BackGrounder™ ii

© Copyright 1986
Plu*Perfect Systems
P.O. Box 1494
Idyllwild, CA 92349

Copyright

BackGrounder ii, its core utilities and modules are copyrighted by Bridger Mitchell and Plu*Perfect Systems. Your single-user license entitles you to use them on your computer(s), make back-up copies for your own use, and demonstrate them to others. You are not entitled to make copies of any copyrighted material, or this manual, for others. That infringes our copyright and violates your licensing agreement with us.

However, the clearly labeled *BackGrounder ii demonstration disk* may be freely copied and distributed to others who are interested in evaluating BackGrounder ii, provided it is copied intact and unaltered and all copyright notices are displayed, that no charge of any type is made for the disk, and that it is not distributed with other materials for which a charge is made without the express written permission of Plu*Perfect Systems. The software on this demonstration disk remains copyrighted by Plu*Perfect Systems. The software on this demonstration disk remains copyrighted by Plu*Perfect Systems and may not be distributed for any other purpose.

Disclaimer of Warranty

Plu*Perfect Systems makes no representations or warranties with respect to contents hereof and specifically disclaims any implied warranties of merchantability or fitness for an particular purpose. This manual and accompanying software are sold â€œas isâ€ and Plu*Perfect Systems will not in any event be liable for direct, indirect, incidental or consequential damages resulting from any defect, error or failure to perform.

However, Plu*Perfect Systems warrants the diskettes on which the BackGrounder ii is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your purchase receipt.

Trademarks

BackGrounder ii, The Backgrounder, and DateStamper (Plu*Perfect Systems); Perfect Writer (Perfect Software); CP/M, RMAC, Link-80 (Digital Research, Inc.); WordStar (MicroPro, Inc.); dBase II (Ashton-Tate); Smartkey (FBN Software); Sidekick, Turbo Pascal (Borland International); Kaypro (Kaypro Corporation); Uniform (Microsolutions); MS-DOS (Microsoft Corporation); Times, Helvetica (Allied Corporation).

This Manual

This manual was prepared using FinalWord II word processing software by Mark of the Unicorn. It was printed on an Apple LaserWriter. The main body of text is in Times® typefaces, the headings are in Helvetica® typefaces and most examples are in Courier typefaces.

Foreword

A Bit of History

Improvements and extensions of the CP/M command processor are due to contributions from many. Ron Fowler, Keith Peterson, Frank Wancho, and others developed ZCPR â€“ the Z80 Command Processor Replacement. Rich Conn greatly expanded on this approach, adding resident buffers in ZCPR3 for communication between sequential tasks.

BackGrounder ii derives its name and early concepts from The Backgrounder, a Plu*Perfect Systems product for Kaypro computers first introduced in 1983 before Sidekick reached the PC market. The Backgrounder was perhaps the first extension of CP/M to provide user access to some background tasks (those commands built into the command processor) while a program is running.

To this basic concept BackGrounder ii has added the capability for complete *context* or *task switching*. Using virtual memory, it is able to save the complete state of one task, switch to a second, and maintain a versatile set of built-in commands in a series of overlays.

My introduction to the inner workings of CP/M came as a rude shock—I naively attempted to add a message printout to the BIOS of my first computer and learned that the BDOS is not re-entrant! (MS-DOS including version 3.x has not advanced matters in this regard.) Several years later, BackGrounder ii represents my solution to that fundamental restriction.

Acknowledgements

Ed Park, Hal Bower, and Ed Plate are the real veterans of BackGrounder ii testing. The suffered the early system crashes and tenaciously persevered to get me the date needed for debugging.

Jay Sage and Bruce Morgen enlightened me about the finer points of ZCPR3, suggested key BackGrounder ii improvements, and tested final versions.

Darell Bethune, Kevin Bowler, Kim Korner, and Greg Platt suggested useful additions.

Derek McKay and I have conceived and designed the two BackGrounders in our years together as Plu*Perfect Systems.

Many thanks to all of them!

Bridger M. Mitchell
December, 1986

Table of Contents

List of Figures

Chapter 1

Introduction

This manual will introduce you to and be your source of reference for **BackGrounder ii**, a powerful extension of the CP/M operating system for Z80 microcomputers.

We've organized the material in the order you're most likely to need it. This Introduction comes first, followed by instructions on how to install BGii, and a Getting Started chapter that will acquaint you with the basic principles of how it works.

When you've read that far you will be ready to use BackGrounder ii for regular computing. You can dip into other sections of the manual as the occasion arises.

The next four chapters describe how to use the advanced BGii commands, with examples. Chapter 4 covers BGii's unique capability to switch between tasks. Chapter 5 describes the commands for moving data from your screen to another program, a file, or the printer. In Chapter 6 we explain how to create macro definitions consisting of strings of text or characters and attach them to convenient keys. And Chapter 7 describes the Spooler, which you can use to print files while running other programs.

All of BackGrounder ii's commands are gathered together in the command reference chapter. Each command is listed alphabetically with its functions and usage.

The final chapter covers configuration options. It is followed by a series of appendices. They contain a list of error messages, technical information, source code, and other specialized items. Most users won't need to refer to these, but if you have a problem please check the appendices for possible solutions.

To find information quickly, consult the comprehensive index at the back of the manual, the table of contents, or the header at the top of each page.

This manual should answer most, if not all of your questions, get you started using BGii, and serve as a ready reference guide. But basically, BGii is designed to be used without a manual—the more elaborate BackGrounder ii functions are presented from a menu, and all BGii commands are described by on-line help, available by just typing `HELP`.

1.1 What is BackGrounder ii?

BackGrounder ii is software that adds new dimensions to your CP/M computer. It consists of a group of powerful new commands that can be used *while running* a program as well as from the CP/M prompt, and it gives you the capability of running *two* programs at essentially the same time.

In more technical terms, BGii comprises a small “resident” program that replaces the regular CP/M command processor, a “swap file” that contains additional parts of the program and severs as temporary storage, and a group of supporting utility programs. BackGrounder ii runs on all Z80 computers that use the basic CP/M 2.2 disk operating system, and on most computers using the ZRDOS system. It is compatible with ZCPR3-type replacements for the standard command processor.

BGii should be compatible with all application programs that adhere fully to CP/M 2.2 standards. Programs that use absolute system addresses or patch the operating system may require modification. There may be conflicts with other resident system extensions such as add-on hard or ram disk drivers.>[1]

1.2 What BGii does for You

BackGrounder ii lets you do useful tasks “behind the back” of the program you are running—hence its name, which we often shorten to just BGii. For example, if you need to be reminded of just what files are on a disk, you can interrupt the program you are running to use the familiar **`DIR`**ectory command, and then resume your work.

BackGrounder ii, with its 37 handy new commands and enhanced old ones, provides these major features:

- help while running any program
- a notepad available at any time for jotting down notes
- a calculator for decimal and hexadecimal arithmetic which can insert results into a program.
- The ability to print files while running you run other programs
- “cut” and “paste” capabilities to move text from one program to another
- “keystroke macros” to replace a single key with an entire string of characters
- the ability to capture printer output in a file for editing or later printing
- most powerful, the ability to run a *second* program, obtain results from it, and then resume your original program.

These enhancements combine to turn your CP/M computer into a more effective and useful too, the equivalent in many cases of two separate computers.

1.3 How BGii Works

BGii is started by running the LOADBG program. It replaces the standard CP/M command processor in the computer's memory and intercepts certain operating system functions for special processing. BGii uses very little main memory, keeping most of its code in a special “swap file” called !!BG.SWP.

Regular (foreground) CP/M commands and programs run normally, except that BGii keeps watch for a special <SUSPEND> key (CTRL-^). When you type this key, BGii temporarily interrupts the running program and waits for a “background” CP/M command. After these background CP/M commands have been processed, a second press of the <SUSPEND> key resumes the original program.

When you type the special **`SWAP`** command, BGii saves all of the memory used by your current program, prompts for a new CP/M command, and loads your second program. You can run that program to completion, or suspend it at any point. In either case, you can return to the first program, resuming exactly where you left off.

When it suspends a program, BGii saves the screen n order to restore it when that program is resumed. Similarly, BGii reads the screen in order to “cut” out a region of text, and to let you type into a notepad. These and the screen dump are the only features for which BGii requires a “smart” terminal and screen driver that can determine exactly what characters are on the screen at any moment; all other BGii commands operate without a screen driver.

When the SPOOLER utility is loaded into memory, BGii prints files concurrently, while any program is running.

1.4 System Requirements

BackGrounder ii requires a Z80 or equivalent processor (National's NSC800 or Hitachi's HD64180); either the standard Digital Research CP/M 2.2 BDOS or ZRDOS version 1.1, 1.2, 1.3 or 1.7;[2] and a terminal with direct cursor addressing, clear to end-of -line, and clear to end-of-screen functions.

BGii requires a complete and correct implementation of CP/M 2.2 BIOS services (IOBYTE is optional).

BGii uses 4.75k of memory, 2K of which replaces the standard command processor. It is compatible with ZCPR3 systems, and when running with such a system BGii's memory requirement can be reduced to just 2.25K by loading segments of the BGii code into unneeded ZCPR3 buffers.

BGii uses a swap file of approximately 100K. For best performance, BGii should be run with the swap file on a ramdisk or hard disk. BGii will also run on a floppy disk system, but slowly, and we don not recommend it.

In order to redraw the screen, and perform cut-and-paste, screen dump and notepad functions, BGii requires either a terminal with a transmit-character or transmit-region function, or a video-mapped memory console.[3]

BGii automatically installs itself into the running operating system, and removes itself on command. It entails no permanent changes to your system.

1.5 Terminology

Where possible, we have used the following conventions throughout the manual:

- Unless needed for clarity, BGii's prompts (e.g., A0>) are not shown
- The carriage-return, which terminates BGii commands, is not shown
- Commands you can type are in **`BOLD`** letters, using Courier or a monospace typeface.
- `Output you can see on your screen is in Courier or another monospace typeface`
- Text to be replaced by the user is in *italics*
- Control keys are shown as CTRL-C (for example)
- Special keys are shown in angle brackets, e.g., <SUSPEND>
- Optional text is shown in square brackets, e.g. [ON]
- *filename* means a valid unambiguous CP/M filename, with optional drive and/or user number, e.g. LETTER.TXT, A:LETTER.TXT, A15:LETTER.TXT.
- *filespec* means a valid CP/M filename, with optional wild cards, and optional drive and/or user number, e.g. L*.TXT, A15:LET*.T??

Chapter 2

Installation

BGii is designed to run on a wide variety of CP/M computers. This chapter provides instructions to enable you to install BGii in a basic configuration and start using it for regular computing. Once you've become acquainted with BGii's operation, you will probably want to further customize some of its features. Further details on configuration are covered in Chapter 9.

There is quite a bit of information here, but you are unlikely to need all of it, and most of the installation steps need to be done only once.

2.1 Before You Begin

2.1.1 Backing Up BGii

Before installing BackGrounder ii, you should make a working copy of your BGii distribution disk(s). If you own a floppy-based system, you will need to construct a working diskette in the largest capacity format your computer accepts. After formatting a fresh disk, copy all files onto it using PIP, a similar file transfer program or, if appropriate, a disk copy program. You will probably also want to SYSGEN a bootable system onto this disk.

If you own a hard disk, copy the distribution disk into an empty user area using a file transfer program such as PIP. At this point you do not need to copy any files to A0.

You should then store your master BackGrounder ii disk in a safe place.

2.1.2 Last-Minute Information

We know that material may be added to BackGrounder ii after this manual is printed. To check for any additions, please *print out* a copy of the file RELEASE.NOT from the working BGii disk. It contains the latest information and any changes, and should be referred to as you install BGii.

2.1.3 Installation Outline

You are now ready to install BGii. The following steps are required:

1. Run **SETTERM** to install the terminal definitions of your computer into the BGii programs. This is a one-time step.
2. Run **SETBG** to configure BGii to your own and your computer's requirements. Initially, only a bare minimum of options are set. A full chapter is devoted later in this manual to all possible configuration options.
3. Run **PUTBG** to create a swap file. This swap file is your “virtual memory” while BGii is running. Again, this is a one-time step, except when the swap file is placed on a ramdisk that loses its contents with power down.
4. Run **LOADBG** to load BackGrounder ii into memory (and the swap file). This step is done each time you start up your machine.

2.2 Terminal Configuration

Terminals come in about as many different flavors as printers, each with its own idea of “standard” control sequences for positioning and moving the cursor, clearing the screen, and deleting regions. Two BGii programs (SETBG and LOADBG) require cursor addressing, home cursor, clear to end-of-screen, clear to end-of-line, insert and delete line functions. You will use **SETTERM** to install your terminal's particular control codes into these BGii programs.

SETTERM obtains its information for most popular terminals from the terminal database file TERMBASE.DAT. Most likely, your terminal is already on the list (or emulates one that's there). If so, just select that terminal number and tell SETTERM to configure both files (SETBG and LOADBG). If not, you can create a new terminal definitions or modify an existing one to match your terminal's characteristics. Having done that, you can install the definitions in the programs.

Instructions for all cases are given below.

2.2.1 Installing a Terminal from the Library

To run SETTERM, simply type

```
SETTERM
```

The program will ask for the drive containing TERMBASE.DAT and then display a menu of the available terminals. If your terminal is in this menu, enter the number of the terminal you wish to use. You will not need to further review or edit the terminal definition.

You should then answer **Y** when prompted to install the definitions for your terminal into the BGii programs SETBG and LOADBG.[4] Follow the on-screen prompts, answering **Y** when asked

```
OK to update (y/n)?
```

Once finished, type **Q** to quit. The BGii utilities are now permanently configure to run with your terminal.

2.2.2 Editing an Existing Terminal Entry

If your terminal emulates closely, but not completely, a terminal in the library, you may wish to edit that library entry to your exact requirements rather than define a new terminal. In this case you would answer **Y** to the question

```
Review or edit the terminal definition?
```

asked just after you load SETTERM. The contents of the database for that terminal will then be displayed, and you will be given the opportunity to change any entries.

After editing them to you requirements, you will then be asked whether you wish to install the new terminal characteristics into the utilities. If the answer is **Y**, proceed as in the previous section.

2.2.3 Defining a New Terminal Entry

If your terminal isn't like one of those on the list, you've probably had experience in configuring it for another program. SETTERM Lets you define a new terminal entry to enter your terminal characteristics. Just type

```
SETTERM
```

and answer **Y** to the question

```
Review or edit the terminal definition?
```

You should then give appropriate answers to the on-screen prompts. Two editing screens will be displayed—one for cursor addressing data, and the other for terminal control strings. After filling in all the relevant information (this should be in the user's guide for your terminal), you can then go on to install your terminal's characteristics into the BGii programs as described above.

If you've done your own configuration for your terminal, you may later find that a sequence or two was mis-entered. SETTERM can be easily reused to edit and update the definitions. Rerun SETTERM, select the number of the terminal you defined, and edit as necessary. (When BGii is running, the **`TYPE`** command makes a good test instrument.)

We and future users will appreciate it if you would also send us the definitions you work out. We can then include them in the database for future releases.

2.3 Initial Configuration of BGii

BGii can be readily configured with a wide variety of options, using the **`SETBG`** program. In this section we will cover only those settings that are necessary to get BGii up and running. You can later rerun SETBG to change these settings and take advantage of the other options.[5]

To use SETBG, just type

```
     SETBG
```

The program will first ask if it is correctly configured for the terminal you are using. If not, exit and use SETTERM as described in the previous section.

SETBG operates by offering you menu choices. The menus are organized in a tree-like structure, beginning at the main menu, which looks like this:

```
     0. exit to CP/M
     1. change parameters
     2. update file with new or changed parameters
     3. print parameters
```

You first select option 1, which will display the following parameter menu:

```
     0. main menu
     1. hardware     - drivers/ speed/ overlay address
     2. control      - password/ drives/ directories/ path
     3. display      - prompt/ separator/ printer-test/ DIR and DATE format
     4. keys         - suspend/ alternate/ delay/ remapping
     5. zcpr3        - command-run/ use of memory
     6. start up     - auto-execute command
```

Again, you should select option 1 to reach the hardware menu, which looks similar to the following:

```
     0. previous menu
     1. terminal:        Kaypro '83
     2. screen driver:   NOT INSTALLED
     3. fn. key drivers  NOT INSTALLED
     4. cpu speed (mhz)  4
     5. overlay at:      4000H
```

2.3.1 Hardware Options

s2.3.1.1 Terminal

You should first verify that item 1 (terminal) corresponds to your terminal. This will have been set by SETTERM. If it is not, exit SETBG and run SETTERM to install your terminal characteristics into SETBG (see Terminal Configuration above).

2.3.1.2 Screen driver

Next select the screen driver option (2), answering **`Y`** and entering the drive and user area of the BGii working files when prompted. This will list the available screen drivers. Check the names of the screen drivers against the list in the BGii RELEASE.NOT file. If your terminal or computer is one of these listed, select it; if not, enter 0.[6]

2.3.1.3 Function Key Driver

The third option to check is the function key driver (3). After typing **`3`**, enter **`Y`** and the drive/user of the BGii working files when prompted. This will list the available function key drivers. Check these filenames against the corresponding list in RELEASE.NOT, and make the appropriate choice, entering **`0`** if a driver for your computer is not present.[7]

2.3.1.4 CPU Speed

If the displayed CPU speed is not correct, select option 4 to change it.

2.3.2 Control Options

You should now return to the parameters menu by typing **`0`** and select the control option (2). A menu similar to the following will appear:

```
0. previous menu
1. password          PASSWORD
2. password for all  N
3. swap drive        A:
4. max. drive        G:
5. max. user         15
6. HELP directory    D15:
7. JOT directory     D0:
8. path              $$:, $0:, A0:, D0:
```

At this time, only the swap drive and help directories need to be set.

2.3.2.1 Swap File Drive (Option 3)

Enter the drive letter that you plan to use for your swap file. Be sure to be consistent with your usage when running PUTBG (see the next section in this chapter). This should be your fastest drive with at least 100K of free space on it. If possible, use a hard disk or ramdisk drive.

2.3.2.2 Help Directory (Option 6)

The help directory contains BGii's two supplied help files. This may be any convenient drive and user area on a permanently available drive. If you are specifying a ramdrive, you will need to copy the help files to the ramdisk on start-up each day.

