HEWLETT·PACKARD
JOURNAL

MAY 1986

HEWLETT-PACKARD JOURNAL

May 1986 Volume 37 • Number 5

Articles

Research Report

In this Issue

By now, there are probably close to a million personal computers in the hands of engineers and scientists, who are taking advantage of their PCs' low prices and extensive applications software to do everything from word processing to computer aided engineering. With that kind of installed base, it was inevitable that instrument control and automated testing would become available for PCs, and this has happened. Hewlett-Packard, of course, is a pioneer and a leader in the field of automated instrumentation and testing. The Hewlett-Packard Interface Bus, or HP-IB, has attained international standard status (IEEE 488, IEC 625) as a means of connecting instruments to computers. More recently, the Hewlett-Packard Interface Loop, or HP-IL, was developed to perform a similar function for battery-powered, portable instruments and peripherals. However, HP's first group of instruments designed specifically to work with PCs doesn't use either of these interfaces. The new product line, PC Instruments, has a new interface called the PCIB, or Personal Computer Interface Bus (see page 11). The reason is in the design of the instruments. One of the major design objectives was low cost, in keeping with one of the main reasons for the PC's attractiveness. Therefore, the new instruments have no displays, knobs, or controls. Instead, the PC screen displays their front panels and a touchscreen, mouse, or cursor is used to change settings. Updating the PC display quickly enough to show an oscilloscope trace in real time requires a high data rate. Also, some of the instruments need to be electrically isolated from the computer and from each other. The PCIB is a dual bus that provides either a high data rate or electrical isolation, depending on which is most important for a particular instrument. HP PC instruments come in low-cost plastic packages and when possible, use the computing power of the host PC instead of built-in microprocessors, so they're simpler and more reliable. One interface card plugged into the PC serves eight instruments. Special software (page 4) ties instrument control closely to the MS™-DOS operating system of the HP Vectra, HP 150, IBM PC, IBM PC/XT, and IBM PC/AT computers. Among the engineering contributions in the design of HP PC Instruments are a custom PCIB interface chip (page 14) and interactive graphics software (page 17). The article on page 29 compares a PC Instruments counter with an HP-IB counter.

There are, of course, applications for which the owner of a PC may want the typically higher performance of an HP-IB instrument, and it is available. There are HP-IB interface cards for PCs, and the article on page 27 describes an HP-IB command library for MS-DOS systems that makes programming HP-IB instruments as easy as programming PC Instruments.

The trend in VLSI (very large-scale integration) circuit design is up—more devices, working at higher speeds, in the same chip area. New processes and new materials are being sought to reduce parasitic impedances such as the sheet and contact resistances of the metal and polysilicon layers that interconnect the devices on a chip. On page 33, Jun Amano of HP Laboratories reports some results of research and process development on the use of titanium silicide for VLSI contacts and interconnections.

-R.P. Dolan

What's Ahead

Next month's issue will feature several articles about the technology behind HP's Doppler Ultrasound Imaging System for medical applications.

Also featured is an article about ICPL, a Lisp-embedded procedural layout language for VLSI design.

The HP Journal encourages technical discussion of the topics presented in recent articles and will publish letters expected to be of interest to our readers. Letters must be brief and are subject to editing. Letters should be addressed to: Editor, Hewlett-Packard Journal, 3000 Hanover Street, Palo Alto, CA 94304, U.S.A.

Low-Cost Automated Instruments for Personal Computers

Designed for the automated test and measurement requirements of a wide range of technical professionals, the components of this personal computer-based system include eight of the most widely used electronic instruments in modular, stackable cases.

by Charles J. Rothschild, 3rd, Robert C. Sismilich, and William T. Walker

HP PC INSTRUMENTS (Fig. 1) is a new line of low-cost programmable electronic instruments designed to be used in conjunction with HP's Vectra Personal Computer and the IBM PC/XT/AT computers. Tightly coupling these instruments to the computational and human interface resources of a personal computer allows Hewlett-Packard to provide programmable instruments at a price comparable to standard bench instruments, creating a new approach to low-end automation.

The PC Phenomenon

Making all of this happen is the extensive acceptance of the personal computer in the engineering and scientific markets. Numerous market studies have shown that there are more than 750,000 personal computers on engineers' and scientists' desks.

Personal CAT

HP's PC Instruments product line adds computer-aided testing (CAT) to a personal computer's repertoire. PC Instruments expands the benefits of CAT capabilities into applications that previously could not justify being automated because of cost and complexity. A major tactic used to reduce automation costs is the elimination of redundant components in the system. The personal computer system's keyboard, touchscreen or mouse, and CRT display serve as the human interface to a wide variety of instruments such as digitizing oscilloscopes, universal counters, digital multimeters, and function generators. Hardware front panels do not exist in this system, but have been replaced by front panels implemented on a personal computer's CRT screen using interactive graphics. These "soft" front panels eliminate the need for expensive hardware components

Fig. 1. *The HP PC Instruments system is designed to work in concert with the HP Vectra and HP 150 Computers and the IBM PC/XT/AT computers. Eight different instruments, offered in modular, stackable cases, and several software packages and accessories are available. A single interface card will support any combination of up to eight modules. More instruments can be added by using additional interface cards.*

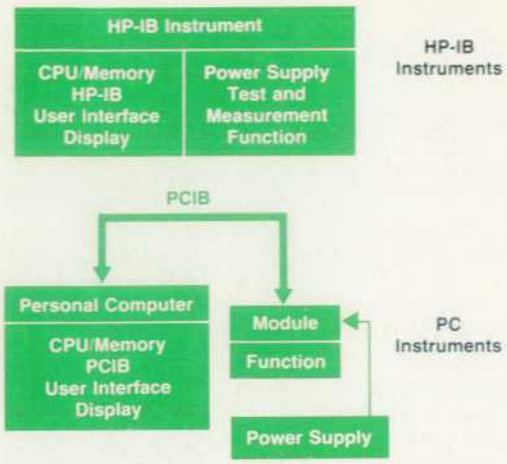

Fig. 2. *Difference in HP-IB and PCIB architectures.*

and provide a consistent user interface to all kinds of instrumentation. Fig. 2 is a comparison of a traditional HP-IB instrument with its PC Instruments equivalent. Note that only the instrument's measurement function remains in the PC Instruments module.

Making use of the computer's user interface saves more than just a few knobs, switches, and seven-segment displays. The traditional instrument has to process the raw measurement data into meaningful information for the instrument's display. This often requires a substantial microcomputer in the instrument. This added hardware requires a larger power supply, a larger cabinet, increased cooling, etc. The result is a more complicated instrument. By contrast, the PC Instruments approach can result in a very simple instrument design.

Even simple instruments such as a digital multimeter (DMM) can realize significant savings. For example, a traditional system DMM requires an inguard power supply for its front end, and a completely separate, electrically isolated, outguard power supply for its CPU and HP-IB interface. A PC Instruments DMM, because of its simplicity, requires only the inguard supply. The data is serialized and sent to the personal computer through an isolation circuit powered by the personal computer. The power supply is so simple that a calculator-type transformer supply for an ac line wall outlet is all that is required.

Programming in English

Another advantage of closely coupling the instruments to a personal computer is that much of the complexity of programming such instruments can be reduced, and an English-like, self-documenting syntax can be achieved. A loosely coupled HP-IB instrument is programmed by sending and receiving ASCII strings to and from the instrument's address. For example, to set an HP-IB DMM located at select code 7, bus address 23, to its dc volts mode and then take a measurement some time later requires programming statements such as:

```
OUTPUT 723,"F0"
    •
    •
    •
OUTPUT 723,"T"
ENTER 723,VOLTAGE
```

With PC Instruments software, the same sequence sent over HP's personal computer interface bus (PCIB) is intuitively more readable and understandable:

```
CALL SET.FUNCTION(MY.DMM,DCVOLTS)
    •
    •
    •
CALL MEASURE(MY.DMM,VOLTAGE)
```

System Hardware Architecture

The hardware components of the HP PC Instruments system are the instrument modules and the PCIB interface card for the personal computer. The instrument modules consist of two parts: the instrument front end and the system interface to the PCIB. The front end is just that—the circuitry necessary to acquire or generate the electronic signals and convert them to or from a raw digital form. The system interface sends or receives this raw digital data to or from the personal computer. PC Instruments currently supports eight types of instrument modules (see box on next page for an overview of the modules).

The computer's side of the PCIB is a custom card that plugs into an expansion slot of the computer. The article on page 11 provides a detailed description of the bus, PCIB interface card, and module system interface.

System Software

The tactic of achieving low-cost automation by removing redundant computational, control, and human interfacing components from the modules implies, of course, that these tasks must now be performed via software residing in the personal computer. This is both an advantage and a drawback. The drawback is that some of the measurement and control algorithms, such as those for an oscilloscope, are not simple, and require significant chunks of memory. In addition, the software is always dependent on the computer's particular operating system, microprocessor, and architecture. That is, you can't take the HP 61060AA system software disc for the HP 150 Touchscreen Computer and plug it into an HP Vectra Computer or IBM PC/AT, which use the HP 61061BA system software. These disadvantages, however, can be minimized by designing for software portability. And the system software can be designed so as to buffer the idiosyncrasies of the particular instruments from the user, and thus allow consistent user and programming interfaces that directly benefit the customer.

Instrument Drivers

The lowest level of software is the instrument driver, which implements the functionality of the instrument by translating user requests to hardware signals and hardware signals to measurement data. This software module, which is stored in the disc file PCIB.PLD, consists of a collection of routines, written in C, which implement the functionality of each instrument. For example, for the HP 61013A Digital Multimeter, there are routines for setting the mea-

(continued on page 7)

PC Instruments Modules

The introductory set of PC Instruments consists of the following eight modules:

- HP 61010A Digital I/O. This module has 16 independent input and output lines. These data lines can be addressed as variable-length words up to 16 bits long. The output bits can be programmed in a TTL or open-collector mode. The input threshold level can be programmed from 10V to −10V. There are also two data control lines for output and input. Random asynchronous and synchronous transfers are both available.
- HP 61011A Relay Multiplexer. This module has eight double-ended inputs multiplexed into a double-ended output. There is an on-board temperature reference that can be used for thermocouple measurements. The relays are bidirectional so that they can be also used to send one signal to eight points. The relays also feature break-before-make scanning.
- HP 61017A Relay Actuator. This module provides eight independent relay switches. Each channel can carry up to one ampere and switch 250V, dc or rms.
- HP 61012A Dual Voltage DAC. This module provides two independently controlled, isolated voltage sources. Each one has three output ranges: ±10V, ±5V, and ±1V. Each output source is electrically isolated from the others and from ground.
- HP 61013A Digital Multimeter. This DMM is a full 4½-digit dc and ac voltmeter and an ohmmeter. It is autoranging and has software calibration. There are four dc and ac ranges (true rms) and six ohms ranges. The reading speed is selectable at 2.5 or 12.5 readings per second.
- HP 61014A Function Generator. This module provides sine, square, or triangle wave outputs up to 5 MHz. By programming its duty cycle, it can also be made to generate pulses and ramps. Other programmable functions include frequency, amplitude, dc offset, and mode of operation. The modes include continuous, gated, or burst. In burst mode the module can generate bursts of one to 65,536 cycles. There are also inputs for amplitude and frequency modulation.
- HP 61015A Universal Counter. This module is a 100-MHz universal counter. There are frequency, period, and totalize modes for channel A. Channel B is provided for frequency ratio and time interval measurement modes. Autofrequency and autoperiod modes are also available. (See the article on page 29 for more information on the counter.)
- HP 61016A Digitizing Oscilloscope. This module is fully programmable and has such features as automatic scaling (autostop), autotrigger, self-calibration, and direct readout of delta voltage and time. Waveforms are captured using a random repetitive sampling technique. The scope has a 50-MHz bandwidth and a resolution of 0.67 mV. (See the boxes on pages 8 and 22 for additional details.)

PC Instruments DMM

Now, let's take a closer look at the design of one of the modules—the DMM. Fig. 1 is a simplified block diagram.

All communication with the host personal computer is controlled by the serial link microprocessor. The data is transferred serially through optoisolators. The serial link microprocessor also talks to the analog-to-digital (A-to-D) control microprocessor. The A-to-D control processor has a number of functions. It takes the control information from the host computer and sets the proper mode range latches. In the triggered mode, the serial link processor tells the A-to-D control processor when to take a reading. In the triggered and auto modes, it looks at the status of the A-to-D control processor and reads back the data when the conversion is complete. The nonvolatile memory stores calibration constants for all the functions and ranges. The control processor uses these calibration constants to correct for offset and gain errors before sending data back to the host computer.

The front end of the DMM has three sections (see Fig. 1). When K3 is closed, the DMM reads dc voltages. The range is controlled by the gain of the amplifier. Ac voltages are measured when K2 is closed via the rms-to-dc converter. The analog-to-digital converter (ADC) reads a dc level proportional to the true rms level of the ac input. For resistance measurements, K1 and K3 are closed. This forces a current through the unknown resistor and a dc voltage is read by the ADC. The voltage reference is used by the current source and the ADC to set the ranges of the DMM.

Allan Levine
Project Manager
New Jersey Division

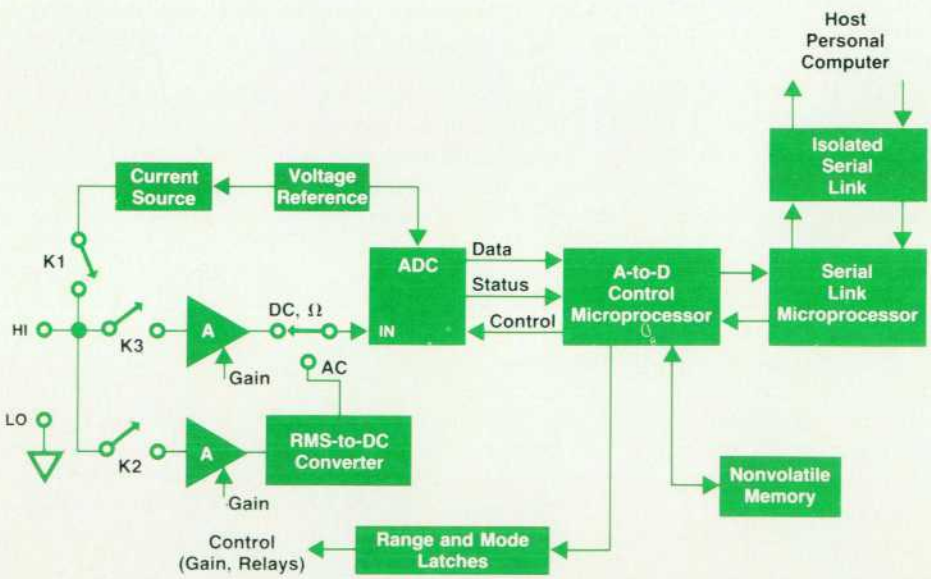

Fig. 1. *Simplified block diagram of the HP 61013A Digital Multimeter module for PC Instruments.*

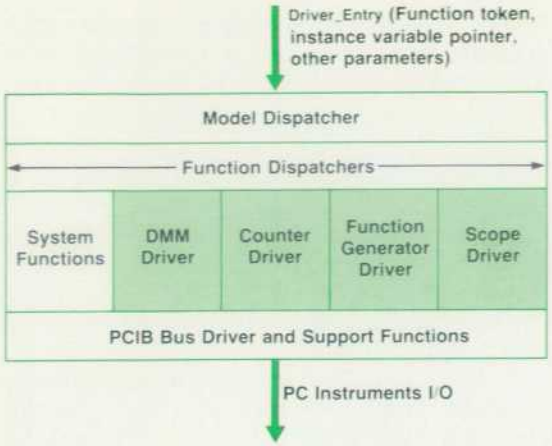

Fig. 3. *Instrument driver architecture. Shaded areas are instrument specific functions.*

surement function, selecting the range, and making the measurement.

The software architecture is object-oriented. In addition to measurement and control functions, every instrument supports a Define function which creates an instance variable (which may be thought of as a software image) of the specific instrument type. This instance variable contains information to describe completely the physical instrument it represents, such as interface and bus address, user-defined instrument name, and current instrument state.

As shown in Fig. 3, the instrument driver module has a single entry point. An example of a call to the module for a DMM measurement is:

```
Driver_entry(Measure_token,Instance_variable_ptr,V);
```

The first parameter is always a token representing the function to be executed. Tokens are not instrument specific. For example, counters, digital input modules, and DMMs can all make measurements. The second parameter is a pointer to the instance variable for the particular instrument to be accessed. A field of this variable binds the token to a particular type of instrument. The model dispatcher obtains the instrument type from the instance variable and then passes off to that instrument's function dispatcher, which invokes the requested function. Any additional parameters in the call, such as the variable V that will contain the measured value when the call is completed, are function specific.

The Instrument Drivers module is the only one that communicates directly with the PC Instruments hardware. The software modules for both the manual mode and the programmed mode of operation make calls as just discussed to the instrument driver module to perform I/O.

Language Cap

Interfacing the instrument drivers to the syntactical requirements of a specific programming language is the responsibility of the language cap. PC Instruments currently offers language caps for the GW™-BASIC interpreter and for ASYST™ Scientific Software (HP 14858A). Language caps can be developed for other languages as well without affecting the instrument driver module since the programming interface to the user is generic, taking the form of an action-oriented verb followed by a list of parameters.

The mechanics of the interface will vary from language to language. In GW-BASIC, for instance, PC Instruments makes use of the ability of the interpreter to call an assembly language routine located at an absolute address. For example, consider the statement:

```
1000 CALL MEASURE(MY.DMM,V)
```

In GW-BASIC, MEASURE is itself a BASIC variable which contains the address of a routine in the language cap that constructs the proper call to the instrument driver module. The BASIC variable MY.DMM contains the pointer to the instance variable for the instrument used to monitor the input voltage to the circuit under test. The BASIC variable V is used to return to the program the measured voltage value. The resultant call to the instrument driver for this statement would be as shown in the previous section.

A listing of a simple stimulus/response program written in GW-BASIC is shown in Fig. 4. It gives an idea of how such an instrument control language works out in practice.

ASYST is a trademark of Macmillan Software Company.

Instrumentless Front-Panel Program Demonstrates Product Concept

Sales of electronic instruments often require a hands-on demo to potential customers to illustrate features, specifications, and user interface. This is usually a requirement for the sale of new types of instruments or systems where the concept is not a familiar one.

PC Instruments modules embody several new concepts. Possibly the most novel one is the replacement of front-panel operating controls with software in the personal computer. Demonstrating the features and operation of this type of control mechanism is usually a requirement.

The PC Instruments Demo Disc for the HP Vectra Personal Computer and the IBM PC, PC/XT, and PC/AT was developed to address this need in the field. The disc contains system software with one modification: the PCIB bus driver that communicates with the instruments is replaced with a driver module that handshakes output data and returns preconfigured input data. An ASCII file on the disc specifies which types of instruments will appear, and at what address; this can be modified by the recipient of the demo disc, if desired.

The input data is dithered, to give the impression of slightly varying readings, and to indicate the update rate of the instruments on the display. All soft front-panel features work as in the real system software; the potential user can even generate the program shell that would be customized when writing an application program.

This disc is free upon request to anyone interested in learning more about PC Instruments.

Robert C. Sismilich
Project Manager
New Jersey Division

The language cap has the added responsibilities of converting the data formats of parameters between that of the programming language and that of the instrument drivers, and ensuring that parameter passing to the instrument drivers is done properly. To make the job of the GW-BASIC user easier, the language cap also automatically loads the instrument drivers into memory at run time, and can define the BASIC variables used as the PC Instruments verbs.

Versatile Microcomputer is Heart of PC Instruments Oscilloscope Module

The HP 61016A module is a medium-performance digitizing oscilloscope designed and manufactured by HP's Colorado Springs Division for the PC Instruments product line. The power and size constraints of the PC Instruments system presented a formidable design task. The design was made feasible by using a 6805R3 microcomputer as the heart of the module. This microcomputer contains many of the hardware functions required including an 8-bit analog-to-digital converter (ADC), an 8-bit counter, and a 4-MHz oscillator. The block diagram (Fig. 1) shows how the microcomputer fits into the design. There are three sections: the acquisition system, the computer system, and the power supply.

Acquisition System

The acquisition system consists of the vertical, sampler, trigger, time base, and interpolator. The vertical converts each of the two high-impedance inputs to low-impedance outputs. Each output drives a two-stage sampler. The sampler output provides a steady-state voltage to the microcomputer's ADC for conversion.

A sampling method called random repetitive sampling is used.[1] The time base clocks the sampler randomly with respect to the input signals. The microcomputer's oscillator provides the master clock to the time base. The trigger selects either input or the external trigger, and looks for a threshold crossing. When this happens, the time base will stop sampling after a programmed delay time. The interpolator measures the time between the trigger and the sample. This timing information is measured as a fine value by the ADC and a coarse value by the counter. The microcomputer uses this timing information to display each sample correctly in time.

Computer System

The computer system consists of the 6805R3 microcomputer, a serial latch chain, calibration digital-to-analog converters (DACs), a RAM, and the PCIB interface. The serial latch chain is 48 bits long and is used to program acquisition functions such as vertical range, trigger level, and sample rate. The calibration DACs adjust the offset and range of the ADC to compensate for errors in the acquisition system.

The PCIB interface uses a custom NMOS IC developed jointly by HP's Loveland Instrument Technology Center and New Jersey Division. It provides a high-speed, low-cost, 8-bit parallel bus between the oscilloscope module and the host personal computer. The RAM is used for storing waveform records and communicating with the PCIB. When the microcomputer is acquiring the input signals, the PCIB does not have access to the RAM. When the PCIB is ready to transfer data, it interrupts the 6805R3. The 6805R3 then stops acquiring data and grants the PCIB access to the RAM.

The PCIB programs the oscilloscope module by storing 12 bytes, called tasks, in the RAM. The PCIB then gives up access to the RAM. The microcomputer reads these bytes, configures the acquisition system, and starts acquiring data.

The microcomputer ROM contains the calibration, setup, and acquisition routines. Because of limited ROM space, a downloading mechanism was devised. The PCIB can download routines such as autoscope and production tests into the module's RAM. The 6805R3 then transfers these routines into its internal RAM for execution.

