DUO BASIC

= = = USER MANUAL = = =

This manual assumes that you have a basic understanding of electronics and computers. It is divided into three parts:

1. **DUO Basic architecture**
2. **How to assemble the DUO Basic**
3. **How to operate the DUO Basic**

If you have purchased the DUO Basic kit not assembled or DUO Basic board, you should read all parts of the guide thoroughly. If you have purchased the DUO Basic kit fully assembled, you only need to read parts 1 and 3.

The DUO Basic webpage containing schematics and diagrams may be found here:

http://ostracodfiles.com/basic/menu.html

If you have any comments or questions, please contact me at esperantanaso@gmail.com.

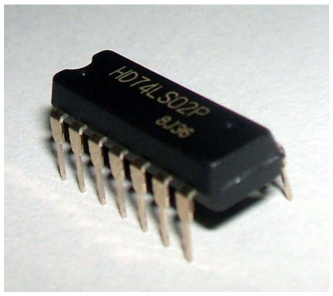
= = = 1. DUO BASIC ARCHITECTURE = = =

The DUO Basic is a relatively simple computer. It has 256 bytes of permanent memory for storing a program and 3 **registers** for storing temporary 8-bit values. The DUO Basic can execute two different kinds of instructions:

- **MNZ** (move if not zero): copy the contents of one register to another register if a third register is not zero.
- **ADD**: add the contents of two registers and store the result in a third register.

In order to understand the DUO Basic architecture, you must first understand some principles of digital logic.

Digital logic circuits are typically composed of **7400 series chips**. Each chip contains many **transistors**, which act as tiny electronic switches. The DUO Basic contains thirteen 7400 series chips. Here is a picture of such a chip:



Each 7400 series chip is labeled with a number. You can use this number to find the chip's data sheet on the Internet. A data sheet describes the function of each pin on a chip. The "top" of a chip is marked by a **semicircular indent**. All chips need to be attached to a **power supply**. Typically, the bottom left pin should be attached to **ground** (negative), and the top right attached to **source** (positive). The remaining pins are either inputs or outputs.

A wire may be in **one of three states**:

- **High** (attached to source)
- **Low** (attached to ground)
- **Detached** (attached to neither ground nor source; sometimes called high-impedance)

Never connect a high output to a low output, or you will have a short circuit. A short circuit will quickly drain your power source and will cause wires to become hot. It is OK to attach a high or low output to a wire in the detached state; **high or low will override detached**. All relevant inputs of a chip should either be high or low, not detached. The chip will become confused if an input is not well defined.

In everyday life, people use numbers in **base 10** (decimal), meaning we represent numbers using 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Computers tend to store numbers in **base 2** (binary). Numbers in this system use only the digits 0 or 1. A binary digit is called a **bit**, and may be electronically represented by a high or low signal.

In base 10, each digit place is worth 10 times more than the place to the right, and the rightmost digit is worth 1. 300 is 10 times bigger than 30 because the 3 in 300 is to the left of the 3 in 30. **In base 2, the rightmost digit is still worth 1, but each digit place is worth 2 times more than the place to the right**. 100 in binary is 2 times bigger than 10, and 10 is 2 times bigger than 1. By doing some multiplication, you will find that 100 in binary is equal to 4 in decimal.

In fact, you can easily conclude that **each binary digit is worth 2^n**, where n is the distance from the rightmost digit. To convert the binary number 1101, we can add 2^3 + 2^2 + 2^0. This is equal to 8 + 4 + 1, or 13.

8-bit computers, as their name suggest, process 8 bits of information at a time. A group of 8 bits is called a **byte**. A byte can store an integer between 0 and 255.

The acronym **OE** stands for "output enable". Some chips have an OE pin. When the OE pin has a high input, all output pins of the chip will enter the detached state. When the OE pin has a low input, the output pins will enter a high state or low state (depending on other inputs).