2.3.3 Installing Your Selections

You should now return to the Parameters Menu (**`0`**) and then to the main menu (**`0`**). The defaults for the remaining options are adequate for this time. Now you are ready to write out the configured file, by selecting option 2, sub option 1 (update existing file) and option 0 (main menu). When finished, select option 0 from the main menu to return to CP/M.

2.4 Creating and Arranging Files

This section deals mainly with the swap file, help files, and files used to load BackGrounder ii. For a full list of files and their functions, see Appendix J.

2.4.1 Creating the BGii Swap File

The final step, before loading BGii, is to use the **`PUTBG`** program to create the BGii swap file. The swap file requires about 100K. The exact amount depends on the size of your operating system, and will automatically be determined by PUTBG.[8]

When running SETBG (see previous section), you will have chosen a drive to hold the swap file—it must have at least 100K free. For fastest performance, a ramdisk is best. Otherwise, a hard disk works well. You can use a floppy disk, but swapping will be slow, and you *must* keep that disk in the drive as long as BGii is in use.

PUTBG creates a swap file, called !!BG.SWP. Because the swap file must be at *exactly* the right place in the directory and the disk, PUTBG will first move any files that are in those locations before writing the swap file, and will display messages showing what it has moved.

Now type the following command, substituting for X the letter of the drive that is to have the swap file:[9]

```
PUTBG -d=X
```

When PUTBG is done, you can check the results by typing

```
DIR X:
```

The !!BG.SWP file should be the first or second file listed, followed by the other files.

If the swap file is on a hard or floppy disk, it will remain there, write-protected when you turn off your computer. However, if the swap file is on a ramdisk, you will need to rerun PUTBG each time you power up your computer before loading BGii.

2.4.2 Help Files

When you ran SETBG, you designated the drive and user area for BGii to find its help files—BG.HLP and BGINFO.HLP—as well as any BGii help files you may wish to create yourself.[10] You should make sure, before loading BGii, that you have copied the relevant help files to that drive and user number. Should that area be on a ramdisk, you will need to copy the help files to the ramdisk each time you switch your computer on. For optimal arrangement of help files see Figure 2.1 in this chapter.

2.4 3 Arranging Other Files

Permanent arrangement of other BGii files depends on your computer configuration. Suggestions for optimal arrangement of files is shown in Figure 2.1. The only thing to remember is to keep the files LOADBG.COM and BG.REL together on the same drive and user number. These are the files used to load BGii. Once BGii is loaded, these two files no longer need to be accessible—i.e., the disk containing them may be removed from your computer.

Floppy Disk Only

!!BG.SWP	*Not Recommended*. On fastest drive, disk must remain mounted.
BG.LP/BGINFO.HLP	On same drive as swap file.
LOADBG.COM/BG.REL	On start-up floppy disk. Remove after start-up.
Other BGii Files	On working disk located on search path.

Hard Disk (No Ramdisk)

!!BG.SWP	On hard disk.
BG.HLP/BGINFO.HLP	On hard disk.
LOADBG.COM/BG.REL	On default hard disks.
Other BGii Files	On hard disk, any DU: on search path.

Hard Disk and Ramdisk

!!BG.SWP	On ramdisk.
BG.HLP/BGINFO.HLP	On hard disk.
LOADBG.COM/BG.REL	On default hard disk.
Other BGii Files	On hard disk, any DU: on search path.

Ramdisk Only

!!BG.SWP	On ramdisk.
BG.HLP/BGINFO.HLP	On ramdisk.
LOADBG.COM/BG.REL	On start-up floppy. Remove after start-up.
Other BGii Files	On ramdisk if sufficient space, else on working disk on search path.

Figure 2.1: Suggested File Arrangement

2.5 Loading BackGrounder ii

Now you are ready to start up BackGrounder ii. To do so, log into the drive containing LOADBG.COM and BG.REL and type:

```
LOADBG
```

The program will read the BG.REL file, relocate and install the BGii code, and display a summary of the configured parameters similar to the following:

BackGrounder ii configuration summary

```
 BGii version: 1.00
     serial #: 12345678
          dos: Standard CP/M 2.2 BDOS
       prompt: > fence: | command separator: ;
    swap file: drive A:, 0340H sectors (104K)
     terminal: Kaypro '83
screen driver: Kaypro '83 driver v 1.1
```

```
        function-key driver: Kaypro 83 & 84 fn.key overlay v. 0.4
                   password: PASSWORD
                       keys: <1E> for <SUSPEND>, \ for
                       path: $$:, $0:, A0:, D0:
                directories D0: for JOTPAD, D15: for HELP
```

memory map

```
             tpa: 0100H        CCP entry: DB03H
     BackGrounder: D000H              DOS: E306H
      800H buffer: D300H             BIOS: F100H
      280H buffer: D080H
```

Below it will be the familiar CP/M prompt for the logged-in drive, plus the user number, e.g. A0>. BackGrounder ii is now running, and the next chapter reviews how it is used.

Chapter 3

Getting Started

3.1 BGii Commands

BackGrounder ii consists of a large set of useful and powerful commands â€“ thirty-seven in all â€“ most of which can be used both from a CP/M prompt and from within any program. These commands fall into the following general categories:

1. General Commands. These include expanded versions of CP/M's familiar commands, ZCPR3 commands, and other useful tools:
   BG, **DATE**, **DIR**, **ECHO**, **ERA**, **FEED**, **FIND**, **GET**, **GO**, **HELP**, **JUMP**, **LIST**, **NDR**, **NOTE**, **OCP**, **PEEK**, **POKE**, **PRINTR**, **REN**, **RESET**, **SAVE**, **TIME**, **TYPE**, **USER**, **WHL**, **WHLQ**.
2. Task Switching Commands. These let you freely move back and forth between any two programs:
   FLIP, **SWAP**.
3. Data Transfer Commands, for saving and moving text from any screen and incorporating it in any other:
   CUT, **JOT**, **PASTE**, **SCREEN**.
4. Key Definition Commands. These let you attach definitions to any key on your computer's keyboard:
   KEYS, **SHIFT**.
5. Print Spooler Commands, for controlling the redirection of list output:
   FORMS, **SPOOL**.
6. Calculator Command. This gives you an on-screen decimal and hexadecimal calculator, with the ability to incorporate its most recent results into any program:
   CALC.

An alphabetical list of these commands, with a full description of their functions, is contained in Chapter 8. In addition, separate chapters are devoted to task switching, data transfer, key definitions and print spooling.

In this chapter, we introduce you to a few of the more familiar and straightforward commands, and demonstrate how BGii lets you use them both in the foreground—at the CP/M prompt—and from the background—while running a program. To try the examples in this chapter, you should first have gone through the installation procedures outlined in the previous chapter.

3.2 Using BGii Commands at CP/M Prompt

Once you have run LOADBG, BackGrounder ii is active. It will continue in memory until you Cold Boot your computer, switch off the power, or use the **BG OFF** command described below. While BGii is present, your screen will look much as usual. There is the usual CP/M prompt, and you may load and run programs in the normal way. The difference is that you now have 37 useful and powerful commands at your fingertips.

This section demonstrates the commands **HELP**, **DIR**, **USER**, **TYPE**, **TIME**, **CTRL-C**, and **RESET**, all at the foreground level—i.e. at the CP/M prompt.

3.2.1 BG – List of Commands

The command **BG** will alphabetically list the current set of allowable commands. To see them, simply type:

```
    BG
```

The screen will appear as follows:

```
Cmds: Foreground A00:CP/M global-keys
bg      calc    cls     cut     date    dir     echo    era     feed    find    forms
flip    go      get     help    jot     jump    keys    list    ndr     note    ocp
peek    poke    printr  ren     reset   save    screen  shift   spool   swap    time
type    user    whl     whlq

swap      -> a00:CP/M    global-keys
```

As you can see, the **BG** command displays your current "state", the list of commands, and the â€œstateâ€ that giving a **SWAP** command would result in. Information includes drive and user, upper/lower and foreground/background states as well as related key definitions. **BG** has other options, which are described in the Command Reference chapter. The most commonly used is:

```
BG OFF
```

which will remove BGii from memory without cold-booting your computer.[11]

3.2.2 HELP – On Line Help Command

All of BGii's commands are listed and explained in the file BG.HLP. This file is set up as an indexed help file, the items of which may be examined by selecting from the initial menu. Make a mental note that **HELP** is *always available*, even when a program is running. Most of the BGii information you will ever need is at your fingertips. To get help, the command is

HELP [*helpfile*]

If the *helpfile* is not specified, the file BG.HLP is used. If you type

```
HELP bginfo
```

you will see the file BGINFO.HLP. You may also use **HELP** to access your own Help files.[12]

The **HELP** command displays an index of subjects for which information is available, each matched with a corresponding letters or number. To display the information, enter the letter or number from the menu. BGii then shows the information one screen at a time. At the bottom of the screen, a one-line menu indicates what keys can be pressed to control the display:

```
EOI 2: Menu Start CR=SP=Next Last Print ^C=exit ==>
```

At the left margin of the menu line the number **2** indicates which screen of information under this index entry is displayed. When **EOI** appears it indicates the End Of Information section—there are no more screens on this menu item.

It is important to remember that the **HELP** command is available at any time. We have only shown how to use it while at the CP/M prompt; however, later in this chapter it will become obvious how to access the HELP information while running a program. This comprehensive on-line information available makes it possible to become familiar with BGii with very little reference to this manual.

3.2.3 DIR – Directory Command

DIR is BGii's directory command. It is very similar to the CP/M directory command,, listing all non-system files in the current drive/user. (The files are listed in the order the directory entries are encountered on the disk.) With BGii, the **DIR** command also gives you file sizes, disk space used and free, and a number of extra command line options.

To obtain a directory of the files of the drive and user you are currently logged into (e.g. E0>), type:

```
DIR
```

This will produce a directory in the following format:

```
!!!TIME&.DAT  12k  |  ANAGRAM .COM   4k  |  DICTSORT.COM   4k  ...
HOMONYMS.TXT   8k  |  HYEXCEPT.TXT   8k  |  HYPHEN  .COM   8k  ...
MARKFIX .COM   4k  |  PF      .COM  36k  |  PFCONFIG.COM  40k  ...
REVIEW  .COM   8k  |  SPELL   .COM   4k  |  TW      .COM   4k  ...
WORDFREQ.COM   4k  |  WS      .COM   4k  |  WS      .INS  52k  ...
PRINT   .BAK   4k  |  WINSTALL.COM  32k  |  FUNCTS  .TXT  20k  ...
CRAYON  .COM  24k  |  PENCIL  .COM  24k  |  SCONFIG .COM  49k  ...
FILT    .COM   4k  |  FILTW   .COM   4k  |  FILTWC  .COM   4k  ...
Drive E: has 2692k used,  284k free, 2976k total.
```

Note that the file size is listed according to the logical block size of your disk. If you disk has 4K block sizes, for example, any file from 1K (or less) to 4K will be listed as 4K. Larger files will be listed as multiples of 4K e.g. a 10K file will be listed as 12K.

You may use the **DIR** command with a number of options on the command line. For example,

```
A0>DIR 15: /F
```

gives the directory of non-system files, without file sizes, Drive A User 15. Full details of the **DIR** options are given in the Command Reference chapter.

3.2.4 USER – Changing User Numbers

The **USER** command lets you move between user numbers 0 – 15, if you have divided up your disk space that way.

Examples:

```
A4>USER
```

logs into A0:

```
A0>USER 15
```

logs into A15:

Note that you can only move between user numbers on the same drive. Once logged into A15>, however, you will move to B15> should you type

```
B:
```

The **USER** command is included to preserve continuity with the CP/M **USER** command. BGii also gives you a much handier way of logging into user numbers on *any* drive. Just type the drive letter, the user number, and a colon (this is called the DU form). For example:

```
A0>B4:
B4>C15:
C15>A0:
```

moves you quickly between drives and user numbers. To change to a different user number on the same drive you can simply type the user number followed by a colon. For example:

```
A0>9:
```

will move you to A9:.

3.2.5 TYPE, LIST – Viewing Files

3.2.5.1 TYPE

One of BGii's most useful commands is its enhanced **TYPE** command, which lets you view any text file. The usage is:

TYPE ***filename***

When you view a file using BGii's **TYPE** command, you have the ability to scroll through the file in a number of ways. A menu with the following information remains at the top of the screen to remind you how to move about the file:

```
>. or SP  => + line
> or CR   => + page
1..9      => +/- pages
G         => goto “string”
^C        => quit
,         => - line
< or TAB  => - page
B         => beginning of file
E         => end of file
L         => scroll screen left
R         => scroll screen right
```

These commands include the ability to scroll forward or backwards line by line or screen by screen, to move forward or backwards 1 to 9 screen pages at a time, to move straight to the beginning or end of a file, and to move sideways from the right to the left margins of a file.

You can also use the **G** command to go straight to a specified string (up to 18 characters) in a file—though only in a forward direction. Lower case letters are case-independent (**a** matches a or **A**); upper-case letters match only upper-case. Precede a control character with **CTRL-Q**.

Finally, the **TYPE** command will also automatically unSQueeze a compressed file, again only in a forward direction.

3.2.5.2 LIST

If you wish to view a text file as hard copy, instead of sending it to the screen, the **LIST** command will send the file to the printer.

Simply type:

```
LIST filename
```

first making sure, of course, that your printer is on line! The text will be printed out in whatever form it appears on your screen, expanding tabs to 8 spaces. A **CTRL-C** will abort printing.

3.2.6 RESET, CTRL-C – Resetting Disk System

There are two interchangeable commands that let you reset your disk system under BackGrounder ii—**RESET** and **CTRL-C**. Both close modified files, reset the disk system and log in Drive A: and the current drive. You should use either of these commands after changing disks.

When you type:

```
RESET
```

or

```
CTRL-C
```

you will see the following message on the screen:

```
Hot Boot BGii vn.nn XXXX
```

where n.nn is the version number and XXXX is the lowest address in hexadecimal used by BGii.

A Hot Boot is BGii's faster version of a Warm Boot because the operating system is not reloaded from disk. Note that typing **RESET** or **CTRL-C** will give you a Hot Boot and retain BGii in memory, whereas *pressing* the RESET button (if you have one) on your computer will cause a Cold Boot, and the removal of BackGrounder ii.

3.3 Using Commands While in a Program

3.3.1 The <SUSPEND> Key

All of the commands mentioned in this chapter are available not only at the CP/M prompt, (A0>, B0>, C0>, etc.), but also from within a program. One of BGii's most useful features is the ability to let you temporarily suspend a program, access the command(s) of your choice—e.g. obtain a directory, view or print another file, rename a file, etc.—and then resume work on your program exactly where you left off.

To do this, BGii has allocated a special key—the <SUSPEND> key—to press whenever you want to temporarily interrupt the program you are running to use a BGii command. This key is **CTRL-^**.[13] When you press the <SUSPEND> key, the program you are working on is temporarily halted, and a CP/M prompt (e.g. A0}) appears on the screen. After this prompt appears, you can then proceed to use nearly all of BGii's 37 commands.

When you press the <SUSPEND> key, your screen will be cleared from the current cursor position. A display very similar to that from the **BG** command will appear.[14]

We call the prompt with a } sign—e.g. A0}, B0}, C0}—**Background CP/M**. Background CP/M refers to the state you reach when pressing the <SUSPEND> key in a program, as opposed to **Foreground CP/M** which is present *before* loading a program. You must be in either Foreground CP/M or Background CP/M to us any of the BGii commands.

Once you have finished using your selected commands, you return to your program by again pressing the <SUSPEND> key (CTRL-^). Your screen will be redrawn exactly as you left it, so long as your copy of BackGrounder ii has the correct screen driver installed.

3.3.2 Using the <SUSPEND> Key: Examples

1. Suppose you wish to obtain a directory of Drive B, while logged into Drive A and running a program. All you would do is:
   a. Press the <SUSPEND> key, and wait for the A0} prompt to appear.
   b. Type

      ```
      A0}DIR B:
      ```

   c. After the directory has appeared and you have the information you require, press <SUSPEND> again to resume your program.
2. If you wish to change disks containing text files, while running a word processing program:
   a. Remove the old disk (first making sure it has no modified files left open), and replace it with the new disk.
   b. Press the <SUSPEND> key.
   c. At the Background CP/M prompt, type

      ```
      RESET
      ```

      or press

      ```
      CTRL-C
      ```

      This will Hot Boot your disk system.

   d. Press <SUSPEND> again to resume your program.
3. The **TYPE** command could be used to look up a telephone number while in the middle of another program:
   a. Press the <SUSPEND> key and wait for the background prompt } to appear.
   b. At the Background CP/M prompt, type

      ```
      TYPE B15:TELEPHON.DIR
      ```

   c. Now use the **G**oto command to search for the name of the person.
   d. Use **CTRL-C** to exit to Background CP/M.
   e. Press <SUSPEND> to resume your program.
4. The **ERA** command could be used to make room on a nearly full disk for a long document that you wish to save from your current word processing session.
   a. Press the <SUSPEND> key and wait for the background prompt } to appear.
   b. At the Background CP/M prompt, type:

      ```
      ERA junkfile
      ```

   c. Press the <SUSPEND> key to return to your word processor and save your precious work.

HELP, **BG** and all other commands mentioned in this chapter can also be easily used while running a program.

3.4 Multiple Commands

Like ZCPR2,3 BGii allows multiple commands to be placed on a single line. The commands are merely separated with a semicolon (configurable with SETBG).

For example,

```
    DIR A:;DIR B:
```

will generate directories of both the A: and B: drives.

Chapter 4

Task-Switching

Perhaps the most powerful of BGii's features is its task-switching ability. This means you can run a *second* program while the original one is still active, and switch between them at will. For example, you may have started running a spreadsheet program and need to interrupt it to write a memorandum. With BGii loaded, you can suspend the spreadsheet, and load your word processing program. When you've finished the memo, you can either exit and resume your spreadsheet where you left off, or simply suspend the memo and switch back to the spreadsheet. If you've adopted the latter course, you can at any time switch back and forth between your spreadsheet and memo.

4.1 The Swap Command

The command that lets you do this is **SWAP**. When you type **SWAP**, you switch from one task to another, without having to exit from either. In effect, it treats your CP/M computer as if it were two computers, sharing one terminal and one set of disk drives. By typing **SWAP**, you switch from one computer to the other.

