

# Visual Guide to 65xx CPU Timing

LAUGHTON ELECTRONICS

NAVIGATE

[Projects](#)

[Servicing the Unserviceable](#)

[Main index](#) —> [for registerheads only](#)

**CPU timing specifications** can be hard to grasp. Often there's just a single timing diagram provided — and such a diagram is necessarily very general, as it's drawn to accommodate umpteen combinations of operating conditions. This article looks at the information that's supplied, and shows how to relate those timing specs to your own project's operating conditions.

Mostly I'll be looking at just a few key signals and how they're specified. Really the subject is **how to read a timing diagram** and the **conventions** that CPU data sheets commonly use. In other words it's NOT an attempt to explain every little detail. It's more about the basics of spec-speak, and getting a feel for why the doc is written the way it is.

Comments and questions regarding this article are hosted on the 6502.org forum. You can read or add to the discussion [here](#).

## Starting At the Beginning: the Clock signal

GIF01

Table 4-2 W65C816S AC Characteristics

Symbol	Parameter	5.0 +/- 5%		3.3
		Min	Max	Min
VDD		4.75	5.25	3.0
tCYC	Cycle Time	70	DC	12
tPWL	Clock Pulse Width Low	35	-	6
tPWH	Clock Pulse Width High	35	-	6
tF,tR	Fall Time, Rise Time	-	5	-
tAH	A0-A15 Hold Time	10	-	1
tADS	A0-A15 Setup Time	-	30	-

The two columns below pertain to operation at 5 volts

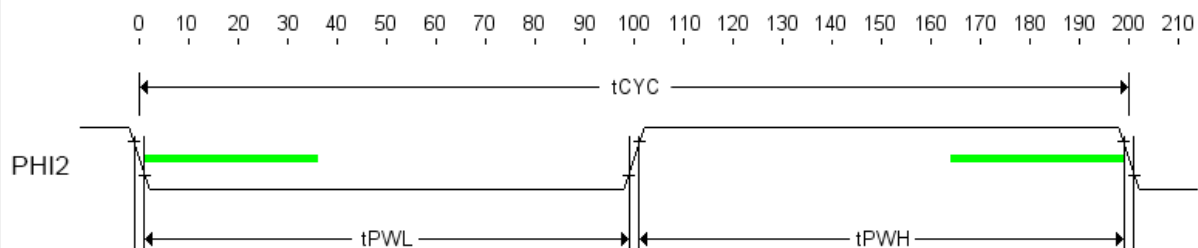
At 5 volts, input signals such as Clock may be rapid, and outputs appear with minimal delay.

Above is an excerpt from Table 4-2 in WDC's [W65C816S data sheet \(Sept 2010 edition\)](#). Each row shows a symbol, a parameter and a series of numeric values. The symbols' meanings are visually depicted in a diagram elsewhere in the data sheet (Figure 4-1).

The two columns under **5.0 +/- 5%** show what we can expect when operating the '816 at 5 volts, and we see that at 5 volts the chip is comparatively fast. **tPWL** and **tPWH** show it is guaranteed to deal with clock-low and clock-high durations as short as 35 ns (minimum).

What we will see is that **these two specs determine the limits of other variables** such as duty cycle — *and* Cycle Time (the reciprocal of operating frequency). In other words the operating frequency (such 14 Mhz) is merely a dependent variable. The dog wags the tail — not the other way 'round! The limits for **tPWL** and **tPWH** are what determine the limits of the other variables.

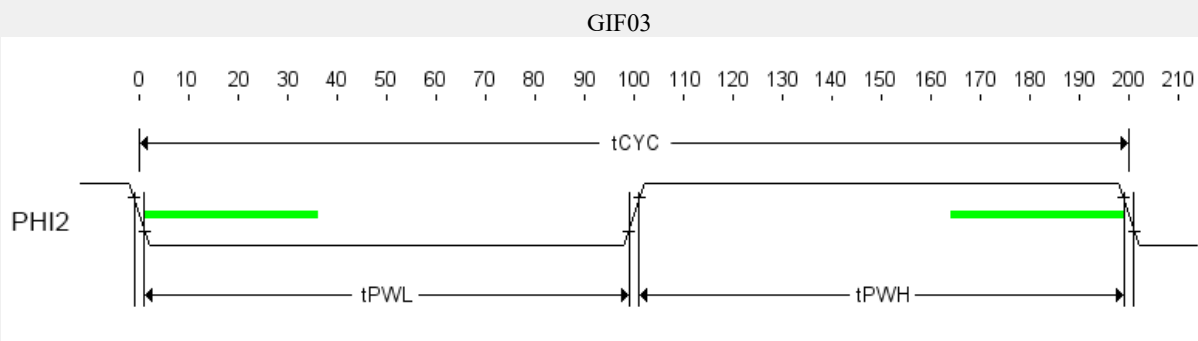
GIF02



Here I've redrawn part of WDC's timing diagram, but with one important difference: **all my diagrams in this topic are drawn to scale**. The graduations along the top are in nanoseconds, and values such as  $t_{PWL}$  and  $t_{PWH}$  can be read directly. It's like viewing the wave on an oscilloscope. Datasheet timing diagrams aren't like that.

