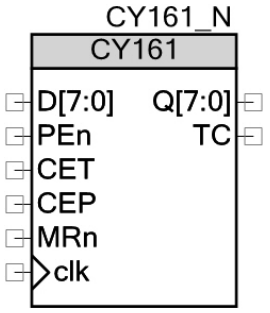
8-bit Binary Counter: Asynchronous Reset

CY161 v1.0



Features

- Synchronous counting and loading
- Positive-edge triggered clock
- Asynchronous reset
- Selectable load value.
- Simple to deconstruct

General Description

This counter is based on the 74HC161. It is a synchronous pre-settable binary counter with look-ahead carry. Synchronous operation is implemented with positive-edge triggered logic. The counter, Q[7:0] (datapath register A0), may be preset to a load value at inputs D[7:0]. A LOW level at the master reset input (MRn) sets the counter to zero, providing an asynchronous clear.

The look-ahead carry simplifies serial cascading of the counters. Both count enable inputs (CEP and CET) must be HIGH to count. The CET input is fed forward to enable the terminal count output (TC). The TC output, when enabled, will produce a HIGH output pulse of duration approximately equal to clock cycle.

This component was built as a teaching tool. This classic component's operation is well understood and this datasheet's function is to help understand how the component was built through its deconstruction.

Pin Description

Pin	Type	Function
D[7:0]	inputs	value to be loaded
PEn	input	load enable input (active low)
CET	input	count enable carry input
CEP	input	count enable input
MRn	input	asynchronous master reset (active low)
clk	input	clock input

Pin	Type	Function
Q[7:0]	outputs	counter value
TC	output	terminal count output

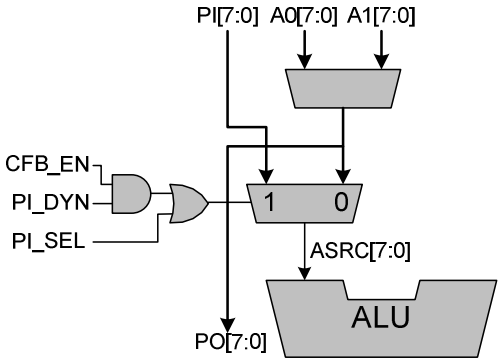
Function Table

Operating Modes	Inputs					Outputs	
	MRn	clk	CEP	CET	PEn	Q	TC
reset	0	x	x	x	x	0x00	0
load	1	↑	x	x	0	D	Q == 0xff && CET ==1
count	1	↑	1	1	1	Q + 1	Q ==0xff
hold	1	x	0	x	1	Q	Q == 0xff && CET ==1
hold	1	x	x	0	1	Q	0

Deconstructing the Component

This component is a good example of how to route parallel data into the ALU and out of the A0/A1 registers. With three synchronous inputs, it easily fits into the eight available datapath configurations (instructions).

Below is simplified block diagram of the parallel In/Out interface to a datapath.



The Parallel Out (PO) interface is always available and connects to either A0 or A1. The parallel in (PI) interface must be routed into the left input of the ALU (ASRC). If the PI_SEL bit (CFG15.7) is set, the PI is the permanent input to ASRC. However if instead the PI_DYN bit (CFG17.5) is set, the CFB_EN bit in each datapath instruction determines the ASRC source for that particular instruction. This bit shares function with the single instruction CRC functionality. If PI_DYN is set, CFB_EN serves to select a parallel input to the ALU. If not set, CFB_EN determines if a CRC operation is performed.

For this counter, there are four different operations:

- Reset
- Hold
- Load
- Count Up

While the reset operation will be handled with datapath reset circuitry, the other three operations need to be implemented with datapath instructions.

Using PSoC Creator, open the CY161 example project to see the project schematic (*TopDesign.cysch*). It has a CY161 component connected to input switches, output LEDs, and a clock.

