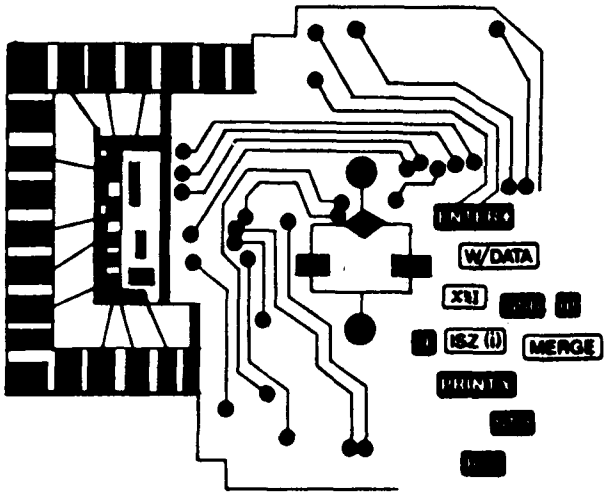

Featuring, this issue:

HEWLETT PACKARD

January-March 1983 Vol. 7 No. 1

HP Key Notes

Editorial

If this section of your *Catalog* looks familiar to you, it should! And if you don't know why it is *here*, you didn't read the notice in the front of this *Catalog*. (See page iv.)

In the last issue (V6N4p13c,14a) of the KEY NOTES *newsletter*, one of those horrible errors that plague editors managed to sneak into the copy during a "cutting and fitting" session. I hate to break the code for routines, especially from a righthand page to the following lefthand page, and that was one time when that attempt to help you resulted in an error. If you haven't found it by now, look at the top, left column on page 14. The second, third, and fourth paragraphs — starting with "For example, solve the two equations:" — should be on page 13 in the righthand column, just below the "LIN" routine listing. Most of you who wrote about this had managed to find the missing paragraphs, but thanks for writing. And, my apologies to **Vasant Patel** for nearly ruining his routine, and to all of you for any inconvenience this caused.

In V6N3p16, we initiated a "Name That Language" contest to establish a name for the "language" used by the HP-41. Because several of the most popular names that were submitted are already being used as trade names by other companies, we've had quite a hard job in finding a winner of that contest and a name for the HP-41 language. However, that problem is now on the way to a solution, and we will soon notify the winner. The winning "name" will be published in August in the first issue of the new *Portable-Computation Guide*.

Although the HP KEY NOTES *newsletter* was discontinued at the end of 1982 (see page iv of this *Catalog*), we still have a supply of back issues. If you are interested in purchasing *any* back issues, just request a copy of the "KEY NOTES Information Sheet," and we will mail it to you. The back issues are very reasonably priced. And if you are a professional, the cost is no doubt tax-deductible.

If you use **Bill Ruderdorf's** excellent routine (V6N4p11b) to obtain the number of bytes in a program, and you are using *only* the X-Functions Module in your HP-41, make sure that your available memory is not *totally full* or you could run into some trouble. Otherwise, it works extremely well.

In August you will see the first issue of our new *Portable-Computation Guide*. I am looking forward to editing this excellent new publication, and I promise that you will be very pleased with it. It was somewhat saddening to see the KEY NOTES *newsletter* discontinue, but economics forced that issue. I will continue to also edit the KEY NOTES "section" of the new *Guide*, so don't stop your excellent contributions by way of your letters. Believe it or not, I do read every letter. Letters to the editor should be sent to:

Henry Horn, Editor
HP PORTABLE-COMPUTATION GUIDE
Hewlett-Packard Co.
1000 N.E. Circle Boulevard
Corvallis, OR 97330 U.S.A.

We cannot guarantee a reply to every letter, but we do guarantee that every letter will be read by the editor, and that as many as possible will be answered in KEY NOTES or in a personal response. Be sure to put your address on the letter; sometimes the envelope gets lost.

Lastly, I regret to inform you that my assistant, and the technical editor of the KEY NOTES *newsletter*, **Ted Wadman**, has left Hewlett-Packard to complete his college degree and then spend some time in the Peace Corp. I shall surely miss him and his excellent contributions to KEY NOTES, but I wish him much success in whatever he does, no matter wherever he goes. I have the strangest feeling that we'll "see" him in KEY NOTES again — soon.

Until August . . . may all your problems be programmable or keystroke solvable.

New Product News

Elsewhere in this issue is a description of the new HP 82183A Extended I/O Module for the HP-41 System, so we won't repeat that here. But if you own — or are thinking of buying — the HP-75C Portable Computer, here is some good news.

On March 1, 1983, Hewlett-Packard announced the HP-75 Text Formatter software, which comes as a module that plugs into the HP-75. This plug-in module provides software in read-only memory (ROM) that does not diminish the HP-75's user memory (which is a maximum of 24 kbytes).

The HP-75 Text Formatter features such formatting options as indented paragraphs and justified text, and also makes available such capabilities as distribution lists, merged text, and plotter-produced text slides.

With this new software, you can quickly and easily type and format a memo or letter — anytime, anywhere. And after the memo or letter has been typed into the HP-75, the text formatter provides embedded commands to format the printed documents as they are edited. Formatting options include setting margins, setting lines per page, indenting paragraphs, left justifying, and left-and-right justifying. Then, with the distribution-list command, a letter can be customized by inserting names and addresses anywhere in the letter.

The merge-file command lets separate text, such as an often-used paragraph, be added to the new letter. And with the slide command, text can be sized and slanted to be plotted on paper or acetate for presentations and reports.

The text formatter also includes a "help" facility. If a user forgets a command, the touch of a key will list all the commands and their purposes.

The HP-75's keys can be redefined as the embedded commands. The redefined keys are highlighted by a custom overlay that comes with the HP-75 Text Formatter. It fits on top of the keyboard and relabels the redefined keys.

Corvallis Library Corner

Because of the Library Contest, program submissions have been nothing short of staggering. So if you have submitted a program in the past few months and are wondering why it is taking longer than usual to hear from us, please be patient a bit longer. At the time this was written, we are still a few months behind in trying to catch up with the avalanche of programs you submitted. But we *will* clean up the backlog and, when the new quarterly *Guide* (see page iv) begins, we hope to totally eliminate such problems. So please bear with us a bit longer, because we are certain that everyone will profit in the long run.

WINNING PROGRAMS

It is not always possible to coordinate printing schedules with Contest results, so we are quite a bit behind in announcing the following Contest winners and their winning programs. However, winners *are* notified by telephone or mail, even though we don't always promptly report on them in print.

