Z3PLUS

The Z-System for CP/M-Plus

User's Manual

by

Jay Sage and Bridger Mitchell

Plu*Perfect Systems
410 23rd Street
Santa Monica, CA 90402

This page will contain the copyright and trademark notices and credits.

PREFACE

Automatic, universal, and dynamic, the two new Z-Systems — NZ-COM for CP/M-2.2 computers and Z3PLUS for CP/M-Plus computers — are the result of the extensive cooperative efforts of Joe Wright, Bridger Mitchell, and Jay Sage. All three authors worked together on many aspects of both products; each has been particularly responsible for one essential characteristic.

Joe Wright brought automatic operation. It was he who first conceived of and demonstrated with Z-COM what many deemed impossible – a version of Z-System that would install itself automatically on almost any CP/M-2.2 computer. Yet, even after Z-COM's success, it still appeared that Z-System could never run on a CP/M-Plus computer with its radically different command processor and banked memory operating system. Z3PLUS proves otherwise.

Bridger Mitchell made the systems universal. From his first exposure to Z-System, he decried the complex, laborious effort required to make each code module run with any particular implementation of the system. He developed a universal ZRL file format and loader that would allow a single file to adapt to any Z-System, no matter what its configuration in terms of module addresses and sizes.

Jay Sage added dynamics. He conceived an operating system that can change its size and character — right in the middle of a command line if necessary — to suit the needs of a particular task. No longer must the user live with a rigid compromise between operating system features and memory consumption. The trade-off can be reconsidered at any time.

These three authors played the primary role in the development of the new systems and are responsible for equally fundamental developments embedded in the new Z-System – the ZCPR version 3.4 command processor (Jay Sage), the Z3PLUS loader and command processor (Bridger Mitchell), and the NZ-COM loader (Joe Wright). But the full cast included many other players, in fact, a whole community of them. It is this shared participation in the ongoing development of Z-System in general that makes the effort so satisfying and rewarding for all of us.

Contents

Chapter 1

What is Z3PLUS?

Your CP/M-Plus computer is normally controlled by one master program — the CP/M-Plus operating system. From the moment you turn on your computer, the operating system is always there. It interprets commands, loads and executes programs, and manages the disk files and connections to your terminal, printer, and modem.

Z3PLUS is a major enhancement of the standard CP/M-Plus operating system. It gives you completely new software for processing commands that is functionally the same as the widely popular ZCPR3 system or Z-System for CP/M-2.2 computers. At the same time, Z3PLUS retains the lower-level software (the BIOS and BDOS) of CP/M-Plus, allowing you to continue to take advantage of the best features of your current system.

We call Z3PLUS the Z-System for CP/M-Plus. This term includes both the Z3PLUS command processing system and its many useful utility programs.

To use Z3PLUS you need a Z80-compatible computer running the CP/M-Plus (CP/M-3) operating system.

1.1 The Benefits of Z3PLUS

Here are some of the general benefits that you will derive from Z3PLUS.

- The new command processing system is much more convenient, powerful and flexible. You can readily customize it to your own style and habits.
- Hundreds of excellent programs written for `Z-System` computers will now run on your computer.
- You can obtain public-domain and user-group `Z-System` programs from `Z-System` bulletin board systems (called `Z-Nodes`) and use them.
- New, high-quality user-group and commercial software is continually being written for `Z-System` computers, programs that you will finally be able to run.
- `Z3PLUS` retains the advantages of `CP/M-Plus`: quick disk operations; convenient redirection of the screen, keyboard, and printer; and automatic execution of submit files.
- The `Z3PLUS` system is ready-to-run. No assembly or technical installation is required.
- You can remove `Z3PLUS` at any time and later restart it.

1.2 The New Command-Processing Features

The central function of the `Z3PLUS` system is to enhance the control and processing of the commands that direct the `CP/M` system. It provides

- convenient editing of mistyped commands
- recall of previous commands for re-use, correction, or modification
- conditional execution of a sequence of commands, where execution can depend on:
  - the success/failure of programs previously executed
  - the availability and characteristics of necessary files
  - the user's privileges and other system characteristics
  - run-time input from the user

- named directories to conveniently segregate files and optionally protect directories with passwords
- command aliases, single commands that cause an entire command script to execute automatically

1.3 Other Major Features

In addition to command processing, `Z3PLUS` provides

- a standardized method of using a terminal's full-screen capabilities, available to all applications wishing to use them
- a programming and user environment that fosters innovation and cooperation
- portability of programs among a wide variety of `Z80`-compatible computers

Taken together, these features mean that you can benefit from a very rich collection of programs, even if they were not written for your particular model of computer.

1.4 Using This Manual

There is a lot of material in this manual, and you will be glad to know that you do not have to read and absorb all of it before you can begin to make effective use of `Z3PLUS`!

Chapter 2 tells you the mechanics of setting up the system and getting it running. Chapter 3 shows you the basics of what you can do with the system. It demonstrates the use of a number of the new commands and includes examples. These two chapters are essential reading.

Chapter 4 contains a lot more information about what `Z-System` is and about some of its most powerful features. In order to really take advantage of `Z-System`, you will certainly want to read this chapter before very long.

Chapter 5 describes some of the advanced capabilites of `Z3PLUS`, particularly how to tailor the system to your personal needs and tastes.

You probably will not be ready to make effective use of the material in this chapter until you have acquired some experience with `Z-System`. We suggest, though, that you skim this chapter just to see what is there, so that you have some inkling as to what advanced capabilities are available to you.

Chapter 6 is primarily a technical reference. It summarizes and collects material that appears elsewhere in the manual, such as the syntax for commands and the names and functions of files supplied with `Z3PLUS`. Most of its topics are rather technical. However, there are some sections, like the one detailing the differences between the `Z3PLUS` system and `CP/M-Plus`, that all users will want to examine at some point.

Finally, Chapter 7, the Bibliography, contains references to other sources of useful and helpful information about `Z3PLUS` and the `Z-System`. Every user should read this chapter.

As you venture forth with `Z3PLUS`, we wish you an exciting and enjoyable experience. We hope that this manual and your experimentation with `Z3PLUS` will help you understand better how computers actually work, and make your own computer easier to use, more productive, and more fun as well!

Chapter 2

Starting Z3PLUS the First Time

2.1 Introduction

A key characteristic of the `Z-System` is its flexibility. Within a very broad range, you can customize it to your own needs and preferences. Of course, this flexibility means that there isn't a single "standard" setup. Getting started is very easy, but to take full advantage of the system you will have to take time to learn about its options and make some choices.

In fact, the `Z-System` is so flexible that veteran users are regularly discovering new capabilities that weren't envisioned in the original design. This dynamic, innovative quality is unusual for a microcomputer operating system, and it attracts both users and developers to continually make improvements.

To get you started, we've established a basic start-up configuration. It includes all available `Z3PLUS` features and will allow you to become familiar with them. A full-up `Z3PLUS` system like this one uses `6.75K` of your transient program area or `TPA` (the memory space available to application programs). Later (p. 44) you will learn how to create custom configurations so that you can make your own trade-offs between operating system features and memory consumption.

This manual is only an introduction to the `Z-System`. As you begin to

make use of your new Z3PLUS system, you will want to become familiar with the earlier, more extensive manuals and other written materials that cover the Z-System in greater detail. Several of these are listed in the Bibliography (Ch. 7).

2.2 Setting Up the Files

First, before you do anything else, make a complete, working copy of the Z3PLUS distribution disk and put the original disk away in a safe place. Then put the working copy into one of your auxilliary disk drives (not drive A) and log into it. The file RELEASE.NOT, if present, contains additional information that was not available at the time this manual was prepared. You should examine it briefly before proceeding.

Next, use the TCSELECT utility to create a terminal-capability file that matches your terminal or computer. There is a suitable file for almost every reasonably popular terminal. To do this from CP/M Plus, type

```
Z3PLUS!:TCSELECT<cr>
```

Note the exclamation mark as the command separator for CP/M Plus immediately followed by a colon to force loading the command from the currently logged drive/user. TCSELECT will display menus of available terminals. Find your terminal or computer and select it. Once you have confirmed the choice, TCSELECT will load this and you can now test it's performance. Once you are sure you want to keep that definition, rename it to "DEFAULT.Z3T". In this manual we will refer to your terminal choice as myterm.Z3T.

If for some reason your terminal or computer is not on the menu, you should first check the manual for your equipment to see if it uses the same screen codes as another, more popular terminal. If you are still unsuccessful, you should contact either the dealer from whom you purchased Z3PLUS or one of the Z-Helpers named in the file ZHELPERS.LST. They will probably be able to assist you in creating the Z3T file. In the meantime, you can continue to use Z3PLUS without defining a terminal, but you will not be able to use programs that require terminal information, such as SALIAS, ZSHOW, and ZFILER.

Next, using PIP or another file-copying program, copy the following files to user area 0 of drive A:

```
Z3PLUS.COM      ARUNZ.COM       LSH.COM
Z3PLUS.LBR      ALIAS.CMD       ZFILER.CMD
SALIAS.COM      LX.COM          ZSHOW.COM
SDZ.COM         IF.COM          myterm.Z3T
```

Finally, you should copy one of the two versions of ZFILER. This program makes use of video highlighting on the screen. Because reverse-video and dim-video are so different, two separate versions of ZFILER are provided: ZF-REV.COM for reverse video and ZF-DIM.COM for dim video. You should choose the appropriate one for your type of terminal. For example, if your terminal uses dim-video highlighting, you would use a command line like the following:

```
PIP A:ZFILER.COM=ZF-DIM.COM<cr>
```

2.3 Running Z3PLUS

Now you are ready to run Z3PLUS. If you haven't done that already, just log into drive A and type

```
Z3PLUS<cr>
```

Your new Z3PLUS system will install itself automatically. The first time you do this Z3PLUS will display

```
STARTZ3P?
```

You can ignore this message for now; it means that Z3PLUS is unable to find the file STARTZ3P.COM. We will take care of that on page 9.

The system is complete, except that it doesn't yet know what kind of terminal you have. We will now exhibit an additional capability of Z3PLUS by having it load the terminal descriptor. Type

```
Z3PLUS myterm.Z3T<cr>
```

That's all you have to do to get Z3PLUS running! You can now go on to the next chapter to learn more about the system.

2.4 Organizing Your Files

After you have learned about Z3PLUS and Z-System and are ready to use them on a regular basis, you will want to organize the files on your disks.

If you have a hard disk or large-capacity floppy disk you will want to keep most or all of your Z3PLUS and Z-System tools in directory A0: and make that your root directory (the first and preferably last directory on your search path – see p. 46). Some Z-System users prefer to use directory A15: for these files, but Z3PLUS, as distributed, is designed to use A0: as its system directory (see p. 53).