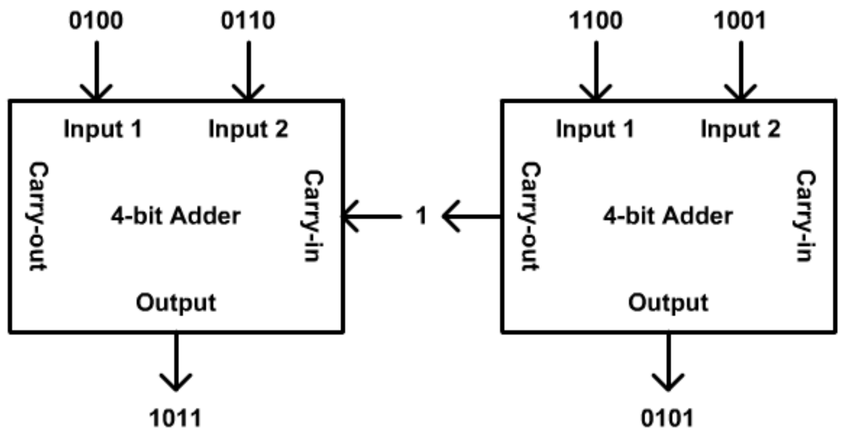
CP stands for "clock pulse", and **WE** stands for "write enable". These are typically used to accept data into memory or advance a counter. They are usually triggered by a rising edge (a low signal followed by a high signal). Note that I may sometimes use WE to refer to both write enable and clock pulse.

In some chips (such as EEPROM), certain pins can behave as both inputs and outputs. The direction of data depends on OE.

One common chip in the DUO Basic is **74HC541**, an **8-bit buffer**. This chip has 8 data inputs, 8 data outputs, and 2 OE pins. When both OE pins are low, values are copied from the data inputs to the data outputs. When both OE pins are high, the data output enters the detached state.

The DUO Basic contains many **74HC574** chips, also called an **8-bit flip-flop**. The chip has 8 data inputs, 8 data outputs, an OE pin, and a WE pin. When WE changes from low to high, the 8 data inputs are stored into the internal register of the 74HC574. When OE is low, the contents of this register are placed onto the 8 data outputs. If OE is high, the data outputs become detached.

The **74HC283**, a **4-bit adder**, has 8 data inputs, 4 data outputs, a carry-in pin, and a carry-out pin. The adder will calculate the sum of the binary input and yield a 4 bit binary number. If the result is greater than 15, the carry-out pin becomes high. The carry-out of one adder may be fed into the carry-in of another adder to increase the number of bits. This technique is shown in the diagram below:



The program counter of the DUO Basic is implemented with two **74HC161**, a **4-bit counter**. This chip has 4 data inputs, 4 data outputs, a clear pin, a load pin, a CP pin, a carry-in pin, and a carry-out pin. The counter stores a 4-bit number which is always placed on the 4 data outputs. When the load pin is high and the CP pin switches from low to high, the number increases by 1. If the count reaches 15, the carry-out pin becomes high. When the load pin is low and the CP pin switches from low to high, the number is set to the value of the 4 data inputs. When the clear pin is low, the number is set to zero. The carry-out of one counter may be fed into the carry-in of another counter to increase the number of bits.

The **74HC540**, an **8-bit inverter**, works in almost the same way as the 8-bit buffer. The difference is that the inverter will invert the data outputs. This means that a low input becomes a high output, and a high input becomes a low output.

The clock of the DUO Basic is centered around the **74HC4017**, a **decade counter**. The chip has 10 data outputs, a CP pin, and a reset pin. The decade counter begins with the first output high and all other outputs low. When CP changes from low to high, the next output in the sequence becomes high and the last one becomes low. If the reset pin is high, the first output becomes high again and all other outputs become low.

The **74HC14**, a **hex inverting Schmitt trigger**, is included in the DUO Basic for technical reasons. When a button is connected directly to the CP pin of any chip, pressing the button once can cause the CP pin to be triggered many times. During the milliseconds when the button metal plates come in contact, the electrical connection is unstable. This phenomenon is known as “bounce”. By adding a capacitor and a Schmitt trigger to the button, the instability is removed from button presses and the resulting output is a single stable signal. This is called “debouncing”.

