------
100  END
```

10 행에서 Option 이라는 문자열 변수를 선택해, 20 행 및 40 행의 CASE 문에서 그 내용을 비교하고 있다. 논리식<Option$="B">가 진(true)이라면 30 행을 실행해, 어느 CASE 문하고도 일치를 하지 않는 경우에는 70 행을 실행한다. 수치 변수를 SELECT 한 경우에는 수치에 의한 논리식을 사용한다.

(1) 서브 프로그램과 함수 1 (SUB,COM,FN)

(i) 주된 프로그램과 변수를 공유하지 않는 경우

```
1 !19_sub_program_without_common_variables.bas
2 !
10      CALL Mysub              ! call subprogram
20      END                     ! end of main program
30      SUB Mysub               ! definition of subprogram
40         PRINT "In My SUB"
50      SUBEND
```

10~20 행이 주된 프로그램으로서 Mysub 라는 서브 프로그램을 부르고 있다. 서브 프로그램은 주된 프로그램이 끝나는 것을 나타내는 <END>문의 다음에 기술하겠다.

(ii) 주된 프로그램과 변수를 공유하는 경우(COM)

```
1 !20_subprogram_with_common_variables.bas
2 !
10      COM /Test/A$(2)[20],P          ! declaration of common variables
20      A$(1)="He"
30      A$(2)="llo"
40      P=7
50      Subit                          ! call subprogram
60      END                            ! end of main program
70      SUB Subit                      ! definition of subprogram
80         COM /Test/C$(*),R           ! declaration of common variables
90         PRINT "A$(*)=";C$(1)&C$(2),"P=";R! processing
100      SUBEND
```

10 행의 (COM)문에서 공통의 변수를 정의하고 있다. </TEST/>는 정의된 변수를 하나의 블록으로서 붙인 이름이다.
50 행에서는 서브 프로그램을 부르고 있다. 보통은 CALL 문을 사용해서 <CALL Subit>처럼 불러내지 않으면 안되지만, 서브 프로그램을 불러내기만 하는 행이라면, CALL 을 생략할 수 있다.
80 행에서는, 예로서 주 프로그램과 다른 변수명을 사용하고 있지만, 별도로 이유가 없는 한 변수명은 주 프로그램과 동일하게 하는 것이 좋다.

(iii) 인수가 있는 경우

```
1 !21_sub_program_with_arguments.bas
2 !
10      DIM A$(2)[20]
20      A$(1)="He"
30      A$(2)="llo"
40      P=7
50      Subit(A$(*),P,Q)                  ! call subprogram with arguments
60      PRINT Q
70      END                              ! end of main program
80      SUB Subit(C$(*),R,S)                ! definition of subprogram
90         PRINT "A$(*)=";C$(1)&C$(2),"P=";R
100         S=R*R                         ! return result
110      SUBEND
```

80 행에서는 변수의 늘어선 순서를 50 행과 같은 순서로 정리할 것. SUB 프로그램을 작성할 경우에는 주 프로그램 라인의 제일 마지막에 추가하지 않으면 안 된다. 80 행의 인수 C$(*) 를, 예를 들어 81 행에서, 10 행처럼 정의를 하려고 하면 에러가 난다.

(3) 서브 프로그램과 함수 2 (GOSUB, FN, CSUB)

(i) 주문맥 중의 서브 루틴(GOSUB)

HT-BASIC 에서는 주 프로그램 중에 서브 루틴을 둘 수도 있다. 이 경우, GOSUB 명령으로 작업의 흐름이 지정된 행으로 나뉘어져(분기돼), RETURN 문으로 GOSUB 명령의 다음의 행에서 되돌아오기 때문에, 작업의 흐름에는 주의를 하지 않으면 안 된다.

```
1 !22_subprogram_in_main_program.bas
2 !
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 행에서 60 행의 서브 루틴에 건너 뛰어, 70 행의 RETURN 문으로 되돌아오지만, 50 행의 STOP 명령이 없으면 재차 60~70 행을 실행하게 된다. <STOP>명령에서는 <END>와 마찬가지로, 프로그램은 idle 상태로 되돌아온다. <CONTINUE>(속행)는 할 수 없다.

(ii) 함수<DEF FN>

임의의 함수를 정의해서 프로그램중에 사용할 수 있다.

```
1 !23_subprogram_defined function.bas
2 !
10      PRINT "5 + 8 =";FNAdd(5,8)  ! quotation of defined function
20      END                       ! end of main program
30      DEF FNAdd(A,B)              ! definition of function
40         RETURN A+B
50      FNEND
```

30~50 행이 함수의 정의이다. 주 프로그램문의 다음에 정의를 부가한다. 함수명은 FN 부터 시작할 것. 이 경우, CALL 문은 필요없다.

(iii)컴파일을 끝낸 서브 프로그램(CSUB)

HT-BASIC 에는 많은 수학 함수가, 컴파일이 끝난 서브 프로그램으로서 준비되어 있다.
그 사용례를 이하에 나타낸다.

```
1 !24_subprogram_compiled mathematical library
2 !
10      LOADSUB ALL FROM "C:¥Program Files¥HTBwin¥mathlib¥Mathlib.hts"
20  !------------------
90      X=3
100      A=FNErf(X)
110      PRINT A
150  !------------------
200      END
```

100 행에서는 x 라는 인수의 함수치 FNErf(X)를 A 에 대입하고 있다. 함수 FNErf(X)는 오차 함수이다. 컴파일이 끝난 서브 프로그램은 디폴트의 디렉토리 안의 파일,

C:¥Program Files¥HTBwin¥mathlib¥Mathlib.hlp

에, 일람 리스트, 함수의 해설 및 사용할 때에 로드를 해두어야만 할 서브 루틴 파일명 등이 표시되어 있다.

그 중에서 특히, 상기의 프로그램 예의 10 행에 표시된,

C:¥Program files¥HTBwin¥mathlib¥Mathlib.hts

에는, 모든 수학 함수의 CSUB 프로그램을 포함하고 있으므로, 이것을 로드해 놓으면, 준비된 모든 수학 함수를 사용할 수 있다. 단, CSUB 서브 루틴이 말미에 부가된 프로그램은 ASCII 파일로서는 저장할 수 없다.

