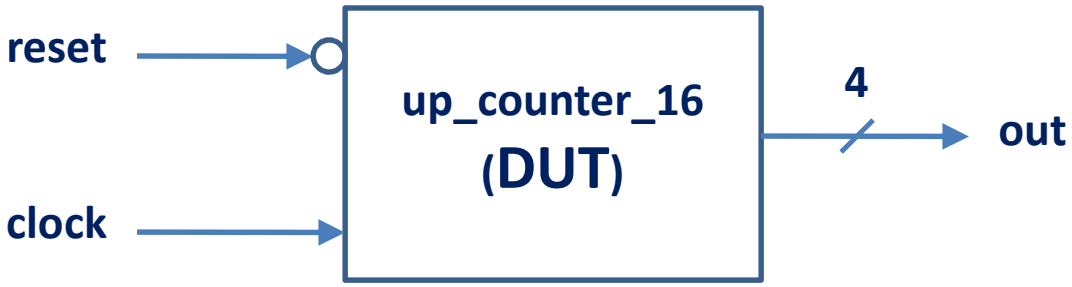
Questasim Tutorial (GUI/Windows) Continued

REF: **ModelSim® Tutorial**
Software Version 10.4c

Part 1: Reqs and Specs

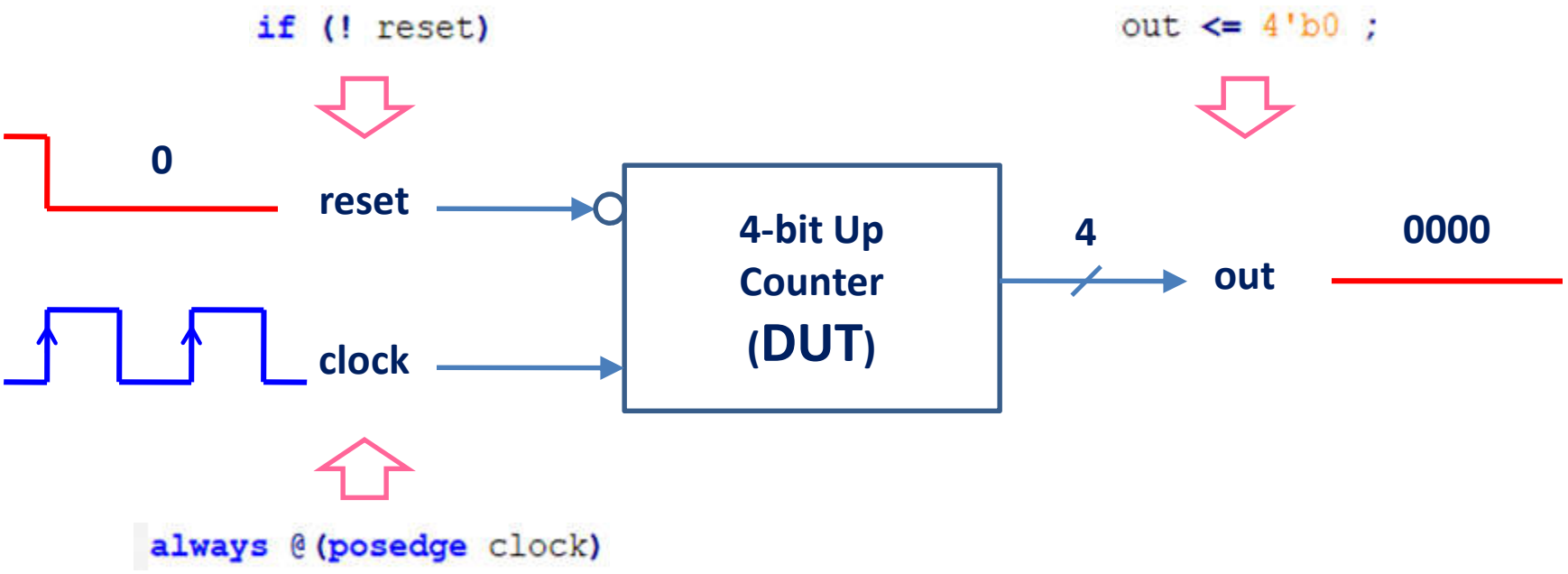
Design of 4-bit Up Counter




```
1   module up_counter_16    (
2                            out,    // Output of the counter
3                            clock,  // clock Input
4                            reset   // reset Input
5                            );
6
7   input clock, reset;
8   output [3:0] out;
9
10  reg [3:0] out; // internal variable
11
12  always @(posedge clock)
13      if (! reset) out <= 4'b0 ;
14      else         out <= out + 1;
15
16  endmodule
```

