The 4029 CMOS Counter IC

Introduction

One of the most common requirements in digital equipment is *counting*. And the most common counting requirement has to do with time. From a basic digital clock (which is incorporated into most digitally-controlled appliances) to interval timers and event counters, the need for counting circuits is very great.

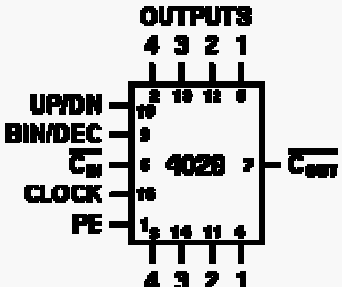
Counters are required for various counting ranges and in all sorts of circumstances. A simple digital clock, for example, requires a decimal counter for the units positions of seconds and minutes, but must count only from 0 to 5 (modulo-6) for the tens positions of minutes and seconds. Hours digits require special handling, since the two-digit output must count either from 1 to 12 or 0 to 23, depending on how we want the display to appear. Some clocks skip the 0 to 23 military display and run from 1 to 24 in that mode. And some simple clocks make no effort at that, or to distinguish between AM and PM.

No matter how the hours are handled, the basic element required to make any digital clock work is the counter circuit. Indeed, counters are so important in so many applications that many different kinds of counter ICs have been designed for both TTL and CMOS logic families. Some count up for clocks and time intervals; others count down to show time remaining until some event. Some are specifically designed to count in decimal mode, while others count in binary, and still others have selectable counting ranges. The list of capabilities and options is quite large, leaving the circuit designer with only the task of selecting the particular IC that best suits the need.

In this experiment, we will explore the capabilities and operation of the type 4029 CMOS counter IC, which can count up or down, and in either decimal or binary mode. It can also be preset with any initial count, and is designed so that multiple 4029 ICs can be cascaded and still maintain fully synchronous counting either up or down.

Functional Diagram

The internal schematic diagram of the 4029 is quite complex, as it involves a gating structure that will allow counting either up or down, in binary or decimal, and with parallel inputs to preset a count. However, we can't reach the internal circuitry in any case, so the functional diagram to the right is far more useful for our purposes.



Most of the designations are straightforward and intuitive. The **up/dn** input, for example, tells the counter to count up if it is a logic 1, or down when it is logic 0. In the same way, a logic 1 to the **bin/dec** input causes

the counter to operate in binary mode, while a logic 0 switches it to decimal (sometimes called *decade*) mode. If multiple 4029s are cascaded for a larger count, all **up/dn** pins are connected together and driven from a common signal, as are all **bin/dec** lines.

The $\mathbf{C_{IN}}$ and $\mathbf{C_{OUT}}$ lines form the means of cascading counters and still keeping a fully synchronous count. If $\mathbf{C_{IN}}$ is logic 1, the counter won't count at all. When $\mathbf{C_{IN}}$ goes to logic 0, the counter operates normally. Then, when the counter reaches its terminal count (9, 15, or 0 depending on the states of **up/dn** and **bin/dec**), $\mathbf{C_{OUT}}$ goes to logic 0. This is connected to the $\mathbf{C_{IN}}$ line of the next IC to allow the next higher order of magnitude to count once. Then $\mathbf{C_{OUT}}$ goes to logic 1 again. The first counter IC in the set, representing the least significant digit, has its $\mathbf{C_{IN}}$ line grounded to logic 0 so it will always count.

The **clock** input, like **up/dn** and **bin/dec**, is fed to all 4029s in an extended counter circuit. This is the signal that represents whatever is to be counted. Any 4029 that is enabled by having its $\mathbf{C_{IN}}$ line at logic 0 will change state to the next count when the **clock** rises from logic 0 to logic 1. Any changes to the $\mathbf{C_{IN}}$ and $\mathbf{C_{OUT}}$ lines will occur just after that rising edge, and so will be ready for the next **clock** pulse.

The **pe** input is the **preset enable** line, and is also shared with all 4029s in the counting set. When this line is logic 0, the counter operates normally. However, when **pe** becomes logic 1, the logic signals present on the four **jam** input lines get copied directly to the four bits of the counter, overriding any prior count. These inputs are not always used, and many applications ignore them. However, there are some applications that do use these inputs to advantage. For example, if we want to count for a multi-digit interval in seconds, we can set the **jam** inputs to the decimal number of seconds to be counted and configure the circuit to count down. When the counter reaches zero, the final $\mathbf{C_{OUT}}$ line falls to logic 0 and can be used after inversion to preset the count again to the selected number, or to stop the count and signal the end of the timing interval.

The four outputs, of course, are the current count as either a binary or BCD number, depending on the state of **bin/dec**. Output **1**, often designated **Q1**, is the Least Significant Bit (LSB), while output **4** (or **Q4**) is the Most Significant Bit (MSB).