RE-STORE 명령으로 저장해, LOAD 명령으로 불러내지 않으면 안 된다.

(4) 행렬 계산 (MAT)

행렬 및 벡터(vector) 상호의 연산은 MAT 연산자에 의해 간단하게 실행할 수가 있다. 이하의 프로그램에서는 몇 갠가의 예를 나타내겠다.

```
1 !25_matrix calculation.bas
2 !
10      OPTION BASE 1                   ! minimum number of descriptor
20      DIM A(3,3),B(3,3),C(3,3)        ! definition of 3 X 3 matrices
30      DIM V1(3),V2(3)                 ! definition of 1 X 3 vectors
40      DATA 1,3,2,-1,-2,-1,2,4,3       ! data for matrix
50      RESTORE 40                      ! designate data read by next read command
60      READ A(*)                       ! read data into A(*)
70      CALL Subdisp1("A(*) = ",A(*))   ! display elements of A(*) using subprogram
80      MAT B=INV(A)                    ! inverse of A(*) into B(*)
90      Subdisp1("INV A(*) = ",B(*))    ! display elements of B(*)
100      MAT C=B*A                      ! multiplication B(*) and A(*) into C(*)
110      Subdisp1("INV(A)*A = ",C(*))   ! display elements of C(*)
120      MAT V1=B(1,*)                  ! substitute B(1,*) into V1(*)
130      Subdisp2("B(1,*) = ",V1(*))    ! display elements of V1(*)
140      MAT V2=V1*A                    ! multiplication V1(*) and A(*) into V2(*)
150      Subdisp2("B(1,*)*A = ",V2(*))  ! display elements of V2(*)
160      END                            ! end of main program
170      SUB Subdisp1(P$,Q(*))          ! definition of subprogram1
180         PRINT ""                    ! print of blank row
190         PRINT P$
200         FOR I=1 TO 3
210            PRINT USING "3(10D.DD)";Q(I,1),Q(I,2),Q(I,3)  ! formatted print
220         NEXT I
230      SUBEND
240      SUB Subdisp2(P$,Q(*))          ! definition of subprogram1
250         PRINT ""
260         PRINT P$
270         PRINT USING "3(10D.DD)";Q(*)! formatted print
280      SUBEND
```

40 행은 프로그램중에 데이터를 부여하는 DATA 문이다. 데이터는<,>로 단락을 짓는다.

50 행은 다음에 나오는 READ 문에서 읽어야만 할 DATA 문을 행(줄)번호로 지정하고 있다.

60 행에서는 행렬(배열) A(*)에 DATA 문이 읽혀진다. DATA 문으로부터 읽혀지는 순서는, 차원을 왼쪽부터 특정을 해, 제일 마지막 열에 대해서 순서대로 읽혀진다. 예를 들어, (P(2,3,2)이라는 배열에는,

P(1,1,1). P(1,1,2). P(1,2,*). P(1,3,*). P(2,1,*). ... P(2,3,*)

라는 순서로, 데이터가 읽혀진다.

70 행은 행렬 A(*)의 내용을 표시하기 위한 서브 프로그램을 불러내고 있다.

80 행은 A(*)의 역행렬을 B(*)에 대입해,100 행은 B(*)와 A(*)와의 적(면적)을 C(*)에 대입하고 있다. 보통의 대입문의 전에 MAT 를 부가하면, 행렬의 대입문이 된다.

90 행,110 행은 70 행과 동일한 서브 프로그램을 부르고 있다. 서브 프로그램을 부르기만 하는 행이라면, CALL 을 생략할 수 있다.

120 행은 B(1,*) 의 요소를 벡터(1 차원 배열) V1(*)에 대입하고 있다. 배열에서 배열로의 대입은, 요소의 수가 대응이 되게 하도록 주의할 필요가 있다.
130 행, 150 행은 벡터 요소를 표시하는 서브 프로그램을 부르고 있다.
140 행은 벡터 V1(*)와 행렬 A(*)과의 적(積)을 V2(*)에 대입하고 있다. 1 차원 배열 V1(*)은 왼쪽으로부터 걸 때에는 행 벡터로서, 오른쪽으로부터 걸 때에는 열 벡터로서 처리가 된다.

```
MAT V2=V1*A      ! V1(*) is row vector, the result is one dimensional array of row vector
MAT V2=A*V1      ! V1(*) is column vector, the result is one dimensional array of column vector
```

170~230 행은 행렬(배열)을 표시하는 서브 프로그램이며, 240~280 행은 벡터(1 차원 배열)를 표시하는 서브 프로그램이다. 양쪽 다 (P$,Q(*) 로, 문자열과 배열 데이터를 인수로서 불려 가고 있다.
210 행, 270 행은 서식이 붙어 있는 표시문이다.

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

<"3(10D.DD)">은 <정수부 10 자리, 소수점, 소수부 2 자리>에 의한 표시를 3 번 반복하는 것을 지시하고 있다. 표시하는 데이터는 <;> 의 다음에 열거한다. 270 행에서는 Q(*)로 데이터 열을 일괄해서 지시하고 있다. HP-BASIC 에서는 데이터를 주고받을 때에, 세세한 지시를 할 수 가 있다. 서식에 관해서는 레퍼렌스 매뉴얼의 <IMAGE>의 항에 자세하게 설명이 되어 있다. 또한, example 폴더에는 명령문을 사용하는 프로그램의 예를 들어 놓고 있다.

