By: Harwood G. Kolsky Date: July 10, 1994

Father of the Personal Computer

The now popular personal computer became part of the vocabulary of the American public in 1981 when IBM announced its famous "PC", advertised by a Charlie Chaplin-lookalike. In the thirteen years that followed PCs from many different manufacturers have become so common that it is hard to imagine how business was conducted without them. For example, this Noviny was written, composed and printed by several members using their home PCs.

About five years ago a local Silicon Valley newspaper ran an article speculating on who was the "father of the PC." The reporter had done interviews and listed several well-known personalities in the computer business who might be considered for the honor. I was teaching a class that included computer history at the University of California Santa Cruz at the time. I was dismayed that so few people were familiar with the foundations of their own field. I wrote a letter to the editor trying to set the record straight, because I had been a witness to the true story.

Dr. Paul J. Friedl, an engineer of Czech descent, working with two colleagues at the IBM Palo Alto Scientific Center in 1973, has the honor of designing and building the first portable, desk-top computer.

In February 1994 the Silicon Valley Engineering Council at their annual ceremonies inducted Paul into the Silicon Valley Engineering Hall of Fame. Their announcement of the honor stated:

"Paul J. Friedl is known to many people as the 'Father of the Personal Computer.' He was the chief architect and inventor of the world's first personal computer and also developed the predecessor of the modern spreadsheet program in 1973, long before personal computers, as we know them today, were introduced. He christened his computer "SCAMP" (Special Computer APL Machine Portable), and it became the father of the IBM 5100 and the grandfather of the ubiquitous IBM PC, which was introduced in August 1981, nearly eight years later. The original SCAMP is now in the Smithsonian Institute.

"Paul's 32-year career with IBM as a senior engineer and manager included pioneering work in industrial process control, laboratory automation, knowledge-based expert systems, distributed computing, and computer conferencing systems. He also authored many technical papers and patent disclosures. He invented the IBM People Sharing Information Network (PSInet) Computer Conferencing System, which is being used by kindergarten through 12th-grade educators throughout the country.

"He is a senior member of the Institute of Electrical and Electronics Engineers and is a Registered Professional Control Systems Engineer in California. He received his bachelor's, master's, and doctoral degrees in Chemical Engineering from Case Institute of Technology, where he was a Westinghouse Fellow for two years.

"Dr. Friedl represents the perfect combination of a practical engineer with numerous scientific and practical accomplishments to his credit and the visionary who works to transform the future into today's reality."

Paul's grandfather came to America in the 1880's from the city of Pisek, settling in Cleveland, Ohio, where Paul's father was born.

Although Paul's knowledge of the language is limited to "jak se mas" and one or two slang expressions, he is proud of his Czech background and sends his greetings to all the readers of Noviny.

1987 H.G. Kolsky

National Computer Conferencing Network
for Science Education

Ψ-NET USER'S GUIDE

1.05

Council of State Science Supervisors

IBM Corporation

Also supported by the National Science Foundation

Ψ-NET (or PSI-NET) is the name of an experimental computer conferencing program being developed by the IBM Scientific Center in Palo Alto, California.

April 20, 1987

Ψ or psi (pronounced s-eye; or ps-eye with the ps as in hops; or ps-ee) The twenty-third letter of the Greek alphabet.

The first pronounciation (s-eye) with network seems especially attractive for the National Computer Conferencing Network for Science Education because it implies a SCIence network. For other applications, one may prefer to make an appropriate acronym out of PSI, and we offer the following one, "People Sharing Information", because that is what this program is all about.

Paul J. Friedl

Table of Contents

INTRODUCTION

What Ψ-NET Is

Ψ-NET, also called SPS (Science Project Software) in this manual, is an experimental IBM computer conferencing system which supports the telecommunications needs of a widely scattered group of users operating IBM Personal Computers. The program allows users to send one another electronic mail and participate in on-going computer conferences using standard telephone service to allow each user to call a remote Personal Computer called the Server.

What You Can Do With Ψ-NET

Ψ-NET allows you to communicate electronically over telephone lines with other people in your building, on your campus, or located in different states or countries. One major advantage of this electronic communication is that it is accomplished at your own rate, and at whatever time of day is convenient for you; and it is not necessary for the talker and listener to be on-line at the same time. Using Ψ-NET you can send text, program, data, and image files from one IBM Personal Computer to another.

In this manual you will learn how Ψ-NET simulates the same style of communication as that of live professional conferences where material is discussed and presented in organized sessions. You can join existing conference sessions of interest and receive items (papers) from them, or send new items to them. You can also add your own new session for an existing conference, or even start your own new conference whenever needs arise. This conferencing capability has been shown to be an extremely powerful way for distributed groups of people to plan new undertakings, solve problems, and develop new policies without the need to hold time-consuming and expensive group meetings.

Ψ-NET not only does the electronic communication for you automatically, but also has a set of powerful facilities to help you browse, find, and sort the various files which you have stored on your IBM Personal Computer. These facilities act like an electronic librarian who is always standing-by to carry out your commands. Thus, you can easily retrieve all of your electronic messages sent to or from another party, or review all or recent items from an on-going Ψ-NET conference session which you have joined.

Ψ-NET allows you to share data and programs among IBM Personal Computers. This makes it possible to use the program to support not only your inter-personal communications, but also many distributed data processing applications which require the transmission and use of shared information. Thus, for example, if a person in one city who is creating a report for his division needs to have some data which is available from a person in another city -- they can use Ψ-NET to transfer a copy of the needed data from one city to another.

GETTING STARTED

What You Need

There are a few items that you need before you can start using Ψ-NET. They are:

- Ψ-NET Workstation Diskettes

  Two Ψ-NET Diskettes accompany this manual. These diskettes contain the programs that you need to run Ψ-NET. You cannot run Ψ-NET with these diskettes alone; you must also have DOS running in your computer.

- DOS Version 2.1 or later

  Ψ-NET runs with DOS Version 2.1 or later. To get Ψ-NET ready to run on your Personal Computer, you need to combine your DOS files and the Ψ-NET files on the same fixed disk. How to do this will be explained later.

- A Text Editor or Word Processor program

  You need to install the Electric Desk program on your Personal Computer's hard file in a directory called ELECTRIC. Instructions for doing this are given in the Electric Desk User's Guide. The Ψ-NET will not work properly if you do not also have a text editor or word processor program installed.

- A minimum of 512 K bytes of storage within your Personal Computer

- A Fixed Disk on your Personal Computer

- Internal or External Programmable Modem

  You need to have a modem which supports the AT command set installed with your Personal Computer. The modem is used to link your Personal Computer to a telephone line. Examples of modems which you can use with Ψ-NET are the IBM 5841 external modem, and the IBM Personal Computer Internal Modem 1200.

- Registration on one or more Ψ-NET Servers

  Before you can communicate with other Ψ-NET users, you must register with the operator of at least one Ψ-NET Server. With Ψ-NET you send your messages and files to the Server which then forwards them to their destinations; this means that there is no direct communication between one Ψ-NET user and another, only indirect communication through the Server.

  Getting a System Name and Password

  The first task you have is to contact each person who operates one of the Ψ-NET Servers with which you intend to communicate.

  Note: CSSS National Science Education Network users should call **Ken Hartman** in Iowa.

This person we will call the Ψ-NET Sysop. You should contact each Sysop on the (voice) phone and tell him/her your choice for a user name and password. The Sysop will check to see that the user name is not already used (if it is you will need to choose another). The Sysop will then register you as a member on the server with the chosen name and password.

Note: The user name and password are both single words of 11 characters or less.

Note: If you will be using more than one Server, you must have the same name and password on each Server because the name and password entered when you start your Workstation is used by the program when it contacts all of your Servers.

In addition, you will need to give the Sysop the phone number to which your Workstation is attached and the time periods during the day you wish to allow the conferencing system to call you if the Server has some items to transmit to you. This "Mail-Waiting" call feature is an optional service which may be supplied by a Server, and it will be discussed later.

If you have everything that's been discussed up to now, you can continue.

IMPORTANT: It is absolutely necessary for you to perform ALL of the steps which are described in the following section BEFORE you attempt to use this program!

Setting Up Your Ψ-NET Workstation Program

1. If you have not already done so, install DOS and Electric Desk on your hard disk. Make sure your a: drive is empty, and then press **Alt Ctrl** and **Del** keys simultaneously to reboot your system; you should get a prompt that looks like C>.

2. Put the Ψ-NET Workstation (SPS) diskette 1 into your a: drive.

3. Enter **a:** (type *"a:"* and then press the Enter key)

4. Enter **install**

   You should now follow the instructions given on the screen by the installation program. In the beginning you will be presented with a form asking which directory you wish to install SPS in. The default directory is *C:\SPS*. To accept this directory name just press the **Enter** key. To choose a different directory enter its name in the space provided. SPS will then be installed in the chosen directory.

   You will be presented with a form asking which communications port your modem is installed in and which editor you would like to use to write documents within SPS. For use with nearly all of the Personal Computers operating with the CSSS National Science Education Network, you should enter **1** for the COM Port (assuming you have only one serial I/O port and it will be used for your modem), and **EDS** for the Editor .

   Note: When creating papers, messages, or abstracts, SPS will attempt to call ELECTRIC DESK, or any other editor you may have specified in the above **install** operation, and pass it the filename, **$$ACT$$.TEX** . If you have chosen to use an editor other than Electric Desk and it does not allow this, you must ensure that you file (export) your papers, messages, and abstracts as plain ASCII files using the name, **$$ACT$$.TEX**. Since **$$ACT$$.TEX** is an empty file when SPS calls your editor, your editor may give you a warning message.

   Note: The above **install** operation may modify your CONFIG.SYS and AUTOEXEC.BAT files if you already use them on your personal computer. Thus, if you have a CONFIG.SYS, and it contains values less than FILES = 15 and BUFFERS = 16, **install** will modify it to have these values. If you have an AUTOEXEC.BAT, **install** will add *C:\ELECTRIC* to PATH and APPEND. If you have installed your copy of Electric Desk in a directory other than *ELECTRIC*, then you must use an editor to change your AUTOEXEC.BAT file to contain the

correct PATH and APPEND information to enable SPS to find your Electric Desk program. **IMPORTANT: If you modify either your CONFIG.SYS or AUTOEXEC.BAT files, you must reboot your PC in order for the changes to take effect. You reboot your PC by pressing** *Alt Ctrl and Del* **keys simultaneously.**

When the installation is complete, you should get the message *"****** Installation of SPS is complete ******"* and you will return to the DOS prompt.

5. Start the SPS program.

   The SPS program can be started in two ways. If you have a color graphics display, you can start the SPS program with the command *action*. This will display a 10 second graphics sign-on and then run the SPS program. If you do not have a graphics display or you wish to skip the graphics sign-on, you can start SPS with the command *SPS*.

   Note: The SPS program is designed to start up using the *SPS* command from any directory. However, if the *SPS* command gives the error message *Bad command or filename*, you should either make sure that your DOS path includes the root (\) directory (refer to the DOS manual for setting the path), or type the DOS command *cd \sps* before the *SPS* command.

   When the program starts up it will display a sign-on screen and then prompt you to enter your Ψ-NET user name and password. These are entered in a standard data entry form, but the password field will not display the characters entered. You can press **Esc** to exit to DOS instead of answering the prompts. When both prompts are answered, the program starts.

   Try starting your Ψ-NET Workstation program now. If you are successful, the program will display a menu bar across the top of your screen after you enter your user name and password on the sign-on screen. If you were not able to start the program you should contact your Sysop. Assuming you had success and have the menu displayed on your screen, you are ready to do some simple but necessary tasks in order to begin using Ψ-NET. We will discuss more details of how to use the Ψ-NET menu system later, but for now, you **must do the following few things**.

