DUO Travel User Guide

By Jack Eisenmann

Contact address: esperantanaso@gmail.com

Table of Contents

Introduction

Congratulations on acquiring your very own DUO Travel! The DUO Travel is a portable 8-bit computer based on the ATMega328. The DUO Travel is designed to be easy to program without any external hardware. Just use the built-in character display and keypad! The computer sports 2 KB of RAM and 32 KB of storage for programs. The DUO Travel is pre-installed with an interpreter for a proprietary programming language called DUO Travel Code. This language features named variables, heap allocation, and custom function definition. Let's dive in!

Getting Started

The DUO Travel operates on a single 9-volt battery. Insert a battery into the battery holder, then toggle the power switch on. You will be welcomed by the following screen:

```
DUO Travel OS
Version 1.0
```

Press any key to enter the file list:

```
File List
*Create file
```

By default the DUO Travel does not come with any programs pre-installed. Let's try making a "Hello world" program. We need to create a file to contain the program. To select the `Create file` option, press the "SELECT" button. The computer will tell you: `Enter file name.` Press "SELECT" again to dismiss the message. Now you will be presented with an interface for entering a name:

```
      _
Lett A
```

The underscore in the middle of the top line is the cursor. The bottom line shows which symbol you have selected (`A`). It also shows which symbol group you are using (`Lett` for letters). To scroll through symbols in a group, press the left and right "SYMBOL" buttons. To scroll though different groups, press the up and down "SYMBOL" buttons. When you have chosen the symbol you want to type, press the "SELECT" button. To delete the symbol before the cursor, press the "DELETE" button. When you finish entering the name of your program, press the "FINISH" button. The computer will tell you: `Created file.` Press "SELECT" to dismiss the message and return to the file list.

Now we want to edit the contents of the file. Press the up and down "CURSOR" buttons to find your file in the list, then press "SELECT" (use these controls for any menu presented with an asterisk). You will be presented with a menu which looks like this, but probably with a different file name:

```
MYFILE
*Run
```

We don't want to run the program just yet (because it is empty). To enter the file editor, select the `Edit` option in the menu (using the "CURSOR" and "SELECT" buttons).

You will now see the same interface as when we entered the file name. The controls here are the same. To print `HELLO WORLD`, we want to enter the following code:

```
print:"HELLO WORLD"
```

⚠ It is important to note that `print:` is a single symbol. You can find it in the `I/O` (input/output) symbol group. The quotation mark and space symbols are in the `Punc` (punctuation) group.

When you have finished entering the code, press the "ESCAPE" button. Here you can select the `Save` option to save your work. When you want to exit, select the `Quit` option from the same menu (use the "CURSOR" buttons to scroll through the menu options).

After you save and quit, you are finally ready to run your program! In the menu for your program's file, select the `Run` option. You should see this text on the screen:

```
HELLO WORLD
```

Press the "SELECT" button to dismiss the text. If you see an `ERROR` message, open the file editor again and try to correct the contents.

Other Interface Features

In addition to editing and running a file, you can also rename or delete files by using the `Rename` and `Delete` options. The computer will give you a confirmation menu before deleting a file.

The file editor allows you to enter multiple lines of code. To insert a newline under the cursor, press the "NEWLINE" button. To scroll between lines, press the up and down "CURSOR" buttons. To delete a newline, scroll to the beginning of a line and press the "DELETE" button.

Use the `Get size` option in the file menu and text editor to inspect the size of the selected file.

To delete the contents of the selected line, press the "CLEAR" button.

Press and hold any button to repeat the button's action quickly.

DUO Travel Code Overview

DUO Travel code consists of a sequence of expressions separated by newlines. An expression may be one of the following:

- Value literal (Ex: `123`)
- Variable name (Ex: `MYVAR`)

- Operation (Ex: `1+2`)
- Function invocation (Ex: `print:"HELLO WORLD"`)

There are 4 types of values in DUO Travel Code:

- Number (Ex: `8.5`) – Single precision float
- String (Ex: `"HELLO WORLD"`) – Mutable with dynamic length
- List (Ex: `[10,20,30]`) – Dynamic length with mixed member types
- Function – Defined by the user

Number values support scientific notation using the `e` symbol (Ex: `1.234e9`).