(5) 논리식

논리식의 판단은, true 의 경우 1 을 돌려주고, false 의 경우 0 을 돌려준다. 따라서 ,논리식을 포함한 수식을 만들 수가 있다. 예를 들어,

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

는 (A=B) 및 (C>D)가 양쪽 다 false 의 경우에는 0, 그 이외의 경우에는 P1(원주율)을 출력(표시, 인자(印字)) 한다.

```
1 !26_logical expression.bas
2 !
10      A=2
20      B=5
30      C=8
40      D=4
50      PRINT (A=B)
60      PRINT (C>D)
70      PRINT ((A=B) OR (C>D))*PI
80      PRINT SIN((((A=B) OR (C>D))*PI)/4)
90      END
```

(6) 비트 조작

BIT 명령 중의 5 는 비트의 위치를 나타내고 있다. 비트의 위치는, <OPTION BASE>의 지정에 상관없이, 최하위(오른쪽 끝)비트를 0, 최상위(왼쪽 끝)비트를 15 로서 지시한다. ROTATE 명령 중의 2 는 정(正)이라면 오른쪽(하위) 방향에 수치의 크기만큼 회전시킨다. 가장자리에서 밀려 나온 비트는 다른 가장자리로 되돌아 간다. SHIFT 명령중의 2 는 정이라면 오른쪽(하위)방향에

수치의 크기만큼 이동시킨다. 밀려나온 비트는 사라지고, 다른 가장자리에는 0 이 들어간다.

```
1 !27_binary manipulation.bas
2 !
10      A=237
20      B=6
30      !
40      PRINT "A="
50      PRINT IVAL$(A,2)
60      PRINT "B="
70      PRINT IVAL$(B,2)
80 !
90      A$=IVAL$(BINAND(A,B),2)        ! 16bit binary expression of integer
100      PRINT "BINAND(A,B)"
110      PRINT A$[1]                   ! 1st and later
120      !
130      A$=IVAL$(BINCMP(A),2)         ! binary expression of I
140      PRINT "BINCMP(A)"
150      PRINT A$[1]                   ! 1st and later
160 !
170      A$=IVAL$(BINEOR(A,B),2)       ! binary expression of I
180      PRINT "BINEOR(A,B)"
190      PRINT A$[1]                   ! 1st and later
200      !
210      A$=IVAL$(BINIOR(A,B),2)       ! binary expression of I
220      PRINT "BINIOR(A,B)"
230      PRINT A$[1]                   ! 1st and later
240      !
250      A$=IVAL$(BIT(A,5),2)          ! binary expression of I
260      PRINT "BIT(A,5)"
270      PRINT A$[1]                   ! 1st and later
280      !
290      A$=IVAL$(ROTATE(A,2),2)       ! binary expression of I
300      PRINT "ROTATE(A,2)"
310      PRINT A$[1]                   ! 1st and later
320      !
330      A$=IVAL$(ROTATE(A,-2),2)       ! binary expression of I
340      PRINT "ROTATE(A,-2)"
350      PRINT A$[1]                   ! 1st and later
360      !
370      A$=IVAL$(SHIFT(A,2),2)        ! binary expression of I
380      PRINT "SHIFT(A,2)"
390      PRINT A$[1]                   ! 1st and later
400 !
410      A$=IVAL$(SHIFT(A,-2),2)       ! binary expression of I
420      PRINT "SHIFT(A,-2)"
430      PRINT A$[1]                   ! 1st and later
440 !
450      END
```

16 비트 수로 표현된 수치의 비트 조작에 관해서는 이하의 명령이 있다.

```
BINAND(A,B)  ! take AND of two 16-bit count values per bit
BINCMP(A)    ! reverse each bit of a 16-bit count value
BINEOR(A,B)  ! take exclusive disjunction of two 16-bit count values per bit
BINIOR(A,B)  ! take OR of two 16-bit count values per bit
BIT(A,5)     ! return specified bit of a 16-bit count value
ROTATE(A,2)  ! rotate the bit sequence of a 16-bit count value in clockwise or
counterclockwise direction
SHIFT(A,2)   ! shift the bit sequence of a 16-bit count value to right or to left
```

(7) 복소수

복소수는 2 개의 실수(8 바이트)의 조(組)(16 바이트)로서 취급이 된다.

```
1 !28_complex number.bas
2 !
10 COMPLEX A,B          ! define complex variables
20   A=CMPLX(2,1)       ! A=2+i
30   B=CMPLX(5,1/5)*A   ! B=(5+i1/5)*(2+i)=9.8+i5.4
40   PRINT A
50   PRINT B
60 !
70 END
```

7. HP-BASIC 의 프로그래밍 4 (그래픽스)

HP-BASIC 은 다양한 그래픽스 명령을 갖고 있다. 이 장에서는 그래픽스에 관한 프로그래밍의 예를 들어 보겠다.

7.1 그래프 출력

(i) 함수의 그래프를 그린다.

이 절에서는, XY 좌표 평면에 함수의 그래프(그림 6)를 그리는 예를 들어, 그래픽스를 출력하는 작업의 흐름을 나타낸다.

```
1 !29_graphic output of a function-e.bas
2 !
10   GINIT     ! initialize parameter for graphical output
20   PLOTTER IS CRT,"INTERNAL" ! direct the output of graphics to display
30   Xlow=1    ! minimum value of X-axis of the graph to be output
40   Xhigh=1   ! maximum value of X-axis of the graph to be output
50   Xlength=Xhigh-Xlow  ! length of value on X-axis of the graph to be output
60   Ylow=-1   ! minimum value of Y-axis of the graph to be output
70   Yhigh=1   ! maximum value of Y-axis of the graph to be output
80   Ylength=Yhigh-Ylow  ! length of value on Y- axis of the graph to be output
90   VIEWPORT 40,120,20,90 ! specify range of the graphics on output device
100  WINDOW Xlow,Xhigh,Ylow,Yhigh  ! define user-specified value in range of the
graphics
110   FRAME    ! frame range of the graphics by solid line
120   AXES Xlength/20,Ylength/20,0,0,5,5,2  ! plot the axis of coordinate
130   CLIP OFF  ! cancel limit of the range of graphics
140   F$="F(X)=X*X*X"  ! assign character string
150   MOVE 0,Yhigh  ! move graphical pen to specified axis
160   LORG 4   ! specify label position corresponding to axis.
170   LABEL F$  ! output character string to label
180   CLIP ON   ! set limit of range of graphics
190   PEN 6    ! select pen
200   FOR X=Xlow TO Xhigh STEP Xlength/20  ! loop for plotting lines
210      PLOT X,X*X*X  ! plot lines
220   NEXT X   !
230   CSIZE 3  ! specify font size
240   LORG 1   ! specify label position corresponding to axis
250   PEN 2    ! select pen
260   FOR X=Xlow TO Yhigh STEP .5 ! loop for plotting scale of X-axis
270      MOVE X,0  ! move graphical pen to specified axis
280      LABEL X  ! output numerical label
290   NEXT X
300   FOR Y=Ylow TO Yhigh STEP .5 ! loop for plotting scale of Y-axis
310       MOVE 0,Y  ! move graphical pen to specified axis
320       LABEL USING "DD.D";Y ! output numerical label
330   NEXT Y
340  END
```

10 행은 묘화(描畵)출력을 위한 파라미터를 초기화하고 있다. 초기화된 파라미터는, 묘화펜의 종류, LABEL 로 그리는 문자의 크기, 좌표에 대한 문자의 배치 등이 포함되어 있다.