6. Identifying Server(s)

   Your first task is to tell your Ψ-NET Workstation about the Servers you wish it to know about. From the menu type *"es"* (***Edit-Server list***). The first time you do this you will be presented with a Server List with one entry, MANUAL. Now press **Ins**, and key in the name of a Server, followed by **Enter**. Now key in the Server's phone number.

   Note: The first character to be specified must be a **P** if pulse dialing (i.e. not tone dialing) is to be used to call the Server over the telephone line.

   Note: Next, the number must contain all of the digits needed for your Workstation to place a telephone call to the Server including special access codes, tieline codes or area codes (just as you would have to do if you placed a voice telephone call to the Server's number), followed by **Enter**.

   Note: In addition, you may have to insert one or more comma's in the string of digits in order to have the modem delay at certain points while dialing. Although programmable modems may be set up to delay up to 255 seconds for each comma, the default delay for each comma is normally set to 2 seconds. For example, if you wished to call a server from your office at work, you may issue a string such as: 9,415-123-4567 Here, the 9 is a code commonly used to access the external telephone system; the comma is inserted in order to delay 2 seconds so that the modem receives a dial tone before continuing with the digits of the telephone number; and then the area code and 7 digit number (the hyphens have no effect on the dial, but can be used to make the telephone number easier for you to read).

Note: For CSSS National Science Education Network users, use the word **CSSS** for the name of the server. Call Ken Hartman in Iowa to find out what phone number to use for the CSSS Server.

When you get the message, *"Confirm new server <Y> or <N>"*, press **Y**, and **Enter**.

If you wish to identify any other Servers to your Workstation program, press **Ins** and repeat the above process for each new server. You may also need to repeat the above steps whenever you wish to run your Workstation program from a different location (such as your home) since the telephone in your home probably does not require special tie line codes, etc. In this case you should find the server whose phone number you need to change in the list and select it with **Enter**. You will then be given a chance to change its phone number.

In the event you wish to delete a Server from the list, move the cursor to the line containing the Server you wish to delete, and press **Del**.

When finished, press **Esc** which will return you to the main menu.

7. Tell the Server About Yourself

From the main menu enter *"msa"* (Mail-Send-Abstract). The program responds with a data entry form showing two fields. Press **Enter** and the program will activate your own editor which you should use to prepare your User Abstract (a short paper describing yourself and giving any information which you want other users on your server to know about you).

Note: If you are using Electric Desk to prepare your abstract, refer to the this.

After you enter the first field of the form, the program moves the cursor to the second field, and also displays an information window containing the list of the Servers you identified in the previous section. You will learn later how to select a Server name from the information window and transfer it to the form, but for now just enter one of the Server names into the second field of the data entry form using your keyboard, and then press **Enter**.

You may now repeat this *"msa"* procedure for any additional Servers you wish to use.

When finished with the above *"msa"* procedure(s), enter *"md"* (Mail-Do). If you have performed an *"msa"* procedure for each one of the Servers in your list, your Workstation should now dial each Server in your list and receive information from each of them. If something does not work, the program will attempt to inform you by displaying an error message of some sort on your screen. Sometimes, however, especially when you are initially setting up your Ψ-NET system, there may be problems caused by improper configuration of your modem or serial adapter on your Personal Computer, or by conflicting interrupt or port assignments. Remember that no two Personal Computer adapters may simultaneously use the same port (e.g. COM1 or COM2) or interrupt level. If you think that you may have some of these configuration conflicts between your adapters, you should turn power off on your Personal Computer, and then remove all device adapter cards except for the ones which drive your display, keyboard, disk drives, and the one you are using to drive your modem. Then turn power on, bring up DOS, and redo steps 5, 6, and 7. If you still have trouble here you should contact your Sysop for assistance.

If your Workstation successfully contacted your Server(s), and you do not see any error messages, then you have successfully informed the Server(s) about who you are, and in return your Workstation has received the control information which it needs in order to allow you to use the conferencing system. You are now ready to use your Ψ-NET Workstation.

Re-configuring your SPS workstation

If you ever need to change your communications port number, turn the modem speaker on or off, or change the editor used by SPS after you have installed it, you can do so by getting into the SPS directory (e.g. with the DOS command *cd \sps*) and then entering *wconfig* to the DOS prompt. You will be shown the existing values and prompted for changes.

IMPORTANT: Do not go beyond this point unless you have successfully completed all of the steps up to this point!

Learning to use Ψ-NET

The Menu

The Workstation program takes your input from the keyboard and displays output on the screen. The general philosophy for this interface is a windowing system with pull-down/pop-up menus. Selecting actions from the menu system can put windows on the screen which contain the information for that action. Windows will overlap each other as different actions are stacked upon each other. In general, pressing **Esc** will cancel the current window and take it off the screen. The information that was under the canceled window will then be restored and you will be put back in the previously used window. At any time during the program, you can select from the main menu bar. If the new action requires a window, the new window will be placed over any existing windows and become the top and current window.

The operation of the keyboard and the layout of the screen are described below.

The Message Line

The last line of the screen is used as a message line. This line always displays (in green for color displays), a one-line explanation of what you can or should do at that particular point in the program, or what the computer is doing if it is busy.

The Menu System

The menu system of Ψ-NET is based on a single global menu structure of pull-down and pop-up menus. The menu system also looks and operates similarly to the menus made popular by various spreadsheet programs. The major difference, though, is that the menu system is not a tree-structured set of "modes". All menu choices are made from the same global menu system. That is, making a menu choice does not put you in a different mode where the menu looks different and the choices you saw before are no longer available.

The Ψ-NET menu system consists of one main menu, several pull-down menus, and additional pop-up menus. The main menu is a horizontal list of choices (words) which fits on one line. The main menu is permanently on the second line of the screen throughout the program. This is called the menu bar. Each choice in the main menu activates a corresponding pull-down sub-menu. These are vertical menus which list one choice per line and are located below the corresponding choice from the main menu. These menus temporarily write over what was under them on the screen, and restore the screen when they are canceled. If a choice in a pull-down menu needs to activate yet another sub-menu, a pop-up menu appears just below and to the right of the choice selected. These pop-up menus can call additional pop-up menus as necessary.

> ***Let's EXPERIMENT*** - Enter *"b"* (Browse), and observe that a pull-down menu appears under the ***Browse*** box on the main menu bar. Once you have se-

lected ***Browse*** or any of the other main menu boxes (or commands), the program will always display the pull-down menu corresponding to that command. When you have any one of these pull-down menus displayed on your screen you are able to use the **left** or **right** cursor key to pull-down an adjacent menu. Press the **right** cursor key now, and notice that the ***Browse*** pull-down menu disappears, and the ***Find*** pull-down menu is displayed instead. In this way you can move the cursor left and right to get a quick view of the contents of each pull-down menu, and this is a handy way to refresh your knowledge of what you can do using the various commands available on the main menu bar. If you want to impress a friend, hold the **left** or **right** cursor key down for a second or so, and watch the screen as the pull-down menus flash by.

Making a choice from a menu can be done in two ways. When a menu is being displayed, an inverse-colored bar is placed over the currently selected (default) choice. The bar can be moved with the cursor keys (arrows). The **left** and **right** keys are used for the main menu, and the **up** and **down** keys are used for the pull-down and pop-up menus. The top line of the screeen always contains a one-line explanation of the currently selected menu choice. Pressing **Enter** selects the highlighted menu choice. This may invoke a sub-menu or an action. All of the choices in the main menu invoke sub-menus (the pull-down menus). The choices in the pull-down and pop-up menus that invoke additional sub-menus are marked for convenience with a "..." after the choice. For example, In the ***Mail*** pull-down menu, the choice ***"Receive..."*** will invoke a sub-menu. A menu choice can also be selected by pressing the first capital letter in the choice. This is almost always the first letter of the choice, but some conflicts make it necessary to use a later letter (eg. using X in the eXit command). A menu can be canceled by pressing **Esc**. This will remove the current menu and return to the previous menu.

Note that the menu system is tree-structured in the sense that it is a set of hierarchical menus. But what makes it different from most spreadsheet program menus for example is that choosing an action from a series of sub-menus always starts at "the root" (the main menu). You do not remain at a tree node after selecting an action (you do not get into any different modes). After an action is selected, the next menu choice starts from the main menu again. The selection of actions, then, is very similar to typing commands, where the command consists of the first letter of the menus necessary to get to the action. For example, *"fp"* will select the ***Find*** pull-down menu, and then the action ***Papers***, if you want to find papers in the database. It is not necessary to remember the commands, though, since the menus will lead you to the choice you want. But experienced users who have memorized the short commands can use the program very quickly.

Of course, not all choices in the menu system are relevant at all times. The choices that are not relevant at a particular time are "dimmed". That is, they are made a dim color. These choices cannot be selected with the bar (the bar will skip over them) and the pressing of the first letter of a dimmed choice is ignored. What choices are active (not dimmed) at a particular time will depend on what you are currently doing, and have already done. For example, the ***Sort*** menu is not active unless you are currently looking at a list which can be sorted.

When in the Ψ-NET program and doing something else besides selecting a menu choice, you can still get to the menu system. The menu system is always available at any time in the program if you press **Alt-*ch*** where ***ch*** is the letter for the choice from the main menu (the name of the pull-down menu to start with). For example, to find papers when not in the menu, the key sequence **Alt-*fp*** (That's **Alt-*f*** followed by *p*) will invoke the appropriate menus to invoke that action.

Additional controls for the menu system are

- When in a pull-down menu, the left and right arrows will switch to the next active pull-down menu to the left and right respectively. This is a fast way to examine all of the options available at a certain time (pull down a menu and use arrows to observe choices in all active pull-down menus).

- Entering menu commands using the **Alt** key and the letter of the choice from the main menu followed by the other command letters will work even if the menu system is active and you are in the process of making a choice. So, no matter where you are in the program, menu commands are *always* available by entering them using the **Alt** method.
- In the *main menu*, the **down** arrow will also select (pull-down) a menu choice (works the same as **Enter**).

List Selection Windows

At many places in the program, you are asked to select an entry from a list of things. For example, when given a list of conferences to browse, you select one. Selection from lists is done through list selection windows. These are windows which have one list item per line and an inverse colored bar over the currently selected item. To select a choice, you move the bar over the desired entry with the **up** and **down** arrows, and press **Enter** to select that entry. The **Esc** key will cancel the list if you do not want to make a choice. If there are more choices in the list than the number of lines available for the window, a "(more)" message will appear on the bottom of the window frame showing which arrow key(s) to press to scroll the list.

Additional controls for the list selection windows are:

- The **PgUp** and **PgDn** keys move up and down a full page (number of lines in the window) in the list.
- The **Home** key moves the bar to the first entry in the list.
- The **End** key moves the bar to the last entry in the list.
- The **Left** and **Right** keys will horizontally scroll the item lines up to a full window width at a time within the window if they are too long to fit within the width of the window.

Data Entry Forms

Several functions of the program require you to enter data. For example, when sending a paper, you must specify the conference, session, title, etc. You enter all data through special data entry forms. These forms are windows on the screen which show you all the fields you have to enter for the particular action, the default (current) values for each, and the maxium length available for the choice (as shown by a colored space to be filled in). One field in the form is the current field and has a different colored bar over it, and the blinking screen cursor is under the current position in the field. Some fields may cause the display of an associated list selection window which contains a list of all possible entries for that field. You may directly enter values into the fields of the form from your keyboard, or for convenience, you may elect to use the associated list selection window to select and transfer the appropriate value into the field as follows:

1. Press **F7** to go to the associated list selection window.
2. Make a choice as discussed above in the section, ***List Selection Windows***, pressing **Enter** to transfer your choice into the data entry form.

