```
***********************************************************
*                                                         *
*                   INSTALLATION MANUAL                   *
*                                                         *
*                           for                           *
*                                                         *
*  ZCPR3 -- Z80 Command Processor Replacement, Version 3  *
*                                                         *
***********************************************************
```

by

Richard Conn

Installation Manual
11 June 1984

ZCPR3 Version 3.0

T A B L E   O F   C O N T E N T S

L I S T   O F   F I G U R E S

1. O V E R V I E W   o f   Z C P R 3   I N S T A L L A T I O N

1.1. Introduction

Installation of ZCPR3 is an involved process, and the installer must have a working knowledge of the following:

1) 8080 and Z80 assembly language programming
2) CP/M 2.2
3) the CP/M SYSGEN procedure

There are three parts of the system which must be created or initialized during the installation process:

1) the _Operating System_ or _SYSGEN image_, which is present on the system tracks for most computers and includes a disk boot, the ZCPR3 Command Processor, the CP/M 2.2 BDOS, and a modified BIOS

2) the ZCPR3 _System Segments_, which are independent files that may be loaded from disk into the appropriate places in memory by the ZCPR3 utility named LDR.COM

3) the various ZCPR3 _utilities_, each of which has to be provided with the address of the ZCPR3 Environment Descriptor

The ZCPR3 System is tied together by the ZCPR3 Environment Descriptor, which is a set of buffers that passes information between all elements of a ZCPR3 System. The ZCPR3 Environment Descriptor contains information such as the addresses of the System Segments, the addresses of several buffers which are significant to ZCPR3, data on what ZCPR3 resources are available, and information about the physical attributes of some of the input/output devices connected to the system (such as the number of columns and lines on the CRT screen).

1.1.1. Operating System Memory Images

The installer must build a proper SYSGEN Memory Image of the target ZCPR3 System (the _target operating system_ is the system being built, as opposed to the _host operating system_ which is the system used to build the target system). In building the target system, the ZCPR3 Command Processor must be assembled and a BIOS containing a modified Cold Boot routine must be prepared.

1.1.2. System Segments

The installer must select and assemble the various ZCPR3 System Segments to be used in conjunction with the target ZCPR3 System. A _System Segment_ is a file which is loaded into a fixed location in memory by the LDR.COM ZCPR3 utility. Each System Segment stays memory-resident until a new System Segment is loaded over it. Depending on the commands issued, the ZCPR3 Command Processor or a ZCPR3 utility may call upon a loaded System Segment to perform a function or provide information.

All System Segments must be initialized by the Cold Boot routine in the BIOS of the target ZCPR3 System. This initialization consists of zeroing out the first N bytes of each segment's memory buffer, where N depends upon the segment being initialized.

The following are the System Segments which are supported by ZCPR3. Each System Segment has a distinctive file type, and LDR.COM recognizes this and loads each segment differently.

Segment File Type	Function of System Segment
*.ENV	Environment Descriptor, including a TCAP
*.Z3T	ZCPR3 TCAP Entry
*.FCP	Flow Command Package
*.IOP	Input/Output Package
*.NDR	Named Directory File
*.RCP	Resident Command Package

A _package_ as referred to above is a set of executable subroutines which is divided into two parts -- the visible section, through which an interface to the routines is provided, and the hidden section, which contains the code of the routines. Being a System Segment, a package can by loaded dynamically any time during a terminal session by running the LDR.COM utility.

1.1.2.1. Flow Command Packages

A _Flow Command Package_ is a package which implements the ZCPR3 flow commands. These commands are IF, ELSE, FI (same as ENDIF), and XIF (exit all IFs), and their function is to control the flow of command execution by setting the _Flow State_ to TRUE or FALSE. If the Flow State is TRUE, all commands are allowed to execute; if the Flow State is FALSE, only Flow Commands (IF, ELSE, FI, and XIF) are allowed to run.

An example of a command sequence containing flow commands is:

```
IF EXIST MYFILE.TXT
     TYPE MYFILE.TXT
ELSE
     ECHO MYFILE.TXT DOES NOT EXIST
FI
```

1.1.2.2. Input/Output Packages

An _Input/Output_ _Package_ is a package which contains a set of input/output drivers. The I/O Package is used by the BIOS to provide the low-level device drivers which support console input/output, list output, punch output, and reader input. An I/O Package can support many more console, list, punch, and reader devices than the standard CP/M I/O byte, and it adds the flexibility of being a package, which can be replaced with a different package dynamically by running the LDR.COM utility.

1.1.2.3. Resident Command Packages

A _Resident_ _Command_ _Package_ is a collection of memory-resident commands which can be used to supplement the commands resident within the ZCPR3 Command Processor itself (the _ZCPR3-Resident_ _Commands_). These commands replace a number of COM files by one *.RCP file, and, being memory-resident, they are executed very quickly without any additional disk accesses taking place. When the user issues a command, the current RCP is checked for a match of the command before a disk access is performed to search for a matching COM file. See the section on "Command Search Hierarchy" for more details.

1.1.2.4. Environment Descriptor and Z3TCAP

The _ZCPR3_ _Environment_ _Descriptor_ is a data file which contains information on several attributes of the ZCPR3 System. Additionally, the Environment Descriptor contains a _ZCPR3_ _TCAP_ _(Terminal_ _Capabilities)_ entry which describes various attributes of the console CRT, such as the sequence of characters to cause its screen to clear or to position its cursor.

1.1.2.5. ZCPR3 Named Directories

The _ZCPR3_ _Named_ _Directory_ file contains data relating a mnemonic, such as PASCAL or ROBERT, with a Disk and User Area (a logical directory). Under ZCPR3, Named Directories and Disk/User (DU) forms can be used to refer to logical directories:

```
DIR A15:          DIR ROOT:
```

1.1.3. Utilities

To be used effectively as a part of a ZCPR3 System, all ZCPR3 utilities must be initialized to contain either (1) a pointer to the ZCPR3 Environment Descriptor if such a descriptor is available as a System Segment or (2) the ZCPR3 Environment Descriptor itself.

The ZCPR3 utility Z3INS.COM is used to perform this initialization. Z3INS.COM will install a group of utilities with the required information very quickly and make this process relatively painless. Z3INS.COM itself does not need to be installed but can be for consistency sake.

Of all the ZCPR3 utilities (over 70) which are contained within the ZCPR3 System, only one, ZEX, cannot be installed by Z3INS. ZEX has to be assembled in order to be installed.

1.1.4. Other Basic Concepts

1.1.4.1. Command Search Hierarchy

Whenever a command is issued, the ZCPR3 Command Processor performs a series of steps in determining how the command is to be performed. These steps are called the <u>ZCPR3 Command Search Hierarchy</u>.

Under CP/M 2.2, whenever a command was issued the Console Command Processor (CCP) would perform these steps:

1. Parse the command name and check to see if it is a resident command within the CCP (like DIR or ERA); if resident, execute it (allowing the built-in command code to parse the command line the rest of the way)

2. Parse the rest of the command line, store various parts of the command line in various buffers, and look on the currently logged-in disk (or some other disk if the command was of the form "d:command") for a file named "command.COM"; if found, execute it, else give error message

The ZCPR3 Command Search Hierarchy is as follows:

1. Parse the command line, storing various parts of the command line in various buffers (similar to the CP/M 2.2 convention)

2. If the flow command package feature is enabled, pass the command name to the flow command package; if the package responds in the affirmative, allow the flow command package to execute the command and resume with the next command line

3. If the flow command feature is enabled, check to see if the flow state is TRUE (no IF is in effect or the current IF is TRUE); if not, skip further command processing and resume with the next command line

4. If the resident command package feature is enabled, pass the command name to the resident command package; if the package responds in the affirmative, allow the resident command package to execute the command and resume with the next command line

5. Check to see if the command is resident within the ZCPR3 Command Processor itself; if so, invoke the code to execute the command and resume with the next command line

6. Search along the Command Search Path for a file named "command.COM"; if found, load it and execute it

7. If the Extended Command Processor (ECP) feature is enabled, locate the ECP; if found, load it and execute it

8. If ZCPR3 Messages are enabled, check to see if an Error Handler is installed; if so, load it and execute it

9. Print a "command not found" error message

1.1.4.2. Command Search Path

The _Command Search Path_ is a buffer which contains an expression (in the form of byte pairs) of the sequence of directories to examine when the ZCPR3 Command Processor searches for a COM file. It is recommended that this buffer be placed external to the ZCPR3 Command Processor (be enabled as an _External Path_) so that the ZCPR3 utilities may readily access and modify it.

The elements of a Command Search Path are byte pairs. The first byte indicates what disk to look on, and the second byte indicates what user area. The value of the first byte may be in the range from 1 to 16 to indicate disks ’A’ to ’P’, or this byte may be the character ’$’ to indicate the current disk. The value of the second byte may be in the range from 0 to 31 to indicate user areas 0 to 31, or this byte may be the character ’$’ to indicate the current user area. _Current Disk_ and _Current User Area_ refer to the disk and user area which were logged into at the time the command was executed by the ZCPR3 Command Processor. A value of 0 for the first byte of a byte pair is used to indicate the end of the Command Search Path.

The following is a sample Command Search Path expression:

```
        DB   ’$’,0     ; Current disk, user area 0
        DB   1,’$’     ; Disk A, current user area
        DB   1,15      ; Disk A, User Area 15
        DB   0         ; End of Path
```

1.2. SYSGEN Memory Images

The SYSGEN memory images of a conventional CP/M system and a ZCPR3-based system are presented below. The actual addresses may vary from system to system, and the installer should be aware of what these addresses are for the specific target system.

```
Address         CP/M Image                ZCPR3 Image

                ---------------------     ----------------------
                | BIOS              |     | BIOS with Modified |
                |                   |     |   Cold Boot *      |
BDOS+0E00H-->   ---------------------     ----------------------
                | BDOS              |     | BDOS (No Change)   |
CCP +0800H-->   ---------------------     ----------------------
                | CP/M 2.2 CCP      |     | ZCPR3 *            |
BOOT+0080H-->   ---------------------     ----------------------
                | BOOT              |     | BOOT               |
BASE+xxxxH-->   ---------------------     ----------------------
                | Dead Space/SYSGEN |     | Dead Space/SYSGEN  |
BASE= 100H-->   ---------------------     ----------------------
```

FIG 1-1: CP/M and ZCPR3-based SYSGEN Memory Images

Installation requires a modified BIOS image and a ZCPR3 image to be placed over the original CP/M 2.2 BIOS and CCP images. The rest of the system can stay the same. Those new images are marked with an asterisk (*) above.

Typical address values are indicated below:

Value	SYSGEN Image Conventional CP/M	SYSGEN Image Morrow CP/M
xxxxH	800H	~ 980H
BOOT = BASE + xxxxH	900H	1080H
ZCPR3 = BOOT + 80H	980H	1100H
BDOS = ZCPR3 + 800H	1180H	1900H
BIOS = BDOS + 0E00H	1F80H	2700H
End of Operating System	????H	2DFFH

FIG 1-2: SYSGEN Memory Image Addresses

1.3. System Segments

Installation of the ZCPR3 System Segments involves selecting the features of the segments and then assembling each segment in turn. It is recommended that the MAC assembler of Digital Research be used to perform these assemblies.

The ZCPR3 Environment Descriptor (*.ENV file) is created by assembling the file SYSENV.ASM. During this process, the files Z3BASE.LIB and SYSENV.LIB are read in and used by the assembler. Z3BASE.LIB defines the memory configuration of the system and makes up most of the environment descriptor information. SYSENV.LIB contains additional details on the system.

The ZCPR3 TCAP files (*.Z3T) are created by running the TCSELECT or TCMAKE programs. TCSELECT allows the user to select his terminal from a list of pre-defined terminals, while TCMAKE allows the user to define the attributes of his terminal directly. TCMAKE is for users whose terminal does not appear in the standard Z3TCAP.TCP file.

Flow Command Packages (*.FCP) are created by assembling SYSFCP.ASM. During this process, the files Z3BASE.LIB and SYSFCP.LIB are read in and used by the assembler. SYSFCP.LIB defines the features supported by the Flow Command Package being created.

Input/Output Packages (*.IOP) are created by assembling SYSIOP.ASM. During this process, the file Z3BASE.LIB is read in and used by the assembler. All features of the I/O Package are hard-coded into the source of the package.

Resident Command Packages (*.RCP) are created by assembling SYSRCP.ASM. During this process, the files Z3BASE.LIB and SYSRCP.LIB are read in and used by the assembler. SYSRCP.LIB defines the features supported by the Resident Command Package being created.

Named Directory Files (*.NDR) are created in one of two ways: (1) by assembling the file SYSNDR.ASM or (2) by running the MKDIR.COM ZCPR3 utility. MKDIR.COM allows the user to dynamically edit and create new named directory structures while online.

1.4. Utilities

The installation of most of the ZCPR3 utilities involves setting up a file containing the names of the utilities to be installed and running the Z3INS.COM ZCPR3 utility on an Environment Descriptor and this file. Z3INS will install each utility named in the file with the information it needs from the Environment Descriptor.

Only one ZCPR3 utility cannot be installed in this way -- ZEX. Due to the nature of ZEX and the way it interacts with the system, installation of ZEX requires an involved sequence of assemblies and other operations. This sequence is described in detail in the ZEX.ZEX command file, and, once ZEX has been installed the first time, the installation of new versions of ZEX can be done automatically by running the ZEX.ZEX command file by the previous version of ZEX (issuing the command "ZEX ZEX").

1.5. Installation Steps

The installation process for ZCPR3 involves these steps:

1) Selecting the features desired for the target ZCPR3 System

2) Planning the memory structure of the target ZCPR3 System (the file Z3BASE.LIB is created)

3) Modifying the Cold Boot routine in the BIOS of the target ZCPR3 System to initialize the selected features which require initialization

4) Enabling the desired features in the ZCPR3 Command Processor (the file Z3HDR.LIB is created)

5) Overlaying the CCP with ZCPR3 and the old BIOS with the new BIOS in the SYSGEN Image

6) Placing the new SYSGEN Image onto the Operating System tracks of the disk

7) Selecting the options for the desired System Segments and creating the System Segments

8) Installing the desired ZCPR3 utilities

1.6. Operational ZCPR3 System

This section shows an operational ZCPR3 System, describing its memory image, System Segments, and utilities.

1.6.1. Memory Image

This memory image shows the memory structure of a ZCPR3 System which includes all of the major features.

```
Address
 FFFF     ----------------------------------------------
          |    ROM Area (System Dependent)             | 2K
 F800     ----------------------------------------------
          |    ZCPR3 External Stack                    |\
 F7D0     ---------------------------------------------- \
          |    ZCPR3 Command Line Buffer               |  \
 F700     ----------------------------------------------   \
          |    ZCPR3 Memory-Based Named Directory (S)  |    |
 F600     ----------------------------------------------    |
          |    ZCPR3 External File Control Block       |
 F5D0     ----------------------------------------------  1K
          |    ZCPR3 Message Buffers                   |
 F580     ----------------------------------------------    |
          |    ZCPR3 Shell Stack                       |    |
 F500     ----------------------------------------------   /
          |    ZCPR3               |      Z3TCAP (S)   |  /
 F480     |         Environment    --------------------- /
          |              Descriptor (S)                |/
 F400     ----------------------------------------------
          |    ZCPR3 Flow Command Package (S)          |  0.5K
 F200     ----------------------------------------------
          |    ZCPR3 Input/Output Package (S)          |  1.5K
 EC00     ----------------------------------------------
          |    ZCPR3 Resident Command Package (S)      |  2K
 E400     ----------------------------------------------
          |    ZCPR3 BIOS with Modified Cold Boot      |
          |         Routine to Initialize All Elements |  3.5K
          |         of the ZCPR3 System Above          |
 D600     ----------------------------------------------
          |    CP/M BDOS                               |  3.5K
 C800     ----------------------------------------------
          |    ZCPR3 Command Processor                 |  2K
 C000     ----------------------------------------------
          |    Transient                               |
          |         Program                            | ~48K
          |              Area                          |
  100     ----------------------------------------------
          |    CP/M and ZCPR3 Buffers                  |256 bytes
    0     ----------------------------------------------
```

FIG 1-3: ZCPR3 System Memory Image (Z3BASE1.LIB)

Notes: (1) All Areas Above E400H are initialized by the Cold Boot Routine in the BIOS
(2) Those Areas marked with (S) are ZCPR3 System Segments

1.6.2. System Segments

Z3BASE1.LIB	The System Segments used in this system

Z3BASE2.LIB Z3HDR1.LIB Z3HDR2.LIB	are provided in the distribution files of ZCPR3. The ZCPR3 System shown here is defined by the file Z3BASE1.LIB, and a much smaller system which does not include the Resident Command Package, Input/Output Package, and Flow Command Package features (only 1K of additional overhead) is defined in Z3BASE2.LIB. Associated with each of the two Z3BASEn.LIB files is a Z3HDRn.LIB file which defines the features of the ZCPR3 Command Processor.
SYSENV.ASM SYSENV.LIB	The Environment Descriptor is created by assembling SYSENV.ASM, which uses Z3BASE1.LIB (renamed to Z3BASE.LIB) and SYSENV.LIB during the assembly process.
SYSFCP.ASM SYSFCP1.LIB SYSFCP2.LIB	There are two Flow Command Packages used in conjunction with this system, and they are defined by the files SYSFCP1.LIB and SYSFCP2.LIB. SYSFCP1.LIB defines an FCP which is self-contained and executes without using any external files. SYSFCP2.LIB executes the ELSE/FI/XIF commands within itself, but it executes IF by loading the file IF.COM from the ROOT directory and transferring control to it. This eliminates the restriction of capabilities of the IF command which is imposed by the small size of the FCP.
SYSIOP.ASM	The Input/Output Package used in conjunction with this system is contained in the file SYSIOP.ASM.
SYSRCP.ASM SYSRCP1.LIB SYSRCP2.LIB SYSRCP3.LIB SYSRCP4.LIB	There are four RCPs used in conjunction with this system, and they are defined by the four SYSRCPn.LIB files (n is between 1 and 4). Each RCP contains a different set of commands with a different set of options enabled for the included commands.

1.6.3. Utilities

Over 70 utilities are associated with the ZCPR3 System. Each utility uses features of the system as it requires, including named directory references, access to the various system segments, access to the TCAP facility, and access to all of the data elements in the ZCPR3 Environment Descriptor that it needs. The ZCPR3 Environment Descriptor is the single source for all information that a ZCPR3 utility needs about the system it is running in.

Consequently, all ZCPR3 utilities access the ZCPR3 Environment Descriptor in one of two ways: (1) they contain a pointer to the descriptor or (2) they contain the descriptor itself. The Z3INS.COM utility is used to install the ZCPR3 utilities with the address of the Environment Descriptor or the descriptor itself. Class 1 utilities are those who contain a pointer to an environment descriptor, and Class 2 utilities contain the descriptor itself.

Supporting the Environment Descriptor in a global memory buffer is the recommended way to implement a ZCPR3 System. This buys the system two distinct advantages:

1. Each utility needs only 2 additional bytes of overhead (the pointer to the Environment Descriptor) rather than the descriptor itself (which occupies 256 bytes).

2. Changes can be made to the system dynamically without having to modify anything other than the Environment Descriptor.

The ZCPR3 utilities are much smaller and faster than their ZCPR2 ancestors. For a complete listing of all ZCPR3 utilities supplied with the distribution, see the appropriate section.

1.7. Software Required for Installation

1.7.1. Commercial Software

ZCPR3 is to be installed on a working CP/M 2.2 system. The commercial software required to do this installation is:

1) A working CP/M 2.2 System
2) Source to the BIOS of the target CP/M 2.2 System or an overlay patch for the Cold Boot Routine
3) the MAC assembler of Digital Research
4) a debugger, like DDT, for the overlay process
5) a disk utility, like SYSGEN, to be used to place the operating system image onto the OS tracks on disk

If the user desires to edit and reassemble the utilities, the Microsoft M80 and L80 assembler and linker are also required.

1.7.2. System Segment Software

The software supplied with ZCPR3 which is required for installation is:

Name of File	Function
ZCPR3.ASM	Source to the ZCPR3 Command Processor
Z3HDR.LIB	Configuration File read in by ZCPR3.ASM to tailor the ZCPR3 Command Processor
Z3BASE.LIB	Definition of the Memory Map of the ZCPR3 System to be created
SYSENV.ASM	ZCPR3 System Environment Descriptor
SYSENV.LIB	Header for ZCPR3 System Environment Descriptor
SYSFCP.ASM	ZCPR3 Flow Command Package source
SYSFCP.LIB	Configuration File read in by SYSFCP.ASM to tailor the ZCPR3 Flow Command Package (this file may be derived from one of the SYSFCPn.LIB files below)
SYSIOP.ASM	ZCPR3 Input/Output Package source
SYSNDR.ASM	ZCPR3 Named Directory Definition File source
SYSNDR.LIB	Header for ZCPR3 Named Directory Definition
SYSRCP.ASM	ZCPR3 Resident Command Package source
SYSRCP.LIB	Header for ZCPR3 Resident Command Package (this file may be derived from one of the SYSRCPn.LIB files below)

1.7.3. Other Useful Files

Name of File	Function
Z3LOC.COM	Utility to locate a CP/M CCP
Z3BASE1.LIB Z3BASE2.LIB	Sample ZCPR3 BASE files (Z3BASE.LIB)
Z3HDR1.LIB Z3HDR2.LIB	Sample ZCPR3 HDR files (Z3HDR.LIB)
SYSFCP1.LIB SYSFCP2.LIB	Sample ZCPR3 Flow Command Package headers
SYSRCP1.LIB SYSRCP2.LIB SYSRCP3.LIB SYSRCP4.LIB	Sample ZCPR3 Resident Command Package headers

1.7.4. Files Required for Installing ZEX

If the ZEX Command File Processor is to be installed to be used under the target ZCPR3 system, the following files are required.