20 행은 묘화 출력을 디스플레이에 출력하도록 지정하고 있다. 이 문장 대신에,

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

라는 문장으로 대신하면, 묘화가 출력이 되는 곳은 GPIB 케이블과 접속이 된, 12 번이라는 주소를 갖는 프로터 (plotter)에 출력이 되게 된다. "HPGL(Hewlett Packard Graphic Language)은 HP 사가 개발한 그래픽스 출력을 위한 언어이다.

30~80 행에서는 그래프를 그리기 위한, X 와 Y 의 값의 영역 등을 임의의 변수로 대입하고 있다.

90 행에서는 PC 화면 등의 묘화 디바이스의 안에, 그래픽스를 출력하는 범위를 지정하고 있다. 즉, VIEWPORT 를 실행하면, 그 후의 묘화 명령문에 의한 출력은, VIEWPORT 로 지정한 범위에 한정이 되어 버린다. 범위의 지정은, 묘화 디바이스의 전화면의 Y 축(세로)방향을 100 눈금의 길이를 취해, X 축 (가로) 방향은, 각각의 디스플레이에 의존하는 길이를 취했을 때의 눈금치로 실행한다. 이 눈금을, HP- BASIC 에서는 GDU(Graphic Display Unit)라고 부르고 있다. 예를 들어, 노트 PC 의 전화면은 세로로 긴 화면이기 때문에, Y 축 방향이 100 이며, X 축 방향은 일반적으로 100 보다 큰 값으로 지정된다. 그것을 확인하기 위해서는,

```
RATIO
```

라는 명령을 실행하면 된다. <RATIO>는 묘화할 수 있는 범위의 X 축 방향의 길이와 Y 축 방향과의 비 (예를 들어 1.4 정도의 값)를 돌려 준다. 만약에, 노트 PC 의 화면의 RATIO 의 값(치)이 1.4 라면 묘화할 수 있는 범위는 Y 축 방향을 100 눈금, X 축 방향을 140 눈금으로 하는 범위이며:

```
VIEWPORT 가로방향의 왼쪽한계, 가로방향의 오른쪽한계, 가로방향의 밑쪽한계, 가로방향의 위쪽한계
```

로서 지정을 한다. 따라서, 90 행은 화면 전체속에서 세로 방향의 전 눈금수(대략 140 정도)중의 40~120 의 범위, 가로 방향의 전 눈금수 100 가운데 20~90 를 범위로 하는 장방형의 범위에 묘화를 할 것(그림을 그 릴 것)이라는 의미이다. 사전에 키보드에서 RATIO 명령을 실행에서 확인을 해 놓으면 좋다. 프로그램중에 RATIO 를 사용해서,

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

라고 하면,묘화 범위는 항상 X 방향의 30%에서 70%의 범위, Y 방향의 20 에서 80 의 범위가 된다.

100 행에서는, VIEWPORT 로 지정한 범위에, 묘화하는데 사용하는 좌표치를 정의하고 있다. 이 경우, 30~80 행에서 정의한 변수에 의해,

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

는, 왼쪽 가장자리가 -1, 오른쪽 가장자리가 1, 밑의 가장자리가 -1, 위의 가장자리가 1 로 정의된 것으로 된다.

110 행에서는 묘화 범위를 프레임(선)으로 둘러 싸고 있다.

120 행은 좌표축을 그리는 명령이다. 파라미터의 지정은,

```
AXES Xt, Yt, 0x, 0y, Cx, Cy, S
  and      Xt : distance between scales on X-axis
           Yt : distance between scales on Y-axis
           0x : value of X-axis when X intersects Y
           0y : value of Y-axis when Y intersects X
           Cx : distance between key scales on X-axis
           Cy : distance between key scales on Y-axis
           S  : length of mark of key scales by the unit used in VIEWPORT
```

이 된다.120 행은,

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

이 되어 있기 때문에,

```
<Xlength/20,Ylength/20,> ! label X-axis equally divided 1/20 of entire range of
X-axis direction and label Y-axis in the same way
<0,0,>  ! X and Y intersect at coordinate origin.
<5,5,>  ! key scale for both X and Y-axis appear at intervals of 5.
<2>     ! length of key scale is 2.
```

130~180 행은 묘화 오브젝트의 이름을 라벨로서 그리는 작업이다, 이전 장까지의 예에서는 문자나 수치를 출력하기 위한 PRINT 명령을 사용했다. 그러나, PRINT 는 워드프로세서로 문자를 출력하는 것과 같은 것으로, 괘선에 따른 정해진 장소 밖에 문자를 출력할 수 없다. 묘화 오브젝트에 맞춰서 임의의 장소에 문자, 숫자를 출력하려면 <LABEL>명령을 사용하지 않으면 안된다.
130 행에서는 90 행의 <VIEWPORT>명령으로 지정된 묘화 범위의 제한을 <CLIP OFF>명령으로 해제하고 있다.
140 행은 문자열의 대입
150 행에서는 펜의 위치를 지정한 좌표에 이동하고 있다. 이 경우, X 좌표가 0, Y 좌표가 Yhigh 가 되어 있다. Yhigh 라는 좌표는, 100 행에서 지정한 것과 같이 묘화 범위의 위의 가장자리에 해당한다. 그 주변에 문자를 쓰면, VIEWPORT 명령으로 지정된 범위에 관해서는, 문자를 그릴 수 있지만, 그 범위를 넘어 버리면 그릴 수가 없다. 그래서 130 행에 의해,일시적으로 묘화 범위를 해제했던 것이다.
160 행에서는, 150 행에서 이동한 펜의 위치가, 라벨로서 쓰는 문자열의 어느 위치에 해당하는지를 지정하고 있다. 숫자는 1~9 의 값을 취하고, 각각 다음과 같은 의미를 갖고 있다.

```
3 : upper-left (of character string)   6 : upper-middle       9 : upper-right
2 : left-middle                        5 : center-middle      8 : right-middle
1 : lower-left                         4 : lower-middle       7 : lower-right
```

