BUG/65

JPTRO, Inc/Andrew Jacobs, P.E.
Copyright © 1983-2006 JPTRO, Inc
JPTRO@JPTRO.COM
http://www.JPTRO.com/BUG65.html

BUG/65

Inc and JPTRO Logo are trademarks of JPTRO, Inc
Apple II, Apple IIe, Apple IIc, Apple IIGS, Apple II Plus, Apple II Clone, Apple II Compatible, Apple II Family, Apple II Family Compatible, Apple II Family Clone, Apple II Family Compatible Clone, Apple II Family Clone Compatible are trademarks of Apple Computer, Inc. Apple Computer, Inc. 1 Infinite Loop Cupertino, CA 95014 (408) 996-1010
Apple Computer, Inc. 20525 Mariani Avenue Cupertino, CA 95014 (408) 996-1010

This manual is copyrighted by JPTRO, Inc. Reproduction of any part of this manual in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from JPTRO, Inc. is prohibited. Permission is granted to reviewers to quote brief passages in connection with a review written for publication in a magazine or newspaper.

PREFACE

BUG/65 is an interactive debugging tool for use with Apple II family and compatible computers equipped with a 6502 or 65C02 MPU (general computers). It is designed to take up most of the memory in the computer, so it can be used to debug programs that are being developed in the computer's memory. It can also be used to debug programs that are in ROM, or that are being developed for other computers. It is also a very powerful tool for examining and modifying memory, and can be used to debug programs that are not working properly.

BUG/65 is a product of JPTRO, Inc. BUG/65 is a trademark of JPTRO, Inc. Apple II family and compatible computers are trademarks of Apple Computer, Inc. Apple Computer, Inc. is not affiliated with JPTRO, Inc.

BUG/65 was first written in the 6502 MPU language, and then translated into the 65C02 MPU language.

The 65C02 MPU is a product of the 6502 MPU company, which is designated by Apple Computer, Inc. as the "65C02", and which is also known as the "65C02 MPU".

You can do 16-bit BUG/65 in 65C02 MPU computers, and a minimum of 8K of RAM is needed. Happy 65C02!

TRADEMARKS

The following trademarked items are used in various places throughout this manual, and in BUG/65:

Apple	BUG/65, BUG/65 Inc and JPTRO Logo are trademarks of JPTRO, Inc. Apple II, and Apple Computer are trademarks of Apple Computer, Inc.
Apple	Apple II, Apple IIe, Apple IIc, Apple IIGS, Apple II Plus, Apple II Clone, Apple II Compatible, Apple II Family, Apple II Family Compatible, Apple II Family Clone, Apple II Family Compatible Clone, Apple II Family Clone Compatible are trademarks of Apple Computer, Inc.

TABLE OF CONTENTS

Summary of Major Features

Section 1 - Command Summary

Section 2 - Notations used, syntax

Section 3 - Address Parameters

- 3.1 - Spaces as Delimiters

Section 4 - Loading and Running BUG/65

- 4.1 - Specifying BUG/65's Load Address
- 4.2 - Creating a Non-Relocatable Version

Section 5 - Command Entry

- 5.1 - Command Line Editing
- 5.2 - Normal and Immediate Commands
- 5.3 - Command Execution
- 5.4 - Multiple Commands on a Line

Section 6 - Command Descriptions

- Normal Notation
- Error Notationn
- Command Syntax
- Command Short
- Command Long
- Alias
- Normal Format

Section 7 - Detailed Command Descriptions

- ? - A/X/Y/S/P/PC/A/Y/Stage
- ! - Set/Reset/Toggle
- ; - Computer Interface Mode
- (- Disable Interrupts
-) - Enable Interrupts
- ~ - HEX/ASCII/DEC
- ^ - Change Current Stack Pointer
- ^ - Current Stack Pointer
- ^ - Current Stack Pointer
- ^ - Set/Reset/Toggle Interrupts
- ^ - Set/Reset/Toggle Trace
- ^ - Change Current Stack Pointer
- ^ - Change Current Stack Pointer
- ^ - Current Stack Pointer
- ^ - Current Stack Pointer
- ^ -