> ***Let's EXPERIMENT*** - If necessary, press **Esc** enough times so that you have only the main menu bar displayed on your screen. We will now demonstrate how to use a list selection window to aid you in filling out a data entry form. Press *"fp"* (Find-Papers) to pull-down the **Find** menu, and then select the **Papers** option. Your screen will display a data entry form with the cursor located in the first field (called **Server**), and a list selection window containing a list of one or more server names. You may move a name from the list selection window to the data entry form using **F7** as follows. Press **F7**, and observe that the cursor moves to the first line in the list selection window.

Use the **up** or **down** cursor key to move the cursor to one of the Server names in the list selection window, and press **Enter**. The selected name is transferred directly to the field in the data entry form, and you do not have to type it in using your keyboard. If the name is the one you wanted in that data entry field, press **Enter**, and the program takes you to the next field, and displays another list selection window with names appropriate for that field. You may use this same process as often as you like until the data entry form is filled in the way you wish.

You can move within the fields of the form to fill in or edit the values. When you enter the last field, the form is accepted with the fields as displayed. Certain fields are verified (checked for format correctness) as you enter them. For example, when you enter a date it is checked to be be sure it is a valid mm/dd/yy format. The controls for the data entry forms are as follows:

- The **Enter** key accepts the value of the current field and moves to the next field in the form. When **Enter** is pressed at the last field, the entire form is accepted.
- The **F10** key accepts the entire form as it is currently displayed (defaults are used for fields which have not been edited).
- The **Esc** key cancels the form and the associated program action.
- The **up** and **down** arrows move to the previous and next field in the form, respectively. Moving **down** at the last field accepts the form, and moving**up** at the first field is ignored.
- The **Home** and **End** keys move to the first and last fields in the form, respectively.
- The **left** and **right** arrows move the blinking cursor a character left and right within the current field.
- The **backspace** and **Del** keys delete the character to the left of and under the cursor in the current field, repectively.
- The **Ins** key toggles insert mode within a field. In insert mode, typed characters are inserted into the field and remaining characters are pushed to the right. Out of insert mode, typed characters overwrite the characters under the cursor.
- The **F3** key undoes any changes in the current field and restores the field to its initial (default) value.
- The **F5** key clears the current field to all blanks.
- The **F6** key clears from the cursor position to the end of the current field to all blanks.
- The **F7** key transfers control to an associated list selection window if one is present.

Dialog Boxes

When the program needs to display a message and wait for a one-keypress answer from you, it displays the message in a dialog box. This is a window which appears in the center of the screen with the message and then a prompt for the key to be pressed. For example *"Press <Enter> to continue"*, or *"Press <Y> or <N>"*. Dialog boxes are used to display messages (normal and error) and to obtain user verification (e.g. a dialog box might ask: "A file by that name already exists. Do you want to overwrite it?").

LET'S DO SOME CONFERENCING

Servers, Conferences, Sessions, Papers, Messages

Computer conferences in Ψ-NET are quite similar to real conferences which are held in hotels and conference auditoriums all over the world. The main difference with Ψ-NET computer conferences is that the conference participants (the speakers, chairmen, and audience) do not have to travel physically to the place where the conference is to be held, but instead use computers and communication systems (e.g. the telephone system) to conduct their presentations and discussions. A second difference is that Ψ-NET conferees may conduct their meetings and discussions any time they wish and at their own rate, and they do not have to be in the same geographic location at the same time as they do with real conferences.

To help you visualize the main components of Ψ-NET conferencing, you can compare them with similar components of real conference systems as follows:

- Ψ-NET Servers are like cities where multiple conferences are going on.
- A Ψ-NET Conference is like a real conference being held in a hotel.
- Ψ-NET Sessions are like real sessions which form every conference, and which meet in various conference rooms in the hotel.
- Ψ-NET Papers are like the presentations and discussions which take place in every session of a conference.

However, unlike real conferences, Ψ-NET gives you an important advantage of allowing you to *simultaneously* participate in multiple computer conferences and sessions going on in multiple Servers located across your state, the nation, or even the world.

In addition to the public conferences, Ψ-NET supports another important means of human communication, the private message. Ψ-NET allows you to send private messages to any other Ψ-NET user who has access to one of the same Ψ-NET Servers as you do.

What Conferences and Sessions are Going On?

When you sent your User Abstract into your Server(s), your Workstation was sent some information about what conferences and sessions are presently going on in each Server. To review this information, from the main menu bar press *"bp"* (Browse-Papers). The program displays a list selection window showing a list of Servers. Select a Server and press **Enter**; and the program responds with a list selection window showing all conferences being held in that Server. Select a conference and press **Enter**; and the program pops up a list selection window full of sessions which are going on in the selected conference.

The names of these conferences and sessions may give you clues as to what topics are being discussed in them. In order to find out more about any conference or session, however, you may wish to look at an appropriate abstract as follows:

1. With a Server, conference and session selected as above, press **Alt-***ac* (i.e. **Alt-***a*, followed by *c*) (Abstract-Conference), or **Alt-***as* (Abstract-Session). This pops up a window containing the appropriate abstract.

2. Press **Esc** when finished reviewing the abstract to remove the abstract window.

You should use the above technique to review and select the conference sessions you wish to join (attend) so that you may begin receiving papers from these on-going sessions, and also sending your own papers (discussions) to them.

Joining and Leaving a Session

Suppose that you would like to join a session which sounds interesting to you. Use the above method to select the session you want, and then press **Alt-***mj* (Mail-Join session) which displays a data entry form with all of the fields containing appropriate values. Press **F10** which causes the form to be accepted by the program as it appears on your screen. You may now select another session of interest and repeat this process as often as you like.

Note: You still do not *have* the papers from these sessions in your workstation; you will find out how to get them later on.

Joining a session will make it easier for you to request new papers being submitted to the session by other people.

If at a later time you wish to leave a session, select the appropriate server, conference, and session as before and press **Alt-***ml* (Mail-Leave session), followed by **F10**.

Receiving Papers from a Server

To get all new papers from every one of the sessions you have joined on any particular Server, press **Alt-***mrn* (Mail-Receive-New papers), and then choose a Server. This request is added to the other requests you have made in your Workstation. It is now time for you to have your Workstation dial your Servers and send all of your requests to them. In return, your Servers will send your Workstation all new papers from the sessions you have joined. If you are a brand new participant to a particular session, the Server will send you *all* of the papers for that session at this time.

Note: If the session contains, for example, several hundred papers, it will take the Server perhaps tens of minutes to transmit all of these papers to your Workstation.

After this first time, **Alt-***mrn* will only get you any *new* papers which have come into the Server which you have not yet seen. To contact the Server go to the main menu bar and press *"md"* (Mail-Do). The Workstation immediately begins the process of communicating with your Server(s). The progress of these communications is noted in a large window in the center of your screen. If the communications were successful (you can tell they were if you don't see any errors in the window), press **Enter** to get back to the main menu bar.

You are now ready to review the new papers and messages you received from your Server(s) and store them in your Workstation's data base.

Reviewing New Papers and Messages

Every time you cause your Workstation to receive papers or messages from the Server, your Workstation places the new items in an *Action* list. You should read each item in this list and then decide whether to delete it (i.e. throw it away) or insert it (i.e. store it) into the Workstation data base. From the main menu bar, press *"ba"* (Browse-Action List). Your screen will now display the Action List in a list selection window which is divided into two portions by a dashed

line across the window. The top portion of the window shows pending messages, and the bottom portion of the window shows pending session papers. The cursor is located on the first message item (or on the first paper item if there are no pending messages). To read the contents of a message or paper move the highlighted bar to that item in the Action list and press **Enter**. To throw away the item press the **Del** key. To store the item in your Workstation's data base, press the **Ins** key.

Note: Unless you **Insert** pending papers and messages into your Workstation's data base, you will not be able to use the **Find** or **Browse** commands to retrieve them.

When you are finished reviewing and deleting or storing the items found in the Action List, press **Esc**. If you do not choose to delete or insert any item, it will be kept in its list and will reappear the next time you browse the Action List (but the item will not be placed into your Workstation's data base - see above note).

> ***Let's EXPERIMENT*** - If you have joined one or more sessions and received the papers from those sessions, you should browse the Action List as explained above, and insert the new items into the Workstation's database. When finished, press **Esc**.

Browsing the Data Base

Selecting ***Browse*** from the main menu will pull down a menu which will allow you to select ***Action***, ***Papers***, or ***Messages*** to browse. Browsing papers will start by displaying a window with the list of servers. When a server is selected, the list of conferences for this server will be displayed. The program will indicate those conferences which contain sessions you have **Join**ed by displaying an ellipsis (...) after those conference names. When a conference is selected (by moving the cursor to the line of your choice and pressing **Enter**), a list of sessions in that conference is displayed. The program will show you which of these sessions you have **Join**ed by displaying an asterisk (*) after those sessions. When a session is selected, a list of the papers from that session is shown in "hit-list" form (one-liners with date, author, title, filename extension, and keywords). The paper list is displayed in the order that the papers were inserted into the data base from the Action list. Note that the one-liners for the papers in the paper list are wider than the window width. The **right** and **left** arrows will scroll the entry line left and right. Everything except the keywords is visible in the initial window.

Selecting a paper or message from the hit-list (by pressing **Enter**) causes the text of the item to be displayed in a window.

> ***Let's EXPERIMENT*** - From the main menu bar enter *"bp"* (Browse-Papers). Select a Server, Conference, and Session from which you have previously received papers into your Action list and **Ins**erted into your workstation's data base. You should see a hit-list of all papers from that session which now reside in your Workstation's data base.
>
> **Note:** Your program automatically installs a Server called **MANUAL** which contains a single Conference called **SPS_INFO** which holds a number of Sessions. The papers in these sessions contain most of the text which comprises the User Guide document. These papers may be useful to you in practicing how to Browse or Find items in the systems data base. Each of the MANUAL papers is filed with certain keywords, and you can use the Find command (discussed later) to locate just the right paper when you need some on-line help.
>
> For now, move the cursor to one of these papers and press **Enter**, and the text of the paper will be displayed. Press **Esc** once to go back to the hit-list.

Browsing messages is similar to browsing papers except that a list of users is shown after a server is selected (no conferences or sessions are chosen, of course, since messages are not

organized by session). When you select one of the users in the list, the program displays a hit-list of all messages which are in the data base, and which have been sent to that user. Thus, by selecting your own name you will see a hit-list of all messages sent *to you* by others, and by selecting another person's name, a hit-list of all messages sent *by you* to that person.

Filling-Out and Mailing Forms

A facility is provided for filling out and mailing forms. A blank form exists as a paper in a session of a conference, and has been prepared by a knowledgable person using the procedure discussed in the section, "Designing a Form" on page 18. To fill out a form, Browse-Papers to the Conference and Session to which you will submit the form, then move the cursor on the paper list for that Session to select the blank form to be filled in. When this context is set, do **Alt-***Mail-Send-foRm* and the form will be displayed to you.

When a form is displayed for filling out, the first line of the form is used to tell you which page of the form you are currently processing, and which page will be the next to be processed. Normally this is the next page in sequence. That is, if you are processing page 1, page 2 will be next. If there is no next page, the top line will indicate that this is the last page.

Pressing **Enter** when the cursor is at the last field, or **F10** anytime, will move you to the page number indicated in the next page field. If you wish to process some other page, simply move the cursor to the page number field with the "down arrow" key and enter the number of the page you wish to process next. When you exit this page, the indicated page will be brought to the screen next. When you are filling out the last page of the form, the next page field is blank, indicating an **Enter** at the last field or the **F10** key will tell the form processor that you are done filling out all the pages of the form. If you wish to return to review or make changes in other pages, use the "down arrow" key and change the page number as described previously. Remember that if you try to select a page number that is not valid for this form (such as "0" or "3" in a 2 page form) you will exit from filling in the form to the next processing step, submitting a paper.