Power Supply

The power supply rectifies, filters, and regulates secondary ac power from the external PC Instruments power pack. The total power consumption of the acquisition and computer systems is approximately six watts.

Reference

1. K. Rush and D.J. Oldfield, "A Data Acquisition System for a 1-GHz Digitizing Oscilloscope," *Hewlett-Packard Journal*, Vol. 37, no. 4, April 1986.

Dennis J. Weller
Development Engineer
Colorado Springs Division

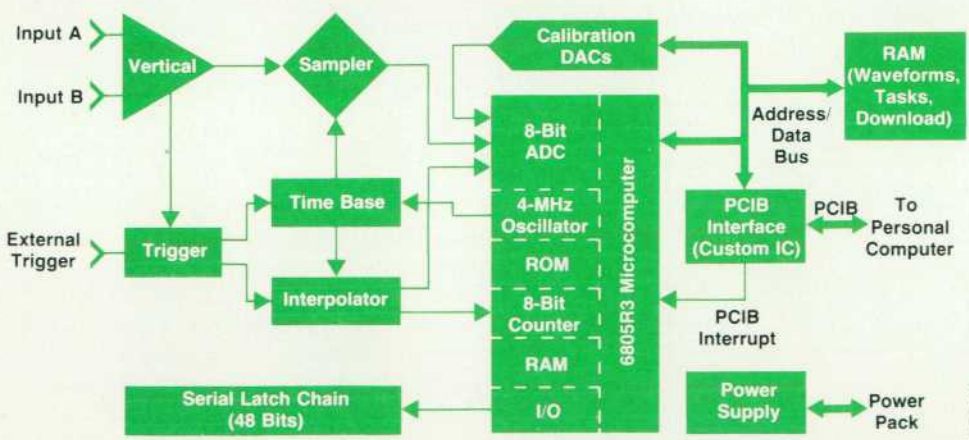

Fig. 1. *Simplified block diagram of the HP 61016A Digitizing Oscilloscope module for PC Instruments.*

```
1000 REM Program shell (lines 1 through 999) initializes
1001 REM verbs and creates instance variables for
1002 REM instruments called BIAS.VOLTS (an HP 61012A Dual DAC)
1003 REM and DEVICE.TEMP (an HP 61013A DMM).

1005 REM Initialize instruments as prestored in file STATEFILE.HPC
1010 A$="A:\STATEFILE.HPC"
1020 CALL INITIALIZE.SYSTEM(A$)

1025 REM Enable instrument outputs
1030 CALL ENABLE.SYSTEM
1040 DIM ARRAY(100,2)

2000 REM Perform stimulus/response test
2010 STIMULUS=.1
2020 FOR I%=1 TO 100
2030 ARRAY(I%,1)=STIMULUS*I%
2040 CALL OUTPUT(BIAS.VOLTS,ARRAY(I%,1))
2050 CALL MEASURE(DEVICE.TEMP,ARRAY(I%,2))
2060 NEXT I%

3000 REM Write the datafile
3010 OPEN "O",#1,"A:\DATAFILE.PRN"
3020 FOR I%=1 TO 100
3030 PRINT #1,ARRAY(I%,1),ARRAY(I%,2)
3040 NEXT I%
3050 CLOSE #1
4000 END
```

Fig. 4. *Listing of simple stimulus/response program using PC Instruments software.*

Soft Front Panels

Manual instrument control is performed with the PANELS applications package (see the article on page 17). PANELS can be used either as a stand-alone program, or in conjunction with GW-BASIC as a program development and debug tool.

An Automation Paradigm

These system software modules, then, provide the framework for the following automation paradigm, whose various elements are illustrated in Fig. 5. First, the instruments on the bus are manually configured using PANELS.EXE, the soft front-panel program. The instrument setup is saved to disc in a state file, and linkages to the program library for the given set of instruments are saved to disc in a program shell file. Second, stimulus and response or data logging test sequences are programmed from within GW-BASIC. The program shell file is the starting basis of the GW-BASIC application. The program shell can access the PC Instruments programming library and the HP-IB Command Library, HP 61062AA/BA. In addition, PANELS.EXE can be called from GW-BASIC for debugging. Third, test results can be stored to disc and easily ported into graphics software for analysis and plotting.

Stimulus and response data gathered by PC Instruments modules can be loaded into a Lotus™ 1-2-3™ spreadsheet, and then plotted with the Lotus PrintGraph utility. Other data presentation and analysis alternatives that use standard data interchange formats can be used as well, such as ASYST. A utility program, CONVERT.EXE, is provided with the system software which converts stripped ASCII, BASIC, or DIF (data interchange format) files into stripped ASCII, BASIC, or DIF files for use by other popular software.

For customers with standard data acquisition problems, such as scanning a few channels of thermocouples or other transducers and keeping a strip chart of the results, a packaged solution is available. The HP 14855A and HP 14856A PC Instruments Data Acquisition Software provides instrument control, data management, and graphics software. Because most users require some customization, the package is written in GW-BASIC and can be modified by the customer.

Acknowledgments

On a project this large it would take a document the length of this article to list all of the contributors. Technical contribution was not limited to just R&D, nor was it limited to one division of HP. The project was successfully completed because of the efforts of people in marketing, manufacturing, quality assurance, and R&D at divisions throughout the company. These divisions included Colorado Springs, the Loveland Instrument Technology Center, the Instruments Systems Laboratory, HP Laboratories, the Roseville Terminal Division, and of course, the New Jersey Division.

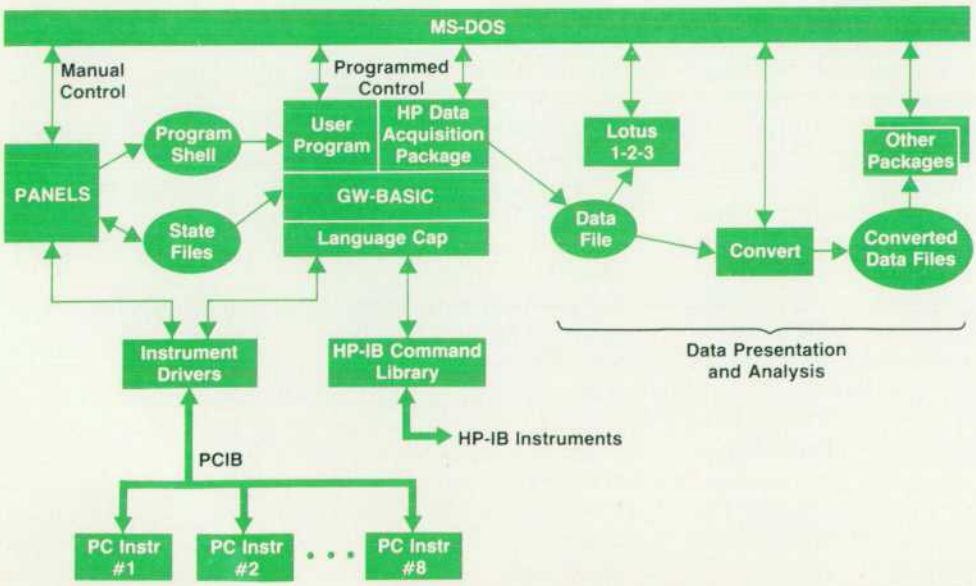

Fig. 5. *Overall software system architecture.*

Mechanical and Industrial Design of the PC Instruments Cabinet

The PC Instruments cabinet concept (Fig. 1) uses an injection molding process to mold many details into one part, thereby lowering the part count and cost. The manufacturing process is designed for progressive assembly—everything attaches to the base. The module's printed circuit board and a rear panel or heat sink snap into the base. No tools or fasteners are used. The printed circuit board outline is standard. All instrument modules use this outline, which makes our manufacturing process and assembly easy, cost-effective, and consistent. A second board can be placed into the base assembly if the instrument warrants a two-board design. The HP logo, an LED light pipe block, and the individual instrument's front-panel plate adhere to the top cover. The cover then slides over the instrument's front connectors and snaps into the base at the front and rear. By designing commonality into many parts, we were able to reduce manufacturing time and increase the volumes of these common parts, thus lowering the overall costs.

To design a homogeneous look for the personal computer environment where these instruments will be placed, we needed to combine different attributes of HP's instruments and computer cabinets. The units will stack on each other and on other HP instruments while blending aesthetically with HP computer products. The distinctive front grille design not only gives the units a family look, but also allows for needed convection cooling. Air entering the front and bottom sides is heated, forms a draft, and exits out the rear top. The top vent detail is similar to that used for the HP 150 Computer and HP printer products.

The cabinet, which is 64.5 mm high, 212 mm wide (half rack size), and 270 mm deep, does not have provisions for racking. If a customer wants to rack PC Instrument modules, an optional shelf will house four instruments and their accompanying power supplies. Thus, the additional cost for racking is paid only by customers requesting this feature.

George Kononenko
Development Engineer
David Schlesinger
Industrial Design Manager
New Jersey Division

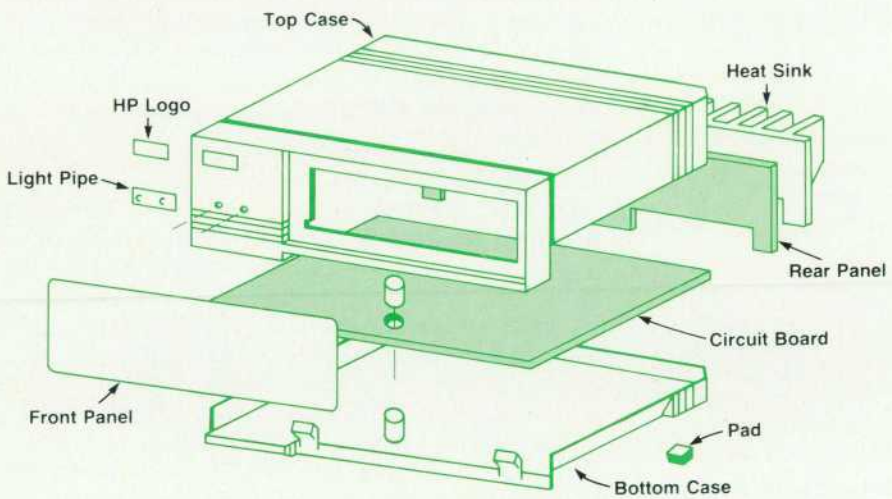

Fig. 1. *Exploded view of PC Instruments cabinet (single height, half rack).*

The acknowledgment section of each of the accompanying articles lists the key people working on that portion of the project effort. But there are others to credit for the extra effort they put in to get PC Instruments to market. In product marketing, those who worked on PC Instruments include Max Trescott, Mary Nee, Tom DiCorcia, Rick Van Ness, and Bruno Codispoti. Additional marketing help came from Rich Tomasetti and his technical writers, the product support group consisting of Sheri Surchek, Bob Zollo, and Emidio Cianfaglione, and Charlie Thompson's application software group including Walt Syzonenko, Mark Harding, Chris Fullam, Mary Seger, and Ray Puszkarczuk. Thanks also go to Warren Schmidt and his environmental group and Win Seipel and his model shop. Strong support from manufacturing came from John Farrell, Ed Gilbert, Marlene Chenowyth, Bill Pickel, Bob Harrison, and Kawing Kwan. In addition to his manufacturing duties, Ed Gilbert also wrote an exceptionally fast and powerful MS-DOS text editor that contributed to the productivity of all the software designers.

The design of the system components was the key to making PC Instruments low-cost. Yefim Kaushansky, George Kononenko, and Ken Woolley were the engineers involved in this effort. Credit should also go to Jerry Nelson, who was the electrical engineer in charge of the PC Instruments power supplies. The great number of printed circuit board designs was handled expertly by Dave Escalante, Nancy Brady, and Dan Mott.

We would also like to thank the module designers: Irwin Cohen and Naum Schneyder who designed the HP 61013A DMM, Bob Bland for the HP 61014A Function Generator, Bob Young on the HP61012A Dual DAC, Dennis Weller for the HP 61016A Oscilloscope, Bob Miller who designed the HP 61015A Counter, and Mike Mazewski for the HP 61011A Relay Multiplexer, HP 61017A Relay Actuator, and HP 61010A Digital I/O. Rich Comins was the chief architect for the modules, coordinating the designs of all the modules to ensure consistency. We would also like to thank the

hardware technicians: Mike Alden, Dan Jaeger, Eugene Micek, Carlos Nadal, and George Metz, who went well beyond the call of duty to ensure the successful completion of the project.

PCIB: A Low-Cost, Flexible Instrument Control Interface for Personal Computers

by William L. Hughes and Kent W. Luehman

THE CHOICE OF THE INTERFACE for HP's PC Instruments product line was very important in the realization of the system objective of significantly lowering the cost of automated applications. Selecting such an interface requires balancing a number of conflicting objectives such as high speed, low cost, and low power. This article discusses the goals for the PCIB interface, compares it with other interfaces, and describes its theory of operation.

System Objectives

The personal computer is proving to be a capable yet economical instrument controller. To make similar price/performance gains for the PC Instruments product line necessitated the following objectives for its interface:

- Provide a reduction of greater than 50% in both part count and cost when compared to traditional approaches.
- Use a low-cost, unshielded cabling scheme that can be easily configured by end users, yet meet HP and regulatory agency standards for EMC (electromagnetic compatibility) performance.
- Have a maximum interface data transfer rate of greater than 100,000 bytes per second so that the personal computer display can be updated at a reasonable rate.
- Provide a low-cost isolation scheme that allows PC Instruments products to float at line voltage potentials and still meet both computer and instrumentation safety standards.
- Adhere to a maximum interface power budget of one watt. Lower power means lower enclosure costs through reduced size and the elimination of the need for cooling fans.
- Support eight instruments that have automatic identification capability in a system with a single interface card.

Interface Selection

Existing interface standards that were considered for PC Instruments included RS-232-C/V.24, HP-IB (IEEE 488/IEC 625), and HP-IL (Hewlett-Packard Interface Loop). The PC Instruments system contains a wide variety of products such as the isolated, low-speed HP 61013A DMM and the higher-speed unisolated HP 61016A Oscilloscope. All of the instruments have an emphasis on low cost and low power. Could one interface satisfy all of the objectives of the system?

RS-232-C and HP-IB interfaces use expensive, large multiple-conductor cables and connectors. Both interfaces use bipolar technology that, because of the drive current requirements, consumes a considerable amount of power. Isolation is difficult and expensive, because of the large number of lines that have to be connected. This typically is not a prob-

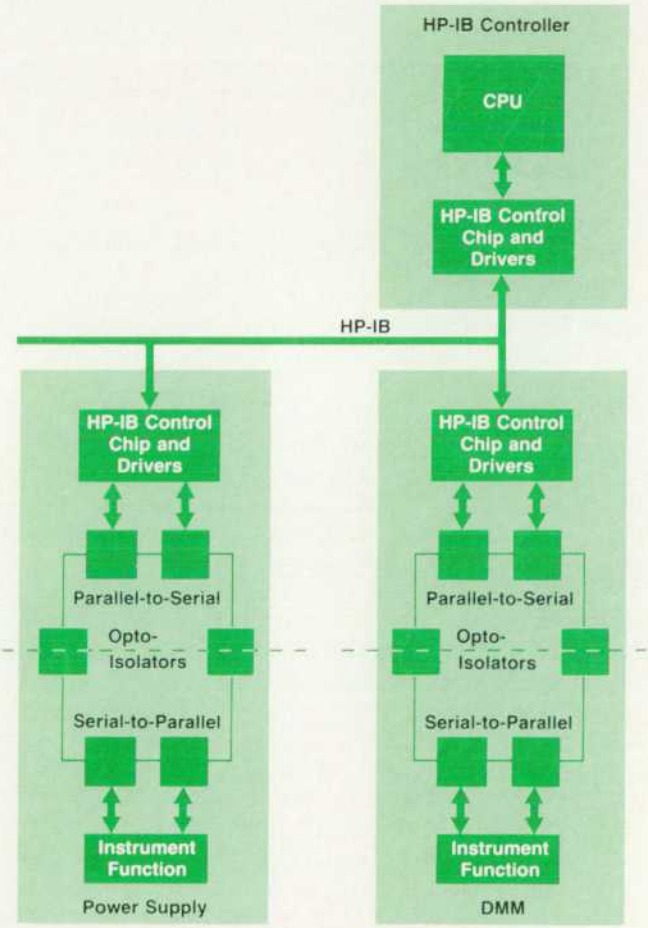

Fig. 1. *Typical HP-IB system with isolated instruments.*

lem where the interface electronics costs are a small portion of the total system price, as is the case for higher-performance, higher-priced instruments, where the HP-IB excels as the dominant standard. But a PC Instrument product would be greatly impacted both in size and in cost if it had to use one of these interfaces.

HP-IL is a low-cost serial interface that has been optimized for portable instrumentation.[1] It is an ideal interface for products that require low power and isolation. However, its data transfer rate cannot approach the 100,000 bytes/s desired for PC Instruments products.

The diversity of PC Instruments products with their conflicting requirements led us to the development of the Personal Computer Instrument Bus, abbreviated PCIB. PCIB is a hybrid interface consisting of two independent communication channels: a parallel channel optimized for high-speed instruments and a serial channel optimized for isolated instruments. Both channels have been designed for low power and low cost. Although the two channels are functionally independent, the communication protocols used are the same. They also present a similar interface from the system software and instrument architecture points of view. The communication channel used by a particular instrument is totally transparent to the user.

At first glance, the idea of having two communication channels in a single bus system appears to be redundant. On closer analysis, however, a cost saving to the user is realized and the redundancy is reduced. In a typical HP-IB system there are instruments that require the inputs and/or outputs to be isolated from earth ground and the computer safety common so that floating measurements can be made. The isolation frequently is provided by optoisolators communicating the data from the HP-IB side to the measurement side in a serial fashion (Fig. 1). The parallel data received from the HP-IB is serialized in a parallel-to-serial converter, sent through the optoisolators, and converted back to a parallel format. In PC Instruments, this converter is on the interface card, so the user pays for it only once rather than with every isolated instrument purchased.

System Description

A PC Instruments system can have up to eight instrument modules connected through the PCIB to the host personal computer. The instruments can be located up to four meters from the computer. A PCIB interface card installed in the computer performs the required translation from the computer's backplane to PCIB signals. Any mix of PC Instruments modules can be connected to the system. Additional increments of eight instrument modules can be added by installing more interface cards.

When setting out to create an interface definition for a new system, many factors have to be considered. Among the most important are how the interface will be used and how it must interact with all the devices connected to it. With this knowledge, an approach to the overall system architecture and performance requirements can be developed. Knowing in advance the types of instruments to be used on the PCIB, and the performance levels required, allowed a straightforward register-oriented architecture to be selected. Each function and data location has an individual register associated with it. Each instrument can have up to 16 directly addressable write registers and 16 directly addressable read registers. This has proven to be a sufficient number for all instruments considered for the PCIB. However, if required, expansion to a greater number of registers is easily handled through an indirect addressing scheme. System software relieves the user of having to learn the details normally required with a register-oriented system. In addition, PCIB uses a very simple communications protocol with only three message types: command, address, and data.

A register-per-function system does not require on-board processing of sophisticated codes and formats to allow the instrument to understand what actions are requested of it. The instrument just accepts data from the bus and directs it to the specified register. The action taken depends upon the register the data is placed into. There are no mnemonics to be translated as in HP-IB or HP-IL systems. The purpose of mnemonics in those interfaces is to make it easier for the human operators and programmers to understand the communications occurring over the bus. With PC Instruments, the human interface is handled at the system software level, relieving the instrument and operator of the burden of interpreting mnemonics. This also permits a standard approach to the programming commands.

Command messages are used for standard instrument operations such as initialize, enable output, and disable output. All commands are a single byte and can be directed to all instruments at once (universal commands) or to an individual instrument (selected commands). Universal commands provide a means of performing an action quickly on all instruments simultaneously, such as "disable outputs." Selected commands implement the same functions but act only on a single instrument, allowing individual control. There are sixteen possible commands available; at present, not all have been defined.

Address messages are used to select the instrument with which subsequent data operations will be performed. All instruments have listen and talk addresses. The listen address is used when the personal computer sends data to

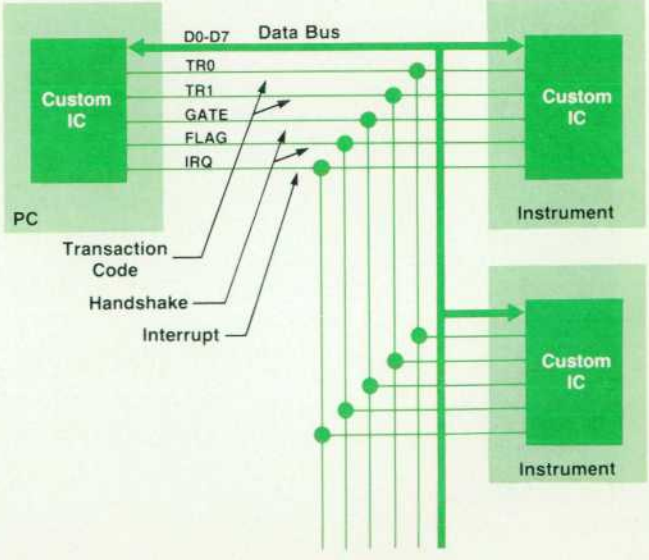

Fig. 2. *PCIB parallel communications channel.*

the instrument. The talk address is used when the computer wishes to read data from the instrument. Addresses are a single byte and include the instrument address as well as the register to be used for the data transfer. Only one instrument on the bus may be addressed at a time. The host personal computer is always either the source or the destination of all data transfers.

Data transfers are also performed one byte at a time to or from the selected register. Data is generally transferred in a form that is used directly by the instrument; it is not translated to and from ASCII. This keeps the number of bytes communicated over the bus smaller and reduces the overhead in each instrument. High-speed, multiple-byte transfers are possible using the parallel communications channel.

Parallel Communications Channel

The PCIB parallel communications channel (Fig. 2) offers a high-speed data path for instruments that do not need isolation. Data can be transmitted at rates up to 100K bytes/s, subject to limitations of the host personal computer. Special output drivers built into a custom IC have limited rise and fall times to help meet EMC goals. The custom IC also implements most of the protocol, register decoding, and system commands for the instruments.

