

# R65 System Software

1. General description of the basic R65 software.....	3
R65 CRT Controller:.....	3
R65 Input/Output Controller:.....	3
R65 Floppy Disk Controller:.....	3
The R65 System Monitor:.....	3
2. STOP, QUIT and SHUTDOWN.....	3
3. Installation.....	4
Installing the mandatory fonts.....	4
Starting the emulator.....	4
Adding additional disks.....	4
Upgrading the R65 Emulator.....	5
4. R65 Emulator front panel.....	5
5. Monitor Commands.....	6
CLRB exp1.....	6
COPY filnam[.cy],[drive],to_drive.....	6
DATE.....	6
DELETE filnam[.cy],[drive].....	7
DFORMAT diskname,drive.....	7
DIR drive.....	7
DIS [exp1],[exp2].....	7
EDIT filnam[.cy],[drive].....	7
EXPORT filnam[.cy],[drive].....	7
FDIR [drive].....	7
FLOPPY disknam[,drive].....	7
GO [exp1],[register definitions].....	7
GSB [exp1],[register definitions].....	7
IMPORT filnam[,drive].....	7
LOAD filnam[.cy],[drive],[exp1].....	7
NEW filnam[.cy],[drive].....	7
PACK [drive].....	7
PROTECT filnam[.cy],[drive].....	8
PRTB.....	8
REG [register definitions].....	8
RENAME filnam[.cy],[drive].....	8
REVIVE entry number[, drive].....	8
RESETB.....	8
RUN filnam[.cy],[drive],[exps1].....	8
SAVE [drive].....	8
SETB exp1.....	8
STEP [exp1].....	8
STORE filnam[.cy],[drive],exp1-exp2,[P]S,[exp3].....	8
SWAP [drive].....	8
TRACE [exp1,exp2].....	8
VOLUME [drive].....	8
/exp1.....	8
6. PASCAL commands.....	9
ARGLIST arguments.....	9
BOUNCE.....	9
CALC.....	9
CLEAN [drive].....	9
COMPILE filnam[.cy],[drive] [options].....	9
COMPILE 1 filnam[.cy],[drive] [options].....	9
COMPILE 2 filnam[.cy],[drive].....	9
COPY wfilnam[.cy],[sourcedrive] desdrive.....	9

DELETE wfilnam[.cy][,drive].....	10
DIR [drive].....	10
EDIT filnam[.cy][,drive].....	10
ERROR errorcode.....	10
EXTIME pname,drive arguments.....	10
FIND filnam.....	10
FLOPPY fname[,drive].....	10
GETSOURCE pname.....	10
GRAPH.....	10
GRTEST.....	10
GRTEST2.....	10
GR3D.....	10
LEDTEST.....	10
MAKETABLE.....	10
NEW filnam[.cy][,drive].....	11
PACK [drive].....	11
PEDIT filnam[.cy][,drive].....	11
PUTSOURCE pname.....	11
PUTOBJECT pname.....	11
SINETST.....	11
TEKTEST.....	11
TEK3D.....	11
TGRAPH.....	11
THROWSIM.....	11
7. Pascal libraries.....	11
ARGLIB.....	11
LEDLIB.....	11
MATHLIB.....	11
POTLIB.....	11
RALIB.....	12
SYSLIB.....	12
TEKLIB.....	12
TIMELIB.....	12
8. Pascal games.....	12
ALIEN.....	12
PONG.....	12
REVERSI.....	12
9. R65 Graphic Basic.....	12
CLOSE [D1].....	12
DIR [D].....	12
FILES.....	12
GET [#D1:] V1 [,V2...].	12
INPUT [#D1] V1 [,V2...].	12
LOAD S1[.D1] [,D2}.....	12
LOAD #D1;.....	13
MOVE N1,N2.....	13
OPEN D1,S1[.D2][,D3,[D4]].....	13
PLOT N1,N2[,D1].....	13
PLOT NEW.....	13
PLOT CLR.....	13
PLOT STOP.....	13
PLOT CONT.....	13
PLOT END.....	13
PRINT [D1:] .....	13
STORE S1[.D1] [,D2].....	13
SYS N1.....	13
WAIT N1.....	13
10. Editing files.....	13
11. Floppy disk drives (tapes are not emulated).....	14

12. Attaching a printer/plotter.....	14
13. File Type.....	15
14. Error Codes.....	15
System error codes.....	15
ASSEMBLER error codes.....	16
EXDOS error codes.....	17
PASCAL error codes.....	17
15. R65 Control keys.....	17
Video control functions.....	17
Other control functions.....	18
16. “Burning” new PROMS/EPROMS.....	18
17. The usual disclaimer.....	19

## 1. General description of the basic R65 software

This manual describes the system software for the R65 computer system. The system software consists of 4 separate modules, which are stored in EPROM.

### R65 CRT Controller:

Handles the Video interface and the keyboard.

Address space: E000-E7FF

### R65 Input/Output Controller:

Routines for printing on a RS-232 needle printer, interrupt handling and audio tape io routines. The audio tape is not emulated in the emulator. Printing to a printer is emulated by printing to a text file printout.txt in the working directory of the emulator.

Address space: E800-Efff

### R65 Floppy Disk Controller:

Handles 2 floppy disk drives. 80 tracks, 10 sectors of 256 bytes per track. Includes the subroutines for sequential file i/o.