Name of File	Function
ZEX.ASM	Source to ZEX
ZEX.ZEX	ZEX Command File used to assemble new versions of ZEX once the first version is running
RELS.UTL	SID/ZSID Utility File (not supplied with ZCPR3)

1.7.5. Required Distribution Files

The following files are required for the installation of a complete ZCPR3 System.

Name of File	Name of File	Name of File
SYSENV.ASM	SYSFCP1.LIB	Z3BASE.LIB
SYSFCP.ASM	SYSFCP2.LIB	Z3BASE1.LIB
SYSIOP.ASM	SYSNDR.LIB	Z3BASE2.LIB
SYSNDR.ASM	SYSRCP1.LIB	Z3HDR.LIB
SYSRCP.ASM	SYSRCP2.LIB	Z3HDR1.LIB
ZCPR3.ASM	SYSRCP3.LIB	Z3HDR2.LIB
SYSENV.LIB	SYSRCP4.LIB	ZEX.ASM

1.7.6. Useful Distribution Files

The following files are useful, but not required, for the installation of a ZCPR3 System.

Name of File	Name of File	Name of File
Z3LOC.COM	Z3INS.COM	ZEX.ZEX

1.8. Required Hardware

1.8.1. Hardware Required for Installation

The hardware requirements for the installation of ZCPR3 are as follows:

```
          CP/M 2.2 - based system (or ZCPR3 - based system)
          8080 or Z80 microprocessor
          32K bytes of memory
          110K bytes of disk space for source, BAK, and HEX files
          computer terminal
```

1.8.2. Hardware Required for Running ZCPR3

The hardware requirements for running ZCPR3 are:

```
          ZCPR3 - based system
          Z80 microprocessor
          48K bytes of memory
          110K bytes per disk (recommended minimum)
          computer terminal
```

2. S T E P 1 : S E L E C T I N G t h e F E A T U R E S

2.1. Features of ZCPR3

The installer must first decide what features the ZCPR3 System is to include, and his choices are:

1) Standard Overhead -- Is the ZCPR3 System to include the standard 1K overhead or not? If not, which parts of the Standard Overhead are to be included?

2) Flow Command Package -- Is the System to include Flow Commands or not?

3) Input/Output Package -- Is the System to include I/O Packages or not?

4) Resident Command Package -- Is the System to include Resident Commands or not?

Beyond these basic decisions, the contents of the following configuration files have to be determined:

1) Z3BASE.LIB -- Base Addresses for the System

2) Z3HDR.LIB -- Configuration Options for the ZCPR3 Command Processor

3) SYSFCP.LIB -- Configuration Options for the Flow Command Packages (only if this feature is selected)

4) SYSRCP.LIB -- Configuration Options for the Resident Command Packages (only if this feature is selected)

2.2. Standard Overhead

The Standard Overhead of a ZCPR3 System consists of all buffers above 0F400H in Fig 1-3. These buffers contain:

1) External Stack
2) Command Line Buffer
3) Memory-Based Named Directory
4) External File Control Block
5) Message Buffers
6) Shell Stack
7) Environment Descriptor

The tradeoff analysis of whether to include these buffers or not follows. As a general recommendation, the features supported by these buffers are fundamental to be basic nature of ZCPR3 and it is highly recommended that all of the Standard Overhead be included. The cost of doing this is 1K bytes.

2.2.1. External Stack

The External Stack occupies 48 bytes, and its purpose is two-fold: (1) to free up this space within the ZCPR3 Command Processor for other purposes and (2) to provide a common stack which can be easily accessed by the ZCPR3 utilities to restore system integrity when required.

Initialization: The external stack need not be initialized before use.

2.2.2. Command Line Buffer

The Command Line Buffer occupies just over 200 bytes, and it is instrumental to many functions of the ZCPR3 System. Its purpose is to store the command line input by the user from his terminal, by an executing SUBMIT file, or by a ZCPR3 utility such as ALIAS or MENU. If this buffer is not supported externally (as recommended), then space will be taken up inside of the ZCPR3 Command Processor for it. If made external to the ZCPR3 command processor, this buffer provides a mechanism to implement the following capabilities:

1) multiple commands on a single line, like:
```
DIR;ERA *.BAK;DIR
```

2) certain useful front-ends, such as MENU

3) the ALIAS feature

Initialization: The Command Line Buffer MUST be initialized before it is used. The first time this buffer is used is the first time the ZCPR3 Command Processor is executed, so this initialization MUST be done during (or before, in some rare cases) the cold boot procedure in the BIOS.

2.2.3. Memory-Based Named Directory

The Memory-Based Named Directory (256 bytes) contains the name-DU assignments for the named directories known to the system. 256 bytes is the recommended size, but this could be extended if more names are desired. Each name requires 18 bytes, so 14 names can be defined in the example. This feature, while is finds immediate application on a hard disk system, is also convenient on a floppy-based system, and it costs little.

Initialization: It is generally recommended that the Memory-Based Named Directory buffer be initialized during cold boot, but this is not mandatory. If the ZCPR3 command processor is set up to give precedence to the DU form over the DIR form, then this directory buffer may be initialized by a STARTUP alias (the command "STARTUP" is stored in the Command Line Buffer as a cold boot command). STARTUP may then run LDR, which will load an NDR (Named Directory) file. As with the Command Line buffer, it is important to remember that the Memory-Based Named Directory buffer must be initialized before it is used.

2.2.4. External File Control Block

The _External_ _File_ _Control_ _Block_ occupies only 36 bytes (48 bytes were reserved for it in the example), and its purpose is two-fold: (1) to free up space inside of the ZCPR3 Command Processor and (2) to provide a mechanism by which a utility can determine the name it was invoked by. The ZCPR3 Command Processor stores the name of the command it just parsed into this buffer so that the command can read it and use it. Shells commonly use this feature to determine the name they were invoked under so they can set themselves up to be reexecuted.

Initialization: No initialization is required for the External File Control Block.

2.2.5. Message Buffers

The _Message_ _Buffers_ of ZCPR3 occupy only 80 bytes, and they are very important as a mechanism thru which the following operations can be performed: (1) ZCPR3 can leave messages about its status which can be read by utilities executed by it, (2) programs can leave messages to ZCPR3 to give it instructions on how to perform certain operations, such as error handling and shell execution, and (3) one program can leave a message to be read and interpreted by another program which is executed later.

It cannot be emphasized enough that the ZCPR3 Message Buffers are MOST important for the operation of the system and should be included as a feature.

Initialization: Like the Command Line buffer, the Message Buffers must be initialized before they are used, and the ZCPR3 Command Processor begins using the Message Buffers immediately after cold boot.

2.2.6. Shell Stack

The Shell Stack permits the shell feature of ZCPR3 to be implemented, costs only 128 bytes, and also permits the shell feature to be extended to include invocation of one shell on top of another shell. Shells are front-end processors which are invoked in place of the ZCPR3 Command Processor input routine, and they allow command input in a variety of different, perhaps more user-friendly forms to take place. The MENU, SH, and VFILER utilities are invoked as shells under ZCPR3. Having a shell stack allows one shell, like MENU, to run another shell, like VFILER. The first shell is suspended, the second shell runs as long as desired, and, when the second shell is exited, the first shell is resumed.

Initialization: The Shell Stack must be initialized by the cold boot routine. Reasons are the same as for the Command Line Buffer.

2.2.7. Environment Descriptor

The Environment Descriptor (256 bytes) contains much detail on the ZCPR3 environment, including information on what features are available and other data on the operation of the ZCPR3 System. The ZCPR3 TCAP entry for the user's CRT terminal is included in the Environment Descriptor. If the Environment Descriptor is not supported externally (as recommended), then each ZCPR3 utility must be assembled to include a copy of the ZCPR3 Environment Descriptor within it. If the Environment Descriptor is supported externally, each ZCPR3 utility contains only a pointer (2 bytes) to the descriptor. Installation of a utility amounts to only setting this pointer.

Initialization: The ZCPR3 Environment Descriptor may be initialized by either the cold boot routine in the BIOS or by the execution of LDR on an ENV file as a STARTUP command.

2.3. Flow Command Packages

The Flow Command Package of ZCPR3 implements the basic flow constructs of the ZCPR3 System. These are the IF, ELSE, FI, and XIF commands, and, with this feature installed, command sequences like the following are possible:

```
IF EXIST MYFILE.TXT
     TYPE MYFILE.TXT
ELSE
     IF ERROR
          ECHO MYFILE.TXT NOT FOUND
     FI
FI
```

Initialization: The Flow Command Package MUST be initialized by the cold boot routine. Reasons are the same as those for the Command Line Buffer.

2.4. Input/Output Packages

The Input/Output Package of ZCPR3 implements a set of Input/Output drivers which can be loaded dynamically to configure and extend the input/output system of the user's computer.

Initialization: The Input/Output Package MUST be initialized by the cold boot routine. Reasons are the same as those for the Command Line Buffer.

2.5. Resident Command Packages

The Resident Command Package of ZCPR3 implements a set of commands which remain in memory until the package is explicitly reloaded by the LDR.COM utility. These commands can be used to extend the set of commands resident within the ZCPR3 Command Processor, and they add the flexibility of being able to be reloaded from time to time with different commands.

Initialization: The Resident Command Package MUST be initialized by the cold boot routine. Reasons are the same as those for the Command Line Buffer.

2.6. Other Buffers

2.6.1. External Path

The External Path (consisting of a few byte pairs) is a buffer which contains the symbolic expression of the Command Search Path to be followed by the ZCPR3 command processor when searching for a COM file.

Initialization: The External Path MUST be initialized by the cold boot routine.

2.6.2. Wheel Byte

The Wheel Byte (1 byte) is a flag read by some ZCPR3 utilities which defines the user to be priveleged or not. If this byte is non-zero, the user is declared to be priveleged, and certain functions are enabled which are not normally available to him. PWD (Print Working Directories), for example, will also display passwords to these directories if the user is priveleged and requests them.

Initialization: The Wheel Byte should be initialized before a utility which reads it is used.

3. S T E P   2 :   Z C P R 3   M E M O R Y   S T R U C T U R E

3.1. Z3BASE.LIB

The file Z3BASE.LIB defines the memory structure of the ZCPR3 System. This file is "included" by a number of the ZCPR3 System Segments when they are assembled in order to provide information to them on the memory structure of the system they are being assembled for. Z3BASE.LIB provides one source of information about the target ZCPR3 System to all utilities and ZCPR3 System Segments.

Z3BASE.LIB is divided into two parts: (1) the comment header, which outlines the memory structure of the system in a manner similiar to Fig 1-3, and (2) the body, which contains a series of equates which define addresses of elements in the system and other information about various attributes of the system.

3.2. Z3BASE Header

The following figure shows the comment header of the example ZCPR3 System Z3BASE.LIB file. It is recommended that the installer fill out the details of the address range and features supported in the target ZCPR3 System in a Z3BASE.LIB file before he do any programming and make a copy of Z3BASE.LIB as a reference for himself during the installation process.

```
;*****************************************************************
;*  Z3BASE.LIB -- Base Addresses for ZCPR3 System by R Conn     *
;*                                                               *
;*      Address Range     Size  Function                         *
;*          0 -   FF    256 b   Standard CP/M Buffers except     *
;*         40 -   4A     11 b    for ZCPR3 External Path         *
;*         4B             1 b   Wheel Byte                       *
;*        100 - BFFF    ~48  K TPA                               *
;*       C000 - C7FF       2  K ZCPR3 Command Processor          *
;*       C800 - D5FF       3.5K BDOSZ                            *
;*       D600 - E3FF       3.5K CBIOSZ with Buffers              *
;*       E400 - EBFF       2  K Resident Command Package         *
;*       EC00 - F1FF       1.5K Redirectable I/O Driver Package *
;*       F200 - F3FF       0.5K Flow Command Package             *
;*       F400 - F4FF    256 b   Environment Descriptors          *
;*                              Bytes 00H-7FH:  Z3 Parameters    *
;*                              Bytes 80H-FFH:  Z3 Terminal Cap *
;*       F500 - F57F    128 b   ZCPR3 Shell Stack                *
;*       F580 - F5CF     80 b   ZCPR3 Message Buffers            *
;*                              Byte 0:  Error Flag (Z/NZ)       *
;*                              Byte 1:  IF (8 Levels)           *
;*                              Byte 2:  IF Active (8 Levels)    *
;*                              Byte 3:  Z3 Cmd Status           *
;*                                      00B - Normal             *
;*                                      01B - Shell              *
;*                                      10B - Error              *
;*                              Bytes 4&5: Error Address if 10B *
;*                              Byte 6: Program Error Code       *
;*                              Byte 7: ZEX Message Byte         *
;*                                      00B - Normal             *
;*                                      01B - Z3 Prompt          *
;*                                      10B - Suspend Intercept *
;*                              Byte 8: ZEX Running Flag (0=No) *
;*                              Bytes 9-10: Address of Next      *
;*                                      Char for ZEX to Return   *
;*                              Bytes 11-12: Address of First    *
;*                                      Char in ZEX Memory-      *
;*                                      Based File Buffer        *
;*                              Byte 13: SH Control Byte         *
;*                                      Bit 0: Enable SHCMT      *
;*                                      Bit 1: Enable SHECHO     *
;*                                      Bit 7: Enable Shell      *
;*                                              Entry Wait       *
;*                              Bytes 14-15: Shell Scratch       *
;*                              Bytes 10H-2FH: Error Cmd         *
;*                              Bytes 30H-39H: Registers         *
;*                              Bytes 3AH-3FH: Reserved          *
;*                              Bytes 40H-4FH: User-Defined      *
;*       F5D0 - F5FF     48 b   ZCPR3 External FCB               *
;*       F600 - F6FF    256 b   Memory-Based Named Directory     *
;*       F700 - F7CF    208 b   Multiple Command Line Buffer     *
;*       F7D0 - F7FF     48 b   ZCPR3 External Stack             *
;*       F800 - FFFF       2  K ROM                              *
;*****************************************************************
```

FIG 3-1: Z3BASE.LIB Comment Header

3.3. Z3BASE Body

The following is a reformatted duplicate of the body of the

Z3BASE.LIB file. It is provided here to provide additional information on how to set the equates. It may be useful to the installer to have this installation manual open to these pages while he is editing the Z3BASE.LIB file.

3.3.1. Version Numbers, Memory Size, and CP/M Base

The following equates define the version numbers of the ZCPR3 Command Processor and the CBIOSZ. They also explicitly state the size of the TPA for inclusion in the CBIOSZ header printed at Cold Boot.

```
Z3REV   EQU     30      ; ZCPR3 REV NUMBER
CBREV   EQU     41      ; CBIOSZ REV NUMBER
MSIZE   EQU     48      ; SIZE OF TPA
```

These equates are usually used by the BIOS to print these details in its Cold Boot signon message. They are not used by any ZCPR3 System Segments other than the BIOS.

BASE - Base Address of user’s CP/M system (normally 0 for DR version). This equate allows easy modification by non-standard CP/M (eg,H89)

```
BASE    EQU     0
```

This equate is provided to establish whether a standard system (ORG 0) is being built or not, and, if not, what the base address of the system is.

3.3.2. Processor Selection

The following equate selects the use of the 8080/8085 micro or the Z80 micro for the target for ZCPR3. Note that selecting the 8080/8085 should be done ONLY if you have an 8080 or 8085. If you have a Z80, by all means set I8080 to FALSE since the code is much smaller and you can put more features into the system as a result.

If the processor is an 8080 or 8085, set this equate to TRUE. If the processor is a Z80, set it to FALSE.

```
I8080   EQU     FALSE
```

ZCPR3 can be assembled to run on an 8080, 8085, or Z80 microprocessor. If this equate is FALSE, the Z80 is selected, and, by using relative jumps, more features can be packed in than under an 8080 or 8085.

3.3.3. External Path

The following equates define the address of the ZCPR3 External Path and the number of two-byte elements contained in this path (maximum). If there is no ZCPR3 External Path, both of these values should be set to 0.

```
EXPATH  EQU     40H     ; EXTERNAL PATH
EXPATHS EQU     5       ; 5 2-byte Path Elements
                        ;  (PATH SIZE = EXPATHS*2 + 1)
```

If no external path is to be employed, set both of these equates to 0. ZCPR3 will then reserve space within itself for the command-search path.

See Section 2.6.1 for more detail.

3.3.4. Wheel Byte

The following equate defines the address of the ZCPR3 Wheel Byte. If there is no ZCPR3 Wheel Byte, this value should be set to 0.

```
Z3WHL   EQU     4BH     ; WHEEL BYTE ADDRESS
```

If there is no Wheel Byte, set this equate to 0. The C3H instruction (JMP) at memory location 0 will be used as the Wheel Byte then, and, being non-zero, the Wheel Byte will always be TRUE. If this equate is set to 0, be sure to not provide the user with the commands to change this byte, since, by so doing, he will wipe out the warm boot jump at location 0.

See Section 2.6.2 for more detail.

3.3.5. CCP Location

The following equate defines the address of the ZCPR3 Command Processor. This address MUST be supplied.

```
CCP     EQU     0C000H  ; ZCPR3 COMMAND PROCESSOR
```

This value can be obtained by calculation or by using the Z3LOC utility.

3.3.6. RCP Location

The following equates define the address of the ZCPR3 Resident Command Package and its size in 128-byte blocks. If there is no ZCPR3 Resident Command Package, both of these values should be 0.

```
RCP     EQU     0E400H  ; RESIDENT COMMAND PACKAGE
RCPS    EQU     16      ; 16 128-byte Blocks (2K bytes)
```

See Section 5.1 for more detail.

3.3.7. IOP Location

The following equates define the address of the ZCPR3 Input/Output Package and its size in 128-byte blocks. If there is no ZCPR3 Input/Output Package, both of these values should be 0.

```
IOP     EQU     0EC00H  ; REDIRECTABLE I/O PACKAGE
IOPS    EQU     12      ; 12 128-byte Blocks (1.5K bytes)
```

See Section 5.3 for more detail.

3.3.8. FCP Location

The following equates define the address of the ZCPR3 Flow Command Package and its size in 128-byte blocks. If there is no ZCPR3 Flow Command Package, both of these values should be 0.

```
FCP     EQU     0F200H  ; FLOW COMMAND PACKAGE
FCPS    EQU     4       ; 4 128-byte Blocks (0.5K bytes)
```

See Section 5.2 for more detail.

3.3.9. ENV Location

The following equates define the address of the ZCPR3 Environment Descriptor and its size in 128-byte blocks. If there is no ZCPR3 Environment Descriptor, both of these values should be 0.

```
Z3ENV   EQU     0F400H  ; ENVIRONMENT DESCRIPTORS
Z3ENVS  EQU     2       ; SIZE OF DESCRIPTOR IN 128-BYTE BLOCKS
```

See Section 2.2.7 for more detail.

3.3.10. Shell Stack

The following equates define the address of the ZCPR3 Shell Stack, the number of entries permitted in the ZCPR3 Shell Stack, and the size of each entry in the Shell Stack in terms of bytes. If there is no ZCPR3 Shell Stack, all three values should be 0.

```
SHSTK   EQU     0F500H  ; ZCPR3 SHELL STACK
SHSTKS  EQU     4       ; NUMBER OF SHSIZE-BYTE SHELL STACK ENTRIES
SHSIZE  EQU     32      ; SIZE OF A SHELL STACK ENTRY
                        ;   (STACK SIZE = SHSTKS * SHSIZE)
```

The total amount of space occupied by the shell stack is SHSTKS*SHSIZE. In this configuration, 128 bytes are used (4*32).

See Section 2.2.6 for more detail.

3.3.11. ZCPR3 Messages

The following equate defines the address of the ZCPR3 Message Buffer. This buffer is always 80 bytes long. If there is no ZCPR3 Message Buffer, this address should be 0.

```
Z3MSG   EQU     0F580H  ; ZCPR3 MESSAGE BUFFER
```

See Section 2.2.5 for more detail.

3.3.12. External FCB

The following equate defines the address of the ZCPR3 External FCB. This buffer is always 36 bytes long. If there is no ZCPR3 External FCB, this address should be 0.

```
EXTFCB  EQU     0F5D0H  ; ZCPR3 EXTERNAL FCB
```

See Section 2.2.4 for more detail.