Where you **SWAP** *from* and *to* depends on your current activity at each computer. The following three examples explain:

1. You have loaded BGii and are logged into Drive B0: (This can be thought of as “COMPUTER 1”). If you type:

   B0>SWAP

   you will swap to the alternate or lower-case “computer 2”, still at CP/M level. This is depicted as:

   b0>

   and referred to as **lower Foreground CP/M**. If you now type:

   b0>SWAP

   you will return to “COMPUTER 1”, again at **upper Foreground CP/M** (the B0> prompt).

2. Suppose you now load a program on “COMPUTER 1” at the B0> prompt—say, **SWEEP**. Once **SWEEP** is loaded, you can suspend it by pressing the <SUSPEND> key (**CTRL-^**) and enter into **upper Background CP/M** (B0}). If you now type:

   B0}SWAP

   you will once again swap to “computer 2”, and, because no program is loaded there, arrive in **lower Foreground CP/M** (b0>). This time, however, when you type:

   b0>SWAP

   you return to the current state on “COMPUTER 1”, which is the **SWEEP** program. The screen will look exactly the same as when you suspended it.[15]

3. Now try another possibility. You already have one program (**SWEEP**) loaded on “COMPUTER 1” and wish to add another program, **PIP**, on “computer 2”. You are at present running **SWEEP** on Drive B.

   First suspend **SWEEP** and enter Background CP/M by pressing the <SUSPEND> key. Now type:

   B0}SWAP

   This switches you to “computer 2” which is still at Foreground CP/M level (b0>).

   At the b0> prompt you are ready to load your second program, **PIP**. To do this, simply type:

   b0>PIP

   as you normally would.

   Once PIP's own prompt has appeared, you can suspend the program by pressing the <SUSPEND> key and enter Background CP/M. Because you are still on “computer 2”, this is referred to as **lower Background CP/M** and appears as b0} on your screen.

   Now you can use the **SWAP** command again! By typing:

   b0}SWAP

   you will return to the current state on COMPUTER 1 which is the SWEEP program.

   Now if you switch tasks from SWEEP (by entering Background CP/M and typing

   B0}SWAP

   you will go to the current state on “computer 2”, which is the PIP program. At this stage, you are switching between two active programs.

The three examples above demonstrate that you can use the **SWAP** command to switch between the current states of your two “computers”. These states are any combination of Foreground CP/M or your running program(s). Figure 4.1 explains the concept. Note that you can never swap to Background CP/M.

```
      COMPUTER 1                     computer 2
       (upper)                        (lower)

   A. Foreground CP/M              a. Foreground CP/M
       A0>SWAP (to a or b)             a0>SWAP (to A or B)
   B. PROGRAM 1                    b. PROGRAM 2
       (suspend to C)                  (suspend to c)
   C. Background CP/M              c. Background CP/M
```

```
        A0}SWAP (to a or b)            a0}SWAP (to A or B)
```

Figure 4.1: State Switching

4.2 The Flip Command

A command very closely associated with the **SWAP** command is the **FLIP** command. This command lets you temporarily switch between the display screens of the upper and lower case tasks. Any key will then restore the previous screen. This command will only function if a screen driver has been installed.

As an example, you are writing a report with your word processor and you wish to examine the spreadsheet that you are running as the alternate task to comment on the results. To do this:

1. Press the <SUSPEND> key to get to the background prompt.
2. Type

   ```
   FLIP
   ```

   to display the screen from your spreadsheet that is in suspended animation as the other task.
3. When you wish to return to your word processor, press any key and your current screen will be restored.

4.3 More About Task Switching

4.3.1 Time Taken to Swap

There are two main factors in the time taken to swap between applications, the most important being the type of drive the swap file is on. The second factor is the speed at which the terminal screen can be saved and redrawn. For memory mapped displays this is usually very fast, but for displays that communicate over a serial link, it can take several seconds.

Some representative times for swapping are shown below:

SB180	6MHz	ramdisk in main memory	<1 sec
Kaypro 10	4MHz	Advent Products ramdisk	<2 secs
Kaypro 10	4MHz	Hard Disk, 1K sectors, TurboROM	4 secs
Kaypro 2	5MHz	Floppy Disk, 1K sectors, TurboROM	20 secs
Kaypro 4	2.5 MHz	Floppy Disk, .4K sectors, Kaypro ROM	60 secs

4.3.2 Is BGii Multi-Tasking?

BackGrounder ii is not a true multi-tasking operating system because the non-visible task is not actually running. It is in a state of suspended animation, waiting to be revived the next time you switch to it. BGii's SPOOLER, however, is truly multi-tasking and will continue to run in either foreground state.[16]

4.3.3 Swap File Limits

The minimum size of swap file that will function with BGii is around 45K, but this will not allow task switching. For a swapping situation at least an 80K swap file is desirable and in general the size should be in the order of 100K. If the swap file is not at its maximum size, the “secondary” or lower-case computer will appear to have less memory than the upper-case machine. This may limit the type of program that can be run in the alternate state.

4.3.4 Unclosed Files

To maintain the integrity of the file system through all of this state switching, BackGrounder ii continually monitors which files have been opened for writing. To avoid excessive memory use, there is a limitation of 8 unclosed files open for writing when you swap or suspend. This limit should pose no problems for most programs. However, some programs like dBase II allow up to two database files, each with up to seven index files, to be open at once. This would cause BGii to lose data should you attempt swapping or background activity while these files were open for writing.

BGii is unable to protect the integrity of files if the program moves the file control block (fcb) after the file has been opened. Borland's Turbo Modula Compiler (at least the Beta version) and the NULU library utility employ this questionable practice. Be sure that files are closed before swapping from such programs.

4.3.5 Installing Resident Modules

If you wish to load a resident system extension, it is good practice to make sure that it conforms to the standards laid out in Appendix I. If the module is loaded in the lower-case state it must be removed before you can swap back. If a module is not smart enough to be removed then it must be loaded in the upper state and preserved with SECURE.

4.3.6 Using a Program in Both Tasks

Think twice before using the same program in two tasks. Editors such as WordStar or Perfect Writer will use the same names for their temporary or work files and end up overwriting work from the other task! WordStar can be safely used in two different user areas as the temporary files will be created in those areas. Perfect Writer, however, would require two PW.SWP files positioned on the disk so that each task would use one—difficult but not impossible to arrange.

4.3.7 Interrupt Driven Software

Any program that modifies the system interrupt vectors should not be swapped unless the interrupt handlers are located in secure memory.[17]

Chapter 5

Data Transfer

5.1 Moving Data Between Tasks

One of the primary reasons for running two programs or tasks at one time is to transfer data between them. BGii gives you four useful commands to do this: **`CUT`**, **`PASTE`**, **`JOT`**, and **`SCREEN`**. Note that all of these commands require that a functioning screen driver be installed. If you do not have a screen driver for your machine and wish to create one see Appendix B.

`CUT` and **`PASTE`** let you "clip" a defined rectangular section of text or figures from your screen and transfer that section as if it were keyboard input to any other program.

`JOT` lets you make backgroung notes, and also "paste" the clipped section into your notepad.[18]

`SCREEN` lets you write the exact (non-graphic) screen to a file or printer. With your text editor, you can then incorporate that file into any other file you are editing.

All are very useful time-saving tools, saving you the laborious task of recreating text, with its possibility of introducing new errors.

5.2 The Cut Command

The **`CUT`** command is used to "clip" and save a portion of a screen for later insertion into another file. It allows you to mark out any rectangular portion of a screen for export, while in any program. The following is an example of how **`CUT`** works.

Suppose you are running SWEEP and wish to use a section of text in another program. The steps to follow are:

1. Press <SUSPEND> to suspend SWEEP. The Background CP/M prompt—e.g. B0}—will appear.
2. Now type

   ```
   CUT
   ```

   You will almost immediately return to your previous screen.
3. At this stage, you may use the four cursor (arrow) keys[19] to mark out the section you want "clipped". First, move the cursor to the top lefthand corner of this region. When the cursor is positioned correctly, type **`X`**.
4. Next, move the cursor to the bottom righthand corner of the section you want clipped. (As you move the cursor at this stage, the section to be cut will blink rapidly or be highlighted.) Again, when the cursor is positioned correctly, type **`X`** (or <CR>).
5. The section you want clipped is now saved. You can now continue using SWEEP as normal.

The steps to **`CUT`** a section of your screen are seen to be very simple. Simply <SUSPEND> from your running program, type **`CUT`** and then move the cursor to the top left hand and bottom right hand corners of the desired section, typing **`X`** after each positioning. The **`CUT`** portion will be saved for future **`PASTE`**ing.

5.2.1 Some Things to Remember about CUT

1. When moving the cursor keys to mark out the section to CUT, you may not move off the screen (i.e., you are limited to the *current* screen).
2. Any portion of the screen may be **`CUT`**, even if it does not belong to the current program (i.e. including any portion of your screen not cleared before entering your program).

3. Only one **CUT** section is saved at a time. If you **CUT** another section before **PASTE**ing an earlier **CUT** section, that earlier section will be overwritten and lost.
4. If you type more than one **X** after delineating the first boundary of the section to be **CUT**, the **CUT** command will abort and you will return to your program.
5. A section that has been **CUT** will remain saved in the BGii swap file until it is overwrittent by another **CUT** operation. It will be available for pasting until you exit BGii.

5.3 The PASTE Command

PASTE is **CUT**'s twin brother. Once you have **CUT** a section of a screen, **PASTE** allows you to insert it into a text file you are editing, for example, in an alternate task or at some later time. The following example shows how **PASTE** works.

Suppose you are running SWEEP from A0> and WordStar from a0> and are currently logged into WordStar. Having **CUT** a section from SWEEP, you now wish to incorporate it into the file you are editing with WordStar. The steps would be:

1. First, position the cursor at the point in the text where you wish to insert the **CUT** section.
2. Now press <SUSPEND> to temporarily suspend WordStar. The a0} prompt will appear./li>
3. Type

   ```
   PASTE
   ```

   The **CUT** region will now be inserted into your text file at the current cursor.

5.3.1 Things to Remember About PASTE

1. **PASTE** will only **PASTE** the most recently **CUT** region or section.
2. You may only **PASTE** a **CUT** section into a program that will accpept the text as if it were being type from the keyboard. The command will not work at any other time.
3. A <CR> is automatically added at the end of each **CUT** line. For example, when used as input to a text editor, and the cut section consists of more than one line, the second and following lines will be **PASTE**d to the left margin.
4. Special cases of the **PASTE** command work within the **JOT** and **KEYS** commands.[20]

5.4 The Jot Command

JOT is a command that lets you make notes at any time—either while running a program, or at Foreground CP/M level. The text that you **JOT** is saved either in a file named JOTPAD, or in a filename of your own choice. The usage is:

```
JOT
JOT filename
```

to write to JOTPAD (in A0) or to write to *filename* (on the current drive and user area unless otherwise specified). After you type **JOT**, the screen is cleared, except for a message line, like the following for Kaypro computers,[21] to remind you of some basic editing functions:

```
JOT: <SUSPEND> - exit, ^X - del rest of line
ESC E = ins. line, ESC R = del. line, ^Paste
```

The editing functions available depend on the particular terminal you are using, but will generally let you insert a line, delete a line or the remainder of a line, and delete individual characters using the <BACKSPACE> or <DELETE> keys. Further editing of this file may be done at a later date with your regular text editor. When you have finished making notes, simply press <SUSPEND> (CTRL-^) to return to the CP/M prompt or your current program. Once you have started a JOTPAD, either of that name or your own filename, you may add to it on subsequent occasions, or choose to delete it. If at any time you wish to review the contents of your JOTPAD, you may view it with either the **TYPE** command (on the screen) or **LIST** command (to the printer). A JOTPAD is an ordinary text file that may be edited with your text editor, and incorporated into any other file of your choice.

5.4.1 PASTEing into a JOTPAD

You can also **PASTE** your **CUT** section into any JOTPAD. To do this:

1. TYPE

   ```
   JOT
   ```

   or

```
JOT filename
```

at either Foreground or Background CP/M. This will open up a JOTPAD, either under that name or your selected *filename*.

2. Place the cursor where you wish the **`CUT`** region to be positioned.
3. Now press **`CTRL-P`** to paste or insert the **`CUT`** region into the JOTPAD file. By moving thew cursor and retyping **`CTRL-P`**, you may insert it as many times as you like.

5.5 The Screen Commands

BGii's **`SCREEN`** command gives you a screen dump, either directly to the printer or to another file. It will reproduce all text (sometimes including the cursor) on the screen, but it will not reproduce graphics. Again, you must have the appropriate screen driver installed for you computer for this command to work.

To send the screen to the printer, type:

```
SCREEN
```

Make sure your printer is on-line, or else your machine will hang.

To send the contents of your screen to a file, type:

```
SCREEN filename
```

Unless you specify a drive and user number with the *filename*, this file will be on your current drive and user number. It may be edited or inserted into the file of your choice at a later time.

The **`SCREEN`** command may be used at Foreground CP/M (e.g. A0> or a0>) and also while running a program. If running a program, you must first <SUSPEND> to Background CP/M (e.g. A0} or a0}) and then type:

```
SCREEN
```

or

```
SCREEN filename
```

When you do this, the previous screen display of your program will reappear on your computer and be sent to the printer or a file. None of the BGii menu will be included in the screen dump.

It is, in fact, impossible to use **`SCREEN`** at Background CP/M level to print or save any Background CP/M messages or text. If you try to, you will always capture instead the previously displayed screen of the currently running program.

Chapter 6

Defining Keys

6.1 The Keys Command

One of the most useful functions of BackGrounder ii is its ability to attach definitions to any of the keys on your computer. Up to 255 characters of text and/or commands may be attached to a key, and displayed or implemented from that time on with a simple keypress. This defined key can be used to:

- execute a BGii command line
- automate repetitive editing tasks
- send login sequences to host computers and bulletins boards
- eliminate retyping of standard parts of correspondence such as names and address
- standardize script writing where characters and layout are repetitive
- insert time or date.

Defined keys are available to you at Foreground or Background CP/M level and while running any program requiring keyboard input. You can define any of the keyboard keys on your computer, and—if you are running the appropriate function key drive under BGii—the keypad and function keys. The key definitions will be retained while you are logged into your current program. If desired, they can be saved for future use eadch time you run a particular program.

There are a number of different steps in the key-defining process To acces them, simply type

```
KEYS
```

either at Foreground CP/M or at Background CP/M after **SUSPEND**ing your current program. The following menu will appear:

```
                              -- KEYS menu --
D - Define  P - Paste to key  R - Record  L - Load  S - Save
V - View    K - view Key      A - Attach  X - eXit
```

All of these commands, plus other aspects of defining keys, will be discussed in the following sections.

6.2 Global and User Definitions

There are two types of definitions available under BGii—global and user-task definitions.

Global definitions, as their name suggests, are available everywhere—in Foreground and Background CP/M and while running user tasks. These definitions are created at Foreground CP/M level and remain in effect until you turn off your computer, redefine the keys or remove BGii.

As shipped, BGii comes with four built-in global definitions. These are:

\a = <F5>	insert the most recent **CALC** result
\d = <FE>	insert the date, if you're running DateStamper
\r = <FC>	start and stop keystroke **RECORD**ing.
\t = <FD>	insert the time, if you're running DateStamper and a real-time clock.

All of these definitions may be used at any time while running BGii, so long as you do not overwrite them with another definition.[22]

User task definitions, on the other hand, are available only while running a program. You may create these definitions only when running a program or task, and they will vanish after you exit that task.

If you use the same key for a user-task definition and a global definition, the user-task definition takes precedence while the task is active. The global definition remains in hiding and takes effect everywhere else.

BackGrounder ii gives you the ability to define, save, and load your own set of global and user-task key definitions. Instructions on doing this are given in the following sections.

6.3 Keys Menu Selections

6.3.1 **D** - Defining Keys

To define a key under BGii, simply type

```
KEYS
```

and choose the D option. If you're wanting to change or add to the golobal definitions, you should be at Foreground CP/M (e.g. the A0> or a0> prompt). If you wish to set up definitions while running a program, **SUSPEND** from that program to Background CP/M (e.g. A0} or a0}) and type

```
KEYS
```

followed by

```
D
```

You may now define any keyboard key, and (if you have the correct function key driver installed for your computer) any function or keypad key with up to 255 characters.

The general procedure in defining a key is:

1. Press that key
2. Enter the definition
3. Terminate the definition with the <SUSPEND> key.

6.3.1.1 Defining Keyboard Keys

BGii lets you define up to about 35 keyboard keys, no matter what type of computer you own. Both lower and upper case versions of the key may be defined—i.e. you may attach one definition to a and another to A.

To define a keyboard key, you should first press the <ALT> key. BGii has initially set the <ALT> key to be the back-slash \.[23] For example, if you wish to attach the definition "hello" to the h key, you would (after selecting the D option):

1. Press the <ALT> key, (the \ sing is not initially seen on the screen)
2. Next press h
3. After the = sign appears, type:

   ```
   hello
   ```

4. To terminate the definition, press the <SUSPEND> key.

Once your keyboard key has been defined, you must press both the <ALT> key and that keyboard key to display or invoke the definition—e.g. type

```
\h
```

to produce *hello*. If you simply type the **h** key, **h** is what you'll get![24]

6.3.1.2 Defining Numeric Keypad and function Keys

Keypad and function keys may be defined only if you are running the correct function key driver for your computer with BGii. Otherwise, you are limited to defining just the keyboard keys.

To define a keypad or function key, you should simply press the key to be defined when prompted. Then enter your definition, terminating it with the <SUSPEND> key in the normal way.

BackGrounder ii's **SHIFT** command gives you the ability to use both the regular characters and functions of your keypad and function keys, as well as their defined functions. To define a function key and to then use it in its defined mode, you must first make sure that you have typed

```
SHIFT ON
```

at Foreground or Background CP/M.

Typing

```
SHIFT OFF
```

puts the keybad back in numeric mode.

No matter which kind of key you define, your user taks definitions will remain in memory as long as the current program is running. Your golbal definitions will remain unti BGii is exited. If you wish to save and later reload your definitions, you must run the **S**, **L** or **A** options of **KEYS**. For user-task defintions, do this before exiting the program.

6.3.1.3 Editing a Definition

There are two ways of editing your definition while you create it. Use the <DELETE> key to delete the previous character (*not* the <BACKSPACE> which will appear as ^H), or type CTRL-X to delete the entire defition.

If you decide you don't want to attach a definition to a particular key after all, or you wish to delete an earlier definition, simply type <SUSPEND> after the = sign.