Use apostrophes to denote a character literal (Ex: `'A'`). A character outside of a string has the "Number" type.

Numbers and functions are stored in the stack. Strings and lists are stored in the heap. If you don't know what the stack or heap are, you don't need to worry about this information.

To create a new variable, use the assignment operator:

```
MYVAR=123
```

Variable names may contain letters, numbers, and underscores, but may not start with a number. You can save time by using single-letter variable names.

Quick Reference

Please see the sections after this one for explanation and examples.

Operators:

`(num)+(num)` or `(str)+(str)`
`(num)-(num)` or `-(num)`
`(num)*(num)`
`(num)/(num)`
`(num)%(num)`
`(num)&&(num)`
`(num)||(num)`
`(num)^^(num)`
`!(num)`
`(num)&(num)`
`(num)|(num)`
`(num)^(num)`
`~(num)`
`(num)<<(num)`
`(num)>>(num)`
`(num var)++` or `++(num var)`
`(num var)--` or `--(num var)`
`(num)>(num)`

`(num)<(num)`
`(num)>=(num)`
`(num)<=(num)`
`(num)==(num)` or `(str)==(str)`
`(num)!=(num)` or `(str)!=(str)`
`(str or list)[(num)]`
`(func):(val),(val),(val)...`
`(func);`

Assignment:

`(var)=(val)`
`(num var)+=(num)` or `(str var)+=(str)`
`(num var)-=(num)`
`(num var)*=(num)`
`(num var)/=(num)`
`(num var)%=(num)`
`(num var)&&=(num)`
`(num var)||=(num)`
`(num var)^^=(num)`
`(num var)&=(num)`
`(num var)|=(num)`
`(num var)^=(num)`
`(num var)<<=(num)`
`(num var)>>=(num)`

Control functions:

`if:(num condition)`
`elif:(num condition)`
`else;`
`end;`
`while:(num condition)`
`break;`
`cont;`
`func:(name),(argument),(argument),(argument)...`
`ret:(val)`
`ret;`
`quit;`

Math functions:

`rand;` (returns number)
`randInt:(num minimum),(num maximum)` (returns number)
`abs:(num)` (returns number)
`round:(num)` (returns number)
`floor:(num)` (returns number)
`ceil:(num)` (returns number)
`sin:(num)` (returns number)
`cos:(num)` (returns number)
`tan:(num)` (returns number)

`sqrt:(num)` (returns number)
`pow:(num),(num exponent)` (returns number)
`log:(num),(num base)` (returns number)

I/O functions:

`print:(num or str)`
`reqStr;` (returns string)
`reqNum;` (returns number)
`menu:(str title),(list options)` (returns number)
`fExists:(str fileName)` (returns number)
`fSize:(str fileName)` (returns number)
`fCreate:(str fileName)`
`fDelete:(str fileName)`
`fSetName:(str oldFileName),(str newFileName)`
`fRead:(str fileName),(num index),(num amount)` (returns string)
`fWrite:(str fileName),(str contents)`
`fImport:(str fileName)`

Value functions:

`num:(num or str)` (returns number)
`str:(num or str)` (returns string)
`type:(val)` (returns number)
`len:(str or list)` (returns number)
`copy:(str)` (returns string) or `copy:(list)` (returns list)
`ins:(str),(num index),(num character)` or `ins:(list),(num index),(val)`
`rem:(str or list),(num index)`
`sub:(str),(num startIndex),(num endIndex)` (returns string) or `sub:(list),(num startIndex),(num endIndex)` (returns list)
`insSub:(str),(num index),(str)` or `insSub:(list),(num index),(list)`
`remSub:(str or list),(num startIndex),(num endIndex)`
`equRef:(str),(str)` (returns number) or `equRef:(list),(list)` (returns number)