3.3.13. Named Directory Buffer

The following equates define the address and size (in terms of 18-byte entries) of the ZCPR3 Named Directory Buffer. If there is no such buffer, both of these values should be 0.

```
Z3NDIR  EQU     0F600H  ; ZCPR3 NAMED DIRECTORY AREA
Z3NDIRS EQU     14      ; 14 18-byte Named Directory Elements permitted
                        ;   (NDIR SIZE = Z3NDIRS*18 + 1 for trailing 0)
```

See Section 2.2.3 for more detail.

3.3.14. Command Line Buffer

The following equates define the address and size (in terms of bytes) of the ZCPR3 Command Line Buffer (formerly called the Multiple Command Line Buffer under ZCPR2). If there is no such buffer, both of these values should be 0.

```
Z3CL    EQU     0F700H  ; ZCPR3 COMMAND LINE BUFFER
Z3CLS   EQU     200     ; SIZE OF COMMAND LINE BUFFER
```

See Section 2.2.2 for more detail.

3.3.15. External Stack

The following equate defines the address of the ZCPR3 External Stack. This stack is always 48 bytes in size. If there is no such stack, this value should be 0.

```
EXTSTK  EQU     0F7D0H  ; ZCPR3 EXTERNAL STACK
```

See Section 2.2.1 for more detail.

3.3.16. User Equates

The following equates are available for the implementer's target system. These are implementation-defined.

```
DJEPROM EQU     0F800H  ; EPROM BASE ADDRESS
```

This is provided mainly as a convenience to the user. This value is used by my BOOT and BIOS, which also read this file for information.

4. S T E P S   3 - 6 :   I N S T A L L A T I O N

4.1. Step 3: Modifying the BIOS Cold Boot Routine

The following is a reformatted and edited copy of my BIOS (Basic Input/Output System) and the header file used to customize it. This information is provided as an example of how to modify the Cold Boot routine (labelled 'cboot' in this example) for a full installation of ZCPR3. Only the pertinent information is included.

4.1.1. CBIOSHDR.LIB -- BIOS Configuration File

```
*****************************************************************
*                                                               *
*  Control Processor/Microcomputer Basic I/O System             *
*       CP/ZM CBIOSZ Standard with CON:=CRT:                    *
*                                                               *
*       Customized for the ARIES-1 Microcomputer by             *
*               Richard Conn, 5 Jan 1984                        *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
* The following revision number is in reference to the DR        *
* 2.8 CBIOS&.                                                   *
*                                                               *
*****************************************************************

revnum  equ     cbrev           ;CBIOSZ revision number
cpmrev  equ     z3rev           ;ZCPR3 revision number
```

The values 'cbrev' was provided by the BIOS source, and 'z3rev' came from Z3HDR.LIB.

<< Detail Left Out >>

```
cbdisk  equ     0F0H            ;Initial Disk to Log In, 0=A, 1=B
                                ;User 15, Disk A

*****************************************************************
*                                                               *
* CP/M system equates. If reconfiguration of the CP/M system    *
* is being done, the changes can be made to the following       *
* equates.                                                      *
*                                                               *
*****************************************************************

bdos    equ     ccp+800h        ;BDOS address
bios    equ     ccp+1600h       ;CBIOS address

wbot    equ     0               ;Warm boot jump address
iobyte  equ     3               ;IOBYTE location
cdisk   equ     4               ;Address of last logged disk
entry   equ     5               ;BDOS entry jump address
buff    equ     80h             ;Default buffer address
tpa     equ     100h            ;Transient memory

retries equ     10              ;Max retries on disk I/O before error

<< Detail Left Out >>

*****************************************************************
*                                                               *
*       Under the new redirectable I/O driver system, the I/O   *
* byte can be anything you desire.  I have set it up as follows:*
*                                                               *
*               -------------------------------                 *
*       IOBYTE  | LST:  | PUN: | RDR: | CON:  |                 *
*               -------------------------------                 *
*        bits ->  7 6 5    4      3     2 1 0                   *
*                                                               *
*       The initial IOBYTE is currently defined as:             *
*               Bits 0-2: CON                                   *
*               Bit 3: RDR                                      *
*               Bit 4: PUN                                      *
*               Bits 5-7: LST                                   *
*                                                               *
*               CON: = CRT: CRT                                 *
*               RDR: = CLOCK: System Clock                      *
*               PUN: = CLOCK: System Clock                      *
*               LST: = TTY: Printer                             *
*                                                               *
*****************************************************************

intioby equ     000$1$1$001B    ; Initial IOBYTE
```

This I/O Byte could also be initialized within the initialization sequence in the Input/Output Package. Whenever LDR.COM loads an I/O Package, it calls an initialization routine.

```
*****************************************************************
*                                                               *
* If there is a command inserted here, it will be given on cold *
* boot.                                                          *
*       For Example:                                             *
*                                                                *
*       coldbeg db      'MBASIC MYPROG'                          *
*       coldend db      0                                        *
*                                                                *
* will execute microsoft basic, and mbasic will execute the      *
* "MYPROG" basic program.                                        *
*                                                                *
******************************************************************

acmd    macro                   ;Define as Macro for Code Insertion
coldbeg:
        db      'STARTUP'       ;Cold boot command goes here
coldend:
        db      0
        endm
```

By ZCPR3 convention, STARTUP is a program created with the ALIAS utility. An Alias is a program created by the ZCPR3 ALIAS utility which generates command lines and places them into the Command Line Buffer of the ZCPR3 System. These command lines can be quite involved, including IF/ELSE constructs, and parameters may be passed into them at strategic points by an elaborate parameter passing mechanism. See the documentation (HLP file) on the ALIAS command for more details. The STARTUP Alias typically does not extract information from the command line and simply generates a sequence of commands which initialize the ZCPR3 System by running LDR on a variety of System Segments and performing other such operations.

```
******************************************************************
*                                                                *
*  Path to be Set for ZCPR2 on Cold Boot                         *
*                                                                *
******************************************************************

idisk1  equ     'A'-'@' ;1st: Disk A, Current User
iuser1  equ     '$'
idisk2  equ     'A'-'@' ;2nd: Disk A, User 15
iuser2  equ     15
idisk3  equ     0       ;No 3rd Entry
iuser3  equ     0
idisk4  equ     0       ;No 4th Entry
iuser4  equ     0
```

<< Detail on I/O Devices Left Out >>

4.1.2. CBIOSZ -- Selections from a ZCPR3 BIOS

```
*  SYSTEM SEGMENT:  CBIOSZ
*  SYSTEM:  ARIES-1
*  CUSTOMIZED BY:  RICHARD CONN

*---- Customize Section ----*
*  Customization Performed in CBIOSHDR.LIB
*---- End of Customize Section ----*

*****************************************************************
*                                                               *
*  Control Processor/Microcomputer Basic I/O System             *
*       CP/ZM BIOSZ Standard with CON:=CRT:                     *
*       CHBIOSZ Body                                            *
*                                                               *
*       Customized for the ARIES-1 Microcomputer with Hard Disk *
*               by Richard Conn, Feb 2, 1984                    *
*                                                               *
*****************************************************************


;
;  Macro Libraries for Customization
;
        MACLIB  Z3BASE
        MACLIB  CBIOSHDR

<< Detail Left Out >>

*****************************************************************
*                                                               *
* The following are internal Cbios equates. Most are misc.       *
* constants.                                                    *
*                                                               *
*****************************************************************

acr     equ     0dh             ;A carriage return
alf     equ     0ah             ;A line feed
XON     equ     11h             ;X-ON
XOFF    equ     13h             ;X-OFF
```

```
****************************************************************
*                                                              *
* The jump table below must remain in the same order, the        *
* routines may be changed, but the function executed must be    *
* the same.                                                    *
*                                                              *
****************************************************************

        org     bios            ;CBIOS starting address

        jmp     cboot           ;Cold boot entry point
wboote:
        jmp     wboot           ;Warm boot entry point
;
        jmp     const           ;Console status routine
        jmp     fconin          ;Console input
cout:
        jmp     fconout         ;Console output
        jmp     list            ;List device output
        jmp     punch           ;Punch device output
        jmp     reader          ;Reader device input
;
        jmp     home            ;Home drive
        jmp     setdrv          ;Select disk
        jmp     settrk          ;Set track
        jmp     setsec          ;Set sector
        jmp     setdma          ;Set DMA address
        jmp     read            ;Read the disk
        jmp     write           ;Write the disk
;
        jmp     listst          ;List device status
;
        jmp     sectran         ;Sector translation
;
        jmp     newio           ;Redirect I/O Drivers
```

My JMP table is extended slightly. NEWIO is for a system not described here.

```
****************************************************************
*                                                              *
*  These are the console I/O routines with buffer flush.       *
*                                                              *
****************************************************************

fconin:
        call    flush           ;Flush Buffer
        jmp     conin

fconout:
        push    b               ;Save char
        call    flush           ;Flush Buffer
        pop     b               ;Get char
        jmp     conout

****************************************************************
*                                                              *
* Gocpm is the entry point from cold boots, and warm boots. It  *
* initializes some of the locations in page 0, and sets up the  *
* initial DMA address (80h).                                     *
*                                                                *
*****************************************************************

gocpm:
        call    const           ;Check for Input Char
        ora     a               ;NZ means char there
        cnz     conin           ;Flush it if so
;
        lxi     h,buff          ;Set up initial DMA address
        call    setdma
;
        mvi     a,(jmp)         ;Initialize jump to warm boot
        sta     wbot
        sta     entry           ;Initialize jump to BDOS
;
        lxi     h,wboote        ;Address in warm boot jump
        shld    wbot+1
;
        lxi     h,bdos+6        ;Address in BDOS jump
        shld    entry+1
;
        xra     a               ;A = 0
        sta     bufsec          ;Disk Jockey buffer empty
        sta     bufwrtn         ;Set buffer not dirty flag
;
        lda     cdisk           ;Jump to CP/M with currently
                                   ;selected disk in C
        mov     c,a
```

This GOCPM section is more-or-less standard. The following section of GOCPM deals with initializing the Command Line Buffer on Cold Boot (or Warm Boot).

```
;
;  This code loads an optional command line on COLD BOOT only
;
        lda     cwflg           ;Test for any loaded command
        ora     a
        jnz     ccp+3           ;Enter ZCPR3 without command if Warm Boot
        lxi     d,coldbeg       ;Beginning of initial command
cldcmnd:
;
        if      z3cl ne 0       ;Multiple Commands Allowed?
;
        lxi     h,z3cl+4        ;Multiple Command buffer
;
        else
;
        lxi     h,ccp+8         ;Command buffer
;
        endif
;
cld1:
        ldax    d               ;Get char
        mov     m,a             ;Put char
        inx     h               ;Pt to next
        inx     d
        ora     a               ;Done?
        jrnz    cld1
        jmp     ccp             ;Run with Command

cwflg:
        db      0               ;Cold/warm boot flag

*****************************************************************
*                                                               *
* If there is a command inserted here, it will be given if the  *
* auto feature is enabled.                                      *
*       For Example:                                            *
*                                                               *
*       coldbeg db      'MBASIC MYPROG'                         *
*       coldend db      0                                       *
*                                                               *
* will execute microsoft basic, and mbasic will execute the     *
* "MYPROG" basic program.                                       *
*                                                               *
*****************************************************************

        acmd    ;Perform Macro from CBIOSHDR.LIB
```

```
*****************************************************************
*                                                               *
* Signon message output during cold boot.                       *
*                                                               *
*****************************************************************

prompt:
        db      acr,alf
        db      '0'+msize/10            ;CP/ZM memory size
        db      '0'+(msize mod 10)
        db      'K TPA ZCPR V'          ;ZCPR version number
        db      z3rev/10+'0','.',(z3rev mod 10)+'0'
        db      ', CBIOSZ V'            ;CBIOSZ version number
        db      cbrev/10+'0','.',(cbrev mod 10)+'0'

        if      first   ;if hard disk is A
        db      'H'     ;say this is a hard disk version
        else
        db      'F'     ;say this is a floppy version
        endif

        db      acr,alf
        db      0

*****************************************************************
*                                                               *
* Path for ZCPR 3.x initialized during cold boot.               *
*                                                               *
*****************************************************************
path:
        db      idisk1,iuser1   ;First Disk and User
        db      idisk2,iuser2   ;2nd Disk and User
        db      idisk3,iuser3   ;3rd Disk and User
        db      idisk4,iuser4   ;4th Disk and User

        db      0               ;End of PATH
```

This path is defined in CBIOSHDR.LIB.

```
*****************************************************************
*                                                               *
* Utility routine to output the message pointed at by H&L,      *
* terminated with a null.                                       *
*                                                               *
*****************************************************************

message:
        mov     a,m             ;Get a character of the message
        inx     h               ;Bump text pointer
        ana     a               ;Test for end
        rz                      ;Return if done
        push    h               ;Save pointer to text
        mov     c,a             ;Output character in C
        call    cout            ;Output the character
        pop     h               ;Restore the pointer
        jr      message         ;Continue until null reached


*****************************************************************
*                                                               *
* Cboot is the cold boot loader. All of CP/M has been loaded in *
* when control is passed here.                                  *
*                                                               *
*****************************************************************

cboot:
        lxi     sp,tpa          ;Set up stack
```

The following code segment copies the default command-search path into the External Path buffer. Since the ZCPR3 Command Processor uses this buffer the first time it searches for a COM file, it is recommended that this initialization always be done in the Cold Boot routine of the BIOS. There are some isolated cases where this is not required, but they will not be discussed here.

```
        if      expath ne 0     ;External Paths Supported
        lxi     d,path          ;Copy Cold-Boot Path
        lxi     h,expath        ;Into System External Path Area
        mvi     b,9             ;Always 9 bytes
        call    movlop
        endif
```

The following code segment initializes the Wheel Byte to non-priveleged status. This initialization may be done by a program executed by STARTUP if desired.

```
        if      z3whl ne 0      ;Wheel Byte Supported
        xra     a               ;Clear Wheel Byte
        sta     z3whl
        endif
```

The following code segment initializes the Resident Command Package buffer to zero. 128 bytes is larger than needed, but rather than specify the exact size for each package, this and the following initializations save code space in the BIOS and don’t waste a significant amount of time. Since the ZCPR3 Command Processor uses the RCP before any COM files are executed, initialization of the RCP buffer is required to be performed in the BIOS Cold Boot routine.

```
        if      rcp ne 0        ;RCPs Supported
        lxi     h,rcp           ;RCP Address (zero fill)
        call    zero128         ;128 bytes
        endif
```

The following code segment initializes the I/O Package with drivers which are contained within the BIOS. Later, when STARTUP executes, LDR will probably load a proper I/O Package over these drivers. Since I/O is used immediately after Cold Boot, this initialization must be performed in the Cold Boot routine.

```
        if      iop ne 0        ;IOPs Supported
        lxi     d,iodrivers     ;Set up I/O Drivers
        lxi     h,iop           ;Location for drivers
        call    mover           ;Copy an arbitrary 128 bytes
        else
        call    lstinit         ;Init Simple LST Device
        call    clkinit         ;Init Clock
        endif
```

The following code segment initializes the Flow Command Package. Like the RCP, the FCP must be initialized during Cold Boot.

```
        if      fcp ne 0        ;FCPs Supported
        lxi     h,fcp           ;FCP Address (zero fill)
        call    zero128         ;128 bytes
        endif
```

The following code segment initializes the ZCPR3 Environment Descriptor. Since the first utility executed is usually STARTUP (which is an Alias), the Environment Descriptor must be initialized during Cold Boot.

```
        if      z3env ne 0      ;ENVs Supported
        lxi     h,z3env         ;ENV Address (zero fill)
        mvi     b,128+16        ;128 bytes of environ + 16 bytes
        call    zerom              ;of TCAP
        endif
```

The following code segment clears the Shell Stack. The ZCPR3 Command Processor queries this buffer almost immediately after Cold Boot, so this initialization must be performed during Cold Boot.

```
        if      shstk ne 0      ;Shell Stack Supported
        xra     a               ;Clear Stack
        sta     shstk
        endif
```

The following initialization must also be performed during Cold Boot since the ZCPR3 Command Processor uses messages extensively if this feature is enabled.

```
        if      z3msg ne 0      ;ZCPR3 Messages Supported
        lxi     h,z3msg         ;Clear Message Bytes
        mvi     b,80            ;80 bytes
        call    zerom
        endif
```

The following initialization is not required during Cold Boot if the analysis of the DU form is performed before the DIR form by the ZCPR3 Command Processor, but the installer is taking a risk that all will be well until LDR loads the Named Directory buffer if he does not perform the following initialization during Cold Boot.

```
        if      z3ndir ne 0     ;Named Directory Based in Memory
        lxi     h,z3ndir        ;Named Directory Base
        call    zero128         ;128 bytes
        endif
```

The following initialization of the Command Line buffer is absolutely required during Cold Boot. Only if the Command Line buffer is NOT external is this initialization unnecessary, but then a significant amount of the power of the ZCPR3 System is lost by having an internal Command Line buffer.

```
        if      z3cl ne 0       ;Multiple Commands Allowed
        lxi     d,cmdset        ;Set buffers for Multiple Command
        lxi     h,z3cl          ;Command Line Base
        call    mover           ;Copy an arbitrary 128 bytes
        endif
```

As mentioned previously, the following initialization is not required and may be performed by the I/O Package loaded by LDR.

```
        if      iop ne 0
        mvi     a,intioby       ;Initialize the I/O Byte
        sta     iobyte
        endif
```

The following completes the Cold Boot initializations.

```
        lxi     h,prompt        ;Prep for sending signon message
        call    message         ;Send the prompt
        mvi     a,cbdisk        ;Select basic disk
        sta     cpmdrv
        sta     cdisk

<< Detail Left Out >>

        jmp     gocpm



*****************************************************************
*                                                               *
* Mover moves 128 bytes of data. Source pointer in DE, Dest     *
* pointer in HL.                                                *
*                                                               *
*****************************************************************

mover:
        mvi     b,128           ;Length of transfer
movlop:
        ldax    d               ;Get a byte of source
        mov     m,a             ;Move it
        inx     d               ;Bump pointers
        inx     h
        djnz    movlop          ;Continue moving until done
        ret

zerofl  set     (rcp ne 0)or(fcp ne 0)or(z3env ne 0)or(z3msg ne 0)
zerofl  set     zerofl or (z3ndir ne 0)
        if      zerofl
;
; Zero 128 bytes of memory pted to by HL
;
zero128:
        mvi     b,128           ;128 bytes
;
; Zero memory for B bytes; memory pted to by HL
;
zerom:
        mvi     m,0             ;store zero
        inx     h
        djnz    zerom
        ret
        endif
```

```
;
; Selection of LST: Device
;       If IOP is not supported, provide for one simple LST device;
; else, set LST device equal to console for later redefinition by
; loading an I/O Package via LDR
;
        if      iop ne 0
;
lstout  equ     djcout          ;same as console for now
punout  equ     djcout          ;same as console for now
rdrin   equ     djcin           ;same as console for now
;
        else
;
;  Initialize MPU Serial I/O Channel Characteristics and Baud Rate
;
lstinit:

<< Detail Left Out >>

;
        endif           ;IOP ne 0
;
```

```
*****************************************************************
*                                                               *
* Primitive I/O Drivers which are loaded at Cold Boot time.     *
*                                                               *
*****************************************************************
uart    equ     origin+3F9H     ;UART address
rda     equu    4               ;UART RDA Bit
iodrivers:
;
        if      iop ne 0
;
        jr      ioerror         ;no Status Routine
        db      0               ;Fill 3 bytes
        jr      ioerror         ;no Select Routine
        db      0               ;Fill 3 bytes
        jr      ioerror         ;no Namer Routine
        db      0               ;Fill 3 bytes
;
        endif           ;iop ne 0
;
        ret                     ;Initialize Terminal
        db      0,0             ;Fill 3 bytes
        jr      ustat           ;Console Input Status
        db      0               ;Fill 3 bytes
        jmp     djcin           ;Console Input Char
        jmp     djcout          ;Console Output Char

        jmp     lstout          ;List Output Char

        jmp     punout          ;Punch Output Char

        jmp     rdrin           ;Reader Input Char
```

Note:
     The above routines are located somewhere within the BIOS. They are very small and simple in nature, and they do not implement the I/O Byte. This is left for the redirectable I/O package which will be loaded later.

```
        mvi     a,0ffh          ;List Status Ready
        ora     a               ;Set Flags

        ret                     ;New I/O Driver Installation Routine

ioerror:
        xra     a               ;No device assignments
        ret
```

```
*****************************************************************
*                                                               *
*  The following equates define the various Redirectable I/O    *
*  routines in the SYSTEM I/O Area.                             *
*                                                               *
*****************************************************************

;
        if      iop ne 0
riobase equ     iop+9           ;Relative I/O Base (less support)
        else
riobase equ     iodrivers       ;Simple I/O Drivers
        endif           ;iop ne 0
;
tinit   equ     riobase         ;Terminal Init
const   equ     riobase+3       ;Console Input Status
conin   equ     riobase+6       ;Console Input
conout  equ     riobase+9       ;Console Output
list    equ     riobase+12      ;List Output
punch   equ     riobase+15      ;Punch Output
reader  equ     riobase+18      ;Reader Input
listst  equ     riobase+21      ;List Output Status
newio   equ     riobase+24      ;Redirectable I/O Patcher

*****************************************************************
*                                                               *
* Initial Values for the External Command Line Buffers          *
*    and Named Directory Memory-Based Buffers                   *
*                                                               *
*****************************************************************
        if      z3cl ne 0
cmdset:
        dw      z3cl+4          ;Beginning of I/O Buffer
        db      z3cls           ;Size of I/O Buffer
        db      0               ;Empty Buffer
        db      0               ;Empty Buffer
        endif

*****************************************************************
*                                                               *
* Wboot loads in all of CP/M except the CBIOS, then initializes *
* system parameters as in cold boot. See the Cold Boot Loader   *
* listing for exactly what happens during warm and cold boots.  *
*                                                               *
*****************************************************************

wboot:

<< Detail Left Out >>
```

4.2. Step 4: Editing Z3HDR.LIB

The following is a reformatted duplicate of the body of the Z3HDR.LIB file. It is provided here to present additional information on how to set the equates. It may be useful to the installer to have this installation manual open to these pages while he is editing the Z3HDR.LIB file.