The last chip in the DUO Basic is the **AT28C64B**, an **EEPROM**. As the part number implies, this chip is not a 7400 series chip. The EEPROM has 8 data pins, several address pins, a WE pin, and an OE pin. EEPROM is used to store an array of values and save them even when power is shut off. When WE changes from low to high, the value on the data pins is stored at the selected address. When OE is low, the value at the selected address is placed on the data pins.

Digital logic circuits may also contain **resistors** and **diodes**. Here are pictures of a resistor (on the left) and a diode (on the right):

Every **resistor** is associated with a level of **resistance** measured in **ohms**. You can find the resistance of a resistor by reading its colored stripes. Resistance measures how difficult it is for **current** (flow of electrons) to pass through a resistor. There is a famous formula called **Ohm’s Law** which quantifies this relationship:

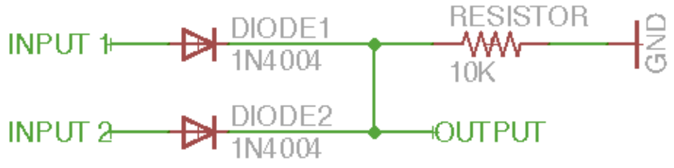
$$I = V / R$$

I stands for current through the resistor measured in amps, V stands for the voltage between the ends of the resistor measured in volts, and R stands for the resistance of the resistor measured in ohms.

A **diode** acts as a one-way valve for current. Current can only travel in one direction through a diode. The **cathode** of a diode is marked by a black stripe. The other side of a diode is called the **anode**. Electrons will only move from the cathode to the anode. Note that **ground is considered to produce electrons while source is considered to consume electrons**. The names of ground and source are somewhat counterintuitive.

Note that an **LED** (light emitting diode) is a special kind of diode which produces light when current passes through it. **Without a resistor, an LED can create a dangerous short circuit! The DUO Basic kit has 24 red LEDs with built in resistors**.

Consider the following circuit. The cathode of each diode is on the right side. What is the state of the output when each input is connected directly to source or ground? We will consider each case in turn.



Suppose both inputs are grounded. Electrons can't travel from the left side of each diode to the right, but the output is still connected to ground with a resistor. As a result the output is considered to be **low**.

Suppose both inputs are sourced. Electrons will now flow from ground through the resistor and through each diode to each input. According to Ohm's Law there will be a voltage drop from the left side of the resistor to the right. The voltage of the output will be high relative to ground, so the output is said to be in a **high** state.

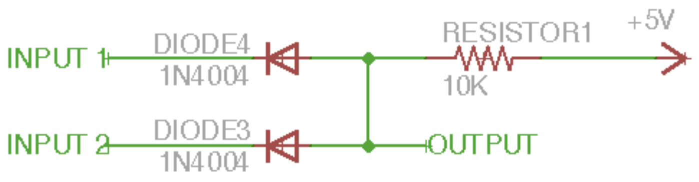
Now **suppose input 1 is grounded and input 2 is sourced**. In a similar fashion to the previous case, electrons will flow from ground through the resistor and through the bottom diode to input 2. There will be no short circuit between inputs because electrons cannot flow from input 1 through the top diode to input 2. The output will be in the **high** state again.

We can form a table out of this information. In digital logic this is known as a **truth table**.

Input 1:	Input 2:	Output:
Low	Low	Low
High	Low	High
Low	High	High
High	High	High

The circuit we created is called an **OR gate** because the output is high when input 1 or input 2 is high. The DUO Basic contains two OR gates of this type.

Now consider this slightly different circuit. Note how the diodes are oriented the other way and the resistor is attached to source instead of ground.



Suppose both inputs are sourced. Electrons can't travel from the right side of each diode to the left, but the output is still connected to source with a resistor. As a result the output is considered to be **high**.

Suppose both inputs are grounded. Electrons will now flow from each input through each diode and through the resistor to source. According to Ohm's Law there will be a voltage drop from the right side of the resistor to the left. Therefore the output will be in a **low** state.

Now **suppose input 1 is sourced and input 2 is grounded**. Similar to the previous case, electrons will flow from input 2 through the bottom diode and through the resistor to source. There will be no short circuit between inputs because electrons cannot flow from input 2 through the top diode to input 1. The output will be in the **low** state again.