The parallel communication channel consists of an 8-bit data path, two control lines, two handshake lines, and an interrupt request line. The 13 signal lines, with appropriate ground returns, are part of the 26-conductor ribbon cable that connects the host personal computer to the instruments.

Two signals make up the handshake group—GATE and FLAG. GATE is used to indicate when data placed on the bus by the host computer is valid. During command, address, and output data operations, it is used by the instrument to strobe the data byte off the bus. During input operations, it is used to strobe the data out of the instrument's internal register onto the bus. FLAG is generated by the instrument in response to GATE generated by the personal computer. During command, address, and output data operations, FLAG indicates that the instrument has received the transaction and has finished accepting it. FLAG is also used during input data operations to indicate that the data placed on the bus by the instrument is now valid and can

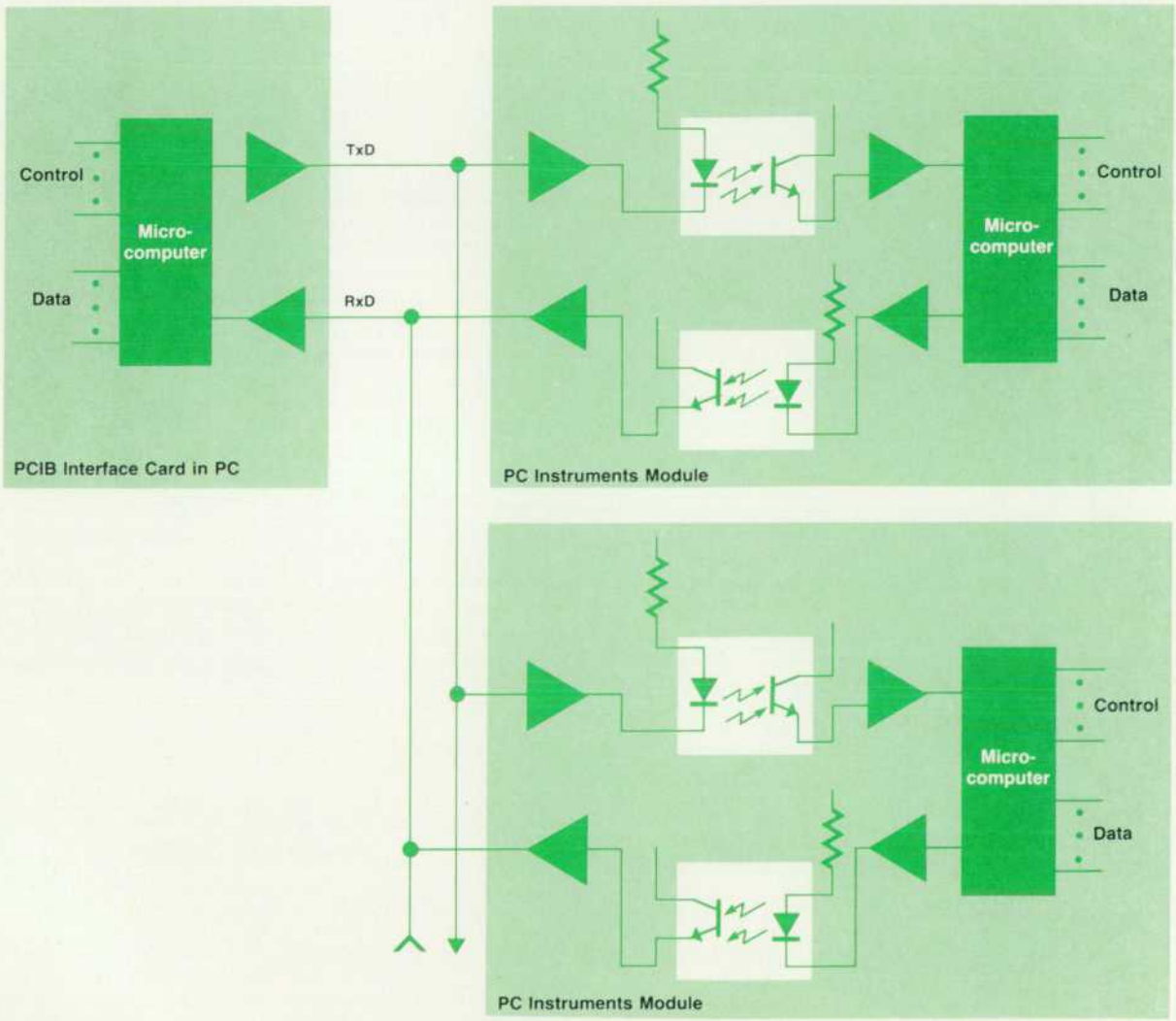

Fig. 3. *PCIB serial communications channel.*

be accepted by the personal computer.

The two control lines TR0 and TR1 contain the transaction code for the current bus operation. As discussed previously, this information determines how the data on the data bus is to be interpreted by the instruments. Of the four possible transaction types, only three are presently implemented:

TR1	TR0	Definition
0	0	Reserved for future expansion
0	1	System command
1	0	Instrument address
1	1	Data byte

The transaction code is only valid during the time GATE is asserted. Note that instruments do not respond in any fashion, not even with a handshake, to a reserved transaction. To respond would compromise the possible future use of the transaction type.

The PC Instruments parallel data bus is an 8-bit, bidirectional bus that is used to transmit and receive data to and from the instruments. Data from the host personal computer is valid on the bus only when the GATE handshake signal is asserted. Data from the instruments is valid only when the GATE and FLAG handshake signals are asserted. In the idle state the bus direction is from the host computer to the instruments. The output drivers on the bus are specially designed with limited slew rate outputs to help eliminate RFI (radio-frequency interference) from the unshielded ribbon cable and to prevent crosstalk. The input receivers have built-in hysteresis to help prevent false inputs caused by line noise and reflections.

An interrupt request signal, IRQ, is also included in the PCIB. It can be used by the instruments to indicate that a condition has occurred that requires the attention of the host personal computer. These conditions can be as simple as "ready for the next data byte" or they may indicate that a fault has occurred in the instrument. The exact nature of the request is instrument dependent. The IRQ line is low true, allowing a wired-OR to be implemented. The fact that an interrupt is being requested is determined by polling

A Custom HQMOS Bus Interface IC

Use of an inexpensive unshielded ribbon cable for HP's Personal Computer Instrument Bus (PCIB) necessitates an unusual custom solution to the problems of radio-frequency interference and signal crosstalk. A custom bus interface chip minimizes these effects and contributes communication protocol hardware for an instrument on the parallel PCIB channel. The integrated circuit provides bidirectional transceivers to both controller and instrument sides of the parallel part of the bus.

The exposed ribbon-cable transmission environment presents a special challenge. The driving circuits must slew predictably within a narrowly specified range, despite wide variations in load capacitance and other physical parameters. The range of acceptable rise and fall times is constrained at both sides; the maximum data rate fixes the slowest specification, while strict HP Class B environmental requirements dictate the fastest allowable slope. The choice of NMOS technology over CMOS prevents any possibility of latchup.

The resulting custom integrated circuit resides in a low-cost, 48-pin plastic dual in-line package. The chip dissipates approximately 0.33 watt and requires only a 5V supply, using an internal negative substrate bias voltage generator.

Operation Modes

This IC implements three modes of operation: personal computer passthrough, pod protocol, and test modes. The passthrough mode, used on the controller side of the parallel channel of the PCIB, configures the chip as a set of buffered, transparent bus transceivers. In the passthrough mode, a single input line controls the interface direction. The pod protocol mode invokes all logic necessary for a parallel PC Instruments module to communicate on the bus. Functions of the pod protocol mode include the PCIB asynchronous handshake, instrument address selection and decoding, bus transaction interpretation, register address decoding, register control including command, read, and write, interrupt masking and status, power-on initialization, bus data transmission, and generation of the LED activity signal. These functions are in addition to the controlled slew rate interface to the PCIB.

The third mode, a test mode, configures a serial scan path through a 20-bit binary divider. The scan path reduces the number of clock cycles required to test the sequential counter completely from 2^{20} to 20^2. The test mode is only used at IC wafer test.

Because of the various operational modes, a typical PC Instruments system might contain several of these custom chips. One IC always operates in passthrough mode on the personal computer interface card, and one resides in each instrument requiring the bandwidth of the parallel bus. The pod protocol mode is

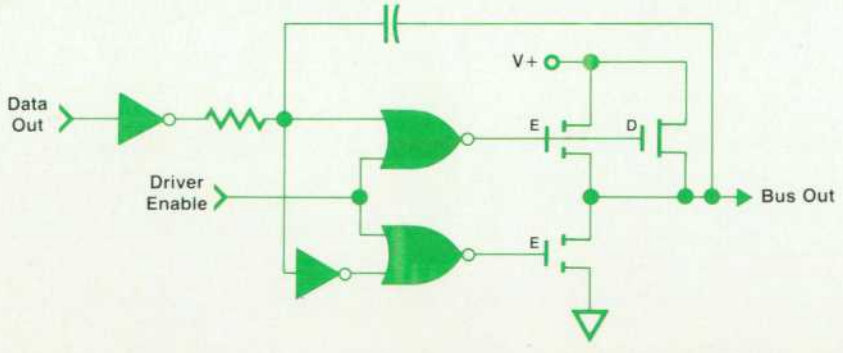

Fig. 1. *Simplified block diagram of the controlled rise-fall PCIB predriver and output driver. Bootstrapping circuitry is not shown.*

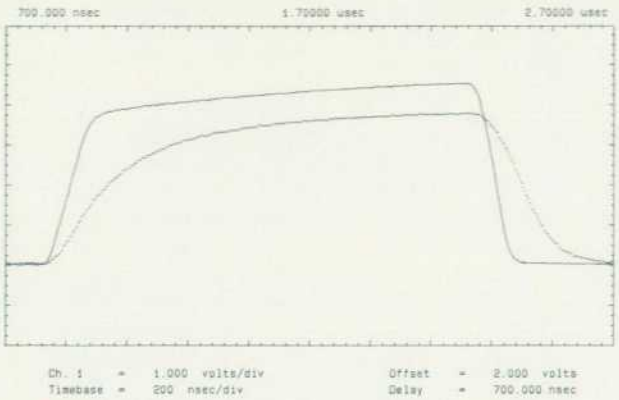

Fig. 2. *Difference in rise and fall times (room temperature) for a 10:1 ratio of capacitive load, same custom IC and axis scaling. Capacitance varies from 130 pF to 1.3 nF. Actual pulse widths on the PCIB are much longer.*

selected for this latter application.

Circuit Design

The custom IC incorporates an inexpensive oscillator for internal timing and provides a 4-MHz buffered signal for external use. The oscillator requires two external capacitors and a low-cost polycrystalline ceramic resonator for operation. The oscillator can be overdriven with an external signal if desired.

The integrated circuit also provides active protection of its own PCIB drivers. When a fully powered instrument module is disconnected from the bus, a sensing circuit in the IC directs the PCIB drivers to drive outward in the active low state. Reconnecting the instrument bus cable allows normal operation to resume.

In its distilled form, the controlled rise-fall PCIB driver consists of a push-pull NMOS output buffer capable of three states and enclosed with negative feedback (see Fig. 1). This produces a dominant pole that prevents the driver from slewing too quickly through its linear region under conditions of light capacitive loading (short bus cable length) and worst-case power conditions (fastest signal edges). An auxiliary bootstrapping circuit (not pictured in Fig. 1) augments the bus waveform rising edge for heavy capacitive loading and worst-case speed conditions. The result reduces a 40:1 process-plus-load spread to less than 6:1 (see Fig. 2).

Acknowledgments

We wish to acknowledge Kent Luehman for conceiving the logical design of the IC, and mask designers Yvette Norman and Carry Perry for generating the computer-aided artwork. George Latham originally investigated the controlled rise-fall driver feasibility. Allen Norskog provided the substrate bias generator design. We also thank Doug Bartlett and Dick Toftness for their guidance and support throughout the project.

Diana G. Bostick
Ricky L. Pettit
Development Engineers
Loveland Instrument Technology Center

the status register of the interface card.

The personal computer interacts with the parallel channel by writing to I/O locations in the computer's address space. There is a separate location for each transaction type. Proper bus sequences are handled by the PCIB I/O drivers. Handshaking between the personal computer backplane and the interface card is provided through an interface status register. This allows the bus operations to be conducted at the rate required by the addressed instrument.

The interface circuits in an instrument module that uses the parallel channel are largely contained in a custom IC. This custom IC handles all of the bus protocol and generates the necessary master data strobe signals for the instrument's registers. The master strobes are used in conjunction with the register number outputs to generate the individual register strobes. This IC also contains the interrupt detection and generation circuits for the instrument.

Serial Communication Channel

The PCIB serial communications channel (Fig. 3) offers an inexpensive isolated data path for instruments that perform floating measurements. The serial bus uses two signals for communication of all messages. These two signals are contained in the same PCIB ribbon cable that includes the parallel bus. A common ribbon cable is used to ensure that the communication channel used by an instrument remains transparent to the user. TxD is the signal from the personal computer used to transmit command, address, and data messages to the instruments. RxD is the signal from the instruments used to return data messages, handshake acknowledgments, and interrupt requests to the personal computer. Allowing multiple instruments to use the same wires for transmitting and receiving messages required the development of a new protocol for serial communications. The new protocol is implemented by a single-chip microcomputer on both the PCIB interface card and in the instruments.

All serial messages from the computer are transmitted as 12-bit frames. The structure of the frame is illustrated in Fig. 4. The first interval is a start bit that synchronizes all instrument microcomputers to receive the transmitted data. The following two intervals are the transaction code as used in the parallel communication channel. They define how the instruments will interpret the data in the remainder of the frame. The next nine intervals are the message byte followed a parity bit for the entire frame. Each instrument examines the frame to determine if it must perform an action. In the case of a universal command, all instruments will execute the command. A selected command will be executed by the instrument that is specified in the command. If an address message is received, the instrument specified becomes addressed. In the case of a talk address, the addressed instrument will retrieve the data from the selected register and return it to the personal computer. For a listen address, the addressed instrument

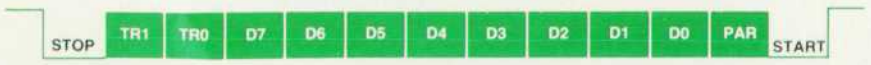

Fig. 4. *PCIB serial frame.*

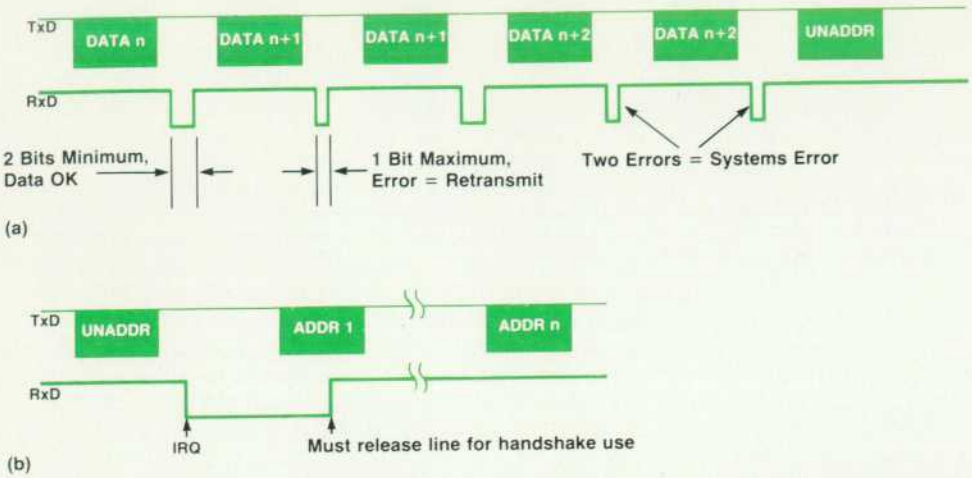

Fig. 5. *PCIB serial communications. (a) Error handshake. If there is no acknowledgment, error or data OK handshake, a system error will result. (b) IRQ protocol. After an unaddress command, an instrument may signal on RxD to indicate that an interrupt is requested. The instrument must release RxD when any transaction is sent on TxD.*

waits for a data byte. When received, it is stored in the selected register.

In addition to the basic operations described above, an instrument's microcomputer performs a number of other functions. The instrument handshakes most messages to acknowledge receipt. Two types of handshakes are returned (see Fig. 5). If the parity of the message is correct, the instrument returns a frame with two bits set indicating that good data was received. A frame with only one bit set is returned if the parity check fails. This indicates that the validity of the data is in question and the microcomputer on the PCIB interface automatically retransmits the previous frame. If a parity error occurs on the retransmission, the transmission is aborted and an error is returned to the system software.

Because all instruments receive all messages, they are aware of the current state of the system with regard to whether someone is addressed. When no instruments are addressed, the RxD line is available for use as an interrupt request line. Any instrument that is enabled to generate an interrupt, and has an interrupting condition, may pull the RxD line low. This signals the microcomputer on the PCIB interface card that some instrument is requesting service. The system software is informed of this and it performs a poll to determine which instrument is requesting service. When a frame is sent on TxD, any instruments requesting service are required to release the RxD line to free it for use as a handshake or data line. In this way, the RxD line performs triple duty.

Acknowledgments

Many individuals contributed in various ways to the success of the interface development and we want to extend our deepest thanks to all. There are several people that deserve special mention because of the extra effort and/or contributions that they made during the development of the system. Genevieve Bliek helped develop the serial communications protocol and implemented the design in the firmware and hardware of the interface. Rich Comins helped define the architecture for the interface in the instruments and worked with Kevin Kayes and Genevieve in developing a processor-to-processor communication method that is used within the instrument. Joel DeLong wrote the interface drivers for the PC and assisted in the verification of the serial communications channel.

Diana Bostick and Rick Pettit of HP's Loveland Instrument Technology Center worked tirelessly to design and produce the custom IC used for the parallel communications channel. Dave Palermo and Dave Wolpert of the Instrument Software Laboratory assisted in the initial evaluation of interfacing methods and instilled the confidence to develop one that specifically addressed the needs of PC Instruments.

Special thanks goes to Eugene Micek who constructed, tested, and debugged most of the hardware. He also wrote the test programs and oversaw the environmental testing of the interface system. Without his help the development would have taken longer and not been as complete.

Reference

1. R.D. Quick and S.L. Harper, "HP-IL: A Low-Cost Digital Interface for Portable Applications," *Hewlett-Packard Journal*, Vol. 34, no. 1, January 1983.

Interactive Computer Graphics for Manual Instrument Control

by Robert C. Sismilich and William T. Walker

ONE OF THE MOST INNOVATIVE ASPECTS of HP's PC Instruments family is the soft front-panel program supplied with the HP 61060AA and HP 61061BA system software, which provides an interactive graphics mechanism for the user to control instruments manually. It calls the same instrument driver module to control the instruments as does the language cap (see "Language Cap" on page 7). It can be invoked directly from the MS™-DOS command line for manual applications, but also can be made coresident with GW™-BASIC and the language cap, and used interactively during program development and debug, or in semiautomated applications.

A soft front-panel application program, PANELS.EXE, provides manual instrument control of each PC Instruments module on the PCIB (personal computer interface bus). The soft front-panels displays look and behave just like their familiar hardware counterparts. Numeric inputs, control functions, and output displays are unified and systematized from instrument to instrument. There is a synergism between manual and programmed instrument control with identical user-defined names, control syntax, and error messages in both environments.

Benchtop Metaphor

PANELS.EXE emulates a benchtop stacked with traditional instruments by allowing the user to control one while viewing them all. This is the definition of the benchtop metaphor as shown in Fig. 1. It is a simple matter to select a new instrument, but in manual mode only one instrument is controlled at a time. PANELS.EXE supports touchscreen, mouse, and keyboard inputs as the pointing interface to control soft front panels. At the same time the user can easily view the full state of other outputs and inputs for other PC Instrument modules on the PCIB.

PANELS.EXE has a self-contained multiple-window manager to control concurrent instrument operation. The benchtop metaphor is implemented through a series of four nonoverlapping windows. The one instrument under direct control resides within the interactive instrument window. All other instruments on the bus can be viewed in the system view window. Updated readings from measurement instruments appear in the system view window as they become available. The status window and the system softkey window round out the operation of this metaphor.

System View Window

Every PC Instrument module on the bus is represented within the system view window as a user-defined name along with device state information. The order of each instrument within this window is determined by the hardware interface address and the hardware bus address. Since there can be many instances of the same instrument type, user-defined names help to differentiate them. For example, two separate digital multimeter modules with factory default labels of DMM.01 and DMM.02 are easily renamed BIAS.VOLTAGE and MOTOR.TEMP. Instrument names are defined using a soft rear panel, which can be brought up into the interactive instrument window.

The system view window presents a summary of each instrument's state. It is like glancing along the bench to check on other measurements while making a critical adjustment. Output, or stimulus devices within the system

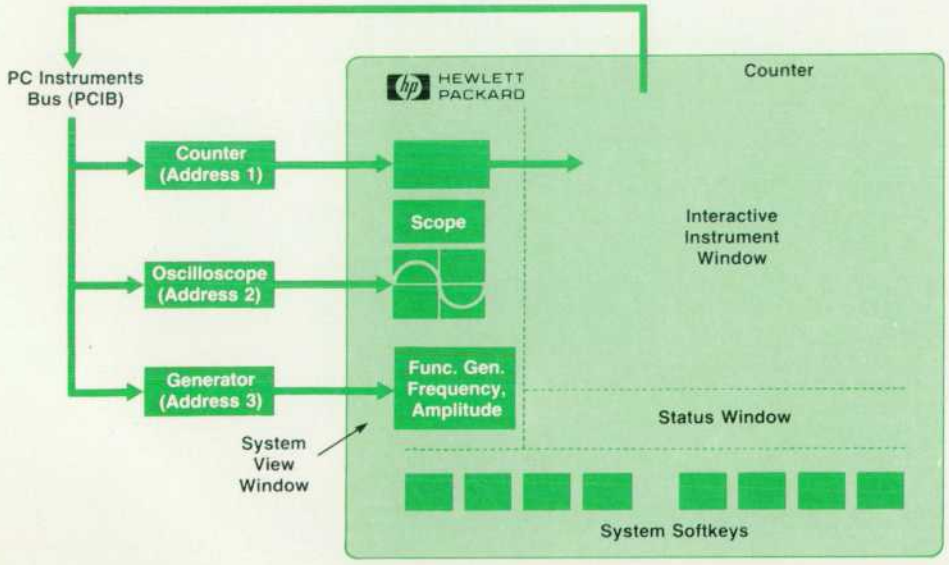

Fig. 1. *Benchtop metaphor: control one instrument, view all.*

view window show their last programmed state. For example, a function generator shows its programmed frequency and amplitude. Properly triggered input, or response devices continuously update within the system view window. For instance, a digitizing oscilloscope updates a waveform trace, while a universal counter displays its latest reading. Merely pointing to an instrument within the system view window brings it into the interactive instrument window. Figs. 2 and 3 show display changes while bringing up a new instrument.