The following is the Banner for Z3HDR.LIB:

```
Z3HDR - Maximum Configuration
Offset:  5100H
```

This offset is a note to the installer. It indicates the value to add to the R command of DDT in order to properly read in the HEX file of the ZCPR3 Command Processor into the Operating System memory image.

```
Module:  Z3HDR
Author:  Richard Conn
Module Used By:  ZCPR3 Version 3.x
```

Note: Z3HDR contains the key customization equates for ZCPR3. These equates allow the user to select various ZCPR3 options and do an extensive amount of tailoring of ZCPR3 to the user's desires.

4.2.1. Basic System Definitions

The following equates may be used to customize this CPR for the user's system and integration technique.

```
    REL - TRUE if integration is to be done via MOVCPM
        - FALSE if integration is to be done via DDT and SYSGEN

    CPRLOC - Base Page Address of CPR; this value can be obtained
          by running the CCPLOC program on your system, and if REL
          is FALSE, this value is supplied through the Z3BASE.LIB
          CCP equate

REL     EQU     FALSE
        IF      REL
CPRLOC  EQU     0
        ELSE
CPRLOC  EQU     CCP     ;VALUE PROVIDED IN Z3BASE.LIB
        ENDIF
```

CCPLOC.COM was an earlier version of Z3LOC.COM, which is now supplied with the ZCPR3 distribution. You may use either program to determine the base page of the CPR (Command Processor Replacement).

Integration by MOVCPM is not addressed here.

4.2.2. Default File Types

The following macros define the file types of the command object files (COM files under CP/M 2.2) to be loaded when a non-resident ZCPR3 command is given and of the indirect command files (SUB files under CP/M 2.2) to be used to extract commands from when the indirect command facility is invoked.

```
COMTYP  MACRO
        DB      'COM'
        ENDM

SUBTYP  MACRO
        DB      'SUB'
        ENDM
```

These equates are provided to allow the installer to select any file type he wishes for object and indirect command files. The indicated values of 'COM' and 'SUB' are conventional.

4.2.3. SUBMIT File Processing

The following flag enables the ability of ZCPR3 to process SUBMIT files (command files of the form $$$.SUB). If SUBON is TRUE, then ZCPR3 will process such files like CP/M's CCP normally does; if SUBON is FALSE, ZCPR3 will not process such files (ignore them). In such a case, only indirect command file facilities like ZEX will work. Much code is saved inside of the ZCPR3 Command Processor if SUBON is set to FALSE, but this rather useful facility is lost.

```
SUBON   EQU     TRUE
```

4.2.4. Command Prefix

The following flag allows ZCPR3 to accept commands of the form "du:command params" or "dir:command params". If DRVPREFIX is TRUE, this form is accepted; if FALSE, this form is not accepted.

```
DRVPREFIX       equ     TRUE
```

This option also affects arguments to commands, such as "DIR A5:*.TXT", and DU or DIR prefixes on these arguments are not processed either if DRVPREVIX is FALSE.

4.2.5. Command Attributes

The following equate allows the user to select the attributes of the COM files which are selected for execution. The ZCPR3 Command Processor can be made to execute only COM files with the System attribute set, with the Directory (non-System) attribute set, or with either attribute set. The following values are defined for this equate:

COMATT	Files Selected
0	System
80H	Directory
1	Both System and Directory

```
COMATT  equ     01H
```

4.2.6. ZCPR3 Resident Command Activation and Wheels

The following equates enable various ZCPR3-resident commands. The user may invoke these as desired, but should keep in mind the size of the resulting ZCPR3 and make sure it does not exceed the required limits.

```
DIRON   equ     FALSE   ;DIR COMMAND
LTON    equ     FALSE   ;LIST, TYPE COMMANDS
GOON    equ     TRUE     ;GO COMMAND
ERAON   equ     FALSE   ;ERA COMMAND
SAVEON  equ     TRUE     ;SAVE COMMAND
RENON   equ     FALSE   ;REN COMMAND
GETON   equ     TRUE     ;GET COMMAND
JUMPON  equ     FALSE   ;JUMP COMMAND
NOTEON  equ     FALSE   ;NOTE COMMAND
```

Most of these commands are available as options in the Resident Command Packages. If selected for incorporation in the Resident Command Packages instead of the ZCPR3 Command Processor, space is freed in the ZCPR3 CPR and the functionality of these commands can be extended easily (eg, ERA in an RCP can have an Inspect option).

The Wheel equate table enables the WHEEL facility of ZCPR3. With this facility, a WHEEL BYTE, which exists somewhere in memory, is examined before a set of installer-selected commands are executed. If this byte is not zero, then the command proceeds. If it is zero, then the command is not allowed to proceed and is exited with an error message.

The following set of equates make each of the indicated commands selectable to respond to the Wheel Byte or not. For instance, if WERA=TRUE, then it responds to the Wheel Byte; if WERA=FALSE, it does not.

```
        IF      Z3WHL NE 0      ;IF A WHEEL BYTE ADDRESS IS DEFINED
WERA    equ     FALSE   ;Make ERA a Wheel-Oriented Command
WREN    equ     FALSE   ; "   REN "   "       "       "
WLT     equ     FALSE   ; "   L/T "   "       "       "  (LIST/TYPE)
WGO     equ     FALSE   ; "   GO  "   "       "       "
WSAVE   equ     FALSE   ; "   SAVE "  "       "       "
WGET    equ     FALSE   ; "   GET "   "       "       "
WJUMP   equ     FALSE   ; "   JUMP "  "       "       "
WDU     equ     FALSE   ; "   DU: "   "       "       " (DU/DIR Change)
WHEEL   equ     WERA OR WREN OR WLT OR WGO OR WSAVE OR WGET OR WJUMP OR WDU
        ENDIF           ;Z3WHL
```

The commands inside of the Resident Command Package can also be set to respond to the Wheel Byte.

4.2.7. ZCPR3 Resident Command Table

This table consists of the names of the various ZCPR3-resident commands and their addresses. The NCHARS equate defines how many characters long each name may be, and all table entries must be exactly the indicated number of characters (trailing spaces are used to fill out shorter names).

Each table entry is structured as follows:

```
                DB      'CMND'  ;Name of Command (NCHARS long)
                DB      CMNDADR ;Address of Command within ZCPR3
```

The installer should only change the names of the commands as desired and should not, as a rule, touch the address definition since this is fixed within the body of ZCPR3.

```
NCHARS  EQU     4               ;NUMBER OF CHARS/COMMAND

CTABLE  MACRO
;
        IF      DIRON
        DB      'DIR '
        DW      DIR             ;DIRECTORY DISPLAY COMMAND
        ENDIF
;
        IF      LTON
        DB      'LIST'
        DW      LIST            ;LIST FILE ON PRINTER COMMAND
        DB      'TYPE'
        DW      TYPE            ;TYPE FILE ON CONSOLE COMMAND
        ENDIF
;
        IF      GOON
        DB      'GO  '
        DW      GO              ;EXECUTE CURRENT TPA COMMAND
        ENDIF
;
        IF      ERAON
        DB      'ERA '
        DW      ERA             ;ERASE FILES COMMAND
        ENDIF
;
        IF      SAVEON
        DB      'SAVE'
        DW      SAVE            ;SAVE TPA COMMAND
        ENDIF
;
        IF      RENON
        DB      'REN '
        DW      REN             ;RENAME FILES COMMAND
        ENDIF
;
        IF      GETON
        DB      'GET '
        DW      GET             ;LOAD FILE INTO TPA COMMAND
        ENDIF
;
        IF      JUMPON
        DB      'JUMP'
        DW      JUMP            ;JUMP TO ANY MEMORY LOCATION COMMAND
        ENDIF
;
        IF      NOTEON
        DB      'NOTE'
        DW      NOTE            ;NOTE - NULL COMMAND (NOP)
        ENDIF
;
        ENDM
```

FIG 4-1: ZCPR3 Resident Command Naming

4.2.8. Controls on ZCPR3 Resident Commands

The following sets of equates provide special controls and parameters on various ZCPR3-resident commands.

The following equates set the width of the spacing between the file names for the DIR command and the character used to separate file names from one another on the same line.

Assuming that FENCE is set to the character '|', If WIDE is TRUE, then the output will look like:

```
                filename.typ__|__filename.typ ...
```

while if WIDE is FALSE, the output will look like:

```
                filename.typ_|_filename.typ ...
```

(underscore represents a space)

```
WIDE    EQU     TRUE
FENCE   EQU     '|'
```

The WIDE equate is intended to provide for shorter lines for those users with 64-column displays. For those with 80-column displays, the output lines are much easier to read.

The following equates define two flags which are used in conjunction with the DIR command on the command line. SYSFLG is the character used to indicate to DIR that all files, both System and Non-System, are to be displayed. SOFLG is the character used to indicate to DIR that only the System files are to be displayed. By default, DIR displays non-System files.

For example, if SYSFLG is set to 'A' and SOFLG is set to 'S', then:

```
                DIR *.COM A
```

displays all COM files with both System and non-System attributes while:

```
                DIR *.COM S
```

displays only COM files with the System attribute. Naturally:

```
                DIR *.COM
```

displays only COM files with the non-System attribute.

```
SYSFLG  EQU     'A'
SOFLG   EQU     'S'
```

The following equate causes ERA to confirm the files to be erased before it goes ahead and erases them. If ERAOK is TRUE, then the user will be prompted each time; if it is FALSE, then the user will not be prompted.

```
ERAOK   equ     FALSE
```

If ERAOK is TRUE, the following equate adds a Verify option to the ERA command which causes the user to be prompted only if the Verify option letter, defined by ERDFLG, is given after the file name. If ERAV is TRUE, then the user will be asked to verify only when ERDFLG is contained in the command line; if ERAV is FALSE, the user will always be asked to verify.

For example, if ERAOK is TRUE, ERAV is TRUE, and ERDFLG is 'V', then the command:

```
                        ERA *.* V
```

will result in the file names being displayed and the user being asked for verification. If the V option were not given, the user would not be asked for verification.

```
ERAV    equ     FALSE
ERDFLG  equ     'V'
```

The following equates set the paging parameters for the TYPE command.

PGDFLT determines if TYPE pages by default. If PGDFLT is TRUE, then:

```
                TYPE FILE.TXT
```

will be paged. If PGDFLT is FALSE, the above command will not be paged.

PGDFLG defines the option character in the TYPE command line which is used to toggle the default set by PGDFLT. Assuming that PGDFLG is set to 'P', then:

```
                TYPE FILE.TXT P
```

will page the file listing if PGDFLT is FALSE and not page it if PGDFLT is TRUE.

```
PGDFLT  EQU     TRUE
PGDFLG  EQU     'P'
```

The following equate defines the number of lines on the user’s CRT screen for use by the TYPE command when it is paging. This value is usually 24.

```
NLINES  EQU     24
```

The following equate defines the option letter used with the SAVE command to indicate that the associated number is 128-byte sectors as opposed to 256-byte pages. For example, if SECTFLG is set to ’S’, then:

```
                SAVE 25 FILE.BIN S
```

save 25 128-byte sectors starting at location 100H into the file named FILE.BIN. IF the S option was not present, SAVE would have saved 25 256-byte blocks starting at location 100H into the file named FILE.BIN.

```
SECTFLG EQU     ’S’
```

4.2.9. Path Definition

The following equate specifies the address of the PATH to be followed for the PATH command-search if the PATH is to be initialized by the BIOS and set by the user via a PATH.COM program. The value of PATH should be the address of the PATH data area in memory. If the internal PATH provided by ZCPR3 is to be used, then PATHBASE should be equated to 0, which selects the PATH located just after the MEMLOAD routine. If the external PATH is to be used, then PATHBASE should be set to the address of the external path.

A PATH is a series of byte-pairs, terminated by a binary 0. The first byte of each pair is the disk number (1-16 for disks A-P), and the second byte of each pair is the user number (0-31). The special character ’$’ indicates the current user or current disk. For example, the path from current disk/current user to current disk/user 0 to disk A/user 0 is selected by the following sequence:

```
                DB      ’$$’    ;current disk/user
                DB      ’$’,0   ;current disk/user 0
                DB      1,0     ;disk A/user 0
                DB      0       ;end of path
```

```
        IF      EXPATH NE 0     ;External Path Selected
```

This equate defines the base address of the external path

```
PATH    equ     EXPATH          ;External ZCPR3 PATH

        ELSE                    ;Internal Path Selected
```

The following macro defines the n-element internal path

```
IPATH   MACRO
        db      'A'-'@','$'     ;Disk A, Current User
        db      'A'-'@',0       ;Disk A, User 0
        db      0               ;End of Path -- MUST be here
        ENDM

        ENDIF
```

The following flag enables ZCPR3 to perform an optimized path search when it is searching along a path for a file. If this equate is TRUE, ZCPR3 will build a path in memory of absolute entries (A1, B7, etc) from the symbolic path (one containing '$') which is the path it would otherwise use. This new path would contain no duplicate path elements, where a symbolic path analysis may. For example, if the path is:

```
                db      'A'-'@','$'     ;disk A, current user
                db      'A'-'@',15      ;disk A, user 15
                db      0
```

then if the user is logged into A15, setting the below equate to TRUE would allow ZCPR3 to build the path:

```
                db      'A'-'@',15      ;only one entry
                db      0
```

in the analysis of this symbolic path, while with this equate FALSE, ZCPR3 may log into A15 as many as three times (once for the default and twice more for the symbolic path) in looking for a file which is not found before it gives up. Using this minimum path facility costs some code in ZCPR3, but it speeds up processing noticably in some cases.

Enable this equate if MINIMUM PATH SEARCH is to be employed.

```
MINPATH EQU     TRUE
```

In searching for a file along a path, ZCPR3 can be commanded to always look in the current logged-in directory before beginning the path search. This equate controls this feature. If SCANCUR is set to TRUE, the current directory need never be referenced in a symbolic path expression (DB '$','$') since SCANCUR insures that the current directory is scanned.

Enable this equate if the current DU is always to be scanned.

```
SCANCUR EQU     TRUE
```

4.2.10. DU and DIR Controls

The following equate enables the appearance of the current disk/user in the ZCPR3 prompt. If set to FALSE, the prompt appears as '>' (assuming > is the current value of CPRMPT). If set to TRUE, the prompt appears as 'd>' or 'dn>'. (see INCLNDR below)

```
INCLDU  equ     TRUE
```

The following equate allows ZCPR3 to accept the DU: prefix or login form for input. Set this to TRUE if DU: prefix is to be allowed.

Setting this equate to TRUE allows the following forms:

```
                A>B1:
                A>TYPE B4:FILE.TXT
                A>B:
                A>1:
```

```
ACCPTDU EQU     TRUE
```

This equate enables ZCPR3 to process DIR: forms internally through the memory-based named directory buffer. This equate and the NDBASE address should be TRUE (non-zero) in order to enable ZCPR3 to process named directories.

If NDINCP is TRUE, the following forms are allowed:

```
                A>ROOT:
                A>TYPE TEXT:FILE.TXT
```

if the other associated equates (below) are set correctly.

```
NDINCP  EQU     TRUE
```

The following equate will cause the name of the current directory to be displayed as part of the prompt along with the DU form if enabled (see INCLDU above).

For example, if INCLNDR is TRUE, the prompt would look like:

```
                B7:TEXT>          -- if INCLDU is also TRUE
                TEXT>             -- if INCLDU is FALSE

INCLNDR EQU     TRUE
```

The following equate allows ZCPR3 to accept the DIR: prefix or login form for input. Set this to TRUE if DIR: prefix is to be allowed.

Setting this equate to TRUE allows the following forms:

```
                A>ROOT:
                A>TYPE TEXT:FILE.TXT

ACCPTND EQU     TRUE
```

The following equate determines the hierarchy of DU:/DIR: evaluation. Set this to TRUE if DU: is to be tested for before DIR: or set this to FALSE if DIR: is to be tested for before DU:. If this is FALSE, named directories like C: (standing for C work area - NOT disk C) are permitted.

Assuming that a directory for C programs, named 'C', and a root directory, named 'ROOT', exist, then if DUFIRST is set to FALSE:

```
        A>C:    -- logs the user into the directory named 'C'
        A>ROOT: -- logs the user into the directory named 'ROOT'
```

while if DUFIRST is set to TRUE:

```
        A>C:    -- logs the user into disk C:
                    (dir C can't be accessed)
        A>ROOT: -- logs the user into the directory named 'ROOT'

DUFIRST EQU     FALSE
```

Enable password check on named directory references. If a named directory is referenced and has a password associated with it, ZCPR3 will ask the user for this password and approve the reference only if he gives a valid response. One and only one try is permitted. Setting this equate to TRUE will enable the password check facility.

```
PWCHECK EQU     TRUE
```

4.2.11. Command Line Buffer Control

The MULTCMD equate enables the feature of having more than one command on the same line, separated by a separation char which is defined by the CMDSEP equate. If this feature is enabled, the command line buffer and buffer pointers are moved outside of ZCPR3 at the indicated address of Z3CL.

MULTCMD indicates if the ability to have more than one command on a line is to be enabled, and CMDSEP is the character used to separate these commands. For example, if CMDSEP is ';' and MULTCMD is TRUE, then commands like this are possible:

```
                ERA *.BAK;DIR
```

```
        IF      Z3CL NE 0
MULTCMD equ     TRUE
        ELSE
MULTCMD equ     FALSE
        ENDIF
CMDSEP  equ     ';'
```

4.2.12. CMDRUN -- ZCPR3 Extended Command Processing

This equate enables the ZCPR3 CMDRUN facility. If CMDRUN is TRUE, then another stage of command processing is invoked should ZCPR3 fail to find a COM file when the user gives a command. This stage involves invoking the COM file specified by CMDFCB and giving it the current command line as an argument. In this way, if, say, M80 PROG2 fails as a command, a new command like LRUNZ M80 PROG2, SUB M80 PROG2, or ZEX M80 PROG2 may be processed. If the new command fails, an appropriate error message is given.

The ROOTONLY option causes ZCPR3 to only look at the Root (bottom of path) for the Extended Command Processor if it is set to TRUE. If it is set to FALSE, the path is searched for the Extended Command Processor. The tradeoff here is that ROOTONLY = TRUE is less flexible but somewhat faster than ROOTONLY = FALSE.

```
CMDRUN  equ     FALSE   ; Enable the Facility

        if      CMDRUN
ROOTONLY        equ     TRUE     ; TRUE if look at Root Only for Extended
                                ; Command Processor, FALSE if look along
                                ; path
CMDFCB  MACRO
        db      0
        db      'CMDRUN  '      ;Name of Program
        db      'COM'           ;File Type
        ENDM
        endif   ;CMDRUN
```

4.2.13. Flow Command Facility

This equate enables ZCPR3 to respond to IF processing. ZCPR3 simply flushes commands if a FALSE IF is currently engaged. FCPs must be enabled for IFON to work correctly.

```
IFON    EQU     TRUE
```

4.2.14. Miscellaneous Equates

```
MAXUSR  EQU     31                ;MAXIMUM USER NUMBER ACCESSABLE
MAXDISK EQU     4                ;MAXIMUM NUMBER OF DISKS ACCESSABLE
```

The DU form will only be allowed to reference disks and user areas up to the indicated maximum values. These limits, however, do not apply to named directories. For instance, if MAXUSR was 20 and the name SPECIAL was equated to B31, then 'SPECIAL:' would be resolved correctly but 'B31:' would cause an error. Named directories can have password protection associated with them, so B31 can be a directory which is accessed only if the user knows the password for it.

```
SUPRES  EQU     TRUE              ;SUPRESSES USER # REPORT FOR USER 0
```

If you are logged into B0:, then the prompt would look something like:

```
          B>        if SUPRES is TRUE
          B0>       if SUPRES is FALSE
```

```
SPRMPT  EQU     '$'              ;CPR PROMPT INDICATING SUBMIT COMMAND
CPRMPT  EQU     '>'              ;CPR PROMPT INDICATING USER COMMAND
```

With these values:

```
          B1>       appears for commands typed by the user
          B1$       appears for commands input from a submit file
```

```
NUMBASE EQU     'H'              ;CHAR USED TO SWITCH FROM DEFAULT
                                    ;NUMBER BASE
```

This option applies to the SAVE command only. It permits forms like:

```
          SAVE 17 myfile.bin       17 is decimal
          SAVE 11H myfile.bin      11H is hexadecimal
```

This feature is handy in that values output by tools like DDT do not need to be converted to hexadecimal before the SAVE command can be used.

```
CURIND  EQU     '$'              ;SYMBOL FOR CURRENT DISK OR USER

COMMENT EQU     ';'              ;LINES BEGINNING WITH THIS CHAR
                                    ;ARE COMMENTS
```

4.3. Step 5: Overlaying the old BIOS and the CCP

The procedure for overlaying the old BIOS is the same as with conventional CP/M. The relative offset for the ZCPR3 Command Processor Replacement will be the same as that used for the BIOS. Refer to the manual published by Digital Research for more detail.