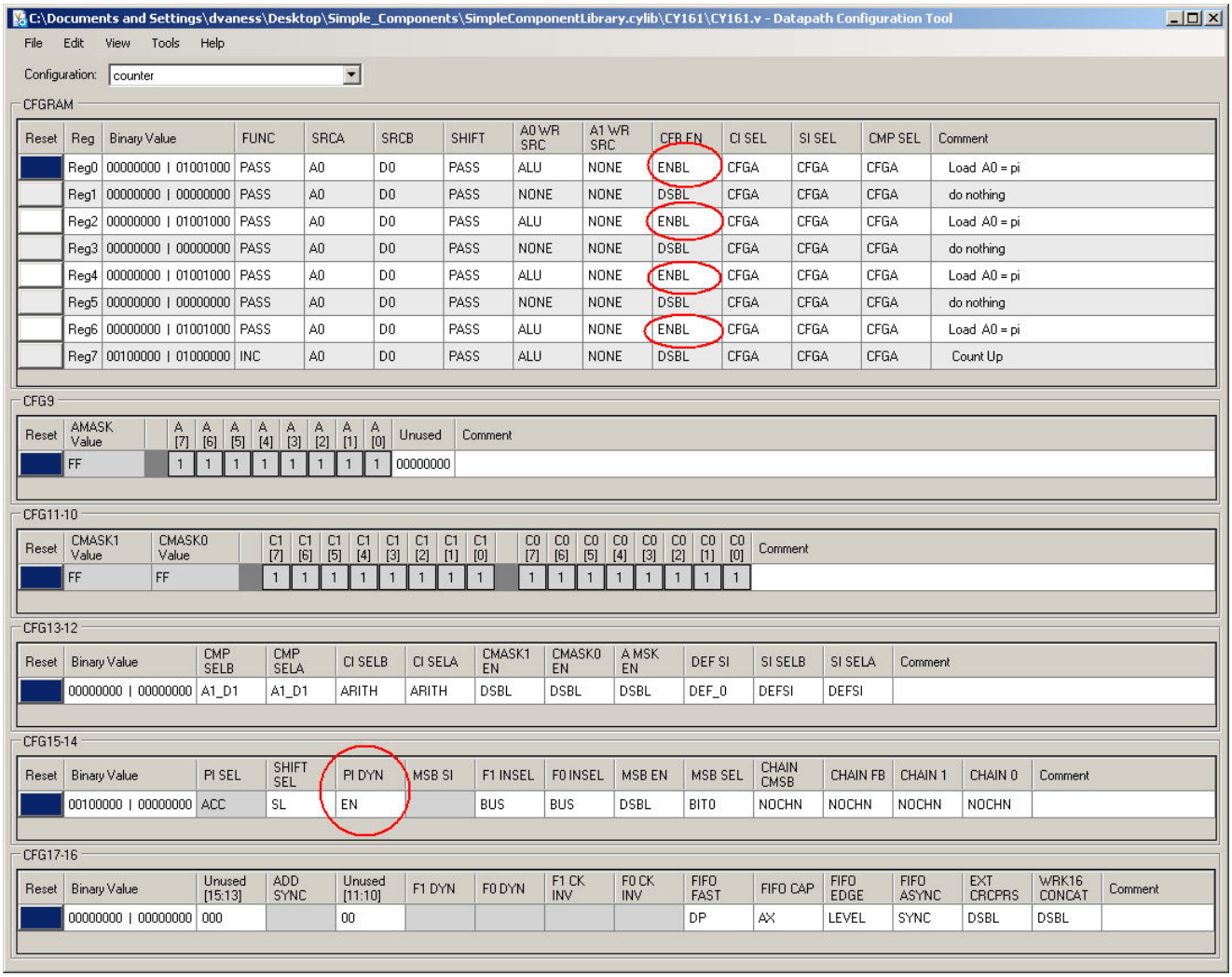
In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComps library project is, and select the CYCC_CY161_v1_0 component. Click **OK** and the following files are shown for the component:

- Symbol file (cysym)
- Datasheet (pdf)
- Verilog File (v)

Open the symbol file to find a symbol with six inputs and two outputs. It looks like the symbol shown on the first page. There are no additional symbol parameters.

Open Verilog file and notice that at lines 30 – 39, these definitions were passed to this Verilog file when it was created. The first 25 lines of this header list register usage and the datapath instruction definitions.

There is a need for intermediate signals and this is handled in lines 42 - 45. What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the Datapath Configuration Tool for this Verilog file results in the following.



Note that although only three operations are needed to be implemented, all 8 of the datapath instructions are used. By judicious selection of their address position and duplication, the logic to control them has been greatly simplified. The PI_DYN bit has been enabled so the CFB_EN bit of each instruction controls the ASRC input to the ALU.

- For the load instructions the parallel input is passed through the ALU and placed in A0.
- For the null instructions A0 is passed through the ALU but not fed to any register. It does nothing.
- For the count instruction A0 is incremented by the ALU and fed back to A0.

When saved, this tool inserts the updated configuration data back into the Verilog file along with an instance of the datapath interface. Just pass the correct signals in to and out of it and you are done.

This counter parameter list may look ominous, but only a few parameters have to be entered as shown:

```
/*  input               */  .reset(reset),          // Reset for datapath
/*  input               */  .clk(clk),              // Clock for Datapath
/*  input   [02:00]     */  .cs_addr({CEP,CET,PEn}),// Control for reg selection
/*  output              */  .ff0(terminalCount),    // TC from datapath
/*  input [07:00]       */  .pi(D[7:0]),            // Parallel input data port
/*  output [07:00]      */  .po(Q[7:0])             // Parallel output data port
```

These parameters do the following:

- Connect a reset signal to the datapath reset circuitry
- Connect a clock signal to the datapath clock circuitry
- Provide three inputs to control the instruction processing
- Connect the “all 1s” flag to an output
- Connect the parallel input to the datapath PI
- Connect the datapath PO to the parallel output

Support

PSoC Creator Community Components are developed and supported by the Cypress Developer Community. Go to www.cypress.com/CommunityComponents to discuss this and other Community Components.

© Cypress Semiconductor Corporation, 2013. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control, or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This work is licensed under a Creative Commons Attribution 3.0 Unported License.

http://creativecommons.org/licenses/by/3.0/deed.en_US

You are free to:

- Share — to copy, distribute and transmit the work
- Remix — to adapt the work
- Make commercial use of the work