Operators

`(num)+(num)` or `(str)+(str)`: Add two numbers or concatenate two strings.
Example program:
```
print:1+2
print:"BREAD"+"BOX"
```
Output:
```
3
BREADBOX
```

`(num)-(num)` or `-(num)` Subtract two numbers or make a single number negative.
Example program:
```
print:3-1
print:-5
```

Output:
```
2
-5
```

`(num)*(num)`: Multiply.
Example program:
```
print:2*3
```
Output:
```
6
```

`(num)/(num)`: Divide.
Example program:
```
print:6/2
```
Output:
```
3
```

`(num)%(num)`: Find the remainder when dividing the first number by the second number.
Example program:
```
print:8%3
```
Output:
```
2
```

`(num)&&(num)`: Boolean AND.
Example program:
```
print:0&&0
print:1&&0
print:0&&1
print:1&&1
```
Output:
```
0
0
0
1
```

`(num)||(num)`: Boolean OR.
Example program:
```
print:0||0
print:1||0
print:0||1
print:1||1
```
Output:
```
0
1
1
1
```

`(num)^^(num)`: Boolean XOR.
Example program:
```
print:0^^0
print:1^^0
print:0^^1
```

```
print:1^^1
```
Output:
```
0
1
1
0
```

`!(num)`: Boolean NOT.
Example program:
```
print:!0
print:!1
```
Output:
```
1
0
```

`(num)&(num)`: Bitwise AND.
Example program:
```
print:3&5
```
Output:
```
1
```

`(num)|(num)`: Bitwise OR.
Example program:
```
print:3|5
```
Output:
```
7
```

`(num)^(num)`: Bitwise XOR.
Example program:
```
print:3^5
```
Output:
```
6
```

`~(num)`: Bitwise NOT.
Example program:
```
print:(~2)&7
```
Output:
```
5
```

`(num)<<(num)`: Bitshift left.
Example program:
```
print:6<<1
```
Output:
```
12
```

`(num)>>(num)`: Bitshift right.
Example program:
```
print:6>>1
```
Output:
```
3
```

`(num var)++` or `++(num var)`: Increment by one.
Example program:
```
A=5
A++
print:A
```
Output:
```
6
```

`(num var)--` or `--(num var)`: Decrement by one.
Example program:
```
A=5
A--
print:A
```
Output:
```
4
```

`(num)>(num)`: Determine whether the first number is greater than the second.
Example program:
```
print:1>2
print:2>2
print:3>2
```
Output:
```
0
0
1
```

`(num)<(num)`: Determine whether the first number is less than the second.
Example program:
```
print:1<2
print:2<2
print:3<2
```
Output:
```
1
0
0
```

`(num)>=(num)`: Determine whether the first number is greater than or equal to the second.
Example program:
```
print:1>=2
print:2>=2
print:3>=2
```
Output:
```
0
1
1
```

`(num)<=(num)`: Determine whether the first number is less than or equal to the second.
Example program:
```
print:1<=2
print:2<=2
print:3<=2
```

Output:
```
1
1
0
```

`(num)==(num)` or `(str)==(str)`: Determine whether two numbers or strings are equal.
Example program:
```
print:1==2
print:2==2
print:"ABC"=="DEF"
print:"ABC"=="ABC"
```
Output:
```
0
1
0
1
```

`(num)!=(num)` or `(str)!=(str)`: Determine whether two numbers or strings are not equal.
Example program:
```
print:1!=2
print:2!=2
print:"ABC"!="DEF"
print:"ABC"!="ABC"
```
Output:
```
1
0
1
0
```

`(str or list)[(num)]`: Access a string character or list member.
Example program:
```
A="ABCDE"
B=[10,20,30,40,50]
print:A[2]
print:B[2]
```
Output:
```
67
30
```
(67 is the ASCII code point for "C")

`(func):(val),(val),(val)...`: Invoke a user-defined function with arguments.
Example program:
```
func:TEST,A,B
    print:A+B+2
end;
TEST:1,3
TEST:5,7
```
Output:
```
6
14
```

`(func);`: Invoke a user-defined function without arguments.