An example of this procedure is provided in the sample session which follows.

4.4. Step 6: Implanting the Operating System Image

Once everything is overlayed in the memory image of the system, SYSGEN is usually used to implant the operating system image on the system tracks of the disk. Refer to the manual published by Digital Research for more detail.

An example of this procedure is provided in the sample session which follows.

4.5. Sample Session

The following directory display shows the files we will be working with. Note that Z3BASE.LIB and Z3HDR.LIB are instrumental to the installation of almost all of the System Segments -- Z3BASE.LIB and Z3HDR.LIB are read in by the MAC assembler during the assembly process.

```
B11>xd /oa
XD III  Version 1.2
Filename.Typ Size K  Filename.Typ Size K  Filename.Typ Size K
-------- --- ------  -------- --- ------  -------- --- ------
CBIOSZ  .ASM     56  SYSRCP  .ASM     44  SYSNDR  .LIB      4
SYSENV  .ASM      4  ZCPR3   .ASM     68  SYSRCP  .LIB     12
SYSFCP  .ASM     20  CBIOSHDR.LIB     12  Z3BASE  .LIB     12
SYSIOP  .ASM     32  SYSENV  .LIB      4  Z3HDR   .LIB     20
SYSNDR  .ASM      4  SYSFCP  .LIB      8
     B 11:  --   14 Files Using   300K ( 2328K Left)
```

4.5.1. Assembling SYSENV

The following illustrates the assembly of SYSENV to produce the SYS.ENV file. This file will be the ZCPR3 System Environment Descriptor.

```
B11>zex mac sysenv
ZEX, Version 3.0
B11> ZEX: ;
B11> ZEX: ;  MAC -- CP/M Standard MACRO Assembler and Loader
B11> ZEX: ;
B11> ZEX: ;     Suppress FALSE IF Printout
B11> ZEX: ;
B11> ZEX: IF NUL SYSENV
B11> ZEX: MAC SYSENV $-S PZ
CP/M MACRO ASSEM 2.0
0200
01AH USE FACTOR
END OF ASSEMBLY

B11> ZEX: IF INPUT Abort if Errors Exist
IF True?
B11> ZEX: ERA SYSENV.BAK  ;NOTE Cleanup
 No Files

B11> ZEX: ERA SYSENV.COM
 No Files
B11> ZEX: MLOAD SYSENV    ;NOTE Load Hex File
MLOAD ver. 1.4   Copyright (C) 1983 Ronald G. Fowler
Loaded 172 bytes (00ACH - 2 records) to file B:SYSENV.COM
Start address: 0100H  Ending address: 01ADH  Bias: 0000H


B11> ZEX: FI
B11> ZEX: ERA SYSENV.HEX
 SYSENV  .HEX
B11> ZEX: FI
B11> ZEX: ;
B11> ZEX: ;  Assembly Complete
B11> ZEX: ;
B11> ZEX: Done>
B11>ren sys.env=sysenv.com
```

4.5.2. Assembling SYSNDR

```
B11>zex mac sysndr
ZEX, Version 3.0
B11> ZEX: ;
B11> ZEX: ;  MAC -- CP/M Standard MACRO Assembler and Loader
B11> ZEX: ;
B11> ZEX: ;     Suppress FALSE IF Printout
B11> ZEX: ;
B11> ZEX: IF NUL SYSNDR
B11> ZEX: MAC SYSNDR $-S PZ
CP/M MACRO ASSEM 2.0
01EB
005H USE FACTOR
END OF ASSEMBLY

B11> ZEX: IF INPUT Abort if Errors Exist
IF True?
B11> ZEX: ERA SYSNDR.BAK  ;NOTE Cleanup
 No Files

B11> ZEX: ERA SYSNDR.COM
 No Files
B11> ZEX: MLOAD SYSNDR    ;NOTE Load Hex File
MLOAD ver. 1.4   Copyright (C) 1983 Ronald G. Fowler
Loaded 235 bytes (00EBH - 2 records) to file B:SYSNDR.COM
Start address: 0100H  Ending address: 01EAH  Bias: 0000H


B11> ZEX: FI
B11> ZEX: ERA SYSNDR.HEX
 SYSNDR  .HEX
B11> ZEX: FI
B11> ZEX: ;
B11> ZEX: ;  Assembly Complete
B11> ZEX: ;
B11> ZEX: Done>
B11>ren sys.ndr=sysndr.com
```

4.5.3. Assembling SYSIOP

```
B11>zex mac sysiop
ZEX, Version 3.0
B11> ZEX: ;
B11> ZEX: ;  MAC -- CP/M Standard MACRO Assembler and Loader
B11> ZEX: ;
B11> ZEX: ;     Suppress FALSE IF Printout
B11> ZEX: ;
B11> ZEX: IF NUL SYSIOP
B11> ZEX: MAC SYSIOP $-S PZ
CP/M MACRO ASSEM 2.0
F0AA
016H USE FACTOR
END OF ASSEMBLY

B11> ZEX: IF INPUT Abort if Errors Exist
IF True?
B11> ZEX: ERA SYSIOP.BAK  ;NOTE Cleanup
 No Files

B11> ZEX: ERA SYSIOP.COM
 No Files
B11> ZEX: MLOAD SYSIOP    ;NOTE Load Hex File
MLOAD ver. 1.4   Copyright (C) 1983 Ronald G. Fowler
Loaded 1192 bytes (04A8H - 10 records) to file B:SYSIOP.COM
Start address: EC00H  Ending address: F0A7H  Bias: 0000H

++ Warning: program origin NOT at 100H ++


B11> ZEX: FI
B11> ZEX: ERA SYSIOP.HEX
 SYSIOP  .HEX
B11> ZEX: FI
B11> ZEX: ;
B11> ZEX: ;  Assembly Complete
B11> ZEX: ;
B11> ZEX: Done>
B11>ren sys.iop=sysiop.com
```

4.5.4. Assembling SYSRCP

```
B11>zex mac sysrcp
ZEX, Version 3.0
B11> ZEX: ;
B11> ZEX: ;  MAC -- CP/M Standard MACRO Assembler and Loader
B11> ZEX: ;
B11> ZEX: ;     Suppress FALSE IF Printout
B11> ZEX: ;
B11> ZEX: IF NUL SYSRCP
B11> ZEX: MAC SYSRCP $-S PZ
CP/M MACRO ASSEM 2.0
EBF1
017H USE FACTOR
END OF ASSEMBLY

B11> ZEX: IF INPUT Abort if Errors Exist
IF True?
B11> ZEX: ERA SYSRCP.BAK  ;NOTE Cleanup
 No Files

B11> ZEX: ERA SYSRCP.COM
 No Files
B11> ZEX: MLOAD SYSRCP    ;NOTE Load Hex File
MLOAD ver. 1.4   Copyright (C) 1983 Ronald G. Fowler
Loaded 2027 bytes (07EBH - 16 records) to file B:SYSRCP.COM
Start address: E400H  Ending address: EBEEH  Bias: 0000H

++ Warning: program origin NOT at 100H ++


B11> ZEX: FI
B11> ZEX: ERA SYSRCP.HEX
 SYSRCP  .HEX
B11> ZEX: FI
B11> ZEX: ;
B11> ZEX: ;  Assembly Complete
B11> ZEX: ;
B11> ZEX: Done>
B11>ren sys.rcp=sysrcp.com
```

4.5.5. Assembling SYSFCP

```
B11>zex mac sysfcp
ZEX, Version 3.0
B11> ZEX: ;
B11> ZEX: ;  MAC -- CP/M Standard MACRO Assembler and Loader
B11> ZEX: ;
B11> ZEX: ;     Suppress FALSE IF Printout
B11> ZEX: ;
B11> ZEX: IF NUL SYSFCP
B11> ZEX: MAC SYSFCP $-S PZ
CP/M MACRO ASSEM 2.0
F3F3
00CH USE FACTOR
END OF ASSEMBLY

B11> ZEX: IF INPUT Abort if Errors Exist
IF True?
B11> ZEX: ERA SYSFCP.BAK  ;NOTE Cleanup
 No Files

B11> ZEX: ERA SYSFCP.COM
 No Files
B11> ZEX: MLOAD SYSFCP    ;NOTE Load Hex File
MLOAD ver. 1.4   Copyright (C) 1983 Ronald G. Fowler
Loaded 499 bytes (01F3H - 4 records) to file B:SYSFCP.COM
Start address: F200H  Ending address: F3F2H  Bias: 0000H

++ Warning: program origin NOT at 100H ++


B11> ZEX: FI
B11> ZEX: ERA SYSFCP.HEX
 SYSFCP  .HEX
B11> ZEX: FI
B11> ZEX: ;
B11> ZEX: ;  Assembly Complete
B11> ZEX: ;
B11> ZEX: Done>
B11>ren sys.fcp=sysfcp.com
```

4.5.6. Creating MYTERM.Z3T via TCSELECT

```
B11>tcselect myterm
TCSELECT, Version 1.0

** Terminal Menu 1 for Z3TCAP Version 1.1  **

A.  AA Ambassador           K.  Concept 100
B.  ADDS Consul 980         L.  Concept 108
C.  ADDS Regent 20          M.  CT82
D.  ADDS Viewpoint          N.  DEC VT52
E.  ADM 2                   O.  DEC VT100
F.  ADM 31                  P.  Dialogue 80
G.  ADM 3A                  Q.  Direct 800/A
H.  ADM 42                  R.  General Trm 100A
I.  Bantam 550              S.  Hazeltine 1420
J.  CDC 456                 T.  Hazeltine 1500

Enter Selection, + for Next, or ^C to Exit - +

** Terminal Menu 2 for Z3TCAP Version 1.1  **

A.  Hazeltine 1510          K.  P Elmer 1200
B.  Hazeltine 1520          L.  SOROC 120
C.  H19 (ANSI Mode)         M.  Super Bee
D.  H19 (Heath Mode)        N.  TAB 132
E.  HP 2621                 O.  Teleray 1061
F.  IBM 3101                P.  Teleray 3800
G.  Micro Bee               Q.  TTY 4424
H.  Microterm ACT IV        R.  TVI 912
I.  Microterm ACT V         S.  TVI 920
J.  P Elmer 1100            T.  TVI 950

Enter Selection, - for Last, + for Next, or ^C to Exit - T

  Selected Terminal is: TVI 950                  -- Confirm (Y/N)? Y

File MYTERM  .Z3T Created
```

4.5.7. Recap

To recap, the main System Segments which will be loaded by the LDR utility have now been created. These System Segments are:

```
B11>xd sys.* oa
XD III  Version 1.2
Filename.Typ Size K  Filename.Typ Size K  Filename.Typ Size K
-------- --- ------  -------- --- ------  -------- --- ------
SYS     .ENV      4  SYS     .IOP      4  SYS     .RCP      4
SYS     .FCP      4  SYS     .NDR      4
     B 11:  --    5 Files Using    20K ( 2304K Left)

B11>xd myterm.z3t oa
XD III  Version 1.2
Filename.Typ Size K  Filename.Typ Size K  Filename.Typ Size K
-------- --- ------  -------- --- ------  -------- --- ------
MYTERM  .Z3T      4
     B 11:  -     1 Files Using     4K ( 2304K Left)
```

4.5.8. Assembling the CBIOS

The following illustrates the assembly of the Customized Basic Input/Output System (CBIOS). CBIOS has already been customized so that its Cold Boot routine performs the proper initializations of buffers required by the ZCPR3 System.

```
B11>mac cbiosz $-s pz
CP/M MACRO ASSEM 2.0
E3CE
01FH USE FACTOR
END OF ASSEMBLY
```

4.5.9. Assembling the ZCPR3 Command Processor Replacement

The following illustrates the assembly of the ZCPR3 Command Processor Replacement. All customization has been done in the files Z3BASE.LIB and Z3HDR.LIB.

```
B11>mac zcpr3 $-s pz
CP/M MACRO ASSEM 2.0
C7E9
01EH USE FACTOR
END OF ASSEMBLY
```

4.5.10. Obtaining the Operating System Image

The following SYSGEN pulls the Operating System Image off of the System Tracks. If you are installing a ZCPR3 System from scratch, you should be running a CP/M system which has been moved down to allow space for the ZCPR3 buffers in high memory. MOVCPM can usually be used to do this (MOVCPM is provided with your system as a CP/M utility).

In this example, the size of the CP/M system which this version of ZCPR3 was built on is the same as the size of the ZCPR3 which is being built. Both have the same size of Transient Program Area (TPA).

The following command pulls the Operating System Image off of the System Tracks:

```
B11>sysgen
SYSGEN VER 2.2
SOURCE DRIVE NAME (OR RETURN TO SKIP)a
SOURCE ON A, THEN TYPE RETURN
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)

B11>save 45 cpm.bin
```

We now have a file named CPM.BIN on disk. This is the Operating System Image. Note that the "45" above is a number unique to my system and it may be different for the installer’s system.

4.5.11. Patching the CP/M System Image

The following illustrates the technique of patching the CP/M Operating System Image. The CCP is replaced by ZCPR3 (ZCPR3.HEX) and the BIOS is replaced by CBIOS (CBIOS.HEX).

In this example, the CP/M Operating System Image begins at 1100H. This may be different on the system being installed.

```
B11>ddt cpm.bin
DDT VERS 2.0
NEXT  PC
2E00 0100
```

The following command displays the CCP area. Your display will probably not be the same as this (I was running ZCPR3 at the time this display was created), but the installer should recognize the two opening JMP instructions as a key to being sure that the CCP is indeed at this location.

```
-d1100 111f
1100 C3 3E C0 C3 3E C0 43 4F 4D 01 24 24 24 20 20 20 .>..>.COM.$$$
1110 20 20 53 55 42 00 00 00 00 00 00 00 00 00 00 00   SUB...........
```

The following Zeroes out the CCP area [I prefer to do this - just in case!].

```
-f1100 18ff 0
```

Now the ZCPR3 Command Processor Replacement is read in on top of the CCP area. The offset here is 5100, and it may be different for the system being installed. After the read-in, a dump is done to make sure the image was loaded properly.

```
-izcpr3.hex
-r5100
NEXT  PC
2E00 0000
-d1100 111f
1100 C3 3E C0 C3 3E C0 43 4F 4D 01 24 24 24 20 20 20 .>..>.COM.$$$
1110 20 20 53 55 42 00 00 00 00 00 00 00 00 00 00 00   SUB...........
```

The following command displays the BIOS area. Your display will probably not be the same as this, but the installer should recognize the two opening JMP instructions as a key to being sure that the BIOS is indeed at this location.

```
-d2700 270f
2700 C3 CC D6 C3 70 D7 C3 0C EC C3 39 D6 C3 3F D6 C3 ....p.....9..?..
```

The following Zeroes out the BIOS area [I prefer to do this - just in case!].

```
-f2700 2dff 0
```

Now the CBIOS is read in on top of the BIOS area. The offset here is 5100, and it may be different for the system being installed. After the read-in, a dump is done to make sure the image was loaded properly.

```
-icbiosz.hex
-r5100
NEXT  PC
2E00 0000
-d2700 274f
2700 C3 CC D6 C3 70 D7 C3 0C EC C3 39 D6 C3 3F D6 C3 ....p.....9..?..
2710 15 EC C3 18 EC C3 1B EC C3 C3 D7 C3 04 D8 C3 C5 ................
2720 D7 C3 B7 D7 C3 BD D7 C3 F6 D8 C3 EF D8 C3 1E EC ................
2730 C3 CA D7 C3 1B FC C3 21 EC CD 64 D9 C3 0F EC C5 .......!..d.....
2740 CD 64 D9 C1 C3 12 EC CD 0C EC B7 C4 0F EC 21 80 .d............!.
-^C
```

The following SAVE places the new ZCPR3 Operating System Image on disk.

```
B11>save 46 zcpr3.bin
```

Just to double-check, the image is reloaded and the end of the CBIOS is displayed to make sure the image is complete.

```
B11>ddt zcpr3.bin
DDT VERS 2.0
NEXT  PC
2E00 0100
-d2dc0 2dff
2DC0 00 00 00 00 00 00 F8 E0 00 00 03 E3 B8 E2 00 00 ................
2DD0 00 00 00 00 00 00 F8 E0 00 00 8E E3 43 E3 45 6E ............C.En
2DE0 64 20 6F 66 20 43 42 49 4F 53 5A 00 00 00 00 00 d of CBIOSZ.....
2DF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
-^C
```

4.5.12. Placing the Operating System Image

The following now places the Operating System Image onto the system tracks of the desired disks. Note that this image is memory-resident from the previous execution of DDT. Write out the image on as many test disks as desired.

```
B11>sysgen
SYSGEN VER 2.2
SOURCE DRIVE NAME (OR RETURN TO SKIP)
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)B ...
```

5. S T E P   7 :   S Y S T E M   S E G M E N T S

The selection of the configuration options for the various System Segments of ZCPR3 is described here in some detail.

5.1. Resident Command Packages

The following is a reformatted duplicate of the body of a SYSRCP.LIB file. It is provided here to present additional information on how to set the equates. It may be useful to the installer to have this installation manual open to these pages while he is editing this file.

Each entry for the resident commands mentions their transient program counterparts. These utilities usually provide capabilities which exceed those of the programs in the Resident Command Package, but the tradeoff is that each utility program is a separate file on disk which usually occupies more disk space than an entire RCP. In essence, the RCP commands provide quick, convenient capabilities to the user, and the transient utilities provide much greater flexibility and utility to the user. In most reasonable ZCPR3 Systems, both facilities are available.

SYSTEM SEGMENT: SYS1.RCP
SYSTEM: ZCPR3
WRITTEN BY: RICHARD CONN

PROGRAM HEADER: SYSRCP.LIB
AUTHOR: RICHARD CONN

This program header selects the commands to be incorporated into SYS.RCP. It also allows selection of some options for these commands.

IDENTIFICATION

The following ID is a single character, displayed as a part of the RCP ID, which distinguishes this RCP from others made from the same base file (SYSRCP.ASM).

```
RCPID   EQU     'A'
```

With the potential of several RCPs being generated from this one file, RCPID is useful in identifying which RCP is currently loaded to the user. The H command, built into every RCP, prints out the version number of the RCP, including the RCPID character, as well as the names of the commands contained within the RCP.

5.1.1. CP Command
TRANSIENT COUNTERPART: MCOPY

The following equate determines if the CP command is made available. Setting this equate to TRUE enables the CP command.

The CP command copies one file from one DU to another or into the same DU under a different name. The syntax is:

```
                CP dir:ufn=dir:ufn
```

Examples:

```
                CP f1.txt=f2.txt
                CP a15:=f1.txt
                CP a15:f2.txt=c5:f1.txt

CPON            EQU     TRUE
```

5.1.2. DIR Command
TRANSIENT COUNTERPART: DIR, XD, XDIR

The following equate determines if the DIR command is made available. Setting this equate to TRUE enables the DIR command.

The DIR command displays the directory of files in alphabetical order across the lines to the user. The syntax is:

```
                DIR dir:afn
```

Examples:

```
                DIR b7:
                DIR root:*.com

DIRON           EQU     FALSE
```

The DIR command allows two options. One is a flag to tell it to look at both System and Non-System files, and the other is a flag to tell it to look only at System files. By default, DIR looks at Non-System files.

SYSFLG defines the character used to instruct DIR to look at both System and Non-System files. The recommended value is ’A’ for All.

SOFLG defines the character used to instruct DIR to look at only System files. The recommended value is ’S’ for System.

```
SYSFLG          EQU     ’A’
SOFLG           EQU     ’S’
```

The following equate determines if the directory displays are sorted by filename and filetype or by filetype and filename. Set SORTNT to TRUE to sort by name and type, FALSE to sort by type and name.

```
SORTNT          EQU     TRUE
```

The following equates define some features of the directory display. If WIDE is TRUE, the file names are spaced farther abort; if WIDE is FALSE, they are closer together (for a 64-column display). FENCE defines the character used to separate the file name entries in the display.

```
WIDE            EQU     TRUE
FENCE           EQU     '|'
```

5.1.3. ERA Command

TRANSIENT COUNTERPART: ERASE

The following equate determines if the ERA command is made available. Setting this equate to TRUE enables the ERA command.