Interactive Instrument Window

When an instrument's soft front panel comes up within the interactive instrument window, it is immediately recognizable. The soft front-panel graphics are not abstract icons, but rather graphics representations of familiar hardware controls. Operation of an instrument from the interactive instrument window is also the same as with its hardware equivalent. A touchscreen or mouse can be used to point to and select a change in instrument function, range, trigger, or mode. New values of frequency, time, amplitude, or offset are easily entered from the host personal computer's keyboard. When another device is brought into the interactive instrument window from the system view window, the system software stores the last state of the previous instrument in the interactive instrument window before returning it to the system view window.

Status Window and File Prompts

The status window contains five lines of information located directly below the interactive instrument window. Here the system software presents the user with status messages, error messages, prompts, and requests for file names. Friendly suggestions for fixes accompany error messages in this system. All messages viewed from the status window and all other text displayed by PANELS.EXE reside in separate ASCII files for ease of localization.

File names for instrument state files and for GW-BASIC program shell files are prompted from the status window. The system software supports full MS-DOS file names to include the disc drive, the directory, any subdirectory, the file name, and any file extension. State files store the current configuration of each PC Instruments module on the bus. State files are very useful for recalling multiple stored test setups from PANELS.EXE, and for programming the complete initialization for any number of instruments with just two lines of GW-BASIC code.

System Softkey Window

System softkeys provide system-wide operations; they are not instrument specific. These softkeys are task-oriented, assisting the user in labeling a device, storing a state, enabling/disabling all outputs, or returning to the last instrument previously viewed in the interactive instrument window. The PRINT SCREEN softkey dumps all four windows of the current screen to a graphics line printer for hard copy, a very useful feature when preparing a lab notebook record of an automated test.

When the user is ready to exit PANELS.EXE to return to the MS-DOS operating system, the system softkeys provide the mechanism. Upon exit, the system software automatically stores the state of all instruments so that the next time PANELS.EXE is run, the instruments will be configured as they were when they were last left. But before exiting, if the user is preparing to control instruments from a GW-BASIC program, the user can create a program shell file,

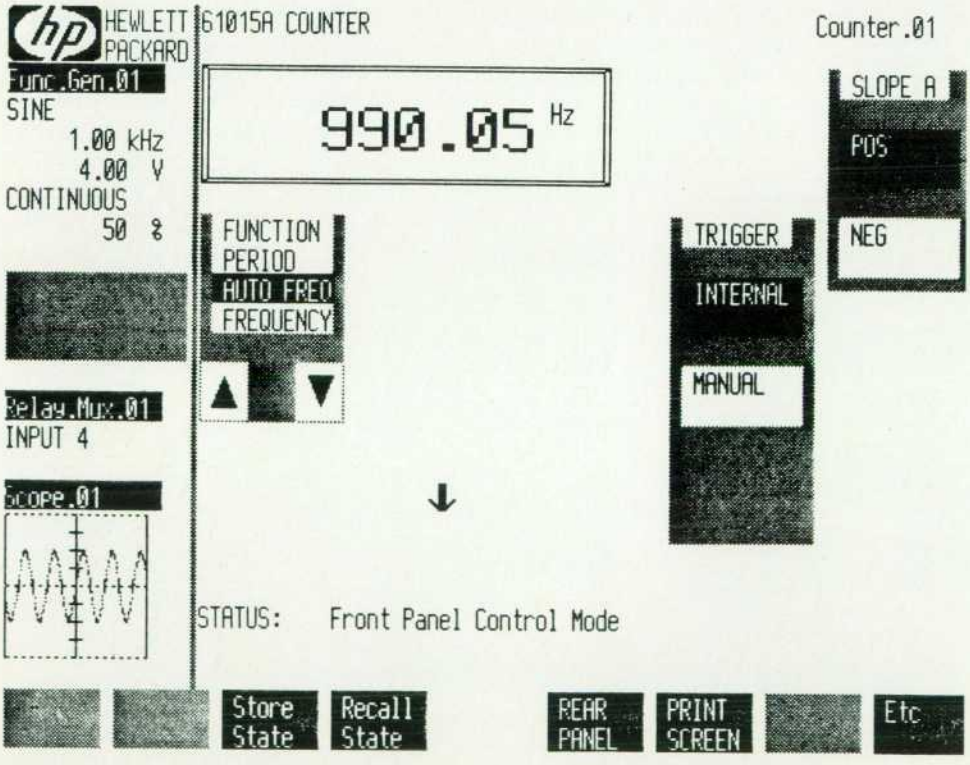

Fig. 2. *Counter soft front-panel display in autofrequency operation.*

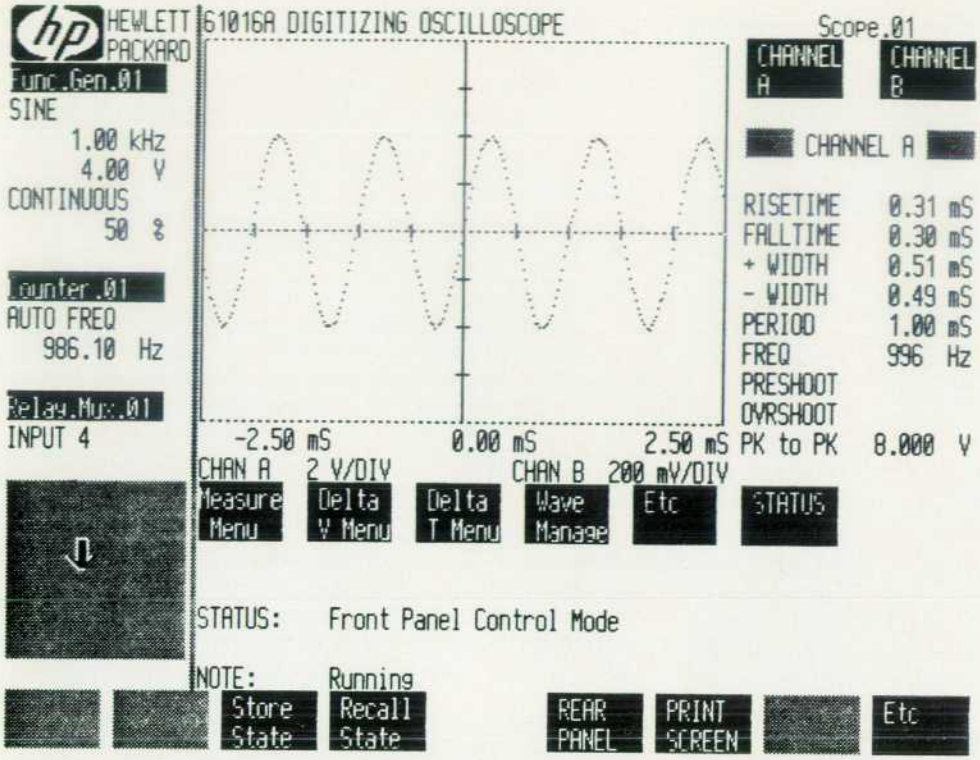

Fig. 3. *Moving the cursor to the indicated position and selecting it causes the HP 61016A Oscilloscope soft front-panel display to appear. (Note that the counter is removed to the system view window.)*

which greatly simplifies the programming task. The shell is a program that initializes all the GW-BASIC variables that are PC Instruments verbs and constants, and builds instance variables for each instrument in the system using the names the user assigned to them. The program shell file is automatically generated and stored to disc with a system softkey. The user just edits the program shell to add computation, control, and instrumentation statements appropriate to the application.

Graphics Component Toolkit

Key to the unified appearance and feel of the PANELS utility is a toolkit of interactive graphics components that the designer of an instrument's soft front panel is able to choose from. These components are analogous to the physical devices that hardware front panels have been built with for years—switches, LEDs, CRT displays, etc. The collection of available components is shown in Fig. 4, along with various attributes of these components which will be explained in the following discussion.

Each type of component has the appearance and operation that its description would suggest, as well as a state attribute. A ganged switch component (see Fig. 5) can have several labeled buttons, only one of which can be on (shown in print as white text on a black field) at any time. When the operator selects a different button (by pointing to it), the previously selected button is turned off. The state of this type of component represents the button that is currently on. Ganged switches are an example of display variance components; that is, the graphics components that appear on the screen can change, depending on the current state of a ganged switch, as will be discussed shortly.

On the other hand, an LCD (large-character display) component (see Fig. 6) would display a changeable, numeric quantity within it. The state attribute of an LCD is the numeric string that is currently displayed in it. LCDs are an example of data display components. LCDs are not selectable since they do not control instrument functions; thus, pointing to an LCD results in an invalid entry beep.

Component Templates

The graphics routines that make these components appear and function in the specified manner are common to all instruments. The designer of an instrument's soft front panel just creates a data structure that declares what components are to be used, where they will be located in the interactive instrument window, and how the various components should be grouped together. This machine-readable description of the graphics appearance of a particular instrument is known as a component template. There is only one copy of the template for an instrument type, regardless of how many instances of that instrument exist in the system. When the PANELS program is executing, instance variables of each instrument in the system store the states of all components in the instrument's template, in a

Component Type	Selectability	Display Variance	Data Display
Ganged Switch	X	X	
Rotary Switch	X	X	
Toggle Switch	X	X	
Momentary Contact Switch	X		
Numeric Entry	X		X
Text Entry	X		X
LCD			X
CRT			X

Fig. 4. *Soft front-panel components in graphics toolkit.*

Mouse in Danger: Managing Graphics Objects

In the PC Instruments PANELS utility, each graphics component that appears on an instrument's soft front panel is treated as separate object. When working with objects, a desirable goal is to have them behave autonomously. The following is an example of how this is accomplished for the soft front-panel displays for PC Instruments modules.

The soft front-panel display for the HP 61016A Oscilloscope module will be used as an example where many objects occupy the same general space and are updated in a nonsynchronous fashion. One worst-case example with seven objects interacting is shown in Fig. 3 on page 19. The specific objects involved are:

- The pointing cursor (mouse)
- The top vertical marker
- The bottom vertical marker
- The left horizontal marker
- The right horizontal marker
- The CRT waveform (trace)
- The CRT graticule.

For this example, assume that the user, after electing to move a marker, is idly moving the mouse around in the CRT area of the soft front-panel display, and that the CRT waveform, or trace, is being continuously updated. Note that the stimulus for updating the display is coming from two different and essentially random sources, the PC Instruments interface and the user.

The trick is to display and update all of the above objects on a single hardware graphics plane without objects interfering with each other's appearances. The final display will appear to have the mouse icon occupy the visual plane closest to the user; that is, it will appear to move about on top of the other objects. Successive underlying layers will have the vertical markers, horizontal markers, and trace objects. The rearmost or background layer has the graticule drawn on it.

A window is made up of a group of objects defined by a component template. When either a user or an instrument requests service, the windows are updated. As part of the update process, each object active in the window inquires about its state from the instrument portion of the front-panel code. If the object's new state differs from its old state, its appearance is updated. Just before it is redrawn, the object checks to see if the mouse is near. If the mouse is "in danger," the mouse is told to hide itself. The object then redraws itself. After it is finished the mouse is told that "all is well" and to show itself. This additional overhead for each object is time well spent; without the mouse-in-danger routine, the mouse would tend to blink continuously as the windows were being updated, whether the mouse was in danger of being overdrawn or not. With the mouse-in-danger routine, the mouse will only appear to blink when near or on an object being updated.

Now that the mouse is safely out of the way, the object (in this case, the CRT) is free to draw itself. The CRT recognizes several messages. These messages are stacked by the instrument code and are handled by the CRT object at update time. For this example, the two commands queued are MOVEMARKERS and DISPLAYERASE.

The only attribute that changes for a marker object is position. The process of moving a marker to the new location involves first taking a snapshot (saving the graphics data) of the area under the new position. The new marker is then drawn. The old marker is then erased using the snapshot taken from the previous move. This process ensures that the objects behind and in front of the marker are not disturbed as the markers are moved. This same process is used to hide and show objects. The act of showing involves taking a snapshot of the display under the object and then drawing the object. To hide the object this snapshot is simply copied back to the display.

The sampling oscilloscope in variable-persistence mode supplies two sets of data. The first, the erase array, represents data points that have grown old enough to fade from the CRT. The other array represents the youngest data. Before any trace data is drawn, the markers are told to hide themselves. The erase array is then plotted. Some of the erased pixels probably coincided with the graticule, so now there are holes in it and it must be redrawn. Next, the new data is plotted. The CRT finishes the update by telling the markers to show themselves again on top of the new trace and graticule.

The above description is a quick overview of the process required to update the six objects making up just the CRT portion of the oscilloscope's soft front panel. On the average there are an additional 20 to 30 objects active in the interactive instrument window of the oscilloscope at any given time. Add in the resource requirements to update the other windows and, last but not least, to service the instruments and it becomes clear, in view of the extensive hardware and software support required, why interactive graphics are just now becoming possible at a personal computer level.

Daniel J. Martin
Development Engineer
New Jersey Division

manner analogous to how instrument states are stored in instance variables in the instrument driver environment.

Solving the Overdensity Problem

One of the problems with the front panels of traditional instruments that the component template solves is that of the density of components on a front panel. Complex, multiple-mode instruments that use traditional front panels have to choose between implementing a large number of controls and displays, many of which are not relevant to an individual measurement and thus contribute to front-panel clutter, or using the same controls and displays for multiple functions, each of which requires its own graphics screened in different colors on the panel—an approach often confusing to the user.

The tree structure of the PC Instruments component template, however, provides a mechanism whereby only those components that are relevant to the current measurement or control operation appear on the display. Each node in the structure is a front-panel component. Components that have the display variance attribute (see Fig. 4) support state-dependent component branches. The state of these components determines which branch of the tree is traversed when checking the template for a match with the coordinates of the position that the user selected, or when the window is updated. These operations are explained in the section "Operation of the PANELS Program" below.

Structure of the PANELS Program

PANELS has an object-oriented structure similar to the

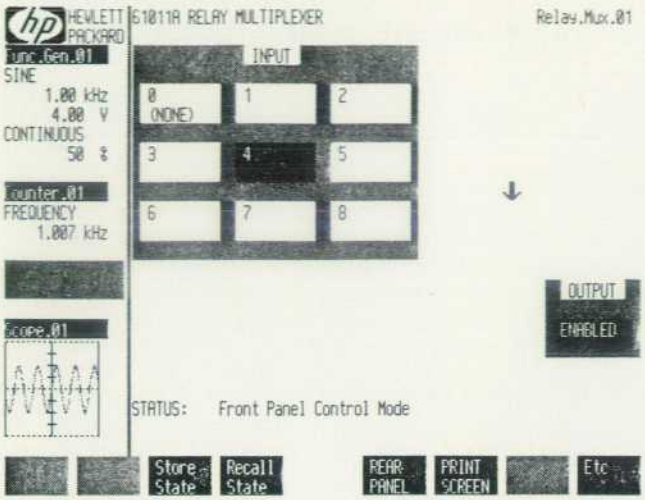

Fig. 5. *Example of ganged switch (Input block at upper left center) from HP 61011A Relay Multiplexer module for PC Instruments.*

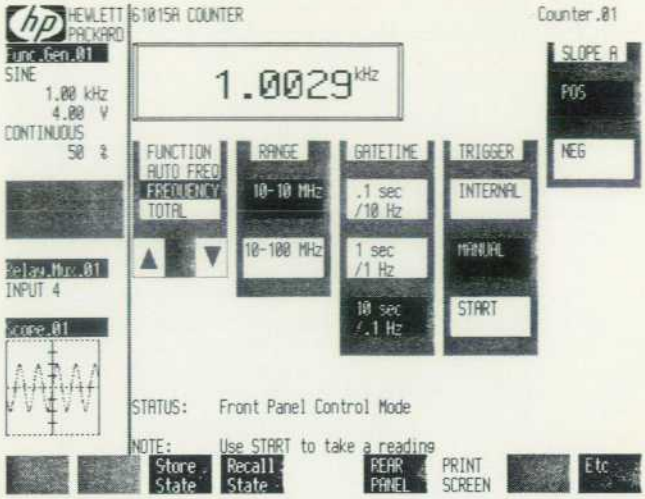

Fig. 6. *Example of large-character display (upper left center) from HP 61015A Counter soft front-panel display.*

instrument driver module. It consists of some generic code at the top and bottom layers, and some instrument specific code sandwiched in the middle. The generic code includes the executive, the window management system, component template processors, softkey processors, the input and display device handlers, and a variety of library functions accessible by the instrument specific code. The architecture of PANELS is shown in Fig. 7.

The instrument specific code, written in C, has two parts: the component template, and a collection of functions which:

- Perform the proper instrument I/O sequences in response to the user pointing to and selecting a component, and set the state of that component accordingly
- Return the state of a component upon request
- Handle service requests from the instrument.

The instrument specific code is operating system and hardware independent. No human input device or display operation is invoked by instrument specific front-panel code. Instead, all of the interface to the user is handled by generic code. As a result, the software is highly portable.

All instrument I/O is done through calls to the instrument drivers, which are always resident while PANELS is being executed. The front panel knows about the functionality of an instrument, that is, what measurement and control functions it supports, but not about how that functionality is implemented.

Operation of the PANELS Program

It is helpful to consider a simple example of a component template to explain the processes occurring while PANELS is executing. For simplicity we will limit the discussion to operations that occur within the interactive instrument window. Fig. 8 shows a portion of the interactive instrument window component template for the PC Instruments counter.

When PANELS is invoked, the states of all components are initialized. If PANELS is invoked from the MS-DOS command interpreter as a manual application, the instruments are first set to the states that are stored in the state file HPSTATE.HPC. If PANELS is invoked from GW-BASIC as a debugging tool, the state of the components is set to reflect the current state of the instruments. For this example, let's assume that the counter is initialized as follows:

Function:	Frequency
Range:	10 Hz to 10 MHz
Gate Time:	1.0 s
Trigger:	Manual
Slope:	Positive

The screen at this time will appear as shown in Fig. 9.

After initialization is completed, the front-panel executive, PANELS.EXE, alternates between two tasks: handling

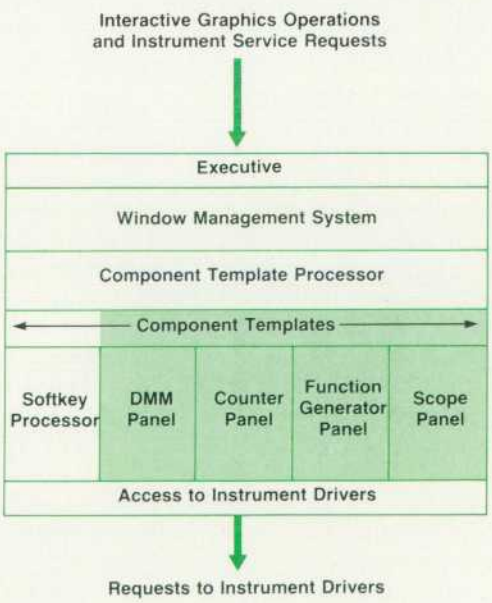

Fig. 7. *Soft front-panel architecture. Shaded areas are instrument-specific functions.*

Oscilloscope Software Leverages Previous Concepts and Algorithms

The software for the HP 61016A Digitizing Oscilloscope module is the most complex in the current PC Instruments line. The software team heavily leveraged existing digitizing oscilloscope software technology available at HP's Colorado Springs Division. The HP 19800 Waveform Measurement Library and the HP 54100A/D Digitizing Oscilloscope became the standard for the HP 61016A software package, which consists of 75K bytes broken into five major modules: hardware setup, data acquisition and display, user interface, measurements, and program control.

The hardware setup for the HP 61016A consists of a 12-byte string which sets the vertical and horizontal sensitivities, delay time, coupling, acquisition mode, and trigger configuration based on user or program specifications. This string is sent to the HP 61016A from the host personal computer over the PCIB. The HP 61016A's firmware then uses the string to configure its hardware.

Data acquisition consists of reading a 251-byte string from the HP 61016A. The data is scaled according to the hardware configuration in preparation for display by the PANELS window updating routine. Scaling is necessary since some functions normally handled by hardware (such as offset and implementation of the very sensitive vertical settings) became software routines because of printed circuit board constraints.

When PANELS is updating the screen, the update_window processor performs a state inquiry operation on the CRT component of the oscilloscope template. The scope software responds with a data buffer that contains commands to reset the soft front-panel display and hardware control strings and clear and reset the data structures.

The measurement package includes control of voltage and time markers and waveform analysis for automatic parametric measurements of rise time, fall time, period, frequency, pulse width, overshoot, preshoot, and peak-to-peak voltage. This package is based on a statistical analysis of the 251-byte data string from the scope using a histogram to determine the absolute and relative maximum and minimum and 10, 50, and 90% points, as well as the location of rising and falling edges.

The HP 19800 Waveform Measurement Library laid the groundwork for the measurement package algorithms. The HP 19800 Library is written in BASIC for the HP 1980 Oscilloscope and was converted to C for the HP 61016A. The confidence in the HP 61016A measurement package is directly linked to the quality assurance testing of the HP 19800 Library.