If you have smaller disks, put just the most-used files there and keep a separate Z-Tools disk at hand. Once Z3PLUS is loaded, the Z3PLUS.COM and Z3PLUS.LBR files are only needed again if you want to change the system configuration, so they could be relegated to a different "boot disk". You can configure Z3PLUS.COM to look for Z3PLUS.LBR in a different directory (see p. 56), such as the B drive. You could do this and then "boot" from B by entering the following command from the A drive under CP/M:

```
B:Z3PLUS<cr>
```

Chapter 3

Getting Acquainted with Z3PLUS

We suggest you actually carry out the following steps to become familiar with a few of the commonly used features of `Z3PLUS` and `Z-System`.

3.1 Creating an Alias

An alias is a single word (a command) that stands for a longer or compound command. To create an alias, run the `SALIAS` (`Screen ALIAS`) program. We'll create a startup alias, called `STARTZ3P`, that will automatically install your terminal file in the `Z3PLUS` system the next time you type `Z3PLUS`. (You've already seen that command name, `STARTZ3P`; it appeared on the screen followed by a question mark when you ran `Z3PLUS` the first time.)

Type

```
SALIAS STARTZ3P<cr>
```

to start the `SALIAS` program and to prepare the `STARTZ3P` alias. You'll notice right away that this utility, like many `Z-System` tools, makes effective use of the display features of your terminal (and that's why it was important to select a terminal-capabilities file at the outset).

The `SALIAS` program is an alias editor, a kind of wordprocessor for alias

scripts. You type in the text of the alias script and use control keys to edit your input. Most of the keys, including those for moving the cursor, are the ones used in WordStar. If you type Control-J, you will see a list of all of the command keys.

For this alias, the only command line you need to enter is

```
Z3PLUS myterm.Z3T /Q<cr>
```

We added the "`/Q`" quiet option to suppress the extensive information that `Z3PLUS` can display when it loads something. Now, type Control-K followed by Control-X to exit and save the new file.

Try executing this alias now by typing

```
STARTZ3P<cr>
```

You'll see from the display that it indeed runs the `Z3PLUS` program and loads the `Z3T` file (again). So far this just saves us a little bit of typing. However, we selected this exact name for the alias because `Z3PLUS` itself will execute this alias automatically the next time you load it. Let's test things so far. Type

```
Z3PLUS OFF<cr>
```

Yes, the `Z3PLUS` system is gone, and the standard `CP/M-Plus` command processor is once again active. Now type

```
Z3PLUS<cr>
```

to restart the `Z3PLUS` system. This time it will automatically load your `Z3T` terminal file using the new `STARTZ3P` alias.

The startup alias you just created is quite simple, but nonetheless useful. An alias, however, can do much more. It can be a whole sequence of commands, with symbolic parameters for filenames, directories, and command line parameters. For a more elaborate example, see the alias created on page 15. For more information, consult the references in the Bibliography. And keep `SALIAS` in mind. It will come in handy for automating a wide variety of tasks on your computer.

3.2 The Available Commands

Now type

H<*cr*>

This will display all of the in-memory commands in the current Z3PLUS system. The screen will show a display like that in Table 3.1.

```
FCP
   IF        AND       OR        ELSE      FI
   IFQ       XIF       ZIF

CPR
   GET       GO        JUMP

RCP
   CLS       ECHO      ERA       H         NOTE
   P         POKE      PORT      R         REG
   SP        TYPE      WHL       WHLQ
```

Table 3.1: Resident commands displayed by the H command.

The commands that are displayed are the ones in memory at the time the H command is given. They will vary according to which FCP (flow command package), RCP (resident command package), and CPR (command processor) have been loaded. As you see, the H command (short for "HELP") is in the RCP package.

Later you will see that you can create versions of Z3PLUS that use less memory by not including an RCP. Also, some RCP packages do not support an H command. In these cases, the ZSHOW command, described on page 20, can be used to show this (and much more) information.

3.2.1 RCP Commands

Let's look at the RCP (resident command package) commands first. We've just used the H command. Try another one by typing

ECHO This is a test.<*cr*>

Before we go on, we should point out that a semicolon (";") is used to separate multiple commands on a single line. Try

```
ECHO Test 1;ECHO Test 2;ECHO Test3<cr>
```

`ECHO` is a simple but important command that is often employed to tell the user what is happening while sequences of commands are being processed by a script. We will see an example of this on page 15. Because command lines are always converted to upper case, your messages appeared in capital letters. The `ECHO` command will interpret the character sequence "%>" to mean "change to lower case" and the sequence "%<" to mean "change to upper case."

Now try the `SP` command, which tells you how much space remains on a disk:

```
SP<cr>
SP B:<cr>
```

The `P` command peeks at memory. Try

```
P 100<cr>
```

This is not too informative if you don't read computer code, but it is very useful and handy if you do. Now try the `POKE` command:

```
POKE 100 "This is a poke test<cr>
```

Be sure you typed the quotation mark before the text; it tells `POKE` that you are giving it text and not hexadecimal numbers. Now peek at the result:

```
P 100<cr>
```

Obviously, you want to reserve `POKE`s for special purposes, when you know you won't damage something (see p. 18). You can peek with abandon, however.

`ERA` should be familiar; it erases files. `TYPE` displays a file on the screen. `CLS` clears the screen (handy in scripts). `NOTE` is a do-nothing command used in scripts to include comments. `PORT` is a combination peek/poke for the computer's input/output ports; it reads a data byte from or

writes a data byte to a specified I/O port. As with the peek/poke commands for memory, these should be used with great caution.

The R command resets the disk system; it is like a Control-C, but it can be included in a multiple command sequence. REG displays or modifies the Z-System software registers, which are used to communicate values between programs. The WHL and WHLQ commands were used with the Z-System security byte called the "wheel byte". You will find that many commands, especially risky ones like ERA, will not work when the wheel byte is off. WHLQ (WHeeL Query) is obsolete by now. Instead, you may use WHL, which, when followed by the correct password, turns on the wheel byte; otherwise it turns it off. The default password is SYSTEM.[1]

3.2.2 FCP Commands

The Z-System can process entire sets of commands in an intelligent, automated way using the commands in the flow command package (FCP). Flow control takes some effort to understand, but it is so powerful and useful that we think you'll find it worth the effort to learn about it. In a nutshell, the purpose of flow control processing is to enable your commands to perform multiple levels of IF-ELSE-ENDIF testing (just as in high-level programming languages) in order to control which of a number of alternative commands are executed.

The main concept you need to grasp is that of "flow states". When Z-System first starts to run, no flow state is active (we call this the "null" flow state). Under control of the commands in the FCP module, up to 8 levels of flow state can become active. Each level can be either true or false (the null flow state acts like a "true" flow state). When the current flow state is true, commands execute in the normal fashion; when the current flow state is false, the command processor ignores all commands except those in the FCP. You will see how this works with some examples in a moment.

The IF command is issued with some type of test expression, which is evaluated as either "true" or "false". For either value, a new level of flow state is activated. If the previous flow state was false, then the new flow state is always put into a false state, no matter what the result of the conditional test was. On the other hand, if the previous flow state

[1] If you need to change it, the file DEFRCP.ZRL in Z3PLUS.LBR can be patched. With your favorite patch utility, search for the string "SYSTEM" and change it as you like. If you need help doing this, consult your dealer or a Z-Helper.

was true, then the new flow state will take on the value resulting from the test.

Here are some examples of conditional tests that can be performed by the `IF` command:

`IF ERROR`	tests whether a previous program set the program error flag
`IF REG 3 > 1`	tests value of a user register
`IF INPUT RUN WS?`	displays the prompt "`RUN WS?`" and waits for user response
`IF ~EXIST DIR:FN.FT`	tests for (non)existence of a file

Enter the command line

`IF //`<*cr*>

to get a condensed listing of the test conditions that can be used. As you can see, the number is quite large. Later you can experiment with them. Only the first two letters are actually used to determine the test condition (thus `EX` is the same as `EXIST`), and a tilde ("~") prefix reverses the sense of the test. Most of the tests are performed by the program `IF.COM`, which is loaded and run automatically by the `FCP`. If you want to see the difference between the built-in IF and the transient one, you are welcome to rename `IF.COM` to `IFCOM.COM`. After you have the experience, don't forget to rename it back to IF.COM!

The `IFQ` (`IF Query`) command is included in the `FCP` to help you visualize the flow state and learn how to use flow control. We will use it now with some examples.

First try out the `IFQ` command. Unless you have already been experimenting on your own (in which case you should run the command "`ZIF`<*cr*>" to return the system to the null flow state), you should see the message "`IF None`", telling you that there is no active flow state (i.e., the "null" flow state). Now enter the command:

`IF T`<*cr*>

This test, like the `IFQ` command, is included largely as a learning aid. As you can probably guess, it returns a value of true. Now run `IFQ` again. The display will read "`IF T`", indicating that one flow state level is active and that it is true. Try entering a command, such as "`ECHO TESTING`<*cr*>'' and note that it executes normally.

Now enter the command line:

`IF F;IFQ`<*cr*>

The display will show "`IF FT`". This tells us that two levels of flow state are active, with the current one false and the one under it true. Now try to execute the "`ECHO TESTING`<*cr*>" command. It is ignored.

The `FI` command, which is `IF` spelled backwards, terminates the currently active flow state and returns the system to the previous level (or to the null state). Enter the command line

`FI;IFQ`<*cr*>

several times and watch how the message changes. Once you are back in the null flow state, enter the command

`IF T;IF F;IFQ`<*cr*>

to get us back to where we were (with the current flow state false and one underlying flow state that is true).

Now we will learn about the `ELSE` command, which reverses the current flow state providing the underlying flow state is true. Enter

`ELSE;IFQ`<*cr*>

and note that the display has changed from "`IF FT`" to "`IF TT`". Commands will now run again. Enter it a few more times, and note how it toggles the flow state.

At this point you know enough about flow control and the `FCP` commands to experiment on your own. When you are finished, enter "`ZIF`<*cr*>" to zero out the flow state.

Before leaving this subject, we would like to take you through one last example, one that will give you a hint as to how flow control can be combined with aliases to make very powerful commands.

Use `SALIAS` to create an alias called `TEST` by entering the command

`SALIAS TEST`<*cr*>

When the input screen appears, enter the following sequence of commands:

```
IF EX $1
ECHO FILE $1 EXISTS
IF EQ $1 *.COM
ECHO COM %>FILE: %<YES
ELSE
ECHO COM %>FILE: %<NO
FI
ELSE
ECHO FILE $1 NOT FOUND
FI
```

Note that we are using the special character sequences that ECHO recognizes as signals to change between upper and lower case (see p. 12). When you are done typing in the commands, press Control-K Control-I. Note how SALIAS conveniently indents the commands to show the nesting of flow states. Now save the alias by entering Control-K Control-X.

The expression "$1" in the alias stands for the first item given on the command after TEST when the alias is used. Try entering the following commands and see what happens in each case:

```
TEST Z3PLUS.COM<cr>
TEST Z3PLUS.LBR<cr>
TEST NOFILE.JNK<cr>
```

3.2.3 CPR Commands

The command processor itself has only three commands. Most of the code in the CPR is needed to perform sophisticated processing operations on *your* commands rather than to execute commands of its own. Most of the resident (in-memory) commands are provided by the RCP, which offers you the advantage of being able to change the commands by loading different RCP modules depending on your current needs.