The ERA command erases files. The syntax is:

```
                ERA dir:afn
or:
                ERA dir:afn I   -- Inspect
```

Examples:

```
                ERA b7:*.bak
                ERA text:*.tmp i
```

```
ERAON           EQU     TRUE
```

5.1.4. LIST and TYPE Commands
TRANSIENT COUNTERPART: PRINT and PAGE

The following equate determines if the LIST and TYPE commands are made available. Setting this equate to TRUE enables these commands.

The LISTON equate can disable the LIST command without affecting the TYPE command.

The TYPE command displays a group of files on the CRT while the LIST command prints a group of files on the Printer. The syntax is:

```
                TYPE dir:afn  -or-  LIST dir:afn
```

Examples:

```
                TYPE b7:*.asm
                LIST text:*.txt
```

```
LTON            EQU     TRUE
LISTON          EQU     TRUE
```

TYPE can be made to page or not page by default. If PGDFLT is TRUE, TYPE pages by default and does not page if the PGFLG character (recommended to be ’P’) is used. If PGDFLT is FALSE, TYPE pages only when the PGDFLG character is seen in the command line.

```
PGDFLT          EQU     TRUE
PGDFLG          EQU     ’P’
```

NLINES defines the number of lines on the user’s CRT screen. This is usually 24.

```
NLINES          EQU     24
```

5.1.5. PEEK and POKE Commands

TRANSIENT COUNTERPART: None (Subset of DDT)

The following equates determine if the PEEK and POKE commands are made available. Setting these equates to TRUE enables these commands.

The PEEK command allows the user to examine a chunk of memory. If the user simply types "P" with no address, the next 256 bytes of memory are displayed. If the user types "P address", 256 bytes of memory starting at the indicated address are displayed. If the user types "P addr1 addr2", memory in this address range is displayed. The syntax:

```
                P
or:
                P address
or:
                P addr1 addr2
```

The POKE command allows the user to change the content of memory. The user must specify an address to POKE, and two basic forms are allowed:

```
                POKE address val1 val2 ... valn
and:
                POKE address "character string
```

The two forms may be intermixed with leading values and a trailing character string:

```
                POKE address val1 val2 ... valn "character string
```

Examples:

```
                P
                P f400
                P f400 f425
                POKE f400 0 1 2
                POKE f400 "this is a test
                POKE f400 1 2 3 "hello, world
```

```
PEEKON          EQU     TRUE
POKEON          EQU     TRUE
```

5.1.6. PROT Command
TRANSIENT COUNTERPART: PROTECT

The following equate determines if the PROT command is made available. Setting this equate to TRUE enables the PROT command.

The PROT command sets the file protection attributes for a group of files. The R/O and System attributes may be set with the R and S options, resp, given in any order as "RS" or "SR". Omission of one of these options toggles the opposite (ie, omission of R makes the files R/W). The syntax:

```
                PROT dir:afn
or:
                PROT dir:afn R
or:
                PROT dir:afn S
or:
                PROT dir:afn RS  -or-  PROT dir:afn SR
```

Examples:

```
                PROT b7:*.com rs
                PROT text:*.txt
```

```
PROTON          EQU     TRUE
```

5.1.7. REN Command
TRANSIENT COUNTERPART: RENAME

The following equate determines if the REN command is made available. Setting this equate to TRUE enables the REN command.

The REN command changes the name of one file to another. The syntax:

```
                REN dir:ufn1=ufn2
```

Examples:

```
                REN newfile.txt=oldfile.txt
                REN root:sys.rcp=sys1.rcp
```

```
RENON           EQU     TRUE
```

5.1.8. REG Command
TRANSIENT COUNTERPART: REG

The following equate determines if the REG command is made available. Setting this equate to TRUE enables the REG command.

```
        The REG command forms are:
                REG D or REG    = display values of all registers
                REG Mreg        = subtract 1 from register (Minus)
                REG Preg        = add 1 to register (Plus)
                REG Sreg value  = set value of indicated register
```

A register is a ZCPR3 register buffer, indicated by a digit from 0 to 9.

```
     Examples:
                REG S0 4        -- reg 0 = 4
                REG S5          -- reg 5 = 0
                REG P           -- reg 0 = reg 0 + 1
                REG P5          -- reg 5 = reg 5 + 1
                REG M9          -- reg 9 = reg 9 - 1
                REG D           -- show values
                REG             -- show values

REGON           EQU     FALSE
```

5.1.9. WHL Command
TRANSIENT COUNTERPART: WHEEL

The following equate determines if the WHL command is made available. Setting this equate to TRUE enables the WHL command.

The WHL command is used to turn off the Wheel Byte (make the user non-priveleged) or to turn on the Wheel Byte (make the user priveleged). The syntax is:

```
                WHL             -- make user non-priveleged
or:
                WHL password    -- make user priveleged
```

Also, this equate enables the WHLQ command, which displays the state of the Wheel Byte. The syntax is:

```
                WHLQ

     Examples:
                WHL
                WHL mypass
                WHLQ

WHLON           EQU     FALSE
```

The following equate defines the password to be used by the WHL command. It must always be 8 bytes long (trailing spaces allowed) and must be upper-case.

```
WPASS   MACRO
        DB      'SYSTEM  '      ;8 characters
        ENDM
```

The Wheel equate table enables the WHEEL facility of ZCPR3. With this facility, a WHEEL BYTE, which exists somewhere in memory, is examined before a set of installer-selected commands are executed. If this byte is not zero, then the command proceeds. If it is zero, then the command is not allowed to proceed and is exited with an error message.

The following set of equates make each of the indicated commands selectable to respond to the Wheel Byte or not. For instance, if WERA=TRUE, then it responds to the Wheel Byte; if WERA=FALSE, it does not.

These options will only be effective if a Wheel Byte is Defined (Z3WHL NE 0)

```
WCP     equ     FALSE   ;Make CP   a Wheel-Oriented Command
WDIR    equu    FALSE   ; "   DIR  "  "       "       "
WERA    equ     FALSE   ; "   ERA  "  "       "       "
WLIST   equ     FALSE   ; "   LIST "  "       "       "
WPEEK   equ     FALSE   ; "   PEEK "  "       "       "
WPOKE   equ     FALSE   ; "   POKE "  "       "       "
WPROT   equ     FALSE   ; "   PROT "  "       "       "
WREG    equ     FALSE   ; "   REG  "  "       "       "
WREN    equ     FALSE   ; "   REN  "  "       "       "
WTYPE   equ     FALSE   ; "   TYPE "  "       "       "

WHEEL   set     WCP OR WDIR OR WERA OR WLIST OR WPEEK OR WPOKE
WHEEL   set     WHEEL OR WPROT OR WREG OR WREN OR WTYPE
```

5.1.10. NOTE Command
TRANSIENT COUNTERPART: NOTE

NOTE is simply a NOP (do nothing) command which can be used to place comments into multiple command lines. For instance, in the following line:

```
                    dir *.com;note this is a dir display;era *.bak
```

the DIR and ERA commands perform normally, and NOTE simply does nothing very efficiently.

Setting the following equate to TRUE enables the NOTE Command.

```
NOTEON  EQU     TRUE
```

The NOTE command is very convenient in the creation of commented displays and command files. It is generally recommended to implement this command as a resident command within the ZCPR3 Command Processor itself rather than within an RCP since the ZCPR3 Command Processors tend to have more room to spare than RCPs and it is frequently desirable to save as much space within an RCP as possible.

5.1.11. ECHO Command
TRANSIENT COUNTERPART: ECHO

The following equate enables the ECHO command.

ECHO is useful in issuing both messages (to the user, say within a command file during execution) and escape sequences. ECHO can send its output to the console (by default) or to the printer (if the first non-blank character is a dollar sign). It uses BIOS calls, so all control characters are passed exactly. Hence, console-level programming of such devices (CRTs and Printers) is possible.

The ECHOLST equate determines if ECHO is allowed to direct its output to the printer. If ECHOLST is TRUE, ECHO may direct its output to the printer via the $ prefix character in the text.

```
ECHOON  EQU     TRUE
ECHOLST EQU     TRUE
```

The ECHO transient is not very large, and it is frequently more convenient to have ECHO implemented in an RCP. However, since space within RCPs is frequently at a premium, it may be necessary to employ the ECHO transient.

5.2. Flow Command Packages

The following is a reformatted duplicate of the body of a SYSFCP.LIB file. It is provided here to present additional information on how to set the equates. It may be useful to the installer to have this installation manual open to these pages while he is editing this file.

A key decision to be made in the creation of FCPs is whether to implement the IF command as a COM file or within the FCP itself. The following tradeoff should be considered:

1) As a COM file, the IF command offers many more options and flexibility for condition processing than an FCP-resident IF.

2) As a COM file, the IF command adds overhead by having to be located and loaded from disk and then executed.

In the following text, sections 5.2.1 to 5.2.11 describe options for an FCP-resident IF command. IF.COM contains all of these options and more. Refer to the associated HLP file for more detail.

SYSTEM SEGMENT: SYS1.FCP
SYSTEM: ZCPR3
CUSTOMIZED BY: RICHARD CONN

PROGRAM HEADER: SYSFCP.LIB
AUTHOR: RICHARD CONN

This program header defines the IF Conditions to be placed into the target SYS.FCP file (generated by assembling SYSFCP.ASM).

5.2.1. IF Negation

The following equate determines if leading negation is to be allowed. If this equate is TRUE, then forms like the following are permitted:

```
                IF ~EXIST filename.typ
```

meaning to complement the meaning of the test (the above returns TRUE if filename.typ does NOT exist).

```
IFONEG          EQU     TRUE
```

Assuming IFONEG to be TRUE, the following equate defines the character to be placed in front of the IF option to indicate that negation is to be performed. In the above example, this character was tilde (~).

```
NEGCHAR         EQU     '~'
```

5.2.2. IF: T (True) or F (False)

Setting the following equate to TRUE enables the simple T and F options to IF. The format of this option is:

```
                IF T or IF F
```

and it always returns TRUE or FALSE, resp.

```
IFOTRUE         EQU     FALSE
```

5.2.3. IF: EM (Empty)

Setting the following equate to TRUE enables IF to test to see if the indicated file is empty or not. The format of this option is:

```
                IF EM dir:filename.typ
```

and it returns TRUE if the indicated file does not exist or is empty.

```
IFOEMPTY        EQU     FALSE
```

5.2.4. IF: ER (Error)

Setting the following equate to TRUE enables IF to test the error code byte (program error code byte). If this byte is 0 (no error), it returns TRUE, else it returns FALSE. The format of this option is:

```
                IF ER
```

```
IFOERROR        EQU     TRUE
```

5.2.5. IF: EX (Exist)

Setting the following equate to TRUE enables IF to test for the existence of a file. The format of this option is:

```
                IF EX dir:filename.typ
```

and it returns TRUE if the indicated file exists.

```
IFOEXIST        EQU     TRUE
```

5.2.6. IF: IN (Input)

Setting the following equate to TRUE enables user input of the character T (or any other character for FALSE). ZEX processing is suspended for this single-character input. The format of this option is:

```
                IF IN
```

and the IF FCP command responds with:

```
                IF True?
```

to which the user types T, Y, SPACE, or CR to set the IF to TRUE and anything else to set the IF to FALSE.

```
IFOINPUT        EQU     TRUE
```

5.2.7. IF: NU (Null)

Setting the following equate to TRUE enables IF to test to see if the second argument which follows is NULL (not specified) or not. This test is particularly useful in command file processing to see if, for example, argument $2 exists and to include it if it does. The format of this option is:

```
                IF NU
or:
                IF NU arg
```

If the first format is encountered, IF NU returns TRUE; IF NU returns FALSE with the second format.

```
IFONULL         EQU     TRUE
```

5.2.8. IF: n (Register Value)

Setting the following equate to TRUE enables IF to test to see if the indicated register contains the indicated value. If this is preceeded by the NEGCHAR and IFONEG is TRUE, then this tests to see if the indicated register does not contain the indicated value. Registers are one-byte memory buffers, and are identified by the digits 0 to 9. The format of this option is:

```
                IF n val

     Example:
                IF 0            -- if Reg 0 = 0
                IF 0 5          -- if Reg 0 = 5
                IF 5 2          -- if Reg 5 = 2
                IF ~0           -- if Reg 0 <> 0
                IF ~9 2         -- if Reg 9 <> 2

IFOREG          EQU     TRUE
```

The REG command (implemented either within an RCP or as a COM file) is used to place values into these register and modify and display these values.

5.2.9. IF: WH (Wheel)

Setting the following equate to TRUE enables IF to test to see if the Wheel Byte is set or not. If so, IF WHEEL is TRUE.

```
IFOWHEEL        EQU     FALSE
```

5.2.10. IF: TC (TCAP)

Setting the following equate to TRUE enables IF to test to see if the ZCPR3 TCAP contains a terminal definition or not. This test is particularly useful in command file or alias processing to see if, for example, a Z3TCAP entry is defined and to invoke screen-oriented routines if it is. The format of this option is:

```
                IF TC

IFOTCAP         EQU     FALSE
```

5.2.11. IF: fcb1=fcb2

Setting this equate to TRUE will enable IF to evaluate the equality condition, checking to see if the two FCBs contain the same values. If so, the IF is TRUE; if not, the IF is FALSE.

Enabling this equate eliminates the need for the NULL test, since a NULL test can be performed by using the syntax:

```
                IF fcb1=
```

```
IFOEQ           EQU     TRUE
```

5.2.12. COMIF - Run IF.COM

Setting this equate to TRUE will cause an IF executed during an IF TRUE or NO IF state to look in the ROOT directory (base of path starting in current directory) for the file IF.COM, and, if found, load IF.COM and transfer control to it. If IF.COM is not found, then IF F is raised. Using IF.COM provides much more power and flexibility but also requires IF.COM to be present and takes up disk space.

```
COMIF           EQU     FALSE
```

5.2.13. NOISE - Have FCP Print IF Status Messages

Setting this equate to TRUE will cause any change in the IF status to be printed to the user. This is useful for debugging purposes, but in normal runs, particularly where ALIASes are concerned, it is usually desirable to reduce the "noise" as much as possible and have this equate set to FALSE.

```
NOISE           EQU     FALSE
```

5.3. Input/Output Packages

Input/Output Packages are very machine-specific, but, like all packages, they provide a machine-independent interface to the ZCPR3 System in their visible sections. The hidden part performs the actual implementation of the routines. Like the structure of the BIOS, the visible section of an I/O Package consists of a JMP table.

The installer who is interested in incorporating Input/Output Packages into the system he is installing is referred to the source code file SYSIOP.ASM. SYSIOP.ASM can be used as a template through which to create other I/O Packages. It is filled with comments outlining the functions being performed, and I feel that this should be adequate.

5.4. Named Directory Files

The following is a reformatted duplicate of the body of a SYSNDR.LIB file. It is provided here to present additional information on how to set the equates. It may be useful to the installer to have this installation manual open to these pages while he is editing this file.

```
DATA FILE:  SYSNDR.LIB
AUTHOR:  Richard Conn
VERSION:  1.0
DATE:  24 Feb 84
```

SYSNDR.LIB defines the structure of the memory-based named directory. It also defines a few elements for it and is suitable for enclosure in an NDR file.

The general structure is:

```
                DB      Disk,User       ; A=1
                DB      'NDIRNAME'      ; 8 chars
                DB      'PASSWORD'      ; 8 chars
                ...                     ; other entries
                DB      0               ; End of NDR

defdu   macro   ?disk,?user
        db      ?disk-'@'               ; Convert Disk
        db      ?user                   ; User is OK
        endm
```

FIG 5-1: Named Directory Structure

The entire file is implemented as one macro (which follows). The SYSNDR.ASM file simply refers to this macro and expands it.

The named directories shown below are recommended standards. In time, there will be utilities which base a part of their operations on these names.

```
sysndr  macro
```

The BASE directory is a working scratch area on the first disk.

```
        defdu   'A',0
        db      'BASE    '
        db      '        '
```

The ROOT directory is the last directory referenced in the Command Search Path. This is where all of the general-purpose COM files are located.

```
        defdu   'A',15
        db      'ROOT    '
        db      '        '
```

The HELP directory is where the online documentation files are stored.

```
        defdu   'A',16
        db      'HELP    '
        db      '        '
```

The BACKUP directory is where files are copied to (by default) for backup purposes.

```
        defdu   'C',0
        db      'BACKUP  '
        db      '        '

        db      0               ;End of List
        endm
```

5.5. TCAP Files

The programs TCSELECT and TCMAKE are used to create the *.Z3T files which are loaded by the LDR.COM utility. The loaded file establishes the characteristics of the user’s CRT terminal, and this information is used by screen-oriented utilities, such as SHOW, to perform their functions.

5.6. Environment Descriptor

The following is a reformatted duplicate of the body of a SYSENV.LIB file. It is provided here to present additional information on how to set the equates. It may be useful to the installer to have this installation manual open to these pages while he is editing this file.

The entire file is one macro which is referenced by SYSENV.ASM. SYSENV inserts a JMP 0 instruction in front of this macro to complete the structure of the SYS.ENV file.

```
LIBRARY:  SYSENV.LIB
AUTHOR:  Richard Conn
Version:  1.0
Date:  18 May 84
Previous Versions:  None


     SYSENV is the definition for my ZCPR3 environment.


sysenv  macro
;
;  Environment Descriptor
;       If inline, there is a leading JMP just before this
;
envorg1:
        db      'Z3ENV'         ; Environment ID
        db      1               ; class 1 environment (external)
```

A Class 1 environment is external to the utility using it. This type of Environment Descriptor is located at a buffer somewhere in memory, and the ZCPR3 utilities simply contain a 2-byte pointer which contains its address. A Class 2 environment is internal to the utility using it. This type of Environment Descriptor is located within the utility itself, taking up 256 bytes. It is recommended that the ZCPR3 System be configured using an external Environment Descriptor.

The following addresses and values are extracted from Z3BASE.LIB.

```
        dw      expath          ; external path address
        db      expaths         ; number of 2-byte elements in path

        dw      rcp             ; RCP address
        db      rcps            ; number of 128-byte blocks in RCP

        dw      iop             ; IOP address
        db      iops            ; number of 128-byte blocks in IOP

        dw      fcp             ; FCP address
        db      fcps            ; number of 128-byte blocks in FCP

        dw      z3ndir          ; NDR address
        db      z3ndirs         ; number of 18-byte entries in NDR

        dw      z3cl            ; ZCPR3 Command Line
        db      z3cls           ; number of bytes in Command Line

        dw      z3env           ; ZCPR3 Environment Descriptor
        db      z3envs          ; number of 128-byte blocks

        dw      shstk           ; Shell Stack address
        db      shstks          ; number of shsize-byte entires
        db      shsize          ; size of a Shell Stack entry

        dw      z3msg           ; ZCPR3 Message buffer

        dw      extfcb          ; ZCPR3 External FCB

        dw      extstk          ; ZCPR3 External Stack
```

The following flag is used by some ZCPR3 System utilities to determine how verbose they are in providing messages and information to the user. The QUIET.COM utility can be used to change this flag dynamically.

```
        db      0               ; quiet flag (1=quiet, 0=not quiet)

        dw      z3whl           ; address of Wheel Byte
```

This data value is used by the timing routines.

```
        db      4               ; Processor Speed in MHz
```

The following values should correspond to those selected in the Z3HDR.LIB file.

```
        db      'D'-'@'         ; maximum disk
        db      31              ; maximum user
```

The following value is used to instruct the utilities as to whether they should accept the DU form or not. If disabled (set to 0), the only way to reference a directory is with the DIR (named) form, and password protection is directly provided by this.

```
        db      1               ; 1=OK to accept DU, 0=not OK
```

Some ZCPR3 utilities, such as PRINT and PAGE, draw information from these buffers to determine several key attributes of the devices they are dealing with. The CPSEL utility can be used to dynamically change the CRT and Printer selections.

```
        db      0               ; CRT selection (0=CRT 0, 1=CRT 1)
        db      0               ; Printer selection (n=Printer n)

        db      80              ; width of CRT 0
        db      24              ; number of lines on CRT 0
        db      22              ; number of lines of text on CRT 0

        db      132             ; width of CRT 1
        db      24              ; number of lines on CRT 1
        db      22              ; number of lines of text on CRT 1

        db      80              ; width of Printer 0
        db      66              ; number of lines on Printer 0
        db      58              ; number of lines of text on Printer 0
        db      1               ; form feed flag (0=can't formfeed, 1=can)

        db      102             ; width of Printer 1
        db      66              ; number of lines on Printer 1
        db      58              ; number of lines of text on Printer 1
        db      1               ; form feed flag (0=can't formfeed, 1=can)

        db      80              ; width of Printer 2
        db      66              ; number of lines on Printer 2
        db      58              ; number of lines of text on Printer 2
        db      0               ; form feed flag (0=can't formfeed, 1=can)

        db      102             ; width of Printer 3
        db      66              ; number of lines on Printer 3
        db      58              ; number of lines of text on Printer 3
        db      0               ; form feed flag (0=can't formfeed, 1=can)
```

The ZCPR3 shell named SH can deal with symbols (variables) which are assigned text strings as values. This buffer defines the name of the file which programs like SH refer to in order to resolve variable references. As many shell variable files as desired may be available in this fashion.

```
        db      'SH      '      ; shell variable filename
        db      'VAR'           ; shell variable filetype
```

These buffers are available to store file names and other data which are passed from one utility to another which is executed later. In general, entries 3 and 4 are available to the ZCPR3 utility programmer as general-purpose buffers. Entries 1 and 2 are used by some ZCPR3 System utilities at this time.

```
        db      '        '      ; filename 1
        db      '   '           ; filetype 1

        db      '        '      ; filename 2
        db      '   '           ; filetype 2

        db      '        '      ; filename 3
        db      '   '           ; filetype 3

        db      '        '      ; filename 4
        db      '   '           ; filetype 4

        ds      80H-($-envorg1+3)       ; make exactly 80H bytes long
                                        ; (+3 compensates for leading JMP)
```

The following is the TCAP entry for the TVI 950. If LDR.COM loads a *.Z3T file, this buffer will be overlaid (if the Environment Descriptor is External).

```
;
; Terminal Capabilities Data
;
envorg2:
        DB      'TVI 950         '      ;Name of Terminal
        DB      'K'-'@'                 ;Cursor UP
        DB      'V'-'@'                 ;Cursor DOWN
        DB      'L'-'@'                 ;Cursor RIGHT
        DB      'H'-'@'                 ;Cursor LEFT
        DB      00                      ;CL Delay
        DB      00                      ;CM Delay
        DB      00                      ;CE Delay
        DB      1bh,'*',0               ;CL String
        DB      1bh,'=%+ %+ ',0         ;CM String
        DB      1bh,'t',0               ;CE String
        DB      1bh,')',0               ;SO String
        DB      1bh,'(',0               ;SE String
        DB      0                       ;TI String
        DB      0                       ;TE String

        ds      80H-($-envorg2)         ; make exactly 80H bytes long

;
;  End of Environment Descriptor
;
        endm
```

6. S T E P   8 :   U T I L I T Y   I N S T A L L A T I O N

6.1. The Z3INS Utility

The Z3INS utility is designed to make the ZCPR3 utility installation process simple. All files to be installed must be in the current directory when Z3INS is executed. A *.ENV file for the target system and an installation file (*.INS) containing the names of the programs to be installed must also be in the current directory.