Reqs/Specs for the Design

```
always @(posedge clock)
    if (! reset) out <= 4'b0 ;
```

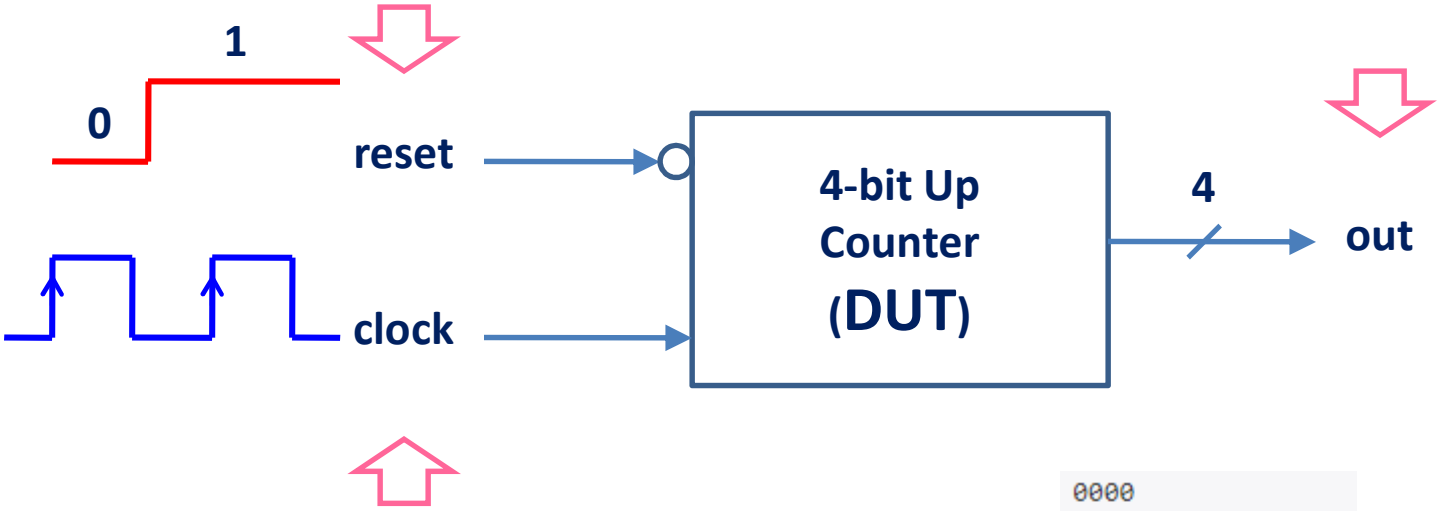


Reqs/Specs for the Design (2)

```
always @(posedge clock)
    if (! reset) out <= 4'b0 ;
```

```
if (! reset)
else
```

```
out <= out + 1;
```