The commands in the CPR are ones that make use of code already there for other purposes. GET and GO/JUMP are basically the two halves of the normal load/execute process that is performed whenever you specify a command that refers to a COM (or possibly SUB or PRL) file. All three commands can be dangerous; if used incorrectly they can result in a system crash that will require rebooting. Used correctly, they can be very powerful and handy.

The GET command performs only the loading process. It offers more flexibility than the normal command loader. First, *you* specify the memory address to which the file is loaded, and, secondly, any type of file (not just a COM, SUB, or PRL file) can be loaded.[2] There is nothing to stop you from loading a program to a wrong address or from loading a file that is not a program.

Try entering the command line

```
GET 200 Z3PLUS.LBR;P 200<cr>
```

The library will be loaded into memory starting at 200H, and the peek command will show you the beginning of the file. If you have ever wondered what a library file really looks like, now you can see. GET and P together can make a handy learning tool.

Just as GET performs only the loading of a file, GO and JUMP perform only the execution of what is already in memory. GO automatically executes whatever is at address 100H. JUMP is similar but lets you specify the execution address (GO is equivalent to JUMP 100). For normal COM-file programs, the combination of GET and GO performs an ordinary execution. Thus the command sequence

```
GET 100 SDZ.COM;GO *.LBR<cr>
```

does exactly[3] the same thing as

```
SDZ *.LBR<cr>
```

Why, then, would one ever want to use these separate commands? We will give two examples. First, the GO command can be used to execute a program repeatedly without having to spend the time loading the file each time from disk. Try

```
SDZ *.COM<cr>
```

and then

```
GO *.LBR<cr>
```

[2] All files are loaded as binary images; PRL files will not be relocated.
[3] Well, almost exactly.

Caution: some programs are not designed to be reexecuted and could cause trouble if used this way. Most Z-System tools can be reexecuted, and those that cannot often exit gracefully at least. Other programs may be less forgiving, so you should experiment cautiously.

A second example of the use of GET and GO is the technique called "GET, POKE, and GO" or simply POKE&GO. The idea here is to load a file using the GET command, modify it using the POKE command, and then run it using the GO (or JUMP) command. For example, if your wordprocessor (let's say it is called WP.COM) keeps its right margin value at address 83BH and normally uses a value of 76, you could make an alias named WP60 with the following alias script to give you a version with a right margin value of 60:

```
GET 100 WP.COM;POKE 83B 3C;GO $*<cr>
```

Note that the poked value is given in hexadecimal form (3CH=60) and that the expression "$*" in an alias script is replaced by whatever the user put on the command line after the name of the command. Thus the command

```
WP60 NEWFILE.DOC<cr>
```

would run WP with the file NEWFILE.DOC but with a right margin of 60. To do this without Z-System, you would have to have a second version of the entire wordprocessor. An SALIAS alias is only 1K long; an ARUNZ alias (p. 30) for this function takes only about 40 bytes of disk space.

3.2.4 Transient Commands

"Transient command" is the name Digital Research (publisher of CP/M) gave to commands that do not reside in memory but are loaded from disk. Under CP/M-Plus, these commands include files of type COM (as in CP/M-2.2), plus files of type SUB or PRL (depending on how the CP/M-Plus system is configured). Files of type SUB and PRL will run under Z3PLUS in much the same way they did under CP/M.

In addition to the normal COM files supported by CP/M, there are several special types of COM file supported by Z-System. These files are designated as type-1, type-3, and type-4.[4]

[4] Yes, there are type-2 files also, but they are rarely used today.

Type-1 files are similar to standard CP/M COM files except that they have a special header at the beginning into which the command processor inserts the address of what is called the Z-System environment descriptor. This is an area in memory that contains a complete description of the Z-System configuration. With that information, a program can make use of the special Z-System facilities.

Type-3 programs are similar to type-1 programs except that, unlike standard CP/M programs, they do not necessarily get loaded to and run at the standard address of 100H. Their execution address is included in the special header, and the command processor automatically loads them to the address specified there. For example, the program ARUNZ.COM included in the Z3PLUS package is a type-3 program that runs at 8000H.

Type-4 programs are special Z-System programs derived from PRL (so-called Page ReLocatable) files. They contain relocation information to allow them to run at any address. The command processor automatically calculates the highest address in memory at which they can run with the existing configuration and then loads and runs them at that address.

The type-3 and type-4 programs were invented for transient programs that function as resident parts of the system (e.g., extended command processors, shells, and error handlers) or as transient versions of common resident commands (e.g., ERA.COM or REN.COM). Since the Z-System on its own often invokes these commands automatically, you cannot always predict when one of them will run, and it is handy if they do not interfere with the lower parts of memory where user programs run.

The special Z-System programs usually cannot run properly under CP/M. Because they were designed to run on computers with *permanently* installed Z-Systems, most of them do not include code to prevent attempts to run them under ordinary CP/M. Most of the ones supplied with Z3PLUS, however, will terminate gracefully from such an attempt. Some of them will display a message; others will simply cause a warmboot.

3.3 More About the System Configuration

We mentioned earlier that under some circumstances you would not be able to use the H command to see what resident commands are available. The Z-System utility program ZSHOW[5] can always be used to display this and a great deal of other information about your system.

The best way to learn about the configuration of your Z3PLUS system is to run ZSHOW and experiment with its menu selections. Some of the things you can do with ZSHOW are:

- display the memory map of the system
- show the commands supported by the CPR, FCP, and RCP, with an indication of which commands require the wheel byte to be on
- show the special features implemented in the CPR
- show the command search path and directory names
- display the information in the shell stack and message buffer

3.4 Incompatible Programs

Most programs written for Z-System will work perfectly under Z3PLUS. There are a few, however, that are incompatible with Z3PLUS and should not be run. Those that we know about are:

CLEANDIR	directory cleanup utility
DU3	disk utility, DU35 now O.K.
MOVE	change a file's user number
MAKE or MAKEUSER	change a file's user number
ZEX before 5.0	in-memory submit processor

Each of these programs, except ZEX, uses the direct services of the CP/M-2.2 BIOS for file or disk operations, something that requires different routines under CP/M-Plus. Such programs will most likely "crash" your system and require a cold boot, so avoid them. This is especially important for files that operate on CP/M 2.2 -type directory

[5] The original name of the program was SHOW, but it has been renamed in the Z3PLUS package to avoid conflict with the CP/M-Plus utility with that name.

structures, like SORTDIR or others. Since they cannot deal with the CP/M Plus date stamping fields, your directory is very likely to get clobbered by such files, up to the point where there is no more cold-boot. Older `ZEX` (before 5.0) attempts to perform I/O redirection in a way that is not compatible with `CP/M-Plus`.

Occasionally, you may encounter a program that reports a bizarre amount of space left on a disk. This is uninformative but harmless and is due to another difference between `CP/M-2.2` and `CP/M-Plus`. Eventually, most `Z-System` tools should be upgraded to report disk space correctly on both operating systems.

There are also a series of CP/M Plus datestamping patches, that allow to really use the file creation and updating stamps also for copying. Look out for those and eventually replace the files you have received by those.

Chapter 4

Learning More About Z-System

The `Z-System` divides into

- resident (in-memory) components
- automatic tools and shells — the special programs that the command processor runs for you
- other tools — programs that are controlled by or make use of the resident components

The components included in the `Z3PLUS` package and described below constitute a fairly basic set, and yet they are already rich in features. The following descriptions are included here for your reference, but you certainly don't need to absorb all of it at the first reading.

4.1 Resident Z3PLUS Components

- The `Z3PLUS` command processor: It interprets commands and loads programs.
- An `RCP` (Resident Command Package): It includes an extended set of commands that can be quickly changed (see p. 11).

- An FCP (Flow Command Package): It tests logical conditions and manages the flow state of the system so that commands can be conditionally executed (see p. 13).
- An NDR (Named Directory Register): It contains a table that associates directories (drive/user-number pairs) with names and optional passwords.
- A PATH: It indicates the sequence of directories to be searched to find a program.
- A Z3T terminal descriptor: It contains the control sequences needed by your terminal for positioning the cursor, clearing the screen, etc.
- Other components of the Z-System environment: These include the command-line buffer, a message buffer, a shell stack, the wheel byte, the external file control block, and a command processor stack.

4.2 Automatic Commands

Several Z-System tools are automatically caused to execute under specific conditions.

4.2.1 Extended Command Processor

When the command processor cannot process a command as either a memory-resident command or a disk-resident file, then it will pass the command to another program called the Extended Command Processor (ECP). Z3PLUS always tries to load as the ECP a program with the name CMDRUN.COM in the root directory (the final directory in your search path). The distributed Z3PLUS system includes two programs that are often used as ECPs:[1]

- ARUNZ (Alias-RUN-for-Z-system), which performs a function very similar to that of the aliases created by SALIAS, except that
  - a large number of aliases can be defined in a single text file called ALIAS.CMD, and

[1] There is even a way to have both of them function together as the ECP.

  - much more extensive parameter processing is supported.

  ARUNZ can function as a very powerful command generator and translator.

- LX (Library eXecutive), which gets commands out of a library called COMMAND.LBR. Putting COM files into COMMAND.LBR saves both directory space and file space on the disk. LX is especially useful on computers with low capacity diskettes or diskettes whose format allows too few file names in a directory.

To make one of these programs function as your extended command processor, copy it to the filename CMDRUN.COM.[2] For example, to use ARUNZ as the ECP, type the command

```
PIP CMDRUN.COM=ARUNZ.COM<cr>
```

You should do that now, since in a little while (p. 30) we will be using ARUNZ for some examples. Later, if you want to try using LX instead, type the command:

```
PIP CMDRUN.COM=LX.COM<cr>
```

4.2.2 Error Handler

When the command processor cannot process a command because the extended command processor cannot be found or because the extended command processor also cannot process the command, then the CPR will invoke an error handler. Advanced Z-System error handlers can display information about what is wrong with the command and can allow you to edit the command line to correct any mistakes.

Any error handling command line can be loaded with the ERRSET utility (not supplied). However, most error handlers will install themselves automatically if you invoke them manually as a command (they can tell whether they have been invoked by the user or by the CPR). The Z3PLUS package includes the LSH error handler, which is described in the next section.

[2] It must be in the root directory (last element on the path). Unless you have changed it, this will be A0, but you can run the PATH command to see what the root is.

4.2.3 Command Shells

Shells are tools that are automatically executed whenever the command processor has completed all the commands it was given — including batch commands from `SUBMIT` — and is ready for new command input. If no shell is loaded, then the normal prompt will appear on the screen, and the user will be asked to enter the next command. If a shell is active, then it will run instead. Most shells generate commands *for* the user and pass them on to the command processor. We have included several shells with the `Z3PLUS` system.