Z3INS reads in an Environment Descriptor file (*.ENV) and an Installation File (*.INS). It then looks for lines in the file containing file names (one name per line) and loads the indicated files, trying to install them with the Environment Descriptor information.

Z3INS is invoked by a command line of the following form:

```
     Z3INS mysys.ENV myinstal.INS
```

A ZCPR3 Installation File is a text file containing two types of lines: a comment line, which begins with a semicolon (;), and a line containing an unambiguous file name (leading spaces are not significant), which is a file to be installed. For example:

```
; This is an installation file for my new utilities
; UTIL1.COM and UTIL2.COM are going to be installed --
 util1.com
     util2.com
; UTIL3 is really neat
util3.com
```

FIG 6-1: Sample Z3INS Installation File

Case is not significant. Leading spaces on each line are ignored. Any file name MUST be unambiguous.

The next section shows the execution of a *.INS file.

6.2. Sample Session

```
B1:ASM>z3ins sys.env zcpr3.ins
Z3INS  Version 1.0
;
;  Installation Begins --
;

                      << Detail Left Out >>

;
;  Set 1
;
** Installing File ALIAS   .COM
** Installing File CD      .COM
** Installing File CMDRUN  .COM
** Installing File COMMENT .COM
** Installing File CPSEL   .COM
** Installing File CRC     .COM
** Installing File DEV     .COM
** Installing File DEVICE  .COM
** Installing File DIFF    .COM
** Installing File DIR     .COM
** Installing File ECHO    .COM
** Installing File ERASE   .COM
;
;  Set 2: Error Handlers
;
** Installing File ERROR1  .COM
** Installing File ERROR2  .COM
** Installing File ERROR3  .COM
** Installing File ERROR4  .COM
** Installing File ERRORX  .COM
** Installing File SHOW    .COM

                      << Detail Left Out >>

;
;  NOTE does not install because it is so small and really does
;  not need to know about ZCPR3
;
;note.com
;
;  Set 9:  Z3INS
;
** Installing File Z3INS   .COM
;
; End of ZCPR3 Installation
;
** Installation Complete **
```

FIG 6-2: Sample Run of Z3INS

6.3. Assembling Distribution Files

The following files require their specialized command files in order to be assembled. If the installer is installing the system for the first time and wishes to assemble these utilities, he may have to follow the steps outlined in the command files in order to perform the assemblies.

Utility	Command File Required
ALIAS.COM	ALIAS.ZEX
ZEX.COM	ZEX.ZEX

The following files in the Phase I distribution are assembled by the command lines (assuming that Z3LIB.REL and SYSLIB.REL are in the current directory and that $1 is the file):

```
          M80 =$1
          L80 $1,Z3LIB/S,SYSLIB/S,$1/N,/U,/E
```

Files:

CD	CMDRUN	COMMENT	CPSEL
CRC	DEV	DEVICE	DIFF
DIR	ECHO	ERASE	ERROR1
ERROR3	ERROR4	ERRORX	FINDF
GOTO	HELPCK	IF	IFSTAT
LDR	MCOPY	MENUCK	MKDIR
NOTE	PAGE	PATH	PROTECT
PWD	QUIET	RECORD	REG
RENAME	SAK	SETFILE	SH
SHCTRL	SHDEFINE	SHFILE	SHVAR
SUB	TCCHECK	TCMAKE	TCSELECT
UNERASE	WHEEL	XD	XDIR
Z3INS	Z3LOC		

The following files in the Phase I distribution are assembled by the command lines (assuming that VLIB.REL, Z3LIB.REL, and SYSLIB.REL are in the current directory):

```
          M80 =$1
          L80 $1,VLIB/S,Z3LIB/S,SYSLIB/S,$1/N,/U,/E
```

Files:

ERROR2	HELP	MENU	SHOW

The following files in the Phase I distribution are assembled by the command lines (assuming that Z3LIB.REL and SYSLIB.REL are in the current directory) if the TIME option is enabled:

```
          M80 =$1
          L80 $1,TIMELIB/S,Z3LIB/S,SYSLIB/S,$1/N,/U,/E
```

Files:

HELPPR	PRINT

Files distributed in Phase II will be provided with associated documentation on their assembly procedures.

7. The Z3TCAP FACILITY

7.1. ZCPR3 Terminal Capabilities (TCAP)

The ZCPR3 Terminal Capabilities (TCAP) Facility is an integral part of the ZCPR3 System. By means of the TCAP Facility, the user’s terminal is defined to ZCPR3 in such a way that programs in the ZCPR3 System can perform a variety of screen-oriented functions with the user’s terminal. The TCAP Facility is fundamental to ZCPR3, and it is a part of the ZCPR3 Environment Descriptor.

The TCAP entries contain the following information on their respective terminals:

- o initialization/deinitialization sequences
- o characters generated by the arrow keys
- o sequence for clearing the screen
- o sequence for positioning the cursor
- o sequence for erasing to end of line
- o highlight/non-highlight sequences

With this information, programs such as VFILER, VMENU, and HELP can perform their functions with a much higher degree of "flash" and user-friendliness than they would otherwise. By simply loading the TCAP entry for another terminal into the environment descriptor, all ZCPR3 programs are automatically reconfigured for the new terminal and can continue to function without modification.

Two utilities are provided to assist the user in the creation of Z3T files for his terminals. TCSELECT allows the user to select a predefined terminal from the Z3TCAP file, and TCMAKE allows the user to define a terminal which is not covered by the Z3TCAP file.

Most of the information in this chapter provides details on the structure of the Z3T files and gives the TCMAKE user enough detail to define his terminal. Providing this information is the main purpose of this chapter.

The file Z3TCAP contains information on over 40 terminals. The TCSELECT program prints a number of menus containing the names of the terminals in this file and allows the user to select one, storing its information either directly into the memory-resident Environment Descriptor or into a file of type Z3T which may later be loaded by the LDR utility.

If the user’s terminal is not already defined in the Z3TCAP file, the TCMAKE program is used to define his terminal. TCMAKE allows the user to interactively define each of the key attributes of his terminal and create a file of type Z3T when done. This file may later be loaded by the LDR utility.

7.2. Internal Structure of a Z3T File

A Z3T File defines the characteristics of a particular terminal. Each Z3T file contains the following information:

```
o the name of the terminal
o the codes generated by the arrow keys
o the byte sequences required:
     to clear the screen
     to position the cursor
     to clear to end of line
     to highlight chars
     to initialize and deinitialize the terminal
```

The following is the exact structure of a Z3T file:

```
Z3T_FILE:
name:
     DS   16   ; Name of Terminal
arrows:
     DS   4    ; Bytes generated by arrow keys
delays:
     DS   3    ; Delays for Screen Clear, Cursor Motion, and
               ;   Clear to End-of-Line

cl:  DS   N1+1 ; Sequence used for Screen Clear
cm:  DS   N2+1 ; Sequence used for Cursor Motion (gotoxy)
ce:  DS   N3+1 ; Sequence used for Clear to End-of-Line
so:  DS   N4+1 ; Sequence used to begin highlighting
se:  DS   N5+1 ; Sequence used to end highlighting
ti:  DS   N6+1 ; Sequence used to initialize terminal
to:  DS   N7+1 ; Sequence used to deinitialize terminal
```

FIG 7-1: Z3T File Structure, Overview

The following defines the TCAP records, which are:

1. the name of the terminal
2. the definition of the arrow keys
3. the delay constants for screen clear, cursor motion, and clear to end-of-line
4. the definition of the screen clear char sequence
5. the defn of the cursor motion char seq
6. the defn of the clear to EOL char seq
7. the defn of the highlight/end-highlight char seq
8. the defn of the init/deinit terminal char seq

FIG 7-2: Z3T Records

Each of these record definitions is similar: the structure of the record (in assembly language terminology), comments on how the record is defined and what values are valid for it, and examples of valid record structures are provided for each record definition.

7.2.1. Terminal Name

Structure:

```
     DS   16   ; Name of Terminal (Space Fill on Right)
```

Comment:

The name of the terminal is always 16 bytes long. If the name takes less than 16 bytes, space fill occurs right of the last character.

Examples:

```
     DB   'ADDS Consul 980 '
     DB   'ADM 2           '
```

7.2.2. Arrow Keys

Structure:

```
     DS   1    ; Byte Generated by Cursor UP
     DS   1    ; Byte Generated by Cursor DOWN
     DS   1    ; Byte Generated by Cursor RIGHT
     DS   1    ; Byte Generated by Cursor LEFT
```

Comment:

If your terminal has arrow keys on it WHICH GENERATE ONLY ONE BYTE WHEN DEPRESSED, then these keys may be defined in the Z3T file. When a program calls for the use of arrow keys, it will use the values stored here.

If your terminal does not have arrow keys or has arrow keys which generate more than one byte when depressed, these keys may not be defined in the Z3T file. Zero (0) is stored in all four bytes of the "arrow key" record. In this case, the program will respond to the Word Star (trademark, Micropro) arrow key convention (^E is UP, ^X is DOWN, ^D is RIGHT, and ^S is LEFT):

```
                  ^E
                   ^
                   |
              ^S <--+--> ^D
                   |
                   v
                  ^X
```

FIG 7-3: Wordstar Movement Convention

Examples:

```
     DB   'K'-'@'   ; ADM 31 ^K for Cursor UP
     DB   'J'-'@'   ;        ^J for Cursor DOWN
     DB   'L'-'@'   ;        ^L for Cursor RIGHT
     DB   'H'-'@'   ;        ^H for Cursor LEFT

     DB   0,0,0,0   ; None for H19 because of 2-char seqs
                    ;   Word Star Convention will be used
```

7.2.3. Function Delays

Structure:

```
     DS   1    ; Delay (in mS) after sending clear screen
     DS   1    ; Delay (in mS) after sending gotoxy
     DS   1    ; Delay (in mS) after sending clear to EOL
```

Comment:
     Each of these bytes defines the number of milliseconds a program will delay after sending a particular sequence to the user's terminal. Some terminals require this type of delay. If a sequence requires no delay, the value of zero (0) should be placed in the corresponding byte.

Examples:

```
     DB   20,0,0    ; BANTAM 550 - 20mS for Clear Screen, 0
                    ;  for Cursor Motion and Clear to EOL

     DB   0,0,0     ; TVI 950 - No Delays
```

7.2.4. Clear Screen Sequence

Structure:

```
     DS   N1   ; Bytes in clear screen sequence
     DB   0
```

Comment:
     This sequence of bytes, up to but not including the terminating 0, is sent to the user's terminal in order to clear his screen. If it is necessary to include a binary zero in this sequence, the two bytes

```
          DB   '\',0
```

will transmit as one binary 0 and

```
          DB   '\\'
```

will transmit as one backslash.

     In general, a backslash (\) is the quote character, and any byte which follows it is transmitted literally to the user's terminal.

     If a terminal requires that trailing nulls follow the last character of the sequence for the purpose of screen settling (rather than using the delay byte), nulls can be appended into the sequence by using the quote character.

Examples:
```
     DB   1BH,';',0      ; Clear Screen for ADM2
     DB   'L'-'@',0      ; Clear Screen for ADDS Viewpoint
     DB   1BH,'?',1BH,'E'-'@',0
                         ; Clear Screen for Concept 108
```

7.2.5. Cursor Motion (GOTOXY) Sequence

Structure:
```
     DS   N2   ; Bytes in gotoxy sequence
     DB   0
```

Comments:
This sequence of bytes is sent to the user's terminal in order to position the cursor on his screen. The quote character (\) can be used, like in the Cursor Motion sequence, to allow quote characters and nulls to be sent.

Unlike the other sequences in the TCAP records, the Cursor Motion sequences vary depending upon the position on the screen.

For instance, to place the cursor at row 4, column 4 (home is row 0, col 0) on a TVI 950, the sequence

```
          DB   1BH,'=$$',0
```

is used, but to position at row 6, column 6, the sequence

```
          DB   1BH,'=&&',0
```

is used.

In order to express such variable-value sequences, the ZCPR3 TCAP provides for equations which define how to compute the byte to be output. The TCAP sequence:

```
          DB   1BH,'=%+ %+ ',0
```

defines how to compute the values to be output in order to move the cursor for the TVI 950.

The TCAP Cursor Motion sequence

```
          DB   1BH,'=%+ %+ ',0
```

is broken down as follows:

Element	Meaning
1BH	Output 1B hex (the ESCAPE char)
'='	Output the character '='
'%+ '	Add ' ' (20H) to the row value and output
'%+ '	Add ' ' (20H) to the column value and output

7.2.5.1. Cursor Motion Interpreter Commands

The percent character (%) instructs the cursor motion sequence interpreter to look for a command, and it processes the following characters as such. If it is desired to output '%' itself, the sequence '\%' is used.

The commands recognized by the cursor motion command interpreter will now be discussed in detail. These commands are the following (case is not significant):

```
%R - Reverse order from row/col to col/row
%I - Home position is (1,1) rather than (0,0)
%. - Print current value (row or col) in binary
%2 - Print current value as 2 ASCII decimal digits
%3 - Print current value as 3 ASCII decimal digits
%d - Print current value as N ASCII decimal digits
          (no leading zeroes)
%+n - Add n to current value and output in binary
%>xy - Add y to current value if it is greater
          than x
```

7.2.5.2. %R Command

The cursor motion sequence interpreter assumes that the value of the row will be output before the value of the column. If the column is to be first, the command

```
'%r' or '%R'
```

instructs the cursor motion sequence interpreter to output the column and then the row. The '%R' command must be present in the sequence before the first value is output, and '%R' acts solely to command the interpreter (no bytes are output by '%R').

7.2.5.3. %I Command

The cursor motion sequence interpreter also assumes that the value of the home position is row 0, column 0. If it is convenient to set this position to row 1, column 1, the command

```
'%i' or '%I'
```

is used. Like '%R', '%I' must be used before the first value is output.

The TVI 950 can be defined in two ways:

```
          DB   1BH,'=%+ %+ ',0
or
          DB   1BH,'%i=%+',1FH,'%+',1FH,0
```

7.2.5.4. Output Commands

The rest of the cursor motion sequence interpreter commands deal with the format of the output. They allow the following types of outputs:

```
%.    binary value (^A is output as 1)
%2    2 ASCII Decimal Digits (^A is output as '01')
%3    3 ASCII Decimal Digits (^A is output as '001')
%d    As many ASCII Decimal Digits as needed (^A as '1')
%+n   Add offset ('%+ ' outputs ^A as 1+' ' or '!')
%>xy  Add offset if limit reached ('%> '1 outputs ^A as 1
          and '!' as '"')
```

To summarize:

```
  Command   Output Format
  -------   -------------
    %.      Binary Value
    %2      2 ASCII Decimal Digit Chars ('23')
    %3      3 ASCII Decimal Digit Chars ('123')
    %d      As many ASCII Decimal Digit Chars as needed
    %+n     Add the value of the byte following the '+'
              and output in binary
    %>xy    If value > x, output value+y in binary; else
              output value in binary

  Command   Cursor Motion Interpreter Action
  -------   --------------------------------
    %i      Set Home to 1,1 (default is 0,0)
    %r      Output Col, then Row (default is Row, then Col)
```

Examples:
```
     DB   1BH,'Y%+ %+ ',0     ; ADDS Viewpoint

          1BH   = output 1BH (ESCAPE char)
          'Y'   = output char 'Y'
          '%+ ' = output row + ' ' (20H) in binary
          '%+ ' = output col + ' ' (20H) in binary

     DB   1BH,'[%d;%dH',0     ; H19 (ANSI Mode)

          1BH   = output 1BH (ESCAPE char)
          '['   = output char '['
          '%d'  = output row as ASCII decimal digits
          ';'   = output char ';'
          '%d'  = output col as ASCII decimal digits
          'H'   = output char 'H'
```

7.2.6. Clear to End of Line

Structure:
```
     DS   N3   ; Bytes in clear to end of line sequence
     DB   0
```

Comments:

The Clear to End of Line sequence is used to clear the line starting at the cursor position to the end of the screen. Only this part of the current line is cleared.

The rules for specifying this sequence are the same as those for Screen Clear.

Example:
```
     DB   1BH,'T',0      ; ADM 2
```

7.2.7. Begin and End Highlighting (Standout Mode)

Sequences:
```
     DS   N4   ; Bytes in sequence to begin highlighting
     DB   0

     DS   N5   ; Bytes in sequence to end highlighting
     DB   0
```

Comments:

The "begin highlighting" sequence is used to begin highlight mode on the user's terminal. This may be reverse video, dim, or some other non-standard method for displaying characters on the screen. In order for a terminal to support this feature, the following must be true:

1. Issuing this sequence must NOT change the position of the cursor on the screen.

2. Characters highlighted must be output in exactly the same way non-highlighted characters are (eg, setting the MSB of the highlighted chars is not allowed).

These sequences are always used as follows:

1. the BEGIN HIGHLIGHT sequence is output
2. a set of characters to highlight is output
3. the END HIGHLIGHT sequence is output

The rules for specifying these sequences are the same as those for Screen Clear.

Example:

```
     DB   'N'-'@',0      ;ADDS Viewpoint
```

7.2.8. Terminal Initialization and Deinitialization

Sequences:

```
     DS   N6   ; Bytes in sequence to init terminal
     DB   0

     DS   N7   ; Bytes in sequence to deinit terminal
     DB   0
```

Comments:

Before any video routines are executed, the terminal initialization sequence is sent to the terminal. After the use of the terminal is completed by a program, the deinitialization sequence is sent.

The rules for specifying this sequence are the same as those for Screen Clear.

7.3. Terminal Control Sequences 1 (General)

The structure of most TCAP control sequences is:

```
     DS   N   ; Bytes in sequence
     DB   0
```

This sequence of bytes, up to but not including the terminating 0, is sent to the user's terminal in order to perform some function. If it is necessary to include a binary zero in this sequence, the two bytes

```
          DB   '\',0
```

will transmit as one binary 0 and

```
          DB   '\\'
```

will transmit as one backslash.

A backslash (\) is the quote character, and any byte which follows it is transmitted literally to the user's terminal.

If a terminal requires that trailing nulls follow the last character of the sequence for the purpose of screen settling (rather than using the delay byte), nulls can be appended into the sequence by using the quote character.

Cursor Motion sequences follow these rules with the addition that the character "%" prefixes a cursor motion interpreter command. If it is desired to simply output this character in a cursor motion sequence, the quote character can be used:

```
          DB   '\%'
```

7.4. Terminal Control Sequences 2 (Cursor Motion)

Cursor Motion sequences are different from the other sequences defined in the TCAP in that cursor motion sequences contain embedded commands for the cursor motion interpreter. All Cursor Motion sequences are of the following general format:

```
<prefix sequence> <commands> <infix seq> <cmd> <postfix seq>
```

For example, the DEC VT100 terminal uses the following sequence for cursor motion:

```
          DB   1BH,'[%i%d;%dH',0
where:
          1BH,'['   = prefix chars 1BH (ESCAPE) and '['
          %i        = command: home is 1,1
          %d        = command: output row as ASCII dec chars
          ';'       = infix char ';'
          %d        = command: output col as ASCII dec chars
          'H'       = suffix char 'H'
```

The prefix, infix, and postfix sequences are optional, and only the commands to output the row and col are required in any cursor motion sequence definition.

