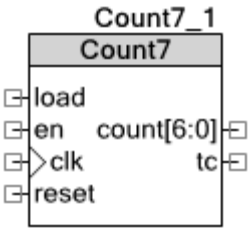
Count7
1.0

Features

- Efficient hardware implementation
- 7-bit down counter



General Description

Component uses the built-in count7 hardware available in every UDB to implement a 7-bit down counter.

When to use a Count7

Use when the requirements of a counter fit within 7-bits and a down counter only implementation. This component is designed to use minimal resources. It is more resource efficient than the alternative datapath or macrocell based implementations.

Input/Output Connections

This section describes the various input and output connections for Count7. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

clk – Input

All functionality is implemented synchronous to this clock except for reset.

reset – Input

Asynchronous reset. Resets counter value to 0.

load – Input *

Level sensitive load signal. When sampled high and the counter is enabled, the period value is loaded into the counter register. If the counter is not enabled, this signal has no effect. The presence of this signal is controlled by configuring the parameters for the component instance.

en – Input *

Level sensitive enable signal. When sampled high and the software enable is also set, the counter is enabled and will count down. Both the software enable and the hardware enable (when present) must be active for the counter to count or for the load to take effect. The presence of this signal is controlled by configuring the parameters for the component instance.

count[6:0] – Output

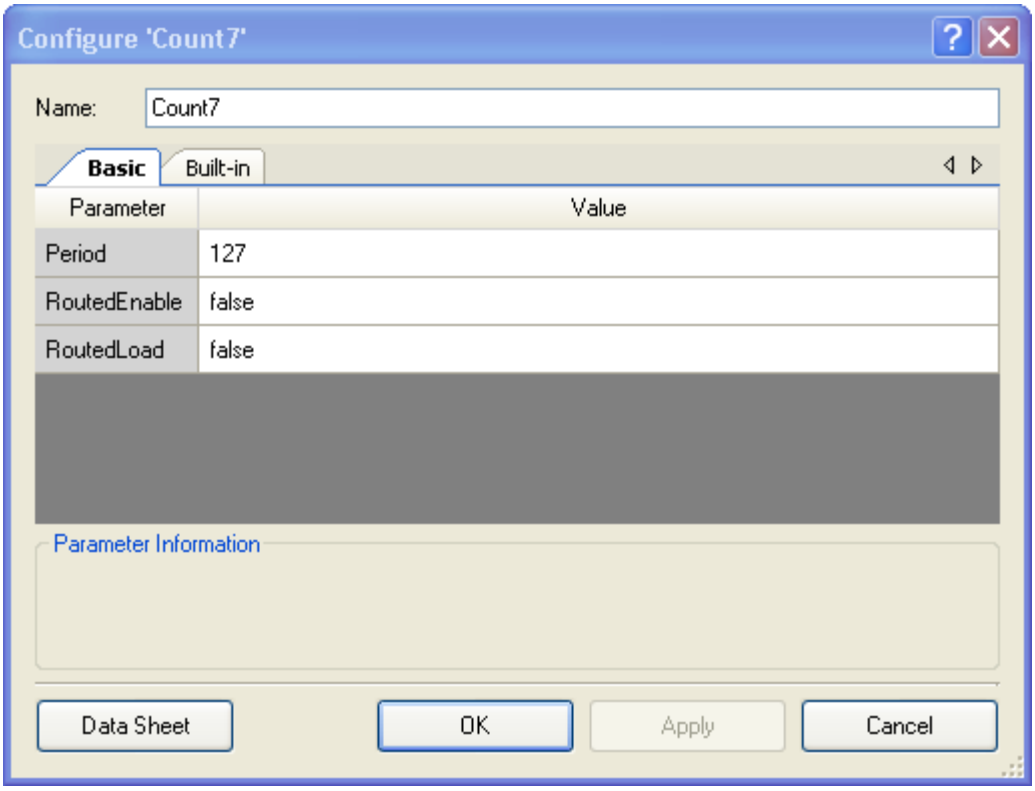
7-bit counter value.

tc – Output

Registered terminal count value. Calculated when the counter transitions to 0 and the registered value is output on the next cycle. Requires a counting transition to 0, so it is not triggered based on power up reset, the reset signal or a load.

Parameters and Setup

Drag a Count7 component onto your design and double-click it to open the Configure dialog.



Period

Value to be placed into the Period register of the Count7. The counter will transition through the values from Period to 0 inclusive for a total of (Period + 1) total values in a repeating sequence. Value must be in the range from 1 to 127 inclusive.

RoutedEnable

Determines whether a hardware enable signal is present on the component. When set to false the counter is enabled exclusively by a software enable. When set to true the counter is enabled when both the hardware and software enables are active.

RoutedLoad

Determines whether a hardware load signal is present on the component. When set to false the counter is reloaded only when the count reaches the terminal count of 0. When set to true the counter is also reloaded if the load signal is high and the counter is enabled.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name “Count7_1" to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "Count7".