따라서, 160 행의<LOG 4>라는 것은, 라벨 전체의 문자열의 중앙 밑쪽 가장자리를, 150 행에서 지정한 좌표치에 일치시키는 것이 된다.
170 행에서는 라벨을 출력하고 있다.
180 행은 라벨을 쓰기 위한 묘화 범위의 제한을 해제하고 있었기 때문에, 다시 <VIEWPORT> 명령의 범위로 제한하는 것이다.
190 행은 묘화를 하기 위한 펜의 색상을 지정하고 있다. 숫자는 0~15 를 취하고, 그 색상은:

```
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)
```

로 되어 있다. 묘화를 할 때에는 색상을 확인하는 것이 좋다.
200~220 행은 PLOT 명령에 의한 꺾인 선 그래프를 그리는 루프이다.
210 행은 지시된 좌표치(X,X*X*X) 까지 선을 그리는 명령이다.
230~290 행은 X 축에 눈금의 값을 그리는 루프이다.
230 행에서는 LABEL 명령으로 그리는 문자의 크기를 GDU 단위(80 행의 설명을 참조)로 지시한 것이다.
<CSIZE>명령은 2 개의 파라미터를 갖고 있고,

CSIZE 문자의 높이・문자의 폭/문자의 높이

로 되어 있다. 문자의 높이를 GDU 로 지시해, 문자의 높이에 대한 문자의 폭의 비율을 부여할 수가 있다. 문자의 높이만을 지시한 경우에는, 디폴트의 문자폭의 비율 0.6 이 사용된다.
240 행은, 150 행에서 설명한 것과 같이, 펜의 이동한 곳의 좌표가 문자열의 왼쪽 밑에 오도록 지시하고 있다.
250 행에서는 펜의 색상을 2(red)에 지시하고 있다.
260~290 행에서는 펜을 0.5 스텝으로 이동시키면서 X 축에 눈금의 값을 그리고 있다.
300~330 행에서는 펜을 0.5 스텝으로 이동시키면서 Y 축에 눈금의 값을 그리고 있다. 320 행의 서식이 붙어 있는 출력에 관해서는, 6 장(4)절, 210 행의 설명을 참조해 주기 바란다.
그림 6 은 그래픽스 출력의 예이다.

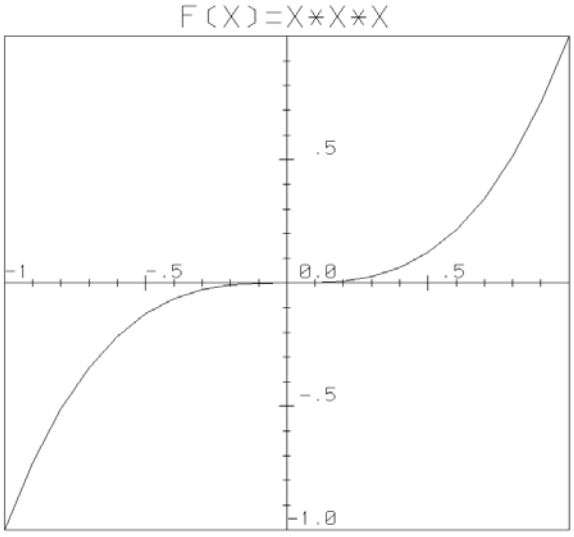


그림 6 그래픽스의 출력의 예

(2) 온라인 입력 데이터를 그래프로 표시한다.