Address space: F000-F7FF

### The R65 System Monitor:

Includes all necessary commands to work with Machinge Language Programs using hex numbers.

Address space: F800-FFFF

## 2. STOP, QUIT and SHUTDOWN

The following keyboard keys are recognized by the R65 Emulator and executed immediately. The buttons on the top left of the screen can also be touched or clicked.

<shift>ESC or STOP button:

Emulate a 6502 non mascable interrupt (NMI)

<shift>MENU or <shift>WINDOWS or QUIT button or x(close window):

Quit R65 emulator, go to dektop

<shift>ALT SHUTDOWN:

Quit R65 emulator and shut down Linux

Note that open sequential files are not closed if one of these keys or buttons are executed. This is not a problem if a sequential file read is in progress, but in the case of a sequential file write this leaves a large useless file on the disk. Deleting this file afterwards and using PACK on the disk will solve the problem. Try to avoid therefore these actions if a sequential disk write is in progress. Disks are closed properly.

### 3. Installation

The R65 Emulator for Raspbian can be downloaded from GITHUB. Open a terminal and type:

```
cd
git clone --depth 1 git://github.com/rricharz/R65
cd R65
cp R65-Emulator.desktop ~/desktop
cd R65-Emulator
mkdir Files
cd Disks
cp EMPTY.disk WORK.disk
```

#### Installing the mandatory fonts

```
cd
cd R65/R65-Emulator/Fonts
sudo ./install_fonts
```

#### Starting the emulator

Now you should be able to start the R65 emulator using the R65 Emulator icon on the desktop. Read chapter 5 Floppy Disk Drives before proceeding.

You can also start the emulator using a terminal window:

```
cd
cd R65/R65-emulator
./emulator
```

“emulator” has the following options:

- full            Display the emulator in a full screen window. Looks much more pretty.
- pixelated      Use a pixelated font for the R65 screen. This resembles the original. Works best together with -full.

As the first thing, you should rename your newly created WORK disk.

Type FLOPPY WORK,1 to ensure that the WORK disk is in drive 1.

Type VOLUME 1<return> and then enter WORK.04 <return> to label your work disk properly.

#### Adding additional disks

Later, you can create additional disks with the name MYNAME (where MYNAME stands for any name you want to use) as you have done above for WORK. In the Raspbian terminal, type

```
cd
cd R65
cd R65-Emulator/Disks
cp EMPTY.disk MYNAME.disk
```

Then, restart R65-Emulator using the Desktop icon.

Type FLOPPY MYNAME,1 to ensure that the MYNAME disk is in drive 1.

Type VOLUME 1<return> and then enter MYNAME.05 <return> to label your new disk properly. Instead of

05 you can give the floppy any number you wish. It is strongly recommended to label your disks with the same names as the name of the disk file in Raspbian in order to avoid confusion. On the top status line of the R65 emulator the name of the Raspbian disk file is displayed.

## Upgrading the R65 Emulator

If you have not modified the PASCAL, SOURCE and PROGRAMS disks, upgrading is easy. If you have modified these disks, you need to first make a backup of your modifications on any other disk. It is necessary that the upgrading process upgrades these disks, because most upgrades are likely programs, libraries and source files on these disks. Also, some of these programs might have been modified to work together properly with any upgraded version of the Emulator.

