

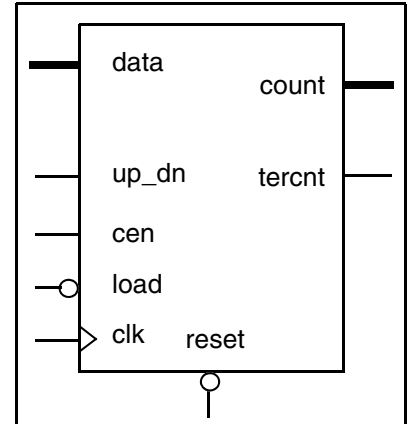
## DW03\_bictr\_scnto

### Up/Down Binary Counter with Static Count-to Flag

Version, STAR and Download Information: [IP Directory](#)

## Features and Benefits

- Parameterized word length
- Parameterized count-to value
- Up/down count control
- Asynchronous reset
- Loadable count register
- Terminal count flag
- Counter enable
- Provides minPower benefits with the DesignWare-LP license ([Get the minPower version of this datasheet.](#))



## Description

DW03\_bictr\_scnto is a general-purpose up/down counter with static count-to logic.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
data	<i>width</i>	Input	Counter load input
up_dn	1 bit	Input	High for count up and low for count down
load	1 bit	Input	Enable data load to counter, active low
cen	1 bit	Input	Count enable, active high
clk	1 bit	Input	Clock
reset	1 bit	Input	Counter reset, active low
count	<i>width</i>	Output	Output count bus
tercnt	1 bit	Output	Terminal count flag

Table 1-2 Parameter Description

Parameter	Values	Description
width	1 to 30	Width of data and count
count_to	1 to $2^{width-1}$	Count-to value

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW03_BICTR_SCNTO_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW03_bictr_scnto_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW03_bictr_scnto.v	Verilog simulation model source code

Table 1-5 Counter Operation Truth Table

reset	load	cen	up_dn	Operation
0	X	X	X	Reset
1	0	X	X	Load
1	1	0	X	Standby
1	1	1	0	Count down
1	1	1	1	Count up

When the count value equals `count_to`, the signal `tercnt` (terminal count) is asserted (HIGH). The signal `tercnt` can be connected to `load` through an inversion to synchronously reset the counter to a predefined value on the data in `bus_data`.

The counter is *width* bits wide and has  $2^{width}$  states from “000...0” to “111...1”. The counter is clocked on the positive edge of `clk`.

The `reset`, active low, provides for an asynchronous reset of the counter to “000...0”. If the `reset` pin is connected to ‘1’, the reset logic is not synthesized, resulting in a smaller and faster counter.

The `count_to` is a parameter that ranges from 1 to  $2^{width-1}$ . When the counter output, `count`, equals `count_to`, `tercnt` goes HIGH for one clock cycle.

The `up_dn` input controls whether the counter counts up (`up_dn` HIGH) or down (`up_dn` LOW), starting on the next positive edge of `clk`.

The counter is loaded with data by applying data to `data` and asserting `load` (LOW). The data load operation is synchronous with respect to the positive edge of `clk`.

When the count enable pin, `cen`, is HIGH, the counter is active. When `cen` is LOW, the counter is disabled and `count` remains at the same value.