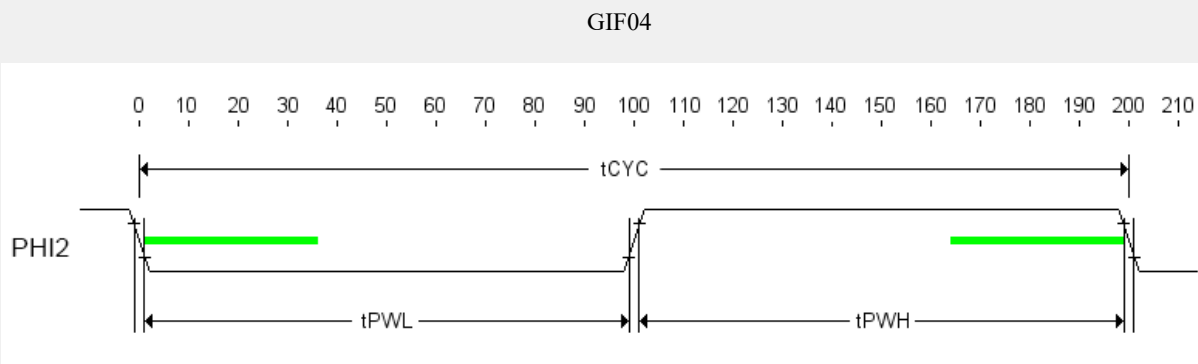
I added the colored bars to serve as visual yardsticks. Also drawn to scale, they illustrate the *minimum* clock pulse width of 35 ns. But clock pulses are allowed to be longer. (The table indicates the '816 has no upper limit. But some 65xx chips, including the old NMOS 6502, do have an upper limit on clock pulse width.)

For discussion's sake let's start with  $t_{PWL}$  and  $t_{PWH}$  of about 100 ns. In other words the  $t_{CYC}$  is about 200 ns (5 MHz), shown below.



This diagram (an animated GIF) shows that at 5 MHz ( $t_{CYC} = 200\text{ns}$ ) the clock duty cycle can be varied greatly. At this speed you can drastically shorten  $t_{PWH}$  while lengthening  $t_{PWL}$ , or vice versa, and that's completely okay. As noted, the colored bars show 35 ns — and that's the minimum clock pulse, the thing that needs to be respected. We see there's a wide margin of tolerance before the minimum spec is approached (**YELLOW**) or violated (**RED**).

## What happens when Cycle Time $t_{CYC}$ is shorter (faster)?




Now let's reduce both  $t_{PWL}$  and  $t_{PWH}$  to about 50 ns, shortening  $t_{CYC}$  (ie; raising the operating frequency).

This has implications regarding duty cycle. But first, here's a point which is absolutely crucial:

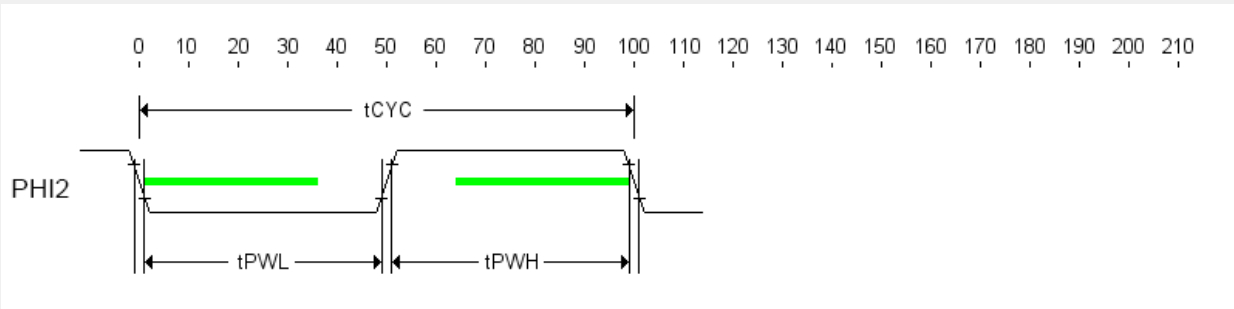
**All delays specified in ns (including  $t_{PWL}$ ,  $t_{PWH}$  and all the others) remain at the specified ns figure regardless of changes in operating frequency (Cycle Time). A change in *voltage* will affect those delays, as we'll discuss later. But a change in operating frequency doesn't mean  $t_{PWL}$ , for example, will suddenly require a different number of ns. The stated figures stand.**

This has implications regarding the "shape" of the cycle. The proportions are different at different operating frequencies. Watching the animation above it's clear that satisfying minimum **tPWL** and **tPWH** takes a modest slice of the Cycle Time at first. Then, when the Cycle Time shrinks by half, there's decidedly less margin in satisfying the minimum 35 ns spec. On a scale drawing the proportions can be seen at a glance. (See also the examples later.)