Finding Papers in the Data Base

The ***Find*** function provides a way to search for documents from different sessions, conferences, and even different servers. Invoking ***Find-Papers*** will bring up a data entry form with fields for server, conference, session, filename extension, author, on or after date, on or before date, string within title, and keywords. You can fill in any number of fields, including none of them. Papers that match all of the fields will be searched for . Any blank fields act as wildcards and match anything. If you enter a name for a server, only that server will be searched. If you enter any name for a conference and/or session, only conferences\sessions by that name will be searched. If server, conference, and session are left blank, the entire database (all documents on the workstation) will be searched. If you give more than one keyword to search for, papers are found that have at least one (not necessarily all) of the keywords given.

Other things to note about ***Find***ing papers are:

- All search fields except string-within-title are *not* case sensitive. The values you enter are automatically converted to upper case since lower case letters are not allowed in these fields.
- How long the search takes depends strongly on what fields are filled in on the search form. The fastest searches will occur when many fields are filled in, especially the server, conference, and session fields, since specifying all of these narrows the search down to one session. If these three are left blank, all sessions must be searched.

Finding messages is similar, except that the form does not have fields for the conference or the session, but does have one for "Written to".

If a list of matching documents is found after a ***Find***, they will be put in a list selection window similar to the paper list in a ***Browse***. The only difference is that the server, conference, and session (or server and written-to for a message search) fields for each matching document are put after the keyword information on the hit-list (the **right** arrow must be pressed to view this information since it is past the width of the initial window display). The documents can then be viewed by moving the bar to the line for the selected document and pressing **Enter**.

> ***Let's EXPERIMENT*** - First get a list (hit-list) of all the documents contained in the Workstation data base. From the main menu bar press *"fp"* (Find-Papers) to bring up the ***Find*** data entry form. If no default server, conference, or session have been set by the program because of your previous actions at the Workstation, these first three fields of the form will appear as blanks. It these fields are not blank, let us make them so by doing the following actions to each of them. With the cursor in the first field, press **F5** to clear the field, then press **Enter** and the cursor moves to the next field. Do the same for all fields which show a default value. When all fields of the form are blank, press **F10** to cause the program to accept the entire data entry form as shown. Since all fields were blank, the program will search the entire Workstation data base and generate a hit-list of all documents in the data base. If you did not get a hit-list generated and displayed, it means that either the data entry form had one or more non-blank fields when you pressed F10, or that your Workstation does not have any documents in its data base. In either case you should see the note in the previous section on "Reviewing New Papers and Messages", and redo this experiment.

Assuming you do show a hit-list, leave the list on your screen, and you will now learn how to manipulate a hit-list.

Doing things with a Hit-List

Once you have generated a hit-list on your display screen, you may view any of the documents by simply moving the cursor to the one in the hit-list you wish to see and press **Enter**; or you may wish to use the commands ***Abstract***, ***Sort***, ***Print***, ***Copy***, or ***Delete***.

Sorting lists (Sort)

Document lists that are displayed by either a ***Browse*** or a ***Find*** can be sorted by different fields in the list with the ***Sort*** commands. ***Sort-by Author*** will sort the list by the author field, ***Sort-by Title*** will sort by title, ***Sort-by Extension*** will sort by filename extension, ***Sort-Oldest first*** will sort by date with the oldest first, and ***Sort-Newest first*** will sort by date with the newest first.

For ***Find-paper*** lists it is also possible to ***Sort-by Session***. If you wish to see which server, conference, and session the papers in your list came from, press the **Right** cursor key while viewing the hit-list. This moves the hit-list window over the supplementary information for the hit-list items. In this new window will appear the keywords assigned to each item, and the server, conference, and session from which the item came.

All sorts except those by date are done in ASCII order, and if two fields match, the comparison continues to the next field to the right. For example, if sorting by author, entries with the same author will be sorted by title.

After a sort, the selection bar remains at the same relative position in the list as it was before the sort (e.g. the third entry), so the entry under the selection bar may change.

> ***Let's EXPERIMENT*** - First, you must have a hit-list of items displayed. If you do not, go back to the Experiment under the paragraph, "Finding Papers in the Data Base", and redo this exercise to generate a hit-list. Now press **Alt**-*s* to pull down the ***Sort*** menu. Select one of the Sort commands and

watch the hit-list as you press either the first letter of the desired command, or **Enter**(after moving the cursor to the desired Sort command). The hit-list is re-ordered according to which sort command you invoke.

Printing documents and lists (Print)

While looking at a hit-list or one of the documents from a hit-list, you may use the command ***Print-Document*** to print the document on the printer. If you are viewing a hit-list, ***Print-All documents*** will print all of the documents in the list. When documents are printed, they are preceeded by a header which gives the relevant information such as server, conference, session, date, author, title, and filename extension. If you wish to print the hit-list itself, use the command ***Print-List*** to print the list on the printer preceeded by a header telling what the list is.

Let's EXPERIMENT - You may find it very useful on occasion to print a sorted list of documents. From the hit-list you generated before, use the ***Sort*** command as described above to create a hit-list in the order you desire. Now enter *"**Alt**-p**l**"* (Print-List), and the sorted hit-list will be printed for you.

Copying documents and lists to DOS files (Copy)

You may wish to save a copy of a document or list as a DOS file for several reasons:

- To edit it for use with other files.
- To capture a piece of it to include with another file to be submitted to a conference session or sent as a message.
- To give the document or list a new name, to enable its use with other programs (e.g. editors or spreadsheets).

The ***Copy-Document*** command is like the ***Print-Document*** command, except that it prompts for a DOS output filename to copy the document to, and does not preceed the copied document with a header (the copied file is identical to the document file). The ***Copy-Document*** command copies files in binary so that binary documents (e.g. programs) can be copied also.

The ***Copy-List*** command is like the ***Print-List*** command except that it also prompts for a DOS output filename to copy the list to, and does not make a header.

As with the ***Print*** commands, the ***Copy*** commands are only active if there is a current document/list to copy.

Deleting documents from the database (<Del> and Delete)

You may delete a single document in either of the following ways:

- After moving the highlighted bar to a document in a hit list, press **Del**.
- While viewing the contents of a document, press **Del**.

If there is a current list of documents (a list from a ***Browse*** or a ***Find***), The ***Delete-Listed documents*** command will delete all the documents in the list from the database. The list window is then canceled.

Before anything is deleted, you are asked if you are sure that you want to delete it. When a document is deleted, the entry from the appropriate list is removed also.

The ***Delete-Command List*** command erases all existing commands which are contained in the command list. If you ever enter a command which you do not wish to send to the Server, you must use the Delete- Command List to clear the command list, and then re-enter all of the commands you do wish to send.

Viewing conference, session, and user abstracts (Abstract)

When there is a "current" conference, its conference abstract can be viewed by the ***Abstract-Conference*** command. A current conference can be obtained in one of two ways:

1. You have a ***Browse*** window with a conference selected somewhere on the screen; or
2. The current window is a hit-list of papers from a ***Find***. In this case, the conference abstract for the conference where the selected paper came from is displayed.

If more than one of the above are true, the most recent action takes precedence.

Similarly, if there is a current session (which can be obtained in the same two ways), ***Abstract-Session*** will display the session abstract for the current session.

If there is a currently selected document (you are viewing a document list or the document itself), then the user abstract for the author of the current document can be viewed with ***Abstract-User***.

If any abstract files that are called for by ***Abstract*** are not on the workstation, a dialog box gives you a message that you should get them from the server using ***Mail-Receive-Abstract***.

Communication with the server (Mail)

The functions that communicate with the server are those functions which need to send and/or receive information over a phone connection with the server. All of these functions are in the ***Mail*** menu.

Since only one phone connection is desired for all mailing actions that you want to do, selecting a mail function does not do it right away. A data entry form is given for you to specify the parameters for the mail "command", and the command is added to a global "mail command list". When you are finished issuing mail commands, you must select the menu action ***Mail-Do commands***. This function will process the mail command list by dialing the server(s) and processing the commands. The commands are removed from the list only if they are successfully processed.

Note: This may cause you to experience recurrent problems if for example your command list contains a command which causes a problem which does not allow the complete processing of the command list by the server. In this case, your workstation will retain the old command list (including any trouble causing commands) and will resend them to the server the next time you issue a Mail-Do command. You should use the Delete- Command List command (as discussed in the previous section) to remedy this.

Submitting a Paper to a Conference Session

The ***Mail-Send-Paper*** command puts up a data entry form which prompts for the filename of a DOS file on the workstation to send, the server, conference, and session to send it to, and a title and list of keywords for the paper. Pressing **Enter** in response to the filename prompt will invoke the editor which you chose when setting up Ψ-NET to allow you to create a file to send.

Note: CSSS Network users of Electric Desk will have the following ED Memory Keys (Macros) defined:

- Alt/A - Open up a fresh document
- Alt/Z - Finish entering document and Export it as $$ACT$$.TEX
- Alt/Q - Quit entering document and exit EDS without Exporting it

When your system invokes EDS, you will see the EDS menu. You can then press Alt/A to open a new document, and then press Alt/Z when you wish to export the document to SPS (or Alt/Q if you wish to throw away the document you started to create).

This file is put on the workstation disk with a unique name and is deleted after it is sent. The server, conference, and session fields give the destination for the paper. The title field gives room for you to give the paper a 50 character name which will appear on paper lists. The keywords field gives you a chance to give the paper a few words which capture the major point of the paper and they can be used by other users to locate papers by keyword using the ***Find*** function. The title and keywords are optional, but are highly recommended.

Designing a Form

When you are in the process of designing a form to be used by others, it is useful to be able to put together a draft of the form definition, see how it is to be filled out, put in corrections and additions, and try again. This process may continue through several iterations until you have arrived at the final design of the form. When the form definition is finished you may send it to the appropriate session as a paper, where others will have access to it with ***Mail-Send-foRm*** to fill out their contributions. To support the initial design process, an ***Edit-foRm*** option has been added to the Edit menu.

When you select ***Edit-foRm*** you will be asked for the name of the file which you have created as the initial draft of the form. (Alternatively, if you have not created one, a null response will invoke your editor just as with sending a paper or message). After you are done with creating or modifying the form, the form processor will be invoked so that you may view the form as though you were filling it out. If there are any errors detected in the form you will receive those messages at this time. When you have finished "filling out" the form, you will be asked if you are done editing or not. If you are, you will be asked if you wish to send the form as a paper to a Conference Session. If you do, you will be presented with the regular Mail-Send-Paper screen to fill out. Of course, the file name to be sent, and any default Conference and Session names that are current will be filled in for you. You will supply the title and key words, as usual. If you are not done editing, your editor will be invoked again on the form definition that you are working on for another iteration. (Note that if you are working with a word processor such as Electric Desk, it is not possible to pass the file name you wish to edit to that word processor. You will have to open up a document as usual with the **Alt-**A key sequence, then ask to retrieve a document, and key in its name. This means you MUST remember the file name you gave, or the temporary file name that SPS created so you can ask for it at this time. When you have finished making revisions, you can file (export) in the usual way with **Alt-**Z.)

The ingredients of a form are simple. Fields to be filled in are indicated with the ":" (colon) character, and followed with as many "_" (underscore) characters as you wish to allow for the field width. You may include a " " (blank) character between the colon and the underscore for readability if you wish. You may also precede the colon with explanatory text, e.g.

Phone Number: _____

Blank lines, or other text lines with no colon characters may be included wherever you wish to provide guidance or instructions for the form. Colons not followed by an underscore will be treated as an error. Underscores not preceded by a colon will be treated as comment text, not as a field to be filled in. A field may not be continued from one line to the next, but a separate field may be started on the next line, e.g.

Include a brief description of your experience.