The shell command line (sometimes along with other information used by the shell) is kept in a memory buffer called the "shell stack". The shell stack in the `Z3PLUS` system can hold up to four shell command lines. The command that was placed on the stack most recently is the active shell command. When that shell is terminated (usually by a special command response to the shell's input request), its command line is removed from the shell stack (the common term is "popped"), bringing the next most recently invoked shell to the top of the stack and making it active. This mechanism allows shells to be "nested".

The ZERRLSH and LSH Error Handler and Shell

`LSH` (`Log Shell`) is a perfect example of the shell concept. It is also particularly handy. When `LSH` becomes active, it puts up a prompt like the one normally presented by the `CPR` but with the highlighted CP/M Plus time before the drive/user to remind you that `LSH` is running. There are three important differences, however. `ZERRLSH.COM` is the perfect counterpart to `LSH.COM` and automatically updates the history file when an error generating command line is used. Customization is easily achieved by means of the `ZCNFG` utility.

1. When you enter a command, you have at your disposal a very powerful editor, what amounts to a wordprocessor for the command line. You can move forward and backward through the command line and can insert and delete characters.

2. When you issue your command by pressing the carriage return key, the command line is saved in a history file before it is passed to the command processor.

3. Commands entered in the past can be retrieved from the history file, edited as you wish, and then executed again.

After you see how convenient these features are, you may want to add LSH to the end of the list of commands in your STARTZ3P startup alias.

The version of LSH included with Z3PLUS is a type-3 program (see p. 19) that executes at an address of 8000H (32K). As a result, unless the user's last program was longer than 32K, LSH will not interfere with rerunning it using the GO command.

We recommend that you now try some examples. Type in the following commands one at a time:

```
ECHO THIS IS COMMAND 1<cr>
DIR *.COM<cr>
ECHO THIS IS COMMAND 3<cr>
LDIR Z3PLUS<cr>
```

When the next prompt appears, type Control-E. The previous command will appear. Try entering Control-E several more times. Then try Control-X; it will move you forward through the history. Continue moving through the history until the command "ECHO THIS IS COMMAND 3" is at the prompt.

Now we will edit the command. Most of the editing command keys are those from WordStar. Use Control-S and Control-A to move left one character or one word. Use Control-D and Control-F to move right. When you get to the "3", enter Control-G to delete (Gobble) it. Then enter "5" and a carriage return.

Now we will try a recall search. At the prompt, type in only the letters "ECHO" (no carriage return). Then press Control-O. Note that LSH has located for you the most recent command line that started with "ECHO". Press Control-O again. LSH will find the next previous occurrence. This feature is remarkably handy!

LSH will continue to record all commands (except very short ones) until you remove it (by typing Control-Q Control-_ [underscore]). The history is stored in the file LSH.CMD. This file will continue to grow, and from time to time you might want to erase it (but not while LSH is running) and start over. There is also a version of the program that can keep only a fixed number of recent commands.

You can generate a file that lists all of the control keys used by LSH by entering the command

```
HELPLSH<cr>
```

or just press ESC-J to get a full page of help.

The ZFILER Shell

ZFILER illustrates how a shell can change the user interface completely. Instead of a command line, it provides a "point-and-shoot" environment. To see ZFILER in action now, enter the command

ZFILER<*cr*>

You do not have to remove LSH before running ZFILER. Shells can be "nested" under Z-System. If LSH was running when ZFILER was invoked, then LSH will become inactive while ZFILER is running but will return automatically upon exit from ZFILER.

You will see a full-screen display of the names of the files in the current directory and a pointer pointing to the first file. Use the WordStar-diamond control keys to move the pointer around (control keys E, X, S, and D to move up, down, left, and right, respectively). Position the pointer to a text file, such as LSH.CMD, ALIAS.CMD, or ZFILER.CMD. Press the "V" key to "View" the file. ZFILER has many such built-in functions, and you can see a listing of them by pressing the slash ("/") key for help. Press slash again to return to the file display.

Try tagging some files using the "T" key. Now experiment with a "Group" operation by pressing the "G" key. As you can see, a number of functions can be performed not only on single files but on all the tagged files. Try selecting "Copy" by pressing "C". When prompted for the destination directory, enter

A1<*cr*>

Note that the colon after directories is optional inside ZFILER.

Let's verify that the files were copied. Log into the A1: directory using the "L" or "Log" key. Answer the prompt with

A1<*cr*>

Delete these duplicate files using the "D" key. Now let's return to directory A0:, but in a slightly different way. Directory A0: has the name COMMANDS. Press the "L" key and answer the prompt with

```
COMMANDS<cr>
```

As you see, named directories can be used just as well as drive-user designations. This is true throughout the Z-System.[3]

The built-in functions are not the only ones ZFILER can perform. A set of "macro" commands can be defined, along with a user information screen, in the file ZFILER.CMD. We have supplied a very simple one to illustrate what can be done. Move the file pointer to the file Z3PLUS.LBR. Now press the escape key. You will be prompted for a macro. Later, when you know the keys associated with the macro functions, you can enter the appropriate key immediately. Alternatively, you can press escape again (do that now), and the information screen will appear. You will see that the "L" macro will give you a directory of a library file. Press "L" now and watch what happens.

ZFILER generated a command line for you and passed it on to the command processor. After that command line had completed execution, the ZFILER shell was reinvoked automatically. In this case, it knew that it had just finished executing a command that put some information on the screen, so it waited for you to press any key before it restored the display of file names.

It is even possible to run macro commands on groups of tagged files. Tag a few files and then press "G" to start a group operaton. When asked for the operation, press the escape key to invoke macro processing. As before, you will be prompted for the macro key. You can again enter the key immediately or press escape to see the information screen. Press "E". ZFILER will now automatically generate a file called ZFILER.SUB in directory A0: and then initiate a SUBMIT batch operation. The "E" macro, as you will see, is a harmless command line using ECHO to show how ZFILER can parse the name of the designated file (pointed to or tagged).

You can now exit from ZFILER by pressing the "X" or "eXit" key. That will terminate the shell. If LSH was running before you invoked ZFILER, then LSH will return. Otherwise, you will again see the prompt from the command processor itself.

[3]That is, for the Z3PLUS command processor and for the Z-System-specific utility programs.

Other Shells

There are quite a few other shells not provided with the standard `Z3PLUS` package that offer a wide variety of functions. Some are in the same family as `ZFILER`. In contrast to `ZFILER`, which shows the files on the screen and displays the available macro commands only on request, the `MENU` shell shows only the command options. It supports multiple levels of nested command menus. The `VMENU` and `FMANAGER` shells are half way between `MENU` and `ZFILER`. They show the files in the upper half of the screen and the command options in lower half.

A different kind of shell is the patching shell `ZPATCH`. It enables you to edit the binary image of a file. For example, you might be customizing the configuration options of a program. From within `ZPATCH` you can execute the modified program to see the results of your handiwork and then return automatically to `ZPATCH`, positioned to the exact byte you were at when you left.

Another set of shells (`SH`, `GETVAR`, `FOR-NEXT`) allow one to create and use "shell variables", much like the "environment variables" in `MS-DOS`.

4.3 Other Z-System Tools

There are many, many other `Z-System` tools, far too many to begin to describe them all here. In fact there is probably no `Z-System` user, no matter how expert, who knows them all! To get you started, we will describe a few particularly important ones. We did not have you copy all of them from the `Z3PLUS` working disk (p. 6) into your `A0:` directory. You can either copy them now or run them from the work disk.

4.3.1 ARUNZ

A `Z-System` alias or alias script is a sequence of one or more commands that can be invoked by a single name. As we saw in Section 3.1 (p. 9), stand-alone aliases are short `COM` files that can be created by `SALIAS`, the screen oriented alias editor.

`ARUNZ` provides an alternative way to define and use aliases. Instead of making each alias its own file, `ARUNZ` allows one to define numerous aliases in a single file called `ALIAS.CMD`. The `ARUNZ` program extracts a designated alias script from the `ALIAS.CMD` file, expands any symbolic

parameters in the script, and passes the resulting command line to the command processor.

The `ALIAS.CMD` file is an ordinary text file that is created with any text editor or wordprocessor (in non-document mode). Since all the scripts are combined in a single file, they require very little disk space. As a result, you can easily have dozens or even hundreds of them, and they can greatly enhance and ease your computing life. Also, this type of Alias is most likely loaded faster than separate COM-files.

`ARUNZ` aliases are invoked by a command of the form:

```
ARUNZ ALIASNAME COMMAND-ARGUMENTS<cr>
```

`ALIASNAME` specifies which of the scripts in `ALIAS.CMD` to use, and `COMMAND-ARGUMENTS` supplies other information needed by the scripts, such as file names. The real power of `ARUNZ` comes when it is renamed to `CMDRUN.COM` and serves as the extended command processor (see p. 24). Then the command can be issued simply as:

```
ALIASNAME COMMAND-ARGUMENTS<cr>
```

An example might make this clearer. One line in the `ALIAS.CMD` file reads:

```
D=SD     sdz $td1$tu1:$tn1*.$tt1* $-1
```

This may look rather forbidding with all those dollar signs, but, when we take it apart piece by piece, it won't be so bad. First of all, the name of the alias is either "`D`" or "`SD`". `ARUNZ` allows multiple names to be assigned to a single script. The program run by this alias is `SDZ`, the super-directory program.

The interesting part of this script is the way the command arguments are handled. To illustrate what happens, let's assume that we entered the following command:

```
D S.C<cr>
```

The parameter expression "`$td1`" means the drive specified in the first token. In our example, the first token is "`S.C`". Since we did not specify a drive in that token, the current drive, A, is assumed. Similarly, "`$tu1`" means the user number specified in the first token. Again,

none was given, so the current user number, 0, is assumed. Next we have a colon. So far the command line reads: "`SDZ A0:`".

The next parameter, "`$tn1`" indicates the file name part of the first token. In our example, this is "`S`". The next characters in the script are an asterisk and then a period. Then we have the parameter "`$tt1`", which stands for the file type in the first token, or "`C`". Then we have another asterisk in the script. So far the command line reads: "`SDZ A0:S*.C*`".

The final part of the script is the parameter expression "`$-1`". This means the entire command argument less the first token ("`$-2`" would omit the first two tokens). In the example, there are no other tokens, but one might have used an option to `SDZ` such as "`/C`" to get the file size as a record count instead of in kilobytes. Thus the command entered as

`D S.C`<*cr*>

has become, courtesy of `ARUNZ`,

`SDZ A0:S*.C*` <*cr*>

Try entering the alias command and see what happens. The script gives us a very convenient way of automatically making the file specification to the `SDZ` program into a wildcard specification, saving us the nuisance of having to type all the asterisks. In the example, we get all files whose name starts with "`S`" and whose type starts with "`C`", including `SALIAS.COM`, `STARTZ3P.COM`, and `SDZ.COM`.

To learn more about how to write and use `ARUNZ` aliases, consult the references in the bibliography. But note that the version of `ARUNZ` included with `Z3PLUS` is more recent than the one described in the reference there. Many new parameters have been added, and a few old parameters have been changed. So read the update documentation provided with the `Z3PLUS` package.