Type

```
cd
```

```
cd R65
```

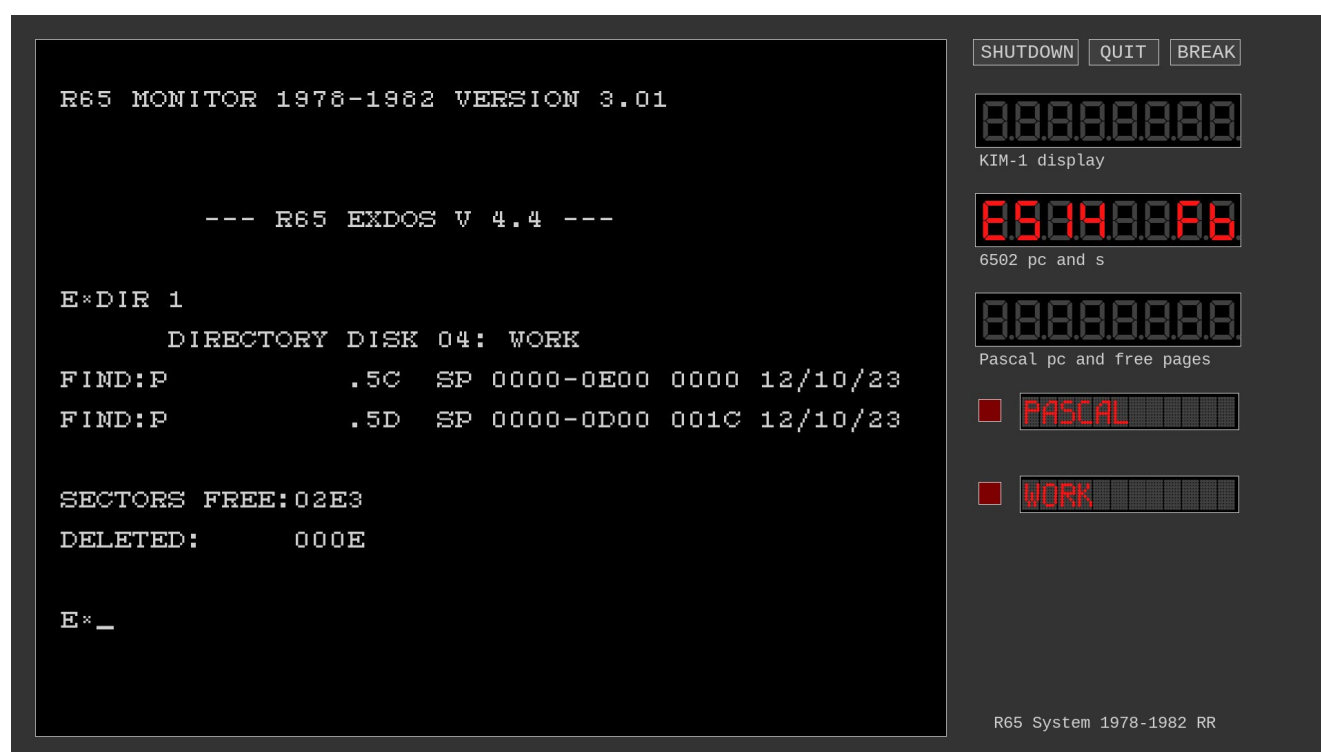
```
git pull
```

The git pull file will give an error message, if any file which will be upgraded has been modified. Type

```
git checkout xxx/xxx
```

where xxx/xxx is the path and file name for a file which git pull needs to overwrite. Once all the error messages of the git pull command have been corrected, you can perform the git pull to upgrade your software.

## 4. R65 Emulator front panel



On the left side is the **CRT screen**, using a pixelated font to resemble the original, while still being readable. The original had a 5x7 pixel character set, while the emulator uses a 7x11 matrix to improve readability on modern high resolution computer screens. It is shown if the emulator is started with the - pixelated option. Without this option, a modern monospace font is used. Do not forget to install the required fonts as described in the installation section of this manual.

On the top right are 3 buttons, which can be pressed with the mouse. "Break" executes a 6502 NMI break as on the original KIM-1 board. "Quit" exits the emulation. "Shutdown" exits the simulation and runs the host Raspberry Pi or Linux system down. On suitable keyboards, <shift><esc> executes the break button, <shift><menu> executes the quit button, and <shift><alt> executes the shutdown button.

The "KIM-1 display" resembles the original KIM-1 seven segment led display. The original had a 6 digit

display with a space between the 6<sup>th</sup> and 7<sup>th</sup> character, which was driven by the R65 software using a background, interrupt driven program. This code was unfortunately lost, so that the R65 emulator handles this in the emulator code, using a 8 digit 7 segment display. This display is under control of user programs written in 6502 code or Pascal. See the Pascal "ledlib" library for an example. As long as the user hasn't written anything onto this display, the emulator shows the current time.

The "6502 pc and s" seven segment display shows the 6502 program counter (pc) and stack pointer (s) in hex. The program counter is frozen while the program waits for user input, even so on the original this was handled in a short 6502 code loop. But to avoid overheating of the host CPU of the emulator this is now done in the emulator software with the proper Linux functions. The stack pointer can be used to monitor the 256 byte stack of the 6502. A too low number suggests that the stack may soon overflow and the program may crash. Typically this happens if a recursive subroutine is called with too many levels of recursion, or if too much data is pushed on the stack, using the corresponding 6502 instructions such as PHP or PHA. There is a delay of 3 seconds at the currently lowest number to allow to monitor this critical level.

The "Pascal pc and free pages" seven segment led display shows the pascal program counter in decimal and the number of free 256 byte pages in hex. The program counter of each program line is shown in the Pascal program listings as printed by the Pascal compiler. The number of free pages is the free memory between the top of the Pascal user memory (topmem), or the heap pointer if heaplib is used, and the Pascal stack pointer, which moves upwards. This should never go below approximately 5 pages to allow the Pascal interpreter enough space to breath. As on the display of the 6502 stack there is a delay of 3 seconds at the currently lowest number to allow to monitor this critical level.

Below the seven segment displays are two LEDs, which glow when the two floppy drives are spinning. Because it took some time to spin up a floppy drive on the original, the drives were kept spinning for several seconds after a read or write was executed. Constantly spinning the drives was not possible to minimize wear out of the floppies. To the right of these LEDs are two dot matrix displays showing the names of the floppies in the drive. I always wanted to have these displays, but they were not yet available in the late 1970s.

## 5. Monitor Commands

The following abbreviations are used:

exp1	A hexadecimal expression
filnam	A file name of up to 16 characters
wfilnam	File name with wild cards (* and ?)
drive	0 or 1 for the two floppy drives
cy	a cyclus number for a file
[ ]	These arguments can be omitted

### CLRB exp1

Erase breakpoint at address exp1.

### COPY filnam[.cy],from\_drive,to\_drive

Copy a file from from\_drive to to\_drive. The drives must both be given and must be different. The wildcard character @ can be used in filnam. Sequential and block files can be copied. The maximal file size to be copied is \$9000 (32768) bytes. The memory space \$2000-\$BFFF is used as a buffer. Extremely large sequential files (such as COMPILE1:P and ASSEMBLER:A) cannot be copied using the copy command. Use EXPORT and IMPORT for these very large files. Do not use the combination of EXPORT and IMPORT for any files except :A and :P files.