inesDAQi250 용의 드라이버를 인스톨 해, 이하와 같은 프로그램을 입력한다. 이 프로그램에는 측정된 데이터를 그래프 표시하는 프로그램이 포함되어 있다.

```
1 !30_graphic output of on-line data.bas
2 !
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 행에서는 데이터 입력의 동작을 지시하고 있다. 프로그램의 예에서는 문자열이 길기 때문에, 2 줄로 나누어서 표시하고 있지만, 입력을 할 때에는 도중에 개행<ENTER>를 집어 넣어서는 안 된다.

90~120 행은 좌표축 묘화 등, 데이터를 표시시키기 위한 준비

130~190 행은 inesDAQi250 에 의한 애널로그 데이터를 읽는 것과 그래프 표시

200-220 행은 읽혀진 데이터의 수치 표시이다.

7.2 그래픽스의 출력과 저장

이전의 절에서 디스플레이에 출력한 그래픽스 화상을 저장하기에는. 3가지 방법이 있다.

(i)첫번째의 방법은 GPIB인터페이스를 개재해서, 프로터를 접속해, 직접 프로터에 그리는 방법이다. 이전의 절의 프로터의 예에서, 20행의 <PLOTTER IS CRT,"INTERNAL">라는 명령문을,

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

라고 고쳐 쓰면, GPIB 에 접속된, 12 번의 주소를 갖는 프로터에 "HPGL" 이라는 명령 코드로 출력이 된다. "HPGL" 은 HP 사가 그래픽스용으로 개발한 언어로서, 사용하는 프로터는 "HPGL"

언어 기능을 갖고 있는 것에 한정된다. 현재는 프로터가 별로 사용이 되지 않게 되었지만, HP 사의 프로터는 이 기능을 갖고 있었다. 판매중의 제품으로는, 그래프테크사의 MYPLOT 시리즈가 "HPGL"기능을 갖고 있다.
(ii)두번째의 방법은 범용의 프린터에 출력을 하는 방법이다. 이 경우도 프린터는 "HPGL"기능을 갖고 있는 것으로 제한된다.
(iii)세번째의 방법은 디스플레이에 출력된 묘화 비트 맵 화상파일로서 사용하는 방법이다. 프로터에 출력하는 경우에 비해, 디스플레이 화상의 질은 떨어지지만, 화상 파일로서 저장을 해 버리면, WINDOWS 상에서는 복사나 서류에 붙여넣기 등의 조작을 간단하게 실행할 수가 있다. 가장 간단한 방법은:

<Alt + Print Screen>
➔ < → Start Menu → Program→ Accessory → Paint >
➔ < → New document → Edit → Paste>

<ALT>키를 누르면서 키보드의 오른쪽 위에 배치되어 있는 <Print Screen>키를 누르면, 그 때의 액티브 윈도 화면이 복사가 된다. 다음에, <시작 메뉴 → 프로그램 → 보조프로그램 → 그림판>으로 WINDOWS 부속의 <그림판> 어플리케이션을 열어, < → 새로만들기 → 편집 → 붙여넣기>를 선택하면, 신규 작성화면이 자동적으로 조정이 돼. 그 때의 액티브 윈도 화면이 화상으로서 표시된다. 이 화상은 파일로서 저장을 할 수 있기 때문에, 나중에 jpg 파일로서 처리하거나, 워드 화면에 붙여넣기를 할 수가 있다. 그 이외에도 Adobe 화상 소프트 등에서 <파일 →새로 만들기>를 열어 백지 화면에 붙여넣기를 하거나, 또는 소스 넥스트사의 스구레모 3 이라는 소프트에는 화면을 찍는 툴이 있어서, 같은 작업을 행할 수가 있다. <Print Screen>키를 기동시키기 위해서는, 반드시 <Alt>키만으로 행한다고는 할 수 없으므로, 기종에 따라 테스트를 해 보는 것이 좋다. 프로터나 프린터에 출력을 해도, 그대로의 상태로는 화상을 서류에 붙여 넣을 수는 없으므로, (iii)의 방법이 편리하리라 생각한다.

7.3 HT-BASIC Plus

HT-BASIC 에는 각종의 GUI 가 준비되어 있어서, 프로그램에서 정의를 할 수가 있다. 원래는 HT-BASIC Plus 라는 이름으로 추가가 된 명령군이었지만, HT-BASIC for WINDOWS Ver9.x 에는 표준으로 포함이 되어 있다. 예를 들어, 프로그램 실행 중에, 변수의 상태를 그래프 등에서 확인을 하고 싶을 경우, 키보드에서 변수에 값을 부여하거나, 몇 갠가의 처리 중에서 적당한 처리를 선택을 하고 싶을 경우, 에러 발생에 처리를 지시하고 싶을 경우 등에 이용할 수 있다. 그러나, GUI 의 명령이 없어도 HT-BASIC 고유의 명령으로 같은 작업을 행할 수가 있기 때문에, 프로그램 실행상 불편함이 없다. HT-BASIC Plus 가 출현하기 이전부터의 사용자로, HT-BASIC Plus 를 사용해 본 적이 없는 기술자도 많다. 이하는 에러 발생시의 작업을 선택하는 다이얼로그(문의 화면)를 표시하는 프로그램의 예이다.