All inputs and outputs are standard CMOS design, and operate in the normal manner for CMOS ICs. The 4029 is specified to operate accurately for clock signals up to 2 MHz minumum. and typically 4 MHz, with a 5 volt power supply. It will run faster with higher power supply voltages.

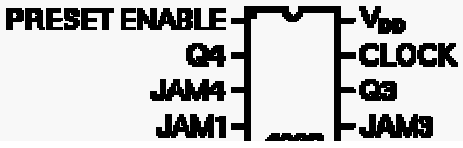
Parts List

To construct and test the 4029 counter IC on your breadboard, you will need your breadboarding system, assorted colors of hookup wire, and the following experimental part:

- (1) 4029 CMOS counter IC.

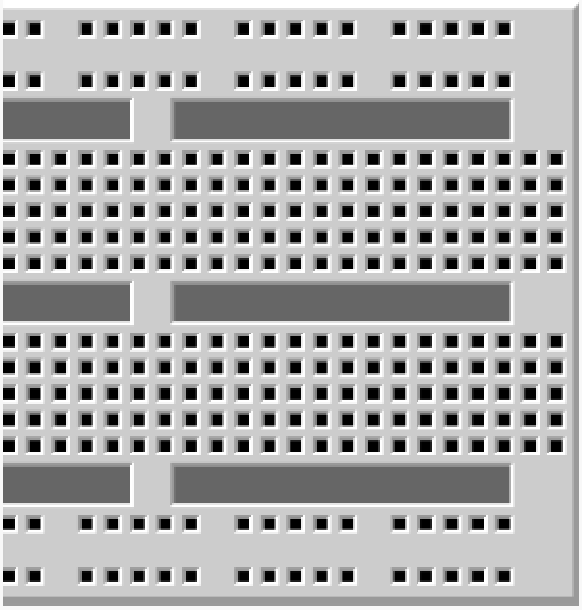
Constructing the Circuit

Select an area on your breadboard socket that is clear of other circuits. You'll need two adjacent sets of five bus contacts for this project. As you construct this project, you may find it helpful to refer to the pin configuration diagram to



the right. When you are ready, refer to the image and text below and install the IC and jumpers as shown.

Circuit Assembly



START

Performing the Experiment

Set all logic switches to logic 0, and then turn on power to your experimental circuit.

Step 1. Note the current state of the LED display. Record this value, as four binary bits (for example, 0010), in the text box in the top row of the table to the right.

Step 2. Press pushbutton B once. What effect does this have on your experimental circuit? Press pushbutton B several more times. Does the display change in any way as a result?

Step 3. Set logic switches S6 and S7 to logic 1. These are connected to the **up/dn** and **bin/dec** inputs of your experimental IC. Press pushbutton B once to initialize the 4029 IC, and record the displayed binary value in the text box for Count 0 underneath the heading for Steps 3-4 in the table to the right.

Initial State at Power On:				
Step:	**3-4**	**5**	**6**	**7**
Count	**Up; Bin**	**Dn; Bin**	**Up; Dec**	**Dn; Dec**
0				
1				
2				
3				
4				
5				

Step 4. Press pushbutton A once, and note the effect on the displayed output. Enter this value in the table to the right, immediately under your entry for Step 3. Continue to press pushbutton A, one pulse at a time, and record all results on successive rows of the table, until you have filled in this column of entries.

NOTE: If you find that the counter constantly counts two or three pulses each time you press pushbutton A and you cannot prevent this even by exercising care, it may be that your particular combination of component tolerances is failing to ovecome contact bounce. In that case, increase the value of the capacitors used in the pushbutton circuits, to increase the pulse width of each pulse. The nominal pulse width with the .01µf capacitors initially specified is about 0.07 second. However, if your particular 4049 has a very low switching threshhold, it could be much shorter. You can increase this interval by increasing the capacitance value to something in the range of .02µf to.05µf. You should select the lowest value you can and still eliminate multiple pulses.

Step 5. Set logic switch S6 to logic 0, and press pushbutton B once to initialize the 4029 IC as before. Record the displayed value as before in the row for Count 0, under the Step 5 heading. Then, as before, press pushbutton A once and record the resulting displayed value in the next row of the table. Continue pressing pushbutton A and recording your results until you have filled in the entire Step 5 column of the table.

Compare your results from this step with the results you recorded from Steps 3 and 4. What's different this time?

Step 6. Set S6 to logic 1 and S7 to logic 0. Press pushbutton B to reset the counter to its initial count again, and record this value in the top cell of the column for Step 6, to the right. Then, as before, press pushbutton A repeatedly, while filling in the Step 6 column of the table.

Compare your results with those you have recorded in earlier steps. What has changed between this step and Steps 3-4?

