MEGA65

USER'S GUIDE

MUSEUM OF ELECTRONIC GAMES & ART

REGULATORY INFORMATION

The MEGA65 home computer and portable computer have not been subject to FCC, EC or other regulatory approvals as of the time of writing.

MEGA65 USER'S GUIDE

Published by
the Museum of Electronic Games and Art (M.E.G.A.) eV., Germany.
and
Flinders University, Australia.

WORK IN PROGRESS

Copyright ©2018 by Paul Gardner-Stephen, Flinders University, the Museum of Electronic Games and Art eV., and contributors.

This user-guide is made available under the GNU Free Documentation License v1.3, or later, if desired. This means that you are free to modify, reproduce and redistribute this user guide, subject to certain conditions. The full text of the GNU Free Documentation License v1.3 can be found at `https://www.gnu.org/licenses/fdl-1.3.en.html`.

Implicit in this copyright license, is the permission to duplicate and/or redistribute this document in whole or in part for use in education environments. We want to support the education of future generations, so if you have any worries or concerns, please contact us.

March 9, 2019

Contents

CHAPTER 1

Introduction

Congratulations on your purchase of one of the most long awaited computers in the history of computing! The **MEGA65** is a community designed computer, based on the never-released Commodore™ 65[1] computer, that was first designed in 1989, and intended for public release in 1990. Twenty eight years have passed since then, but the **simple**, **friendly** nature of the 1980s home computers is still something that has not been recreated. These were computers that were simple enough you could understand not just how your computer worked, but how computers themselves work.

Many of the people who grew up using the home computers of the 1980s have grown up to have exciting and rewarding jobs in many companies, in part because of what they learnt about computers in the comfort of their own home. We want to give you that same opportunity, so that you can experience the joy of learning how to use computers to solve all sorts of problems, whether that be writing a letter to a friend, working out how much tax you owe, inventing new things, or working out how the universe works. This is why we made the **MEGA65**.

The **MEGA65** team thinks that owning a computer should be like owning a house: You don't just use a house, you change things big and small to really make it your own, and maybe even renovate it or add on a room or two. In this guide we will show you how to more than just hang your own pictures on the wall, but instead how you can dream up new ways of using the powerful capabilities of computers by coding your own computer programmes, and even changing the computer itself!

To help you have fun with your **MEGA65**, we will show you how to use the exciting **graphics** and **sound** capabilities of the **MEGA65**. But the **MEGA65** isn't just about writing your own programmes. It can also run many of the thousands of games and other programmes that were created for the Commodore 64™[2] computer.

Welcome to the world of the **MEGA65**!

[1] Commodore is a trademark of C= Holdings
[2] Commodore 64 is a trademark of C= Holdings,

PART I

GETTING TO KNOW YOUR MEGA65

CHAPTER 2

SETUP

- Unpacking and connecting the MEGA65
- Installation
- Optional Connections
- Operation
- Adjusting the Video Display

UNPACKING AND CONNECTING THE MEGA65

The following instructions will show you all of the steps required to setup your MEGA65 home computer. The MEGA65 computer comes with the following:

- MEGA65 computer
- Power supply (black box with socket for mains supply)
- This book, the MEGA65 User's Guide

In addition, you will need the following things to be able to use your MEGA65 computer:

- A television set or computer monitor with a VGA input, that is capable of displaying an image with 800x600 pixel resolution at 60Hz, and preferably also at 50Hz.
- A VGA video cable.

You may also wish to use the following to get the most out of your MEGA65:

- 3.5" audio cable and suitable speakers or hifi system, so that you can enjoy the sound capabilities of your MEGA65.
- RJ45 ethernet cable ("normal network cable") and a network router or switch that it can connect to, so that you can use the high-speed networking capabilities of your MEGA65.

INSTALLATION

OPTIONAL CONNECTIONS

OPERATION

ADJUSTING THE VIDEO DISPLAY

CHAPTER 3

GETTING STARTED

- Keyboard
- that

KEYBOARD

Now that you have everything connected, it is time to get familiar with the Mega65 keyboard.

You will notice that the keyboard is a little different from the modern standard used on computers today. While most keys will be in familar positions, there are some specialised keys with special graphic symbols marked on the front.

What follows is a brief description of how some of these special keys function.

RETURN

The **RETURN** key tells the computer to process the information you typed.

SHIFT

The **SHIFT** key works just like on a typewriter or modern keyboard. It allows you to type uppercase letters when in lowercase mode, or to gain access to special graphic characters that are displayed on the right hand side of the front part of the key.

In the case of the function keys, pressing **SHIFT** will give you the function that is marked at the front of the key.

← ↑ → ↓ Cursor Keys

These keys move the cursor around the screen... etc etc.

RUN/STOP

Normally, pressing the **RUN/STOP** key will stop execution of a program. When holding **SHIFT** while pressing **RUN/STOP** will load the first program from disk.

RESTORE

etc etc.