The winning programs for July and August are listed below. All of these are included in the *Catalog*, so we haven't included their abstracts here.

JULY CONTEST WINNERS

#02014C Ultimate Beam Strength
by S. F. Dusterwald

#02051C Short Cylinder Analysis
by S. A. Porter

#04843D Estimating Endurance and Fuel Consumption for a Ship
by J.E. Baskerville

#02056C AAS — Data Analysis — Report Generator for Bench Top
by Chan-Fai Chong

#02080C Sun Ephemeris
by J. J. Lynch

#02003C Short Circuit Analysis for 1-9 Buses
by B. B. Kahn

#02058C AISC Structural Steel Column Design
by P. K. Belcher

#02004C Metering Orifice Program for Flange Tap Square-Edge Orifice
by H. K. Deakins

#02079C Bridge Deck Elevations
by S. L. Stroh

#02064C Wedge Failure Analysis
by T. Langland

AUGUST CONTEST WINNERS

#02114C Integrated Cost/Task Schedule
by C. Lamar Williams

#02140C Complete Fanning (or Moody) Friction Factor Chart
by Mark G. Pool

#02107C Sectional Cone Fabrication
by Gary C. Lizalek

#02083C Multicomponent Distillation Column Design
by Robert J. Wooley

#02081C Design of Fillet Welds
by A. Jayaraman

#04881D Estimating Fuel Load Requirements for a Ship
by James E. Baskerville

#04856D Quality Control Program for R1A
by Mike McDonald

#02111C Spectrum Analysis by the Maximum Entropy Method
by Stephen C. Kenney

#02139C Three-Phase Distribution Networks
by Ernesto M. Vazquez

#02163C Physical Growth Percentile
by Peter C. Jensen

Congratulations to all twenty authors for their fine work. For both July and August, the first two winners listed received either an HP 82161A Digital Cassette Drive or an HP 82162A Thermal Printer/Plotter. The other monthly winners received an HP 82180A Extended Functions/Memory Module or comparable module.

PROGRAMMER'S REFERENCE GUIDE

Promised long ago but delayed for reasons beyond our control, this 72-page book is now available from the Corvallis Users' Library. If you are a member of the Corvallis Users' Library, a copy was sent to you either in your initial membership package or with this *Catalog*.

For non-members, the *Programmer's Reference Guide* is available by mail from the Corvallis Users' Library for $12 postpaid to the U.S. and Canada and for $14 postpaid to all other countries. The book includes a description of what the Library is all about, plus a section on "Programming — With the User in Mind," and a section that discusses Devices (including peripherals) and Functions. Section III is a detailed listing of our Review Standards and Section IV presents Sample Documentation. Also included are two appendices: one for ROM identification numbers and one that clearly shows an example of program listing comments — and why they are so important in documentation.

If you intend to submit programs to the Library and want your submissions to compete for the many excellent prizes we offer, this book is an excellent investment. Also, its cost probably is tax-deductible.

New HP-41 Extended I/O ROM Introduced

Hewlett-Packard released to its Dealers a new **HP 82183 Extended I/O Module** on February 1, 1983. By the time you read this, the new module should be readily available. But what is an extended input/output module, you ask? Well, this one contains 59 HP-41 functions that extend the HP-41's control over HP-IL beyond the capability provided by the base HP-IL module. And since a lot of you are going to wonder what this new module can do for *you*, herewith is a detailed explanation of the module and its functions.

GENERAL DESCRIPTION

The I/O module's function set is divided into four general classes (each under a separate CAT 2 header): mass storage functions (–X MASS 1A), character-manipulation functions (–X EXT FCN — "extended extended functions functions??"), HP-IL control functions (–X CTL FNS), and advanced control functions (–ADV CTL FN). The character-manipulation functions are stand-alone functions; all other I/O module functions require the presence of the HP-IL module.

The definition of the I/O module has changed somewhat from the time of its initial announcement in December of 1981. The central purpose of the module is to provide the HP-41 with the ability to control all present and future HP-IL devices in such a way that programmers normally need not concern themselves with such low-level HP-IL overhead as auto-addressing, frame error-checking, talking or listening devices, etc. The mass storage functions are designed to work only with mass storage devices — the other functions in the ROM are not restricted to any particular HP-IL device class. Consistent with this definition, the HP 82162A barcode printing functions are not included — they are provided in the HP 82164A Plotter Module (see KEY NOTES V6N4p4).

MASS STORAGE FUNCTIONS

The mass storage functions include mass duplication functions and directory information functions. COPYFL copies a single file, specified in the ALPHA register, from the primary mass storage device to a second mass storage device at the HP-IL address specified in the X-register. MCOPY copies the entire contents of the primary medium to all other mass storage devices on the loop regardless of their addresses. MCOPYPV is the same as MCOPY, except that programs on the duplicate media are made private. Finally, MVERIFY verifies the recording on a user-specified number of records on all mass media on HP-IL. Directory information functions return information from the primary mass storage device directory that are necessary for programmatic manipulation of medium files. DIRX returns to the ALPHA register the filename of the "Xth" directory entry. FLTYPE returns a code representing the file type of the file named in ALPHA; FLLENG returns the file length.

CHARACTER-MANIPULATION FUNCTIONS

The HP-IL module uses the ALPHA register as an "I/O buffer" — a temporary storage register for strings of bytes to be transmitted on or received from HP-IL. The I/O module follows this convention, but provides a more extensive function set to deal with the transmission, creation, and deciphering of ALPHA strings, including the elusive null characters. The character-manipulation functions include ATOXL, ATOXR, XTOAL, and XTOAR, whose names suggest their operation. ATOXR, for example, removes the rightmost character from the string in ALPHA, and places its decimal character code into the X-register. XTOAR reverses that process. (ATOXL and XTOAR are identical to the X-functions module's ATOX and XTOA, respectively.) ALPHA/X interchange is extended further by the functions ATOXX and XTOAX, which exchange characters between the X-register and arbitrary positions in the ALPHA register. Both functions operate on a character position in ALPHA specified by a number in X. A positive character position is counted from left-to-right, starting with the first non-null character in ALPHA (position 0). A negative character position is counted to the left from the rightmost position of the displayed "word" in the ALPHA register, independent of its contents. Thus you

can change, for example, the word "STING" to "STUNG" by placing the character code for "U" (85) in the Y-register, then executing YTOAX with either 2 or −3 in X.