:_____
:_____
:_____
:_____

Forms which are longer than 19 lines must be split into multipage forms. You must indicate where the page breaks are to occur by inserting a line into the form definition which marks the page boundary. That line begins with the characters "$page$". Any other characters on that line will be ignored. For example,

$page$ start of page 2 definitions

Beginning a New Session or New Conference

The ***Mail-Begin-Conference*** and ***Mail-Begin-Session*** commands are used to start new conferences and sessions on a server. They prompt for the server, conference and/or session names. The name for the new conference or session is checked to make sure that it does not already exist.

Also when a session is begun, it is automatically ***Join***ed for convenience.

Receiving Private Messages

The receiving of messages is automatic whenever a server is contacted. So, if any command has been entered into the mail command list for a certain server, the messages from that server will be automatically received. If all you want to do is receive messages from a certain server, you must select the command ***Mail-Receive-Messages***. This command will prompt for a server to get messages from. If this server is left blank, messages are received from all servers. Note that adding the ***Mail-Receive-Messages*** command is only necessary if no other commands are being sent to the desired server(s). In other words, Ψ-NET assumes that you always wish to receive your new messages from other people so it automatically will execute a ***Mail-Receive-Messages*** command along with your other commands.

Sending Private Messages

The ***Mail-Send-Message*** command puts up a data entry form which prompts for the filename of a DOS file on the workstation to send, a server , a list of one of more users to send it to, and a title and list of keywords for the message. As for papers, if ***Enter*** is pressed in response to the filename prompt, the editor you chose when setting up Ψ-NET is invoked to allow you to create the message. All of the users specified must be on the server as specified. The title and keyword fields are as for papers.

Receiving Papers from Sessions

The ***Mail-Receive-New papers*** command is used to get all new papers from all sessions that you joined. It prompts for the server to get paper s from. All papers submitted to sessions on that server which you have joined, and which have not already been sent to you will be sent by the Server to your workstation for display in the Action List.

The ***Mail-Receive-Selected papers*** command is used to get papers from a certain server, conference, and session since a selected date. It is useful as an alternative to use ***Mail-Receive-New papers*** for getting papers from just one session, getting some papers from a

session which you are not a member of, or re-loading papers which have been deleted from the workstation. It prompts for a server, conference, session, and date. All papers in that session which were submitted on or after date will be sent by the Server to your workstation's Action list.If you leave the date field empty, you will receive all the papers for that session.

Note: It is important to note again that you MUST insert papers and messages residing in your workstation's Action list into the workstation's data base before you can use the **Find** operation. This step enables you to throw away unwanted papers and messages before they clutter up your data base.

Ending the Ψ-NET Program

The ***eXit-Quit*** command will end the program and return to DOS. Before quitting, you are asked if you are sure you want to leave, and if there are any pending mail commands which have not been sent, you are asked if you wish to save them for next time. If so, the list of pending mail commands is written to a special disk file on the workstation. This file is always read by Ψ-NET on start-up so that if any commands were saved here from a previous session, you will be asked if you want them to be loaded in.

The ***eXit-Suspend*** command starts a new DOS command shell above the Ψ-NET program presently loaded in your Personal Computer storage. Here, you can do any sequence of DOS commands or run other programs providing you have enough storage to contain both Ψ-NET, and the DOS COMMAND.COM file in addition to that needed by any other programs you wish to run. To return to Ψ-NET, simply type *"exit"* at the DOS prompt. This command is useful for several things including:

- Getting DOS directory information to find files prepared earlier which you intend to send to conference sessions as papers.
- Processing a file to generate information to be included in a document for sending to the Server after Ψ-NET has already been started.

Error Reporting

When Ψ-NET discovers an error condition, it reports it to you using a Dialog Box (colored red if you are using a Color Display Monitor). The program attempts to give you some clue as to the cause of the difficulty such as *"File cannot be found"*. In some cases, the errors are temporary conditions which the program tries to recover from such as when the telephone line to the Server appears to be busy (in which case the Dialog Box will not be colored red).

Message Waiting Indicator

If your Sysop has installed the necessary equipment on the Server and chooses to use the Message Waiting Indicator service, the Server will periodically attempt to call your Workstation by telephone in order to inform your workstation that you have mail waiting at one of your servers. For this service to work, it is necessary that your workstation be in a power-on state and that the SPS program is running. If this is the case, when a server sends a Message Waiting Indicator signal to your workstation, your workstation relays this event to you in the form of a dialog box which suggests that you contact your servers to pick up your messages.

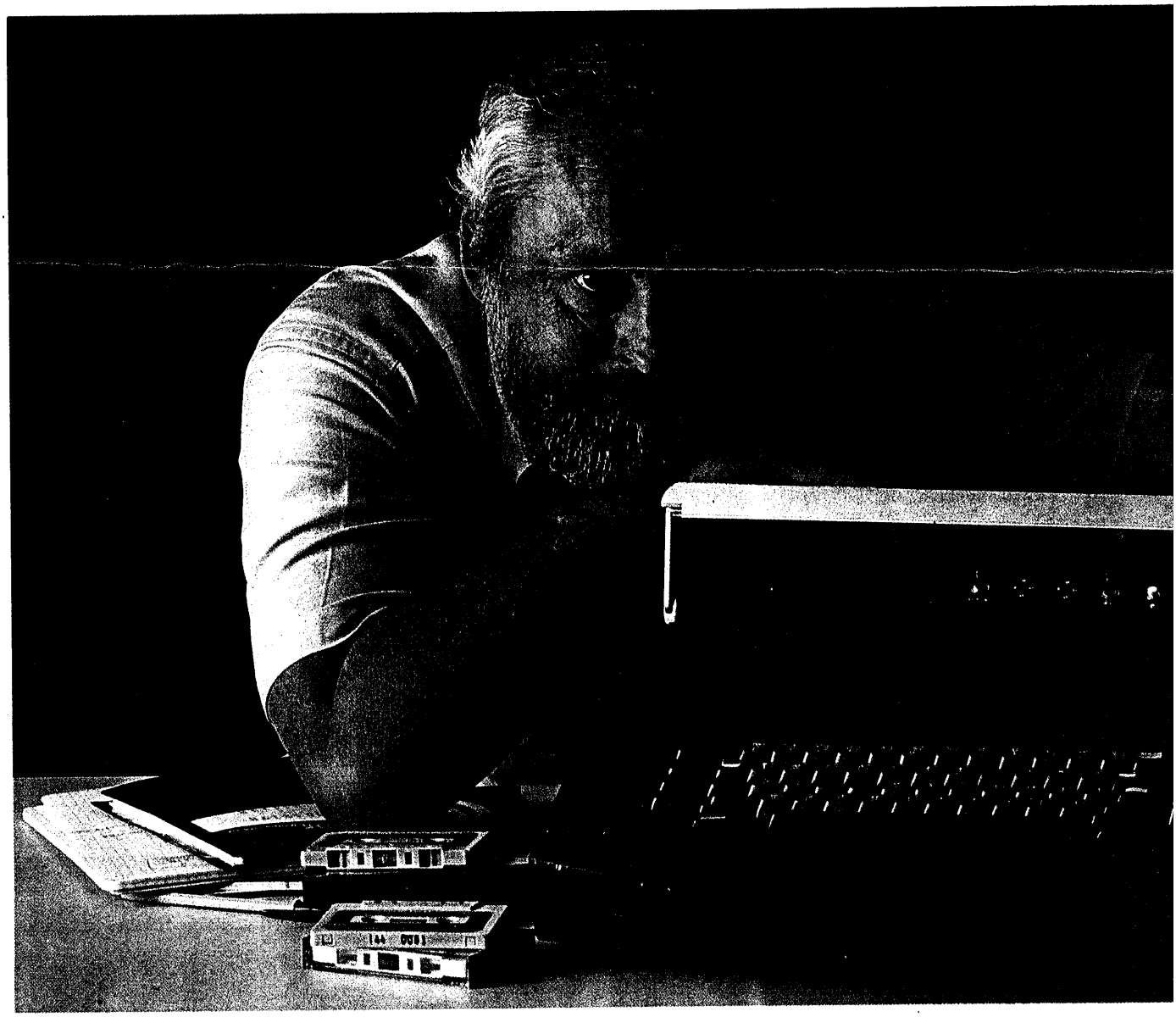
If you use the same phone line for both voice and your Personal Computer, your phone will ring once or twice when a server sends you a Message Waiting Indicator signal. This is a handy way for you to find out that messages are waiting for you, even though you may not be using your Personal Computer or this program at that time.

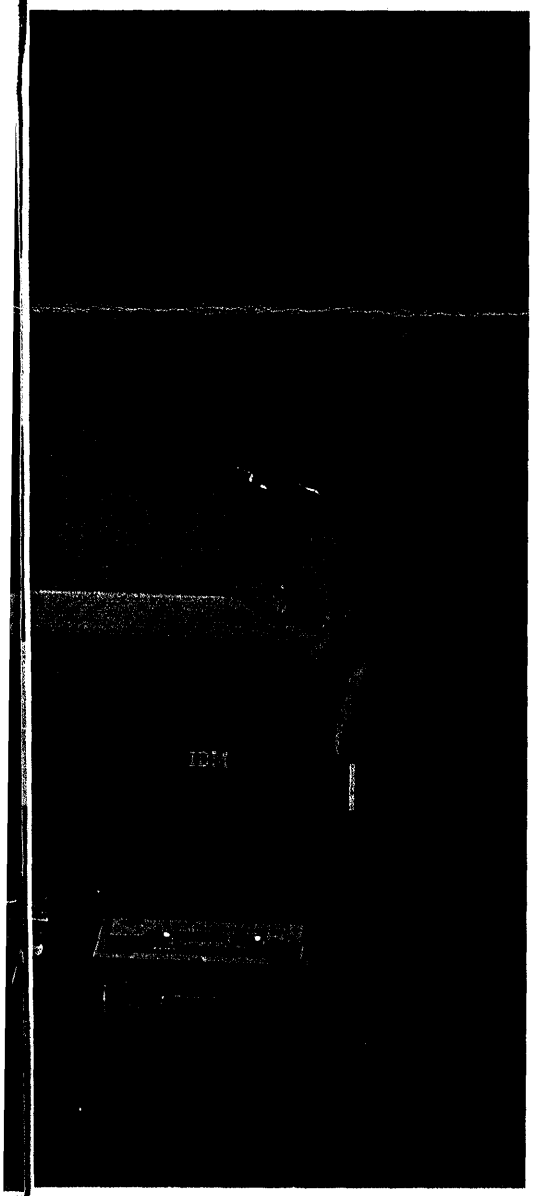
1983

The First Portable Computer

The genesis of SCAMP, grandfather of the personal computer

Jonathan Littman





History is the last thing that comes to mind when we think of computers. Computers are synonymous with change and obsolescence. Many of us who have grown up with or adapted quickly to computers have forgotten what life was like in the stone age of the typewriter and the hand-held calculator. We sit before our personal computers and, like the man driving a Cadillac, have no idea of what life was like in a Model T with its hand-cranked engine and other curious features.

In 1973, when hand-held calculators were becoming popular, the concept of a portable personal computer did not exist; nor was there a reason why anyone would want such a machine. This was unimportant to Dr. Paul Friedl of the Palo Alto IBM Scientific Center.

Not all computers are built by whiz kids who retire at 30 to live off their investments. Dr. Friedl is 50, a family man with three sons and a daughter about to be married. Friedl did not spend his youth playing video games or punching computer keyboards. Those technologies were years in the future. He played baseball.

Friedl became acquainted with computers in an informal way. He studied chemical engineering at the Case Institute of Technology, earning his Ph.D. in 1960. One day, as he relates it, "I found a notice on the board that there was going to be a demonstration of an electronic computer. It was an analog computer, and afterward I thought I knew all about computers. Much to my confusion, the next week there was a demonstration of yet another computer—the digital computer." That computer was an IBM 650, and it caught Friedl's fancy.