The basic feature set, data acquisition modes, voltage and time marker conventions, and display techniques were defined using the HP 54100 Digitizing Oscilloscope as a model, with modifications to meet the HP 61016A constraints. For example, the HP 54100 implements a variable-persistence mode of display using clock interrupts. Since the clock is not available to the HP 61016A, the variable-persistence mode is implemented as a function of the number of data acquisitions.

Helen Muterspaugh
Mimi Beaudoin
Development Engineers
Colorado Springs Division

user selections and handling instrument service requests.

User Requests

Positioning of the PANELS cursor (via mouse, touchscreen, or keyboard) is handled at an interrupt level. A user request occurs when the operator selects a graphics component by clicking the mouse button, or releasing a pointing finger from the touchscreen. As an example, let's assume that the operator wants to change the counter to make period measurements. The operator positions the cursor on top of the rotary switch controlling the counter's function, and clicks the mouse. A component template processing routine, hit_detector, is passed the mouse coordinates. It traverses the component list for the instrument currently in the interactive instrument window to determine if the selected coordinates fall within the area of a selectable component currently displayed on the screen. If the end of the tree is reached without a match, a beep occurs indicating that no component exists there. If a component match is found, control is passed to the instrument specific front-panel code for the counter, which executes the instrument I/O actions associated with that component by issuing the proper calls to the instrument driver.

Since not every component in the template is displayed on the screen at the same time, how does hit_detector know which components to check? This is where the display variance attribute comes into play. The hit detector routine is recursive; at each node that represents a component that has display variance, the current state of that component is obtained, and the hit detector routine is reinvoked for the variant branch that corresponds to that state.

What remains to be done after the I/O actions have been completed is to update the interactive instrument window display to reflect the changed state of the instrument. The routine update_window serves as a viewer for the state of the instrument contained in the interactive instrument window; that is, it displays this state information according to the graphics metaphor chosen for the instrument. It does this by traversing the component template and drawing the instrument according to the state of each of the components. This routine always starts at the top of the template. It has three modes: erase, create, and modify. Like hit_detector, this routine is recursive; at each node that represents a component of display variance type, it is invoked again for one or more of the variant branches, possibly with a different mode.

Assume that what the operator did was to use the rotary switch controlling the counter module's function to select period instead of frequency. For the example of setting the counter into period mode, after the instrument I/O is completed, the window management system calls the update window processor in the modify mode.

Modify mode is really a composite of erase and create modes. At each node in the component template, the new state of the component is obtained from the instrument specific code, and compared to the currently displayed state. If the new state is the same as the current state, the routine performs no graphics operation. If the component has variant branches, update_window is invoked recursively for the branch that corresponds to that state, still with

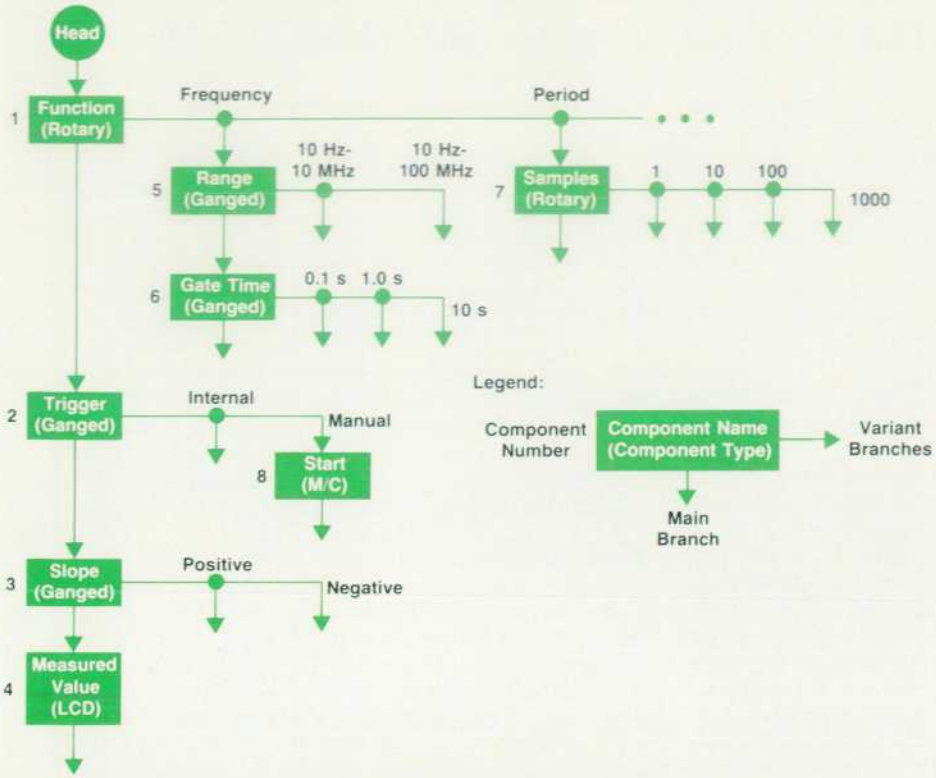

Fig. 8. *Portion of component template for PC Instruments HP 61015A Counter module.*

modify mode.

If the new state is not the same as the current state, the component in question is redrawn to display its new state. If the component has variant branches, a two-step operation takes place: (1) the branch corresponding to the currently displayed state is updated with erase mode and (2) the branch corresponding to the new state is updated with create mode. As indicated in the component template for the PC Instruments counter (Fig. 8), when the state of the function component comes back with current state = frequency and new state = period, both the range and gate time components are erased, and the samples component is created (the variant branches of all these components are null). Returning to the original update_window process, the states of the trigger, slope, start, and reading components are all unchanged, so no further graphics operations are performed. At the completion of the update, the screen looks as shown in Fig. 10.

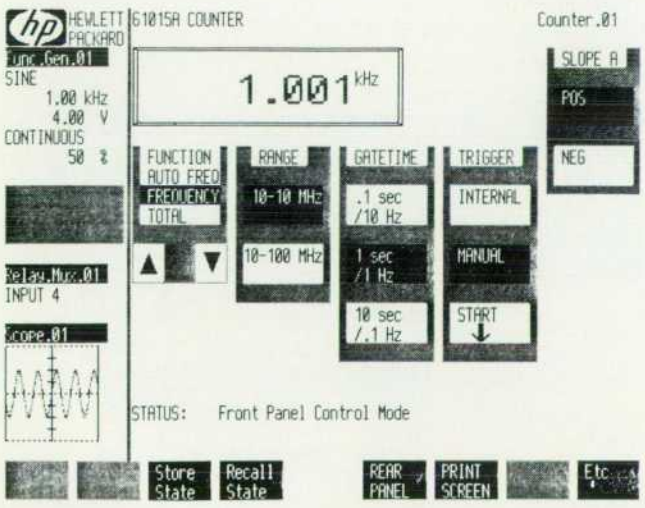

Fig. 9. *Counter soft front-panel display in frequency mode.*

Instrument Service Requests

Now, assume that the operator selects the internal button of the trigger ganged switch. The instrument is now put into free-run mode, and the display changed as shown in Fig. 11. When a new reading becomes available, the counter sets status bits to indicate a service request. As mentioned previously, a main task of PANELS is to look for these service requests from instruments, and update the display to reflect them. When not processing user requests, the instruments are polled for service requests in a circular fashion. When a request is detected, instrument specific code takes care of the I/O, and the window in which the instrument is currently displayed (interactive instrument window or the system view window) is updated. In the above counter example, the only component that will indicate a state change is the measured value LCD, and the operator will see the new value appear within the graphics bezel.

Acknowledgments

Many of the concepts of interactive graphics for instrument control embodied in the PANELS program resulted from the work of John Uebbing at HP Laboratories and Charles Kingsford-Smith at Lake Stevens Instrument Division. John's consultation early in the project gave us a big head start in this new area of instrumentation technology.

Automated Testing of Interactive Graphics User Interfaces

Providing a reliable, defect-free product should be a main design goal of any system software design. Many techniques to find and fix defects are typically employed during the QA phase of a software project. One major concern is that fixes in one module do not introduce new defects into other, previously verified modules. The most useful technique to ensure this is regression testing. A regression tester generally consists of a target machine and a special set of automatic test procedures to exercise and verify as much of the software as possible.

Regression testing of a programming language, such as the GW™-BASIC programming library for PC Instruments modules, is straightforward in the sense that the automated test procedures use the same programming statements that make up the end product.

Testing of the interactive graphics PANELS application program was more difficult since the test stimulus consists of user requests from a keyboard, mouse, or touchscreen, and the response is measured by the graphics results that appear on the host personal computer's CRT display. For this reason, programs of this nature are often left to be tested manually, sometimes with little or no repetitive structure to the test beyond a sequence of keystrokes entered at the operator's whim. Repeating the same type of manual testing each time a change to the software is made is both time-consuming and error-prone. Therefore, an automated process of entering keyboard input stimuli and verifying the graphics output responses was required to perform meaningful regression testing of the PANELS program.

The major problem in testing graphics interfaces is verifying the graphics objects. An object can be verified if it has consistent and predictable form. In the PANELS environment, graphics components with the data display attribute do not have consistent form. For example, the LCD display for the digital multimeter module contains a changing value corresponding to the measured voltage, and the CRT portion for the oscilloscope module displays a trace of the waveform the scope is currently measuring. The PANELS software tester, therefore, limits itself to verifying only those graphics components whose appearance can be known *a priori*.

PANELS uses the graphics plane for all information displayed on the CRT. A simple approach would be to compare the contents of the selected portion of the current graphics plane to a predefined bit pattern at the appropriate points in the test. There are two disadvantages with this approach: pattern storage and retrieval. The storage requirement may become unacceptable as the number of objects and their sizes are increased. Although speed is not a major objective, retrieving the patterns becomes a major controlling factor of the test. An efficient way of representing and identifying objects is needed.

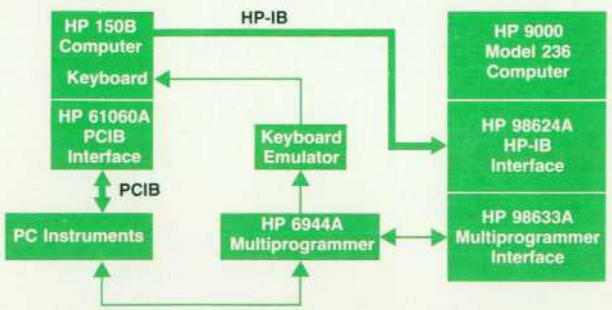

Fig. 1. *Block diagram of tester for PC Instruments PANELS software package.*

Signature Analysis Helps

The digital measurement technique of signature analysis can be applied to identifying graphics components. Since only a single signature is generated for a graphics object, it does not have the storage or retrieval overhead disadvantages of the bit pattern comparison method. In the signature analysis method, the graphics component's pixel pattern is sent as a bit stream into a software cyclic redundancy code (CRC) generator to produce a 16-bit integer representing the signature of that object. An optimized CRC generator routine running on an 8-MHz 68000 microprocessor can generate a signature of a component the size of the entire graphics display, 512 × 390 pixels, in less than four seconds. Usually the graphics areas analyzed are much smaller, and a signature is generated within a fraction of a second.

Tester Hardware

Automating the graphics test required an external processor to control and generate the keystrokes, and to capture and verify the graphics image. Some of the objectives were:

- Low cost
- Minimal hardware modification to the target personal computer
- Execution of the PANELS software without modification
- Relatively high-speed performance
- Minimal test set development time.

The PANELS tester (Fig. 1) is a relatively low-cost solution which requires no hardware modification to the host computer, and the PANELS software is run unmodified. Performance in pattern analysis is achieved by using an HP 9000 Model 236 controller with an HP 6944A Multiprogrammer. The short development time objective was accomplished because of the high productivity offered by the Model 236's BASIC operating system.

As an example, the version of PC Instruments that operates on the HP 150 Touchscreen Computer will be considered.

Generating the Keystrokes

Keystroke generation is done by a keyboard emulator (Fig. 2) which is a direct plug-in replacement for the HP 150 Computer's keyboard. A dual-ported RAM design is used for the emulation. The keyboard emulator is implemented on a digital I/O card in the multiprogrammer, under control of the Model 236. The controller programs the desired keystroke by setting a RAM location through one of its ports. The other port is then read by the HP 150. A

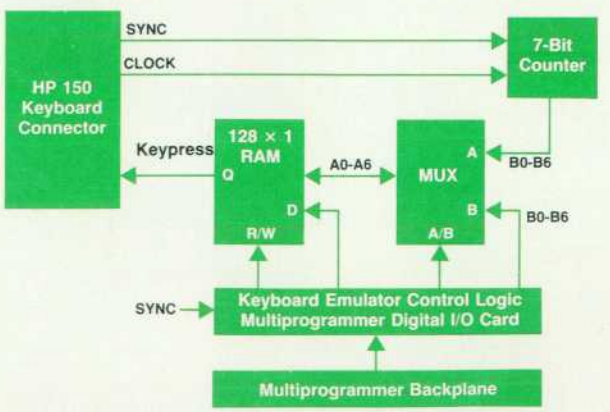

Fig. 2. *Block diagram of keyboard emulator.*

seven-bit binary counter clocked by the standard keyboard pulses from the HP 150 sequences through the RAM locations. A keypress signal is returned when the counter addresses the RAM location that was programmed to be sent. The HP 150 determines which key is pressed by keeping track of the clock pulses it had to send before it received the keypress signal.

Getting the Graphics Screen Information

To capture the graphics image, the Print Screen softkey provided by PANELS is used. This feature causes the graphics image on a host HP 150 display to be output to a graphics printer through the HP 150's built-in HP-IB (IEEE 488/IEC 625) port.

In the PANELS tester, the HP 150's HP-IB port is not connected to a printer, but is connected to another HP-IB interface within the Model 236 controller which emulates a printer. The graphics image is stored in an array, and the areas corresponding to the appropriate graphics components are run through the signature analysis algorithm.

Tester Software

Finally, a test executive program is used to control the operation of the tester. The test executive has four modes of operation: live keyboard, log, playback, and user subprogram execution mode. It has utilities for emulating the printer, generating the cyclic redundancy code, and controlling the keyboard emulator.

In live keyboard mode, an operator can interact with the HP 150 Computer through the keyboard of the Model 236 controller. Various utilities can also be called by using the function keys on the Model 236.

Log mode operates similarly to live keyboard mode with one exception—the keystrokes, and the time delay between them, are recorded as they are entered and stored into a user-specified file. This is a useful feature for specifying test sequences and recreating bugs.

In playback mode, the user specifies a file that was created through log mode, and the associated keystroke sequence is executed.

The subprogram execution mode allows the user to run a subprogram in which keystrokes can be programmatically sent, graphics data received, and the results analyzed. Automated regression testing of the PANELS program is executed in this mode. Typically, a regression test for an instrument will start from a node in the instrument component template, moving the cursor to the appropriate screen locations and making user requests to PANELS via the keyboard. After the screen is updated as a result of the user request, the graphics image is sent to the Model 236 controller via the Print Screen softkey. After the graphics data is received, a signature is generated for the specified area of the screen. If the signature matches the stored reference, the particular test passes, and the next node in the template is tested.

Buck H. Chan
Project Leader
New Jersey Division

Kevin Kayes, now with Logic Design Operation, was the architect of the instrument drivers. He also contributed heavily to much of the internal design of PANELS, and his ability to adhere to module interfaces defined early in the design phase was a large reason that the software worked right the first time.

Dan Martin was responsible for the PANELS graphics, from the component template processors right down to pumping the pixels onto the screen. His concern for tuned code helped a product whose top goals were portability and expandability to achieve performance fast enough to avoid having the user wait between commands.

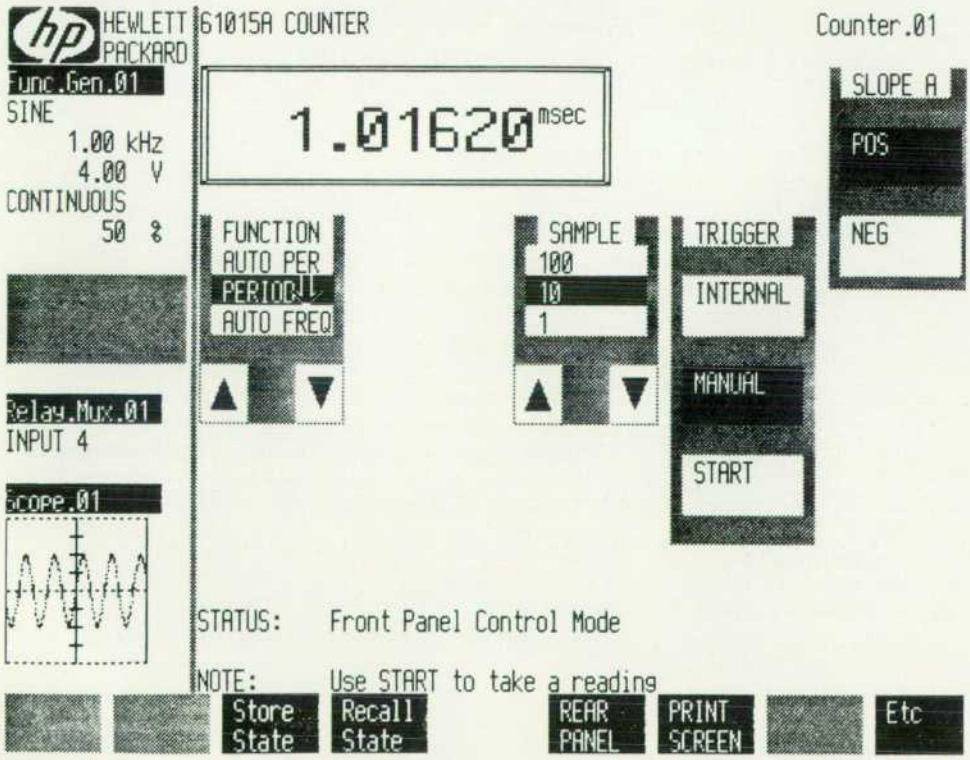

Fig. 10. *Counter soft front-panel display in period mode.*

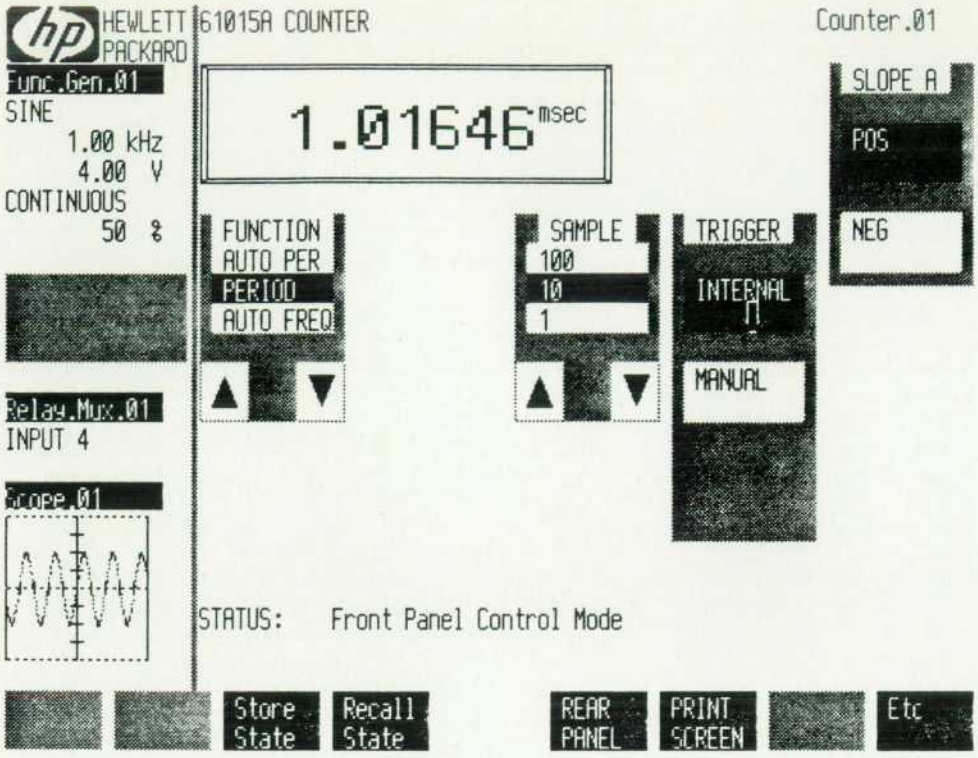

Fig. 11. *Counter soft front-panel display in internal trigger mode.*

Dan Beinart brought his considerable MS-DOS expertise to bear on developing the language cap for GW-BASIC in a way that, among other things, shields the customer from much of the crudeness of the GW-BASIC CALL mechanism. Ed Laczynski, Pat Lam, Helen Muterspaugh, Mimi Beaudoin, and Joel Delong designed the instrument specific front-panel and instrument driver code for the initial PC Instruments modules. Dale Breustedt designed all the component templates, in close coordination with Dave Schlesinger of the industrial design group. Pat Benson designed the low-level human input drivers, and Mark Seroka coordinated a very thorough software QA phase. Ali Al-Aref contributed the CONVERT utility, which gives customers access to a wide variety of their favorite MS-DOS presentation and analysis packages with PC Instruments data.

Industrial Design of Soft Front Panels

Interactive graphics on a personal computer's screen can be compared to hardware user interfaces. Industrial designers have worked with mechanical and electrical engineers to lay out the front and rear panels of HP's hardware designs. Now that customers must deal with software user interfaces, industrial designers are working with software engineers to design these interfaces. The future man/machine interface is the computer-aided work environment.

In approaching this new environment of soft front panels, we need to evaluate hardware and software constraints of the system realistically. Taking this into account, the PC Instruments designers wanted to produce an easy-to-use, familiar appearance to our customers. Graphics components represent the real-world displays and controls. Screen partitioning locates things in consistent places on the screen, similar to general front-panel guidelines.

For example, HP front-panel industrial design guidelines for hardware locate the HP logo in the upper left, the ac line switch in the lower left, displays to the left or top, and controls to the right or bottom. The soft front-panel display for PC Instruments modules is similarly sectioned to have the logo in the upper left, the system view window along the left side, the instrument to the right, and the status window and menu softkeys along the bottom. (Refer to figures in accompanying article for examples of PC Instruments displays.)