THAT

PART II

FIRST STEPS IN CODING

CHAPTER 4

How Computers Work

- Computers are just a pile of switches

Computers can do amazing things, and you can make them do amazing things for you, too. But to do that, you need to understand how computers work. This can be hard to find out these days, because computers are now so complicated, that it isn't obvious how computers work anymore, just by looking at them and using them. The MEGA65 is designed to be simple enough that you can learn how computers work as you use it. But we don't want to leave you to have to work out everything for yourself. That is why we have written this chapter, so that you can learn how computers work, and then use that knowledge to help you make computers do what *you* want them to do.

COMPUTERS ARE JUST A PILE OF SWITCHES

What are computers *really*? Well, the answer to that question is quite simple, if a little surprising: Computers really are just made of lots of switches. These switches work very similar to the switches you use to turn a light on or off. Light switches connect or disconnect the power supply to a light. The switches in computers connect or disconnect circuits in the computer to power. But computers can do a lot more than just turn on and off, so something else must be going on. That something else is also a bit of a surprise: A computer can turn its own switches on and off by itself. Let's explore how this simple idea of switches that can turn themselves on and off makes a computer.

As we have just heard, computers are full of switches. Obviously those switches must be tiny, and they are. When you hear people talking about microns and nanometres with regard to computers, they are often talking about the size of the little switches that make up the computer chips. These switches are called transistors. The switches in the main chip of the MEGA65, for example, are 28 nanometres long. That's about 100,000 times skinnier than a single strand of hair. This is good, because a computer might need millions or billions of switches in its design.

PART III

SOUND AND GRAPHICS

APPENDICES

APPENDIX A

ACCESSORIES

APPENDIX B

BASIC 10 Command Reference

- **Format of Commands, Functions and Operators**
- **Commands**

- **Functions**

- **Operators**

AND – Logical AND operator31

FORMAT OF COMMANDS, FUNCTIONS AND OPERATORS

In this appendix each of the commands, functions and other callable elements of BASIC 10 are described. Some of these can take one or more arguments, that is, pieces of input that you can (or sometimes must) provide as part of the command or function call. Some also require that you use special keywords. Here is an example of how commands, functions and operators will be described in this appendix:

KEY <numeric expression>,<string expression>

In this case, KEY is what we call a **keyword**. That just means a special word that BASIC understands. Keywords are always written in CAPITALS, so that you can easily recognise them.

The **<** and **>** signs mean that whatever is between them must be there for the command, function or operator to work. In this case, it tells us that we need to have a **numeric expression** in one place, and a **string expression** in another place. We'll explain what there are a bit more in a few moments.

You might also see square brackets around something, for example, **[,numeric expression]**. This means that whatever appears between the square brackets is optional, that is, you can include it if you need to, but that the command, function or operator will work just fine without it. For example, the `CIRCLE` command has an optional numeric argument to indicate if the circle should be filled when being drawn.

The comma, and some other symbols and punctuation marks just represent themselves. In this case, it means that there must be a comma between the **numeric expression** and the **string expression**. This is what we call syntax: If you miss something out, or put the wrong thing in the wrong place, it is called a syntax error, and the computer will tell you if you have a syntax error by giving a `?SYNTAX ERROR` message.

There is nothing to worry about getting an error from the computer. Instead, it is just the computer's way of telling you that something isn't quite right, so that you can more easily find and fix the problem. Error messages like this can't hurt the computer or damage your program, so there is nothing to worry about. For example, if we accidentally left the

comma out, or replaced it with a full-stop, the computer will respond with a syntax error, like this:

```
KEY 8"FISH"

?SYNTAX ERROR

KEY 8."FISH"

?SYNTAX ERROR
```

It is very common for commands, functions and operators to use one or more **"expression"**. An expression is just a fancy name for something that has a value. This could be a string, such as `"HELLO"`, or a number, like `23.7`, or it could be a calculation, that might include one or more functions or operators, such as `LEN("HELLO") * (3 XOR 7)`. Generally speaking, expressions can result in either a string or numeric result. In this case we call the expressions either string expressions or numeric expressions. For example, `"HELLO"` is a **string expression**, while `23.7` is a **numeric expression**.

It is important to use the correct type of expression when writing your programs. If you accidentally use the wrong type, the computer will give you a `?TYPE MISMATCH ERROR`, to say that the type of expression you gave doesn't match what it expected, that is, there is a mismatch between the type of expression it expected, and the one you gave. For example, we will get a `?TYPE MISMATCH ERROR` if we type the following command, because `"POTATO"` is a string expression instead of a numeric expression:

```
KEY "POTATO","SOUP"
```

You can try typing this into the computer yourself now, if you like.

COMMANDS

Commands are statements that you can use directly from the **READY.** prompt, or from within a program, for example:

```
PRINT \"HELLO\"
HELLO

10 PRINT \"HELLO\"
```

```
RUN
HELLO
```

APPEND - Open disk file for append

APPEND logical_file_number, "filename" [,Ddrive] [<ON|,>Udevice]

logical_file_number, **drive** and **device** are all **numeric expressions**. **filename** is a **string expression** that indicates the name of the file to append to.

APPEND functions similar to the OPEN command, except that if the file already exists, the existing content of the file will be retained, and any PRINT commands made to the open file will cause the file to grow longer.