Friedl began taking computer classes but rarely saw a computer. Students did everything at a desk. They wrote the program and hand debugged it, all with a graduate student peering over their shoulders. When they finally thought they had the program right, they took it up to their favorite keypunch operator who checked it again and then said something like, "Yeah, that should work." Then, as Friedl says, "The great day came when they would even let you into the same room with the machine. You approached it like the Wizard of Oz—big doors opening up, fans going, tapes spinning, and lots of noise. Then someone took your card, put it into the hopper, and hit the button. The machine went 'chun, chun, chun' and the computer operator said, 'Well, that's the right answer...next.' I said, 'Wait a minute, you mean it's done already?' 'Sure, why not?' " Friedl was hooked.

His friends could never understand why he spent so much time with the computer. They asked why he wasn't boiling oil, pulverizing atoms, or doing something "modern." Instead, he started developing a specialty in computers and became interested in the potential of using the computer for process control in chemical engineering.

Friedl never forgot that IBM Wizard of Oz. After graduating he went to work for IBM Advanced System Development, joining a new group that was studying process control. Prophetically, because of the success of his process control work and long before a personal computer even existed, Friedl became known as Mr. PC. After leading several projects in advanced computer control, he joined the fledgling IBM Scientific Center. He became the Project Leader on applying real-time computers to high-energy physics. In the late '60s, he worked on several projects with microprocessors.

SCAMP, grandfather of the IBM PC

Project SCAMP

In December 1972 Friedl was sitting in for his boss when he received a call from the IBM General Systems Division in Atlanta. General Systems wanted to see if it could raise the visibility of APL (A Programming Language) in its division product line. Friedl's name had been advanced as someone who knew about microcomputers. Two executives flew out to California the next day to ask Friedl if he could come up with something using APL. "We don't exactly know what," they said, "maybe something similar to a hand-held calculator." When Friedl told them that he would love to work on the project, they asked when he could come back with a proposal. He said one month.

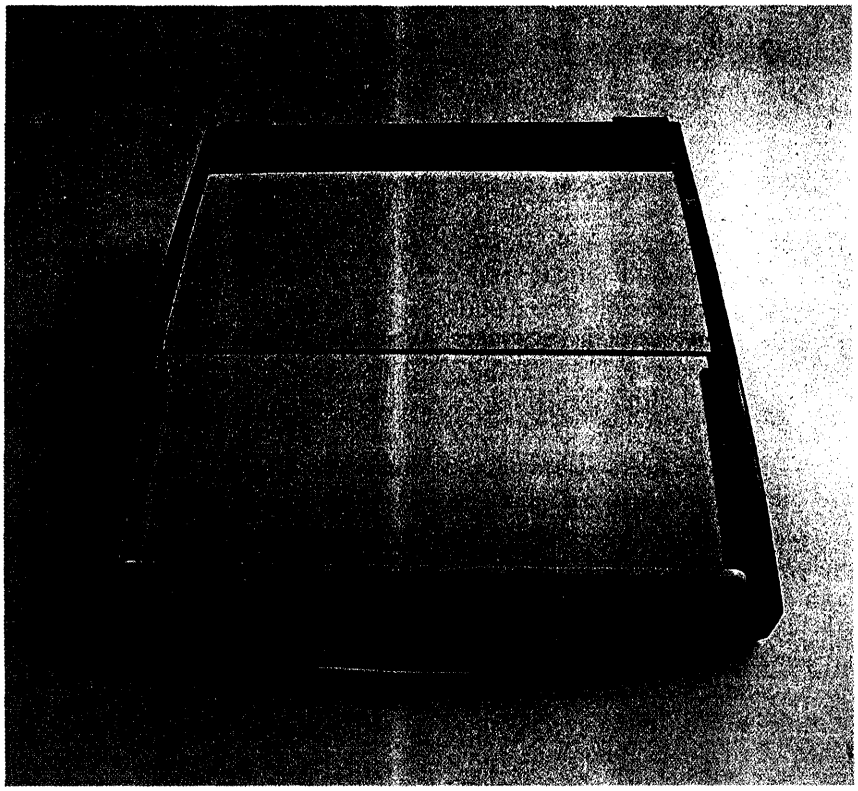
Friedl had the notion of a portable personal microcomputer, but it wasn't going to be easy to get the most out of a full-fledged high-level language like APL. The technology didn't seem ready. What was needed was a display, keyboard, printer, and a magnetic data storage device. At this time there were no floppy disks, just big bulky tape drives, and the idea of a portable display was rare.

His friends could never understand why he spent so much time with the computer.

Friedl had to devise a useful machine, while somehow circumventing the typical five-year development cycle. Says Friedl, "The first thing I did was enlist the help of an excellent programmer-engineer, Pat Smith. We considered a number of IBM and other technologies already in existence that we could incorporate into the machine. Fortunately, we didn't have to create a new microprocessor for the computer."

IBM manufactured a suitable processor, the PALM microcontroller that was designed to control other elements of a microcomputer. It offered Friedl the flexibility he would need to make an all-purpose machine, incorporating within its architecture the notion of an integrated raster (TV) display. He used a standard IBM keyboard and 64K of RAM (more memory than many of today's computers). Curiously enough, the memory cards were called Snoopy Cards, and since they were 16K each, only four were needed. The last basic piece of hardware was a new I/O card that allowed the computer to hook up to a printer, audio cassette recorder, and keyboard.

On January 22, 1973, with his ideas sketched on paper, Friedl took his plan for a portable computer to



The portable SCAMP prepared for travel

Atlanta. He told IBM executives that the only way to demonstrate the feasibility and uses of a portable computer would be to build one. When asked how long it would take, Friedl said six months.

At 30,000 feet, on the plane back to California, Friedl began to wonder what he had gotten himself into. He already had a full-time job, yet he had just agreed to build something that had never been built before. "I was excited about the project, but I can't figure out why I said six months. I didn't have a single worker or the assurance that it could be built in two years, let alone six months.

Friedl needed an engineering team. He went to see Joe Ma, engineering manager at IBM's Los Gatos Advanced Systems Development Laboratory. Ma had a big blackboard in his office. Friedl said, "Can I erase your board Joe? This is what I want to build...." He sketched the whole thing out. When he was finished, Joe said, "Boy would that be fun." Joe George, who had been working on a printer adapter for the PALM microcomputer, came in to look at the sketch. Friedl asked him if he thought it could be built. George nodded. Then he asked George how long he thought it would take. George paused and then calmly said, "Six months." Friedl breathed a sigh of relief.

Not only was Friedl blessed with available technology, he was also fortunate to have a tremendous team. Joe George had recently joined the staff of an IBM knight, Roy Harper (IBM "knights" a few of its exalted workers, allowing them to work on virtually any technical project, or "dragon," they wish). Because the knight was between dragons, George was able to work with Friedl. In only two weeks Friedl rounded up five engineers and five programmers. The project was christened SCAMP (Special Computer APL Machine Portable).

That was only the beginning. As Friedl says, "Once everyone else was busy making the system, I started thinking, 'so what! Why would anyone want such a thing? What possible use could a manager have for a portable microcomputer?' I was trying to figure out how to make it more than just a magical black box.

"I was thinking about six months into the future, when SCAMP would be sitting on the executives' desks. I realized that I never wanted them to be faced with a blank screen. The system should tell you all the possible choices you can make. I wanted menus, so that someone who had never used a computer could read the choices, make their decision, and then press the appropriate key. Originally, I thought up four or five possible applications, such as computer-aided instruction, project planning, and financial analysis." Friedl even developed word processing, but the APL implementation of the word processing was, as Friedl says, "horrendously slow." Instead, he created what was probably the first electronic spreadsheet.

A full three years before *VisiCalc* was born, SCAMP was able to do much of the financial modeling found only in today's modern financial programs. Says Friedl, "While the small display allowed you to see only a window of 16 rows down and 64 columns across at any one time, your spreadsheet could actually be much larger. With the power of APL it was a trivial matter to do some of the basic modeling and calculations you find in modern programs." Friedl had an old picture of SCAMP, and sure enough, the display showed the year, 1973, with months going across the screen and financial categories listed in the left corner.

SCAMP's Test Run

Six months after his promise, virtually to the day, Friedl and Joe George were on a plane to Atlanta taking the world's first portable computer on its maiden voyage. While

Community

Friedl had planned well ahead for this day, it wasn't until he saw the executives staring up at him from the conference table that he realized the enormity of his task. Says Friedl, "I had planned numerous applications for the executives to do on SCAMP. Talk about apprehension. These poor people had either never had typing or the skill had evaporated. And then, all of a sudden, to have something thrust at you—here is the keyboard—and your peers waiting for the slightest mistake. It was terrifying for some of these executives. All they had to do was press a letter or number, but there were questions such as, 'Where is the A key?' One man broke into a sweat when I asked him to type his name."

People had difficulty understanding SCAMP. Friedl remembers one executive saying, "Gee, that thing almost does what a computer does!" Says Friedl, "We had to convince him that it was indeed a computer." C.B. (Jack) Rogers, president of the General Systems Division, liked the machine and wanted every manager in the Atlanta headquarters to take part in a demonstration. Everything went well until Friedl got fancy and tried to show them how portable it was: "I reached for the handle and pulled it shut, and said, 'See, now the computing goes with you!' While everyone was clapping and shaking hands, I was wincing inside. I was sure I heard a crunch as I closed SCAMP. I tried to grab Joe George as unobtrusively as possible. We tried repowering the system but got nothing. Joe pulled the back off and announced that it was dead. I said 'What do you mean Joe?' He then explained that I had sliced the cable for the keyboard. The executives were starting to look overly curious, so we asked them if they would give us 20 minutes alone with SCAMP. And so, in a fully enclosed, oak-paneled conference room, we opened SCAMP's guts on the huge walnut table and pulled out the soldering gun—talk about smoke!"

That was not the first time Friedl was concerned about his electronic baby. Before SCAMP left for its maiden voyage, Friedl was called over to the Los Gatos laboratory. The next day he was to fly to Atlanta to give the opening demonstration to Rogers. Says Friedl, "They had made a special wooden case for SCAMP. It was two toned with nice wood and was quite attractive. I was standing there admiring SCAMP when Joe walked up and said, 'Watch this,' and gave SCAMP a little kick. I felt like screaming. The whole project was about to go splat on a hard tile floor.

We opened SCAMP's guts on the huge walnut table and pulled out the soldering gun—talk about smoke!

And then I couldn't believe my eyes. Instead of going crunch, SCAMP just sort of sighed to the floor, rocking back and forth like a baby carriage. Jerry Garvis, an industrial designer, had created a special curve to give it a dampening effect. Of course, nobody had told me."

Excluding the initial cable-slicing incident, Friedl went on to give over 100 trouble free demonstrations. Exhausted, Friedl was at home enjoying his first real weekend in months. The phone rang. It was Atlanta. Friedl was to get on the next plane. It was time for the ultimate demonstration. John Opel, the president of IBM, wanted to meet SCAMP.

By Monday afternoon Friedl was in the conference room of division headquarters in Atlanta. The room

was full of executives peering nervously at the darkening thunder clouds. Friedl describes the situation: "Opel had to give a speech later that evening in Atlanta. He was going to take a company plane to Atlanta and then drive out to headquarters. With the storm gathering, we knew planes would be stacked up over the airport. We started to realize that he wouldn't have time to drive all the way out to headquarters and still get back to Atlanta in time for his speech.

"An IBM vice-president took over. He started pointing at people, saying, 'You get us a room at a hotel at the airport! You get all the flip charts! You pack this machine up! And you, meet him at the airport!' In a couple minutes they were all running down the hall. They had forgotten all about me. I yelled out, 'Where are you going?' and they cried back, 'The Hyatt on Peachtree Street.' "

An hour later, in the middle of a downpour, Friedl had driven down just about every Peachtree Street in Atlanta. Finally, he found the hotel, parked across the street, and ran through a thunder shower to get there. Says Friedl, "I was hoping to give a nice polished demonstration, but I was leaving a trail of water behind me. As I came down the hall, they called out, asking where the heck I'd been. Luckily, there wasn't time for explanations. I powered up SCAMP, and just as the menu came up, Opel walked in." Opel then pulled up a chair, and the two of them sat down next to SCAMP in the middle of a large dining room. SCAMP performed beautifully.