```
1 !31_HTBasic_Plus
2 !
10      MASS STORAGE IS "C:¥Program Files¥HTBwin¥ "
20      LOAD BIN "BPLUS"         !HP-BASIC Plus
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$(*)),TIMEOUT 5
90      SELECT Btn
100      CASE =-1               !Btn=-1
110         PRINT Btn
120         DISP "Timeout"
130      CASE =0
140         PRINT Btn
150         DISP S1$(Btn)
160      CASE =1
170         PRINT Btn
180         DISP S1$(Btn)
190      END SELECT
200      END
```

HT-BASIC Plus 를 사용하는 경우는, 10~20 행에서 나타낸 것과 같이. 반드시,

LOAD BIN “BPLUS”

을 실행해, 바이너리 프로그램을 읽혀 놓지 않으면 안 된다.

80 행에서 “ERROR”라는 다이얼로그를 설정하고 있다.

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

파라미터 중에 “ERROR”는 다이얼로그의 종류를 나타내고 있다. P$는 이 다이얼로그가 기동이 되었을 때에 나타나는 메시지로, 70 행에서 정의한 문자변수로 지정이 되어 있다. Btn 은 어떤 단추가 선택 되었는지에 따라 0,1,2..의 정수가 대입이 된다. SET( )의 안에서는, 표시하는 단추명을 문자 열배열로 지시하고 있다. TIMEOUT 5 는 타임아웃 시간이 5 초라는 의미이다.

이 프로그램을 실행하면 다이얼로그 화면에, “Abort” 및 “Continue”라는 2 개의 단추가 표시된다. 어느쪽인가를 선택하면, 왼쪽의 단추라면 0, 오른쪽이라면 1 의 숫자와 단추의 문자가 표시된다. 단추는 3 개 이상을 정의해도 된다. 아무것도 입력을 하지 않은 상태에서 5 초이상이 경과하면, -1 및 “Timeout”이라는 문자가 표시된다.

HT-BASIC Plus 에서는 10 종의 다이아그램(diagrams;문의 화면) 및 30 종의 위젯(widgets; 미터, 메뉴, 그래프 등의 표시 도구)이 준비되어 있다. HT-BASIC Plus 에서 사용할 수 있는 명령은 헬프 파일에 상세하게 설명이 되어 있으며, examples 이라는 이름의 폴더의 안에, 프로그램의 예가 수록이 되어 있다. 단, 위에서 설명한 것과 같이, 위젯의 안에는 설정 항목을 많이 포함한 것들이 있기 때문에, 7 장의 (1)절에서 예를 든 그래프 표시와 같이, 보통의 HT-BASIC 의 범위에서 그리는 것이, 프로그램으로서는 간단하다.

(i)Naming rule of Variable
Max 15charactors, includes numeric, alphabets, and <_>.
The first character must be alphabet and capital.
The last character of string variable must be <$>.
(ii)Kinds of Variable
- Numeric Variable :numeric (e.g.: Data, Name)
Real Precision: 8Bytes, +/- effective digit 15 +/- power 308
Maximum and minimum limit is refereed by instruction sets of <MAXREAL>,<MINREAL>.
Integer Precision: 2Bytes, -32768 ~ +32767
- Strings Variable: 1Bytes/1chractor, default setting is Max.18charactors.
Given character value is defined by DIM.
(example) Data$: Default setting is max18 characters.
- Simple Variable: Variable, which is not defined any dimension.
- Array Variable: Variable, which is not defined any dimension of array.
(iii)Rule of Variable Definition
- Numeric Variable
Real number: REAL X,A(5) : Definition of real number (REAL) can be omitted.
(example) INPUT X : X is defined as a real number.
DIM A(5) : 5 dimensions as a real number array
DIM Data(10:14) : 5 dimensions as a real number array , which are called as suffix 10 ~ 14.
Integral number: INTEGER Y,B(5) : integral simple variable, and integral array variable
Complex number is handled as a couple (16Bytes) of real number (8Bytes).
- String Variable:
(example) DIM Name$[20] : string variable and simple variable
INPUT B$: B$ is string variable, MAX 18 characters.
DIM Name$(5) :5pcs of string variable, MAX 18 characters.
DIM Name$(10:14)[20]: 20 characters array , which is called Suffix 10~15.
(iv)Definition of Array
Number of dimension is MAX 6.
Element of each dimension is MAX 32767.
Suffix of array: case of no assignment of suffix dimension,
DIM A(5) : 5 real number array
OPTION 0 : call 0 ~ 4 (default)
OPTION 1 : call 1 ~ 5
(v) Numeric Operator and Priority Sequence
(): bracket
FN: function
^: exponential
*, / , MOD , DIV: product, quotient, modular, division,
+, -: plus, minus
=, <, >, >=, <=, <>: relational operator
NOT: logical operator
AND: logical operator
OR, EXOR: logical operator
+, -, *, MOD, DIV: If both operator object is integer number, operator is integer arithmetic.
Real number of relational calculus:
DROUND(A,12)=DROUND(B,12) : half adjust by digit 12 .
ABS(A-B)<=1E-10 : Absolute value of deference is smaller than range of error.

9. 맺음말

현재의 노트 PC 는, 1)전자 문방구 또는 정보단말과 같은 이용 방법이 대부분이다. 원래의 목적이었던 2)계산기능조차도 기성의 프로그램에 맡기는 일이 많아졌디. 본론에서 설명을 한 것과 같은, 3)계측, 제어에의 이용이라는 것은, 이러한 정보단말, 계산기능과는 전혀 다른 차원에서의 계산기의 이용 방법이 있다. 정보단말로서의 이용에 익숙해 있는 사람이 계산기능을 이용하려고 하면, 이용하는 하드웨어로서는 컴퓨터 본체만 하더라도, 개념이나 용어가 틀리기 때문에 장벽에 부딪히고 만다. 또한, 정보단말이나 계산기능에 익숙한 사람이라 하더라도, 계측, 제어에 이용하려고 하면, 컴퓨터 본체의 조작 뿐만이 아니라. 센서, 측정기. 액츄어터 등의 하드웨어가 케이블을 통해서 접속이 되기 때문에, 그것들의 사용법에 익숙해 있지 않으면 커다란 장벽이 된다. 시판의 AD 컨버터 보드가, 이와 같은 내용의 번잡함에는 전혀 언급을 하지 않아도, 정해진 법칙대로 접속을 하면, 전용의 소프트로 데이터의 수록을 할 수 있게 설계가 되어 있는 것도 이유가 있다. 이러한 경향은 앞으로도 점점 활발해져 갈 것이라고 생각한다. 그러나. 연구자, 기술자, 학생의 레벨에서는, 자신의 발상대로 자유롭게 데이터를 얻어 처리를 할 수 있다는 시스템이 필요 불가결하다. 주어진 시스템만으로는 자유로운 발상을 살릴 수가 없으며, 외부에 맡긴다고 해도, 수주를 한 기술자가 그 연구분야에 대한 노하우가 없으면, 사용자는 자신이 희망하는 시스템을 만들 수가 없다.