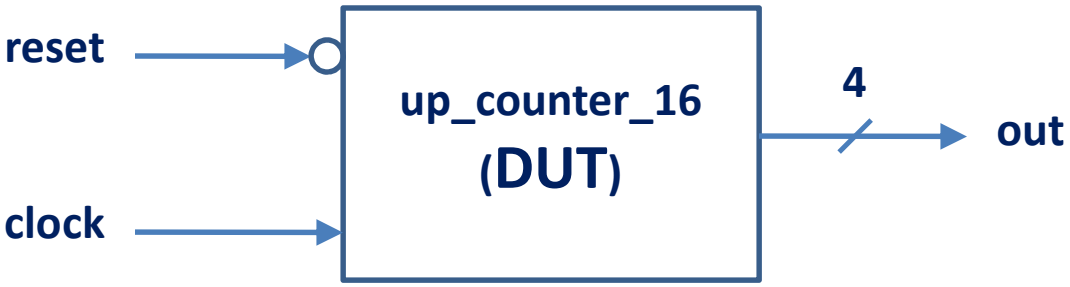



```
always @(posedge clock)
```

```
0000
0001
0010
...
1110
1111
        rolls over
0000
0001
...
```

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

4-bit Up Counter (3)

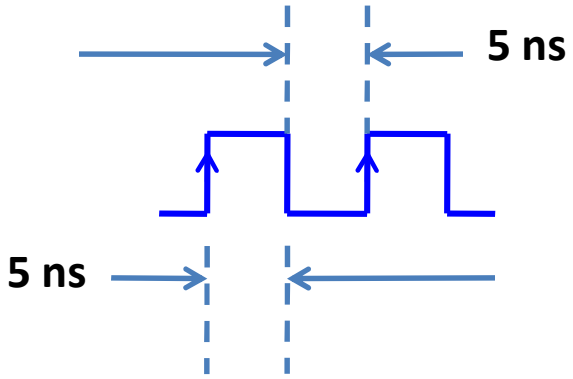



```
up_counter_16 dut (
                   .clock (   ),
                   .reset (    ),
                   .out (  )
                  );
```

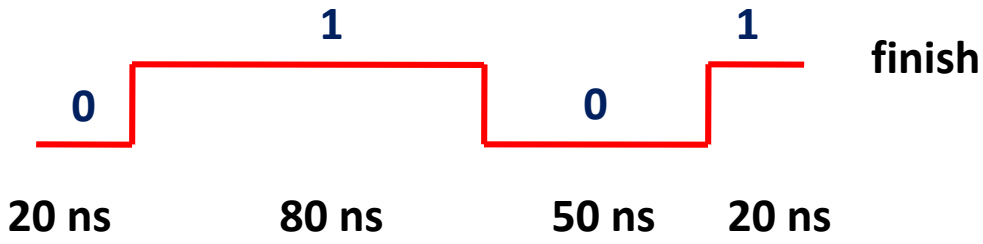
Part 2: Using Testbench File for Stimulus

Testbench File for Stimulus

```
always #5 clk = ~clk;
```




```
// 1. Initialize testbench variables to 0
clk <= 0;
rstn <= 0;
// 2. Drive rest of the stimulus
#20 rstn <= 1;
#80 rstn <= 0;
#50 rstn <= 1;
// 3. Finish the stimulus after 200ns
#20 $finish;
```

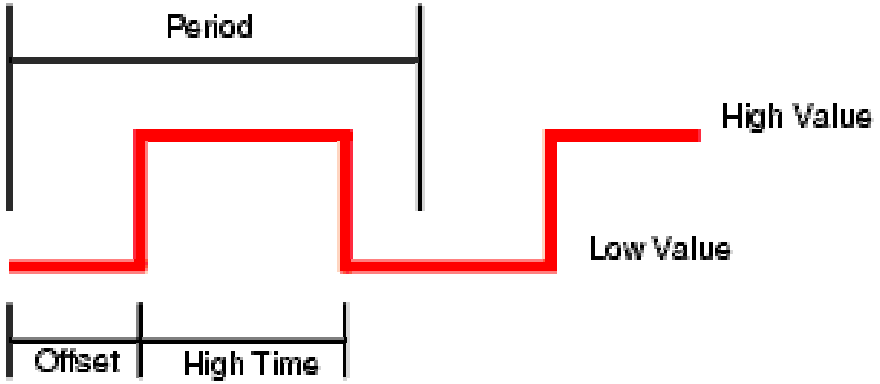


```
`timescale 1 ns/10 ps

module up_counter_16_tb;

reg clk;
reg rstn;
wire [3:0] out;

// Instantiate counter design
up_counter_16 dut (
                .clock (clk),
                .reset (rstn),
                .out (out)
                );


always #5 clk = ~clk;

initial
    begin
        // 1. Initialize testbench variables to 0
        clk <= 0;
        rstn <= 0;
        // 2. Drive rest of the stimulus
        #20 rstn <= 1;
        #80 rstn <= 0;
        #50 rstn <= 1;
        // 3. Finish the stimulus after 200ns
        #20 $finish;
    end
endmodule
```