When the demonstration was finished, Opel said, "It's all very interesting." Then Dave Slattery, a vice-president, spoke up, "We'd like to get financial backing to go ahead with the project." There was an agonizing silence. Finally, Opel said, "Don't look at me—I don't have any money." Then, after another lengthy pause, Opel said, "Why don't you just go over budget."

Community

Ancestor of the PC

IBM went on to develop the progeny of SCAMP, the 5100, consulting with Friedl on a number of issues, including some experimental applications software. With the help of a class of students at Stanford, Friedl developed a set of interactive statistics programs. He devised a menu-driven system that had the capability of capturing each entry, allowing you to see the sequence of keys you pressed. The system was ideal for identifying where you made your mistake.

Some time after the last demonstration for Opel, when all the excitement had died down, Friedl's chief programmer, Pat Smith, returned from a vacation in Hawaii. Friedl called him up and invited him to sit in on one of the demonstrations he had done countless times. Smith accepted. Friedl was running through the different applications and was about to show Smith the hold key, which acted just like an interrupt key. If you were scrolling information up, it would freeze the screen. When you wanted to see more, you just pressed it again. As Friedl reached out to press the hold key, Smith grabbed his hand: "Don't press that key. I was trying to debug that thing before I left, but I just couldn't get it fixed in time. It doesn't work."

Friedl wasn't convinced: "That can't be, Pat. I used it in over 100 demonstrations and never had a problem. Watch." Friedl pressed the key, and the screen went poof—system blowout. He couldn't believe it. The system took over five minutes to reboot. If that had happened during the demonstrations, it would have been the end of SCAMP. They tested the key over and over again, but a blowout happened almost every time. To this day they haven't figured the problem out. As Friedl says, "It must have been a certain sequence that, by some stroke of luck, I didn't repeat in the other 100 demonstrations."

Despite all its triumphs, SCAMP was almost thrown out with the trash. When the project was finished, Friedl went on with his work and forgot all about the wooden-encased computer that had consumed his life for over six months. He became curious about the whereabouts of SCAMP when a colleague showed him an article proclaiming that the 5100 was the world's first truly portable computer. What the writer didn't know was that the portable computer technology of the 5100 evolved from SCAMP. Says Friedl, "I always felt the work we had done was worthwhile. But with the explosive growth of personal computers, I suddenly thought that perhaps we had played some historical role in that popular movement."

Friedl called up the Los Gatos lab and asked if SCAMP was still there. IBM technicians told him that it was, and if he wanted it he had better come quickly because they were just about to throw out a lot of junk. "I jumped in my car and drove over there. We searched back in the corner of the lab and finally found SCAMP covered with dust and cobwebs behind a work bench. After all that time, we plugged it in and it started right up, a tribute to all the fine people who worked on the project."

And so SCAMP was born, the father of the 5100 and grandfather of the IBM PC. Although SCAMP was based on a technology that preceded the age of floppy drives and 16-bit processors, the lineage is clear. Sometime when you're feeling nostalgic, look at the back of your PC. Right above the power plug are the numbers 5150, child of the 5100, grandchild of SCAMP.

Jonathan Littman is a freelance writer who is working on a book about IBM PC communications. He also conducts training seminars on WordStar *and develops educational software.*

```
type a:ucfriedl.scr
```

```
@MSG:FROM: FRIEDL  --PALOALTO TO: KOLSKY  --PALOALTO          11/10/89 09:10:15
 To: KOLSKY  --PALOALTO

 *** Reply to note of 11/09/89 22:22
 From:    Dr. P. J. Friedl
 IBM Scientific Center
 Palo Alto, California
 Subject: NO SUBJECT

 Harwood, I was going to be on vacation, but I'll swap the day with another.
 November 22 about 10:00 sounds fine.  I'd like to also take advantage of the
 visit to try an experiment which goes like this ...

 I'll bring along a PS-2 which has my PSInet system on it.  The PS-2 is
 prepared to hook up with ethernet at UCSC.  If we can have someone at UC
 give me a temporary internet address for the UC ethernet, I will then attempt
 (for the first time in history, and without the fingers leaving the hand)
 to have a PSInet session across NSFnet.  I have set up a PSInet server at
 Stanford (which is BARRnet HQ), and UCSC is a member of BARRnet, so we should
 be able to pull this off.  This is a real feasibility test of PSInet on
 NSFnet, which was requested by NSF in Washington.  I'll need to work with
 someone at UCSC who knows how to assign internet addresses from UC to
 BARRnet.  This should only take 10 minutes.  If all works fine, then I'll
 even have a live PSInet to demonstrate to you and Pat.  Think you can
 arrange for this?  If so, I'd like to get to UC by 9:30 in order to get
 the net connection made.  Please tell me the turn-by-turn instructions on
 where you want me to go; last time I got lost on campus.

 Paul

 VNETID: FRIEDL at PALOALTO    TIE LINE: 8/465-4113
  rNO SUBJECT                                                          R

C:\>
```

1975 DRAFT

Harwood,

Your comments, please

Paul

[redacted]CONFIDENTIAL

PADC-16-IBM

AN ARCHITECTURAL STRATEGY FOR DEVELOPMENT OF EVENT-DRIVEN MICROSYSTEMS

Dr. Paul J. Friedl
Palo Alto Scientific Center
August, 1975

TABLE OF CONTENTS

1.0 INTRODUCTION

The market opportunity for hierarchical distributed systems (HDS) has been presented in CMR Statement No. 4997 (12/3/74). The present paper applies to this CMR Statement, and presents a design approach for achieving a timely solution to a number of IBM event-driven system requirements.

2.0 PRESENT ENVIRONMENT AND TRENDS

2.1 Technology

As progress in electronic technology continues to lower the price of computing hardware, the increasing complexity of todays system requirements continues to drive the cost of software support upwards. These trends will continue into the future and cause greater and greater proportions of system development costs to be allocated to the software support component. Even now, the cost of support software for today's event-driven systems generally exceeds that of the hardware.

2.2 Microsystem Proliferation

The lowering price of hardware has made possible the proliferation of mini and microsystems of many types and from many manufacturers. IBM has itself developed a great variety of microsystems for use in various products. These microsystems, however, are not compatible with each other, and therefore each system requires its own unique software support. When taken across all IBM product divisions and across all IBM microsystems, the cost of writing and rewriting such redundant software is a significant drain on development resources. The drain is so significant that it nearly always limits the extent of support software for any individual system to the barest essentials (eg. a basic assembler and simulator), hardly ever including high-level language or operating systems support, or program production and debugging aids. Moreover, the skills needed for developing high-level language, and operating systems are not widely available, thus, not only further increasing development time and expense, but also the risk of producing less than desirable software quality. These effects appear to be mainly responsible for our present situation where none of IBM's microsystems has a real-time event-driven operating system as capable as those developed by IBM in the 1960's.

2.3 New IBM Systems Architecture

The modern swing towards placing more function in hardware (and microprogram) has increased the need for developing more microsystems and related programming (ref FS). The HDS requirement also states a large need for multiple interconnected computers including mini, midi, and maxi varieties, all requiring large amounts of new software

to be developed with the massive expenditure of time and IBM resources.

3.0 PROPOSED STRATEGY

It is proposed that the optimal technical solution to the development of an advanced IBM microsystem be based upon a design which is highly reliable, offers a wide range of cost/performance/applicability, is well understood, and (above all) for which there exists an abundance of complex event-driven systems software support. More specifically, it is recommended that the optimal approach (i.e. lowest cost, highest reliability, fastest development time) would be the development of an extended 1800-based architecture.

both IBM and customer software

3.1 Hardware

To the existing 1800 design could be added any of a number of enhancements including larger memories and DASD, newer terminals and displays, dynamic address translation, vector operations, hardware stack, network and security control, etc. The design would feature full downwards instruction set compatability with the IBM 1800. System implementations could be executed at the low cost entry end by extensive use of Programmable Logic Array (PLA) technology, and at the high performance end by means of Emitter-Coupled Logic (ECL) technology.

3.2 Software

This plan would begin with the availability of the TSX, MPX, and PALS operating systems (at no cost). New software development dollars would be invested in supporting the new peripherals, network control, distributed data base management, and distributed tasking (not assemblers, macro-assemblers, simulators, compilers, interpreters, etc.).

3.3 Migration

Existing customers for 3790 and related UC .5-driven devices could be migrated to the new system by providing a co-resident UC .5 (or Trinity) instruction set via logic or via emulation. Notice that if the new ECL version of the system could perform 1800 instructions at a rate of approximately 20 MIPS, the system could itself be considered and used as a microsystem for building (emulating) other systems. This concept would make available to all of IBM a microsystem with a complete real-time micro-operating system environment (somewhat related to that proposed for FS). This development would leapfrog present industry trends, and may even result in an "industry first" for IBM.

3.4 Applicability

The proposed system would not only serve as a component of other systems (eg. HDS etc.), it could also be considered as a standalone product for direct sale along the lines of Peachtree. Thus, this software development investment is able to gain even more value than with other approaches.

HYPERNODE

A High Performance Data Node

submitted by

Paul J. Friedl

October 2, 1978

THE HYPERNODE CONCEPT

The main objective is to develop a powerful personal tool for users in scientific, engineering, and advanced business applications. HYPERNODE would give the user the following capabilities:

1) at least 2 MIP as a System/370. (includes Floating-Pt hardware)

2) up to 2 MByte of real storage

3) at least 16 MByte of virtual storage

4) a highly interactive operating system with full screen support for context editor, file manager, data base manager, and program generation, test, and execution facilities. Included will be an executive to handle multiple concurrent program and task execution, all physical I/O devices, and external interrupts.

5) a color display of at least 512 x 512 pels with line drawing capability, and a user-programmable character set of 256 characters.

6) a resident fixed disk storage device with at least 100 MByte capacity.

7) a floppy disk drive

8) a flexible communications adapter for use up to 50 Kbaud

9) an optional IEEE 488-197 adapter (including interrupt)

10) a high speed cycle-stealing channel adapter

11) an optional array processor attachment

12) (release and use of microprogram interface by user to be investigated)

13) a real-time clock

THE BUSINESS OPPORTUNITY FOR HYPERNODE

HYPERNODE is aimed at the same IBM customers who used the 7090, 1620, and 1130 systems of the past. These users were not only leading scientific and engineering institutions, but also advanced commercial accounts. The intent of HYPERNODE is to attract this same category of customers to use a modern IBM product as a personal data base oriented tool in either a stand-alone or distributed network fashion. HYPERNODE is the 7090 of the 1980's, but with a new touch - it's six times as powerful as the 7090, fits in a desk, offers powerful interactive data base and display facilities, and if you're the user it's all yours!

PROJECT PHASING

This project will take place in three main phases:

Phase I - the design and fabrication of a number of prototype systems by IBM. (1/79 - 4/80)

Phase II - the installation and test of the prototypes at all 25 major universities (plus selected other customers) via multiple joint studies. (4/80 - 4/81)

Phase III - the development and manufacture of an IBM product family based upon the results of the first two phases. (10/80 - 2/83)

ADVANTAGES

This proposal has the following advantages to DPD:

1. allows IBM to work on a product aimed at the university and research areas.

2. at Phase II time, allows IBM to closely interact with all joint study partners, giving them a chance to voice their opinions on the product's limitations, extensions, applications. (Such close interaction is exactly what is being sought at the present time.)

3. this product (when proven feasible) would undoubtedly have broad application to business and commercial applications as well.