The function X◄►F is identical to X◄►F in the extended functions/memory module. It enables you to test or change individual bits in an 8-bit byte. ALENGIO (identical to ALENG in the X-functions module) returns the length of the current ALPHA string to the X-register. The final character-manipulation function is ANUMDEL, which like ANUM, finds an ASCII-coded number in the ALPHA register in ALPHA and returns its value to X. ANUMDEL differs from ANUM in that it also deletes all characters in the ALPHA string up through the last numeric character used from the string to form the number placed in X.

HP-IL CONTROL FUNCTIONS

The largest group of functions in the I/O module is the HP-IL control functions. This group can be further divided into loop configuration functions, device control functions, and data transfer functions.

Loop configuration functions give the HP-41 the ability to automatically determine the number, type, and statuses of all devices currently connected on HP-IL. RCLSEL recalls the address of the primary device. (The device ID is an ASCII string, like "HP 82165A", that identifies a specific device, whereas the accessory ID is a one-byte numeric code that identifies a type of device, like "mass storage", accessory ID's from 16 to 31, or "display", ID's from 48 to 65.) FINDAID returns the HP-IL address of the first device found on the loop (starting from the primary device) whose accessory ID corresponds to the number in X. (If the number in X is negative, only the first 4 bits are matched, so that you can search for a device class rather than a specific device.)

The remaining loop configuration functions deal with the service request needs of loop devices. SRQ? is an HP-41 conditional that tests true if any device on the loop is signalling a service request. STAT reads the status bytes from the primary device into the ALPHA register, where they can be deciphered using the character-manipulation functions. (INSTAT looks only at the first byte of status.) Four functions implement parallel polling: POLLE (parallel poll enable the primary device), POLLD (parallel poll disable), POLLUNC (parallel poll unconfigure — disable parallel poll response of all devices), and POLL (send the Identify frame, and return the value of the 8 parallel poll bits to X).

The I/O module device control functions extend the partial set (LISTEN, LOCAL, PWRDN, REMOTE, TRIGGER) of the HP-IL command frame functions provided in the HP-IL module. Each sends a particular HP-IL command frame:

CLRDEV — clear the primary device (SDC)
CLRLOOP — clear all device (DCL)
DEVT — send a device-dependent talker message (DDTx)
DEVL — send a device-dependent listener message (DDLx)
LOCK — lock-out manual controls on primary device (LLO)
NOTREM — send Not Remote-Enable message (NRE)

As with all HP-IL module and I/O module functions (except data transfer functions in addressing-off mode — see below), the specific command frames are embedded in a stream of frames sent by the functions. The additional frames take care of loop addressing, looking for a printer for tracing, designating talkers and listeners, etc.

Data transfer functions control the transmission of data messages on HP-IL. The functions group into three categories:

INAC	OUTAC	XFERC
INACL	OUTACL	XFERCL
INAE	OUTAE	XFERE
INAN	OUTAN	XFERN
INP	OUTP	XFER
INXB	OUTXB	

The various parts of the function names translate as follows:

IN ... data input from the primary device.
OUT ... data sent to the primary device.
XFER ... data transferred between two devices.
...A... data sent to or from ALPHA.
...XB one byte sent to or from X.
...P HP-41 program sent to/from memory.

The INA..., OUTA..., and XFER... functions have a final postfix that determines the manner in which the corresponding data transfer is terminated:

...C terminate when a character specified in X is transferred.
...CL terminate on a carriage return/linefeed.
...E terminate on an End frame.
...N terminate after N (specified in X) characters are sent.

Thus, for example, OUTACL sends the contents of the ALPHA register to the primary device, terminating the transmission with a carriage return/linefeed. XFER (no postfix) terminates on an end-of-transmission (ETO) frame sent by the talker.

The INA... and XFER... functions will continue to request data from the primary device until the specified termination condition is met, even if the device sends its data in short blocks ending with ETO frames. An INA...function will also terminate if the ALPHA register fills up, in which case flag 17 is set to indicate that the desired termination condition was not satisfied. (When the condition is met, flag 17 is cleared.) Note that the HP-IL module function INA does not have this property — it will halt upon receiving an ETO.

The problem of nulls in the ALPHA register is handled in the following amusing manner. The INA... functions (and STAT) clear the ALPHA register, place a single non-null "dummy" character (a "D" for INA...; and "S" for STAT) into the register, then append the data bytes input from HP-IL. The dummy character serves to mark the beginning of the input string, even if the string has one or more leading nulls, which would be invisible if the dummy character were not present. Similarly, the OUTA...functions send the contents of the ALPHA register, starting with the next character after the first non-null character is not sent, and thus serves only as a marker showing the start of the string to be transmitted. All nulls following the marker are transmitted in their proper order in the string.

INP and OUTP transfer HP-41 programs directly to and from HP-41 user memory. OUTP, for example, transmits the program containing the global label designated in the ALPHA register, as a string of ASCII characters representing the hexadecimal program bytes. ASCII characters 0-F are used (at a price of two bytes sent for every actual program byte) so that OUTP and INP will work through interfaces such as modems, which may interpret certain bytes as handshake signals. The program bytes are preceded by four bytes that represent the program length, and followed by a 2-byte checksum. INP expects to receive a data stream in the same format. These functions differ from READP and WRTP in that INP and OUTP require the send/receive device to be configured explicitly (using other I/O module functions) to send or receive the program bytes, whereas READP and WRTP are designed only for mass storage devices, and automatically position the medium and send the appropriate read/write commands.

ADVANCED CONTROL FUNCTIONS

Data transfer functions work in two modes, controlled by flag 34. In "addressing-on" mode (set by executing ADRON, which clears flag 34), data transfer functions work like all the other HP-IL module and I/O module functions, automatically taking care of loop overhead. In "addressing-off" mode (set by executing ADROFF, which sets flag 34), only the actual data transfer is controlled by the functions. No loop addressing or talker/listener designation is performed. The advanced control functions are therefore provided to permit the user or his program to take care of these operations explicitly.

The advanced control functions are DDL, DDT, LAD, SEND, TAD, UNL and UNT, plus ADRON and ADROFF. SEND transmits the HP-IL command frame corresponding to a number from 0 to 255 in the X-register. Only that frame is sent, followed by the obligatory ready-for-command frame. Device-dependent listener (DDL), device-dependent talker (DDT), listen address (LAD), talk address (TAD), unlisten (UNL), and untalk (UNT) are the most frequently used commands, so the functions with the corresponding names are provided to aid in program legibility even though the commands could be sent with SEND.