Let us make another truth table.

Input 1:	Input 2:	Output:
Low	Low	Low
High	Low	Low
Low	High	Low
High	High	High

The circuit we created is called an **AND gate** because the output is high when input 1 and input 2 are high. The DUO Basic contains eight AND gates of this type.

Having described the elemental components and logic gates used in the DUO Basic, we can now specify the elements of each functional area of the computer. The DUO Basic is divided into four functional areas:

- **Registers**
- **Arithmetic logic unit (ALU)**
- **Program memory**
- **Processor clock**

There are three **registers** in the DUO Basic. The purpose of each register is to store an 8-bit value. The A register and B register are implemented with 74HC574 flip flops. The PC register (program counter) is implemented with two 74HC161 counters. The purpose of the program counter is to keep track of the current instruction being executed.

The **program memory** is implemented with a single AT28C64B EEPROM. The purpose of the program memory is to store instructions to be executed by the processor. The address pins of the EEPROM are connected to the output of the PC register.

The purpose of the **ALU** is to either add numbers or copy a number. The ALU contains two 74HC283 adders and two 74HC541 buffers. The input of the ALU is connected to the output of the A register, B register, and program memory. The output of the ALU is connected to the input of the A register, B register, and PC register.

The **processor clock** contains various parts including a 74HC4017 decade counter, a 74HC14 Schmitt trigger, a 74HC574 flip flop, a 74HC540 inverter, and many diodes and resistors. The purpose of all of these parts is to control the timing of OE and WE signals in the computer. The processor clock uses a flip flop to store the opcode (operation code) of the current instruction being executed. The input of this flip flop is the output of the program memory. The processor clock uses the output of the A register to conditionally inhibit certain instructions.

The DUO Basic has a power switch, 6 buttons, and 8 data bus switches. The **reset decade counter** button resets the processor clock. The **reset program counter** button sets the program counter to zero. The **decade counter clock** button advances program execution. The **program counter clock** button increases the program counter by one.

The **data bus** is a group of 8 wires connected to many locations in the DUO Basic, including program memory. The current value on the data bus is displayed on 8 LEDs and may be set with the 8 data bus switches. To place the contents of EEPROM onto the data bus, press the **EEPROM OE** button. To store the value on the data bus in EEPROM, press the **EEPROM WE** button.

= = = 2. HOW TO ASSEMBLE THE DUO BASIC = = =

If you have ordered the DUO Basic kit, you will already have all the electronic parts you need to build the DUO Basic. If you have ordered the DUO Basic board, you must order the parts in the following list:

- (x2) 4-bit counter: SN74HC161N
- (x3) 8-bit buffer: SN74HC541N
- (x3) 8-bit flip flop: SN74HC574N
- (x2) 4-bit adder: CD74HC283E
- (x1) 8-bit inverter: SN74HC540N
- (x1) Decade counter: CD74HC4017E
- (x1) Schmitt trigger: SN74HC14N
- (x1) Small EEPROM: AT28C64B-15PU
- (x1) 28 pin IC socket: 110-99-628-41-001000
- (x7) 20 pin IC socket: 4820-3000-CP
- (x5) 16 pin IC socket: 1-390261-4
- (x1) 14 pin IC socket: 1-390261-3
- (x7) 1K ohm resistor: 291-1K-RC
- (x18) 10K ohm resistor: 291-10K-RC
- (x24) LED with resistor: WP710A10ID5V
- (x32) Diode: 1N4150TR
- (x3) Small capacitor: EEA-GA1H100H
- (x6) Button: PTS645SH50-2 LFS
- (x9) Switch: MHSS1104
- (x1) Battery holder: 12BH331/C-GR

You can find all of these parts on www.mouser.com. Other suppliers, such as www.digikey.com, may also have some of these parts.

⚠ **Warning! Chips can be damaged by electrostatic charges!** I recommend that you use an antistatic wrist strap and avoid carpets when working with chips. If you don't want to use an antistatic wrist strap, you can touch a large metal object instead. A metal grate or pipe should work OK.