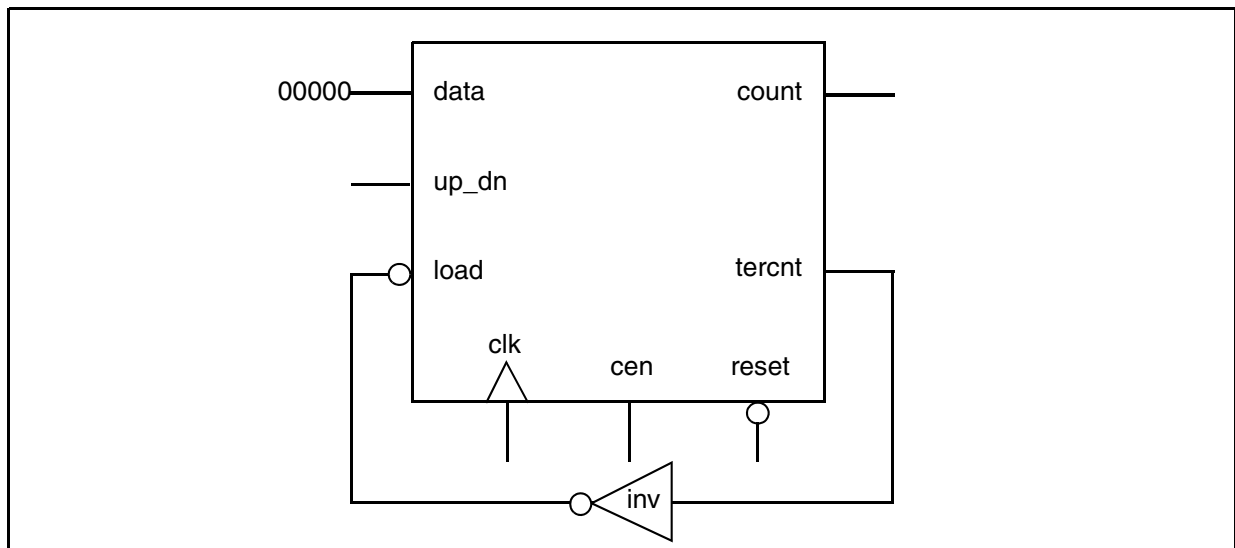
## Application Example

An example application of DW03\_bictr\_scnto is to count from 0 to 17, repeatedly. This is done by:

1. Connecting the terminal count output signal, `tercnt`, to the `load` input.
2. Setting the data input to the start of the count sequence (for example, “00000”).
3. Setting the `count_to` parameter to the end of the count sequence (for example, 17).

The `count` output then cycles through the states 00000 (`data`) to 10001 (`count_to`). Refer to [Figure 1-1](#).

**Figure 1-1 Counter Application: “count\_to”**



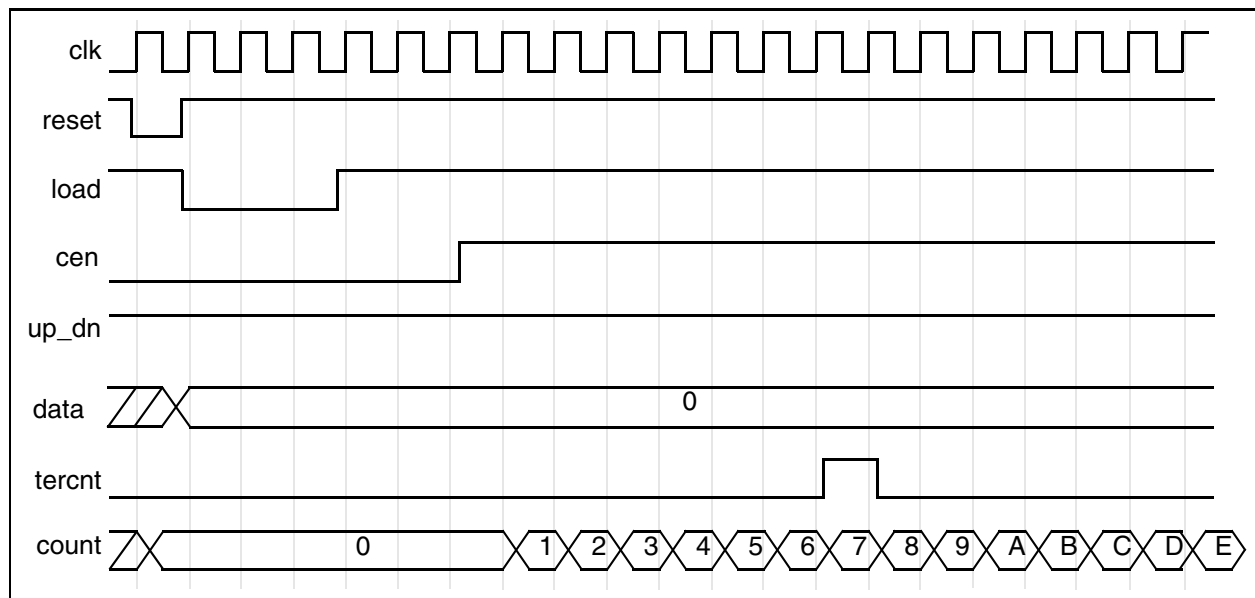
## Low Power Implementation

This component provides low power (`minPower`) benefits when the “`lpwr`” implementation is chosen and the DesignWare-LP license is available. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command.