Step 7. Set S6 to logic 0, leaving S7 at logic 0. As in prior steps, first press

6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

pushbutton B, the repeatedly press pushbutton A, while recording the observed LED display for each input pulse, in the last column of the table to the right. When you have completed the table, again compare your results with those from prior steps. At this point, can you tell what effect S6 has on the 4029 IC? How about S7?

Step 8. Set S6 and S7 to logic 1, and once more press pushbutton B. Verify the result on the LED displays. Now, set S3 and S1 to logic 1, so that logic switches 3, 2, 1, and 0 are set to a binary pattern of 1010. Press pushbutton B again, and note the result on your LEDs.

Repeat this step with various binary patterns for S3 through S0. Each time you press pushbutton B, how do the four LEDs compare with the settings of these four switches? What happens when you press pushbutton A after releasing pushbutton B?

Step 9. Set S7 to logic 0 and repeat Step 8. What happens now with various settings for S3 through S0? Especially, set S3 through S0 to each of patterns 1010, 1011, 1100, 1101, 1110, and 1111 in turn, and observe the LEDs as you press pushbutton B once, and then pushbutton A several times, for each of those patterns.

When you have completed all steps of this experiment, turn off the power to your experimental circuit and compare your results with the discussion below.

Discussion

In Step 1, the LEDs displayed a random pattern when you first turned power on. This has no special meaning; the four flip-flops inside the 4029 may each power up in either state, so any binary display was possible. However, that display was only momentary. Because your two pushbutton pulse generator circuits power up generating a pulse, the 4029 simultaneously received a single clock pulse (from A) and a preset enable signal (from B). With S3 through S0 (the Jam inputs) all set to logic 0, the preset enable pulse effectively reset the 4029 to a count of 0000. This happens so quickly that you might not be able to see the change on the LEDs. If so, you can turn off power and remove the connection from pin 1 to pushbutton B. Instead, simply ground pin 1, and then turn power back on. The 4029 will power up in its initial random state.

This is something to be aware of when using these ICs, and indeed most digital counter ICs: they power up in an unknown state and must be initialized to a known state in many applications. Otherwise they can present a false count.

In Step 2, when you pressed pushbutton B deliberately, nothing changed. All four LEDs remained

off, since the IC is simply being reset multiple times in this step.

In Steps 3 and 4, you first set S6 and S7 to logic 1. This set the 4029 IC to count up (S6) in binary mode (S7). Pressing pushbutton B once again had no effect; the 4029 retained its count of 0000. However, each time you pressed pushbutton A, the LED display incremented by 1 until, at Count 15, it was 1111. The sixteenth count saw the IC "wrap around" to a count of 0000 again.

In Step 5, you set S6 to logic 0. This caused the 4029 to count down, or backwards. The counting sequence was still binary, but it was exactly the reverse of the sequence you recorded in Step 4.

Beginning with Step 6, you switched the 4029 to decimal counting mode by setting S7 to logic 0. You also set S6 back to logic 1, so for this step the 4029 counted up from 0 (0000) to 9 (1001), then wrapped around to 0000 again for Count 10, and continued the counting sequence from there.

In Step 7, you set S6 to 0 to count down in decimal mode. The IC operated as you would expect, with the count wrapping smoothly from 0000 to 1001 as it counted below zero and therefore rolled to 9 again. Thus, this IC does indeed count accurately up or down, in either binary or decimal mode.

In Steps 8 and 9, you made use of the external inputs and the preset enable control line. Here, you found that pressing pushbutton B immediately copied the settings of S3 through S0 into the 4029 IC, which would then begin counting, either up or down, from that count. The only exception occurs when the IC is counting up in decimal mode. In that case, presetting the counter to a count higher than 9 (1001) caused the counter to follow an odd sequence, until it dropped into the legitimate decimal range. Once that occurred, counting proceeded normally from there.

In binary mode, or counting down in decimal mode, the 4029 accepted any input value and counted in sequence from there, but remained within the selected range thereafter. These characteristics allow the 4029 to operate smoothly and well in many applications, either by itself or cascaded with more ICs of the same type. We'll explore a number of practical applications in future experiments.

When you have completed this experiment, make sure power to your experimental circuit is turned off. Remove all jumpers and the IC, and put them aside for future experiments.

Note: If you have an application that requires only decimal or only binary counting, but also requires a reset input as well as preset capability, you may want to consider the 4510 and 4516 ICs. These two ICs are pin compatible with the 4029, except that the **bin/hex** input (pin 9) has been replaced with a **reset** input. The 4510 will count either up or down in decimal mode, while the 4516 will count up or down in binary mode. In all other respects, the 4510 and 4516 behave exactly like the 4029, and will handle the same range of frequencies.

Prev: Basic Clock Sources **Next: The Bicolor LED**