Example program:
```
func:TEST
    print:"BREAD"
end;
print:"BEFORE"
TEST;
TEST;
print:"AFTER"
```
Output:
```
BEFORE
BREAD
BREAD
AFTER
```

Assignment

`(var)=(val)`: Assign a value to a variable.
Example program:
```
A=6
print:A
```
Output:
```
6
```

`(num var)+=(num)` or `(str var)+=(str)`: Add a number to a variable or concatenate a string to a variable.
Example program:
```
A=6
A+=1
print:A
B="BREAD"
B+="BOX"
print:B
```
Output:
```
7
BREADBOX
```

`(num var)-=(num)`: Subtract from a variable.
Example program:
```
A=6
A-=1
print:A
```
Output:
```
5
```

`(num var)*=(num)`: Multiply a variable.
Example program:
```
A=6
A*=2
print:A
```
Output:
```
12
```

`(num var)/=(num)`: Divide a variable.
Example program:
```
A=6
A/=2
print:A
```
Output:
```
3
```

`(num var)%=(num)`: Set a variable equal to the remainder when it is divided by the given number.
Example program:
```
A=6
A%=4
print:A
```
Output:
```
2
```

`(num var)&&=(num)`: Boolean AND a variable.
Example program:
```
A=1
A&&=0
print:A
```
Output:
```
0
```

`(num var)||=(num)`: Boolean OR a variable.
Example program:
```
A=0
A||=1
print:A
```
Output:
```
1
```

`(num var)^^=(num)`: Boolean XOR a variable.
Example program:
```
A=1
A^^=1
print:A
```
Output:
```
0
```

`(num var)&=(num)`: Bitwise AND a variable.
Example program:
```
A=3
A&=5
print:A
```
Output:
```
1
```

`(num var)|=(num)`: Bitwise OR a variable.

Example program:
```
A=3
A|=5
print:A
```
Output:
```
7
```

`(num var)^=(num)`: Bitwise XOR a variable.
Example program:
```
A=3
A^=5
print:A
```
Output:
```
6
```

`(num var)<<=(num)`: Bitshift a variable left.
Example program:
```
A=6
A<<=1
print:A
```
Output:
```
12
```

`(num var)>>=(num)`: Bitshift a variable right.
Example program:
```
A=6
A>>=1
print:A
```
Output:
```
3
```

Control Functions

`if:(num condition)`: Perform the following code block if the given condition is true.
Example program:
```
if:0
    print:10
end;
if:1
    print:20
end;
```
Output:
```
20
```

`elif:(num condition)`: Perform the following code block if the given condition is true and none of the prior code blocks were executed.
Example program:
```
if:0
    print:10
elif:0
```

```
    print:20
end;
if:0
    print:30
elif:1
    print:40
end;
if:1
    print:50
elif:1
    print:60
end;
```
Output:
```
40
50
```

`else;`: Perform the following code block if none of the prior code blocks were executed.
Example program:
```
if:0
    print:10
else;
    print:20
end;
if:1
    print:30
else;
    print:40
end;
```
Output:
```
20
30
```

`end;`: End a code block.
Example program:
```
if:1
    print:10
    if:0
        print:20
    end;
    print:30
end;
```
Output:
```
10
30
```

`while:(num condition)`: Repeat the following code block until the given condition is false.
Example program:
```
A=0
while:A<4
    A++
    print:A
end;
```

Output:
```
1
2
3
4
```

`break;`: Exit out of a `while` loop.
Example program:
```
A=0
while:A<4
    A++
    if:A==3
        break;
    end;
    print:A
end;
```
Output:
```
1
2
```

`cont;`: Jump to the beginning of a `while` loop and re-evaluate the condition.
Example program:
```
A=0
while:A<4
    A++
    if:A==3
        cont;
    end;
    print:A
end;
```
Output:
```
1
2
4
```

`func:(name),(argument),(argument),(argument)...`: Declare a function defined by the following code block. When a function is invoked, it creates its own local variable scope.
Example program:
```
func:TEST,A,B
    print:A*10
    print:B*100
end;
TEST:2,3
TEST:5,8
```
Output:
```
20
300
50
800
```