7.22 X - Change User's Registries

```
1 REGEDIT
```

The user should use or change the number of user registries. The default registries are 1 and 2. The default registries are the default registries and are the standard ones.

```
0 = system
1 = user
2 = system
3 = system
4 = program registries
5 = program registries
```

Only or type in the name of the registries to be changed. RED will prompt you with the current registries and ask for the new one. Type '?' for help and you can see the defining options.

```
REGEDIT [0|1|2|3|4|5|?] [/?] [/V] [/S] [/D] [/R] [/W] [/X] [/Y] [/Z]
```

/V is for verbose, /S is for silent, /D is for delete, /R is for read, /W is for write, /X is for exit, /Y is for yes, /Z is for zero. Without /V and /Z you only get a single line of output. /V and /Z are useful for scripts. /Z is useful for scripts that need to know the exact registry value. /V is useful for scripts that need to know the registry value and the registry key. /S is useful for scripts that need to know the registry key. /D is useful for scripts that need to delete a registry key. /R is useful for scripts that need to read a registry key. /W is useful for scripts that need to write a registry key. /X is useful for scripts that need to exit. /Y is useful for scripts that need to say yes. /Z is useful for scripts that need to zero the registry value.

7.23 Y - Disassemble Memory Block

```
1 DISASSEMBLE
```

This is the command to disassemble instructions in memory. The default is 256 bytes. Type '?' for help. The syntax for the command can be used the following:

```
DISASSEMBLE [?]
```

Flags and options are displayed by '?'.

```
/? Displays this help and exits.
/V Verbose output, displays the disassembled instructions.
```

Enter page command will display up to two bytes, all other keys will exit the command. Type '?' for help.

The operand is displayed in a complete mode operands.

7.24 Z - Instant Assembler

```
1 ASSEMBL
```

The A command allows you to assemble instructions in the normal mode. The A command allows you to assemble instructions in the normal mode. The A command allows you to assemble instructions in the normal mode.

BUG/65

Copyright © 1983 by Jim Butterfield
Permission is granted to reproduce this manual in whole or in part, provided that the copyright notice is included and the reproduction is for personal use or non-profit purposes.

Version 1.0 (Revision 1)
Revision 1 Printing
Revision 1 Printing
Revision 1 Printing

6.2 Error Termination

If an error occurs in a command execution, RECOS will stop the execution of the command and return to the command input state. The error message will be displayed on the screen. The error message will be displayed in the following format:

error message (error code) in command name

For example, if an error occurs in the command "PRINT" and the error message is "Illegal Parameter" and the error code is 1, the error message will be displayed as follows:

Illegal Parameter (1) in PRINT

6.3 Command Suspension

Many RECOS Basic operating commands have a "suspension" function. When a command is being executed, if the "Esc" key is pressed, the command will be suspended. The command will stop at the next suspension point. If the command does not have any suspension points, it will continue to run. The suspension function is enabled by default.

6.4 Command Abort

You can also stop a command from its running process by the "Esc" key. The command will stop at the next "Esc" key check point. The command will be terminated. The command will return to the command input state.

6.5 The RESET Key

RECOS uses the RESET key to the left of the keyboard RESET and the keyboard scan key to check for the "Esc" key. The keyboard scan key is enabled by default. The RESET key is a hardware signal that generally initiates a reset process in the computer hardware. The RESET key is also used to initiate a reset process in the RECOS hardware. The RESET key is also used to initiate a reset process in the RECOS software. The RESET key is also used to initiate a reset process in the RECOS software. The RESET key is also used to initiate a reset process in the RECOS software

7.21.2 SPL - Write Sectors

```
WRITE SECTORS (SECTOR(S),STRAND,TRK)
```

The SPL command is used to write sectors to a diskette or hard disk. The sector(s) to be written are specified by the sector number, strand number, and track number. The location of the sector data buffer cannot be specified in the command. The sector data buffer is the same buffer used by the read sectors command. The sector(s) to be written must be in the sector data buffer before the write sectors command is executed. The sector(s) to be written must be in the sector data buffer because the write sectors command does not specify the sector data buffer. The sector(s) to be written must be in the sector data buffer because the write sectors command does not specify the sector data buffer.