6.3.2 P - Pasting to a Keyboard

The **P** option of **KEYS** gives you a convenient way to attach some already existing text to a key. This is an extension of BGii's **>CUT** and **PASTE** functions and requires a screen driver. For example, if you wish to extract your address from a letter and attach it to akey for future use, you would:

1. Use BGii's **CUT** command to clip and save the address.
2. While running the program—e.g. WordStar—that you wish to use your defined key with, **SUSPEND** to Background CP/M and type

   ```
   KEYS
   ```

3. Choose the **P** option.
4. 4. When prompted, enter the key to which you wish to attach your "pasted" definition—i.e. your address.
5. Once the definition has appeared on the screen, edit it if you wish, and then terminate the definition by pressing <SUSPEND>.

Note that a carriage-return (^M) is added to each row of any pasted definition.

If you choose the **P** option and you have not yet used the CUT command, BGii will simply beep and return you to the **KEYS** menu.

6.3.3 R - Keystroke Recording

Another convenient way to avoid repeating text or to incorporate a group of commands in a definition is to selct the Record (**R**) option in the **KEYS** menu. When you do this, you will hear a double beep, and the screen will instantly switch to the task being carried out before you typed **KEYS**.

Any text or commands you type from this point on—up to 255 characters—will be included in your definition. When you wish to terminate your definition, just press <SUSPEND> or the <RECORD> key (**\r** in the predefined global defitions). You will then be prompted for a key to define. After typing in the key, you will again return to your program. Your definition, however, has been recorded and will be displayed by choosing the **V** or **K** options of the **KEYS** menu, or implemented by pressing the defined key itself.

Note that the Record feature can only be used to capture keystrokes inside a program. It does not work at the CP/M level, and recording is terminated when a program exits to CP/M.

Recording of keystrokes can, alternatively, be started by pressing the key[25] at the point in a program where you want to begin a defintion.

6.3.4 V & K - View Definitions

The **KEYS** menu gives you two means of viewing your current definitions—the **V** and **K** options. The **V** option lets you view all definitions currently loaded. At Foreground CP/M, these are the global definitions, while at program level (i.e. when using KEYS within a program, at Background CP/M) they are the definitions currently loaded with your program.

The **K** option lets you view the definition of any specified key in the currently laoded definitions. When prompted, just type in the <ALT> key or the function key you wish to see displayed.

6.3.5 S - Saving Your Definitions

Once you've created a set of definitions to use either at Foreground or Backgroun CP/M or in a specific program, you may decide you'd like to save them for future use. You can do this very simply by suing the **S** option of the **KEYS** menu.

When you type **S** you are prompted first for a descriptive albel for your definitions. You may choose any word you like—e.g. *formatdefs*—of up to 12 characters in length.

You will then be asked for a filename for the labels. It is usually most convenient to use the name of the program the definitions are used for—e.g. WORDSTAR. A filetype of .BG is automatically given to the file.

To complete saving your definitions, simply press <CR>. Your definitions are now saved in a .BG file, which will be on the same drive and user number as you are currently logged into.

6.3.6 L - Loading Definitions

Once you have saved your definitions, you can manually load them at any point by selecting the **L** option of the **KEYS** menu.

When you select the **L** option, you will be prompted for the filename of your saved definitions. Once you've entered it and pressed <CR>, the definitions on that file will be loaded.

There are a couple of points to remember with the **L** option:

1. To load global defintions, you must be at Foreground CP/M and logged into the drive and user on which your key definition file is saved.
2. User definitions can only be loaded once you've entered a program. The only user definition files that can be loaded at this level are those on the same drive and user number as your program.

A second way to load a set of definitions is to type the filename after the **KEYS** command. In Foreground CP/M this will load new global definitions. For example, type:

```
KEYS GLOBAL
```

This will at once load the definitions in MYGLOBAL.BG, returning you to the CP/M prompt rather than the **KEYS** menu. Within a user program, this form of the command can be used to load user definitions from a drive and user number. For example,

```
KEYS B3:NEWDEFS
```

6.3.7 **A** - Attaching Definitions

If you have a set of definitions you always use with a partciular program, you may prefer to have them automatically load with your program, rather than use the **L** option every time. BGii lets you do this with the **A** option of the **KEYS** menu.

To use this option, you should first be logged into the drive and user number as your program and definitions file. After typing

```
KEYS
```

and selecting the **A** option, you will be prompted for the .COM file that you want your definitions attached to. You will then be given the name of the definitions file (if any) currently attached to your program, and three options: quitting with no changes, attaching a (different) definitions file, or detaching the current definitions file, should one already be attached.

Simply select your option, and the definitions will be attached for automatic loading with your program.

Note that the **A** option is only for attaching user definitions to a *program*.[26] To automatically load a set of global definitions you can include the

KEYS *definitions*

command in your start-up command line using **SETBG**.[27]

6.4 Reserved and Special Characters

6.4.1 Including Hex Characters in your Definition

You will notic when you choose the **D** option, that the following menu appears on your screen:

```
                        --Special keys --
<FE>   = DATE        <FD> = TIME         <FC>  = RECORD
<FB>n  = delay n*256 <FA> = FALSE on/off <F9>  = set high bit
<F8>   = to TERMINAL <F7> = to PRINTER   <F6>  = printer RESUME
<F5>   = calc RESULT
use: DEL = del. char., ^X = del. all,  <SUSPEND> = quit
     # = hex digits,   ^Q = quote next

Enter GLOBAL <ALT>+key  or  fn.key  or  r (redraw):
```

These hexadecimal characters denote special functions and are useful when including a group of commands in a definition. To define a hex character—those listed above, or any other hexadecimal character you want to include in a definition—you must first type the **#** character, and then follow it with two hex characters. For example, typing **#FE** will produce <FE>.

6.4.2 What the Special Characters Do

6.4.2.1 <FE> Insert DateStamper

<FE> inserts the current date in your definition, if you have DateStamper installed. It's useful for dating letters, etc. as it will automatically pick up the current date from your computer.[28] There are five different date formats available to you and you should use **SETBG** to make a selection.[29] The default format as shipped with BGii is "January 1, 1987".

As shipped, the BGii global definitions assign the Date command to **\d**.

6.4.2.2 <FD> Insert Times

<FD> inserts the current time into a definition. It is only available if you are running DateStamper with a real-time clock. The time will appear in the following format: hh:mm:ss (e.g. 11:53:25). It's useful for clocking up how long you're spending on different computer tasks, for recording when programming changes were made, etc.

BGii's global definitions assign the time command to **\t**.

6.4.2.3 <FC> Record Keystrokes

<FC> turns on the Record mechanism. If you attach this definition to a key—i.e. create a <RECORD> key—you can at any time decide to incorporate the text you are typing into a definition.

For example, if you are writing or editing a letter with your word processing software, and decide at some point you want to put a frequently used phrase,—say, "Thank you for your letter of"—into a key definition, you can do so immediately. The next time you are about to type "Thank you ... etc", just press your <RECORD> key (pre-defined as **\r**). A double beep will sound.

Now go ahead and type "Thank you ... etc" until you have reached the end of the part to go into your definition. At this point, press the <RECORD> key again. The **KEYS** Define menu will appear, asking you which key you wish to define. Choose a key—either an <ALT> prefixed key or a function key. The recorded definition will immediately be attached to this key, and you will be returned to the screen of your current program. You can test your definition by pressing the newly defined key.

Keystrokes are recorded only when a user program is running, not at Foreground CP/M. If you have been recording and exit to CP/M, the partially recorded characters will be lost. The <RECORD> key is initially set to **\r**.

6.4.2.4 <FB> Delay

<FB> is the Delay character. This literally inserts a time delay into your definition. It is useful—and often necessary—if you have a definition (macro) consisting of a group of commands, some of which require a minimum time to be implemented before going on to the next command. An example would be if you set up a definition to load a program and then implement some function in that program. Many programs would need some kind of delay built in after the loading command before going on to the next command.

You can set the length of the delay by putting a number between 1 and 9 after the delay character. **<FB>2** gives you a delay of 2 x 256 keyboard status checks whereas **<FB>9** will give you a delay of 9 x 256 keyboard status checks; using a letter will give a still longer delay. If many of your programs require delays then perhaps you should use the global option in **SETBG**.[30] The time for a keyboard status check varies according to the type of computer and program running. You'll just have to experiment to find the right length.

For example,

```
    D<FB>2I<FB>2R<FB>2^M
```

puts a suitable delay in many modem programs to get the directory of the RCPM.

6.4.2.5 <FA> False

<FA> is the False character. It is closely linked to the Delay character. The CP/M operating system has a function that will return "true" if it detects that a key has been pressed on the console device. This allows programs to proceed with some activity and periodically check if they need to fetch a character from the keyboard. Many programs such as WordStar will alter what they display if a character is typed before the current program activity is finished.

The <FALSE> command forces the operating system to tell the program that there is no character waiting. It will continue to return "false" until another <FALSE> is encountered or the string is exhausted.

Some of the commands such as **DIR** will abort if a key is typed before they have finished. For example, if you wish to put two **DIR** commands into one string definition, the first would immediately abort because the command processor would detect that there were more characters waiting. The <FALSE> command allows you to prevent this.

Example:

```
    <FA>DIR *.com^MDIR *.MSS^M
```

List the directory, first of .COM files, then of MSS files.

If the <FALSE> command is not included in the string, the **DIR** command would immediately abort.[31]

Do *not* use <FALSE> with programs such as Perfect Writer that only go to fetch a character if console status is true—they will wait forever!

6.4.2.6 <F9> Meta

This special character causes the *following* character to be sent to the program with the high order bit set. This is useful for sending the special reserved BGii characters (those being discussed in this section) to certain programs, as BGii does not recognize this version of the command. It enables thekeyboard to transmit all 256 characters to the computer—useful with foreign, graphics and special-character sets.

Example:

```
    <F9>v
```

Send <F6> character ("v" with the high bit set) to the running program.

If the definition were, instead, <F6>, BGii would interpret this as the printer RESUME character.

6.4.2.7 <F8> Terminal

`<F8>` is the Terminal character. When this character is encountered in a string, the characters following it are sent directly to the screen rather than to the program that is running. This redirection continues until another <TERMINAL> character is encountered, or until the string is completed. The <TERMINAL> character is particularly useful where some direct manipulation of the screen is required. It is also useful for sending escape sequences directly to the terminal and for configuring some special terminals.

For example, with an ADM-3A or Kaypro we can clear the screen and put the string "How are you?" at row 16, col. 16 and return to prompt with the definitions:

```
<F8>^Z^[=00How are you?^M
```

(Note: <ESC> displays as ^[and <CR> as ^M.)

6.4.2.8 <F7> Printer

`<F7>` is the Printer character. When <F7> is encountered in a string, the following characters in the string are sent directly to the printer rather than to the program that is running. This redirection continues until another <PRINTER> character is encountered or the string is completed. This is particularly useful for sending control (configuration) strings to the printer without having to run a program.

Example:

```
<F7>this string goes to the printer >F7>this does not
```

If the printer is not ready, BGii gives two long beeps and will abandon trying to print.

6.4.2.9 <F6> Printer Resume

`<F6>` is the Printer Resume character. This character is used in connection with the **`SPOOLER`**. It allows the printer to pause at page breaks, when printing files from the **`SPOOLER`** queue, so that another sheet of paper may be inserted. It also acts as a keyboard ON/OFF switch for the printer, temporarily stopping all unspooling when pressed and starting it when pressed again.[32]

6.4.2.10 <F5> Insert CALC Result

`<F5>` displays the last **`CALC`** result. This is useful if you wish to insert the result of BGii's Calculator into your text. BGii's global definitions assign the **`RESULT`** command to \a (for "answer").

6.5 More About Defining Keys

6.5.1 Inserting Special Characters

Any key on the keyboard, with one or two exceptions, can be inserted into a definition simply by pressing the key. Fo example, to put a carriage return into your string, just press the <RETURN> key (you will note that it apears on the screen as **`^M`**). Other control characters, which do not have a key dedicated to them (as do <ESC>, <TAB>, <BACKSPACE>, and <LINEFEED>) can be entered simply by pressing the CTRL key with another key in the normal manner—e.g. CTRL-A will produce **`^A`**.

The exceptions to the above are as follows:

1. A CTRL-Q must be pressed twice to enter a singlt CTRL-Q in your definition.
2. A <DEL> must be preceded by a CTRL-Q to enter it into your definition.
3. A CTRL-X must be preceded by a CTRL-Q to enter it into your definition.
4. The <SUSPEND> character must also be preceded by a CTRL-Q to prevent it from terminating the definition.
5. The hexadecimal flag character # must be preceded by a CTRL-Q to enter it into a definition.
6. To get the <ALT> (\) character into a definition, it must be pressed twice. Not that it is used to "quote" itself, rather than using a CTRL-Q.

6.5.2 Defining a Screen Redraw String

There is one special "key" that can be defined without the <ALT> prefix and its purpose is to redraw the screen after some background activity. This string is definied to be simply the **r** key and is automatically sent to the user program on exit from Background CP/M if there is no screen driver. This is useful if you have not installed a screen driver and the program has the ability to redraw its screen. For example, the Perfect Writer editor redraws the screen with the command **ESC CTRL-L**. WordStar will do the same with the **CTRL-\** command if the patch WSRDRxx.HEX is installed.[33] If you use this feature you will want to attach the definition to your program, using the **A KEYS** menu option. Note that the actual **r** key on the kayboard still produces an **r** when pressed; **r** is only used in this special way when actually defining the redraw string.

6.5.3 Long Definitions

Some programs are unable to process long definitions correctly.[34] If you have such a program you can experiment with a longer delay value between characters using **SETBG**.[35]

Chapter 7

BGii's Print Spooler

7.1 Print Spooling: An Overview

Among BGii's many features is a Print Spooler. This gives you the ability to redirect printer output to a file and later—as a background activity—to Unspool, or print this file. Any number of files may be queued for printing in this way on one or more printers. You may also set different forms for your printer(s), so that printing may be automatically grouped into batches according to the types of paper they use—e.g. letters in the print queue will not print until letterhead paper has been installed on the printer.

The BGii commands that let you do this are **SPOOL** and **FORMS**. Print spooling also requires the presence of the SPOOLER module in high memory, and comes with two programs:

Q	to manage the unspooling queue
REMOVE	to remove the SPOOLER module from memory.

7.1.1 How it Works

As you might suspect from the preceding paragraph, BGii's **SPOOL** command works slightly differently from the other built-in BGii commands. To outline its structure:

SPOOL is a built-in command that redirects output sent to the list device (usually a printer) into a file. Spooling allows you to delay printing and to capture as a file what would normally be printed output.

Q (a COM file) takes a queue of files and prints them in order on one or more printers. This â€œunspoolingâ€ operates as a background process, allowing printing to proceed while another program is running.

Both **SPOOL** and **Q** require that the SPOOLER module be loaded into memory, below BG. It uses about 2K of memory and can be removed when no longer needed. SPOOLER can only be loaded from the upper foreground (e.g. A0>) task, not the lower foreground task (a0>). To load this resident module simply type

```
    SPOOLER
```

at the upper foreground prompt.

Once SPOOLER is loaded, typing the command

```
    SPOOL
```

controls redirection of the list-output. A separate program, Q.COM, manages the print queue, which is kept in the SPOOL.QUE file. Q can be run as an upper or lower task.

REMOVE (a COM file) removes the SPOOLER from memory, reclaiming the space for large user applications. Type

```
    REMOVE
```

and answer **Y** to remove SPOOLER and **N** to retain BGii.

7.2 SPOOL

If you wish to divert printed output to a file, you should use BGii's **`SPOOL`** command just before you run your regular printing program—e.g. WordStar's Print option, or Perfect Printer. The **`SPOOL`** built-in command has the following valid forms:

`SPOOL [ON]`	starts spooling (diverting the current list device output to the default file SPOOL.$PL). If the file exists, BGii asks before deleting.
`SPOOL` ***`filename`***	starts spooling to the specified file.
`SPOOL OFF`	stops spooling and closes the active spool output file. BGii asks if you wish to rename the file. To avoid conflicts between spooling and unspooling operations, you should always rename a SPOOL.$PL file before placing it in the unspooling queue.

Spooling can use up a large amount of disk space rather quickly, particularly when the output is to a microspacing printer. Often there can be 3 or 4 bytes sent per character to be printed. Make sure that you have susfficient disk space available for the SPOOL.$PL file—a ramdisk or hard disk is the best position for this file.

7.3 Q

`Q` is the program to unspool a file, sending it to the printer. It can be run at any time, including while you are working on another program.[36] Printing will continue while you are working on that program, but will halt when you use the <SUSPEND> key to go to Background CP/M. It will resume when you return to your foreground task or Foreground CP/M.

Q prompts for input with a **`==>`**

For a review of Q's commands and parameters, type **`HELP`** at the Q prompt. You may also type the following commands at the **`==>`** prompt:

`Q`	Displays the currently queued files
`CANCEL n`	Cancels printing of the n'th file in the queue. If the n is omitted then the currently printing file is canceled.
`CHANGE m to n`	Changes the n'th file to be the m'th.

To add a file to the print queue, at the prompt enter:

1. Duu:filename.type (drive and user are optional)
2. Parameters, separated by spaces
3. <CR>

7.3.1 Q Parameters

There are many parameters you can add after the filename to determine the exact form in which the file will be printed out. The parameters are:

Parameter	(Defaults)	Remarks
`Snn`	(1)	Spaces between lines
`Hnn`	(60)	Height = number of lines per page
`Tnn`	(8)	Tabs = number of columns per tab
`Nnn`	(1)	Number of first page to print
`U`	(no)	print Unnumbered
`#nn`	(40)	column for pagenumber
`Cnn`	(1)	number of Copies
`R`	(no)	print Raw (no tabs, numbers, banner)
`W`	(no)	Wait at end of page for Printer RESUME key[37]
`Fx`	(none)	use Form type x, where x is any single character
`Z`	(no)	omit End-of-file (CTRL-Z) character check, print raw

`Dd`	(current)	print on Device (d=T,C,L,U)
		T=TTY:
		C=CRT:
		L=LPT:
		U=UL1:

`E`	(none)	Enter custom banner (the heading) for each page. On a new line type in desired text; use # for page number.
`Btext`	(none)	print Banner with filename (and its date/time) plus all text characters up to the <CR> at the top of each page.

Note that the Banner parameter should be the last thing on the line.