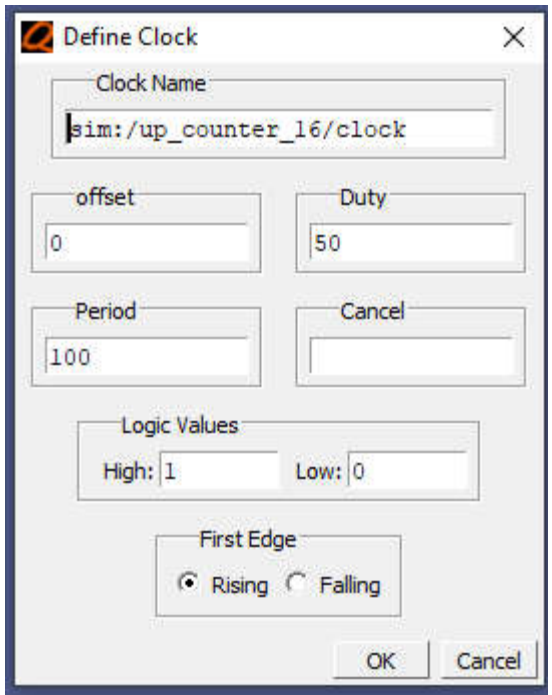
Part 3: Using ‘Add to List’ for Stimulus

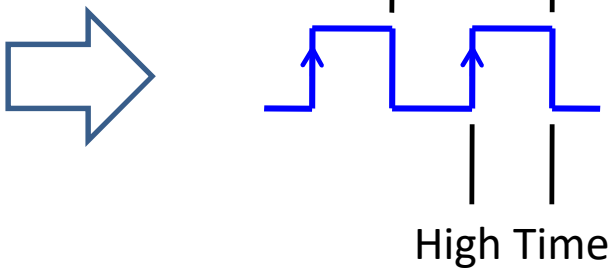
Defining Clock

- For clock signals starting on the **rising edge**,
  - the definition for Period, Offset, and Duty Cycle is as follows:

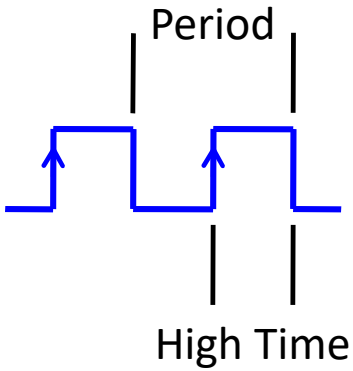


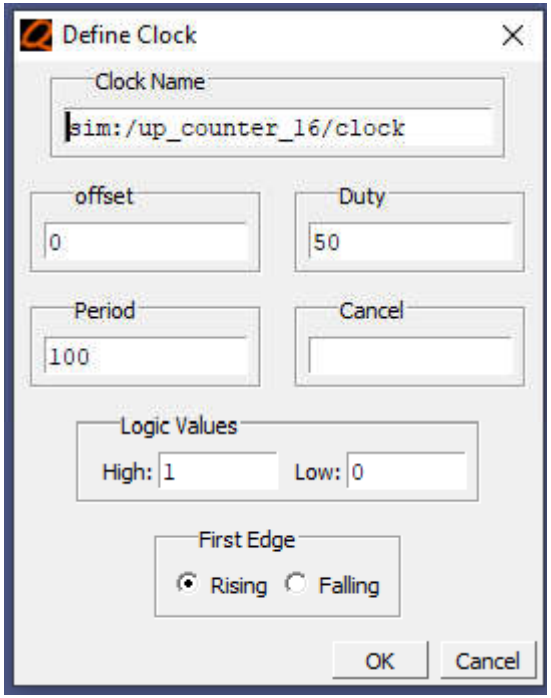
Duty Cycle = High Time/Period

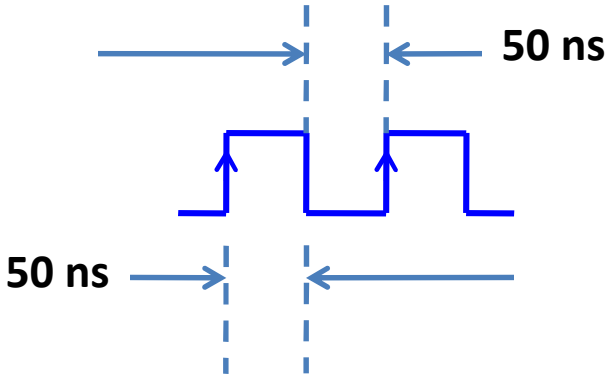




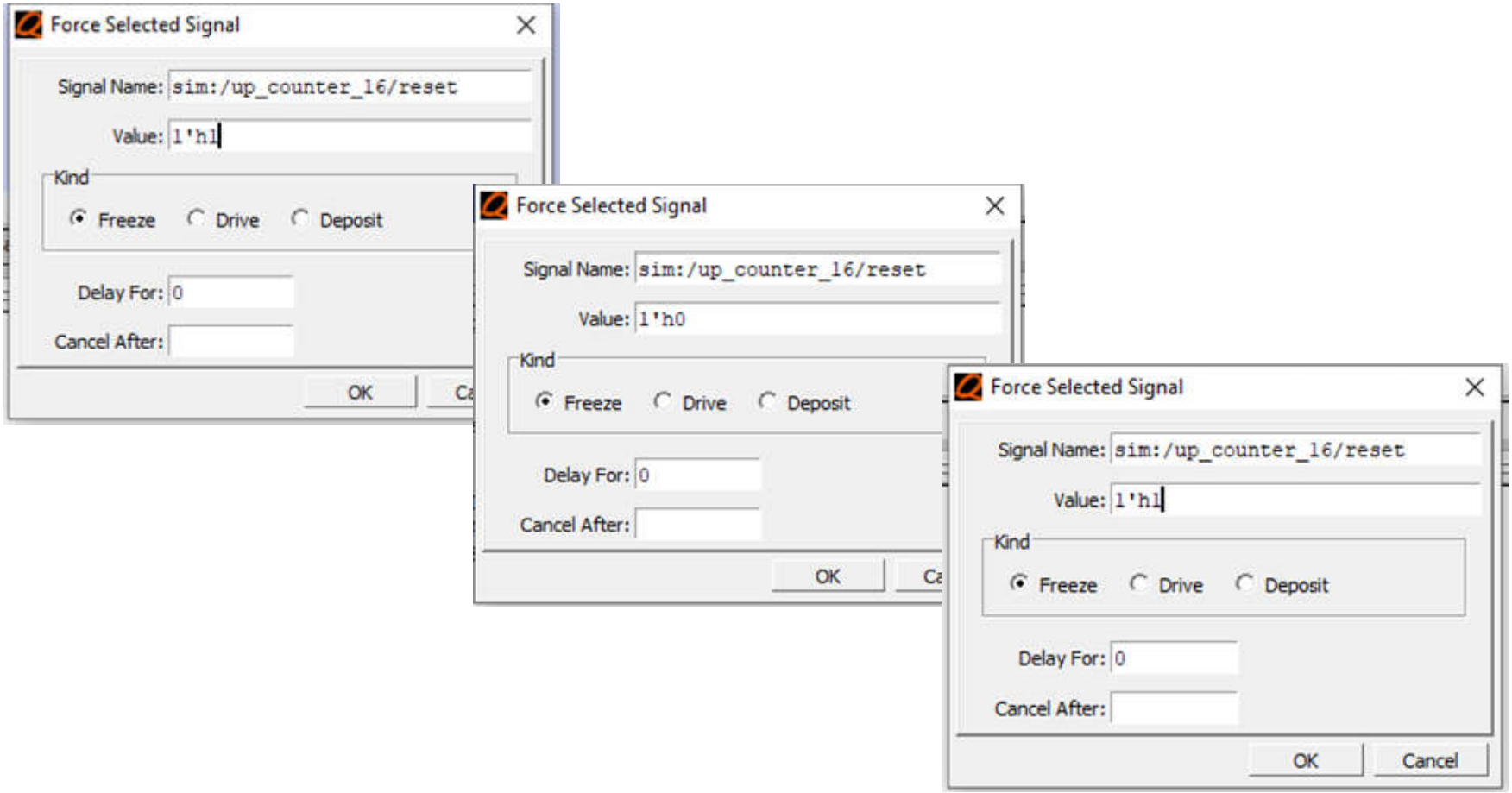
Apply Clock


Period
High Time


Define Clock
Clock Name
sim:/up_counter_16/clock
offset
0
Duty
50
Period
100
Cancel
Logic Values
High: 1
Low: 0
First Edge
Rising
Falling
OK
Cancel


50 ns
50 ns

Force Reset to ‘1’ / ‘0’ / ‘1’



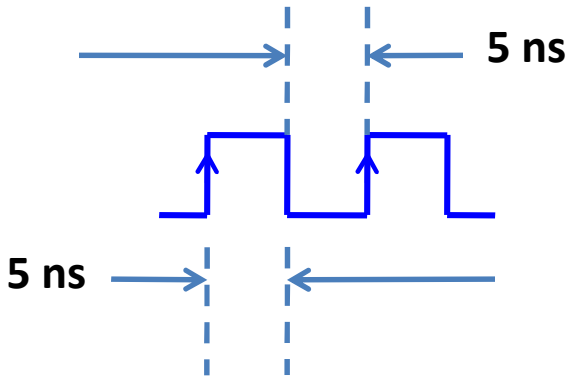
Part 4: Using .DO file for Stimulus

.DO file: Apply Clock