The advanced control capability is necessary because certain devices change their internal states when they are addressed as a talker or listener. So, for example, we can't use 0 DEVT 19 INAN to read the control registers of the HP 82165A HP-IL/GPIO interface (assuming that the interface is the current primary device), because the DEVT function sends the untalk frame after the DDTO. (DDTO instructs the interface to send the contents of its 19 control registers rather than of its transfer buffer when it receives the next Send Data frame.) Then when INAN makes the interface a talker again, the effect of the DDTO is cancelled, and the interface will send transfer buffer data instead of the control register bytes. The correct sequence is ADROFF RCLSEL TAD 0 DDT 19 INAN UNT ADRON. (The RCLSEL TAD makes the primary device a talker.)

IN THE MANUAL...

In addition to shorter examples illustrating the use of the I/O module functions, the Extended I/O Module Owner's Manual contains listings (and barcode) for a complete loop catalog program and for a set of programs that fully automate the process of transferring programs between two HP-41's via HP 82168A Acoustic Couplers (modems). The loop catalog program illustrates the use of loop configuration functions. It prints (on the first printer or display device) a list showing the number of devices on the loop, the currently selected address, and each device on the loop including the device address, accessory ID, device ID, and device class.

The program-transfer routines, while specific to the modems, are illustrations of the use of the various data-transfer functions, and can be used as prototypes for programs to be used with other interfaces. Using these programs, two users can exchange programs by telephone, where they establish voice contact, then connect their receivers to the modems and execute the transfer programs. All of the modem handshaking is performed automatically.

"Nantucket" Revealed

In the last issue of KEY NOTES (V6N4p3), we told you that, in the next issue, we would reveal how **Robert (Keystroke) Gardner** created his "Nantucket" picture on his stock HP 82162A Thermal Printer/Plotter. And we *still* want to tell you all about Mr. Gardner, but that will have to wait for a few more months. We'll print an article about him in the first *Portable-Computation Guide* (see page iv).

Actually, it is pretty difficult to create the "Nantucket" scene. First, you need an HP-41, an HP-IL module, and the 82162A printer — plus lots of patience. We can't reproduce in this limited space his sketches and examples, but we can "show" you his letter. It will tell you what it takes to "paint" pictures with the 82162A printer.

Here is the material that I promised you. I think that it will show how I created Nantucket. As you can see, it takes two long programs to print it: Nan 1 and Nan 2. I have included a printed listing of each. I hope that you have a Cassette Drive or a Card Reader so that you can store them. Also, you will need an HP Interface Loop and the new 82162A printer. If you assign ACCOL, SKPCOL, and PRBUF to your top row of keys, the keying-in goes quite rapidly.

Also inclosed is a copy of the upper, left quarter of the layout that I used. The full layout is 84 squares wide and 103 squares long. It is divided vertically in columns of ten with the extra four at the end. Horizonally, it is divided into rows of 7 squares and 5 squares, alternately. The rows of 7 are Nan 1 and are numbered 01 to 09. The rows of 5 are Nan 2 and are marked with an X. I then sketched in the drawing lightly in pencil and inked-in the appropriate squares with a red felt-tip pen. If you compare row 01 with LBL 01 of Nan 1 you can see how they match.

The printing is quite difficult and is not always successful, as you can see by one of the samples. I have included a separate print of Nan 1 and Nan 2 so that you can see the results of each program. Also, there is a successful print of Nantucket.

To print the programs, I first print a solid row of X's and pull out about 20 inches of paper and cut it off. I remove the roll from the printer. I cut off the excess paper right at the top of the X's as in the sample. I then feed the paper back into the printer until the leading edge is even with the plastic tear-off bar. I then print 5 copies of Nan 1, advancing the paper 8 steps between each print. I then remove the paper and trim off the X's at the bottom of the X's, as indicated by the dotted line on the sample. I then proceed to print Nan 2 in the same manner except that I advance the paper 12 steps between each print. It is best to trim off a little too much paper rather than not quite enough for the second printing as the paper can be adjusted by pulling it backwards through the printer. It cannot be adjusted forward. There is nothing else I can tell you about the printing. You just have to work it out. It can be very difficult at times. At other times it goes wonderfully well. Good luck . . .

Mr. Gardner now has an HP-75C Portable Computer and is well on his way to making it "sing." Two months ago, he sent us a cassette tape with four musical arrangements, but the first one is a mind-bender; it's a fantastic rendition of "Flight of the Bumblebee" in 10,955 HP-75C bytes. And when you read the article about him in August, you'll learn why we think R. L. (Keystroke) Gardner is a most unusual person.

An Unusual Book . . .

Most HP-41 Handheld Computers are used for fairly common applications or to solve fairly common problems. But this marvelous and versatile machine has virtually no limits to the *types* of applications it can be called upon to pursue. Consider, for example, the unusual problems confronting a hand surgeon. A severely impaired human hand can cause various degrees of disability, and insurance companies must determine to what degree that disability will affect the patient's life or ability to earn income.

Charles F. Johnson, M.D., of Pawtucket, Rhode Island, has written a truly remarkable book entitled: *Impairment Rating of Hand Function*, and it is a superbly documented program for the HP-41CV. The program (book) determines impairment of hand function based on guidelines suggested by the American Medical Association's Committee on Rating of Mental and Physical Impairment. These guidelines have become the mainstay reference for the active hand surgeon.

After an injury, the patient regains hand function over a variable period of time. When the patient no longer continues to improve, he or she reaches an "end result." Hand function is then evaluated and an impairment rating is given for permanent impairment of function of that hand and associated upper extremity. Legally, workman's compensation laws provide that a physician must make this evaluation. "Disability," not to be confused with "impairment," is a determination made by the courts under provision outlined by law.

The value of the "Impairment Rating of Hand Function" program is that it provides a reference for calculating impairment of hand function. This is particularly useful in the case of a complex hand injury involving several combinations of amputation, fusion, decreased range of motion, and sensory loss of digits. The purpose of the program is not to supplant the "paper and pencil" approach to the calculation of hand impairment, but to provide a standard for checking results. In this frame of reference the program will also be useful to the insurance claims examiner or attorney.