### DATE

Display actual date. Date can be changed once displayed. <return> returns to monitor. The emulator reads

the Linux system dates at startup and stores it in the R65 date memory. Afterwards it can be changed if necessary.

### **DELETE filnam[.cy][,drive]**

Delete a file. The wild card letter @ can be used in filnam.

### **DFORMAT diskname,drive**

Erase the disk directory completely and create a new directory

### **DIR drive**

Display directory of a disk drive.

### **DIS [exp1][,exp2]**

Disassembles machine language program at address exp1 and following. exp2 (default 10) commands are decoded. Use <esc> to return to the monitor.

### **EDIT filnam[.cy][,drive]**

Edit a file with the Linux mousepad editor. Use the command NEW (see below) if you want to edit a new file.

### **EXPORT filnam[.cy][,drive]**

Export a sequential file to the Linux Files directory

### **FDIR [drive]**

Full disk directory, including deleted files which can be recovered with the revive command. Also shows how many sectors can be recovered with the pack command.

### **FLOPPY disknam[,drive]**

Change the floppy disk in drive (default 0) to the one with the name disknam. Floppy disks are stored in the Linux Disks folder with the extension .disk, but disknam must be given without this extension. Additional floppies can be made there by using Linux to make copies of existing disks. The emulator can handle an unlimited number of floppies, but only 2 floppies can be in the 2 drives at the same time.

### **GO [exp1][,register definitions]**

Start executing the machine language code at address exp1. See command REG for the register definitions.

### **GSB [exp1][,register definitions]**

Same as GO, but instead of a jmp to the start address a jsr (subroutine call) starts the execution.

### **IMPORT filnam[,drive]**

Import a Linux file from the Linux Files directory. The Linux file must have the extension .asm or .txt, but filnam must be given without this extension.

### **LOAD filnam[.cy][,drive][,exp1]**

Load a disk file with name filnam (and cyclus) from drive to the stored memory address. If exp1 is given, the file is loaded to the address exp1.

### **NEW filnam[.cy][,drive]**

Create a new empty sequential file.

### **PACK [drive]**

Permanently erase all deleted files and recover the disk space. FDIR displays how much space can be recovered with the PACK command.

## **PROTECT filnam[.cy][,drive]**

Protect a file. Delete will ask for permission before deleting protected files.

## **PRTB**

Display the active breakpoints.

## **REG [register definitions]**

Display saved CPU registers, allow to change them. Register definitions are name=exp[,...], where the register names are P,S,F,A,X,Y. Use <esc> to return to the monitor.

## **RENAME filnam[.cy][,drive]**

Rename an existing file. The command asks for the new name and subtype. The file is protected if subtype is preceded with !.

## **REVIVE entry number[, drive]**

Recover a deleted file. Use FDIR to get the entry number.

## **RESETB**

Clear all active break points.

## **RUN filnam[.cy][,drive],[,exps1]**

Loads a file like load, but starts it afterwards automatically. The file type must be "M" (machine language).

## **SAVE [drive]**

Make a backup copy of the disk directory on the last track of the disk.

## **SETB exp1**

Set a breakpoint at address exp1

## **STEP [exp1]**

Execute one machine language instruction. Use <esc> to return to the monitor.

## **STORE filnam[.cy][,drive],exp1-exp2,[P]S[,exp3]**

Store a file from memory exp1-exp2, file type S. If exp3 is not given, the address for LOAD is exp1, otherwise exp3.

## **SWAP [drive]**

Exchange the disk directory with the one backed up with save before.

## **TRACE [exp1,exp2]**

Like STEP, but executes exp2 (default 6) instructions

## **VOLUME [drive]**

Set the internal name of a disk displayed if DIR or FDIR is run in EXDOS. Does not change the file name of the floppy disk file, which is used in the FLOPPY command. It is therefore recommended to set this name to the name of the disk file.

## **/exp1**

open memory address at exp1

    /exp2    open new address exp2

    <return> open next address

    L        open previous address

    'C        set memory cell to Ascii code



exp2    set memory cell to exp2

## 6. PASCAL commands

The PASCAL environment is started with the command

RUN PASCAL

The Pascal system tries first to read the command from drive 1, if that fails from drive 0. Drive 0 is the drive where most of the programs should be (normally on the disk PASCAL. Commands, which are under ongoing work, should be on drive 1 (normally on disk WORK).

In the PASCAL environment, the following commands are available, if they can be found on disk 0. In their PASCAL version, most commands have drive 1 as the default drive for their argument in order to facilitate Pascal development on the WORK disk on drive 1. The source of the Pascal programs should be on the disk SOURCEPASCAL.

### ARGLIST arguments

Displays the list of arguments as they are forwarded to Pascal programs.

### BOUNCE

Bouncing ball on graphics display.

### CALC

A small calculator. Makes basic calculations and displays result in decimal, hexadecimal and binary, when possible.

### CLEAN [drive]

Automatically deletes unnecessary files on the disk drive (default 0).

- All files with the type :Q (compiler temporary files)
- All but the latest cyclus of any file

The command CLEAN does not PACK the disk, in case you decide that you need to REVIVE a file which has been deleted by CLEAN. Use PACK after CLEAN if you want to recover the space.

### COMPILE filnam[.cy][,drive] [options]

To compile a Pascal source file (:P). Give the file name without the :P. Options can be several letters stringed together. The Pascal compiler is a 2-pass compiler. The second pass is only executed if the first one is successful.