이 리포트에서 설명한 HT-BASIC 은 이러한 3 종류의 이용방법을 상호간에 연결하는 all-in-one 타입의 언어시스템이다. 즉, (i)계측시스템을 구성해서 데이터를 파일에 수록한다. (ii)수록된 데이터를 파일에서 불러내어, HT-BASIC 또는 시판의 소프트웨어를 이용해서 수치계산이나 그래프화 등의 처리를 한다. (iii)결과의 수치를 그대로 이용하거나, 아니면 그래픽스 화면을 화상파일로서 저장해, 프로젠테이션에 이용한다. 이와 같은 일련의 작업을, 연필을 사용해서 종이에 쓰는 것과 같은 감각으로, 명령라인을 쓰는 일이 가능한 것이다. 이러한 작업은 물론 기존의 프로그램 언어에서도 할 수가 있다. 그러나, 열심히 하면[만들 수 있는 가능성이 있다]라든지, 특수한 훈련을 받은 사람만이[만들 수 있다]라는 것과, 일반의 연구자, 기술자 학생이 실제로 할 수 있다는 것과는 전혀 의미가 다르다. 보통은 현지조사, 실제의 장치의 조립, 문헌의 조사 등에 쫓기고 있는, 프로그램이나 계측시스템의 전문가도 아닌 연구자나 학생이, 갑자기 이용하고 싶은 경우에도, 보통의 논리를 더듬어 가는 것만으로, 무리 없이 사용할 수 있는 언어 시스템은 HT-BASIC 이외에는 없다.

이 리포트를 집필하는 과정에서 많은 대학, 기업, 연구소의 사람들과 과거, 현재, 미래의 계측의 수법에 관해서 많은 의견을 나누었다. 여러가지 의견이 있었지만, 공통된 것은, WINDOWS 의 등장 이후에 컴퓨터의 개념이 변해 버려서, 각자가 자기자신만의 독특한 수법을 찾아서, 만들고 있다는 것이었다. 물론, HP-BASIC 에 관해서도 알고 있는 세대이다. 그러나. 압도적으로 C 언어가 보급되어 있는 시대에 왜 B SIC 을? 이라는 의견도 많았다. 과거의 역사를 알고 있고, 경험이 있는 중년이상의 분들 중에는, 내용은 잘 몰라도 LabVIEW 로 충분히 데이터를 뽑을 수가 있고, 처리 소프트도 달려 있어서 편리하다고 말씀하시는 분들도 있었다. 한편, 젊은 사람들은, Lab VIEW 등의 기성의 소프트만의 환경에서 자랐기 때문에, 예를 들어 C 언어라 하더라도, 자신이 직접 실제로 커맨드라인을 써서 데이터를 수록한 경험이 없습

니다, 라고 말씀하시는 분들도 계셨다. 학생 제군의 세대에 공통되고 있는 것은, 소프트가 붙어 있는 시판의 계측 보드로 데이터를 수록하면, 그 처리나 표시는, 인터넷 등에서 취향의 소프트를 찾아내서 이용한다는 것이었다. 3.1 절에서 설명을 한 것과 같이, 시행착오 중에서 파라미터를 변경해서 RUN 키를 누르면 즉시 실행이 된다는 시스템이 아니면, 실험이나 연구의 현장에서 자유로운 발상을 살릴 수가 없다. 1977 년 8 월 23 일 새벽, Dr.Paul B.Mac Cready 등은 입력 비행기 Gossamer Conder 에 의한 인류 최초의 8 자 비행을 성공시켰다. 날개의 설계는 HP 사의 데스크탑 컴퓨터 HP9820A 를 구사해서 Dr.Peter Lissaman 에 의해 이루어졌다. 그는 낡은 이론과 (그들에 있어서), 낡은 계산기였음에도 불구하고, 성공을 한 이유로서는, 개조하기 쉬운 기체를 설계했다는 것과. HP9820A 에 의해 문득 생각난 힌트를 바로 확인을 할 수 있었다는 것을 들 수 있다[4][1]. 이 비행 실험의 기록 영화는 1978 년도의 다큐멘터리 부문의 아카데미 상을 수상했다. 그들은 2 년 후의 1979 년에 입력 비행기 Gossamer Albatross 에 의해 인류 최초로 영불 해협의 횡단에도 성공을 했다. HT-BASIC 은 설계 현장으로부터의 요망에 충분하게 응할 수 있을 뿐만 아니라, 컴파일을 갖고, 컴파일화 된 서브 루틴과 현재의 PC 의 성능의 향상도 있고 해서, 수치, 계산의 면에서도 상당한 일이 가능할 것이다. 또한 대학의 연구실에 있어서는, 선배의 프로그램을 개량해서 활용할 수도 있을 것이다.

이 리포트에서는, 이 상태로는, 적어도 일본 국내에서는 확실하게 잊어 버리게 되고 말 것이라고 생각되는 HT-BASIC 이라는 언어의 아우트라인을 설명했다. 실제로는 수치계산이나 논리판단 이외의, GPIB 시스템의 감시에 관한 수많은 명령, 입출력 제어 명령, 에러 메시지 등이 있어, HT-BASIC 을 상당히 사용한 사람이라도, 그 전모를 아는 사람은 적을 것이다. 그러나, 보통의 사용법이라면, 본문중의 예가 좋은 실마리가 될 것이다. 본문 중에서는 inesDAQi250 에 관해 상세하게 동작을 설명했지만, 궁리를 하면, GPIB 를 사용하지 않아도, 이 카드 1 장으로 상당한 데이터의 수록 시스템을 작성할 수 있기 때문이다.

서론에서도 설명을 한 것과 같이. 이것에 관한 모든 해설서가 현재에는, 영문의 온라인 매뉴얼로서 밖에 제공이 되고 있질 않다. 그 위에, 이러한 매뉴얼에서도, 전체상을 해설 한 것은 없다. 이 리포트가 얼마만이라도 그 역할을 할 수 있기를 기대하는 바이다.

참고문헌

1) TransEra (2002): TransEra HTBasic Installing and Using Manual.

2) TransEra (2002): TrandEra HTBasic Online Manual

3) ines (2001): inesDAQ Online Manual

4) Blair, John (1979): Keyboard, Jul-Aug, pp.14/16, Hewlett-Packard.

5) CQ Publishing (1996): Introduction of Instrumental and Control by PC, Transistor Gijutu special issue, No.53.

6) CQ Publishing (1999): Introduction of Instrumental and Control by Windows PC, Transistor Gijutu special issue, No.68.

7) Hirobumi Fujiwara (1994): C Programming Special Course, Gijutsu-Hyohron Co., Ltd.

8) Iwanami Shoten (1981): Science Journal(special), 51-10

요지

HT-BASIC 을 사용한 노트 PC 에 의한 계측, 제어시스템의 프로그램에 관해 자세하게 설명을 하고 있다. 현재, PC 를 이용한 설계, 계측 시스템으로서는, 시판의 애널로그 신호 변환보드에 부속이 되어 있는 소프트를 사용하거나, GUI 를 다용한 범용의 계측,제어소프트를 사용하는 것으로 나뉘어진다. 전자는 사용자가 독자의 변경을 하기 위한 자유도가 적고, 후자는 간편함을 추구한 나머지, GUI 의 편리함에 숨겨져 실제의 계측의 흐름이 보이지 않는다는 성질이 있다. 또한 C 언어를 기초로 두고 있는 시스템에서는, 얼마 정도 지나면, 설계자라 할지라도 프로그램의 흐름이 보이지 않게 된다는 본질적인 결함이 있다. HT-BASIC 은 전신을 HP 사의 HP-BASIC 에 두고 있는 계측, 제어시스템 구축을 위한 소프트웨어로, 작업의 흐름이 대단히 알기 쉬운 언어 구조를 갖고 있다. 특히 최근의 버전에서는 WINDOW 에 대응을 해서 사용하기 편리한 소프트로 개량이 되어 있다. 실험 장치의 조립, 실험의 수행과 데이터의 정리, 현지 조사, 문헌 조사 등, 평소에 프로그램만에 집중을 할 수 없는 학생, 연구자,기술자에 있어서, HT-BASIC 은, 노트 PC 레벨로 계측, 제어시스템을 자유롭게 짜기 위한 필수품이라고 말할 수 있다.

키워드: BASIC, 노트 PC, GPIB, 유저 지향, 계측