All in all, this book by Dr. Johnson is a brilliant exercise in fully using the power of the HP-41 System. The book is also a unique contribution to *programming* for the HP-41 in a medical area for which no programs are available. Even more astonishing is the complete *range* of programming. The program can be run using any of three system set-ups: (1) programs stored on magnetic cards; (2) ROM-stored programs; and (3) cassette-stored programs.

If you are interested in further details or want to purchase the book or the program, get in touch with:

Plastic and Reconstructive Surgery, Inc.
333 School Street
Pawtucket, Rhode Island 02860

As for Dr. Johnson, he is not only a superb HP-41 programmer but also an actively practicing plastic surgeon. He obtained his A.B. degree from Princeton University (1960) and in 1964 graduated from the Johns Hopkins University School of Medicine. He is certified by the American Board of Surgery and the American College of Surgeons. He is also a Fellow in the American College of Surgeons and an active member of the American Society of Plastic and Reconstructive Surgeons, the New England Society of Plastic and Reconstructive Surgeons, and the American Association for Hand Surgery. Excellent credentials, you no doubt agree, but wonder how a hand surgeon writes such esoteric programs. Actually, it is because Dr. Johnson has a long-standing interest in mathematics, and he is an HP-41 System enthusiast of the highest order. He is member #6928 of the national PPC and is a member of the Rhode Island Chapter of PPC.*

If you have any interest in the area of hand function impairment rating, be it from a medical, legal, or insurance basis, here is the definitive answer to your questions.

It might be a narrow field, Dr. Johnson, but we congratulate you for a superb job of using the HP-41 System to accurately solve a vexing problem.

**For more information about the independent users' club, PPC, and a sample issue of the club's newsletter, send a self-addressed, large (folded) envelope (9x12 inches; 23.8 x 30.5 cm) with first-class postage for 2 ounces (56.7 grams) to: PPC CALCULATOR JOURNAL; 2545 West Camden Place; Santa Ana, California 92704 U.S.A. If you live outside the U.S., make sure you include a legible address label and international postal coupons for 56.7 grams (2 ounces). A letter is not necessary and will only slow the response.*

Routines, Techniques, Tips, Etc...

The routines and techniques furnished in this column are contributed by people from all walks of life and with various levels of mathematical and programming skills. While the routines might not be the ultimate in programming, they do present new ideas and solutions that others have found for *their* applications. *You* might have to modify them to fit your personal applications.

HP-41 X-FUNCTIONS MODULE

Because there is a great interest in this relatively new module for the HP-41 System, we are starting this column with some interesting inputs that will get the HP 82180A Extended Functions/Memory Module off to a good start in solving your problems.

Our first input is from sunny Aruba, an island of 69 square miles (178.7 square km) in the Netherlands Antilles off the coast of Northwestern Venezuela. It was contributed by **Dr. J. M. J. van Lis.**

(41) I would like to submit the following routine for your "Routines, Techniques, Tips, Etc." column. It can be used to assign labels, using ALPHA characters, to keys with corresponding row and column numbers. It works as follows:

```
01♦LBL "ASN"
02 CF 29
03 FIX 0
04 CLA
05♦LBL 00
06 31
07 GETKEY
08 X≠Y?
09 GTO 01
10 GETKEY
11 CHS
12♦LBL 01
13 ARCL X
14 PASN
15 CLA
16 STOP
17 GTO 00
18 END
```

Thus, by pressing the COS key, LBL "24" is assigned to this key. LBL " – 24" is assigned to the shifted COS key by pressing the shift and COS keys successively.

Finally, I want to ask you if it is possible to publish the KEY NOTES more often. They are my daily vitamins!

(Almost everyone who reads KEY NOTES would like, as you say, a more regular "dose of vitamins." However, Dr. van Lis, there is a point where cost-effectiveness enters into the formula — or diet — and as you will see on page iv of this Catalog, we have resorted to "multiple vitamins" rather than more single doses in order to somehow keep KEY NOTES nourishing you — Ed.)

Back in Ankeny, Indiana, **Ed Keefe** has been busy writing the following "translation" routines for his HP-41 with X-functions module.

(41) I know you've seen a plethora of base-translation routines, but the use of ALPHA operators from the X-functions module makes these routines a little more interesting. Needless to say, the HP-16C will do the following much more elegantly, but just in case one is not available . . .

```
01♦LBL "XD"
02 CLX
03♦LBL 01
04 ATOX
05 X=0?
06 GTO 02
07 55
08 MOD
09 48
10 MOD
11 +
12 RCL 06
13 *
14 GTO 01
15♦LBL 02
16 RDN
17 RCL 06
18 /
19 RTN
20♦LBL "DX"
21 CLA
22♦LBL 03
23 RCL 06
24 /
25 ENTER↑
26 FRC
27 RCL 06
28 *
29 1 E1
30 X<=Y?
31 GTO 04
32 48
33 GTO 05
34♦LBL 04
35 55
36♦LBL 05
37 X<>Y
38 RDN
39 +
40 XTOA
41 -1
42 AROT
43 RCL Z
44 INT
45 X≠0?
46 GTO 03
47 AVIEW
48 END
```

INSTRUCTIONS: LBL "XD" will translate a number (written in any base from 2 to 37) in the ALPHA register to a decimal number in the X-register. LBL "DX" takes a decimal number in the X-register and translates it to a base number in the ALPHA register and in the display. The base must be stored in register 6. (Register 6 is arbitrary and may be changed to suit the user.)

EXAMPLE: Press [ALPHA] FFFFFFF [ALPHA], then 16 [STO] [X◄►Y], and [XEQ] [ALPHA] XD [ALPHA]. You'll see 268,435,455.0 in the display. For the reverse operation, and with the number remaining in the X-register, and the same base, press [XEQ] [ALPHA] DX [ALPHA] and see FFFFFFF in the display.

(On that other matter in your letter, Mr. Keefe, no, "HsiloP" did not win the contest. And I cannot tell you — yet — which "name" did win. See the "Editorial" — Ed.)

Houston, Texas, has grown in leaps and bounds in the past 20 years, but **Bill Rudersdorf's** HP-41 System expanded considerably in only one day — the day he added an X-functions module. Here's a neat routine he sent to KEY NOTES.