4.3.2 HELP

Most of the Z-tools will provide a terse reminder of their command line syntax if you enter them with just a double slash, as in

`Z3PLUS //`<*cr*>

Much more complete on-line information can be obtained using the Z-System HELP utility. It provides an organized method (tree structured) for searching special text files with a file type of HLP and displaying the information in them. It is similar to the CP/M-Plus HELP facility but much more powerful. The program was originally called HELP, but to avoid confusion with the CP/M program of that name, we have renamed it to ZHELP for the Z3PLUS system. You run ZHELP as follows:

```
ZHELP<cr>
ZHELP filename<cr>
```

The first form uses a file with the name HELP.HLP; the second form used one with the name "filename". For example, if you want help with SALIAS and there is a help file for it, enter the command

```
ZHELP SALIAS<cr>
```

Help files for a number of other Z-System programs can be found on the Z-Nodes. You can also write your own HLP files.

4.3.3 Library Tools

Z-System library tools allow you to put an entire set of files into a single library file and then access the individual members of the library as needed. This is effective for keeping short and related files organized and for saving directory and file space on a disk. Programs like Z3PLUS and JETLDR are able to access multiple members of a library more rapidly than individual files could be read. Libraries are also widely used on remote access CP/M systems (RASs or RCPMs). The essential Z3PLUS system files, for example, are all kept in Z3PLUS.LBR.

The two multi-purpose library tools are NULU (new library utility) and VLU (visual library utility). These two programs can create a library, insert and extract files, type files to the screen, and a host of other functions. NULU is a generic CP/M program readily available from RASs; VLU is a Z-System-specific tool and works somewhat like the ZFILER shell. We have included it with the Z3PLUS package. Note, that despite later version numbers, the only version to work fairly o.k. with CP/M Plus is the revised version 1.02 as supplied with this package!

Library tools that perform more limited functions are:

LDIR	display the directory of a library
LGET	get files out of a library
LHC	display a library HELP file)
LPUT	put files into a library
LT	display a member (library type)
LX	extract and execute a member

4.3.4 File Compression

Text compression and decompression tools allow files to be "crunched" to a fraction of their original size for more compact storage and faster transmission by modem. Crunched files have a "Z" as the second letter of their file type (or a file type of ZZZ if the original file had no file type at all). CRUNCH and UNCRUNCH do the job. We have provided them with the Z3PLUS package.

LT.COM is quite versatile. It was listed with the library tools because it can display text in library member file. LT.COM will display individual files, and in both cases the file can be crunched as well as normal text. It even figures out automatically whether uncrunching is needed. [4] To type a library member to the screen, use the following command form:

```
LT LBRNAME FILENAME.TYP<cr>
```

To display an individual file, use the form:

```
LT FILENAME.TYP<cr>
```

4.3.5 Named Directory Tools

The ability of Z-System to associate names with directories can make life easier, especially when the computer has a hard disk. EDITNDR[5] is a tool for editing the assignment of directory names. It can be used in either interactive mode or from the command line. Let's use it interactively first. Enter the command

```
EDITNDR<cr>
```

[4] LT31 will also display Lempel-Zev crunched files (.?Y?)

[5] This was originally released as EDITND. We have changed its name to make it consistent with SAVENDR.

You will get a prompt inviting you to enter "?" for help. Why not accept the invitation! Now try entering a carriage return. `EDITNDR` will now show you the names that are currently assigned. Now enter the line

`A1:TEXT`<*cr*>

This will assign the name `TEXT` to user area 1 on drive A. Press <*cr*> to see the new listing. You may not notice the assignment for `A1:` because it is not with the other assignments for drive A. Enter the command "`S`" to sort the listing and then another <*cr*>. Now the names are in order. If you look at the built-in help screen, you will see that `EDITNDR` is very flexible in the syntax it will accept.

The `ALIAS.CMD` file contains an alias to facilitate assigning names to directories. You just enter the command

`NAME DU:DIRNAME`<*cr*>

where `DU` is the drive/user to be given the name `DIRNAME`. If "`DU:`" is omitted, the current directory will be given the name. Try entering

`NAME SYS`<*cr*>

and see how the command prompt changes. You can look at the contents of `ALIAS.CMD` to see how this alias works.

The named directory assignments made by `EDITNDR` are temporary. You can use the `SAVENDR` utility to write the assignments out to a file. The file can be loaded by the `STARTZ3P` alias to reinstall your names. For example, the command

`SAVENDR SYS`<*cr*>

will create a file called `SYS.NDR` in the current directory. This file can be loaded by `Z3PLUS` using the command

`Z3PLUS SYS.NDR`<*cr*>

`Z3PLUS` can load many files at once, so you can combine this with the loading of the `Z3T` file as follows:

```
Z3PLUS myterm.Z3T SYS.NDR<cr>
```

If you put those two files inside `Z3PLUS.LBR` using `LPUT`, you can then load them even faster using

```
Z3PLUS Z3PLUS.LBR myterm.Z3T SYS.NDR<cr>
```

For the ultimate, you can rename the two files to `DEFAULT.Z3T` and `DEFAULT.NDR` before putting them into `Z3PLUS.LBR`. Then `Z3PLUS` will load them automatically when it starts, and you will not have to include any command in the `STARTZ3P` alias to do it.

4.3.6 Other Tools

A number of other utility programs may be included with your `Z3PLUS` distribution diskette. You should experiment with them on your own. Enter the command with "//" after it to see any built-in help information. If there is a help (`HLP`) file with the same name as the command, you can use `ZHELP` to learn more about the command.

4.4 Command Hierarchy

The `Z3PLUS` command processor is highly sophisticated in the way it processes commands. Two steps are involved in command processing. First, a command must be acquired from some source of commands. Then, that command has to be processed in an appropriate way. We will now describe briefly these two aspects of command processing.

4.4.1 Command Acquisition

The `Z3PLUS` command processor acquires its next command according to the following hierarchy:

1. the multiple command line buffer
2. a `SUBMIT` file
3. a shell command
4. user input

Commands in the multiple command line buffer have the highest priority. Once the command line buffer is empty, then the command processor will see if a SUBMIT job is running, in which case it will read its next command line from the SUBMIT file. This means that a SUBMIT job acts, in effect, like a very long command line. Once that stream of commands is exhausted, the command processor will return to any shell that has been selected. Only if there is no shell will the command processor turn, as a last resort, to the user!

4.4.2 Command Resolution

Once Z3PLUS has the next command, it resolves the command according to the following hierarchy:

1. scan the FCP commands
2. scan the RCP commands
3. scan the command processor's build-in commands
4. search for a file (COM, SUB, PRL) along the search path
5. invoke the extended command processor
6. invoke the error handler

The types of file searched for along the path are controlled by the CP/M-Plus search order.[6] If the order is set, for example, to COM then SUB, then the entire path will be searched first for a COM file. If no matching COM file is found, then the path is searched again for a SUB file.

Several factors can modify the resolution hierarchy. If the current flow state is false, only FCP and shell commands are scanned. RCP and CPR resident commands are ignored; transient commands and the extended command processor (ECP) are ignored. Shells will still run.

Several prefixes can modify the resolution hierarchy. A slash prefix directs the command immediately to the extended command processor. When you know that your command is one that must be processed by the ECP, then you can save the time otherwise required to search the

[6] That order and which file types are included can be changed with the CP/M-Plus utility SETDEF.

path for a transient command. The slash prefix can also be used when there are both transient and ECP versions of the same command, and you want the ECP one.

A prefix of just a colon or a period will tell the Z3PLUS command processor to skip resident commands.[7] It also adds the currently logged directory to the command search path for this command. If a transient command is not found, the ECP will still be invoked. These prefixes can be used when the path does not include the current directory ($$) and you know that the command is there. It can also be used to force the execution of a transient program with the same name as a command resident in the RCP or CPR.

An explicit directory prefix of the form D:, U:, DU:, or DIR: has the same effect as the colon or dot prefix except that the command is searched for *only* in the specified directory and the ECP is not invoked if the command is not found there.

If an error occurs, the command processor first attempts to invoke an error handler. If none has been installed or if the one installed cannot be found, it prints the entire remaining command line in the multiple command line buffer followed by a "?".

[7] This does not apply to resident commands in the FCP. Flow control commands are checked for with *every* command, no matter what prefix might be present (a colon, a slash, DU:, or DIR:).

Chapter 5

Getting More Out of Z3PLUS

In the previous chapters, you learned about the basic operation of the `Z-System` and of the particular implementation of it provided by `Z3PLUS`. What you have seen, however, is only the tip of an iceberg! `Z3PLUS` offers a vast array of possibilities, and we will try to introduce them to you in this chapter. In fact, there are surely ways to use `Z3PLUS` that even we have not thought of, so we certainly cannot claim to be offering an exhaustive treatment here.

5.1 Alternative Invocation Commands

So far, you have learned the simplest way to get the `Z3PLUS` system running. In this section we will discuss some other ways to get the system loaded and some other versions of the system that can be loaded almost as easily as the one you've seen.

5.1.1 Alternative Loading Methods

You have learned the standard method of starting `Z3PLUS`, which is to type the simple command

`Z3PLUS`<*cr*>

Z3PLUS can also be loaded in the following ways:

- from a SUBMIT script
- from a CP/M-Plus multiple command line

A Z3PLUS command line can be included in a SUBMIT file, but it must be the last line of the SUB file. Otherwise, when the load of Z3PLUS is attempted, it will give the error report "First remove: GET."[1] Including the Z3PLUS command in a SUB file can be useful for automatically invoking Z3PLUS as part of your CP/M-Plus cold-boot procedure.

Z3PLUS can also be started as part of a CP/M-Plus multiple-command line. For example:

Z3PLUS!MYSTART<*cr*>

will start Z3PLUS and then run the command MYSTART. In this case, the default startup command (typically STARTZ3P) will not be executed; instead, the remaining commands in the CP/M-Plus multiple command line will be converted to Z-System format and copied into the Z-System multiple command line buffer, from which they will be executed. This technique can be useful if you want to start up with a custom system configuration. Note that the CP/M-Plus multiple-command separator, the exclamation point ("!"), should be used only in this particular case, i.e., when the command line is being processed by the CP/M-Plus command processor, before Z3PLUS becomes active. Also note that this loading technique cannot be combined with the SUBMIT loading technique.

5.1.2 Alternative Default Systems

All of the discussion so far in this manual has concerned the "standard" default system. There are actually two other default systems that can be loaded almost as easily. One of these is a larger version of Z3PLUS which is loaded by the command

Z3PLUS LARGE<*cr*>

[1] This is because CP/M-Plus's resident system extension GET remains loaded until the last command in a SUBMIT job has been read. Z3PLUS cannot be loaded while GET, PUT, or DosDisk RSXs are in place. Other RSXs are automatically removed.

This system uses more memory for the operating system and offers correspondingly more capability. It is almost the same as the standard system, differing only in having a 25% larger RCP buffer (four extra records or 512 more bytes) so that more resident commands can be supported. Users with low capacity or slow disk drives might prefer this system. See page 54 for more detailed information on how this system is defined and what modules are loaded.