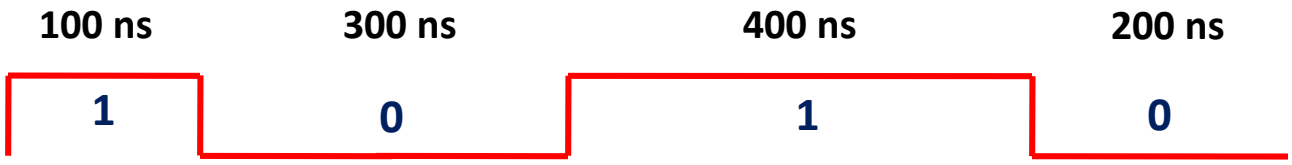
force -freeze clock 0 0, 1 {5 ns} -r 10

0: value at time 0 ns
1: value applied at 5 ns
-r : to repeat at required interval
10: the number of ns of the repeat interval

```
always #5 clk = ~clk;
```



Apply Reset signal



.DO file (sim.do)

```
1  vsim up_counter_16
2  add wave out
3  add wave clock
4  add wave reset
5  force -freeze clock 0 0, 1 {50 ns} -r 100
6  force reset 1
7  run 100
8  force reset 0
9  run 300
10 force reset 1
11 run 400
12 force reset 0
13 run 200
```

Thank You