⚠ **Second warning! Heat from a soldering iron can damage chips!** You should not solder chips directly into the DUO Basic board. Instead solder the IC sockets into the board, and then insert the chips into the sockets.

To solder a component into the board, first place the component on the front of the board, pushing the pins through corresponding holes. Next place a piece of masking tape on top of the component so it does not move around. Then flip the board over, heat one of the pin and hole junctions, and push **60/40 rosin core solder** onto the area. Repeat for all other pins. If the component has long pins, cut the extra wire length with wire cutters after soldering. For a video explanation of how to solder, please visit this link:

http://www.youtube.com/watch?v=I_NU2ruzyc4

Avoid putting solder metal directly on the soldering iron. If you accidentally form a solder bridge between pins, try to pull the solder away by using the soldering iron. If you put solder onto a hole without a pin, you will not be able to push a pin into the hole in the future. If this does happen, simultaneously heat the hole with the soldering iron while pushing the pin through.

Resistor positions are marked on the board with the letter R and a resistance. 10K resistors have the color code brown-black-orange. 1K resistors have the color code brown-black-red. If you mix up the resistors, the computer may not work.

Diode positions are marked on the board with D and a plus and minus sign. The black stripe on the diode should be oriented toward the minus sign.

Capacitor positions are marked on the board with a C and a plus and minus sign. A capacitor looks like a black cylinder with a white stripe. Orient the white stripe toward the minus sign on the board.

LED positions are marked on the board with an L and a plus and minus sign. The shorter pin of the LED should be closer to the minus sign. **If you decide to use LEDs not included with the DUO Basic kit, they must have built in resistors! Otherwise your computer will become hot and melt**.

Button positions have four holes in a rectangle shape. The button pins may require slight bending to fit in the holes.

Battery pack wire positions are marked on the board with "–Pwr+". The black wire of the battery pack should be connected closer to the minus sign, and the red wire of the battery pack should be connected closer to the plus sign.

When you have finished soldering all of the parts, **insert the chips into the IC sockets with the semicircular indent on top**. **Be careful not to bend IC pins** when inserting ICs into sockets.

The DUO Basic requires 3 AA batteries. **Before putting batteries in the DUO Basic, make sure that the power switch is in the position to the right**.

= = = 3. HOW TO OPERATE THE DUO BASIC = = =

The DUO Basic interprets its own set of instructions called **machine code**. Each DUO Basic machine code instruction is 2 bytes long and has the following format:

[2 bit operation] [3 bit source] [3 bit destination] [8 bit constant]

The **operation** may be **01** for **MNZ** (move if not zero) or **10** for **ADD**. MNZ will copy data from source to destination only if the A register is not zero. ADD will add source and the A register then store the result in the destination.

The **source** may be **011** for the A register, **101** for the B register, or **110** for the instruction constant (the last 8 bits of the instruction).

The **destination** may be **100** for the A register, **010** for the B register, or **001** for the program counter. Note that the program counter is incremented after each instruction is executed, so jump instructions must be adjusted accordingly.

As an example, suppose that you want to create an instruction which performs MNZ from the constant number 3 to register B. We put together the instruction by concatenating the bits 01 (MNZ), 110 (instruction constant source), 010 (B register destination), and 00000011 (instruction constant). The resulting machine code is 01110010 00000011.

The process of putting together an instruction is called **assembly**. The DUO Basic **assembly code** we just assembled is “MNZ 3 B”. As another example, let us assemble the assembly code “ADD B A”. We concatenate 10 (ADD), 101 (B register source), 100 (A register destination), and 00000000 (instruction constant). The resulting machine code is 10101100 00000000. In this case, the instruction constant field can be anything because it is not used as an operand of the instruction. However, if the constant field is excluded, the DUO Basic will be confused about the location of the next instruction.

It is possible to assemble an entire program by hand using the rules described above. However, this process is somewhat repetitive and boring. To facilitate program creation, I have made an **assembler** in JavaScript. It is available on the DUO Basic webpage: http://ostracodfiles.com/basic/menu.html

Note that the assembler outputs machine code in **base 16** (hexadecimal). The digits of hexadecimal in order are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Each hexadecimal digit place is worth 16 times more than the digit place to the right. The rightmost digit place is worth 1.