A second alternative default system is a smaller version of Z3PLUS, which is loaded by the command

```
Z3PLUS SMALL<cr>
```

This system uses less memory and has correspondingly fewer features. In fact, it is about as spartan as a Z-System can get. It has no buffers for an RCP, FCP, or NDR. It gives the user almost 3K more TPA space for application programs and yet still offers many valuable Z-System features: shells, error handlers, extended command processing, aliases, a dynamic search path, TCAP support. See page 54 for more detailed information on how this system is defined and what modules are loaded.

5.2 Changing Systems on the Fly

We have been discussing the choices among the various Z3PLUS configuration options as though one has to choose one of them and stick with it for the duration of a session. Obviously, one could always go back to CP/M using the command

```
Z3PLUS OFF<cr>
```

and then start all over again with a new version of Z3PLUS. This is not necessary, however. New Z3PLUS configurations can be loaded while Z3PLUS is already running.

When Z3PLUS is loaded from the CP/M-Plus command prompt (a "cold" load), all of the values specified in the Z3P descriptor file[2] take effect. On the other hand, when Z3PLUS is invoked from an already running Z3PLUS system (a "warm" load), all system information with the exception of the addresses and sizes of the system modules is preserved. Shells and error handlers continue to run in the new system; the

[2]These files are described in the next section.

command search path, values in the user registers, system file names, etc. do not change. The contents of the multiple command line buffer are carried over to the new system. This means that new configurations can be loaded as part of a complex sequence of commands, even one that includes flow control operations.[3] The system doesn't miss a beat!

Why would you want to change systems on the fly like this? The usual reason is memory. If there were infinite memory, you would load the biggest, most powerful version of `Z3PLUS` you could get and leave it there all the time. In real life, especially on a `CP/M` system with its address space of only `64K`, trade-offs have to be made.

Most people most of the time will probably want to have a rather powerful version of `Z-System` (such as the standard default system). There will be occasions, however, when an application program will be run that needs more working memory than this configuration allows. Then one will want to load a smaller system.

For example, suppose we have an application program called "`BIGPROG`" that requires a lot of memory (perhaps it is a database manager). We could then enter the command

`Z3PLUS SMALL;BIGPROG ...;Z3PLUS`<*cr*>

This command line will first install the small `Z3PLUS` system with its larger `TPA`. Then it will run `BIGPROG`. Finally, after `BIGPROG` has finished, the standard `Z3PLUS` system will be reloaded. Apart from the time penalty associated with changing systems, there is no disadvantage to dropping `Z-System` features that are not needed by the application program being run.[4] In the above example, by the time the prompt has returned asking the user for his next command, the powerful verson of `Z-System` is back in place. As far as the user can tell, it was always there.

The process of changing systems can be automated to various degrees by taking advantage of `Z-System` capabilities. Suppose, for example, that `BIGPROG.COM` is in directory `BIGDIR:` and that we rename it to `BGPRG.COM`. Then, if we put the following script definition into our `ALIAS.CMD` file

[3] Provided, of course, that no flow commands have to be processed while a system with no `FCP` is active.

[4] Few application programs make use of `Z-System` features. WordStar Release 4 is the exception among applications from major software houses; it can use named directories. In this case you might want to create a `Z3PLUS` system that includes only an `NDR` buffer.

```
BIGPROG    z3plus small;bigdir:bgprg $*;z3plus
```

then we can just enter our command as we used to:

```
BIGPROG FILENAME ...<cr>
```

The ARUNZ extended command processor will automatically handle the system switching for us. Even fancier scripts can even test to see if the TPA is already large enough for a program and only change systems when necessary. There is no limit to what ingenuity can accomplish with the flexibility offered by Z3PLUS!

There are two fine points that we should mention here. The first is to note that when Z3PLUS encounters an error when attempting to load a new system, it does not simply give up. It displays an error message on the screen reporting the nature of the problem and *then it invokes the error handler.* In this way, you have a chance to correct any errors before the command sequence plows ahead, possibly leading to a catastrophe (or at least a great inconvenience).

The second point to note is that when loading a new Z3PLUS system, the size of the new multiple command line buffer must be large enough to hold the still-remaining commands from the original multiple command line buffer. If the pending commands would not fit, the new system is not loaded; instead, the error handler is invoked. You can then decide how you want to handle the situation (e.g., eliminate some of the pending commands or abort the entire operation).

5.3 Customizing Your Z3PLUS System

There are many ways to customize your Z3PLUS system and adapt it to your needs and tastes. We will consider two kinds of changes, those that you make more-or-less temporarily by using commands and those that you make more-or-less permanently by creating entirely new system configurations.

5.3.1 Temporary Changes

After you become more familiar with the basic Z3PLUS system, you will undoubtedly want to change a number of system characteristics.

Some of these changes will be temporary ones. For example, you might change the command search path to include a directory with files that you do not use all the time — and thus do not usually include in the path — but which you need for the current task. For this you would use the `PATH` utility. When you are finished with this job, you would use the utility again to restore the path to its usual configuration.

Other parameters that are often changed temporarily are the characteristics of the printer or `CRT` screen. Many `Z-System` programs will adjust automatically to these "environment" values. For example, if you are printing with a spacing of 6 lines per inch, you might set the printer data to 66 lines per page with 58 lines of text. When printing with a spacing of 8 lines per inch, you would use values of 88 and 78 instead. The `CPSET` utility is used to specify these characteristics.

Sometimes you will want such changes to be in effect for an extended period of time, but not permanently. In that case, you can add the command lines required to install these changes to your startup alias. They will then take effect automatically when you first load the `Z3PLUS` system.

5.3.2 Custom Configurations

Although, as we just described, you can easily make temporary changes in some of the system characteristics, there will probably come a time when you will want to make some of those changes permanent. It is also likely that you will at some point want to change system characteristics that cannot be changed using utility programs, such as the sizes of the system buffers (`RCP`, `FCP`, and `NDR`). For many users, none of the default system configurations will adequately meet their needs. For example, if you have a hard disk, the standard `NDR` with its capacity of only 14 names will not be large enough, and you will want to increase the capacity to 28, 42, or even more names. We are happy to report that *all of these things can be done, and quite easily.*

The characteristics of `Z3PLUS` systems are defined by "descriptor" files with a file type of `Z3P`. These files are ordinary text files that can be edited using any text editor or wordprocessor (in non-document mode). The `Z3P` files are modeled after assembly-language source files and contain definitions of values for the configurable `Z-System` environment variables along with optional comments.

A definition line has the format:

```
VARIABLENAME [ = | EQU ] VALUE [;comment]
```

The bracketed items are optional. The symbol appears first on the line. Then there can be either an equal sign (for those accustomed to high-level languages), or an `EQU` (for those accustomed to assembly language), or neither (for those accustomed to neither). Next comes the value to be assigned to the symbol. Finally, there can be a semicolon followed by a comment. Blank lines and additional comment lines can also be included freely.

You should look at the file `DEFAULT.Z3P` in `Z3PLUS.LBR` to see a complete list of the reserved variable names that can be configured. By editing a copy of this file (or one of the other `Z3P` files), you can customize the system to provide for different buffer sizes, available drives, screen and printer sizes, etc. The variable names may occur in any order, and comments in the `DEFAULT.Z3P` file explain the meanings of the variables.

Numerical values may be entered in decimal, hexadecimal, or binary form. Decimal values are the default. Append an "`H`" for hexadecimal or a "`B`" for binary. Some variables take character string values; these values must be entered as an unbroken string, that is, with no spaces.

There are some values that almost all users should customize. For example, there is a variable called `DRVEC` that tells the system which drives (A, B, etc.) exist on your computer. This variable has a bit for each of the possible 16 drives supported by `CP/M`. The default configurations have all 16 drives enabled, but very few users will actually have that many. By turning off the bits for drives that do not exist on your system, the command processor and some `Z-System` programs will gracefully prevent attempts to access these nonexistent drives.

The `Z3PLUS` command search path is an example of a string variable. Its value is a sequence of characters (remember, with no spaces) representing from one to five pairs of drive/user directories. A "$" character in the drive position indicates the currently logged-in drive; in the user number position it represents the currently logged-in user area. For example,

```
PATH = $$B0A15
```

specifies the search path:

- first, the current drive and current user area

- next, drive B:, user area 0
- finally, drive A:, user area 15

The final drive/user pair in the path (`A15:` in this example) is called the root directory.[5] Many `Z-System` operations automatically refer to this directory. For example, the `Z3PLUS` command processor looks in the root directory for the extended command processor, and `ARUNZ` and `ZFILER` look there for their `CMD` files.[6]

Another string variable that you might want to change is the name of the startup command. All three default systems use the name `STARTZ3P`. You might want to perform different startup operations for different custom configurations. If so, enter different names for the startup command and create individual startup aliases.

There are two ways to make custom configurations. One is to create completely new definitions in `Z3P` files with completely new names. The other is to modify the default definitions.

Creating New Definitions

Completely new definitions offer the greatest flexibility: all variable definitions can be changed, and whatever system modules you like can be loaded into the custom system. The default systems, as we will see in a moment, work with predetermined system modules.

The disadvantage of fully custom configurations is that the command to load them must specify every module that is to be loaded. When a module is not specified, a default module will *not* be substituted for it. If you forget to name a CPR module, for example, an error will result, since the system cannot run without a command processor.

The following example should help clarify what is required. Suppose you have created a custom system defined by `SUPER.Z3P` that will use the default command processor (`DEFCP3.ZRL`) but two special modules called `SUPERFCP.ZRL` and `SUPERRCP.ZRL` for the `FCP` and `RCP` modules,

[5] Don't confuse this root directory with a directory that happens to have the name `ROOT`. The latter is simply a name and confers no special significance on the directory. To avoid confusion, it is recommendable to actually call that directory ROOT.

[6] The versions of `ARUNZ` and `ZFILER` supplied with `Z3PLUS` have been configured to look for their `CMD` files in the root directory. Those programs can, if desired, be configured to use a different directory.

respectively. Assuming that all the files have been put into Z3PLUS.LBR, the loading command would be as follows:

```
Z3PLUS Z3PLUS.LBR SUPER.Z3P DEFCP3.ZRL
   SUPERFCP.ZRL SUPERRCP.ZRL DEFAULT.NDR
   DEFAULT.Z3T<cr>
```

That's an awful lot of typing! A good way to simplify things is to enter this command line as a script in the ALIAS.CMD file, perhaps under the name SUPER. Then to change systems while a Z3PLUS system is already running, all you have to type (assuming ARUNZ is the extended command processor) is

```
SUPER<cr>
```

On a "cold" load from CP/M you can use the following trick. Invoke the system as a secondary load by including the alias command in a CP/M multiple command line as follows:

```
Z3PLUS!SUPER<cr>
```

This will cause the standard default Z3PLUS system to load, but it will chain immediately to the "SUPER" system. You end up having to load two systems, but Z3PLUS is so fast that you will hardly notice. It will be faster than typing that long command line and will probably be faster than running a SUBMIT job (you could put the long command line in SUPER.SUB).