Cursor Motion is the only required entry in a TCAP for a terminal. All other sequences may be empty (null), and the lack of these other sequences will be compensated for. Cursor motion, however, cannot be simulated easily and is required.

The following table summarizes all of the Cursor Motion interpreter commands.

Command	Output Format
%.	Binary Value
%2	2 ASCII Decimal Digit Chars ('23')
%3	3 ASCII Decimal Digit Chars ('123')
%d	As many ASCII Decimal Digit Chars as needed
%+n	Add the value of the byte following the '+' and output in binary
%>xy	If value > x, output value+y in binary; else output value in binary

Command	Cursor Motion Interpreter Action
%i	Set Home to 1,1 (default is 0,0)
%r	Output Col, then Row (default is Row, then Col)

FIG 7-4: Summary of Cursor Motion Interpreter Commands

Examples:

```
     DB   1BH,'Y%+ %+ ',0     ; ADDS Viewpoint

          1BH   = output 1BH (ESCAPE char)
          'Y'   = output char 'Y'
          '%+ ' = output row + ' ' (20H) in binary
          '%+ ' = output col + ' ' (20H) in binary

     DB   1BH,'[%d;%dH',0     ; H19 (ANSI Mode)

          1BH   = output 1BH (ESCAPE char)
          '['   = output char '['
          '%d'  = output row as ASCII decimal digits
          ';'   = output char ';'
          '%d'  = output col as ASCII decimal digits
          'H'   = output char 'H'
```

7.5. Overview of VLIB

VLIB (Video LIBrary) is the ZCPR3 library which is used to provide a series of low-level routines for Z3TCAP access to the ZCPR3 system programmer. VLIB is described in much more detail in the VLIB.HLP file, and this overview only serves to summarize its capabilities.

The VLIB routine Z3VINIT is used to initialize VLIB for use with a ZCPR3 system. The address of the ZCPR3 environment descriptor is passed to the Z3VINIT routine in HL, and all VLIB routines know the address of the Z3TCAP entry from that point forward.

Some low-level functions provided by VLIB are:

Routine	Function
TINIT	Initialize terminal
DINIT	Deinitialize terminal
CLS	Clear screen
EREOL	Erase to End of Line
GOTOXY	Position Cursor
STNDOUT	Begin highlighting
STNDEND	End highlighting

7.6. Standard ZCPR3 TCAP File

The file Z3TCAP contains information on over 40 terminals. It is provided as a part of the ZCPR3 System, and it is used by TCSELECT. TCSELECT can display the names of the terminals contained in Z3TCAP and allow the user to select one, generating a *.Z3T file or storing the selection directly into memory for immediate use by the ZCPR3 System utilities.

7.7. TCAP Check Program

TCCHECK is used to check the Z3TCAP file for consistency. Its sole function is to ensure the validity of the Z3TCAP file and provide some statistics on it.

Sample run of TCCHECK:

```
B4:SCR2>tccheck //
TCCHECK, Version 1.0TCCHECK - Select Entry from Z3TCAP.Z3T
Syntax:
        TCCHECK infile  -or-  TCCHECK infile.typ

where "infile" is the file to be checked by
the execution of TCCHECK.  If no file type is
given, a file type of Z3T is the default.

Syntax:
        TCCHECK
where this alternate form may be used to check
the Z3TCAP.TCP file.

B4:SCR2>tccheck
TCCHECK, Version 1.0  File Z3TCAP  .TCP Not Found - Aborting
     -- Note: Z3TCAP.TCP MUST be in the same directory

B4:SCR2>tccheck root:z3tcap.tcp
TCCHECK, Version 1.0  File Z3TCAP  .TCP Not Found - Aborting
     -- Note: TCCHECK does not recognize named dirs

B4:SCR2>root:
A15:ROOT>tccheck
TCCHECK, Version 1.0
Z3TCAP File Check of Z3TCAP  .TCP Version 1.1
        File Checks with    44 Terminals Defined
```

7.8. TCAP Entry Definition Program

TCMAKE is used to create a *.Z3T file. Once created, the ZCPR3 utility LDR can load it into memory at the proper location (command is "LDR filename.Z3T").

Sample run of TCMAKE:

```
B4:SCR2>tcmake //
TCMAKE, Version 1.0
TCMAKE - Create a Z3T File
Syntax:
        TCMAKE outfile  -or-  TCMAKE outfile.typ

where "outfile" is the file to be generated by
the execution of TCMAKE.  If no file type is
given, a file type of Z3T is the default.
```

```
B4:SCR2>tcmake myterm2
TCMAKE, Version 1.0

        ** Z3TCAP Main Menu for File MYTERM2 .Z3T **

Define: 1. Clear Screen Sequence
        2. Cursor Motion Sequence
        3. Clear to End of Line Sequence
        4. Standout Mode Sequences
        5. Terminal Init/Deinit Sequences
        6. Arrow Keys
        7. Terminal Name

Status: S. Print Status (Definitions so far)

Exit:   X. Exit and Write File
        Q. Quit and Abort Program without Writing File

Command? 2

Cursor Motion Definition
 1. Timing Delay
 Enter Delay Time in Milliseconds: 5
 2. Enter R if Row/Column or C for Column/Row: R
 3. Enter Equation for Row:    %+
 4. Enter Equation for Column: %+
 5. Enter Prefix Byte Sequence
  Char #1 - Type Char, .=Number, or <CR>=Done: Enter Number: 1bh
  Char #2 - Type Char, .=Number, or <CR>=Done: Char =
  Char #3 - Type Char, .=Number, or <CR>=Done:
 6. Enter Middle Byte Sequence
  Char #1 - Type Char, .=Number, or <CR>=Done:
 7. Enter Suffix Byte Sequence
  Char #1 - Type Char, .=Number, or <CR>=Done:

        ** Z3TCAP Main Menu for File MYTERM2 .Z3T **

Define: 1. Clear Screen Sequence
        2. Cursor Motion Sequence
        3. Clear to End of Line Sequence
        4. Standout Mode Sequences
        5. Terminal Init/Deinit Sequences
        6. Arrow Keys
        7. Terminal Name

Status: S. Print Status (Definitions so far)

Exit:   X. Exit and Write File
        Q. Quit and Abort Program without Writing File

Command? 6
```

```
Arrow Key Definition
 Your Terminal's Arrow Keys may be defined ONLY
 if they generate only one character each.  If they
 do, type Y to continue.  If not, type anything else.
        Define Arrow Keys (Y/N)? Y
 Strike the Appropriate Arrow Key
 1. Arrow UP?    ^K
 2. Arrow DOWN?  ^V
 3. Arrow RIGHT? ^L
 4. Arrow LEFT?  ^H

        ** Z3TCAP Main Menu for File MYTERM2 .Z3T **

Define: 1. Clear Screen Sequence
        2. Cursor Motion Sequence
        3. Clear to End of Line Sequence
        4. Standout Mode Sequences
        5. Terminal Init/Deinit Sequences
        6. Arrow Keys
        7. Terminal Name

Status: S. Print Status (Definitions so far)

Exit:   X. Exit and Write File
        Q. Quit and Abort Program without Writing File

Command? S

        ** Z3TCAP Status for File MYTERM2 .Z3T **

Review: 1. Clear Screen Definition
        2. Cursor Motion Definition
        3. Clear to End of Line Definition
        4. Standout Mode Definition
        5. Terminal Init/Deinit Definition
        6. Arrow Key Definition
        7. Terminal Name Definition

Exit:   X. Exit to Main Menu

Command? 1
Review of Clear Screen Definition
 1. Timing Delay = 0 Milliseconds
 2. Clear Screen Sequence:
  (1) ^[  1BH    (2) *  2AH
        Strike Any Key to Continue -
        ** Z3TCAP Status for File MYTERM2 .Z3T **

Review: 1. Clear Screen Definition
        2. Cursor Motion Definition
        3. Clear to End of Line Definition
        4. Standout Mode Definition
        5. Terminal Init/Deinit Definition
        6. Arrow Key Definition
        7. Terminal Name Definition

Exit:   X. Exit to Main Menu

Command? 2

Review of Cursor Motion Data
 1. Timing Delay = 5 Milliseconds
 2. Row or Column First: R
 3. Row Equation:    -->%+ <--
 4. Column Equation: -->%+ <--
 5. Prefix Byte Sequence:
  (1) ^[  1BH    (2) =  3DH
 6. Middle Byte Sequence:
  -- Empty --
 7. Suffix Byte Sequence:
  -- Empty --
        Strike Any Key to Continue -

        ** Z3TCAP Status for File MYTERM2 .Z3T **

Review: 1. Clear Screen Definition
        2. Cursor Motion Definition
        3. Clear to End of Line Definition
        4. Standout Mode Definition
        5. Terminal Init/Deinit Definition
        6. Arrow Key Definition
        7. Terminal Name Definition

Exit:   X. Exit to Main Menu

Command? 6

Review of Arrow Key Definitions
 1. Arrow UP =    ^K
 2. Arrow DOWN =  ^V
 3. Arrow RIGHT = ^L
 4. Arrow LEFT =  ^H
        Strike Any Key to Continue -
        ** Z3TCAP Status for File MYTERM2 .Z3T **

Review: 1. Clear Screen Definition
        2. Cursor Motion Definition
        3. Clear to End of Line Definition
        4. Standout Mode Definition
        5. Terminal Init/Deinit Definition
        6. Arrow Key Definition
        7. Terminal Name Definition

Exit:   X. Exit to Main Menu

Command? X

        ** Z3TCAP Main Menu for File MYTERM2 .Z3T **

Define: 1. Clear Screen Sequence
        2. Cursor Motion Sequence
        3. Clear to End of Line Sequence
        4. Standout Mode Sequences
        5. Terminal Init/Deinit Sequences
        6. Arrow Keys
        7. Terminal Name

Status: S. Print Status (Definitions so far)

Exit:   X. Exit and Write File
        Q. Quit and Abort Program without Writing File

Command? X
  Selected Terminal is: Rick's Terminal  -- Confirm (Y/N)? Y
File MYTERM2 .Z3T Created
```

7.9. TCAP Entry Selection Program

TCSELECT is used to select a terminal from the standard Z3TCAP file. The selected terminal may be loaded directly into memory or a *.Z3T file may be created. If a *.Z3T file is created, the ZCPR3 utility LDR can load it into memory at the proper location (command is "LDR filename.Z3T").

Sample run of TCSELECT:

```
B4:SCR2>tcselect //
TCSELECT, Version 1.0
TCSELECT - Select Entry from Z3TCAP.TCP
Syntax:
        TCSELECT outfile  -or-  TCSELECT outfile.typ

where "outfile" is the file to be generated by
the execution of TCSELECT.  If no file type is
given, a file type of Z3T is the default.

Syntax:
        TCSELECT

where this alternate form may be used to store
the Z3TCAP entry for the selected terminal directly
into the Z3 Environment Descriptor.
```

Example 1: Create MYTERM.TCP

```
B4:SCR2>tcselect myterm
TCSELECT, Version 1.0

** Terminal Menu 1 for Z3TCAP Version 1.1  **

A.  AA Ambassador           K.  Concept 100
B.  ADDS Consul 980         L.  Concept 108
C.  ADDS Regent 20          M.  CT82
D.  ADDS Viewpoint          N.  DEC VT52
E.  ADM 2                   O.  DEC VT100
F.  ADM 31                  P.  Dialogue 80
G.  ADM 3A                  Q.  Direct 800/A
H.  ADM 42                  R.  General Trm 100A
I.  Bantam 550              S.  Hazeltine 1420
J.  CDC 456                 T.  Hazeltine 1500

Enter Selection, + for Next, or ^C to Exit - +

** Terminal Menu 2 for Z3TCAP Version 1.1  **

A.  Hazeltine 1510          K.  P Elmer 1200
B.  Hazeltine 1520          L.  SOROC 120
C.  H19 (ANSI Mode)         M.  Super Bee
D.  H19 (Heath Mode)        N.  TAB 132
E.  HP 2621                 O.  Teleray 1061
F.  IBM 3101                P.  Teleray 3800
G.  Micro Bee               Q.  TTY 4424
H.  Microterm ACT IV        R.  TVI 912
I.  Microterm ACT V         S.  TVI 920
J.  P Elmer 1100            T.  TVI 950

Enter Selection, - for Last, + for Next, or ^C to Exit - +
** Terminal Menu 3 for Z3TCAP Version 1.1  **

A.  VC 404
B.  VC 415
C.  Visual 200
D.  WYSE 50

Enter Selection, - for Last, or ^C to Exit - -

** Terminal Menu 2 for Z3TCAP Version 1.1  **

A.  Hazeltine 1510          K.  P Elmer 1200
B.  Hazeltine 1520          L.  SOROC 120
C.  H19 (ANSI Mode)         M.  Super Bee
D.  H19 (Heath Mode)        N.  TAB 132
E.  HP 2621                 O.  Teleray 1061
F.  IBM 3101                P.  Teleray 3800
G.  Micro Bee               Q.  TTY 4424
H.  Microterm ACT IV        R.  TVI 912
I.  Microterm ACT V         S.  TVI 920
J.  P Elmer 1100            T.  TVI 950

Enter Selection, - for Last, + for Next, or ^C to Exit - T

  Selected Terminal is: TVI 950                  -- Confirm (Y/N)? N

** Terminal Menu 2 for Z3TCAP Version 1.1  **

A.  Hazeltine 1510          K.  P Elmer 1200
B.  Hazeltine 1520          L.  SOROC 120
C.  H19 (ANSI Mode)         M.  Super Bee
D.  H19 (Heath Mode)        N.  TAB 132
E.  HP 2621                 O.  Teleray 1061
F.  IBM 3101                P.  Teleray 3800
G.  Micro Bee               Q.  TTY 4424
H.  Microterm ACT IV        R.  TVI 912
I.  Microterm ACT V         S.  TVI 920
J.  P Elmer 1100            T.  TVI 950

Enter Selection, - for Last, + for Next, or ^C to Exit - S

  Selected Terminal is: TVI 920                  -- Confirm (Y/N)? Y

File MYTERM  .Z3T Created

     -- Example 2: Select terminal and store it in memory
B4:SCR2>tcselect
TCSELECT, Version 1.0
```

```
** Terminal Menu 1 for Z3TCAP Version 1.1  **

A.  AA Ambassador           K.  Concept 100
B.  ADDS Consul 980         L.  Concept 108
C.  ADDS Regent 20          M.  CT82
D.  ADDS Viewpoint          N.  DEC VT52
E.  ADM 2                   O.  DEC VT100
F.  ADM 31                  P.  Dialogue 80
G.  ADM 3A                  Q.  Direct 800/A
H.  ADM 42                  R.  General Trm 100A
I.  Bantam 550              S.  Hazeltine 1420
J.  CDC 456                 T.  Hazeltine 1500

Enter Selection, + for Next, or ^C to Exit - +

** Terminal Menu 2 for Z3TCAP Version 1.1  **

A.  Hazeltine 1510          K.  P Elmer 1200
B.  Hazeltine 1520          L.  SOROC 120
C.  H19 (ANSI Mode)         M.  Super Bee
D.  H19 (Heath Mode)        N.  TAB 132
E.  HP 2621                 O.  Teleray 1061
F.  IBM 3101                P.  Teleray 3800
G.  Micro Bee               Q.  TTY 4424
H.  Microterm ACT IV        R.  TVI 912
I.  Microterm ACT V         S.  TVI 920
J.  P Elmer 1100            T.  TVI 950

Enter Selection, - for Last, + for Next, or ^C to Exit - T

  Selected Terminal is: TVI 950                 -- Confirm (Y/N)? Y

 ZCPR3 Environment Descriptor Loaded
```

8. D O C U M E N T A T I O N

The documentation on the ZCPR3 System is available in the following forms:

1. This Installation Manual describes some of the concepts of the ZCPR3 System and gives details on the steps to take to install a ZCPR3 System on a target computer.

2. The ZCPR3 Utilities contain internal documentation on themsevles which is displayed by typing the name of the utility followed by two slashes (eg, "ERASE //" gives the syntax and options of the ERASE command).

3. Some ZCPR3 Utilities, such as SH, are intended to be executed continuously, and, as such, provide documentation on themselves by '?' or 'H' commands (eg, VFILER supports both commands).

4. Details on all of the ZCPR3 Utilities and more are provided by the online documentation system in the form of HLP files. The file "ZCPR3.HLP", which can be read by typing the command "HELP ZCPR3" on a fully-installed ZCPR3 system, provides a top-level index for all HLP files.

5. A book on ZCPR3 is to be published by New York Zoetrope, Inc. It will be available commercially thru a number of sources, such as book stores, and Echelon, Inc (see the references), and computer clubs are welcome to negotiate for rights to sell it for fund-raising purposes. This book will be a comprehensive reference on the ZCPR3 System.

9. R E F E R E N C E S

The following sections outline other sources of information on ZCPR2, ZCPR3, SYSLIB2, and SYSLIB3 that may be useful to the installers and users of ZCPR3.

9.1. ZCPR2 and SYSLIB2 Publications and Documentation

9.1.1. ZCPR2 Manuals

Conn, Richard. _Concepts Manual for ZCPR2 -- Z80 Command Processor Replacement, Version 2_, Manual Revision 0, 3 February 1983, 21 pages.

Conn, Richard. _Installation Instructions for ZCPR2 -- Z80 Command Processor Replacement, Version 2_, Manual Revision 0, 2 February 1983, 48 pages.

Conn, Richard. _Rationale Manual for ZCPR2 -- Z80 Command Processor Replacement, Version 2_, Manual Revision 0, 26 February 1983, 65 pages.

Conn, Richard. _User's Guide for ZCPR2 -- Z80 Command Processor Replacement, Version 2_, Manual Revision 0, 4 February 1983, 138 pages.

9.1.2. SYSLIB2 Manuals

Conn, Richard. _SYSLIB User and Reference Manual for SYSLIB Version 2.4_, 4 February 1983, 112 pages.

Conn, Richard. _User's Guide to SYSLIB 2.3_, Revision B, 14 December 1982, 56 pages.

9.1.3. Software Upgrades to SYSLIB2 and ZCPR2

Four upgrades to SYSLIB2 and ZCPR2 were issued. There were all written by Richard Conn, and their dates are:

```
 4 March 1983
30 March 1983
30 April 1983
22 June  1983
```

9.1.4. Sources

Contact the following sources for copies (on disk or hard copy, depending on the source) of the above-mentioned documentation.

SIG/M (disks)
New York Amateur Computer Club (hardcopy)
SIMTEL20 (DDN access)

9.1.5. CP/M Books

The following books on CP/M are recommended.

Hogan, Thom. _Osborne CP/M User Guide_, Osborne/McGraw-Hill, 1981, 283 pages.

Johnson-Laird, Andy. _The Programmer's CP/M Handbook_, Osborne/McGraw-Hill, 1983, 501 pages.

9.2. ZCPR3 Sources

The following are the addresses of sources for information on ZCPR2, ZCPR3, SYSLIB2, and SYSLIB3.

9.2.1. Selected Computer Clubs

SIG/M provides a library of software on a number of formats of floppy disk. They are a source for the ZCPR2, SYSLIB2, ZCPR3, and SYSLIB3 software.

The New York Amateur Computer Club has published hardcopy of the ZCPR2 and SYSLIB2 documentation.

9.2.1.1. ACGNJ and SIG/M

SIG/M-Amateur Computer Group of New Jersey
PO Box 97
Iselin, NJ 08830

9.2.1.2. New York Amateur Computer Club

New York Amateur Computer Club, Inc
PO Box 106
New York, NY 10008

9.2.2. Echelon, Inc.

Echelon, Inc
101 First Street
Los Altos, CA 94022
(415) 948-5321

Echelon, Inc has been selected as exclusive agent for licensing commercial uses of ZCPR3. Companies who wish to incorporate ZCPR3 into their products should contact Echelon for licensing arrangements.

Echelon also distributes ZCPR3 (including SYSLIB3) for non-commercial use. Individual users may obtain copies of the programs for essentially cost of disks plus handling and mailing expenses.

Echelon will soon provide a computerized bulletin board service in support of ZCPR3. Information on how to acquire a copy of the system, upgrades and changes to the system, and general user/creator feedback and communication will be supported by this bulletin board.

9.2.3. New York Zoetrope, Inc.

New York Zoetrope, Inc.
Suite 516
80 East 11th St.
New York, NY 10003
(212) 420-0590

Cable: NYZOETROPE, N.Y.
Source: TCN 121

New York Zoetrope, Inc, is the publisher of the hardcopy documentation on ZCPR3 and SYSLIB3. This documentation will be released before 4th quarter of 1984. Contact New York Zoetrope for details.