FUNCTIONS

Functions are different from commands in that they cannot be used on their own. Instead, a function can only be used in an expression. There are a variety of functions provided in BASIC 10. Some can be used only on strings, while others can only be used on numbers.

ABS - Absolute value function

ABS(expression)

This function returns the absolute value of the numeric expression. That is, it ignores the sign. That is, if the value of the expression is positive, the result will be the same, but if the value is negative, then the result will be the positive number with the same magnitude, as the following examples show.

```
PRINT ABS(123)
123
PRINT ABS(-123)
123
PRINT ABS(4.5)
4.5
PRINT ABS(-4.5)
4.5
```

```
PRINT ABS(7-5)
2
PRINT ABS(5-7)
2
```

The expression must be numeric. If you give it a string, the computer will indicate a type mismatch error.

```
PRINT ABS("POTATO")

?TYPE MISMATCH ERROR

PRINT ABS("TWENTY")

?TYPE MISMATCH ERROR

PRINT ABS("12")

?TYPE MISMATCH ERROR
```

ASC - PETSCII Value Function

ASC(<string expression>)

This function returns the PETSCII value of the first character of the string expression. If the string expression is blank, it returns 0. For example:

```
PRINT ASC(\"FOO\")
70
A$=\"MOO\": PRINT ASC(A$)
77
PRINT ASC("")
0
```

OPERATORS

Operators are very much like functions, except that instead of having their arguments inside round brackets, the keyword goes in between the arguments, or in the case of the few operators, like **NOT** , that take only one argument, the keyword goes before the argument like in a function, but there are no round brackets.

AND - Logical AND operator

<numeric expression> AND <numeric expression>

The AND operator calculates the logical and of the two arguments. The calculation is performed after converting the arguments to 16-bit binary values. If either expression is outside of the range of -32768 to +32767, then an **?ILLEGAL QUANTITY ERROR** will be generated.

```
PRINT 1 AND 3
1
PRINT 15 AND 23
7
PRINT 128 AND 64
0
PRINT -1 AND -7
-7
PRINT 2348734 AND 7

?ILLEGAL QUANTITY ERROR
```

The AND operator can also be used to test boolean conditions, because the **=**, **<**, **<=**, **<>**, **>=** and **>** inequality operators return a numeric value, as can be seen by printing the result of such an expression:

```
PRINT \"ABC\"=\"ABC\"
-1
PRINT \"ABC\"=\"DEF\"
0
PRINT 3 < 9
-1
```

Thus, constructions like the following can be constructed:

```
PRINT (\"ABC\" < \"FOO\") AND (7 <> 9)
-1
PRINT (\"ABC\" = \"FOO\") AND (23 > 8)
0
```

Bibliography

INDEX

copyright, ii

PART IV

ELEMENT CATALOGUE

KEYBOARD KEYS

RUN/STOP

Text to the left RUN/STOP and text to the right.

SHIFT CTRL 9 CLR/HOME ■ RETURN

INST/DEL * RUN/STOP ← ↑ → ↓

SCREEN OUTPUT

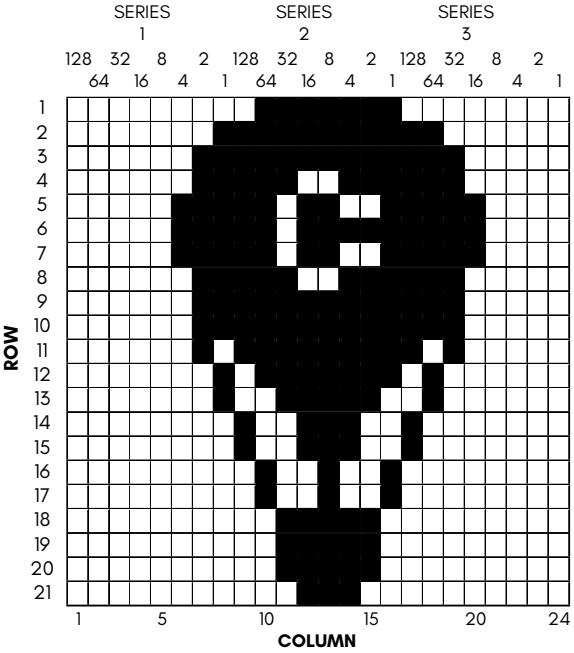
```
    10 INPUT A$
    20 PRINT "YOU TYPED: ";A$
    30 PRINT
    40 GOTO 10
    RUN
    ? MEGA 65
    YOU TYPED: MEGA 65
```

```
10 OPEN 1,8,0,"$0:*,P,R
20 : IF DS THEN PRINT DS$: GOTO 100
30 GET#1,X$,X$
40 DO
50 : GET#1,X$,X$: IF ST THEN EXIT
60 : GET#1,BL$,BH$
70 : LINE INPUT#1, F$
80 : PRINT LEFT$(F$,18)
90 LOOP
100 CLOSE 1

RUN
```

SPRITE GRIDS

Original Commodore Balloon Demo



Wizball Demo Sprite