7.3.2 Print Queue Examples

Entry	**Prints**
`==>LETTER1 S2`	LETTER1 from current drive/user, double-spaced
`<==>B4:DEMO.BAS C2 #50/span>`	DEMO.BAS from B4:, 2 copies, number in col. 50
`==>MEMO.FIN R`	MEMO.FIN Raw
`==>PROG1.ASM H50 Bshort remark`	PROG1.ASM, 50 lines/page, with filename, date/time, and â€œSHORT REMARKâ€ at top of each page.
`==>LETTER2 S2 FL`	LETTER2, double-spaced, on form 'L'
`==>DRAFT1 S2 E special banner # on 6/1/86`	DRAFT1, double-spaced, with custom banner and pagination

7.3.3 Command Line Options

Q will also accept a command line, with the same filename specification and paramters as above. Used in this way Q processes the single command line and does not prompt. Example:

```
Q LETTER S2
```

enters LETTER1 into the queue, double-spaced.

7.3.4 Escape and Graphics Sequences

The Spooler recognizes certain printer ESCape sequences that could contain a CTRL-Z character that would ordinarily terminate printing. Dablo-type printers have a small number of ESCape commands of the form:

```
<ESC><command-byte><data-byte>
```

to set spacing pitch and tabulation. Other printers may use different command bytes, and some of these commands could have a CTRL-Z as the data byte of the sequence. Q.COM has a nul-terminated table of these command bytes at 106 hex, configured for the standard Diablo printer. The table can be patched, if needed, for a different printer's sequences. For Diablo-type printers, the **`Z`** parameter should not be necessary.

In cases when graphics character data is to be printed, or when a non-Diablo type printer is used and the table cannot be patched satisfactorily, the Z parameter can be used.

The Z parameter turns off checking for the CP/M end-of-file character CTRL-Z as well as any pagination, double-spacing, banner, etc., and sends every byte in the final sector of the file to the printer. Note that if the final sector contains extraneous characters following the logical end of file they may be printed; should this be a problem, the final sector can be filled out with non-printing characters (e.g. 0D hex = <CR>). Or, the file can be printed with a word processor.

7.4 Forms

The BGii **`FORMS`** command lets you set a current form for each printer. It works in conjunction with Q's **`F`** parameter.

When you type

```
FORMS
```

you will see the current forms setting (initially dashes, indicating none) for your printers. The devices listed are TTY:, CRT:, LPT: and UL1:. To set a form for any printer, type

`FORMS` ***`device form`***

where the device is denoted by T, C, L or U, and the form is denoted by any single character (e.g. **`L`** for Letterhead). For example,

```
FORMS L L
```

would set the device LPT: to letterhead paper thus allowing the Unspooler to print only files that specify that type of paper.

The setting can be cleared by typing

`FORMS` ***`device`***

If you have certain files that require printing on certain forms, e.g. letters to be printed on letterhead, you should specify **`FL`** (or the form character that you have chosen to represent â€œletterheadâ€) as one of your parameters for those files in the queue. BGii will not unspool such files until the **`FORMS`** command has been used to set that form for the printer.

7.5 BGPRINT

The BGPRINT utility, another resident system extension, translates specified characters before sending them on to the printer. It is an easy way to change printer spacing in the middle of a document, or to access a special character on a print whell. It is not included on the BGii distribution disk but is available on the BackGrounder Accessory Disk from Plu*Perfect Systems. It is used as the major example of how to write your own resident system extensions.[38]

The BackGrounder Accessory Disk may be obtained from Plu*Perfect Systems. It is only available in Kaypro SSDD format.

Chapter 8

Command Reference

8.1 Introduction

This chapter gives a complete alphabetic listing of all BGii commands in a standard format. As many of the commands are discussed in detail elsewhere in the manual, there will often be a cross-reference rather than a repeat of the examples and discussion. Each command is listed with its purpose, the syntax of the various uses, any special requirements and remarks explaining the meaning of the parameters.

8.2 Information Common to All Commands

The following terminology is used in the command explanations:

filename	Any CP/M acceptable unambiguous filename and filetype plus an optional drive letter and user number.
filespec	Any ambiguous CP/M filename plus an optional drive letter and user number.
`[..]`	Optional parameters.
`|`	Optional either/or items.

Each command also specifies whether a screen driver is required for its operation. If you wish to write such a screen driver please refer to Appendix B.

A few commands are not available at Background CP/M and this information is included with each command summary.

8.3 Alphabetic List of Commands

BG

Purpose: To control BGii options and remove BackGrounder ii from memory.

Usage: **`BG [OFF|V|Q|Y|N]`**

Foreground Only: No.

Screen Drv. Req: No.

Remarks: The parameters or lack of them have the following effect:

- When used with no arguments this command will display an alphabetic listing of the allowable commands, show active tasks and attached key definitions.
- When used with **`OFF`** BGii will remove itself from memory restoring your standard operating

system.

- The V,Q,Y,N flags have the following effect:
  V Prompt Verbosely with command listing
  Q Prompt Quietly without listing commands
  Y Yes, enable <SUSPEND> and <ALT> keys. This is the default.
  N No, disable <SUSPEND> and <ALT> keys. This may be necessary for some programs that require all keyboard keys as input.

See Also: Chapter 3

CALC

Purpose:	To access built-in hexadecimal and decimal calculator.
Usage:	`CALC`
Foreground Only:	No
Screen Drv. Req:	No
Remarks:	The calculator provides addition, subtraction, multiplication, division and negation of 11-digit decimal numbers. It also has three memory storage registers (A,B,C). Use the RA, RB or RC command to recall values from these registers. The hexadecimal mode provides all the above operations on 16 bit words, and also includes the modulus operation. A binary display mode is also available. Final results can be recalled and inserted into an applications program with the calc <RESULT> key definition (initially attached to \a).[39]

CLS

Purpose:	To clear the terminal screen.
Usage:	`CLS`
Foreground Only:	No
Screen Drv. Req:	No
Remarks:	This provides a terminal independent method for the operator to clear the screen.

CUT

Purpose:	To cut a rectangular region of text from the screen of an applications program for later insertion as input into another program.
Usage:	`CUT`
Foreground Only:	No
Screen Drv. Req:	Yes
Remarks:	After giving the command, move the cursor to the top left hand corner of the region and press the **X** or **x** key. Now move the cursor to the bottom right hand corner of the rectangular region and press **X** (or **<CR>**) again. The region is now stored in the cut buffer.
See Also:	Chapter 5

DATE

Purpose:	To display current date on screen.
Usage:	`DATE`
Foreground Only:	No
Screen Drv. Req:	No
Remarks:	This command reuires that Plu*Pefect Systems' DateStamper software be installed. The format of the date display may be changed with the **SETBG** utility.

See Also: The TIME command.

DIR

Purpose: To display files in a disk directory with filesize.

Usage: `DIR [filespec] [/FSA]`

Foreground Only: No

Screen Drv. Req: No

Remarks: The *filespec* above can be any valid ambiguous file reference with drive and user specification if required. The default is

```
DIR *.*
```

F Fast directory without file sizes
S Display only System files
A All, display both System and Directory files.

The meaning of the **F** parameter can be reversed with **SETBG**.

See Also: Chapter 3

ECHO

Purpose: To echo a character string typed on the command line to either the console or the list device.

Usage: `ECHO [$]character-string`

Foreground Only: No

Screen Drv. Req: No

Remarks: The optional **$** will direct the string to the list device (printer). The *character-string* may contain any characters except those that the command line editor will act upon. All alphabetic characters will be translated to upper case. Any control characters will be sent literally even though they display on the command line in up-arrow format.

The **ECHO** command is useful for controlling your terminal and printer functions, e.g. lock the 25th line and selecting compressed print. To make best use of it, you should be familiar with the control codes of your particular printer and terminal.

ERA

Purpose: To delete the specified file(s) from the disk directory.

Usage: `ERA filespec [V]`

Foreground Only: No

Screen Drv. Req: No

Remarks: The *filespec* above can be any valid ambiguous file reference and may contain optional drive and user specification. The file(s) that are deletedd are echoed on the console. By specifying the **V** parameter verification will be requested before any deletions occur. if ***.*** is used then confirmation is automatically requested.

FEED

Purpose: To send single form feed character to the list device (i.e. eject a sheet of paper on the printer).

Usage: `FEED`

Foreground Only: No

Screen Drv. Req: No

Remarks: The list device must respond to a CTRL-L for this command to work.

FIND

Purpose: To locate a set of files on a disk drive, regardless of user number.

Usage: `FIND` *`filespec`*

Foreground Only: No

Screen Drv. Req: No

Remarks: The *filespec* above can be any valid ambiguous file reference with drive specification if required. `FIND` is similar to the directory command except that *all* user areas on the specified disk are searched for a match to the ambiguous file specification.

FLIP

Purpose: To view the screen of the alternate task.

Usage: `FLIP`

Foreground Only: No

Screen Drv. Req: Yes

Remarks: This command simply swaps the screen from the current task to that of the alternate task. When any key is struck the original screen is restored. Useful for viewing results of another task for including in current task.

See Also: Chapter 4

FORMS

Purpose: To inform the spooler of a change in form type currently loaded in a printer.

Usage: `FORMS [`*`device`*`] [`*`form-character`*

Foreground Only: No

Screen Drv. Req: No

Remarks: The parameters have the following effect:

- If no arguments are given, display the current forms setting for each list device.
- If just a device is specified then the form setting is cleared for that device. The unspooler is reactivated. The devices are as follows:
  - T = TTY:
  - C = CTR:
  - L = LPT:
  - U = UL1:
- If both device and form code are given, then that printer is set to the specified form and the unspooler is reactivated. The forms code can be any single character (e.g. `L` for letterhead).

See Also: Chapter 7

GET

Purpose: To lad a file into memory at specified location.

Usage: `GET` *`address filename`*

Foreground Only: Yes

Screen Drv. Req: No

Remarks: Loads *filename* into memory at the (hex) *address* specified. Note that no translation takes place.

GO

Purpose: To re-execute a program which is already in memory.

Usage: `GO [`*`any command line that program requires`*`]`

Foreground Only:	Yes
Screen Drv. Req:	No
Remarks:	For this command to work the program in question must fully reinitialize itself. Some programs use static initialization and will not correctly re-execute. Key definitions, if attached to original, will still be attached.

HELP

Purpose:	To provide onscreen help.
Usage:	**`HELP [`*`helpfile`*`]`**
Foreground Only:	No
Screen Drv. Req:	No
Remarks:	Displays *helpfile*.HLP in the configured HELP directory, or in the specified drive and user number. If no *helpfile* is specified then the file BG.HLP is used. An onscreen menu gives options to control and print the display. The help function fully supports the ZCPR3 standard.
See Also:	Chapter 3, Appendix 3

JOT

Purpose:	To jot notes to a file.
Usage:	**`JOT [`*`filename`*`]`**
Foreground Only:	No
Screen Drv. Req:	Yes
Remarks:	The jotpad page is limited to one screen and the editing commands are limited to those video functions your terminal supports. If no *filename* is given this screen will be appended to the default JOTPAD file on A0:.[40] The command **`CTRL-P`** will insert the current contents of the CUT buffer into the jotpad.
See Also:	Chapter 5, and the CUT command.

JUMP

Purpose:	To execute a program residing anywhere in memory.
Usage:	**`JUMP`** ***`address`***
Foreground Only:	Yes
Screen Drv. Req:	No
Remarks:	Executes the program at hexadecimal *address*. This command is a general version of the **`GO`** command which is essentially equivalent to **`JUMP 100`**.

KEYS

Purpose:	To load key definitions and access the main KEYS menu.
Usage:	**`KEYS [`*`filename`*`]`**
Foreground Only:	No
Screen Drv. Req:	No
Remarks:	If *filename* is included then the file *filename*.BG will be loaded. If not then the main KEYS menu will be displayed.
See Also:	Chapter 6

LIST

Purpose:	To send a text file directly to the printer.

Usage: `List filename`

Foreground Only: No

Screen Drv. Req: No

Remarks: *Filename* (which may be specified with an optional drive and user number) will be sent to the current list device with any embedded TAB characters being expanded to align on multiples of 8 columns.

See Also: Chapter 3

NDR

Purpose: To load a named directory.

Usage: `NDR filename`

Foreground Only: No

Screen Drv. Req: No

Remarks: This command loads the file *filename*.NDR as the current named directory.

See Also: Creating Named Directories

NOTE

Purpose: To allow embedded comments in multiline commands or other scripts.

Usage: `NOTE [optional line of text]`

Foreground Only: No

Screen Drv. Req: No

Remarks: All text up to the command separator (;) or the end of the line is ignored.

OCP

Purpose: To load an Overlay Command Processor.

Usage: `OCP filename`

Foreground Only: No

Screen Drv. Req: No

Remarks: Loads the file *filename*.OCP as the current overlay command processor.

See Also: Appendix D for details on creating your own commands.

PASTE

Purpose: To insert a previously CUT region into an application.

Usage: `PASTE`

Foreground Only: It is *not"* available in foreground, only in background, i.e. while a program is running.

Screen Drv. Req: Yes

Remarks: Returns last `CUT` region to current program as if it were keyboard input. A <CR> is inserted at the end of each line.

There are two special cases of `PASTE` where the CUT region can be inserted into the JOTPAD or attached as a key definition.

See Also: Chapter 5, Chapter 6, and the CUT command.

PEEK

Purpose: To examine blocks of memory in hexadecimal and ascii.

Usage: `PEEK [address0] [address1]`

Foreground Only: No

Screen Drv. Req: No

Remarks: Displays the contents of memory in standard debugger format from *address0*to *address1*. If only *address0* is given then approximately a page of memory will be displayed from this location. If neither address is given then either display will start from 0000H or the point at which the previous **`PEEK`** command terminated.

POKE

Purpose: To modify memory locations.

Usage: **`POKE`** ***`address(byte0 ... ...) | $character-string`***

Foreground Only: No

Screen Drv. Req: No

Remarks: The *bytes* to be inserted shuld be in hexadecimal and separated by either a comma or a space. They cannot extend over more than one line. The *character-string* will be converted to upper case before it is inserted.

PRINTR

Purpose: To display and modify the printer portion of the IOBYTE.

Usage: **`PRINTR [`*`device`*`]`**

Foreground Only: No

Screen Drv. Req: No

Remarks: If the *device* is given then that device is made the default printer. This is equivalent to running the CP/M STAT.COM

```
STAT LST:=xxx:
```

where xxx is equivalent to TTY, CRT, LPT or UL1. With this command the *device* is simply the first letter of the normal CP/M device name: **`T`**, **`C`**, **`L`**, or **`U`**.

A user program prints on the printer currently set by the CP/M IOBYTE. You can use the **`PRINTR`** command to switch the printer currently attached to a program; this does not affect unspooling, which will use the printer specified for each queued file. It is possible, for example, to use one printer for background printing from the queue, and a second for a running program's print output.

Users with a single printer can ignore this command.

REN

Purpose: To rename a file.

Usage: **`REN`** ***`newname=oldname`***

Foreground Only: No

Screen Drv. Req: No

Remarks: This command will change the name of the file *oldname* to *newname*. If the file *newname* already exists then the user is asked whether the existing file should be deleted.

RESET

Purpose: To reset the disk operating system.

Usage: **`RESET`** (or **`CTRL-C`**)

Foreground Only: No

Screen Drv. Req: No

Remarks: This command closes any open files that have been modified and changes all disks to Read/Write status. Drive A and the default drive are logged in. If resetting the disk system at Background CP/M, you must make sure that the running program has completed writing any open files.

A command line disk reset under BGii produces a **Hot Boot**.

See Also:

SAVE

Purpose: To save a block of memory to a disk file.

Usage: `SAVE nn[H] filename [S]`

Foreground Only: No

Screen Drv. Req: No

Remarks: This command saves a block of memory starting at 0100H to a file named *filename*. The number of pages (100H or 256D bytes) saved is determined by *nn* which is interpreted as a decimal number unless it has **`H`** appended to signify a hexadecimal. If used with the **`S`** parameter, the command saves *nn* 128-byte Sectors. If the file already exists then the user is queried whether to erase the existing file.

SCREEN

Purpose: To dump an image of the screen to either the printer or a file.

Usage: `SCREEN [filename]`

Foreground Only: No

Screen Drv. Req: Yes

Remarks: An image of the screen (less any graphic characters) is sent to either the current list device, or appended to the specified *filename*. The command is disabled if spooling is active.

See Also: Chapter 5

SHIFT

Purpose: To change the state of any function keys between "shift" and "unshift".

Usage: `SHIFT [ON | OFF]`

Foreground Only: No

Screen Drv. Req: No

Remarks: If your function key driver has the supporting functions this will toggle your function/numeric keypad between its native mode and shifted mode. In shifted mode you can attach definitions to the function keys, and use such definitions.

See Also: Chapter 6 and Appendix C.

SPOOL

Purpose: To control the spooling of list output.

Usage: `SPOOL [On | OFF | filename]`

Foreground Only: No

Screen Drv. Req: No

Remarks: If no argument or **`ON`** is specified, the output to the current list device is spooled to the file SPOOL.$PL on user 0 of the swap file drive. If **`OFF`** is specified then spooling is discontinued and the list output is directed to the printer. The spool file may be optionally renamed at this point. If a *filename* is specified then this is the file that receives the output.

See Also: Chapter 7

SWAP

Purpose: To switch to the alternate task. If no alternate task is active, switch to alternate CP/M.

Usage: `SWAP`

Foreground Only: No

Screen Drv. Req: No

Remarks: None

See Also: Chapter 4

TIME

Purpose: To display the current time and control the system prompt.

Usage: `TIME [ALL | ON | [$] | OFF]`

Foreground Only: No

Screen Drv. Req: No

Remarks: This command requires that Plu*Perfect Systems' DateStamper be installed. The parameters have the following effect:

blank	displays current time.
ALL	displays current time and date.
ON	turns on time display in the CP/M prompt with optional Seconds display (using **`S`** subargument), e.g.: `10:24:39 A0>`
OFF	turns off time display in CP/M prompt.
TIME ON	turns off time display in CP/M prompt.

TYPE

Purpose: to display a text file on the console with scrolling, searching and unsqueezing.

Usage: `TYPE` *`filename`*

Foreground Only: No

Screen Drv. Req: No

Remarks: *Filename* will be displayed one page at a time. A menu of movement and utility commands is displayed at a window at the top of the screen. If the second character of the fily type is a Q then the fill will be automatically unsqueezed.

See Also: Chapter 3

USER

Purpose: To change user area.