(41) Here's a routine that is useful in decoding nonstandard ALPHA strings; that is, nondisplayable characters, including characters 128 to 255, and nulls. It works equally well with or without a printer, and it does not use any storage registers. To use the routine, have the string in question in the ALPHA register, then XEQ "XA." The character codes (bytes) will be displayed (printed) in turn. Use R/S if you do not have a printer.

```
01♦LBL "AX"
02 FIX 0
03 SF 21
04 CF 29
05 FS? 55
06 AVIEW
07♦LBL 00
08 ALENG
09 X=0?
10 GTO 03
11♦LBL 01
12 ATOX
13 VIEW X
14 DSE Y
15 GTO 01
16 GTO 03
17♦LBL 01
18 X<>Y
19 ALENG
20 -
21 X=0?
22 GTO 00
23 0
24♦LBL 02
25 VIEW X
26 DSE Y
27 GTO 02
28 GTO 00
29♦LBL 03
30 TONE 9
31 END
```

We received several letters about a routine in V6N4, but this was the first one, so we'll use it to correct a small flaw. The letter is from **Roger Bailey**, who lives in Pawcatuck, Connecticut.

(41) Another informative issue has just arrived, and I would like to thank you and your staff, as well as the family of HP enthusiasts who send in their ideas. Many of those ideas have found their way into my programs. I found immediate use for **Hans Aspenburg's** sort routine (V6N4p14c) but a minor correction is required in the explanation. Since the routine uses ISG rather than DSE, the block of registers sorted is one more than .eee. The simplest solution is to key-in bbb.eee" – 1" or for the "purists," to adjust the preset of ISG immediately after line 01 with (.001), (–).

(Thank you, Mr. Bailey, both for the correction and the compliments. I saw that error when the newsletter came off the press, but one day too late to correct it. Thanks also to all the others who wrote — Ed.)

There is at least one X-functions module in Athens, Greece, and it attends college in the HP-41 carried by **John P. Ioannidis**, who contributed both of the following ideas.

(41) Here is a tip for the X-functions module. A function "missing" from the module is the inverse of SAVEX, the function that saves the X-register in the working file and increments the pointer by one. The inverse, POPX, should decrement the pointer by one (to gain access to the last saved X) and recall that number without incrementing the pointer. This way, a LIFO (Last-In-First-Out) stack can be built. My version of POPX does exactly that; the draw-back is that it uses register 00, so that the RPN stack will remain unaffected. If POPX terminates in error (that is, if there is nothing left in the data file to recall), the message "STACK EMPTY" is displayed and the RPN stack is left unchanged. Otherwise, the latter is lifted by one register, and now you see in the X-register the recalled number. A second routine, RESET, zeroes the pointer, so that a new stack can be started.

```
01♦LBL "POPX"
02 R↑
03 STO 00
04 RDN
05 RCLPT
06 DSE X
07 STO X
08 X<0?
09 GTO 01
10 SEEKPT
11 STO 00
12 CLX
13 GETX
14 X<> 00
15 SEEKPT
16 X<> 00
17 RTN
18♦LBL 01
19 "STACK EMPT
20 AVIEW
21 X<> 00
22 RDN
23 RTN
24♦LBL "RESET"
25 0
26 SEEKPT
27 END
```

This next tip is what I think is the shortest way to convert a hex digit in the ALPHA register (0 thru 9, A thru F) to its numeric value in the X-register and vice versa.

From hex to decimal:	From decimal to hex:
ATOX	9
64	-
-	X=0?
X=0?	64
9	X=0?
X=0?	57
16	+
+	XTOA

These two routines can be used in a larger hex-to-decimal converter etc. As it can be clearly seen, the solution to the problem that character 9 is not contiguous to character A in the ASCII code is to subtract a number in-between (64) so that a test can be made (less or greater than zero), and then add the right offset (9 or 16). Apply the reverse procedure to change a decimal into a hex digit.

Now, here is a "security" routine that can be used to provide some measure of protection for your sensitive files. It was contributed by **Carsten Bockermann**, who lives in Essen-Stadtwald, W. Germany.

(41) This routine converts a string in the ALPHA register to a series of so-called "starburst characters" with an ASCII character code greater than 126, so that the original message becomes completely illegible. If the routine is used again (now with the "starburst" string in the ALPHA register), the original string will be restored. I use this coding program in a large telephone directory program to protect the names of my clients from unauthorized access. You can use any number greater than 127 in line 02, however, smaller numbers will not always produce "starbursts."

```
01♦LBL "CODE"
02 236
03 STO 00
04 ALENG
05 0
06♦LBL 00
07 RDN
08 ATOX
09 RCL 00
10 -
11 ABS
12 XTOA
13 DSE Y
14 GTO 00
15 END
```

There are many different "formatting" possibilities on the HP-41 System. Here is one from **A.M. Platt**, who calculates in Richland, Washington.

(41) Here is a 48-byte subroutine for formatting ALPHA and data output; for example, COST = 100.00. It compensates for negative data and returns X, Y, and Z of the stack unaltered. The weird logic in lines 08 through 15 is there to preserve Z.

```
01♦LBL "FORM"
02 STO 00
03 X<0?
04 SF 00
05 "├ ="
06 ACA
07 ARCL X
08 CLX
09 ALENG
10 CHS
11 FC? 00
12 23
13 FS?C 00
14 24
15 +
16 X<0?
17 0
18 SKPCHR
19 CLX
20 RCL 00
21 ACX
22 PRBUF
23 RTN
24 END
```

The next input about the X-functions module is the result of an idea from V6N3. It is from **Sean M. Hill**, who is a student living in Lively, Ontario, Canada.

(41) Mr. Lafolet's routine for accessing registers greater than 99 (V6N3p12a) really caught my eye. I began wondering how one could access registers in an extended memory file more easily. After a great deal of thinking (and key-clicking), I developed this routine. "STEX" stores a number into a record defined by the user. "RCEX" does the opposite. The routine assumes that the user is positioned to a working data file before it is used. "STEX" leaves the stack alone but LASTX is replaced by the pointer value. "RCEX" lifts the stack in the same manner as RCL. LASTX is not affected in this case.

```
01♦LBL "STEX"
02 X<> L
03 RDN
04 "STEX   "
05 XEQ 00
06 SEEKPT
07 X<> L
08 SAVEX
09 RTN
10♦LBL "RCEX"
11 "RCEX   "
12 XEQ 00
13 SEEKPT
14 CLX
15 GETX
16 RTN
17♦LBL 00
18 CF 22
19 AVIEW
20♦LBL 01
21 PSE
22 FC? 22
23 GTO 01
24 END
```

Examples: Store the number 100 in register 12 of a data file. The keystrokes are: 100, [XEQ] [ALPHA] "STEX" [ALPHA], 12. Then, to recall a number from record 03, the keystrokes are: [XEQ] [ALPHA] "RCEX" [ALPHA], 3. (Note: "STEX" and "RCEX" could be assigned to the STO and RCL keys for convenience.)