- |   |  |
|---|--|
| L | Print hard copy (into printout.txt)                          |
| R | Compile with runtime index checking (slower and larger file) |
| N | Do not produce a loader file for pass 2                      |

Use COMPILE1 instead of COMPILE to compile libraries. Libraries do not need the second pass COMPILE2, and an error message will be produced when COMPILE2 wants to load the loader file :Q of a library. The default drive is drive 1.

### COMPILE 1 filnam[.cy][,drive] [options]

Execute only pass 1 of the compiler. The default drive 1 drive 1.

### COMPILE 2 filnam[.cy][,drive]

Execute only pass 2 of the compiler. Libraries do not need the second pass of the compiler and cannot be loaded with COMPILE2. The default drive 1 drive 1.

### COPY wfilnam[.cy][,sourcedrive] desdrive

Copy a file from thesource drive to the destination. Copies all types of sequential files and Pascal binary runtime files. Cannot be used for machine language programs. The length of sequential files is only

limited by available floppy disk space.

## **DELETE wfilnam[.cy][,drive]**

Delete a file. The wild card letter @ can is not available in Pascal. Use the command CLEAN instead to delete multiple unwanted files. Drive 1 is the default drive in the Pascal version of the command.

## **DIR [drive]**

Display directory of a disk drive. The entries are displayed in several columns, depending on the longest name and the number of files on the disk. The free space and the space occupied by deleted files is displayed in decimal. The default drive is remains drive 0 as in EXDOS.

## **EDIT filnam[.cy][,drive]**

Edit a file with the Linux mousepad editor. Use the command NEW (see below) if you want to edit a new file. The default drive is drive 1, and the command automatically adds :P to filnam, if no other type is given. Use pedit instead of edit to use the internal Pascal editor.

## **ERROR errorcode**

Display Pascal error code as text.

## **EXTIME pname,drive arguments**

Measure and diplay the execution time of a Pascal program pname, called with its arguments.

## **FIND filnam**

Find files on drive 0 and 1. Wildcards \* and ? are allowed, except as the first character.  
Examples: find name\*, find name:?, find name:p

## **FLOPPY fname[,drive]**

Change the floppy disk in drive (default drive 0) to the one with the name fname. Floppy disks are stored in the Linux Disks folder with the extension .disk, but fname must be given without this extension. Additional floppies can be made there by using Linux to make copies of existing disks. The emulator can handle an unlimited number of floppies, but only 2 floppies can be in the 2 drives at the same time.

## **GETSOURCE pname**

Copy a source file from SOURCEPASCAL or SOURCECOMPIL to WORK.

## **GRAPH**

Example of displaying a graph of a table of a real numbers (TABLE.X on disk 1). Create first the table with MAKETABLE. The example demonstrates how to read from a binary random access file.

## **GRTEST**

Display a test pattern on the R65 graphics window.

## **GRTEST2**

Create a test plot on the R65 graphics display. The same plot is available as TEKTEST on an attached tek4010 emulator (see chapter 9).

## **GR3D**

Create a 3D plot on the R65 graphics display. The same plot is available as TEK3D on an attached tek4010 emulator (see chapter 9).

## **LEDTEST**

Create test patterns on the 7-segment display of the R65 replica, or on the top line of the R65 window.

## **MAKETABLE**

Example of creating a binary random access file holding a table of real numbers. The file is created on disk1 and is called TABLE:X. The example created a fading sine wave. A graph of the file can be displayed

with GRAPH.

### **NEW filnam[.cy][,drive]**

Create a new empty sequential file. The Pascal version of the command has drive 1 as the default drive, and sets the file type automatically to :P.

### **PACK [drive]**

Permanently erase all deleted files and recover the disk space. DIR displays how much space can be recovered with the PACK command, and CLEAN automatically deletes certain files (see details in the description of the command). The default drive is remains drive 0 as in EXDOS.

### **PEDIT filnam[.cy][,drive]**

Edit a file with the internal Pascal editor. Use the command NEW (see below) if you want to edit a new file. The default drive is drive 1, and the command automatically adds :P to filnam, if no other type is given. Use edit instead of pedit to use the linux mousepad editor.

### **PUTSOURCE pname**

Move a source file from WORK to SOURCEPASCAL or SOURCECOMPIL.

### **PUTOBJECT pname**

Move a Pacal object file from WORK to PASCAL.

### **SINETST**

Create a sine and cosine graph on the R65 graphics screen.

### **TEKTEST**

Create a test plot on an attached tek4010 emulator (see chapter 9). The same plot is available as GRTEST2 for the R65 graphics display.

### **TEK3D**

Create a 3D plot on an attached tek4010 emulator (see chapter 9). The same plot is available as GR3D for the R65 graphics display.

### **TGRAPH**

Example of displaying a graph of a table of a real numbers (TABLE.X on disk 1) on an attached Tektronix 4010 terminal. Create first the table with MAKETABLE.

### **THROWSIM**

Display a flight path of a thrown object on the R65 graphics screen.