`ret:(val)`: Return a value from a user-defined function.
Example program:

```
func:TEST,A
    ret:A+15
end;
print:TEST:30
print:TEST:12
```
Output:
```
45
27
```

`ret;`: Return from a function without producing a value.
Example program:
```
func:TEST
    print:"BREAD"
    ret;
    print:"HELLO"
end;
TEST;
TEST;
```
Output:
```
BREAD
BREAD
```

`quit;`: Exit from the program.
Example program:
```
print:"BREAD"
quit;
print:"HELLO"
```
Output:
```
BREAD
```

Math Functions

`rand;` (returns number): Return a random number between 0 inclusive and 1 exclusive.
Example program:
```
print:rand;
print:rand;
print:rand;
```
Output:
```
0.7889
0.4825
0.5321
```

`randInt:(num minimum),(num maximum)` (returns number): Return a random number between the given minimum and maximum inclusive.
Example program:
```
print:randInt:1,10
print:randInt:1,10
print:randInt:1,10
```
Output:
```
3
10
```

```
8
```

`abs:(num)` (returns number): Return the absolute value of the given number.
Example program:
```
print:abs:-5
```
Output:
```
5
```

`round:(num)` (returns number): Round the given number to the nearest integer.
Example program:
```
print:round:8.4
print:round:8.5
print:round:8.6
```
Output:
```
8
9
9
```

`floor:(num)` (returns number): Round the given number down.
Example program:
```
print:floor:8.4
print:floor:8.6
```
Output:
```
8
8
```

`ceil:(num)` (returns number): Round the given number up.
Example program:
```
print:ceil:8.4
print:ceil:8.6
```
Output:
```
9
9
```

`sin:(num)` (returns number): Calculate the sine of the given number.
Example program:
```
print:sin:1
```
Output:
```
0.8415
```

`cos:(num)` (returns number): Calculate the cosine of the given number.
Example program:
```
print:cos:1
```
Output:
```
0.5403
```

`tan:(num)` (returns number): Calculate the tangent of the given number.
Example program:
```
print:tan:1
```
Output:
```
1.5574
```

`sqrt:(num)` (returns number): Calculate the square root of the given number.
Example program:
```
print:sqrt:9
```
Output:
```
3
```

`pow:(num),(num exponent)` (returns number): Raise the given number to the given power.
Example program:
```
print:pow:7,2
```
Output:
```
49
```

`log:(num),(num base)` (returns number): Calculate the log of the given number with the given base.
Example program:
```
print:log:100,10
```
Output:
```
2
```

I/O Functions

`print:(num or str)`: Print the given value and wait for user input.
Example program:
```
print:123
print:"HELLO WORLD"
```
Output:
```
123
HELLO WORLD
```

`reqStr;` (returns string): Request a string from the user.
Example program:
```
A=reqStr;
print:"HELLO "+A
```
Output:
```
HELLO (something)
```

`reqNum;` (returns number): Request a number from the user.
Example program:
```
A=reqNum;
print:A*2
```
Output:
```
(some number * 2)
```

`menu:(str title),(list options)` (returns number): Prompt the user to select an option from a menu. Return the index of the selected option.
Example program:
```
A=menu:"PICK ONE",["BREAD","PASTA","BUTTER"]
print:A
```
Output:

`(some index)`

`fExists:(str fileName)` (returns number): Return whether the given file exists.
Example program:
```
print:fExists:"MYFILE"
fCreate:"MYFILE"
print:fExists:"MYFILE"
```
Output:
```
0
1
```

`fSize:(str fileName)` (returns number): Return the size of the given file.
Example program:
```
fCreate:"MYFILE"
fWrite:"MYFILE","1234567"
print:fSize:"MYFILE"
```
Output:
```
7
```

`fCreate:(str fileName)`: Create a file with the given name.
Example program:
```
print:fExists:"MYFILE"
fCreate:"MYFILE"
print:fExists:"MYFILE"
```
Output:
```
0
1
```