4. would develop a timely DP product for use in distributed IBM systems in the 1980's.

the following advantages to GPD:

1. offers a new systems context for new storage technology (eg. MENDOCINO, bubbles,)

2. provides a possible basis for distributed data base product hardware

3. explores the new application potential for a new product which combines data storage with display and CPU power in a small integrated package.

THE HYPERNODE PROTOTYPE (Phase I)

The prototype would be developed around as many available hardware and software components as possible. Leading candidates are as follows:

CPU - 801/Sequoia (GPD)
Color Display - Rainbow/Hacienda (RD/SCD)
Disk - Piccolo/Mendocino (GSD/GPD)
Floppy Disk - (GSD/GPD)
Packaging - all components except color display and keyboard to fit inside a standard office desk-sized enclosure. (Alternative credenza-sized enclosure will also be investigated).

HYPERNODE PROGRAMS

Major systems support

- CMS
- RASP to be investigated

New software needed

- full screen color display
- real-time multiprog. executive
- *

Language support

- APL
- BASIC
- PASCAL *
- FORTRAN

Data Base System

- STAR

STAFFING

Though many alternatives exist, one proposal would have all three Scientific Centers collaborate on the required HYPERNODE software.

Cambridge - CMS, RASP, real-time multiprog. exec.

Los Angeles - full screen color display

Palo Alto - PASCAL, STAR

Engineering support is needed from an appropriate group. One possible group is GPD Advanced Development in San Jose/Los Gatos.

FUNDING

Detailed figures depend upon which staffing alternative is chosen, how many new hires needed, engineering costs, etc. A first order estimate for Phase I development costs is on the order of $1.2 Million, not including prototype hardware costs.

THE ORIGINAL PERSONAL COMPUTER,

THEN ... and NOW

Paul J. Friedl

"In the beginning were the words,
and the words became SCAMP,
and SCAMP dwelt among us,
and gave us PeaCe." - PJF

This year marks the tenth anniversary of a milestone event in the history of computing, the creation of SCAMP, the world's first personal computer. On this occasion it seems fitting to recall SCAMP not only as a project and as a device, but also as a revolutionary concept.

SCAMP - as a revolutionary concept

Webster defines revolution as "a sudden, radical, or complete change" and evolution as "a process of continuous change ... a progressive development". By these definitions, it would appear that SCAMP as a concept was indeed revolutionary, whereas its realization as a device was evolutionary.

In the heyday of centralized systems it was a radical change to consider building a system which could go to and work with a single user as opposed to requiring multiple users to approach a common shared system which often had insufficient ports, storage, or computing capacity available.

This notion to bring computers to people was supported and stimulated by several key trends. Integrated circuit technology had become sufficiently advanced, and high-level programming language technology had combined with interactive computing techniques to form systems such as APL[1], a very popular

and powerful programming system. Why not combine these technologies into a portable personal computer which would free the user from some of the constraints of timeshared systems.

SCAMP - as a project

In January 1973, this author proposed to IBM management a design for SCAMP (Special Computer, APL Machine Portable) and a project plan for developing it as a prototype system. I was commissioned in Atlanta, Georgia by Jack Rogers[2] (President of IBM's General Systems Division at the time) to lead a project with the ambitious goal of developing a SCAMP prototype in six months. Was I mad? I had agreed to lead a fully-funded project to develop a revolutionary concept into a working prototype in a very short time, and I didn't have any people or resources to work with. (Moral: Always be prepared to take yes for an answer.)

Good fortune smiled down on this project throughout, and within two weeks enough people with the needed skills were borrowed from other IBM groups in order to staff a hardware team in the IBM Advanced Systems Development Laboratory in Los Gatos, California[3], and a programming team in my laboratory, the IBM Scientific Center in Palo Alto, California. We were also fortunate to have as team leaders our Chief Engineer, Joe George and our Chief Programmer, Patrick Smith. Together, these teams took the idea and formed the hardware, microcode, and software into the SCAMP prototype which we delivered on schedule to Atlanta six months later. Let's take a look at what was done.

---

[1] Iverson, K. E., *A Programming Language*, John Wiley & Sons, New York, 1962

[2] C. B. Rogers, Jr. is now Corporate Senior Vice President and Group Executive, Information Systems Group

[3] Now part of IBM's General Products Division in San Jose

SCAMP - as an evolutionary device

The architecture of the SCAMP system was quite conventional and is shown in Figure 1. Moreover, the short time available for development meant that the system had to be built from existing hardware and software components as much as possible. The thing that was new about SCAMP as a device was its packaging - the missing link needed in the evolution of the personal computer. Thus in 1973, SCAMP represented a significant physical design departure from the systems at that time, even though all of its components were quite conventional. Solid state technology had become sufficiently compact to allow the circuit cards for a microprocessor, 64 kilobytes of read/write memory, and device control logic to be packaged in the same frame with a CRT display, keyboard, and tape cassette drive. To make the system truly portable, all of the components were mounted in a set of covers having the appearance of a small piece of luggage. This package is shown in Figure 2.

To operate the system, one had merely to place the unit on its side, slide open a hatch cover, and pop the system into its raised position as shown in Figure 3. A new high-frequency switching power supply was being simultaneously developed but was not finished in time to allow integration into the system. A conventional standalone power supply was therefore used throughout the project.

As previously mentioned, the major system components were already available from various sources. Thus we selected a 5-inch CRT from Ball Brothers, Inc. for our display, a Norelco audio tape cassette recorder as a secondary storage device, a keyboard from IBM in Raleigh, NC, 16 Kbyte memory cards from IBM in _____, and a PALM (Put All Logic in Microcode) microprocessor from IBM in Boca Raton, FL.

The necessarily short development time had a similar effect upon our software plans, namely that we could not afford to develop a new APL language

processor from scratch. The solution was therefore to emulate[4] a processor for which a suitable APL system already existed. With this as a strategy, we chose to write an IBM 1130 emulator in PALM microcode and then "plug in" almost all of an 1130 APL system which was available. In this way, we were able to replace a several person-year programming effort with one of several person-months.

SCAMP - as a Personal Computer

With both of our development teams blazing away at their respective hardware and programming activities, I had a chance to work on several other problems and questions. So you have a personal portable APL machine, what do you do with it? How should applications be presented to the user? How could SCAMP best be handled as a product? How could I get these messages across to IBM management?

To the first question there were several answers: one could use SCAMP as a super desk-top calculator, as an interactive APL programming device and as a "dispenser" of canned application programs. The first use as a desk-top calculator, though interesting, was not in itself persuasive because the cost of the system seemed to far outweigh this usage. The use as an APL programming device was much more appealing, but the first sale had to be made to high level company management who were not, in general, experienced APL users. It seemed that my best shot was to demonstrate SCAMP's capabilities in the third area, i.e., as an application machine for non-programmers.

In approaching this task, I believed that the computer should direct the interaction with the user by always presenting a question (along with the form

---

[4] Emulate: In computing - to stimulate the operation and behaviour of one computer using microcoded procedures on another computer.

of possible expected answers) or a multiple choice menu. In addition, I felt it quite important to have a "HELP" facility which could be invoked at any time. Figure 4 shows the contents of the main MENU screen which I used to demonstrate SCAMP to management. This menu lists the types of applications which I believed might best project the wide capability of such a machine to my audience. Some examples of the HELP function and applications appear in Figures 5 through _____.

In spite of the above efforts to show how easy SCAMP was to use, how could I be sure that an audience composed of inexperienced users would get this vital message? The answer hit me like a thunderclap - I would make my inexperienced user manager audience execute the SCAMP hands-on! This may seem obvious to you, or trivial, or old-hat - but let me assure you that this was a revolutionary approach at the time. Many of my "victims" had not had recent typing experience much less hands-on computing or programming experience. When demonstrating SCAMP to various IBM executives, I had no notion of how much pressure I'd be putting them under by asking them to execute a simple (to me) demo. Here I was, insisting that they perform tasks which they had never done before, in front of their peers (I hope all of these fine people have either forgotten or forgiven me.). The outcome was superb - no system failures, and my new set of no-longer-inexperienced users performed beautifully. I knew we had "arrived" when one of the execs said, "Gee, anyone should be able to do this.". (Boy, was he right!)

SCAMP - as a product

There were a number of challenging questions which needed answers. How should such a product be mass produced? How should it be marketed, installed, and maintained? Traditional IBM methods and service used to support the larger

computers did not apply here. Thus, as part of the SCAMP "package", a number of new approaches were proposed, including:

- The notion of customer installation,
- Machine problem determination by the user,
- IBM retail stores.

All of these approaches are of course commonplace today.

SCAMP then; PC now; WHAT next?

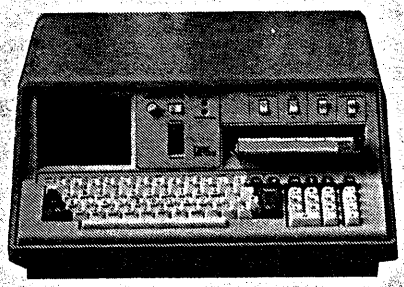
As many readers may know, SCAMP very directly became the IBM 5100. Succeeding generations became the IBM 5110 and 5120 systems, and now we have the IBM 5150, the PC. The family portrait is shown in Figure _____. When one observes the striking similarities of most of today's personal computers with SCAMP and each other, one may wonder if this vein of computing has at last paid out. I submit that we are seeing not the beginning of the end to the progress of personal computing, but rather the end of the beginning. Just as SCAMP introduced a revolutionary concept with an evolutionary technology, its descendants will themselves evolve and by combination with other new technologies will further revolutionize computing. Thus, worldwide communications networks, knowledge-based systems, and voice processing will join with personal computers to form exciting new generations of systems to plan, advise, teach, play, and communicate with users everywhere. Who are the users? Ten years ago SCAMP met the users, and they is US!

ACKNOWLEDGEMENTS

SCAMP belongs in many different ways to many IBMers. I wish to thank Patrick Smith and Joe George for their outstanding work, and all of the following people for their participation.

Roger Abernathy
Christopherson
Ed Finnegan
Roy Harper
Paul Hodges
Dell Hollenbeck
Jerry Jarvis
Phil LaVeau
Joe Ma
George Marenin
Art McCarthy
John McPherson
Joe Myers
John Opel
Kitty Stark Price
Dennis Roberson
Jack Rogers
John Rudd
Dave Slattery
Lou Stevens
Greg Tobin
Bill Tutt

IBM's First PC



Mike Shadick

The year was 1973—nearly ancient history, by microcomputing standards.

The event was IBM's proud, and laudably premature, introduction of its Model 5100, the world's first truly portable computer.

Its potential competitors—Apple, TRS-80, and all the rest—were then little more than microgleams in the eyes of their respective researchers and developers. Indeed, there was nothing else in existence even remotely comparable to the Model 5100.

Which, ironically, seems to have been the very crux of the problem.

Not that the 5100 itself had any problems. It showed every prospect of being the answer to a statistician's, an engineer's, a scientist's, and a financial and business analyst's dreams.

"Now you can have a computer right on your desk, exactly where you need it," began the advertising copy. "It incorporates the latest in semi-conductor technology. It features a typewriter-like keyboard and numeric key-pad for simplified data entry, a 1024 character display screen, an integrated magnetic tape drive, and 16K characters of memory."

What else could one ask for? Especially in 1973?

So why did the 5100 fail to take the personal computer market by storm? Because *the market had yet to come into existence.* Professionals and private parties alike had not yet even begun to see the possiblities inherent in personal computing.

So there you have it: a bit of recent, yet ancient, history. Think of how far the personal computer has come, in under a decade. Yet maybe we ourselves have come even further.

What became of the 5100? Well, check the back of your PC's system unit, and you'll see the spirit of the 5100, quietly living on: The PC is the Model 5150.