## **7. Pascal libraries**

Look at the source of these libraries on SOURCEPASCAL for details

### **ARGLIB**

Handles arguments given by the pascal system to a program

### **LEDLIB**

Handles output to the 7-segment display

### **MATHLIB**

Handles math functions and the output of floating point numbers

### **PLOTLIB**

Plot library for the R65 graphics screen

## **RALIB**

Random access (binary) file access library

## **SYSLIB**

Main R65 Pascal system library

## **TEKLIB**

Plot library for an attach tek4010 plotter

## **TIMELIB**

Handles system and execution time

## **8. Pascal games**

### **ALIEN**

Save the earth from an alien invasion.

### **PONG**

Play the game of Pong.

### **REVERSI**

Play the game of Reversi. The rules can be found at <https://en.wikipedia.org/wiki/Reversi>. Please note that the first two moves need to be placed in the center.

## **9. R65 Graphic Basic**

The R65 Graphic Basic includes the basic commands found in Basic Interpreters of that time. The following commands have been added in the R65 version or are working slightly different:

### **CLOSE [D1]**

Close a sequential file (if D1 is given) or all open sequential files

### **DIR [D]**

Display the directory of floppy disk drive D (default 0).

### **FILES**

Displays information about all currently open sequential files

### **GET [#D1:] V1 [,V2....]**

Read one character from device D1. If V1 is a string variable, one character is read, otherwise one digit of a number. The following devices are supported:

- D1=0: Read from keyboard, wait until a key is ready
- D1=1: Read from keyboards, return with empty string if no character available
- D1=2: Not implemented in Emulator
- D1=3: Read from keyboard and display immediately on Graphics Display
- D1>=4: Read from a open disk file

### **INPUT [#D1] V1 [,V2...]**

Read from device D1 (see GET for details). Reads one full string or one number)

### **LOAD S1[.D1] [,D2]**

Load a Basic program from floppy drive D2. Example LOAD "LIFE:B"

**LOAD #D1;**

Import a basic program from a text file which has previously been opened with OPEN D1

**MOVE N1,N2**

Move graphic cursor to n1,n2

**OPEN D1,S1[.D2][,D3,[D4]]**

Open a sequential file (D1>=4). D3 is the floppy drive number, D4=0 for read and 1 for write.

**PLOT N1,N2[,D1]**

Plot a dot on the graphics display at N1,N2. D1=0 in white, D1=1 in black, D1=2 inverted.

**PLOT NEW**

Initialize graphics display, Switch alphanumeric display buffer to 16 lines

**PLOT CLR**

Clear graphics display

**PLOT STOP**

Go back to alphanumeric display without losing graphics display buffer

**PLOT CONT**

Go back to graphics display after a PLOT STOP

**PLOT END**

Erase graphics display area and go back to alphanumeric display. Switch back alphanumeric display buffer to 42 lines.

**PRINT [D1;] ....**

Basic print statement. Devices are:

D1=0: Standard display

D1=1: Not implemented in emulator

D1=2: Not implemented in emulator

D1=3: Graphics display at current position of graphic cursor

D1>=4: Open sequential file

**STORE S1[.D1] [,D2]**

Store Basic program on drive D2.

**SYS N1**

Execute 6502 code at N1

**WAIT N1**

Wait N1 times 10 msec. The emulator process is suspended in intervals of 10 msec to avoid overheating of the Raspberry Pi CPU as during character input.

**10. Editing files**

Editing on a system without a mouse is quite cumbersome by modern standards. Therefore the adaption for the virtual R65 system includes the EDIT command in exdos and Pascal to use the external mousepad editor.

There is also an internal, Pascal based editor available by using the PEDIT command. It is limited to 500 lines and slow for certain operations. Nevertheless it is quite usefull.

The following escape sequences are available in PEDIT:

- t        Move to top of text. Available also with the HOME key where available.
- b        Move to bottom of text. Available also with the END key where available.
- ln       Put the cursor on line *n* and scroll if necessary.
- f        Find text. The string to find is entered after typing return. Entering an empty string clears all marked fields.
- g        Find again using the previous string.
- cn       Copy marks *n* lines for copying. The copying will be done when the paste command is executed. It is therefore possible to modify the marked lines before the paste command is executed.
- p        Paste the marked (copied) lines to the cursor line. This duplicates the lines.
- M        Move the marked (copied) lines to the cursor line. The original is deleted.
- dn       Delete *n* lines
- w        Write the file to disk. The next cyclus number is used. It is therefore possible to save write multiple times.
- q        write file to disk and quit.
- k        kill editor without writing to disk.

## 11. Floppy disk drives (tapes are not emulated)

Drive 0 disk drive A

drive 1        disk drive B

Floppy disks can be changed with the EXDOS command:

FLOPPY disknam[,drive].

The recommended setup is to use PROGRAMS, PASCAL or SOURCE in drive 0, and WORK in drive 1. The defaults are set for this setup. The disks PROGRAMS, PASCAL and SOURCE are distribution disks. They are replaced with their standard version during a software upgrade. Use WORK for your own work and make backup copies on another disk before upgrading the software. The latest version of the R65 emulator supports disks with 256 file entries and 160 tracks of 16 sectors, which results in a capacity of 2560 sectors.