Modifying the Default Definitions

Another way to make custom configurations is to modify the existing default Z3P files. This approach has the advantage that the default systems can be loaded using very simple command lines. Z3PLUS automatically specifies the modules to be loaded with these configurations, so we do not have to do it.

This imposes some constraints on the kinds of changes that can readily be made to the default definitions. Changing the drive vector (DRVEC), maximum drive and user, initial path, startup command name, and printer/CRT characteristics to match your system and preferences makes perfect sense.

The sizes of modules in the default configuration *can* be changed, but only if done carefully and within limits. The `DEFAULT` system, for example, always loads an `RCP` module with the name `DEFRCP.ZRL`. If you modify `DEFAULT.Z3P` to specify a smaller `RCP`, there will be a problem. When `Z3PLUS` attempts to load `DEFRCP.ZRL`, it will discover that the module will not fit in the space allocated. It is smart enough to terminate the operation at that point with an error message. To overcome this problem, you would have to replace the default `RCP` module with a new, smaller one (but still with the name `DEFRCP.ZRL`). With enough cleverness, all of the problems of this sort can be overcome (we've done it), but we don't think it is worth the effort and we don't recommend it. There are easier ways.

Chapter 6

Technical Reference

6.1 Definition of File Types

The Z3PLUS operating system identifies the functions of certain special files by their file types. These are listed in Table 6.1 on page 50. ZRL files are special relocatable files that can be adapted to any Z3PLUS system configuration. They may be in the formats supported by either SLR Systems or Microsoft. The modules must be named as indicated in Table 6.1 by including a line in the source code using the NAME pseudo-op. There are no restrictions on the names used for the files themselves. At load time, Z3PLUS determines the kind of module coded by the file by examining the module name embedded in the code.

6.2 Files Supplied with Z3PLUS

Many files are provided with the Z3PLUS system. Most of them are listed in this section along with explanations of their functions. See the file RELEASE.NOT for information about any changes.

6.2.1 Z3PLUS System Files

The files that comprise the Z3PLUS product are listed in Table 6.2 on page 51. We want to remind you that these files are copyrighted and

FILE TYPE	CONTENTS
LBR	library file
Z3P	Z3PLUS system descriptor
Z3T	Z3 terminal capability descriptor (TCAP)
NDR	named directory register file
ZRL	named-common Z-system ReLocatable file with a REL module name of:
	CP3xxx Z3PLUS Command Processor
	RCPxxx Resident Command Package
	FCPxxx Flow Command Package
	where "x" is any character

Table 6.1: Table of file types used with the Z3PLUS system.

are licensed for a single user only. It is illegal to copy or distribute these files to any other person. See the copyright notice page at the beginning of the manual for details.

6.2.2 Tools and Utilities

Many files that are not a part of the proprietary NZ-COM system are included with the distribution package *as a convenience to users.*[1] Some of these files are public-domain. Others are copyrighted by their authors or by ZSIG, the Z-System Interest Group. With all of them, however, the authors have granted permission for them to be copied and distributed free of charge to other users for *non-commercial* use.

The following files support the Z-System TCAP facility:

TCSELECT	.COM	select a terminal descriptor
Z3TCAP	.LBR	database of terminal descriptors

The following files support Z-System aliases:

SALIAS	.COM	stand-alone alias generator
ARUNZ	.COM	alias command processor
ALIAS	.CMD	alias script file for ARUNZ

[1] Therefore, we cannot take responsibility to support these programs.

Z3PLUS	.COM	Z3PLUS system loader
Z3PLUS	.LBR	library of Z3PLUS system modules
DEFAULT	.Z3P	default system descriptor
DEFAULT	.Z3T	default terminal descriptor (adm3a)
DEFAULT	.NDR	default named directory register
DEFCP3	.ZRL	default command processor
DEFRCP	.ZRL	default resident command package
DEFFCP	.ZRL	default flow command package
SMALL	.Z3P	small system descriptor
LARGE	.Z3P	large system descriptor
LARGERCP	.ZRL	RCP for large system
TIMECP3	.ZRL	CPR with hour:minute prompt
JETLDR	.COM	Z-System package loader

Table 6.2: List of the Z3PLUS system files.

The following file is used by the FCP to process extended conditional tests:

IF	.COM	extended flow condition tester

The following utilities define or display various Z-System environment variables and system capabilities:

CPSET	.COM	displays/defines CRT/PRT characteristics
PATH	.COM	set/display command search path[2]
ZSHOW	.COM	display configuration information [3]
EDITNDR	.COM	edit named directory register in memory[4]
SAVENDR	.COM	save named directory register to file

The following programs are utilities of general interest:

FF	.COM	file finder
CRUNCH	.COM	file compression tool
UNCRUNCH	.COM	file decompression tool

[2]Name changed from the original SETPATH.COM
[3]Name changed from the original SHOW.COM
[4]Name changed from the original EDITND.COM

The following programs support library files:

LDIR	.COM	library directory program
LGET	.COM	library member extractor
LPUT	.COM	library file inserter
VLU	.COM	video-oriented library file utility
LX	.COM	library file executive

The following files are related to shells and error handlers:

ZF-REV	.COM	ZFILER shell for reverse-video terminals
ZF-DIM	.COM	ZFILER shell for dim-video terminals
ZFILER	.CMD	macro script file for ZFILER
LSH	.COM	command history shell
ZERRLSH	.COM	error handler

The following files support the Z-System help facility:

ZHELP	.COM	displays menu-driven help files [5]
xxx	.HLP	various help files

The following programs are type-4 transient programs (see p. 19) that run at the very top of the available TPA and leave low memory undisturbed. They are especially useful in a "minimum" system, where transient programs are used for functions otherwise performed by RCP commands. You will want to rename them to omit the leading TY4.

TY4SP	.COM	disk space
TY4SAVE	.COM	save memory to file
TY4REN	.COM	rename file
TY4ERA	.COM	erase file

There are several ways to tap into the rich lode of ever-expanding Z-System user-group files. The telephone numbers of the Z-Node remote access systems are listed in:

ZNODES	.LST

A number of individuals have volunteered to help others install and use Z-System. Their names, addresses, and phone numbers are listed in

[5] Name changed from the original HELP.COM

the file

```
ZHELPERS .LST
```

6.3 Z3PLUS Command Lines

6.3.1 Help Screens

A built-in help screen designed to remind you of the syntax required for Z3PLUS commands is displayed by typing either of the following commands:

```
Z3PLUS //<cr>
Z3PLUS ?<cr>
```

These help screens also report the configuration of the quiet option and system directory (see below).

6.3.2 Loading the Default Systems

There are three default systems provided with Z3PLUS; they are loaded with the following command lines:

```
Z3PLUS<cr>
Z3PLUS ON<cr>
Z3PLUS LARGE<cr>
Z3PLUS SMALL<cr>
```

The first two forms load the "standard" default system. The other forms load the LARGE and SMALL default systems.

Option switches of "/Q" or "/V" can be placed at the end of the command lines to select "quiet" or "verbose" mode. In verbose mode, Z3PLUS displays a full screen of information about the system that has been loaded. In quiet mode this display is suppressed. The built-in help screen of Z3PLUS.COM will indicate which of these two modes is used when no option switch appears on the command line.

The system modules are loaded from the library file Z3PLUS.LBR, which must be in a special directory called the "system directory". In the

distributed version of Z3PLUS.COM this is directory A0:. See Section 6.5 (p. 56) for information on how to change this. The currently configured system directory is indicated in the Z3PLUS help screen.

The standard default system uses 6.88K of memory and loads the following modules from Z3PLUS.LBR:

DEFAULT	.Z3P	system descriptor
DEFCP3	.ZRL	command processor
DEFRCP	.ZRL	resident command package
DEFFCP	.ZRL	flow command package
DEFAULT	.Z3T	terminal descriptor
DEFAULT	.NDR	named directory assignments

The LARGE system uses 7.38K of memory. The same modules as with the standard system are loaded except that:

- LARGE.Z3P is used instead of DEFAULT.Z3P
- LARGERCP.ZRL is used instead of DEFRCP.ZRL.

The SMALL system has no buffers for an RCP, FCP, or NDR, and it uses only 4.00K. The loading of modules differs from that for the standard system in the following ways:

- SMALL.Z3P is used instead of DEFAULT.Z3P
- DEFRCP.ZRL, DEFFCP.ZRL, and DEFAULT.NDR are not loaded.

6.3.3 Removing Z3PLUS

The Z3PLUS operating system can be removed from the system and the CP/M-Plus operating system restored by issuing the command:

Z3PLUS OFF<*cr*>

6.3.4 Loading Specific System Modules

Z3PLUS can be used not only to load complete operating systems but also to load specific system modules. These modules can be either individual files or members of library files. More than one module can be loaded at a time. The following command forms are used:

```
Z3PLUS filelist [/option]<cr>
Z3PLUS lbrfile memberlist [/option]<cr>
```

The item "`filelist`" is a list of one or more file names, each of the form

```
[dir:]name.typ
```

The "`lbrfile`" term has the form

```
[dir:]lbrname[.LBR]
```

Finally, "`memberlist`" is a list of one or more files contained in the designated library. The format for a member file name is the same as for a name in "filelist" except that no directory prefix is allowed.

Any pair of files listed in the command can be separated by either spaces, commas, or both. The option switches `/Q` and `/V` can be used with these commands as well.

Directories may be specified for each individual file in the first form and for the library file in the second form. Whenever there is no explicit directory specification for a file, the system directory (see Section 6.3.2) is used. When directories are specified, they can always be in `DU:` form. When a `Z3PLUS` system is running, the `DIR:` (named directory) form can also be used.

The file type `LBR` is optional in the second form. When the first file listed has no file type, `LBR` is assumed. For the second command form, all the modules listed after the library name must be in that library. One cannot load both individual files and library members with a single command.

6.4 The JetLDR Program

Included with the `Z3PLUS` package is a special program called `JetLDR`. This program is an extremely powerful general-purpose module loading program. All of the usual module loading functions required for the `Z3PLUS` system, however, can be performed by `Z3PLUS.COM`, which got that capability by borrowing code from `JetLDR`.

There are some important differences between `JetLDR.COM` and `Z3PLUS.COM`. `JetLDR` can only be run under `Z-System`, and it cannot be

used to build new systems, as Z3PLUS can. It is used only to load modules. However, it can load many more types of modules than Z3PLUS can, and it performs much more sophisticated checking to validate modules and to try to ensure that a module is a proper one for the buffer to which it is being loaded.

The JetLDR program is extensible using special configuration or CFG files. These are modules which JetLDR loads into itself and which control the way it loads other modules. This facility can give JetLDR the ability to load special modules like resident system extensions (RSXs), BIOSs, and DOSs.

JetLDR has a built-in help screen that can be invoked using the standard command

JETLDR //<*cr*>

You will see that its command syntax is nearly identical to that of Z3PLUS.

6.5 Patching Z3PLUS.COM

There are several options in Z3PLUS.COM that are controlled by configuration bytes in the first page of code. The bytes are all identified by text strings so that they will be easy to find using a debugger (such as SID, which comes with CP/M-Plus) or a patching utility.