Function	Description
void Count7_Start(void)	Starts (Initializes and Enables) the Counter
void Count7_Stop(void)	Stops the Counter
void Count7_Init(void)	Initializes
void Count7_Enable(void)	Sets the software enable
void Count7_Disable(void)	Clears the software enable (same function as Count7_Stop())
uint8 Count7_ReadCounter(void)	Reads the current counter value
void Count7_WriteCounter(uint8 count)	Writes the current counter value
uint8 Count7_ReadPeriod(void)	Reads the Period register
void Count7_WritePeriod(uint8 period)	Writes the Period register

void Count7_Sleep(void)	Saves configuration and disables
void Count7_Wakeup(void)	Restores configuration and enable state
void Count7_SaveConfig(void)	Saves the configuration
void Count7_RestoreConfig(void)	Restores the configuration

void Count7_Start(void)

Description: Starts the Count7. Initializes the Period value.

Parameters: None

Return Value: None

Side Effects: None

void Count7_Stop(void)

Description: Stops the Count7

Parameters: None

Return Value: None

Side Effects: None

void Count7_Init(void)

Description: Initializes the component.

Parameters: None

Return Value: None

Side Effects: None

void Count7_Enable(void)

Description: Sets the software enable.

Parameters: None

Return Value: None

Side Effects: None

void Count7_Disable(void)

Description: Clears the software enable. Has the same functionality as Count7_Stop().

Parameters: None

Return Value: None

Side Effects: None

uint8 Count7_ReadCounter(void)

Description: Reads the current value of the counter.

Parameters: None

Return Value: (uint8) counter value

Side Effects: None

void Count7_WriteCounter(uint8 count)

Description: Writes the current value of the counter.

Parameters: (uint8) value to load into the counter register

Return Value: None

Side Effects: None

uint8 Count7_ReadPeriod(void)

Description: Reads the Period register

Parameters: None

Return Value: (uint8) period value

Side Effects: None

void Count7_WritePeriod(uint8 period)

Description: Writes the Period register

Parameters: (uint8) value to load into the Period register

Return Value: None

Side Effects: None

void Count7_Sleep(void)

Description: Saves the configuration and non-rentention register values. Disables the counter.

Parameters: None

Return Value: None

Side Effects: None

void Count7_Wakeup(void)

Description: Restores the configuration and non-retention register values including the enable state.

Parameters: None

Return Value: None

Side Effects: None

void Count7_SaveConfig(void)

Description: Saves the user configuration of non-rentention registers. Called by the Count7_Sleep routine to save the component state before entering sleep.

Parameters: None

Return Value: None

Side Effects: None

void Count7_RestoreConfig(void)

Description: Restores the user configuration of non-rentention registers. Called by Count7_Wakeup routine to restore the component state after returning from sleep.

Parameters: None

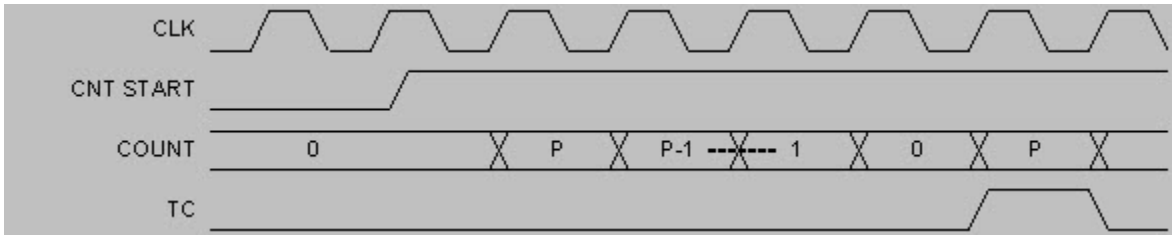
Return Value: None

Side Effects: None

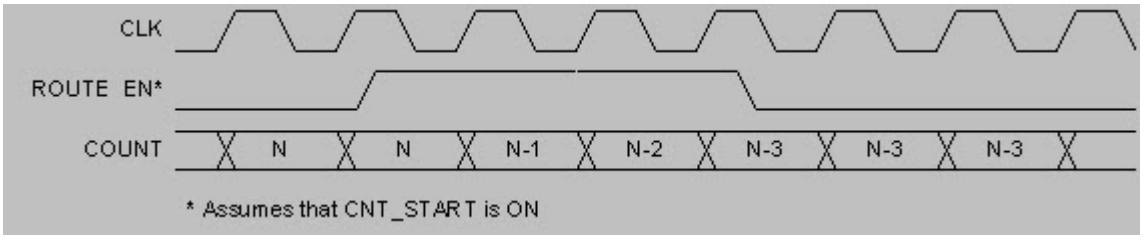
Functional Description

This component is implemented directly by the Count7 hardware implementation in each UDB. The Count7 hardware is present in each UDB, but it uses resources that are shared with the Control and Status registers. The Count7 uses the Control register resources and the Mask register from the Status register. If a Count7 is used in a UDB, then the Control register is not available and the interrupt capability of the Status register is not available. If the routed enable or load options of the Count7 are used, then none of the Status register functionality is available.

The basic functionality of the Count7 without routed enable or load signals is shown in the diagram below. The “CNT START” signal is the software enable signal controlled by the APIs. The counter starts at 0 and once enabled loads the period value and counts down to 0. Each time it reaches 0 the period value is reloaded. The terminal count is active one cycle after the count has reached the terminal value of 0.



When the routed enable signal is present the counter only counts when the routed enable is high and the software enable is also set.



When the routed load signal is present the counter loads the period value if the counter is enabled. If the counter is not enabled the load signal is ignored.