In a natural presentation, screen sections are arranged into one total environment. The screen simulates the hardware world of instruments, controllers, and display prompts. The system view represents the instruments in either a rack or bench stack situation. The instrument currently selected appears in the interactive instrument window ready for full viewing and control access. A soft rear-panel view of the instrument displays the interface bus address and an entry field to allow the user to select a custom name for the instrument. The front-panel layout within the interactive instrument window emulates a typical hardware layout with displays to the left, controls to the right, and a top-to-bottom, left-to-right user flow. Controls are activated in the same manner as hardware switches. They are either on or off. The icons representing the controls change from dark to light as the controls are switched on.

The status area, which is located below the interactive instrument window, displays annunciators, error conditions, messages, and other programming information. The softkey menu is consistent in style and location with that used by HP computers.

David Schlesinger
Industrial Design Manager
New Jersey Division

HP-IB Command Library for MS-DOS Systems

By David L. Wolpert

THE HP-IB COMMAND LIBRARY provides HP-IB (IEEE 488/IEC 625) instrument control capabilities for MS™-DOS computer systems. The HP 61062BA version is used for the HP Vectra PC and the HP 14857A version is used for the HP 150 and Touchscreen Computers. The Command Library offers interfaces between the HP-IB and MicroSoft Pascal and C, interpreted and compiled BASIC, and Lattice C. Its features include a number builder and an interface with HP's PC Instruments software.

The Touchscreen Computer was introduced in 1981 as the HP 150A. It had a connector on the back marked **HP-IB**. In the business world that this product is designed for, the Hewlett-Packard Interface Bus is not well-known, but HP's technical customers have come to know that HP-IB means "Hewlett-Packard's Implementation of IEEE Standard 488, A Standard Digital Interface for Programmable Instrumentation." At its introduction, however, the HP 150 could not program any Hewlett-Packard instruments, because the HP-IB interface was only included to allow the HP 150 to use HP's many computer peripherals based on the HP-IB.

At that time, a team of engineers at HP's Instrument Systems Laboratory was investigating near-term requirements for low-cost instrument controllers. Making the HP-150 a low-cost instrument controller seemed to fit right in with the lab's charter. We estimated that the investment required would be minimal, because no hardware development would be involved and the components in the interface were familiar ones, having been used in other HP controllers and instruments. The HP 150 already contained an elementary HP-IB driver for disc drives and other peripheral devices. This driver for disc drives and other functionality, meaning that we would have to provide only a software layer to make it accessible to programming languages such as BASIC, Pascal, C, and Fortran.

Another development effort occurring at this time was the PC Instruments program at the New Jersey Division. The PC Instruments designers wanted to supply systems based on the HP 150 and the IBM PC. They were considering developing an HP-IB capability for these computers as an enhancement to their system. Users could then add measurement capabilities not included in the initial offering of PC Instruments. For example, a future user might need a higher-resolution voltmeter or a spectrum analyzer.

The PC Instruments system is intended to be user-programmable in BASIC, and the PC Instruments command subprograms are compiled. The PC Instruments design team developed a language cap (see "Language Cap" on page 7) for calling these compiled programs from BASIC, and we found that we could use this to access the assembly language routines of the HP-IB Command Library. The language cap is more than just an interface between MicroSoft GW-BASIC and compiled subprograms. The language cap developed for PC Instruments makes GW-BASIC more usable as an instrument control language. It gives the personal computer programmer convenience close to that obtained if I/O functions were actually built into the language.

The language cap made providing I/O for BASIC much easier, because we were able to borrow that code and reduce the development time required. Hence, we decided to develop a command library to complement the PC Instruments program and to serve general needs for HP-IB I/O on personal computers.

Since the IBM PC does not have a built-in HP-IB interface, HP needed to provide one. HP's Greeley Division was interested in offering HP mass storage solutions to IBM PC owners, and had started development of an HP-IB interface card. Responsibility for this card was eventually transferred to the Roseville Networks Division. The design teams at these two divisions worked with us to ensure that the HP-IB card could meet the needs of instrument I/O as well as the needs of disc and tape drives. Because the interface card uses hardware similar to that used in the HP 150, not much additional software effort was required to provide the same I/O capability for both the HP 150 and the IBM PC.

Definition

The technical documentation for the HP 150 mentions several ways to access its HP-IB capability from a program. We considered each of these methods, and modifications to them, as possibilities for our product. Meanwhile, we decided to create an industry-standard version of the library for the IBM PC and its compatibles. Our final choice of implementations was largely influenced by this decision.

One standard way of adding arbitrary devices to MS-DOS for system expansion is through the mechanism of an installable device driver. This allows the device to be managed by the operating system through standard IOCONTROL, OPEN, CLOSE, READ and WRITE calls. This looked like a promising way to interface the Command Library to the HP-IB hardware.

The HP 150 comes with a built-in driver for the HP-IB, called HPIBDEV. This is used by the disc and peripheral drivers, and is a fairly complete implementation. It is documented in the *HP 150 Technical Reference Manual*, and works well for block transfers. An IOCONTROL call passes a data structure to the driver, indicating the operations to be performed and the buffer area for data transfer. However, there is not enough functionality for control of instruments. For example, there is no way to send arbitrary commands, no way to terminate transfers on EOI (end or identify), and no way to change the time-out value (except on and off). We also needed a few more functions, mainly

related to control of the REN (remote enable) and IFC (interface clear) lines, and access to status information.

Hence, the Command Library is written as a separate linkable library module rather than as an extension to HPIB-DEV or as an additional driver. This has several advantages for the users as well as the developers:

- When the computer is running a program that does not require I/O, no extra memory is used.
- The language interface module is complete, and the programmer need not make any assumptions about the user's system configuration.
- The Command Library works with programming languages that do not allow access to the MS-DOS IOCONTROL system function.

Two languages, BASIC and Pascal, were chosen for the initial Command Library offering because of their familiarity to designers of automated instruments.

Technical Details

Developing a product for simultaneous introduction on two different hardware configurations in two different programming languages was a real exercise in structured programming. The requirements of operating in BASIC and Pascal, both on the HP 150 and the IBM PC, forced us to separate the hardware interface and the language interface. This will allow us to add support for additional programming languages easily in future versions.

The design used has the structure shown in Fig. 1. Level 1 of the hierarchical structure is the user's program, written in BASIC or Pascal. When an I/O function is needed, the program transfers to the I/O library through a language interface (Levels 2 and 3). This interface takes the passed parameters (numbers or strings) and loads the processor registers with the values expected by the lower-level routines (e.g., select code, device address, and pointers to data).

In MicroSoft Pascal, the interface is very simple. The Pascal compiler provides extensive type checking, so the parameters are simply declared to be of the expected type. The language interface just moves the parameters to the proper registers.

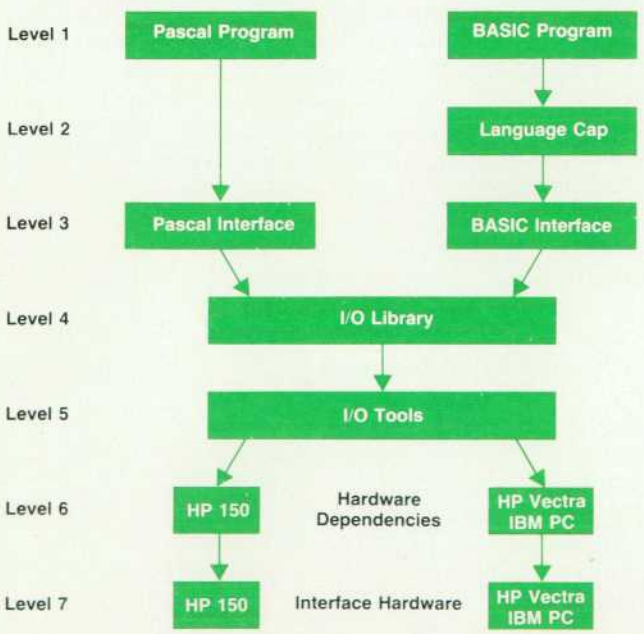

Fig. 1. *Structure of the I/O library.*

BASIC is another story. The only facility built into the language for using external subprograms is a direct CALL to an absolute machine memory location. The language cap for the PC Instruments system provides some parameter checking, handles numeric conversions, initializes variables to point to the subroutine locations, and does some error handling. This language cap is used as a preprocessor (Level 2) for the BASIC language interface. Underneath this language cap is a simple interface similar to the one used by the Pascal version of the I/O library.

Level 4 contains the library. All the routines to control the performance of HP-IB functions are written in assembly language. The library-level code divides these functions into their simplest component parts. For example, the HP-IB function "read an array of numbers from a device" is broken down into the following sequence:

- Check for a valid interface select code and device address.
- Address the device by sending HP-IB commands.
- Do until array full:
  Read bytes and convert to numeric form.
- Check for errors, time-outs, number of elements, etc.
- Return to user program.

The process of converting the stream of bytes to numbers, and vice versa, is handled by a number builder, loosely based on the I/O formatter of the HP Series 80 and HP 9000 Series 200 Computers. As many operations as possible are performed with integer arithmetic to save execution time, even though the result for "read a number" is in floating-point form. The module was written in a high-level language to save development time. For the BASIC version of the library, this module is coded in C; for the Pascal version, it is coded in Pascal.

Routines within the I/O tools module (Level 5) are not directly accessed by the user's program. These are called by the library module to perform such operations as bus setup, reading and writing strings and bytes, and sending commands. This level, as well as the library level of code, is the same for all the implementations.

All hardware dependent code is isolated in a module (Level 6) that is different for the HP 150 and the IBM PC. This module knows about address and I/O mapping, and which registers on the interface controller chip to use for data and status.

Another part of the HP-IB Command Library is a driver for HP-IB peripherals. This is installed at system bootstrap time and allows the user to access printers and plotters. It is accessed through the BIOS interrupt system, so that most standard software will work with an HP-IB printer as well as the standard parallel printer interface. This will allow both HP's Vectra Personal Computer, which is an enhanced IBM PC compatible, and the IBM PC to be used with a customer's HP-IB peripherals.

Acknowledgments

By the time the HP-IB Command Library for MS-DOS was introduced, many people at different HP divisions had made significant contributions to its success. Bill Terry,

an HP executive vice president, suggested the idea, and gave us the go-ahead for the project. Bill Hunt wrote most of the I/O library code, and Julie Funk provided material for the technical reference section of the manual and did extensive testing on our library and on competitive products. Joe Mueller suggested improvements to the number builder to make it fast and accurate. Paul Maybaum outlined a marketing and support strategy, Donna Kimble helped with product marketing, and Van Walther helped with training and support. Mark Burak wrote the manual.

The New Jersey Division, as the actual supplier of the product, was, of course, a major contributor. Dan Beinhart deserves special mention for providing the language cap interface to GW-BASIC. Charlie Rothschild and Charlie Thompson and their staffs were always ready to answer our questions about PC Instruments, as well as work with us on related support issues.

The Roseville Networks Division provided the hardware required for operation with the IBM PC, with Steve Goody doing the design, Aaron Masters (project manager) keeping the schedule honest, and Karen Dudley handling product management duties. Greeley Division's mass storage group provided the initial motivation for the card, and Sterling Mortensen and John Szlendak worked with us to ensure instruments and mass storage could be supported by the same hardware. Don Clark made a great effort to put the low-level I/O code in the ROM on the card.

People at the Personal Office Computer Division provided insight into the guts of the HP 150, and some others at the Fort Collins Systems Division helped us understand what "real I/O" is all about by describing the Series 200 I/O system.

Case Study: PC Instruments Counter Versus Traditional Counters

Combining the power of a personal computer with the measurement capability of a low-cost module with no front-panel controls of its own can be an attractive alternative to using traditional instruments for the owner of a personal computer.

by Edward Laczynski and Robert V. Miller

THE DESIGN OF THE HP 61015A Universal Counter for HP's PC Instruments product family is leveraged from the designs of the HP 5314A and HP 5316A Universal Counters. The HP 5314A was chosen because it offers excellent price/performance at the low end of standalone, manual applications. Similarly, the HP 5316A covers the low end of Hewlett-Packard's system offerings. The HP 61015A Universal Counter addresses both of these areas for personal computer users, providing clear and convenient manual operation using PC Instruments' soft front-panel application software as well as a programming mode that is easy to use and learn for system applications. The following paragraphs describe the HP 61015A Universal Counter in terms of its differences from and its similarities to its traditional instrument relatives, the HP 5314A and HP 5316A Counters.

Manual Operation

The HP 5314A Counter is a 100-MHz, 100-ns universal counter that features a seven-digit, seven-segment LED display with overflow indication, input signal conditioning, and the following measurement functions:

- Frequency
- Period
- Time interval
- Ratio (A to B)
- Totalize (unit count)
- Self-check.

The user sets up the desired measurement by selecting the appropriate controls on the HP 5314A's front panel (Fig. 1). Since the front-panel area is limited and because of the low-cost application focus of the HP 5314A, the counter employs several dual-function controls. The counter's measurement function, for example, is selected by three function switches. The shift key determines four of the available functions. With the shift key out, placing either (but not both) of the other two function switches out selects the corresponding frequency or period measurement. With the shift key in, depressing either (but again, not both) of the other two function switches selects the corresponding totalize or time interval measurement. The other two available functions, ratio and self-check, are selected by placing both function keys out or in, respectively, in which case the state of the shift key is a "don't care."

Fig. 1. *HP 5314A front panel.*

The next three switches on the HP 5314A's front panel are also multidimensional. For frequency measurements, for example, these switches determine the length of the gate time that the input signal is applied to the counter (and therefore, the resolution of the reading) and the input conditioning. (For inputs over 10 MHz, a divide-by-ten prescaler must be selected by pushing the **10Hz/N = 1** switch in.) For period measurements, however, these controls determine the number of cycles that are averaged for a measurement (i.e., the resolution of the reading).

The remaining controls on the front panel are used for selecting additional signal conditioning, specifying the slope of the input signal, input attenuation, and level adjustments, and to select whether separate or common (single signal source) inputs are to be used for time interval measurements.

It is apparent that the user is presented with a sometimes bewildering combination of choices in making any of the several measurements available with the HP 5314A. A major challenge in designing the interactive instrument window of the HP 61015A's soft front panel, therefore, was to simplify the counter's manual operation. While the specifications of the HP 61015A are very similar to the HP 5314A (with the addition of autofrequency and autoperiod measurements), the user's perception of the instrument in manual mode is quite different. Since the HP 61015A's front panel is realized in software, much more flexibility was available to present only the controls that are necessary for making measurement choices. The user need not sort through and evaluate every control even if a particular parameter is not meaningful. All switches are clearly labeled and provide direct control over the indicated parameter. There are no multidimension switches whose meaning is determined by the state of other controls (i.e., there are no shift keys).

To simplify measurement setup, the control states of the HP 61015A are remembered for each function. That is, the choices selected by the user for each function are independent of the user's choices for the other functions. When switching between frequency and period measurements, for example, all previously made selections are reestablished when the function is reinvoked and the user does not have to make the selections again. (Appropriate defaults are programmed the first time a function is selected). This feature is in addition to the state file initialization capability that is available within the soft front-panel design of PC Instruments products.

The interactive instrument window display of the HP 61015A for a 10-Hz-to-10-MHz frequency measurement is shown in Fig. 2. The highlighted (dark) area of the function rotary switch indicates the selected function and this switch is visible to allow the user to select another function. The user views the other functions available by touching (on a touchscreen) or pointing to (with a mouse) either of the arrowhead icons to scroll the switch up or down. A new selection is made by indicating the selection box (area between arrowhead icons) of the switch. All controls used on the soft front panel are standard PC Instruments components that operate in the same way on any of the PC Instruments products.

The controls and the measurement parameters they select are clearly marked and only meaningful choices are visible. The selected range of inputs in Fig. 2 is 10 Hz to 10 MHz (no prescaler) and the gate time selection of 1 s provides 1-Hz resolution. If the 10-Hz-to-100-MHz range is selected, the interactive instrument window changes. In this range,

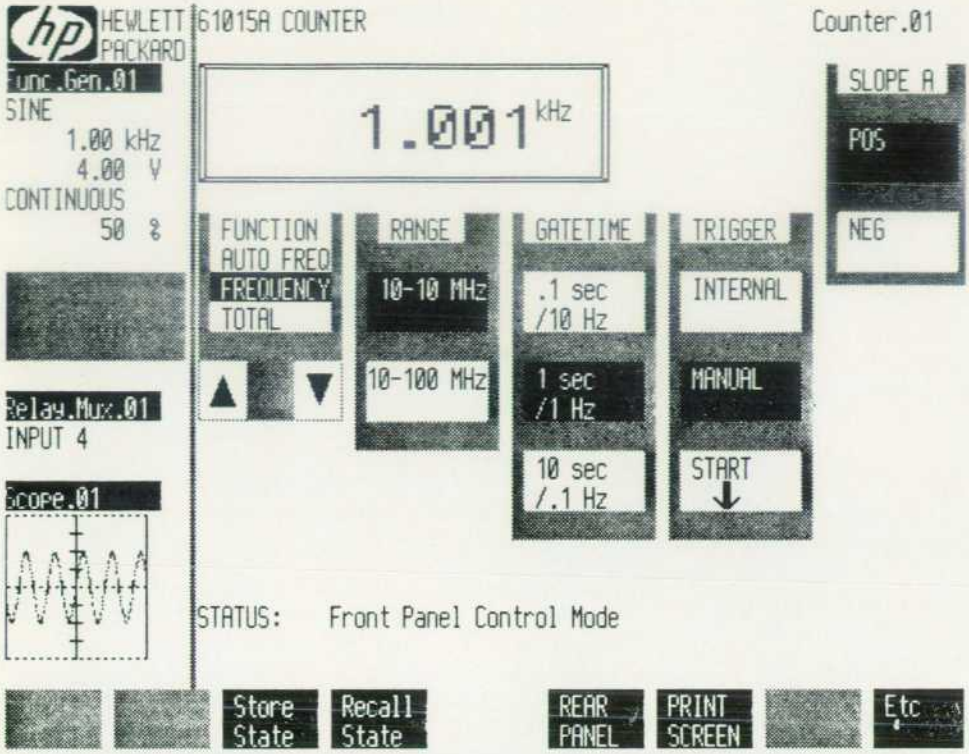

Fig. 2. *HP 61015A frequency measurement soft front-panel display on host personal computer.*

the prescaler is invoked and only two gate times are available (0.1 s for 10-Hz resolution and 1 s for 1-Hz resolution). This illustrates the "you see only what you can control" philosophy of the PC Instruments soft front-panel application software. The autofrequency measurement (Fig. 3) is the most striking example of this minimum display philosophy as applied to the HP 61015A.

The autofrequency/autoperiod measurements also illustrate another design advantage that members of the PC Instruments family have over some of their traditional instrument counterparts—the availability of microprocessors (both in the host personal computer and in the instrument modules) that allow ease-of-use features to be added for little or no additional cost. The traditional method for making frequency measurements is to count pulses for a precise length of time (gate time) based on an internal time base. For example, an instrument might count 1000 pulses in one second and display 1000 Hz. The traditional method for making period measurements is to use the input signal to gate the internal time base oscillator. A 1000-Hz signal input to a universal counter with a 10-MHz time base would gate the time base for 0.001 s. The counter collects 10,000 pulses and displays 1.0000 ms as the period.

These methods have a significant disadvantage. In the frequency example, the measurement has a resolution of 1 count in 1000 or 1000 ppm while the basic accuracy of the time base might be 10 ppm. The resolution of the measurement, then, does not take advantage of the basic accuracy of the instrument. The period measurement is an order of magnitude better with a resolution of 1 count in 10,000 or 100 ppm. In addition, the period measurement is three orders of magnitude faster. The period measurement can be improved even more at the expense of sample time by cycle averaging. With cycle averaging, the time base can be gated for 1, 10, 100, or 1000 cycles of the input to increase the overall resolution of the measurement. If the period of the 1000-Hz signal were to be measured with 100-cycle averaging, the 10-MHz time base would be gated for 0.1 s versus 0.001 s and 1.000000 ms would be displayed. The resolution is now two orders of magnitude greater than before and equal to the basic accuracy of the instrument.

This is the basis for a technique known as reciprocal counting (see box, page 32). Since frequency and period are reciprocal functions, reciprocal counting makes a fre-

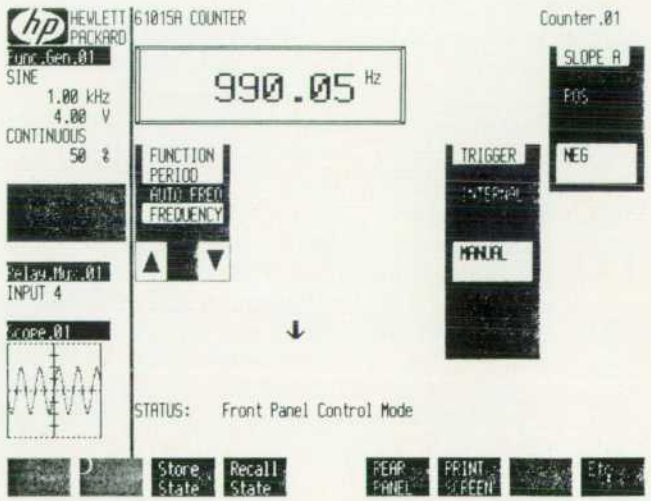

Fig. 3. *HP 61015A soft front-panel display for autofrequency measurement.*

quency or period measurement based on getting the most resolution in a given time. The reciprocal of the measurement is calculated if necessary to display the desired result. For example, suppose a reciprocal counter is set to measure frequency and the input signal is again 1000 Hz. The instrument actually makes a period measurement with its inverse taken to display the resulting frequency. The entire process is transparent to the user and the greatest resolution is always achieved in the minimum gate time. The crossover point between making a frequency measurement or a period measurement is determined by:

$$f_{crossover} = \sqrt{f_r C / t_g}$$

where t_g = gate time, C = input cycles averaged, and f_r = time base frequency (10 MHz).