The first pair of bytes is identified by the string "SYSUD". The two bytes following this string define the user number (00H to 0FH) and drive (A=00H to P=1FH) to be used as the system directory (see p. 53).

The next byte is identified by the string "QUIET". It determines whether the /Q or /V mode will be the default for the information screen display. When the byte is 00H, quiet mode is off (verbose mode is selected). When the byte is FFH, the default is quiet mode.

The byte after the string "TEST" is used for development and debugging purposes. Users should have no need to change this byte, but, of course, you are welcome to experiment with it if your curiosity is irrepressible.

EDITZ3PL.COM, the modern version of Z3PLUS patching avoids all these complicated things and presents you with a menu-driven choice. You may save the file Z3PLUS.COM under the same name with no risk, provided that you work with a copy.

6.6 Differences Between Z3PLUS and CP/M-Plus

There are, obviously, many differences between Z3PLUS and CP/M-Plus. In this section we want to point out some that might cause confusion or difficulties if they were not drawn to your attention.

6.6.1 GET Command

Under Z3PLUS, GET is a CPR (built-in) command that loads a file to a specified address in memory. To use the CP/M-Plus GET.COM program, prefix the command with some type of drive specification as in the following examples:

```
:GET ...<cr>
A:GET ...<cr>
BASE:GET ...<cr>
```

Alternatively, rename it to something else, such as GETCPM.COM.

6.6.2 IOP — Input/Output Packages

In the Z3PLUS version of Z-System there is no support for the Input/Output Package (IOP) of ZCPR3. Instead, the support already provided by CP/M-Plus is used. Terminal and printer control and redirection from and to a file or another device are supported through the GET.COM and PUT.COM utilities and RSX's. Device selection is supported by other CP/M-Plus utilities. See your CP/M-Plus manuals for further information.

6.6.3 Public Files

CP/M-Plus supports a "public" file system whereby executable files in user number 0 which have the "system" or $SYS attribute set can be invoked from all user areas on that drive. This feature is not recognized in Z3PLUS, where the command search path is used instead. Unlike the search path configured by SETDEF in CP/M-Plus, the Z-System search path specifies not just drives but full directories, including user numbers.

6.6.4 Passwords

Z-System provides directory passwords and drive/user limits to protect all files in specified named directories and drive/user areas. The Z3PLUS command processor does not restrict access to files that are protected by CP/M-Plus file-passwords. To protect sensitive files from execution or viewing, place them in a password-protected directory beyond the allowed maximum drive and/or user number. Books listed in the Bibliography have information on how to set up and use the Z-System security features.

Any CP/M-Plus file passwords remain in effect when a program attempts to access a file.

6.6.5 Command Search Path

The Z3PLUS command processor searches for command files along the Z-System path. It is initially set in the Z3P description file and can be changed at any time using PATH.COM. Z3PLUS disables the CP/M-Plus command search path.

A valid entry in a system with a RAM-disk M: might look like this:

PATH = M15B0$$$0M15 ; comments

Instead of M15 or B0 you might want to use a named directory entry. The $ character is a substitute for "the currently logged", thus $$ is expanded to the currently logged drive and user, $0 will refer to the currently logged drive, user 0. The last entry in your path is termed the root directory; it should be the one in which you have CMDRUN.COM, the extended command processor. To speed up searches along the path, the first entry should be the same system directory, allowing for fastest execution of aliases in ALIAS.CMD.

6.6.6 Multiple Commands

In Z3PLUS, multiple commands on one line are separated by a semicolon (";"). The Z3PLUS command processor doesn't support file passwords, so the semicolon, which is used in CP/M-Plus to prefix a password, is not needed for that. The CP/M-Plus command separator, the exclamation point ("!"), does not have a special meaning to Z3PLUS, except when Z3PLUS is first loaded (see p. 40).

6.6.7 Conditional Execution

Z3PLUS provides very extensive capabilities for testing and conditionally executing commands based on flow states and the commands in the Flow Command Package. Under Z-System, the leading colon (":") in a line of a submit file does not cause the line to be skipped if the CP/M-Plus program return code shows an error. Instead, it will be interpreted as a directive to search for the following word as an executable disk file. You will need to modify any submit files containing a leading colon before using them with Z3PLUS.

6.7 Theory of Operation

Z3PLUS is a special CP/M-Plus Resident System Extension (RSX) that loads into high memory just below the CP/M-Plus common-memory BDOS. It contains these major segments:

- warm-boot intercept
- program and RSX loader
- command processor
- Z-System resident segments:
  - external environment description
  - terminal capabilities buffer
  - message buffer
  - path
  - wheel byte
  - external file control block
  - multiple command line
  - shell stack
  - external command processor stack
- optional Z-System segment buffers
  - named-directory register
  - resident command package
  - flow command package

Z3PLUS.COM reads Z-System files into working memory from a library (e.g., Z3PLUS.LBR) or as individual files, constructs the specified system addresses and parameters, and then relocates that system into high memory. Z3PLUS protects itself by intercepting the normal BIOS

warm-boot function, diverting control to the resident Z3PLUS command processor to act on the next command.

If the Z-System is already running, Z3PLUS saves the state of the current Z-System environment, determines what changes are required (e.g., a different-sized buffer), relocates and installs new packages, and restores the unchanged components of the environment (such as message buffer values and remaining commands in the command line buffer).

When the system is removed, Z3PLUS disables the warm boot intercept, removes the Z3PLUS RSX, and lets the host CP/M-Plus system reload its standard command processor.

Both the command processor and the Z3PLUS loader set numerous parameters in the CP/M-Plus system control block. Otherwise, except for the patch to the bios warm-boot jump address, Z3PLUS makes no internal changes to the host CP/M-Plus system.

Chapter 7

Bibliography

In an evolving, improving environment the documentation always lags behind practice. Here we recommend several sources of additional information about Z-System.

7.1 The Z-Nodes

The most current information about Z-System tools and standards and the latest versions of the freely distributed programs are to be found on the Z-System remote access systems (RASs), called Z-Nodes. These "bulletin boards" are also good places to find friendly help from fellow users. The crunched file ZNODESxx.LZT contains a list of the Z-Nodes operating.

Of these many Z-Nodes we want to call special attention to some (all of which are accessible using Telenet's PC-Pursuit service). Z-Node #1, the granddaddy of them all, known as Z-Node-Central is out of service.

Z-Node #2 in Los Angeles is the nearest node to Z3PLUS-author Bridger Mitchell and the one on which he can be reached most quickly and directly. The phone number there is (213)-670-9465.

Jay Sage, one of the architects of Z3PLUS and author of ZCPR34 (on which the command processor of Z3PLUS is based), is the sysop of Z-Node #3 in the Boston area. Its phone number is (617)-965-7259. Unlike most remote access systems, it is an open system, with no indi-

vidual user registration or passwords and only public messages between users. There is, however, a general system password designed to allow access only to users of Z-System-compatible computers. This password is presently "DDT", the name of the debugging utility supplied with CP/M-2.2. Now that Z3PLUS has at last made CP/M-Plus computers Z-System-compatible, perhaps an alternative password of "SID" will be allowed!

Finally, we want to mention Chicago's Lillipute Z-Node, a misnomer if ever there was one. This node has two computers, each with a very large hard disk, and two phone lines. It is probably the largest and most active of the Z-Nodes. The sysop, Richard Jacobson, is very aggressive in making sure that he always has the very latest files available. It is a subscription system, and well worth the price. The phone numbers in area code 312 are 649-1730 and 664-1730.

7.2 *The Computer Journal*

An excellent ongoing source of Z-System material is *The Computer Journal* (TCJ), perhaps the last of the major hobbiest computer magazines with significant coverage of 8-bit systems. It has regular columns and special articles by Z-System experts, including both Jay Sage and Bridger Mitchell. A subscription is highly recommended! Contact the publisher at P.O. Box 1697, Kalispell, MT 59903.

Included with the Z3PLUS package are disk files containing a number of Jay Sage's columns from TCJ. The files are all crunched to save space; see page 34 for a discussion of how to handle crunched files. Here is a list of those files with an indication of the information in each one that may be of particular interest:

TCJ26 .MZG	optimizing a floppy-disk-based system
TCJ27 .MZG	aliases and shells
TCJ28 .MZG	recursive aliases
TCJ29 .MZG	the ZCPR33 command processor
TCJ30 .MZG	SALIAS and VLU
TCJ31 .MZG	ARUNZ documentation
TCJ32 .MZG	NZCOM/Z3PLUS/ZCPR34 information
TCJ33UPD.MZG	ALIAS.CMD information
TCJ34 .WZ	ZCPR-filetypes and programmer's information
TCJ34BMM.WZ	Z3PLUS- and RSX - information

TCJ35 .WZ	shells and commandline information
TCJ35BMM.WZ	Z3PLUS programmer's-information
TCJ36BMM.WZ	Z3PLUS programmer's-information
TCJ37 .WZ	NZ-COM/Z3PLUS/ZFILER information
TCJ37BMM.WZ	Z3PLUS BDOS + console progr. info
TCJ38 .CZL	ZEX shell + command information
TCJ38BM .WZ	ZEX ZEX + interrupt information
TCJ38BMM.WZ	ZEX ZEX + batch processing
TCJ39 .WZ	NZ-COM NZ-COM programmer's info
TCJ42 .WZ	NZ-COM NZ-COM + system security
TCJ421M .WZ	NZ-COM/BYE NZ-COM environment + BYE
TCJ43 .WZ	NZ-COM NZ-COM environment
TCJ49 .WZ	NZ-COM NZ-COM environment adjustment
TCJ52 .WZ	programming for compatibility
TCJ53 .WZ	NZ-COM NZ-COM Virtual BIOS

7.3 Other Published Information

For other printed information about Z-System, we recommend the following books.

Read first:

> *The Z-System User's Guide* (Bruce Morgen, Richard Jacobson). This is an introduction to the Z-System that tries to be comprehensible to the less technical user of Z-System.

Then read:

> *The ZCPR 3.3 User's Guide* (Jay Sage). This is the manual that accompanied the ZCPR version 3.3 command processor. It includes many examples of how the features of Z-System can be used to advantage. Extended command processing and security features, in particular, are covered. Almost all of this information applies to ZCPR version 3.4 and Z3PLUS.

An older reference, with information that is no longer always current, is:

ZCPR3: The Manual (Richard Conn). This was the bible for `ZCPR3`, but much of the material is now out of date. The treatments of the `Z-System HELP` facility, the menu shells, and `TCAP`s (terminal capability descriptors, including the utilities `TCSELECT`, `TCMAKE`, and `TCCHECK`) are still very useful.

For the technically inclined who want to write their own `Z-System` programs, the following book, along with the relocatable subroutine libraries code, will be extremely useful:

ZCPR3: The Libraries (Richard Conn). This book provides complete documentation on the hundreds of pre-written (and debugged) subroutines that make writing `Z-System` programs in assembly language almost as easy as writing in a high-level language.