BASIC HP-41 ROUTINES

Because a lot of you do not own the X-functions module, here are some ideas and routines that you can use for your more basic HP-41 System.

The guitar is now an extremely popular musical instrument all over the world, and we know of one in Medina, West Australia, that is tuned by **D. T. Rigden's** HP-41CV (and good ears, we'll bet!)

(41) I had just written a metronome routine for my HP-41CV (which I play with when I'm not playing my guitar), when KEY NOTES V6N4 arrived. My routine differs from **Paulo de Salles Mourao's** version (V6N4p12c) in that it includes a tempo factor and will not operate *vivace*. Also, the tempo alters slightly with battery condition. I don't yet have a time module.

```
01♦LBL "MM"
02♦LBL A
03 "TPO?"
04 PROMPT
05 STO 02
06 "MM?"
07 PROMPT
08 STO 01
09 TONE 7
10♦LBL 01
11 DSE X
12 GTO 01
13 DSE Y
14 GTO 02
15 RCL 02
16 RCL 01
17 TONE 7
18 GTO 01
19♦LBL 02
20 RCL 01
21 +
22 TONE 1
23 GTO 01
24 END
```

Incidentally, tuning a guitar in any key requires a compromise between equal temperament and just intonation. I have found my HP-41CV very useful in simplifying the tuning.

We received a lot of mail about the next subject, but the following letter by **Ken Trumble** of Calgary, Alberta, Canada, best expresses the interest generated by one routine.

(41) I am writing in regard to the routine "SHOP," by **C. Lamar Williams** and published in V6N4p12c. The routine is admirably short and simple, but it does not cover every factor considered when shopping, especially grocery shopping. In deciding between brands, the choice is based on personal preference as well as price. Knowing that brand A is cheaper is fine and good, but it is also necessary to know, in some meaningful way, *how much* cheaper. Therefore, I have modified "SHOP" to give the percent change in unit cost from the cheaper brand to the more expensive.

SHOP 2 works in the same manner as SHOP, but after receiving the answer "A IS CHEAPER," you press R/S. For Mr. Williams' example, the response displayed is "BY 0.87%." If there is a preference for brand B because of taste, or the larger quantity provided, a price advantage to brand A of less than 1% may be irrelevant. SHOP 2 provides the necessary information to make this decision.

```
01♦LBL "SHOP2"      14 %CH
02 FIX 2            15 "BY"
03 "A"              16 ARCL X
04 XEQ 00           17 "├ %"
05 XEQ 00           18 AVIEW
06 X>Y?             19 FIX 4
07 "A"              20 STOP
08 "├IS CHEAPER"    21♦LBL 00
09 X=Y?             22 "├=?$↑QUAN"
10 "A=B"            23 PROMPT
11 PROMPT           24 /
12 X<Y?             25 "B"
13 X<>Y             26 END
```

Routine SHOP 2 is a highly useful modification of SHOP, and I believe it is worth publishing. In the meantime, I will continue to follow KEY NOTES for HP-41 product and programming information, and for serendipity.

(Mr. Williams' SHOP routine generated a lot of interest in this type of programming. Evidently a large number of readers are using their HP-41's to guarantee shopping economy. I thought Mr. Trumble's version worth printing because of the wide interest in this subject — Ed.)

Okay, how about a "battery" routine for a change of pace? Here's a clever contribution by **Robert Peraino**, who lives about 2,700 miles from Corvallis, in Fairfax, Virginia.

(41) Many times I have wanted to run long programs on my 41C and have worried about the rechargeable battery power running out while the program was running unattended. Utilizing this global subroutine, if the battery power runs low while running a program, the 41C will shut itself off. When you turn the 41C back on, it will tell you why it shut off and let you continue the program. (Knowing why is important, since some programs are set up to shut down when execution is terminated, and this can be confusing). Or, if you wish, you can shut off the 41C, recharge it, and still continue your program. The routine disturbs only the ALPHA register. To set up a program to use the routine, simply pick a spot in your program that is executed frequently and insert FS? 49 and XEQ "POWER."

```
01♦LBL "POWER"      11 "POWER FAIL"
02 SF 11            12 AVIEW
03 "LOW POWER"      13 PSE
04 AVIEW            14 "R/S CONTINUE"
05 PSE              15 SF 11
06 "SYSTEM DOWN"    16 PROMPT
07 AVIEW            17 FC?C 11
08 PSE              18 GTO 01
09 OFF              19 RTN
10♦LBL 01           20 END
```

From Tucson, Arizona, the hometown of **Robert J. Greenberg's** HP-41 System, comes this neat routine that builds on a routine from V6N4.

(41) In reference to John Erik Setsaas' routine "D-C" in "Feedback" (V6N4p10c), it is nice to see someone else using BLDSPEC to create characters from decimal numbers. Mr. Setsaas' routine will only create characters 1-127. Characters 128-255 are useful for controlling some peripheral devices as well as for inverse characters on the 82163A video interface display. The following routine will create characters 1-255 in the X-register. In addition, it will convert numbers 256-65535 to a corresponding two-character string that can include null characters (character 0). It consumes 11 more bytes than Mr. Setsaas' routine.

```
01♦LBL "D-C"        07 R↑
02 STO Z            08 BLDSPEC
03 128              09 X<>Y
04 ST/ T            10 BLDSPEC
05 MOD              11 END
06 ENTER↑
```

It is used identically; place a number in the X-register and execute D-C. The previous contents of the T, Z, and X-registers will be lost.

What does one do on cold winter evenings in Brookline, Maine? For **Alex Gold-Pitegoff**, the answer is easy. He organizes his HP-41 programs so he *knows* what he has when he wants it.

(41) I have had my HP-41C for about a year, and early on in my programming of it I adopted a trick that is standard in programming in larger machines.

The problem this trick addresses is, how do you know that the various copies of a given program are accurate representations of each other? In other words, suppose you return to the development of a program by pulling its listing out of your file and loading the program thru your card reader. How can you tell that the listing is an accurate representation of the program in memory?