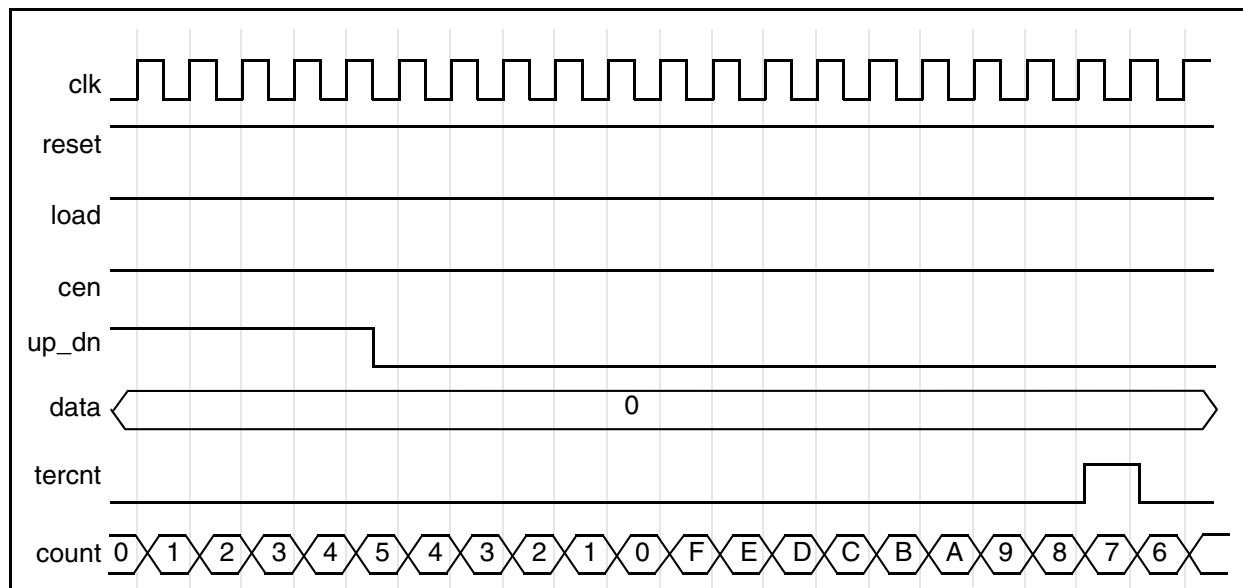
## Timing Diagrams

Figure 1-2 and Figure 1-3 show various timing diagrams for DW03\_bictr\_scnto.

**Figure 1-2 Functional Operation: Count Up with tercnt = 7**



**Figure 1-3 Functional Operation: Count Up with tercnt = 7**



## Related Topics

- [Logic – Sequential Overview](#)
- [DesignWare Building Block IP Documentation Overview](#)

## HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW03_bictr_scnto_inst is
  generic (inst_width      : POSITIVE := 8;
           inst_count_to : POSITIVE := 8);
  port (inst_data      : in std_logic_vector(inst_width-1 downto 0);
        inst_up_dn    : in std_logic;
        inst_load      : in std_logic;
        inst_cen       : in std_logic;
        inst_clk       : in std_logic;
        inst_reset     : in std_logic;
        count_inst     : out std_logic_vector(inst_width-1 downto 0);
        tercnt_inst    : out std_logic);
end DW03_bictr_scnto_inst;

architecture inst of DW03_bictr_scnto_inst is
begin

  -- Instance of DW03_bictr_scnto
  U1 : DW03_bictr_scnto
    generic map ( width => inst_width, count_to => inst_count_to )
    port map ( data => inst_data, up_dn => inst_up_dn, load => inst_load,
              cen => inst_cen, clk => inst_clk, reset => inst_reset,
              count => count_inst, tercnt => tercnt_inst );
end inst;

-- pragma translate_off
configuration DW03_bictr_scnto_inst_cfg_inst of DW03_bictr_scnto_inst is
  for inst
    end for; -- inst
end DW03_bictr_scnto_inst_cfg_inst;
-- pragma translate_on
```

## HDL Usage Through Component Instantiation - Verilog

```
module DW03_bictr_scnto_inst( inst_data, inst_up_dn, inst_load,
                             inst_cen, inst_clk, inst_reset,
                             count_inst, tercnt_inst );

    parameter width = 8;
    parameter count_to = 8;

    input [width-1 : 0] inst_data;
    input inst_up_dn;
    input inst_load;
    input inst_cen;
    input inst_clk;
    input inst_reset;
    output [width-1 : 0] count_inst;
    output tercnt_inst;

    // Instance of DW03_bictr_scnto
    DW03_bictr_scnto #(width, count_to)
        U1 ( .data(inst_data), .up_dn(inst_up_dn), .load(inst_load),
            .cen(inst_cen), .clk(inst_clk), .reset(inst_reset),
            .count(count_inst), .tercnt(tercnt_inst) );

endmodule
```

## Copyright Notice and Proprietary Information

© 2018 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.  
690 E. Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://www.synopsys.com)