Usage: `USER [`*`n`*`]`

Foreground Only: No

Screen Drv. Req: No

Remarks: This command will make user area *n* the new default. If no number is given then user 0 will be selected. Note that you can only move between user areas on the same drive.

See Also: Chapter 3

WHL

Purpose: To control the setting of the ZCPR3 wheel byte.

Usage: `WHL [`*`password`*`]`

Foreground Only: No

Screen Drv. Req: No

Remarks: this command controls the setting of the ZCPR3 wheel byte, which in turn controls user privileges. If no *password* is given, the wheel byte is set OFF and only non-private programs can be run. The *password*, if given correctly, turns on the wheel byte and allows all programs to be run. the *password* is set with `SETBG`.

See Also: Appendix H, Wheel Byte and Password for All.

WHLQ

Purpose: To report the current state of the wheel byte.

Usage: `WHLQ`

Foreground Only: No

Screen Drv. Req: No

Remarks: None

Chapter 9

Configuration Options

The SETBG program enables you to configure BGii to suit your particular needs. A wide variety of hardware, control, display and other options are available. Chapter 2 has already covered the basic settings needed to run BGii. This chapter describes the other `SETBG` options.

The final sections of this chapter deal with other, more advanced configuration options not connected with SETBG.

9.1 SETBG Menus

SETBG is organized in a series of nested menus. Changes are made to the file only when reuested. Therefore, you can readily experiment to learn what the different options do, and then write out the changes to a different file, in order to test them out.

The SETBG choices are obtained from the following menus:

```
Main menu

        Parameters menu

           Hardware menu
           Control Parameters menu
           Display Parameters menu
           Keys Parameters menu
           ZCPR3-related Parameters menu
           Startup Command menu
```

9.1.1 Main Menu

The main menu gives the following options:

```
0. exit to CP/M
1. change parameters
2. update file with new or changed parameters
3. print parameters
```

Option 1 takes you to the Parameters menu, while Option 2 installs any changes you may have made. When you choose this option, you will be given the choice of updating your Source File (usually LOADBG.COM) or creating a new file. If you do not select Option 2 after making configuration chagnes, your changes will be lost after exiting SETBG.

Option 3 lets you print out all current parameters. It is probably worthwhile doing this before embarking on a big configuration session.

9.2 Parameter Menus

The Parameters menu leads you to six sub-menus, which contain the configuration options. Most options are very straightforward and self-explanatory; a few demand more technical knowledge and are nly for experienced users.

If you are running a ZCPR3 system, you should note that the following parameters are taken from the external environment, and that the SETBG settings are therefore ignored:[41]

- cpu speed
- search path
- max. drive
- max. user
- HELP directory[42]

If you wish to examine or change a parameter, simply select the appropriate number and respond to the online prompts. Remember, if you make a change, but *don't* wish to save it, simply skip Option 2 when returning to the main menu.

9.3 Hardware Options

The Hardware menu looks as follows:

```
0. previous menu:
1. terminal:
2. screen driver:
3. fn. key driver:
4. cpu speed (mhz):
5. overlay at:
```

Options 1 through 4 are all dealt with in Chapter 2. These options should be configured before first installing BGii.

9.3.1 Overlay

This option is only for those few users whose machines will not let them run screen drivers, function key drivers and OCPs at the default address of 4000H, e.g., machines that have a memory-mapped video at this location. If you change this, you *must* reassemble all screen drivers, function key drivers, and OCPs to the new address.[43]

9.4 Control Parameters

The Control Parameters menu looks as follows:

```
0. previous menu
1. password
2. password for allocated3. swap drive
4. max. drive
5. max. user
6. HELP directory
7. JOT directory
8. path
```

Options 3 and 6 are both dealth with in Chapter 3.

9.4.1 Password

This refers to the password applicable to the Wheel Byte. It can be made to control access to commands and programs. The default password is password. Option 1 lets you change it to any word of up to 8 characters.

9.4.2 Password for All

This option makes *all* commands private. To run anything except WHL and WHLQ, the wheel byte must be turned on with the password.

9.4.3 Maximum Drive and User

Options 4 and 5 let you set the maximum drive and user that the Command Processor will log into. The defaults are P and 15. You should change the maximum drive to the actual maximum you are running, to protect yourself from inadvertantly attempting to log into a non-existent drive. There may be security reasons (e.g., with a BBS) for setting the number of drives to a lower than actual number.

9.4.4 JOT Directory

As shipped, BGii places the JOTPAD file on Drive A0.To change this, select Option 7.

9.4.5 Search Path

BGii looks for a program by searching along a path of up to five directories. Each directory is specified by a drive and a user number.

A $ designates the current or logged-in drive or user number at the time the command is typed. For the default path of:

```
$$:, $0:, A0:
```

BGii first searches in the current drive and user, next in the current drive, User 0, and finally in Drive A, User 0.

9.5 Display Parameters

The Display Parameters menu looks as follows:

```
0. previous menu
1. prompt character
2. prompt + high bit
3. command separator
4. printer test char.
5. DIR fence character
6. DIR with filesize
7. date format
```

9.5.1 Prompt Character

The default Foreground CP/M character is >, e.g. A0>. To change it to something else, choose Option 1. Remember that the symbol ")" has already been allocated to Background CP/M. (This prompt cannot be changed with SETBG.)

9.5.2 Prompt with High Bit

Terminals that do not display the high bit of a character can use this option to set th eprompt's high bit to permit extended command processors to recognize the prompt. It is of use to people using this feature of ZEX.

Command Separator

The command separator is uesed to separate multiple commands on the same line. The default separator is a semicolon (;). It can be changed using Option 3.

Printer Test Character

Before sending characters to the printer, BGii sends a test character (default <CR>) to select whether the printer is ready. Use Option 4 to change this character. It should be a non-printing character that will not modify the last-printed output.

DIR Fence Character

This is the character that separates the columns of a **`DIR`** display. The default is a "|". Some terminals may have a better character for this function.

DIR with Filesize

As shipped, BGii's **`DIR`** command gives filesizes in its display. These can be turned off with the **`/F`** parameter of **`DIR`** or by choosing Option 6. Doing this will slightly speed up the directory display. Note that if you change **`DIR`**'s default to *not* display filesizes, a **`/F`** will then *display* filesizes.

9.5.7 Date Format

If DateStamper is installed, BGii's <DATE> key returns the current date in one of the following five formats:

```
December 31, 1986 (default)
31 December, 1986
mo/da/yr
da/mo/yr
yr/mo/da
```

The separator in the last three formats is also configurable. Option 7 lets you change the default display.

9.6 Key Parameters

The Key Parameters menu is as follows:

```
0. previous menu
1. SUSPEND key
2. ALT key
3. key delay
4. remap (0-8) keys
```

<SUSPEND> Key

BGii's <SUSPEND> key is initially set as CTRL-\ (<1E>). To change it to something else, choose Option 1. It should be a key that is never used in any program that you run.

<ALT> Key

Use Option 2 to change the <ALT> key from the default \. It should be a seldom used key. When pressed twice, the <ALT> key will have its regular meaning.

9.6.3 Key Delay

This sets an automatic delay—in addition to any explicit delays—between characters in a definition. It may be required for some programs—e.g. the unmodified Perfect Writer—that cannot process long definitions correctly.

9.6.4 Remapping Keys

In addition to the ability to define keys using the **KEYS** command, BGii lets you remap up to 8 keys using Option 4 of the Key Parameters in SETBG. For example, the semicolon (;) can be remapped to a colon (:). Remapped keys can also be defined with the **KEYS** command; however, it is a good idea to mark such keys in some way (with stickers, etc.) to avoid confusion.

9.7 ZCPR3-Related Parameters

The ZCPR3-related Parameters menu looks as follows:

```
0. previous menu
1. command-run is ok
2. cmd-run is ROOT only
3. cmd-run addr. (+ bios)
4. use RCP space
5. use IOP space
```

ZCPR3 users should refer to their documentation for the first three options. Options 4 and 5 allow ZCPR3 buffer areas to be used by BGii to save memory. Functions of these ZCPR3 areas can be replaced by equivalent BGii functions that do not need the additional memory. You can reassemble an RCP to turn it into an OCP, thus saving 2K of memory.

9.8 Start-Up Command

The Start-Up Command menu lets you specify a command line to be executed after BGii has finished loading. This is useful for loading such things as a set of key definitions and/or your favorite program.

9.9 Creating Named Directories

BGii enables you to refer to any specific drive/user-number by *name* as well as by the "DU:" form. This feature is automatically implemented if BGii is run under a ZCPR3 system that has an active named-directory buffer.

If you have a different system you will need to use the **MAKENDR** program to create a NDR file that associates specific drive/user-number pairs with unique names.

the **MAKENDR**

commands are:

```
C -- Change directory (Add/Rename/Delete Entries)
I -- Initialize directory
D -- Display directory
L -- List directory on printer
R -- Read directory file
S -- display Status of MAKENDR environment
```

```
W -- Write directory file
X -- eXit to CP/M.
```

The program contains short command reminders, which you can recall by entering "?".

the first time you use the program, select "C:. Then enter "?" to get this reminder:

The Change commands are:

```
DU:dirname<CR> <-- create/rename dir. entry
DU:<CR>        <-- delete dir. entry
<CR>           <-- display directory
X<CR>          <-- exit from change mode to main menu
?<CR>          <-- display this help
```

To add a new name, enter the drive, user number, a colon, and the name (1-8 characters). For example, to make drive A, User 0 synonymous with "BASE", enter:

```
A0:BASE
```

The program will then ask if you wish this directory to be password-protected. If you enter a password, then that password will be required in order to log into that drive; for most users, this feature is seldom used, and a <CR> should be entered.

Continue naming directories that are useful to you, using the D main-menu command to display the current list. When you're ready to test your work. Write the named-directory out to a file (let's suppose you call it TEST.NDR). Then eXit the **MAKENDR** program.

To test your directory, enter:

```
NDR TEST
```

Now, when you type

```
BASE:
```

BGii should login A0: and display the prompt:

```
A0:BASE:>
```

9.10 Changing BGii Command Names

This section is for advanced users only.

After BGii has been loaded, the BGii command table can be located in the !!BG.SWP file with a disk utility, such as DU.COM. Search for the command table visually—the commands are in the order displayed by the **BG** command, with high bits set for some characters.

To change the name of a command, use the disk utility to replace the command letters with a ficticious name, or alias name. For example, you could change "PEEK " to "P ". In doing so, be sure to preserve any high bits that are set.

If you create an alias (in ZCPR3 systems) or an EX script to automate the patching process, and then add that command to the BGii auto-execute command line, the new command name will be installed each time you load BGii.

Appendix A

Error Messages

A.1 SETBG Messages

Message:	Open Error on *filename.typ*
Cause:	A BIOS open error has occurred.
Action:	File not in specified drive/user directory; misspelled filename.

Message:	Create Error on *filename.typ*.
Cause:	A BDOS create error has occurred.
Action:	Directory is full. Copy, then Delete files; or change disks.

Message: Rename Error on *filename.typ*.
Cause: A BDOS rename error has occurred.
Action: Old filename can't be found. See Open Error.

Message: Close Error on *filename.typ*.
Cause: A BDOS close error has occurred.
Action: Directory is full. Copy, then Delete files; or change disks.

Message: Read Error on *filename.typ*.
Cause: A BDOS read error has occurred.
Action: Part of file is missing; disk has a bad spot.

Message: Write Error on *filename.typ*.
Cause: A BDOS write error has occurred.
Action: Disk not logged in with control-C; disk has a bad spot.

A.2 PUTBG Messages

Message: The Backgrounder requires CP/M 2.2!
Cause: Incompatible operating system.
Action: Run BGii with CP/M 2.2 or ZRDOS.

Message: Directory is full! Erase mm entry(s).
Cause: Obvious
Action: Obvious

Message: Can't create the !!BG.SWP file
Cause: Directory may be full.
Action: Try erasing some files.

Message: Insufficient space, need mmK more
Cause: Obvious
Action: Copy, then Erase files; or use another disk.

Message: Disk is full! Erase mmK (nn groups).
Cause: Above.
Action: Copy, then Erase enough files to accommodate !!BG.SWP file.

Message: Directory read error, sector mm.
Cause: BIOS read error.
Action: Disk is probably not usable.

Message: Directory write error, sector mm.
Cause: BIOS write error.
Action: Disk is probably not usable.

Message: Can't read group mm
Cause: IOS read error.
Action: Disk is probably not usable.

Message: Can't update directory group
Cause: BIOS error.
Action: Disk may not be usable.

Message: Group mmH(nn) out of range!
Cause: Above.
Action: Fatal system error. Retry with fresh copy of PUTBG.

Message: !!!TIME&.DAT update error # nn
Cause: Above.
Action: DateStamper file may be bad. Try new disk.

A.3 LOADG Messages

Message: Can't load!
Cause: Many.
Action: Cold-boot system and retry commnd.

Message: Corrupt LOADBG file. Coldboot required.
Cause: LOADBG.COM damaged.
Action: Recopy LOADBG.COM from original disk.

Message: Dos overlays missing!
Cause: LOADBG.COM is defective.
Action: Recopy LOADG.COM from original disk.

Message: Non-standard dos at relative: nnnn
Cause: BDOS section of CP/M is packed or non-standard version.
Action: Boot a standard CP/M system and retry.

A.4 BGii Messages

Message: AFN Error
Cause: '*' or '?' character in filespec.
Action: Do't use wildcards with `ERA` or `REN` commands.

Message: Bad size:
Cause: OCP or NDR file is wrong size.
Action: NDR: copy original file or make new one with the `MAKENDR` program.
OCP: correct source file and reassemble.

Message: BG ERR:r
Cause: BIOS error reading BG swap file.
Action: Fatal. Check disk for bad sectors; use fresh disk.

Message: BG ERR: w
Cause: BIOS error writing BG swap file.
Action: Fatal. Check disk for bad sectors; use fresh disk.

Message: Can't open!

Cause: Can't find the file in specified directory.
Action: Check drive, user, and spelling of filename.

Message: Can't find it!<CR>
Cause: Can't find search pattern specified in TYPE's "G" command.
Action: Recheck pattern. Use "B" command to search from beginning.

Message: EOF on Help File
Cause: Premature end of help file.
Action: Edit help file, following HLP format.

Message: ERROR
Cause: Error entering/calculating in CALC.
Action: Re-enter data.

Message: File not found.
Cause: Obvious.
Action: Obvious.

Message: Full!
Cause: Key definition space is full.
Action: Delete some definitions.

Message: FULL
Cause: Memory full; file too large to load.
Action: Retry after removing SPOOLER and BGii.

Message: FULL
Cause: Directory full; can't create or close file.
Action: Change disk or copy and delete some files.

Message: Key file err: r,w,o,c
Cause: Read, Write, Open or Close error using *.BG definitions file.
Action: Save definitions in a new file and retry.

Message: No File
Cause: Can't find help file.
Action: Copy BGii help files to configured help directory; check directory selected with SETBG. In ZCPR3 system, check HELP directory for correct help file.

Message: --No Label--
Cause: Definitions not yet labeled/saved (not an error).
Action: To label and save, use **`KEYS "S"`** command.

Message: Lo err
Cause: BDOS file error in low-memory overlay
Action: Fatal. Reload BGii with fresh copy.

Message: Node Level Limit
Cause: Help files nested too deeply.
Action: Combine several nested help files.

Message: NONE
Cause: No definition for key (not an error).
Action: None.

Message: Not an OCP
Cause: File not in Overlay Command Processor format.
Action: Name mispelled, or, edit and reassemble OCP file.

Message: Not an NDR
Cause: File not in Named Directory format.
Action: Name mispelled, or, use **`MAKENDR`** to make correct file.

Message: Not loaded!
Cause: SPOOLER.COM not in memory
Action: Load SPOOLER.

Message: Nul file!
Cause: File has no bytes. Possibly copied to full disk.
Action: Copy file to new disk, or, delete files to gain space, then recopy.

Message: Printer not ready!
Cause: Printer busy or off-line
Action: Place on-line. Reissue command.

Message: Real error!
Cause: BDOS error reading file.
Action: Copy file to new disk and retry.

A.5 Possible Difficulties

1. BGii and especially the Spooler uses the BIOS List Status function to determine when the printer is ready to receive a character. It must be implemented correctly.
2. PUTBG moves existing files toward the inside of the disk, if necessary, to make room for the BGii swap file. It is essential that the DOS rebuild the drive's allocation map when called with a drive reset function (even if the drive is not removable). Should the DOS not do this, files in the first 100K of the drive should first be copied to another drive, then erased, and then PUTBG run. The files can then be copied back to the original drive, where they will be located inside of the swap file.
3. Programs that directly modify disk directories could give problems if run while BGii is installed.
4. Several BIOSes do not handle host buffer flushing and disk media determination correctly. If the BIOS wrongly places these functions in the warmboot routine they will not be executed while BGii is loaded and it may not be possible to change type of disk while BGii is installed. In addition, incorrect de-blocking routines can make BGii run extremely slowly. If the swapping speeds you achieve are dramatically different from the sample times listed on page 32 you should investigate this.
5. BGii may not load if the BDOS has been patched. The deblocking, DateStamper, and backspace/rubout patches are ok. The CTRL-S patch is not.
6. A very few programs require a maximum amount of memory. either or both Spooler and BGii should be removed from memory.

Appendix B

Coding a BGii Screen Driver

BGii provides a uniform interface for the many varieties of CP/M terminals and video consoles. Since it's impossible for us to supply drivers for more than a few, we are making the source code available in both printed and machine readable form.[44]

If you develop a driver for a new terminal, we hope you will make the code available to other BGii users. Many bulletin boards around the country have expressed willingness to act as clearinghouses for this information.

B.1 Types of Console Devices

to implement all of BackGrounder ii's screen-related features, BGii needs a screen driver that can:

- determine the cursor location on the screen,
- read the entire screen, including attributes.

There are several types of output console devices:

- video-mapped memory screens
  - video ram accessible in banked memory
  - video ram accessed through a video controller
- terminals
  - transmit screen region
  - transmit character at the cursor
  - multiple screen page memory

The prototype code illustrates several variations:

Kaypro '83:	video ram in banked memory, no transmit-cursor command
Kaypro '84:	video accessed through controller, cursor indirectly determined from commands and registers
heath/Zenith 19:	transmit screen region commands with attribute codes embedded in stream

A very smart terminal may have several switchable pages of screen memory. If the terminal can be caused to copy its current screen to a designated page and to switch pages in response to escape sequences, most of the screen save/restore work can be done by the terminal itself as a parallel process. This degree of intelligence could also be included in a terminal emulation program, for example, when a computer is used as a terminal for another host that is running BGii.