The second line in every program I write is of the form, ALPHA nVm ALPHA, where n is the letter or number for minor revisions (for example, bug corrections) and m is the letter or number for major revisions (for example, feature additions). In other words, this ALPHA string is a revision number, and whenever I edit a program and produce a new listing and program card, I update the number.

Tricks are possible. For example, if you make the string a label, it will show up in the Catalog and will not affect the ALPHA register. Alternatively, you can make the string nVm and place it after the ALPHA string program name for an initial prompt, as follows:

```
"PROGRAM NAME"
"nVm"
AVIEW
```

In any event, the trick will keep you from fixing bugs in an obsolete version of a program.

Why would a "crop duster" need an HP-41 System? Why would I print such a routine? The routine answers the first question and the editorial note answers the second one. The input is from **George J. Parker**, who "lands" in Newburgh, New York.

(41) Here is a small routine for the "back and forth" people. (He's referring to those daredevil pilots known as "crop dusters" — Ed.)

```
01♦LBL "ACRES"      10 RCL 01
02 FIX 1            11 *
03 "SWATH?"         12 1 E3
04 PROMPT           13 /
05 2                14 CLA
06 *                15 ARCL X
07 STO 01           16 "├ ACRES/MIN"
08 "SPEED?"         17 AVIEW
09 PROMPT           18 END
```

(It's easy to use and easy to modify to make it fit your application. For example, you could use it for a range of agricultural applications — no pun intended. Or you could alter it for a sawmill application. And to use it, all you do is XEQ "ACRES" and input the feet — for width — when SWATH? appears. Then press R/S and see the prompt SPEED? Enter the speed and press R/S, and you will get the correct answer in acres per minute. If you "shorten" line 16 to ACR/MIN you will have a more easily read display when large numbers are used in the routine, By the way, Mr. Parker, thanks for the correction to the manual for the HP-41 — Ed.)

Here is a tip for those who own the HP-IL printer for the HP-41. This came to us by way of **Don R. Leavitt**, who lives in Sandy, Utah.

(41) My suggestion concerns a convenient way of clearing the print buffer in the HP 82162A printer without printing the contents. In some of my programming, I have often wished there were such a "CLEAR BUFFER" function, but the only suggestion I had seen for doing this was to turn off the printer and then turn it on again. I recently discovered, however, that the buffer can be cleared without printing by executing the PWRDN function from the HP-IL module. The PWRDN function can be included in a program for automatic execution, or it can be assigned to a key on the HP-41 for convenient manual use. When the printer is in the ON mode, rather than STANDBY, you also don't have to worry about executing PWRUP.

(This tip does work, but be careful that you don't lose everything on the loop. The new Extended I/O Module discussed elsewhere in this issue has a CLRDEV function that now takes care of the above problem — Ed.)

HP-67/97 ROUTINES

Although the above title is plural, we have only one routine to offer for this issue. So if you have HP-67/97 contributions, send them in so we can evaluate them for KEY NOTES.

This contribution is from **Thomas C. Carrington**, who lives in Plano, Texas, which is a nice city just north of Ft. Worth.

(67/97) Here is a simple routine for the HP-67/97 that will determine if a number is prime. With the proper functions, it can be easily adapted to the HP-41 or any other programmable calculator. The routine will return a zero for non-prime numbers and a one for prime numbers. The number must be an integer, and greater than zero, and less than 100,000 for the routine to yield proper results.

Because of the method of factor-hunting, this routine has some inefficiencies. Every factor less than or equal to the square root of the number is exhaustively tried, starting at the integer of the square root and proceeding to zero. It would be much faster, of course, to try only the prime numbers as factors.

The routine may be shortened by the elimination of lines 06, 07, and 08 if it is not necessary to return 1 2, and 3 as prime numbers.

```
001  *LBLA
002   STOA     Save number and compute first guess bas-
003    IX        ed upon integer of the square root.
004   INT
005   STOB     Save this as the first guess.
006     1      Compare the first guess to one.
007   X=Y?     If it is equal to one, then the number must
008   RTN        be prime, return with a one.
009  *LBL1     Here, try the guess as a factor.
010   RCLA     Divide the number by the guess.
011   RCLB
012     ÷
013   FRC      If the fractional part is zero, then the num-
014   X=0?       ber is not prime, return with zero.
015   RTN
016     1      Load a one for comparison later.
017   RCLB     Subtract one from the guess to give the
018     1        next guess.
019     -
020   STOB
021   X>Y?     If the next guess is greater than one, then
022   GTO1       try that guess, else the number is prime,
023   RTN        return with a one.
024   R/S
```

HP-75 ROUTINES

Although this is a shorter-than-usual edition of KEY NOTES, here is at least one tidbit for the HP-75C fans. We'll have lots more for you in August.

This contribution was just a bit late for the last issue, so I saved it for a "rainy day" (something NOT rare in Oregon!). It is from **Howard Punches**, who lives in Corvallis, Oregon.

(75) For those times when I want to show off my HP-75C to friends who don't understand programs or programming, I have taught it to play music. Bach's fugues or two-part inventions are easy to program, and the HP-75C plays them beautifully. Two general formats can be used.

For tunes where timing of musical notes varies or there are rests to be allowed for, then a program may be written that is just a sequence of BEEP and WAIT commands. For example:

```
10 BEEP 700, .1 @ WAIT .1 @ BEEP 884, .1
   @ WAIT .1 @ BEEP 700, .1 @ WAIT .1
20 BEEP 1044, .1 @ WAIT .1 @ BEEP 700, .1
   @ WAIT .1 @ BEEP 1400, .2 @ BEEP 1316,
   .1
30 BEEP 1156, .1 @ BEEP 1044, .1 @ BEEP
   1168, .1 @ BEEP 1044, .1 @ BEEP 934, .1
   @ BEEP 884, .1
40 BEEP 934, .1 @ BEEP 884, .1 @ BEEP
   784, .1 @ BEEP 700, .1 @ WAIT 1.0
```

starts the F major two-part invention.

When all the notes have the same length, with no rests (such as C.P.E. Bach's "Solfegietto"), programming is simplified with READ, DATA statements:

```
10 ON ERROR GO TO 70
20 READ A
30 BEEP A, 0.1
40 GO TO 20
50 DATA 308, 261, 308, 392, 522, 628, 586,
   522, 492, 392, 492, 586, 784, 700, 628,
   586, 628
60 DATA 1256, 1172, 1044, 1172, 1044,
   984, 884, 784, 700, 628, 586, 628
70 END
```