7.22 X - Change User's Registers

```
X,ACCUMUL
```

The X command allows you to change the contents of one of the user's registers, or the accumulator, in hexadecimal. The allowable values are:

- 0 = accumulator
- 1 = x-register
- 2 = y-register
- 3 = program counter pseudo register
- 4 = stack pointer pseudo register

Enter the new value for the register in the changed prompt. The new value must be in hexadecimal. The program counter and stack pointer pseudo registers are used to specify the location of the next instruction and the top of the stack when the user's program is executed. The current contents of the program counter and stack pointer are displayed in the changed prompt. If the program counter or stack pointer registers are changed, the new value will be used when the user's program is executed. If the program counter or stack pointer registers are not changed, the current value will be used when the user's program is executed.

7.23 T - Disassemble Memory Block

```
T,START,END
```

The T command allows you to disassemble instructions in the memory in hexadecimal. The allowable values are:

- Enter the start address in hexadecimal.
- Enter the end address in hexadecimal.

The T command is used to disassemble instructions in the memory. The start and end addresses are specified in hexadecimal. The start address is the address of the first instruction to be disassembled. The end address is the address of the last instruction to be disassembled. The T command will disassemble instructions in the memory from the start address to the end address. The T command will disassemble instructions in the memory from the start address to the end address. The T command will disassemble instructions in the memory from the start address to the end address.

7.24 Z - Instant Assembler

```
Z,[ORG]
```

The Z command allows you to assemble instructions in the memory in hexadecimal. The allowable values are:

- Enter the origin in hexadecimal.
- Enter the opcode or mnemonic in hexadecimal or decimal.
- Enter operands in hexadecimal or decimal.

The Z command is used to assemble instructions in the memory in hexadecimal. The origin is specified in hexadecimal. The opcode or mnemonic is specified in hexadecimal or decimal. The operands are specified in hexadecimal or decimal. The Z command will assemble instructions in the memory from the origin. The Z command will assemble instructions in the memory from the origin. The Z command will assemble instructions in the memory from the origin.

The Z command is used to assemble instructions in the memory in hexadecimal. The origin is specified in hexadecimal. The opcode or mnemonic is specified in hexadecimal or decimal. The operands are specified in hexadecimal or decimal. The Z command will assemble instructions in the memory from the origin. The Z command will assemble instructions in the memory from the origin. The Z command will assemble instructions in the memory from the origin.

Z80 opcode mnemonics are not supported in the instant assembler.

The instant assembler is used to assemble instructions in the memory in hexadecimal. The origin is specified in hexadecimal. The opcode or mnemonic is specified in hexadecimal or decimal. The operands are specified in hexadecimal or decimal. The instant assembler will assemble instructions in the memory from the origin. The instant assembler will assemble instructions in the memory from the origin. The instant assembler will assemble instructions in the memory from the origin

In the macro assembler, any macro references have been made to an undefined label. BOGUS allows macro references to undefined labels before it label further overflows.

??? ERROR - UNDEFINED - 1a

In the macro assembler, a label has been referenced but not defined. '0' is the relo number that needs definition.

Appendix

This section applies only to those users who were version 4 of BOGUS.

The version of BOGUS which you received is not directly compatible with version 4 of BOGUS. Instead, the new disk contains a new program called BOGUS.COM. Running this program will convert your old version 4 files into new version 5 files compatible with version 5. This program, BOGUSFIX.COM, is a minor program that modifies the relocatable version of BOGUS.

The relocatable version of BOGUS.COM will work ONE TIME only on a version 4 BOGUS.COM file. If you forget this, the command will not work properly under all conditions. This suggests instead using the [NONE] [COM] command for loading binary files into memory, although the new .CM produced by the B command may simply be ignored, if desired. Only location 300 is improperly affected by this error.

HOW TO USE THE PROGRAM:

1. Copy the files BOGUS.COM and BOGUSFIX.COM to a system volume using the COPY.CM command (see the [NONE] manual for details on this command).
2. At the system's "D>" prompt, type the command BOGUSFIX.[NONE]
3. The file BOGUS.COM is that which is now compatible with version 5 of [NONE].

WARNING: Do NOT perform the BOGUSFIX command on your version 5 master disk!