BGii calls the screen driver entry points with a flags byte in register A. If bit 1 is set, the upper task is active; otherwise, the lower task.

The code has been kept in two sections: a "generic" part that should apply to almost any terminal, and a "specific" part that must be customized for the features of the particular terminal. The driver runs at 4000H (configurable) and interfaces to BGii through a jump table. The code cannot exceed 2K and the sum of the data areas and code cannot exceed 4K.

B.2 Screen Driver Functions

The major screen driver functions are summarized here. For details, study the prototype code.

B.2.1 Screen-Save

1. Find the cursor position:
   - if available, use transmit-cursor-position
   - else, read entire screen, write signature character, read screen again until signature found
2. Read entire screen:
   - if necessary, home cursor
   - transmit character(s) from screen
   - transmit attribute(s> from screen
   - if needed, repeat for 25th(status) line

B.2.2 Screen-Restore

1. If necessary, home cursor.
2. Clear attributes.
3. Write entire saved screen, including attributes.
4. Set cursor to saved position.

B.2.3 Cut-Region

1. Edit keystrokes:
   - allow cursor movement (but prevent going off screen or scrolling), until X or x
   - set first mark
   - allow only cursor right, cursor down, but prevent going off screen, or scrolling
   - highlight or blink the marked region as it grows until X, x, or CR

    - set second mark
2. Write cut region to BG:
    - calculate byte count, number of rows, and number of columns
    - send each char in row
    - add CR at end of row

B.2.4 Jot

The **JOT** command uses whatever control-characters and ESCape sequences the terminal/video driver has. If space permits, additional editing commands can be installed.[45]

1. Clear screen.
2. Display 2-line summary of editing command.
3. Edit entered keystrokes:
    - convert CR to CR,LF
    - convert DEL to BS,SPACE,BS
    - on CTRL-P, call paste-region
    - on CTRL-D, abort
    - send anything else to screen

B.2.5 Paste-Region

1. get bytecount, rows, cols from BGii's last cut-region
2. get 1st row, put on screen at cursor
3. get next row, put vertically below cursor, ... etc.
4. prevent going off screen, or scrolling

Appendix C

Coding a Function Key Driver

C.1 Key Driver Functions

The driver consists of:

1. An identification sector (at 3E80H)
2. Two sectors of function key codes and labels (at 3F00H).
3. Up to 4 sectors of code (at 4000H) and initialized data to:
    - initialize and de-initialize the terminal's function keys
    - "shift" the function keys to sent single 8-bit codes
    - "unshift" the keys to send their normal codes/strings
    - save the current mode (shift or unshift)
    - restore the saved mode

As regards function keys, there are two types of terminals, those that:

1. Have a ram table of key values in the bios (e.g. Kaypro).
2. Change modes in response to ESCape sequences (e.g. Heath-19).

Source code for drivers of each type is available both in printed and machine readable form.[46] In most cases only limited modifications will be needed to the code. In addition, the table of key values and identifying labels will need to be adjusted to fit the particular terminal.

The code should be assembled with an absolute address of 3E80H for the identification sector (see examples). Load it into memory with DDT and move 3E80H ... to 100H. Then:

```
SAVE N filename.FNK
```

C.2 Multi-Character Function Keys

If the terminal cannot be configured to send a single 8-bit character ("native mode') when a function key is pressed, BGii cannot distinguish the function keypress from a series of single keypresses that send the same codes from the main keyboard. Some terminals do not provide a native mode and always send ESC plus one or more characters. In this case it may be possible to modify the BIOS console input routine to convert these keys to a native mode, as shown in the following pseudo-code example:

```
get characterif char is ESC
    wait for 1 char time at current baud rate
        if next char is ready,
            get it, set bit 7
            return that value
        else
            return ESC,
            and return the second char on next call
    else
        return char
```

Because the timeing routine must be specific to the system and be cognizant of the interrup structure, it is not practical to include this type of routine in BGii itself.

Appendix D

Coding an Overlay Command Processor

To create an Overlay Command Processor (OCP) module:

1. Code the module following the notes and prototype below.
2. Assemble in an absolute hex file at origin 4000h.
3. Load hex file under debugger and move 4000h-4FFFh to 100h.
4. Save (up to) 16 pages as: *filename*.OCP

If an OCP command runs in foreground, all memory from 005CD to (0006H)-1 is available. In background, the OCP must use only the 005CH-00FFH and 4000H-4FFFH areas.

When a command is called, A = 0 if in foreground, A > 0 in background. Any command parameters are parsed into the default file control block at 005Ch; the full command tail is in the default buffer at 0080H.

Register pair HL contains the address of the ZCPR3 environment descriptor, or 0 if none.

BGii scans the currently-loaded OCP command table second, after scanning the resident BG commands (and before checking a resident command package in a ZCPR3 system). Thus, an OCP command will effectively replace a COM file by the same name, unless it is prefixed by a drive/user designation.

OCP data can be saved between commands by setting the 'wrtflg' non-zero, in which case BG will write the 4K OCP buffer back to the swap file when a command terminates. Note that if this feature is used, *all* OCP commands must fully initialize their data areas.

The overlay buffer at 4000H can be changed with SETBG. If this is done, all OCP packages, the screen driver and function key driver must be assembled to that address. This should only be necessary on a system that requires the 4000h-4fffh area for bank switching to access a video driver.

The hi bit (bit 7) of the characters in each BGii, OCP, or RCP command are reserved. If the bit is set it has the following effect:

character	effect
1	command requires wheel byte set
2	foreground-only command
3	cbackground-only command
4	reserved
5	reserved
6	reserved

Prototype OCP Assembly-language Module

```
            N         equ       4          ;max # chars in a command
                      org       4000h
4000                  db        'OCP'      ;required identification
                      db        'x'        ;optional module id
4004        wrtflg:   db        0          ;save-module-on-exit flag
;
; Each command must be exactly N chars long. Pad with spaces.
;
4005        cmdsiz:   db        N          ;max # chars in a command
4006        cmdtbl:   db        'CMD1'     ;first command, uppercase
```

```
                      dw         cmd1       ;addr of first command
                      db         'C2  '     ;second command
                      ...
            cmdend    db         0          ;NUL - end of table
;
; Note: the command table must not
; extend beyond 407fh.
;
            cmd1:    sta         bgflg      ;save background flag  (A)
;
; initialize data for this command
                     xra         a
                     sta         data1
                     ...                    ; code for the command
;
                     lda         5dh        ; check for any parameters
                     cpi         ' '
                     cnz         parse      ; if any, process then
                     ...
;
; set wrtflg if data needs to be saved
                    mvi          a,1
                    sta          wrtflg
;
; return to command processor, or warmboot if needed
                    ret                     ; or rst 0
            parse:  ...          0          ; pick up parameters...
                    ret
            data1:  db           0          ; sample data
            bgflg:  db           0          ; NZ if called in background
;
            cmd2:   sta          bgflg
                    ...                     ; second command:
;
; Address of last byte must be less than 5000h.
;
                    end
```

Appendix E

Creating Help Files

the BGii **HELP** command displays Help files, both at Foreground CP/M and while running a program.

It's easy to create Help files for use with your programs by using your text editor in ascii mode (for WordStar users non-document mode). And BGii makes it a snap to polish up the layout of your Help file. From within your editor you can save the current version of the Help file you are creating as, e.g. TEST.HLP. Then, by suspending your editor and typing HELP TEST, you can check the spacing of messages, eetc. and return to the editor to make corrections.

E.1 Components of a Help File

Help files are made up of "information sections" plus an optional "index". The beginning of an information section is designated by a colon in column 0. the BGii **HELP** command displays an information section one screen at a time, until it reaches the end of that section.

The Help file may begin with an index—a screen that shows which key to press to display each information section. There are two types of indexed Help files—"automatically-indexed" and "user-indexed". An automatically-indexed file assigns the letters "A","B","C" ... to the successive information sections. In a user-indexed file, the index characters are specified in the file.

In addition to the colon in column 0, Help files treat the following characters in a special manner:

CTRL-L	causes the screen to be cleared when a key is pressed to display the next screen
CTRL-A	turns on highlighting (reverse video)
CTRL-B	turns off highlighting

You may wish to inspect the BGii Help files with a text editor. BG.HLP provides an extensive example of a user-indexed help file. BGINFO.HLP, which is invoked by BG.HLP, is automatically indexed.

E.2 Non-Indexed Files

The simplest form of a Help file is just a series of information sections. Each section begins with a line having a colon in column 0. For example:

:
lines of text in the first information section
...
:
:
lines of text in the second section
...

E.3 Automatically-Indexed Files

Each information section is preceded by a one-line label for that section. The label is automatically displayed as an entry in the index, with the first section keyed to "A", the second to "B", etc.

label for A'th section
:
lines of text in the A'th information section
...
label for B'th section
:
lines of text in the B'th information section
...

E.4 User-Indexed Files

The initial lines are displayed exactly as written, to provide a user-defined menu showing an index character to be typed for each information section. Each information section is preceded by a line with a colon in column 0 followed by the index character for that section.

This is an example of a user-indexed menu.
It might have lines such as—
A - How to use HELP files
...
1 - BGii and DateStamper
...
:
lines of text for
the information section about HELP files
...
:1
lines of text for
the information section about BGii and DateStamper
...

E.5 Nesting Help Files

An information section can invoke another Help file. That file is also in Help file format, with any type of indexing. IIt may invoke yet another Help file ... up to a maximum of 10 nested files. the name of the file to be invoked is placed on the "colon line", following the colon and index letter.

For example, the BG.LP Help file contains the index entry:

```
    F - Features summary
```

In the file following the index lines, this "colon line" indicates that when "F" is selected BGii should chain to the BGINFO.HLP file:

:F:BGINFO.HLP

Appendix F

Batch File Processing

BGii's multiple command line facility is very convenient for a short group of commands. For more extensive processing, BGii supports Submit files created on drive A0:. The SUBMIT.COM program or its substitutes creates a temporary file $$$.SUB which BGii then executes as a sequence of commands. In BGii the commands in the $$$.SUB script can change the logged-in drive and user number without terminating the batch process. Also, you can use <SUSPEND> to interrupt scripts in mid-process to execute background commands.

BGii uses the semicolon ";" (configurable with SETBG) to separate commands on the same line. Comments can be included in a submit script by using the `NOTE` command: all characters following `NOTE` up to the command separator ";" or the end of the line are considered a comment that will be echoed to the console.

BGii similarly supports temporarily-resident batch processors, such as EX and ZEX, provided that they coform to CP/M standards. However, scripts run by a memory-resident batch processor cannot be suspended successfully, and should be allowed to run to completion. Note that the standard versions of EX and ZEX incorrectly calculate addresses if a resident module like BGii is present. We have provided modified versions that correct this error and will run under BGii.

BGii also supports programs such as SYNONYM that move a command line directly into the BGii command processor's command line buffer and call the autoexecute entrypoint.

Appendix G

Redraw the Screen in WordStar

If you are unable to obtain or construct a screendriver for your computer you should use the techniques described in Chapter 6 to redraw the screen after background activity. Not all programs, however, have a convenient method of redrawing the screen.

As distributed by MicroPro, WordStar does not have any built-in mechanism for redisplaying the screen after it has been disturbed by some background activity. To overcome this problem, we have included on the BGii disk a patch which adds a new command to WordStar, enabling the exact screen to be redrawn almost instantaneosly. The command we have used is a CTRL-\. This command is available both at the No-File menu and while editing.

Before it cvan be used, this patch must be installed in your working copy of WordStar using DDT. Details of the patch are given at the end of this section on WordStar. Once you have installed it, you can use the new redraw screen command—CTRL-\— in your definition files.

G.1 Installation

Installing this patch into WordStar gives you a new command—CTRL-\—which redisplays the screen. You will need to use the program DDT.COM to implement the patch. It is not a complex process but the steps must be followed carefully and nautrally you should make sure that you are fully backed up before you begin.

There are two separate patches contained on the BGii disk:

```
WSRDR30.HEX, WSRDR33.HEX
```

which apply respectively to WordStar versions 3.0 and 3.3. Make sure that you use the correct patch for your version of WordStar—look at the sign-on message carefully before beginning. These patches should *not* be used if you have already added other custom patches to WordStar, as there will most likely be a conflict. These patches are installed in the area that WodStar uses for direct memory mapped video drivers. Very few machines use this technique, and if this situation describes your computer you have the potential to write a full-function screen driver.

It is a good idea to do this patch on a freshly prepared disk. You should copy the files WS.COM, WSMSGS.OVR, WSOVLY1.OVR, WS.COU, WSRDR3x.HEX and DDT.COM to a freshly formatted disk. Log into this disk.

To install, type:

```
DDT WS.COM
```

When DDT responds with its "-" prompt, it is ready to continue. Now type:

```
IWSRDR3x.HEX
```

where *x* corresponds to the version of WordStar that you have (i.e. 0 or 3). Again, when DDT gives the "-" prompt, type:

```
R
CTRL-C
```

You will be returned to the CP/M prompt where you must save the newly patched version of WordStar. Initially, we suggest that you use a different name until you have fully tested that your patch has been successful. Type:

```
SAVE nn WSRD.COM
```

where *nn* is either 63 for version 3.0 or 71 for version 3.3.

You can now test this version by typing

```
WSRD
```

(Note that some of the WordStar commands that cause other programs to run will not return correctly, because they expect WordStar to be called WS.COM, not WSRD.COM.) If you type a CTRL-\ while in WordStar you should notice a slight rippling on the screen as it is redrawn.

The next test is to load BGii and then rerun WordStar (still called WSRD.COM). After WordStar is loaded you should press the <SUSPEND> key then type:

```
KEYS
```

followed by:

```
D
```

to get to the key definition menu. Next type

```
r<CR-\>
```

which will set up the redraw string. Now exit the **KEYS** menu and return to WordStar. The screen should be redrawn and appear just as it was before you suspended. Try some other Background activities and you will notice on each return to WordStar that the screen will automatically be redrawn.

If all appears to be working as it should, you should rename WSRD.COM to be WS.COM:

```
REN WS.COM=WSRD.COM
```

You can then use the **KEYS** command to permanently attach the definition to this new version of WordStar. You can of course put this simple return string into any set of WordStar definitions by simply defining the **r** key to be <CTRL-\>.

Appendix H

Using BGii with ZCPR3

BGii does not, itself provide the external environment descriptor and buffers used in a ZCPR3 system. However, if they are present BGii attempts to provide complete support for all aspects of a ZCPR3 environment, including full context-switching between upper and lower tasks. This appendix covers technical features.

H.1 Environment Parameters

If BGii detects an external ZCPR3 environment descriptor, those parameters override any values configured with SETBG (e.g. for help directory or max drive).

BGii distinguishes between features of the ZCPR3 environemnt that are global, and therefore unchanging when tasks are swapped, and those that are task-specific.

H.1.1 Globals

The Following addresses are determined when BGii is loaded and cannot be dynamically changed while BGii is running.

- external path
- RCP
- FCP
- NDR
- external command line
- environment descriptor
- shell stack address and size

- message buffer
- external fcb
- wheel byte

H.1.2 Lower-Task Initialization

On the first swap to the lower task, BGii initializes the lower task environment as follows:

- shell stack to 0
- message buffer to 0, except that the error command line and message byte 0 are presevered
- command line to 0
- filenames to blank, except that the shell variable filename is preserved

Thereafter these variables become task specific.

H.1.3 Task-Specific Variables

These variables/data areas are swapped when a task is switched, and restored when that task is resumed:

- shell stack (must be 128 bytes total)
- message buffer bytes 01-0FH and 30H=4FH (i.e. the error command line is a global, the same in both tasks)
- external fcb
- external command line
- 4 filenames (at environment-descriptor address + 52 H)

BGii dynamically determines the values of the following variables from the environment descriptor—therefore, these values (but not their locations) can be changed while BGii is running:

- max. disk
- max. user
- ok-to-accept-du flag
- wheel byte
- path
- named directory entries

H.2 Command Processing

H.2.1 Hierarchy

In Foreground CP/M, BGii processes the command according to the following hierarchy:

1. IF error

```
IF (external cmd processor exists)
       AND NOT (ext cmd proc error)
    GO TO external command processor
ELSE
    GO TO command processor error routine
```

2. ZCPR3 shell
3. ZCPR3 flow-control
4. BG built-in commands
5. OCP (overlay command procesor) commands
6. RCP commands
7. External commnd processor
8. COM file on search path

H.2.2 Interface Details

H.2.2.1 Wheel Byte

In BGii and in the OCP, setting bit 7 of the first byte of a command name makes it a wheel-only command. See here.

H.2.2.2 Environment Descriptor

BGii passes the external environment address to the OCP, RCP and to COM files in register pair HL. If there is none, HL contains 0000.

H.2.2.3 Automatic Tool Configuration

BGii automatically configures any ZCPR3 tool (a COM file having a "Z3NEV" string at 103h) by installing the environment address (or 0000) at 109h when it is loaded. ZCPR3 tools can therefore be used in different sized systems without manual installation.

H.2.2.4 External Command Processor

If the command-run flag is activated when BGii is configured, BGii uses (by default) CMDRUN.COM as the external command processor, or if configure, the user/drive/filename pointed to by the address supplied.

H.2.2.5 State Determination

The Z80 Z flag is set (zero) if Foreground CP/M is active; it is NZ if the OCP/RCP command has been called in Background CP/M. the active task—upper or lower—can be determined from the bit 1 of the flags byte at:

```
    subtract 1602h from the address at 0001h
    subtract 1 from that address
```

BGii itself is identified by the signature "BGii". Its address:

```
    subtract 1602h from the address at 0001h
```

H.2.2.6 Loading Named Directories

The NDR command will load a named directory to the in-memory ZCPR3 named-directory buffer if it exists, otherwise to BG's swap file. BG does *not* check the size of an NDR file before loading it to memory.

H.2.2.7 Exiting from BGii

If the RCP and/or IOP buffers are used for BGii code, the corresponding pointers and size bytes in the environment are set to 0. If `BG OFF` is run, the RCP and/or IOP must be reloaded with LDR. This can be automated with an alias that might be called "BGOFF".