## 12. Attaching a printer/plotter

The R65 emulator emulates a printer by printing to the Linux file "printout.txt" in the R65 directory. This file is cleared or created each time the emulator starts. The text printed is formatted for Linux, and page headers are added. It is possible to print raw R65 text (without any formatting) by switching the printer driver to raw mode:

Device control 1        hex 11 switch to raw mode

Device control 2        hex 12 switch back to normal formatted mode

If you want to see the printed text while it is generated, open a shell window, go to your R65 directory and type

tail -f printout.txt

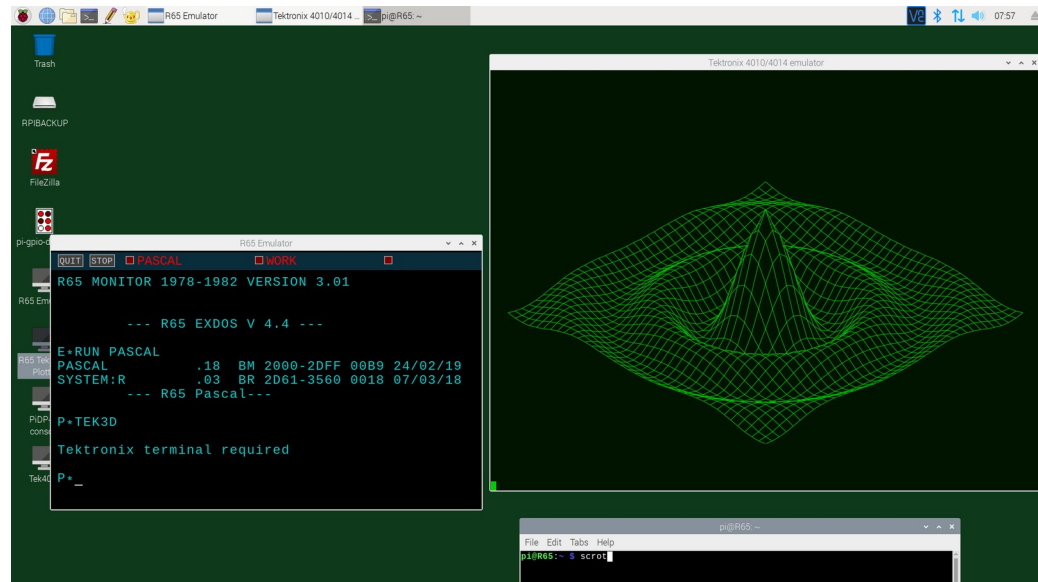
It is also possible to use my Tektronix 4010 emulator as a plotter window. Download tek4010 from <https://github.com/rricharz/Tek4010>. After having tested your installation, put a copy of the executable "tek4010" in your R65 directory.

Now start R65 and afterwards tek4010 by executing the command

```
./tek4010 -autoClear tail -f printout.txt
```

You can now see any printed text in the tek4010 window.

With R65 you can in the window. "teklib" on the disk and is on the



PASCAL, also plot tek4010 The library is PASCAL its source

SOURCECOMPIL disk. There are also examples on the SOURCEPASCAL and PASCAL disks: "tektest", "tek3D" and "tgraph". Please note that R65 floating point calculations using the 8-bit R65 Pascal are rather slow as compared to modern standards, but the examples show that it was possible to produce high resolution graphics with a Tektronix 4010 and a 6502 8-bit computer in the late 1970s.

## 13. File Type

"Block" files (binary files)

- M Machine language program
- R :R Pascal binary runtime file
- X :X Other binary file

Sequential files (text files)

- A :A Assembler source file
- P :P Pascal program or library source file
- Q :Q Pascal loader program file
- T :T Pascal loader library file
- L :L Pascal loader ident table
- W :W Pascal compiler reserved words table
- B Other text file

## 14. Error Codes

### System error codes

- 01 READ/WRITE ERROR
- 02 CHECKSUM ERROR
- 03 ESCAPE EXIT DURING READ/WRITE

04 RECORD NUMBER ERROR  
05 FILE TYPE ERROR  
06 FILE NOT FOUND  
07 DISK NOT READY  
08 DIRECTORY FULL, FILE NOT STORED  
09 ILLEGAL IRQ  
10 EXPRESSION MISSING  
11 MEMORY CELL CANNOT BE CHANGED  
12 BREAK TABLE FULL, NOT INSERTED  
13 ILLEGAL MEMORY CELL FOR BREAK  
14 DOUBLE BRAK POINT SETTING  
15 END OF LINE EXPECTED  
16 SYNTAX WRONG IN REGISTER NAME OR =  
17 BREAKPOINT NOT FOUND IN TABLE  
18 SYNTAX FRONG IN STORE  
19 FILE SUBTYPE WRONG OR MISSING  
20 WRONG FILE TYPE NOT RUN  
21 UNKNOWN MONITOR COMMAND  
22 ILLEGAL OPCODE FOR STEP/TRACE  
23 TOO MANY OPEN FILES, NOT OPENED  
24 DIRECTION ERROR IN SEQUENTIAL R/W  
25 WRONG FILE NUMBER, FILE NOT OPEN  
26 DISK FULL, FILE NOT STORED  
27 RANDOM ACCESS INDEX OUT OF RANGE  
28 ILLEGAL DRIVE FOR RANDOM ACCESS  
29 FILE NOT OPEN FOR RANDOM ACCESS WRITE