`fDelete:(str fileName)`: Delete the given file.
Example program:
```
fCreate:"MYFILE"
print:fExists:"MYFILE"
fDelete:"MYFILE"
print:fExists:"MYFILE"
```
Output:
```
1
0
```

`fSetName:(str oldFileName),(str newFileName)`: Rename the given file.
Example program:
```
fCreate:"MYFILE"
fSetName:"MYFILE","BREAD"
print:fExists:"MYFILE"
print:fExists:"BREAD"
```
Output:
```
0
1
```

`fRead:(str fileName),(num index),(num amount)` (returns string): Read from the given file.
Example program:

```
fCreate:"MYFILE"
fWrite:"MYFILE","ABCDEFG"
print:fRead:"MYFILE",2,3
```
Output:
```
CDE
```

`fWrite:(str fileName),(str contents)`: Set the contents of the given file.
Example program:
```
fCreate:"MYFILE"
fWrite:"MYFILE","ABCDEFG"
print:fRead:"MYFILE",2,3
```
Output:
```
CDE
```

`fImport:(str fileName)`: Run the given file in the current scope.
Example program:
```
fCreate:"MYFILE"
fWrite:"MYFILE","print:123"
fImport:"MYFILE"
```
Output:
```
123
```

Value Functions

`num:(num or str)` (returns number): Convert the given value to a number.
Example program:
```
print:(num:"123")+1
```
Output:
```
124
```

`str:(num or str)` (returns string): Convert the given value to a string.
Example program:
```
print:(str:123)+"ABC"
```
Output:
```
123ABC
```

`type:(val)` (returns number): Return a number corresponding to the type of the given value.
Type numbers:
- 1 = Number
- 2 = String
- 3 = List
- 4 = Function

Example program:
```
print:type:"ABC"
```
Output:
```
2
```

`len:(str or list)` (returns number): Retrieve the length of the given sequence.
Example program:
```
print:len:"ABC"
```

Output:
```
3
```

`copy:(str)` (returns string) or `copy:(list)` (returns list): Return a shallow copy of the given value.
Example program:
```
A=[10,20,30]
B=copy:A
A[1]=100
print:A[1]
print:B[1]
```
Output:
```
100
20
```

`ins:(str),(num index),(num character)` or `ins:(list),(num index),(val)`: Insert a value into the given sequence.
Example program:
```
A="ABCD"
ins:A,1,'X'
print:A
```
Output:
```
AXBCD
```

`rem:(str or list),(num index)`: Remove a value from the given sequence.
Example program:
```
A="ABCD"
rem:A,1
print:A
```
Output:
```
ACD
```

`sub:(str),(num startIndex),(num endIndex)` (returns string) or `sub:(list),(num startIndex),(num endIndex)` (returns list): Retrieve a subsequence from the given sequence. The start index is inclusive, and the end index is exclusive.
Example program:
```
A="ABCD"
print:sub:A,1,3
```
Output:
```
BC
```

`insSub:(str),(num index),(str)` or `insSub:(list),(num index),(list)`: Insert a subsequence into the given sequence.
Example program:
```
A="ABCD"
insSub:A,1,"XY"
print:A
```
Output:
```
AXYBCD
```

`remSub:(str or list),(num startIndex),(num endIndex)`: Remove a subsequence from the given sequence. The start index is inclusive, and the end index is exclusive.
Example program:
```
A="ABCD"
remSub:A,1,3
print:A
```
Output:
```
AD
```

`equRef:(str),(str)` (returns number) or `equRef:(list),(list)` (returns number): Determine whether the two values have the same identity. In other words, determine whether changing a member of the first argument would change the second argument.
Example program:
```
A="ABCD"
B="ABCD"
print:equRef:A,B
print:equRef:A,A
```
Output:
```
0
1
```

Epilogue

I hope you enjoy your DUO Travel. If you have any questions, or if you just want to chat about the device, I would love to hear from you. My email address is esperantanaso@gmail.com. Good luck in your programming endeavors!