Programmed Operation

The HP 5316A is a traditional HP-IB (IEEE 488/IEC 625) instrument that implements program codes based on two ASCII-coded command bytes, most of which are followed by a numeric value. Two commands, RE and IN, are available for resetting and/or initializing the HP 5316A. All other commands require a numeric value to specify the parameter to be programmed. For example, to set the function of the HP 5316A to frequency measurement, the user programs the command string FN1. The HP 5316A includes predefined defaults (the frequency function is one) so that if the defaults can be used for all measurements, only the function need be programmed. For many measurement problems, however, the defaults are not appropriate and commands must be used to set the other measurement parameters. Commands are available to set the gate time (GAn), the slope (ASn or BSn), and other measurement states.

A typical command sequence to set the HP 5316A to make single-period measurements (a new reading is taken after the counter is addressed to talk) with a short gate time (500 μs to 30 ms, controlled by a front-panel knob) is FN7WA1GA1. The actual program statement used to output this command string to the counter depends on the controller and language used (wrt720 for the HP 9825A Computer using HPL or OUTPUT 7,20; for Series 80 and HP 9000 Series 200 Computers using BASIC). After the reading is taken, it is returned to the user's program as variable A in response to a red720,A or ENTER 7,20;A statement, respectively.

The PC Instruments syntax is designed to make instrument programming more natural and consistent. Many of the commands are generic and execute identically for every instrument that responds to them. In addition, instrument specific commands are used for those parameters that apply to a specific instrument. The GW™-BASIC commands needed to set the HP 61015A to take single-period measurements with 10-cycle averaging are:

```
CALL SET.FUNCTION(Counter.01,PERIOD)
CALL SET.SAMPLES(Counter.01,R10)
CALL DIS.INT.TRIGGER(Counter.01)
```

The reading is made and returned to the user's program with the statement CALL MEASURE(Counter.01,A).

In addition to the individual instrument control statements, the PC Instruments syntax includes system-wide statements that are designed to increase productivity and ease of programming. Initializing is a good example in that it allows all instruments in the system or individual instruments to be set to states contained in state files. The statement CALL INITIALIZE.SYSTEM(FILENAME$) programs all of the instruments in the system to the states they were in when the state file was created or written to by the soft front-panel application. CALL INITIALIZE(Counter.01,FILENAME$) programs only the named instrument to the state it was in when the state file was created or last written to.

Reciprocal Counting in Firmware

Low cost was a major driving force in the design of the HP 61015A Counter for the PC Instruments product family, but reciprocal counting was a very desirable feature. The HP 61015A Universal Counter is designed around an LSI universal counter IC and an 8049 microprocessor controller. The controller sets the functions and ranges of the universal counter IC and reads the data from the IC after each measurement cycle.

The counter IC in the design represents approximately 30% of the electrical parts cost of the HP 61015A and uses traditional methods for both frequency and period measurements. Most important, however, the counter chip cost less than 25% as much as a chip set that would implement reciprocal counting directly. To have reciprocal counting and the lower-cost chip, an algorithm was developed by which the 8049 controller implements a reciprocal counter in firmware.

The algorithm is set up to allow a maximum gate time of one second. This limit keeps the maximum measurement time including firmware overhead to approximately 1.25 s. The technique begins by setting the counter IC to take a 0.1-s, 100-MHz frequency reading. The result of that measurement is eight BCD digits which are held in memory in the 8049 controller. These digits are analyzed to determine whether a frequency or period measurement should be made. If a period measurement is to be made, additional analysis is done to determine what degree of averaging (1, 10, 100, or 1000 cycles) should be used. The degree of averaging is limited by the one-second gate time constraint. In addition, if the algorithm decides to make a frequency measurement, it is always a one-second gate time measurement.

The 8049 controller sets up the counter IC to take the measurement that has been decided upon and begins the measurement. During this measurement a value is stored in one of the 8049's registers representing the type of measurement being made. After the measurement cycle is finished, the firmware signals the personal computer controlling the system that data is available. When the personal computer takes the measurement data, it also takes the value from the type-of-measurement register to construct the proper display. If reciprocating the result is necessary, it is done in the personal computer, not in the firmware.

Let's look at one example. Take a 1000-Hz signal, connect it to the input of the HP 61015A Universal Counter, and set the counter for autofrequency. The counter will decide to take a period measurement with 1000-cycle averaging. The counter will collect 10,000,000 pulses and set the measurement-type register to indicate that a 1000-cycle period measurement was made. This automatic feature in the HP 61015A makes it the lowest-cost such counter in the HP product line.

Robert V. Miller
Development Engineer
New Jersey Division

Salicide: Advanced Metallization for Submicrometer VLSI Circuits

A self-aligned titanium silicide process can be used to provide lower contact and interconnect resistances in VLSI circuits if one accounts for the effects of impurities, dopant redistribution, phase formation, and grain growth.

by Jun Amano

TO ACHIEVE FURTHER ADVANCEMENT in VLSI circuit technology, higher packing density and improved device performance are continuously required. One of the critical factors to consider in accomplishing this goal in MOS devices is the reduction of parasitic effects such as source and drain series resistance and contact resistance between the metallization and the junctions.[1] In addition, the gate sheet resistance must be reduced at least a few ohms/square to reduce the RC propagation delay of a device. Therefore, the present use of polysilicon contacts and interconnect material with sheet resistances of 30 to 50 Ω/□ cannot satisfy future VLSI requirements. It is clear that new materials for contacts and interconnect metallization must be used for advanced VLSI circuits. Some of the important characteristics for advanced metallization are:

- Low resistivity
- Low contact resistivity
- High-temperature stability
- Low lithographic requirements
- Compatibility with silicon and final metallization
- Above characteristics to be maintained during subsequent high-temperature processing.

Self-Aligned Silicide (Salicide)

Silicon reacts with many metals and forms binary alloys called silicides. The refractory metal (groups IV-A, V-A, and VI-A in the periodic table) silicides have metal-like characteristics with high-temperature stability. The very-low-resistivity silicides satisfy requirements for advanced VLSI metallization. Titanium disilicide ($TiSi_2$) has the lowest resistivity (13 microohm-cm) among the refractory metal silicides and a relatively high melting temperature (1330°C). Another desirable characteristic of titanium is that it can selectively react with silicon to form its silicide at a relatively low temperature (around 600°C).

Using this selectivity, a self-aligned silicide (salicide) process has been developed[2,3] as shown in Fig. 1. In this process, the gate, source, and drain of an MOS device are established first and the side walls of the polysilicon gate are covered by silicon dioxide to prevent silicide formation (Fig. 1a). After these processes, a thin film of titanium is deposited across the entire wafer and a low-temperature annealing cycle is conducted (Fig. 1b). During this annealing step, titanium silicide is formed only on the exposed silicon and polysilicon surfaces (e.g., the source, drain, and gate regions), thus aligning it with the desired areas. The unreacted or excess titanium layer is then etched away. A second annealing step is then performed to form the low-resistivity $TiSi_2$ layer on the source, drain, and gate regions (Fig. 1c).

This titanium silicide process produces not only low-

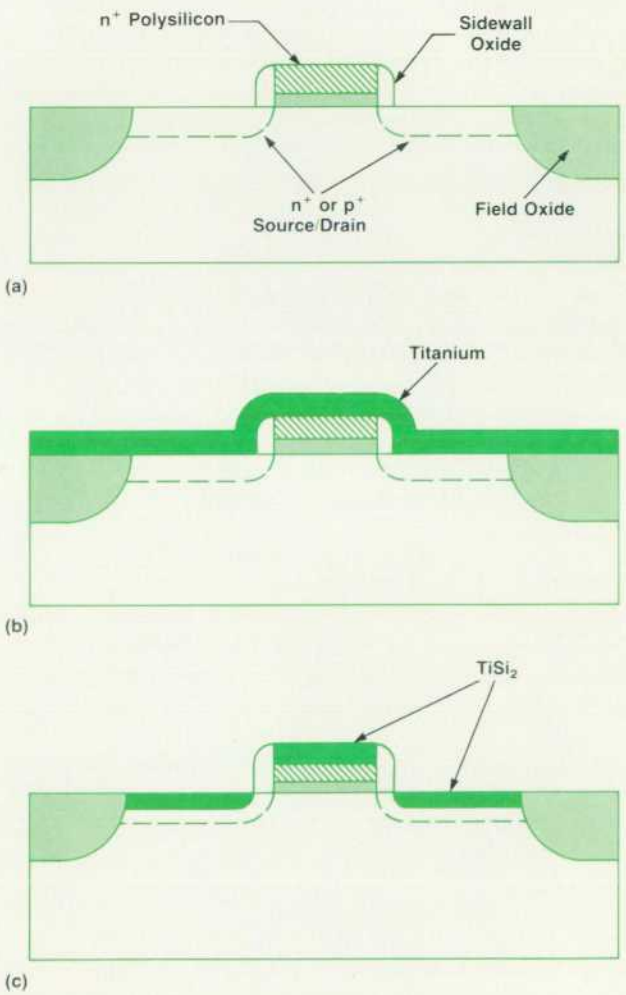

Fig. 1. *Cross-section diagrams of steps for titanium self-aligned silicide (salicide) process.*

sheet-resistivity gate, source, and drain contacts, but also low-contact-resistivity junctions. There are no special lithographic requirements for the salicide process other than a standard MOS masking process to define the exposed silicon areas where the titanium silicide is desired. However, to implement the salicide process as a practical device fabrication process successfully, several critical material problems have to be considered in detail because of the highly reactive nature of titanium.

Oxygen Redistribution

The gas content in the refractory metal film and in the annealing ambient can have a pronounced effect on the silicide growth rate and, in some cases, on the sequence of phase formation. In particular, titanium is highly reactive with oxygen and capable of containing up to 30 atomic percent oxygen.[4] The importance of understanding the role of oxygen in titanium silicide formation lies in the competition of oxygen and silicon for the available titanium. The evolution of oxygen depth profiles is studied by MeV helium ion backscattering (RBS), which provides a nondestructive composition analysis of the thin films, and is correlated with sheet resistivity changes for vacuum-annealed titanium/silicon structures.

In Fig. 2, oxygen depth profiles of the low-temperature annealed samples are shown together with the profile of the as-deposited sample. The horizontal axis is the channel number, which corresponds to the depth scale—around channel 170 and 95 are the titanium surface and titanium/silicon interface positions, respectively. The vertical axis is the backscattering yield, which corresponds to the absolute concentration of oxygen atoms in the films. At annealing temperatures of 350°C and below, most of the oxygen is found within 30 nm of the titanium surface. No changes in the total oxygen concentration are observed in those annealed samples and the as-deposited sample, indicating that oxygen is incorporated in the titanium films during deposition and storage in an air ambient. After a 450°C, 20-minute heat treatment, the surface oxide has decomposed and oxygen has diffused uniformly throughout the titanium film. After this annealing step, the sheet resistivity of the titanium film increases to almost twice that of the as-deposited film.

After annealing at higher temperatures where titanium silicide begins to form, the once uniformly distributed oxygen is expelled (snowplowed) from the growing titanium silicide layer as shown in Fig. 3.[5] The sheet resistivity of the layer also starts to decrease after titanium silicide formation. Once the silicide thickness becomes large enough, its resistivity begins to dominate the combined sheet resistance of the layers. After a 580°C, 150-minute annealing step, the sheet resistivity of the layer is about half that of the as-deposited layer.

The redistribution of oxygen during titanium silicide formation has several implications for IC processing. The expulsion of oxygen from the growing silicide layer is apparently connected with the low resistivity of the $TiSi_2$ layer. This also implies a lower resistivity in reacted layered structures as compared to codeposited films in which the oxygen expulsion process may not take place.

Dopant Redistribution

In the salicide structure, titanium silicide is formed on highly doped n+ and p+ source and drain regions as well as on the polysilicon gate. For VLSI, the source and drain junction depths are very shallow (less than 0.25 μm). Therefore, it is essential to understand the dopant behavior during high-temperature silicide formation so that reproducible dopant profiles can be established and low contact resistance can be obtained.[6,7] The shallow n+ and p+ junctions are fabricated by low-energy arsenic and boron-difluoride ion implantation and annealing. Titanium

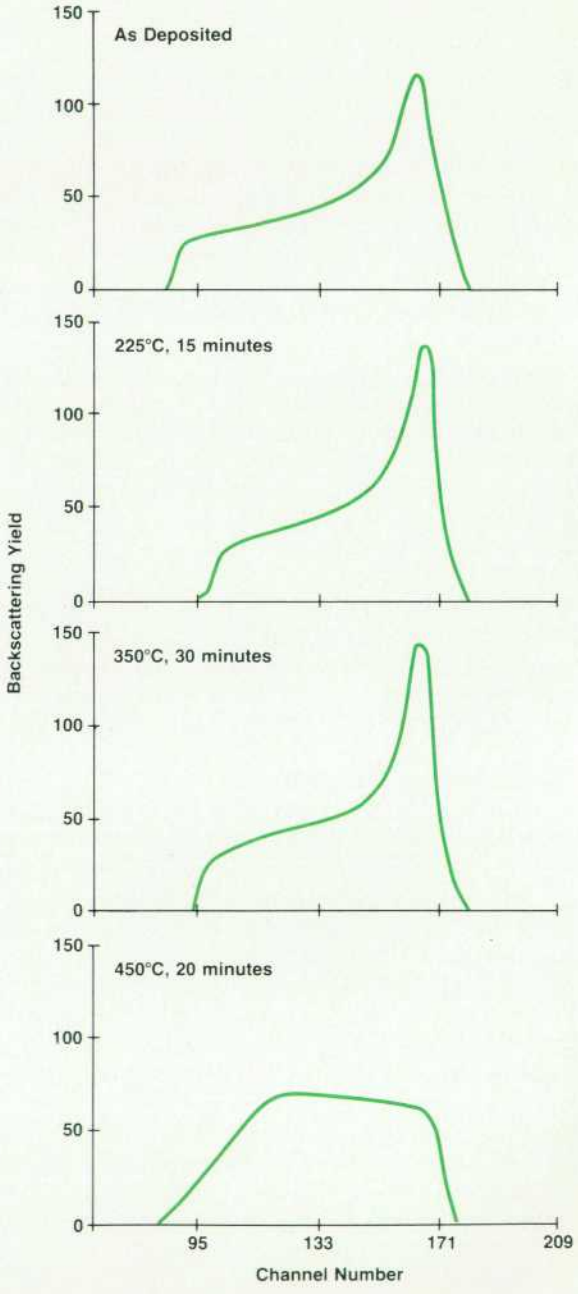

Fig. 2. *RBS spectra of oxygen profiles for as-deposited and vacuum-annealed (at 225, 350, and 450°C) titanium/silicon structures.*

silicides are then formed under ultrahigh-vacuum annealing conditions. The redistribution of implanted arsenic during titanium silicide formation was investigated by RBS. Profiles of the boron and fluorine distributions were obtained by secondary ion mass spectroscopy (SIMS), which provides chemical analysis of the thin films with very high sensitivity.

Fig. 4 shows the RBS spectra of arsenic profiles resulting from different titanium silicide formation temperatures (580°C to 800°C, 30 minutes in vacuum). After a 580°C anneal, the arsenic profile shows a distribution almost identical to the typical implanted range distribution of the as-deposited, nonannealed samples. However, after a 625°C anneal, the arsenic near the titanium and silicon interface is redistributed because of the titanium silicide formation and a second arsenic peak is observed at or near the surface because of the segregation of arsenic. The redistribution of arsenic extends farther into the silicon substrate after a 675°C anneal. Two different titanium silicide compositions are observed by RBS at this annealing stage. After a 725°C anneal, the surface arsenic peak has almost disappeared, indicating a significant arsenic loss from the silicide layer during annealing. The complete conversion of the titanium layer into a single-composition (Ti:Si = 1:2) silicide is also observed at this annealing temperature. After an 800°C anneal, an additional loss of arsenic from the titanium silicide layer is observed and the maximum arsenic concentration in the silicon substrate is decreased by a factor of two compared to that of the as-deposited sample. The arsenic

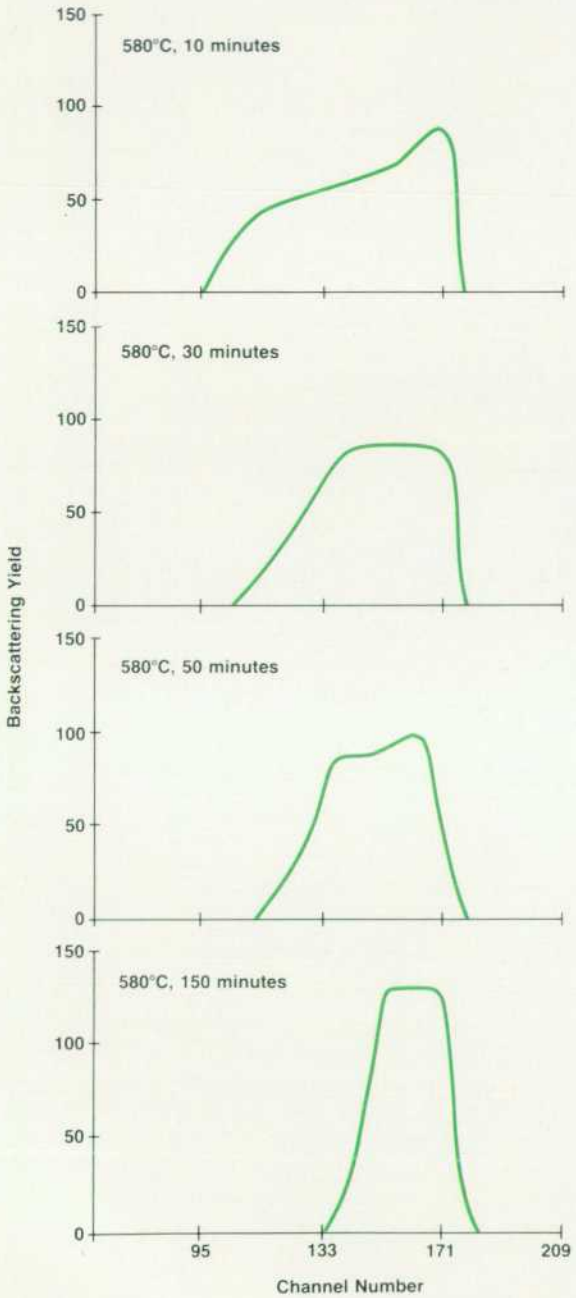

Fig. 3. *RBS spectra of oxygen profiles for isothermally (580°C) vacuum-annealed titanium/silicon structures.*

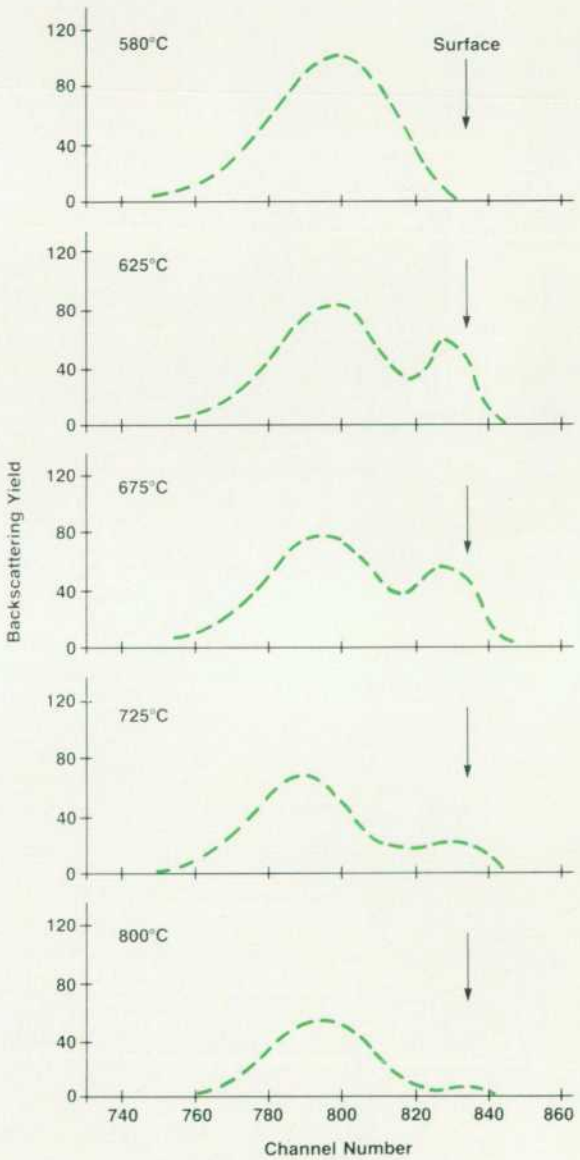

Fig. 4. *RBS spectra of arsenic profiles in the silicon substrate and titanium silicide layer after isochronal annealing in a vacuum.*