To convert from hexadecimal to binary, convert each hexadecimal digit into its binary representation using the table below. Then concatenate all of the bits.

Base 16	Base 2		Base 16	Base 2
0	0000		8	1000
1	0001		9	1001
2	0010		A	1010
3	0011		B	1011
4	0100		C	1100
5	0101		D	1101
6	0110		E	1110
7	0111		F	1111

Try typing “MNZ 3 B” into the assembler and pressing the “convert” button. The output text will be “0000: 72 03”. “0000” represents the start address for the machine code. “72 03” is 01110010 00000011 in binary, which matches the machine code we produced manually.

To run a program, you must first load the program into the machine. Let us begin by trying a simple counter program:

```
MNZ 0 A
ADD 1 A
MNZ 1 PC
```

This program will first clear the A register, increase the A register by 1, and then loop to the second instruction. If you assemble the program, you will obtain the following hexadecimal machine code:

```
0000: 74 00 B4 01
0004: 71 01
```

For your convenience, here is the same machine code in binary:

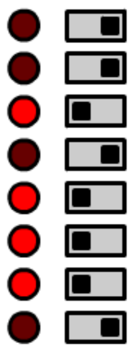
```
01110100 00000000
10110100 00000001
01110001 00000001
```

When you are ready to start using the DUO Basic, **make sure that the data bus switches are in the right position, and toggle the power switch to the left position**.

The program counter and decade counter need to be set to zero before loading the program. **Press the reset decade counter and reset program counter buttons** to reset each counter.

Now **place the first 8 bits of the program onto the data bus** by using the data bus switches. Note that **the switches (and LEDs) are ordered from least significant on top to most significant on the bottom**. As an example, the value 01110100 should be entered in the following way:

01110100



After you have placed the value onto the bus, **press the EEPROM WE button** to store the value in EEPROM. To make sure that the value was written correctly, **toggle all of the data bus switches to the right and press the EEPROM OE button**. The value you wrote should be displayed on the data bus LEDs.

When you are certain that the value was written correctly, **press the program counter clock button** to advance the program counter (and hence increase the current EEPROM address). The program counter LEDs will display the new value of the program counter. Note that the program counter LEDs are ordered from least significant on the left to most significant on the right.

Repeat the directions listed above to write the remaining groups of 8 bits in the program. After you have finished, you should check the whole program. To check the program, first **toggle the data bus switches to the right and reset the program counter**. Then **hold the EEPROM OE button and inspect the value on the data bus**. Advance to the next 8 bits by **pressing the program counter clock button**.

After you load the program, running it is easy. First **toggle all of the data bus switches to the right and press the reset decade counter and reset program counter buttons**. Then **press the decade counter clock button repeatedly**. You will see the LEDs on the computer blink while the program runs. If the contents of the A register slowly increase, your program is working! If not, make sure that the program has been entered properly and all components have been soldered appropriately.

Here are some programming challenges for you to try:

1. Add the numbers 50 and 17.
2. Count down from 10 by ones.
3. Display the numbers 3 and 12 in alternating order.
4. Add the consecutive integers from 6 to 1.
5. Count up until the number 8 has been reached.
   The solutions to these problems are on the next page.

Solutions to programming challenges:

1.

```
MNZ 0 A
ADD 5 A
MNZ 7 B
ADD B A
```

2.

```
MNZ 0 A
ADD 10 A
ADD 255 A
MNZ 3 PC
```

3.

```
MNZ 0 A
ADD 1 A
MNZ 3 A
MNZ 12 A
MNZ 3 PC
```

4.

```
MNZ 0 A
ADD 6 A
MNZ 0 B
ADD B B
ADD 255 A
MNZ 5 PC
ADD B A
MNZ 13 PC
```

5.

```
MNZ 0 A
ADD 1 A
MNZ A B
ADD 248 A
MNZ 13 PC
ADD 1 A
MNZ 11 PC
MNZ B A
MNZ 1 PC
```