### **ASSEMBLER error codes**

31 CLOSING ) EXPECTED IN EXPRESSION  
32 SYNTAX ERROR IN LABEL  
33 HEX CHAR EXPECTED AFTER \$  
34 LABEL TABLE OVERFLOW  
35 LOGICAL CHAR EXPECTED AFTER #  
36 EXPRESSION NOT RESOLVED (PASS 2)  
37 SYNTAX ERROR IN OPCODE  
38 MNEMONIC OR ADDRESSING ILLEGAL  
39 ILLEGAL ADDRESSING MODE  
40 SYNTAX ERROR IN OPERAND  
41 ABSOLUTE ADDRESS ILLEGAL



- 42 MORE THAN 1 UNRESOLVED LABEL IN FORWARD BRANCH
- 43 BRANCH EXCEEDS BOUNDS
- 44 FORWARD BRANCH TO THIS LABEL EXCEEDS BOUNDS
- 45 DOUBLE LABELDEFINITION
- 46 MISSMATCH IN SECOND PASS
- 47 LABEL MISSING IN EQU
- 48 OPERAND OF BYT TOO LONG
- 49 EXPRESSION MUST BE RESOLVED
- 50 LINE TOO LONG
- 51 CHAR FOLLOWS LOGICAL END OF OPERAND
- 52 TOO MANY UNRESOLVED BRANCHES (NOT INSERTED INTO TEST TABLE)

**EXDOS error codes**

- 61: WILD CARD NOT ALLOWED
- 62: ONLY FOR DISK, NOT TAPE
- 63: ILLEGAL COPY
- 64: FILE TOO LARGE
- 65: WRITE ERROR
- 66: IMPORT ERROR
- 67: UNKNOWN EMU COMMAND
- 68: UNABLE TO RUN MOUSEPAD

**PASCAL error codes**

- 81: DIVISION BY ZERO
- 82: STACK OVERFLOW
- 83: INDEX OUT OF BOUNDS
- 84: PROGRAM NOT FOUND, OR NOT PASCAL PROGRAM
- 85: ILLEGAL P-CODE
- 86: ESCAPE DURING EXECUTION
- 87: NO LOADER FILE MADE BY COMPILER
- 88: HEAP OVERFLOW

**15. R65 Control keys**

See also capture 2: STOP, QUIT and SHUTDOWN

**Video control functions**

	CTRL	VCOD	ASCII
CURSOR DOWN	CDOWN	E9	X 18
CURSOR RIGHT	CRIGHT	E8	V 16
CURSOR LEFT	CLEFT	E7	C 03
CURSOR UP	CUP	E6	Z 1A

CURSOR HOME	CHOME	E5 A 01
INSERT CHAR	INSCHR	E4 U 15 CTRL-RIGHTARROW
DELETE CHAR	DELCHR	E3 Y 19 CTRL-LEFTARROW
CLEAR LINE	CLRLIN	E2 E 17
CLEAR TO END OF SCREEN	CLRDSP	E1 Q 11 CTRL-RETURN
INSERT LINE	INSLIN	C4 D 04
DELETE LINE	DELLIN	C3 F 06
TOGGLE ALPHA/GRAPHICS	TALGRA	E9 L 0C
ESCAPE	ESCAPE	91 1B
SET TABULATOR	SETTAB	8B
ROLL DOWN	RDOWN	89 B 02 CTRL-DOWNARROW
TO RIGHT MARGIN	CRMARG	88
TO LEFT MARGIN	CLMARG	87
ROLL UP	RUP	86 H 08 CTRL-UPARROW
REVERSE HOME	RVHOME	85 P 10
RESUME WRITING	RWRITE	84
QUIET WRITING	QWRITE	83
TOGGLE BLACK/WHITE	TBLWHI	82 E 05
CLEAR GRAPHIC DISPLAY	CLRGRA	81

### Other control functions

PRINT ALL ON	PRTON	R 12
PRINT ALL OFF	PRTOFF	T 14
DISPLAY CONTROL CHAR	DSPCC	S 13
CLEAR TABULATOR	CLRTAB	O 0F
INVERSE VIDEO	INVVIDN	0E
NORMAL VIDEO	NORVID	K 0B
CARRIAGE RETURN	EXCR	0D
LINE FEED	EXLF	0A
TABULATOR (8)	TAB	I 09
BELL	BELL	G 07
PAD CHARACTER	PADCHR	@ 00

## 16. “Burning” new PROMS/EPROMS

The System software on the original system was burned in EPROMS and used the KIM-1 ROM. The corresponding Assembler source files are:

- KIM1.asm
- CRT.asm
- DISK.asm
- IOCONTROL.asm

- MONITOR.asm
- EXDOS.asm

With the exception of EXDOS, the corresponding object code was read only in the original system, and the R65 Emulator does not allow to write into these memory areas. Nevertheless, it is quite easy to patch these programs (in fact much easier as it was to burn new EPROMS on the original system).

If you just want to make a simple patch (just change a few locations, but not move anything!), or look at the programs, go to RASPIAN and there to R65/R65-Emulator/Assembler. The source files are there with the extension .asm, and the Assembler listings with identical line numbers with the extension txt. The emulator just reads the the object code from the .txt files at startup. This sounds crasy at first, but it is very fast and eliminates the additional step of making binary files. You can therefore just patch those files by editing them carefully. Write down somewhere what you have patched in case you later want to upgrade the software, and make a backup copy before starting to patch, so that you can go back if things don't work anymore.

## **17. The usual disclaimer**

The contributions in this repository are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Please report any problems or suggestions for improvements to [r77@bluewin.ch](mailto:r77@bluewin.ch)