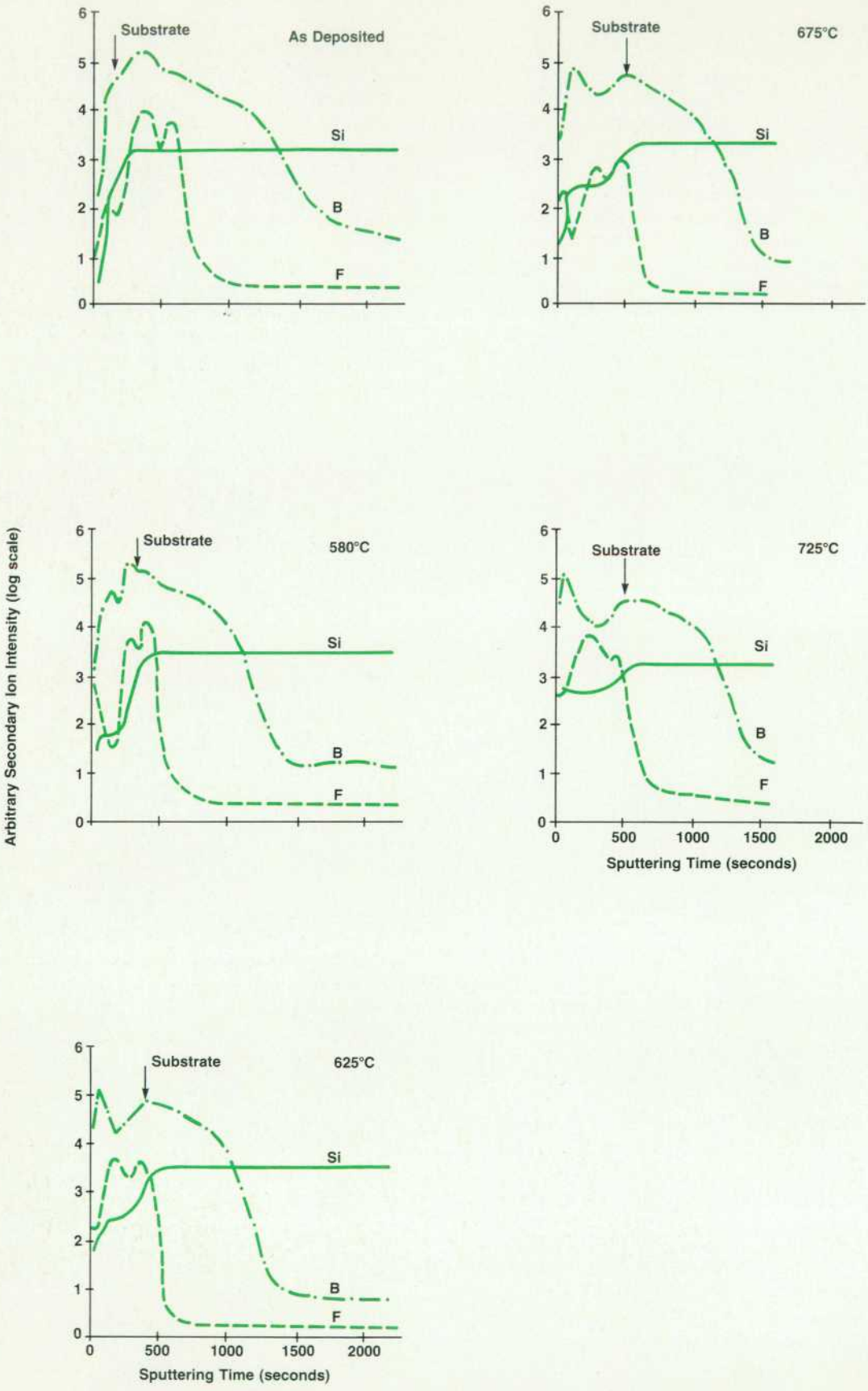

Fig. 5. *SIMS profiles of boron, fluorine, and silicon in the silicon substrate and titanium silicide layers after isochronal annealing in a vacuum.*

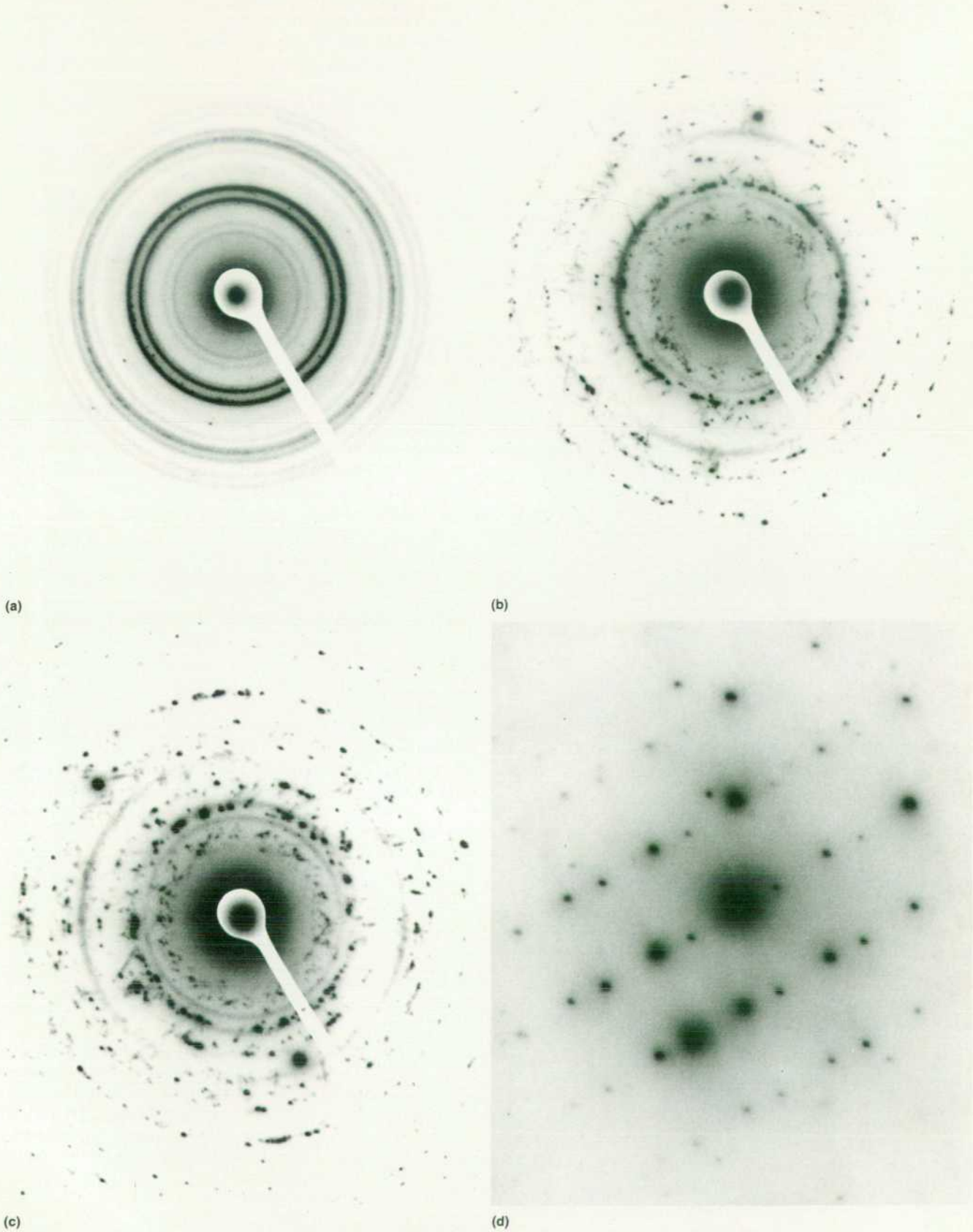

Fig. 6. *Electron diffraction patterns obtained from near the top of the titanium and titanium silicide layer on n + silicon substrate after isochronal annealing in a vacuum. (a) 580°C. (b) 625°C. (c) 725°C. (d) 800°C.*

loss mechanism from the titanium silicide layer consists of the diffusion and evaporation of the arsenic atoms to and from the surface during vacuum annealing.

Fig. 5 shows boron, fluorine, and silicon profiles at different annealing temperatures as obtained by SIMS. The horizontal axis is the sputtering time, which corresponds to the depth scale (i.e., the surface is at the zero sputtering time). The vertical axis is a log scale of the secondary ion intensity, which corresponds to the relative concentration of the elements. A much broader distribution for boron compared to that of fluorine is observed in the as-deposited sample because of the very high diffusivity of ion-implanted boron in silicon. The fluorine distribution shows a typical double peak which is commonly observed for boron difluoride implanted and annealed samples. The interface position between the silicon substrate and the titanium or titanium silicide layers is marked by the arrows in Fig. 5. As the annealing temperature is increased, the interface position moves deeper into the silicon substrate.

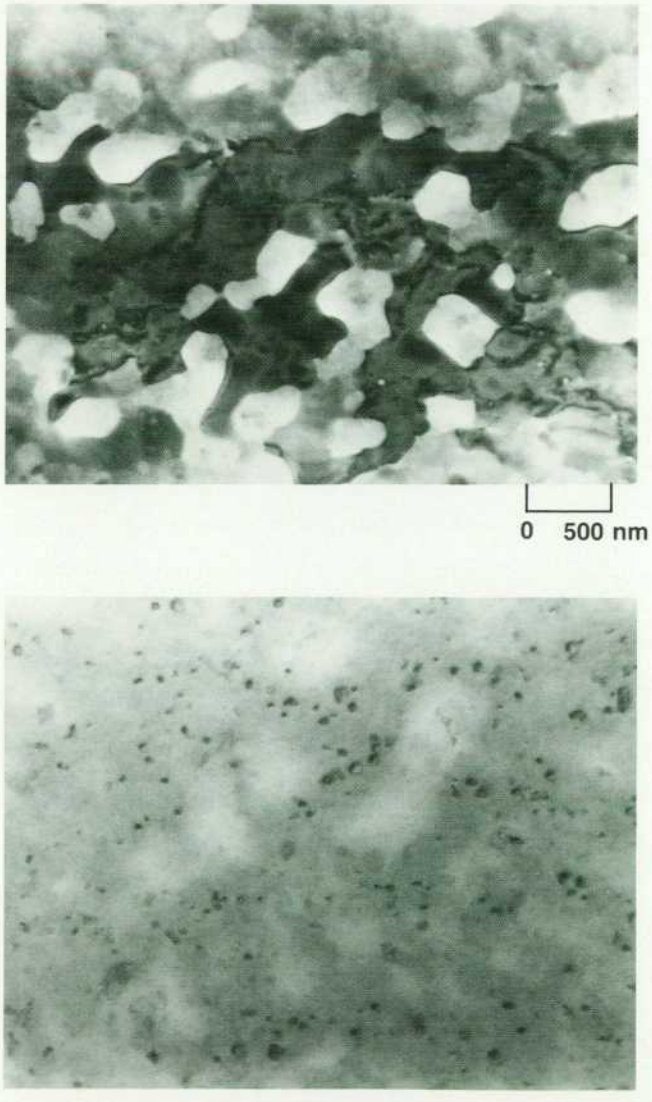

Fig. 7. *TEM photomicrographs of the $TiSi_2$ layer after annealing for 60 minutes at 900°C in a vacuum (top) and the silicon substrate with the $TiSi_2$ layer chemically removed (bottom).*

The redistribution kinetics of boron are quite different from those of arsenic. Boron redistribution occurs when the anneal temperature reaches 580°C. In addition, a step in the silicon signal near the surface indicates formation of the titanium silicide layer at this temperature. After a 625°C anneal, the redistribution of the boron is almost complete, and surface segregation is observed. On the other hand, fluorine appears to be retained in the titanium silicide layer. After annealing at 725°C, the maximum boron concentration in the silicon substrate is decreased by a factor of three compared to that of the as-deposited sample without any significant broadening of the boron profile in the silicon substrate. This indicates that some boron is also lost during the 725°C annealing. The near disappearance of the double fluorine peaks after annealing at 725°C and higher temperatures indicates a uniform distribution of fluorine in the titanium silicide layer.

Phase Formation and Grain Growth

As described in the previous section, the titanium silicide is formed over very shallow junctions. Therefore, silicide phase formation and grain growth have to be well characterized to avoid junction leakage and shorts. Transmission electron microscopy (TEM) is used to investigate the titanium silicide phases and grain structures.

The electron diffraction patterns shown in Fig. 6 were obtained from the titanium and titanium silicide layers on n+ silicon substrates after annealing in vacuum for 30 minutes at various temperatures (580°C to 800°C). The 580°C annealed sample consists of TiO (cubic structure with a_o = 0.418 nm) on a thin silicide layer of either TiSi or Ti_5Si_3 phases. The silicide grain size is on the order of 10 nm, accounting for the broad weak diffraction rings. The 675°C and 725°C annealed samples also exhibit the TiO surface layer. Besides the oxide diffraction rings, the diffraction patterns from these samples contain discrete diffraction spots arranged in discontinuous circles and the amount and relative intensity of the spots increases from 675°C to 725°C. The microstructure of the 675°C annealed sample consists almost entirely of a heavily faulted silicide phase with about 100-nm grain size. This phase is proposed to be a metastable $TiSi_2$ which has the C49 structure.[8] The 725°C annealed sample is found to be a mixture of the faulted phase and the defect-free, stable C54 $TiSi_2$ phase.

After annealing at 800°C, the diffraction pattern exhibits only the stable $TiSi_2$ phase with a relatively large grain size of about one micrometer. The phase development observed by electron diffraction provides good agreement with the composition changes observed by RBS.

After prolonged annealing at 900°C, it is observed that the continuous titanium silicide layer transforms into a discontinuous structure. The formation of a discontinuous structure causes a drastic increase in the resistivity of the silicide layers. Fig. 7 shows TEM photomicrographs of a $TiSi_2$ layer formed on p+ silicon after a 900°C, 60-minute anneal. The dark region in the top photomicrograph is the $TiSi_2$ layer and shows that the layer is not continuous. The bright regions correspond to the exposed silicon substrate adjacent to the $TiSi_2$ layer. Note that there are no dislocation loops observed in the bright silicon substrate regions. The bottom photomicrograph shows the silicon substrate after

the $TiSi_2$ layer is removed by chemical etching. The darker areas with many small dislocation loops caused by the implantation damage correspond to regions where the $TiSi_2$ layers are chemically etched off. These observations indicate:

- The interface of $TiSi_2$ to Si is reasonably flat under the large-grain $TiSi_2$ layer
- A significant amount of silicon is diffused laterally into the $TiSi_2$ grains from the adjacent areas
- Some of the exposed silicon surface may be deeper than the underlying p+ junction. This phenomenon will cause high junction leakage and degradation of the junction's integrity.

Some process limitations for the titanium silicide devices are apparent from the above investigation. The upper limit on subsequent processing temperatures and time cycles should be carefully evaluated to maintain device integrity.

Additional details about the implementation of the titanium salicide process at Hewlett-Packard Laboratories are discussed in references 9 and 10.

Acknowledgments

This study is the result of the combined efforts of several persons in Hewlett-Packard Laboratories and HP's Northwest IC Division. In particular, the author wishes to thank Paul Merchant for close collaboration throughout the entire work, Tom Cass for his excellent TEM studies, Jeff Miller for SIMS analysis, and Tim Kock for some of the sample preparations and electrical measurements.

References

1. H. Shichijo, "A Re-Examination of Practical Performance Limits of Scaled n-channel and p-channel MOS Devices for VLSI," *Solid-State Electronics*, Vol. 26, no. 10, October 1983, p. 969.
2. C.K. Lau, et al, "Titanium Disilicide Self-Aligned Source/Drain + Gate Technology," *Technical Digest of 1982 IEDM*, December 1982, p. 714.
3. C.Y. Ting, et al, "The Use of $TiSi_2$ in a Self-Aligned Silicide Technology," *VLSI Science and Technology/1982*, edited by C.J. Dell'Oca and W.M. Bullis, Electrochemical Society, 1982, p. 224.
4. E.S. Bumps, H.D. Kessler, and M. Hansen, "The Titanium Oxygen System," *Transactions of the ASM*, Vol. 45, 1953, p. 1008.
5. P. Merchant and J. Amano, "Oxygen Redistribution during Sintering of Ti/Si Structures," *Journal of Vacuum Science and Technology*, Vol. B2, no. 4, 1984, p. 762.
6. J. Amano, P. Merchant, and T. Kock, "Arsenic Out-Diffusion during $TiSi_2$ Formation," *Applied Physics Letters*, Vol. 44, no. 8, 1984, p. 744.
7. J. Amano, et al, "Dopant Redistribution during Titanium Silicide Formation," *Journal of Applied Physics*, April 1986.
8. R. Beyers and R. Sinclair, "Metastable Phase Formation in Titanium-Silicon Thin Films," *Journal of Applied Physics*, Vol. 57, no. 12, 1985, p. 5240.
9. D.C. Chen, et al, "A New Device Interconnect Scheme for Sub-Micron VLSI," *Technical Digest of 1984 IEDM*, December 1984, p. 118.
10. S.S. Wong, et al, "HPSAC—A High Performance Silicided Amorphous-Silicon Contact and Interconnect Technology for VLSI," to be published in *IEEE Transactions on Electron Devices*, 1986.

Authors

May 1986

4 PC Instruments

Charles J. Rothschild, 3rd

With HP since 1976, Charlie Rothschild is R&D section manager for HP PC Instruments. In his first HP assignments he worked on the HP 6942A Multiprogrammer and later was project manager for a number of multiprogrammer I/O cards and systems. He was born in New York City and graduated from the University of Michigan with a BSEE degree in 1975 and an MSEE degree in 1976. In addition to his work at HP he teaches microprocessor development at the New Jersey Institute of Technology. Charlie lives in Randolph, New Jersey with his wife and two sons. He is a pilot and amateur radio operator (WB2INB).

Robert C. Sismilich

Bob Sismilich began his HP career in 1979 as a systems engineer in the Paramus, New Jersey sales office. After transferring to the New Jersey Division, he served as R&D project leader for the HP 14750A CAT package, the HP 6944A Multiprogrammer, and the system software for HP PC Instruments. He was also the architect for the soft front panel. He's now project manager for HP PC Instruments software. Bob was born in Jersey City, New Jersey and graduated from Stevens Institute of Technology with a BSEE degree in 1974 and an MSEE degree in 1975. He's married, has two daughters, and lives in Rockaway, New Jersey. He and his wife are active in marriage enrichment and family life programs for their church. His other interests include genealogy, racquetball, and photography.

William T. Walker

Born in Reading, Pennsylvania, Bill Walker was educated at Lehigh University (BSEE 1969 and MSIE 1980) and was a Lieutenant in the U.S. Army Signal Corps from 1969 to 1971. With HP since 1969, he has held a number of research and management positions and most recently was the project manager responsible for the development of HP PC Instruments system software. He is named inventor for a patent related to power supply design and is the author of several technical papers and a book on production and inventory control titled *Managing the Growing Plant*. He's a chapter president of the American Production and Inventory Control Society and is Certified in Production and Inventory Man-

agement by APICS. Bill lives in Summit, New Jersey, is married, and has two children. He is active in his church and in Boy Scout activities, and enjoys reading and dabbling in the stock market.

11 PC Interface Bus

William L. Hughes

A project manager at HP's New Jersey Division, Bill Hughes has been with HP since 1975. He has worked on power supply products, including the HP 6234A Power Supply, has managed a manufacturing engineering group, and was the project manager responsible for the interface, enclosure, and power supplies for HP PC Instruments. His work on a feedback control technique for magnetic power supplies is the subject of a patent. A New Jersey native, Bill was born in Bayonne and now lives in Allamuchy. His outside interests include boating, tennis, racquetball, and the stock market.

Kent W. Luehman

Kent Luehman has been working on products at HP's New Jersey Division since 1975. He contributed to the development of the HP 59501A HP-IB Power Supply Programmer and the HP 6002A Power Supply. Later, he provided engineering applications support for the HP 6940B and HP 6942A Multiprogrammers and more recently was project leader for PCIB design for PC Instruments. He is named inventor on two patents on circuits for an HP-IB power supply and a pending patent on PCIB serial protocol. He's also the coauthor of a 1977 *HP Journal* article. A lifetime resident of New Jersey, Kent was born in Paterson and attended Stevens Institute of Technology (BSEE 1975 and MSEE 1979). He lives in Rockaway with his wife and two daughters. His outside interests include professional photography, woodworking, and camping.

17 Soft Front Panels

Robert C. Sismilich

Author's biography appears elsewhere in this section.

William T. Walker

Author's biography appears elsewhere in this section.

27 HP-IB Command Library

David L. Wolpert

Dave Wolpert, the project manager for the HP-IB command library for HP PC Instruments, has been at HP since graduating from the Georgia Institute of Technology in 1972 with a BEE degree. His work in data acquisition includes contributions to the HP 3495A Scanner, the HP 3497A Data Acquisition/Control Unit, and the HP 3054DL Data Logger. He has also worked on HP-IB code and format standards. A Georgia native, Dave now lives in Loveland, Colorado. His outside interests include music and writing.

29 Case Study: Counter Module

Edward Laczynski

Ed Laczynski is a native of New Jersey, joining HP in 1971 from AT&T Bell Laboratories. He worked on the counter, DMM, and function generator software for HP PC Instruments and is now a project leader for enhancements and support for PC Instruments on MS-DOS systems. His other contributions at HP include a stint as facility information systems manager and work on the design of the HP 6942A Multiprogrammer. Ed was born in Elizabeth and has a BS degree from Rutgers University (1969) and an MS degree in computer science from Stevens Institute of Technology (1977). He and his wife and three children live in Union and enjoy traveling. He's active in Cub Scout and Little League groups.

Robert V. Miller

Bob Miller received his BSEE degree from Rensselaer Polytechnic Institute in 1980 and joined HP the same year. He contributed to the design of the HP 12A, the HP 6031A, and HP 6011A Power Supproducts and developed the hardware and firmware for the HP 61015AA PC Instruments Universal Counter. His work on a reciprocal counting algorithm is the subject of a patent application. Bob was born in Pittsburgh, Pennsylvania, and now lives in Dover, New Jersey with his wife and four children. He likes radio-controlled cars and airplanes, fly-fishing, scuba diving, cooking, and woodworking.

33 Titanium Silicide

Jun Amano

With HP Labs since 1979, Jun Amano is a project manager for the thin-film and interface characterization group in the Materials Research Laboratory. He was born in Tokyo and studied electrical engineering at Sophia University (BSEE 1970) and at the University of Alberta (MSEE 1973 and PhD 1976). His postdoctoral research at Cornell University involved ion beam solid interactions and surface science, and his past responsibilities at HP Labs include thin-film and surface studies using a high-energy ion beam. He is the author of over 30 scientific papers on thin films, surface characterization, and application of ion beams and is a member of five professional societies. A resident of Palo Alto, California, Jun enjoys outdoor sports, especially bicycling, backpacking, cross-country skiing, tennis, and sailing.

Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

May 1986 Volume 37 • Number 5

Technical Information from the Laboratories of Hewlett-Packard Company
Hewlett-Packard Company, 3000 Hanover Street
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529, Startbaan 16
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

Bulk Rate
U.S. Postage
Paid
Hewlett-Packard
Company

CHANGE OF ADDRESS: To subscribe, change your address, or delete your name from our mailing list, send your request to Hewlett-Packard Journal, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Include your old address label, if any. Allow 60 days.

5953-8547