H.3 Miscellaneous

Key definitions attached to .COM files will not be automatically loaded if a shell invokes the program (but they can be manually loaded with the background `KEYS` command, which could be included in an alias).

Swapping is not possible while running ZEX and other tools that patch the BIOS. However, `SWAP` can be included in a SUBMIT script.

BGii's terminal data is determined by SETTERM and SETBG. BGii itself ignores the ZCPR3 termcap (but it is of course available for user programs).

User running ZRDOS should be aware that bugs have been discovered in version 1.1 - 1.7. Contact your supplier for details. BGii disables the warm-boot trapping feature of ZRDOS v 1.7 to work around one such bug.

Appendix I

Resident System Extensions

I.1 Compatibility

Because BackGrounder ii replaces the standard command processor in memory and remains resident, other resident system extensions must either reside above the command processor (in or above the BIOS) or be loaded after BGii and reside below BGii. The preferred method is to have system extensions reside in or above the BIOS and loaded before BGii is installed. Plu*Perfect System's DateStamper, for example, can be configured to load in highest memory.

Compatiblity with other resident system extensions (RSX[47]) is a complex issue.

BGii is generally compatible with device drivers located above the BDOS and loaded prior to BackGrounder ii. However, it conflicts with other resident system extensions that themselves replace the command processor or are required to reside immediately below it.

BGii requires that any disk-driver system extensions (ramdisk, hard disk, and format-altering modules) be loaded before BGii is loaded. This means that those modules must be located within or above the BIOS. Most external ramdisk and hard disk drivers have an option to do this.

Resident system extensions that do not conform to the BG RSX standard and are loaded below BGii will be swapped out by a **`SWAP`** command. If a program then causes a system extension to be used in the alternate task it will crash.

System extensions can be caused to stay resident when swapping occurs by coding them with the BG RSX header specification,[48] which is described later in this appendix.

Alternatively, the BGii utility SECURE.COM can be run after the system extension has been loaded. SECURE protects memory by installing the stnadard BG RSX header just below the currently-resident system extensions. If you have an important system extension that you wish to use with BGii, test it first by loading BGii, loading the extension, and then running SECURE. SECURE can only be run in the upper task, not the lower task.

Because BGii provides built-in keyboard macro capability and the SPOOLER modlue provides print spooling and background printing, major features of many reisent programs are already available. If you do use a resident program that does not conform to the BG RSX standard, SECURE it before using the SWAP command.

I.2 BG RSX Standard

Resident system extensions provide a standardized metod of adding semi-permanent features to the basic CP/M operating environment. Plu*Perfect Systems has developed a uniform specification for extension modules that provides these key features:

- mutual compatability of several modules
- removability of modules (in reverse order of loading
- standardized loading of modules

This method of extending basic operating system functions is used extensively in Plu*Perfect products—BackGrounder ii, its major resident utilities (SPOOL, SECURE and BGPRINT), and DateStamper (which is not, however, removable). The same method was also used for The Backgrounder. The Backgrounder Accessory Disk contains source code for BGPRINT, a relatively simple system extension. It can be used as is, enhanced, or used as a model for customized system extensions. Be sure to read the RELEASE.NOT file on that disk if you are using on anything but a Kaypro terminal.

I.3 Module Functions

Each module consists of an 18-byte header in a standard format, several standard functions, and custom routines that perform the module's particular functions. The header includes the following information:

1. at its lowest address, a jump to the BDOS entry point
2. a jump to the module's warm-boot routine
3. a jump to the module's remove routine
4. the address of the BIOS warmboot entry point
5. the address of the first byte of the module that must be protected
6. the address of the module's name
7. a jump to the warmboot routine of the next (higher) module

The required module functions are module_warmboot, mdoule_remove, and module_name_string.

Module_warmboot must:

- restore the BIOS warmboot address at 0001
- if there is no lower resident system extension module, install the module's protect address at 0006
- jump to the next warmboot routine

Module_remove must:

- restore all addresses in the operating system that were patched when the module was loaded
- set the carry flag

Module_name_string:

- a nul or hi-bit terminated ascii module name and version string

The remaining module code performs the custome functions of the module.

It helps to understand the relationship of different modules by referring to the memory maps, shown below.

I.4 Loader Functions

In outline, the loader must:

1. Announce itself
2. Check the system environment to be sure that:
    1. no non-ermanent (non-RSX) module is in memory
    2. that BackGrounder ii is in memory, if required.
    3. that the module to be loaded is not already in memory
3. Install standard module parameters
    1. calculate system addresses
    2. relocate and move the resident module code to its high location
    3. install header addresses
    4. install warmbot and remove address
4. Install module-specific parameters
    1. divert module-supplied bios functions
    2. other parameters
5. Install module data from the loader configuration area (if any)
6. Announce a successful installation of the named module
7. Jump to a warmboot

The loader must know the relevant addresses in the system extension module in order to patch the module into the control chain. In the example this communication is accomplished by an inelegant, but workable device—a parameter record that is shared by both the loader and the overlaying module code. By assembling each with the same labels at the addresses in that record, the module is able to pass its needed address offsets to the loader. Our convention is that the module area 000H - 00FFH is reserved for parameters, followed by the module code beginning at 0100H. A debugger (DDT) is used to overlay the module and its bitmap onto the end of the loader.

The layout of the BGPRINT loader is:

```
0100        Loader code
...         ...
0700        relocation routine
0701-0702   length of module code
0800        parameter record
0900        Module code (0000 origin)
...         ...
0900+length module relocation bit map
```

I.5 Producing a Resident System Extension

Code for a resident system extension consists of the high module, which will remain resident until removed (or cold-boot), and the module loader. The module itself is most conveniently written for a relocating assembler; the assembled code is then relocated by the loader, which also patches the module's linkages into the BIOS chain.

The Accessory disk includes a complete example, BGPRINT. We have written it for the Digital Research RMAC assembler and LINK-80 linker. LINK-80 produces pagerelocatable (PRL) files which contain the code relative to an origin of 0000 and a bitmap of the code bytes that are relocatable.

BGPRINT.ASM is the source file for the relocatable high module; LOADBGPR.ASM is the source for the loader. Each can serve as a prototype for developing custom system extensions, and only sections of the loader will need customaization for another application.

I.6 Memory Maps

Before any modules are loaded:

```
(bios) jmp vector)
WBOOT:  jmp     BIOSWB
CONST:  jmp     BIOSCS
CONIN:  jmp     BIOSCI
        ...
LIST:   jmp     BIOSLIST
        ...
```

After Module 1 is loaded:

```
(bios jmp vector)           (module header:)
WBOOT:  jmp     wbent1      prot1:  jmp     BDOSENTRY
CONST:  jmp     BIOSCS      wbent1: jmp     wb1
CONIN:  jmp     BIOSCI              jmp     remove1
        ...                         DW      WBOOT
LIST:   jmp     list1               DW      prot1
        ...                         DW      name1

                            uwboot1:jmp     BIOSWB

                            wb1:    restore 001h
                                    (if bottom module,
                                    restore 0006H)
                                    jmp uwboot1

                            remove1:
                                   restore WBOOT+1
                                   restore LIST+1
                                   set carry

                            list1:
                                   (custom code)
                                   jmp     BIOSLIST

                            name1: DB      'name1',0
```

After Module 2 is loaded:

```
(bios jmp vector)           (module header:)                (module2 header:)

                            prot1:  jmp     BDOSENTRY       prot2:   jmp     BDOSENTRY
WBOOT:  jmp     wbent2      wbent1: jmp     wb1             wbent2:  jmp     wb2
CONST:  jmp     BIOSCS              jmp     remove1                  jmp     remove2
CONIN:  jmp     BIOSCI              DW      WBOOT                    DW      WBOOT
        ...                         DW      prot1                    DW      prot2
LIST:   jmp     list2               DW      name1                    DW      name2
        ...                 uwboot1:jmp     BIOSWB          uwboot2: jmp     wbent1

                            wb1:    restore 001h            wb2:     restore 001H
                                    (if bottom module,               (if bottom module,
                                    restore 0006H)                   restore 0006H)
                                    jmp uwboot1                      jmp     uwboot2

                            remove1:                        remove2:
                                   restore WBOOT+1                   restore WBOOT+1
                                   restore LIST+1                    restore LIST+1
                                   set carry                         set carry

                            list1:                          list2:
                                   (custom code)                     (custom code)
                                   jmp     BIOSLIST                  jmp     list1

                            name1: DB      'name1',0        name2:   DB      'name2',0
```

There are several key points to observe here. As each module is loaded, it is patched into the BIOS vector/module chain to become the lowest link. Control flows from the BIOS jump vector to module n, the last-loaded module, then n-1, .. and finally to the internal BIOS routine. This linkage is required for the warmboot function. Other BIOS functions may appear in just a single module, or in several, as illustrated for LIST.

High memory is protected by the page-0 address at 0006H. On a warmboot, each module's warmboot routine is traversed. The lowest module's warmboot routine detects that it is indeed at the bottom of the chain, and has the responsibility for resetting location 6 to point to its protective jump vector; the higher modules must leave this address intact, but can take other action appropriate to their function. The warm-boot routines also restore the warmboot address at 0001H, in case an errant program has changed it. Note well that the restore routine in each module restores all patches that were made when it was loaded. Thus, if module 2 is removed, the memory map will again be that of the second memory map.

I.7 An Example: BGPRINT

BGPRINT is a fairly basic resident system extension that adds two new capabilities to the standard BIOS List function.

1. printer pause when a PAUSE_CHARACTER is sent
2. string substitution for specified special characters

These features are handy for several types of applications. When embedded in a text document, the pause character halts printing to allow the operator to change printwheels, ribbon, or sheet paper at an exact point in the text. String substitution permits access to special printwheel or dotmatrix characters that cannot be sent by the preferred wordprocessing software. It can also be used to send the printer's escape sequence to activate a different font or spacing.

As assembled, BGPRINT and LOADBGPR provide a simple demonstration of the pause and control strings for an OKIDATA 82/83 printer. We placed the printer-specific data at 103H in LOADBGPR, and you can readily patch in your own codes without reassembling and linking the package. However, the supplied configuration program provides a much simpler method of setting up your strings.[49] Embellishment of the basic BGPRINT is left as a useful exercise to the interested reader.

Although we called it BGPRINT, because the module was developed for use with The Backgrounder, this particular system extension does not require The Backgrounder or BGii for its operation; it should work with any standard CP/M 2.2 system. When used with BackGrounder ii, BGii should be loaded first.

Appendix J

List of Files

J.1 Main Files

i = installation only
l = required for loading, then removable
x = have on-line
o - optional on-line

FILE (req.)	**Function**
BG.HLP(x)	On-line help file for BG
BG.REL(lx)	The relocatable BG code.
BGINFO.HLP(x)	On-line help file for BG
EX14A.COM(o)	Corrected resident batch processor.
H19SCRN.DRV(i)	Heath/Zenith 19/89/90 screen driver.
K8384FNK.DRV(i)	Kaypro '83 and '84 functionkey driver.
K83SCRN.DRV(i)	Kaypro '83 screen driver.
K84SCRN.DRV(i)	Kaypro '84 screen driver.
LOADBG.COM(lx)	Load and activate BG.
MAKENDR.COM(o)	Create and edit Named DiRectories.
PUTBG.COM(ilx)	Create BG swap file.
Q.COM(x)	Manages the print queue.
REMOVE.COM(x)	Removes Backgrounder RSX modules from memroy.
S19FNK.DRV(i)	Super 19 Heath 19/89/90 functionkey driver.
SECURE.COM(o)	Protects high-memory resident modules. (RSX)
SETBG.COM(ix)	Configure user BG parameters.
SETTERM.COM(i)	Define & install terminal capabilities.
SPOOLER.COM(x)	Loads Spool/Unspool module. (RSX)
SYNONYM.COM(o)	Creates alias for command line.
TERMBASE.DAT(i)	Terminal capabilities database.
WSRDR30.HEX(i)	WordStar 3.0 redraw-screen function (Cntl-\) patch
WSRDR33.HEX(i)	WordStar 3.3 " "
ZEX31A.COM(o)	Corrected ZCPR3 resident batch processor.

J.2 Release Dependent Files

FILE (req.)	**Function**
*.DRV	Other screen drivers as available - see RELEASE.NOT
*.FNK	Other function key drivers as available - see RELEASE.NOT

H19PATCH.DOC	Info on patching various Heath 19 terminal roms to remove the 'bell' at the end of a transmit-screen-region command.
KAYPRO.BG	Arrow and numeric-keypad definitions for Kaypro.
WSARROW.BG	WordStar arrow-key definitions for Kaypro.

J.3 BG Accessory Disk Files

FILE (req.)	**Function**
BGPRCFG.COM	Configures BGPRINT.COM. (configured for Kaypro terminal)
BGPRINT.ASM	Source for BGPRINT; illustrates a BG RSX.
BGPRINT.LIB	Library files for BGPRINT (RMAC/LINK-80)
BGPRINT.SUB	Submit file to generate BGPRINT.COM
BGPRINT.COM	Remaps list-device characters. (RSX)
K83SCRN.ASM	driver source - for CDL/TDL assembler
K84SCRN.ASM	driver source - for CDL/TDL assembler
H19SCRN.Z80	Hal Bower's driver source - for M80 assembler
K8384FNK.ASM	Kaypro '83 and '84 functionkey driver (CDL assembler)
LOADBGPR.ASM	Source for BGPRINT RSX loader. (RMAC/LINK-80)
S19FNK.ASM	Super 19 heath 19/89/90 functionkey driver (CDL assembler

Endnotes:

[1]See Appendix I.

[2]It is compatible with DateStamper additions, but not other system patches. BGii will not run with CP/M emulators or other variants of the CP/M BDOS. The actual command processor can either be DRI's CCP or any version of ZCPR. The only limitation on the command processor is that it be overwritable.

[3]Naturally, a screen driver as described in Appendix B, customized for this terminal must also be present.

[4]SETTERM is a generic program that is also used with Plu*Perfect Systems' DateStamper. Ignore these other utility programs that are listed by SETTERM.

[5]A full chapter is devoted later in this manual to the SETBG options.

[6]If there is not a screen driver for your machine, some BGii features will not be available to you. This manual contains an appendix on writing your own screen driver. If you are not able to write your own screen driver or otherwise procure one, you should pay careful attention to the section in Chapter 6 on incorporating a "redraw" string into your key definitions.

[7]Without a function key driver, you may not be able to attach definitions to the function keys on your keyboard or use the `SHIFT` command. Appendix C gives instructions on writing a function key driver.

[8]The automatic size determination is based on a non-ZCPR3 system. If you are using the option that allows you to overlay some of the ZCPR3 buffer areas then the swap file wil be slightly too small to allow full TPA size in the alternate task. Simply use the “-k=nn” parameter of SEETBG to increase the swap file size by the amount reported when LOADBG is run.

[9]For those who have only limited disk capacity or are using a slow device, the command can have another parameter to limit the size of the swap file produced. By adding “-k=nn” to the command line where nn is the desired file size in Kilobytes a smaller swap file can be produced. A smaller swap file will limit the TPA available to the alternate task. The absolute minimum value is around 45K and this will totally inhibit any task swapping. In practice, do not use a value of less than about 80K if you plan to use the task swapping features of BGii.

[10]See Appendix E.

[11]LOADBG will reinstate the program.

[12]These may be created by following the directions in Appendix E./p>

[13]If this is not a convenient key, you may set it to something else by running SETBG. If you need to send this key to a running program you can prefix it with the backslash key (called the <ALT> key). In addition the <SUSPEND> key can be temporarily disabled with the `BG N` command (see the BG command).

[14] See the BG command. This can be turned off with the Q option of the **BG** command.

[15] This assumes you have also installed the correct screen driver for your computer. If you have not, you'll have to adopt the alternate method of recreating your screen. See Chapter 6 for general information on the redraw string and Appendix G for information on the WordStar redraw function.

[16] The Spooler will pause whenever you enter Background CP/M.

[17] The preferred way to handle this situation is to create a “device driver” as an RSX and do all hardware interaction through its interface.

[18] The notepad file created as a default is called JOTPAD.

[19])Or, depending on your driver, some other movement keys.

[20] See PASTEing into a JOTPAD and Pasting to a Key.

[21] The exact message depends on the screen driver that has been installed. Note that the escape sequences in this example are case sensitive—you must use upper case 'E' and 'R'.

[22] Keystroke recording is not possible at the CP/M prompt level.

[23] If this key doesn't suit you, you may change it to some other character by running SETBG.

[24] Unless you have used the option in SETBG which allows up to 8 of the standard keys to be re-mapped to other single characters. See Remapping Keys.

[25] See Recording Keystrokes for more discussion on the <RECORD> key.

[26] When definitions are attached to a program, the COM file is increased in length by one secotr and the name of the definition file is encouded in this sector. This technique is incompatible with Writer's Guide which use the same method to stor its current filename.

[27] See Startup Command.

[28] More accurately, the date at the moment LOADBG was run. After midnight, you must re-run LOADBG to get the correct date.

[29] See Date Format.

[30] See Key Delay.

[31] There is an easier way to get two DIRectory commands, by using BGii's multiple command line option:

```
DIR *.COM;DIR *.MSS
```

[32] See Chapter 7.

[33] See Appendix G for more details.

[34] For example the unmodified Perfect Writer. Plu*Perfect Writer does not have this problem.

[35] See Key Delay.

[36] Assuming that your alternate task is not also running a program.

[37] See Printer Resume

[38] See Appendix I

[39] See Insert CALC Result in the **KEYS** command for details.

[40] Location of JOTPAD file is configurable with SETBG.

[41] It should also be noted that BGii does *not* provide a ZCPR3 environment for programs which require it, unless there was one present when BGii was loaded.

[42] BGii uses HELP: named-directory, if one exists.

[43] See Appendices B, C and D.

[44] To receive printed copies of this source code, send your request plus $1 to cover shipping and handling to Plu*Perfect Systems.

The code is also avaiable along with sample RSX code on the The Backgrounder Accessory Disk for $14.96, but only in Kaypro SSDD format.

[45] See H/Z-19 driver for an example of WordStar cursor-control keys. Details on obtaining source code for the drivers is given in endnote 44.

[46] Details on obtaining the source code are given in endnote 44.

[47] This is the terminology used by Digital Research in their CP/M Plus Documentation.

[48] This is how SPOOLER works.

[49] Note that the configuration utility is set up for a Kaypro terminal. You must use SETTERM to modify this if you are not using a Kaypro.