

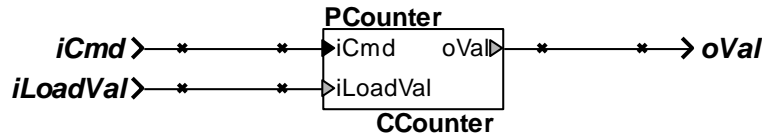
ExamplesSeq 01

Counter

This example illustrates the sequential implementation of a counter.

The psC program

The main schematic of the counter is:



The counter has two input ports and one output port, the count value “oVal”. It receives commands on “iCmd” and the “iLoadVal” is used to change the output “oVal”. The commands are defined in a small library:

```
library CounterLib
{
    enum Cmd_t { cReset, cUp, cDown, cLoad };
};
```

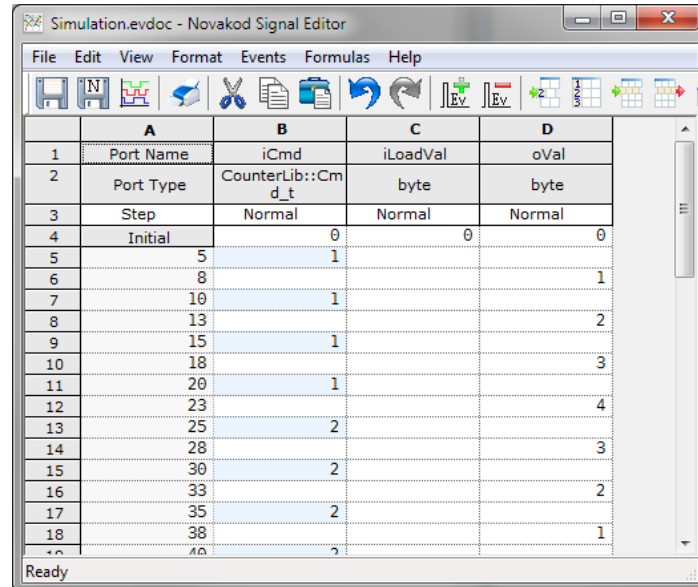
The psC code of the counter is self explanatory:

```
component CCounter (in active Cmd_t iCmd,
                    in passive byte iLoadVal,
                    out passive byte oVal)
{
    sequential ExecCmd()
    {
        while(!istrig(iCmd)) {}; // PC
        switch(iCmd) // 1
        {
            case cReset:
                oVal = 0b; // 2
                break(true); // 3
            case cUp:
                oVal++; // 4
                break(true); // 5
            case cDown:
                oVal--; // 6
                break(true); // 7
            case cLoad:
                oVal = iLoadVal; // 8
        }; // Infinite loop 9
    };
};
```

Compiling and running the application

You will now compile and execute the program, then use the signal editor and signal viewer to see the results.

- 1) Double-click on “main.rpj” to start Novakod Studio.
- 2) Double-click on the “main” component to view the schematic.
- 3) Now select the menu Run→Run N Steps to simulate.
- 4) Double click on “Simulation.evo”. You will see the simulated outputs.

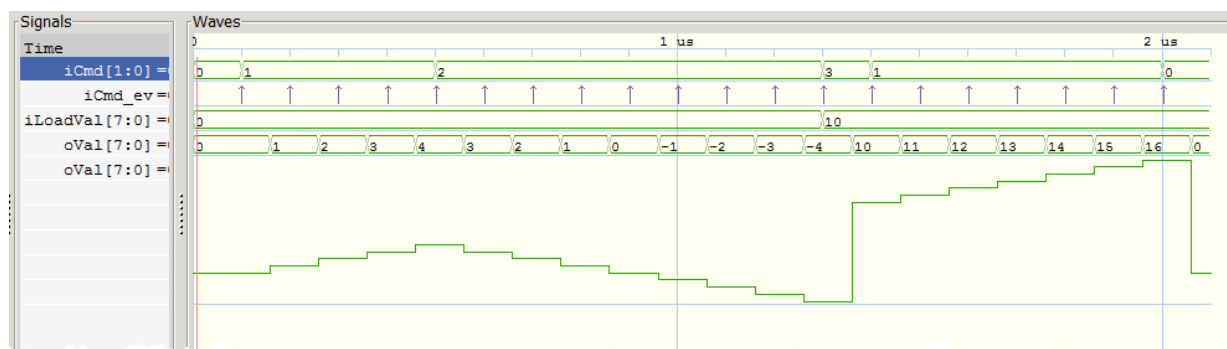


The screenshot shows the Novakod Signal Editor window with a table of simulation data. The table has four columns: A (Port Name), B (iCmd), C (iLoadVal), and D (oVal). The rows represent simulation steps from 1 to 19. The 'Initial' row shows values 0 for iCmd, 0 for iLoadVal, and 0 for oVal. Subsequent rows show values for iCmd (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19) and iLoadVal (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19). The oVal column shows values 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19.

	A	B	C	D
1	Port Name	iCmd	iLoadVal	oVal
2	Port Type	CounterLib::Cmd_t	byte	byte
3	Step	Normal	Normal	Normal
4	Initial	0	0	0
5	5	1		
6	8			1
7	10	1		
8	13			2
9	15	1		
10	18			3
11	20	1		
12	23			4
13	25	2		
14	28			3
15	30	2		
16	33			2
17	35	2		
18	38			1
19	40	2		

As it is executed sequentially, it takes a few cycles for the output to change. A command is sent every 5 steps and the result appears 3 steps later.

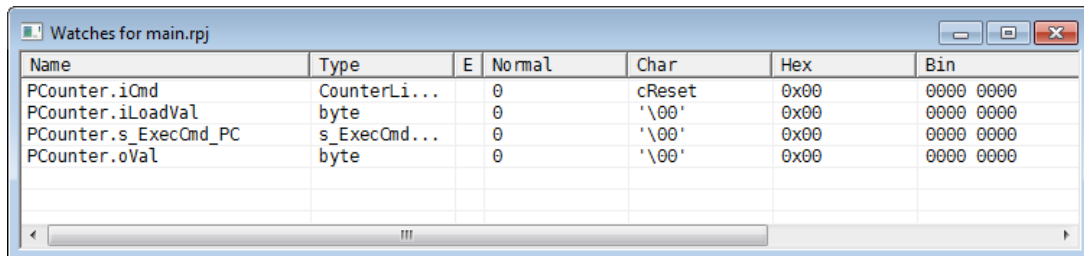
- 5) In the signal editor click the view signal button to see the signals:



Running in watch mode

For a better understanding of sequential execution, you can execute in watch mode.

- 1) Now select the menu Run→Paused to begin simulation.
- 2) Add the signals from the Inspect windows, as shown:



The screenshot shows a window titled "Watches for main.rpj" with a table of variables. The table has columns for Name, Type, E, Normal, Char, Hex, and Bin. The variables listed are PCounter.iCmd, PCounter.iLoadVal, PCounter.s_ExecCmd_PC, and PCounter.oVal. The values for these variables are shown in the Normal, Char, Hex, and Bin columns.

Name	Type	E	Normal	Char	Hex	Bin
PCounter.iCmd	CounterLi...	0		cReset	0x00	0000 0000
PCounter.iLoadVal	byte	0		'\00'	0x00	0000 0000
PCounter.s_ExecCmd_PC	s_ExecCmd...	0		'\00'	0x00	0000 0000
PCounter.oVal	byte	0		'\00'	0x00	0000 0000

- 3) Click on F8 to single step and observe the watch window changes, specially the PC – Program Counter.