Unfortunately, **timing diagrams in data sheets typically aren't drawn to scale, which makes them useless for visual evaluation of proportions**. If you want a scale drawing it can be very helpful to **make your own** : idea: — specific to the operating frequency you've chosen. But, because a scale drawing *is* specific to the operating frequency chosen, data sheets usually avoid them. Data sheets need to be general; they can't include an individual timing diagram for every possible prospective operating frequency. But if you want that visual insight it's not hard to modify the datasheet's timing diagram using an image editor. Or just redraw it from scratch. There's a modified '816 timing diagram [here](#), which I scaled for 14 MHz.

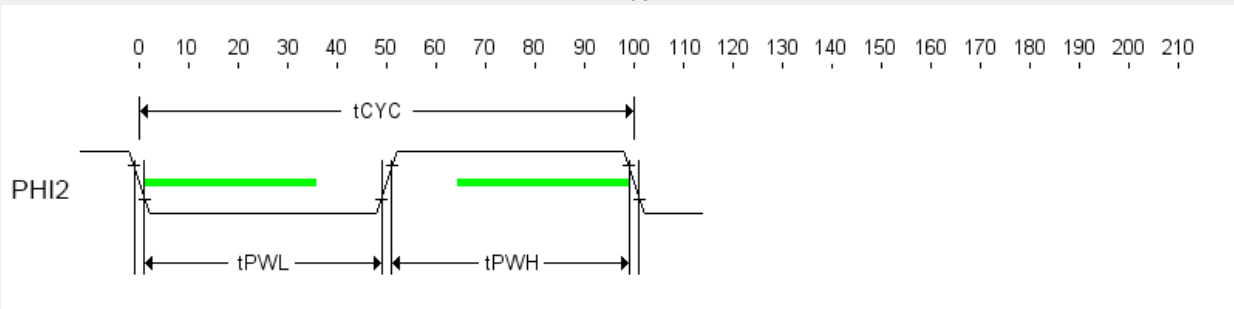
Now back to the matter of duty cycle. :)

GIF05



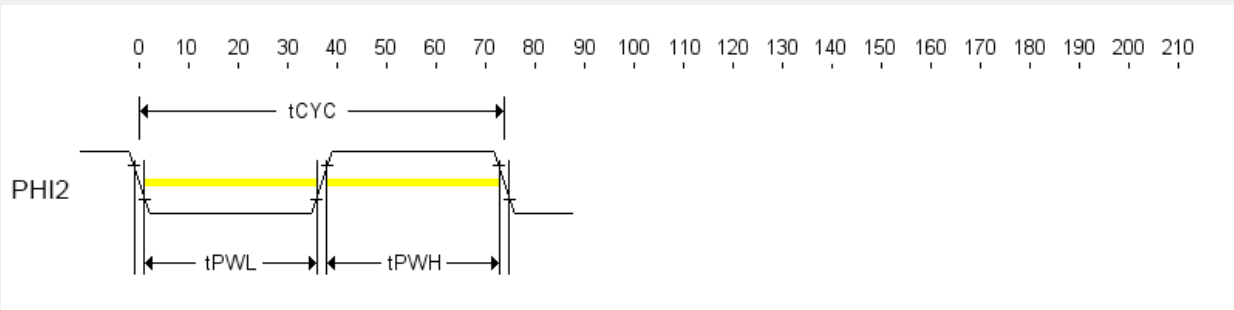
**tCYC** is now 100 ns (10 MHz). Compared to GIF03, the tolerance for duty-cycle variations is reduced but still remains.

GIF06



Finally we shorten **tPWL** and **tPWH** to the minimum value (35 ns).

GIF07



With **tPWL** and **tPWH** at the minimum value, **tCYC** is now about 70 ns (14 MHz). At this speed, the duty-cycle must be 50:50, as any variation would mean a violation of either the **tPWL** or **tPWH** minimum spec.

We have seen that **tPWL** and **tPWH** determine not only the permissible duty cycle but also the maximum operating frequency (14 MHz at 5 volts). Operating frequency is a dependent variable, and could be omitted from the chart. Nevertheless it's convenient that WDC included it.

## What happens when VDD is reduced?

GIF08

Table 4-2 W65C816S AC Characteristics

Symbol	Parameter	5.0 +/- 5%		3.3 +/- 10%		3.0
		14MHz		8MHz		6
		Min	Max	Min	Max	Min
VDD		4.75	5.25	3.0	3.6	2.8
tCYC	Cycle Time	70	DC	125	DC	150
tPWL	Clock Pulse Width Low	35	-	63	-	60
tPWH	Clock Pulse Width High	35	-	62	-	60
tF,tR	Fall Time, Rise Time	-	5	-	5	-
tAH	A0-A15 Hold Time	10	-	10	-	-
tADS	A0-A15 Setup Time	-	30	-	40	-

Next to the columns for 5V operation are the columns for 3.3 volts.

At 3.3 volts, input signals such as Clock need to be slower, and output signals take longer to appear.

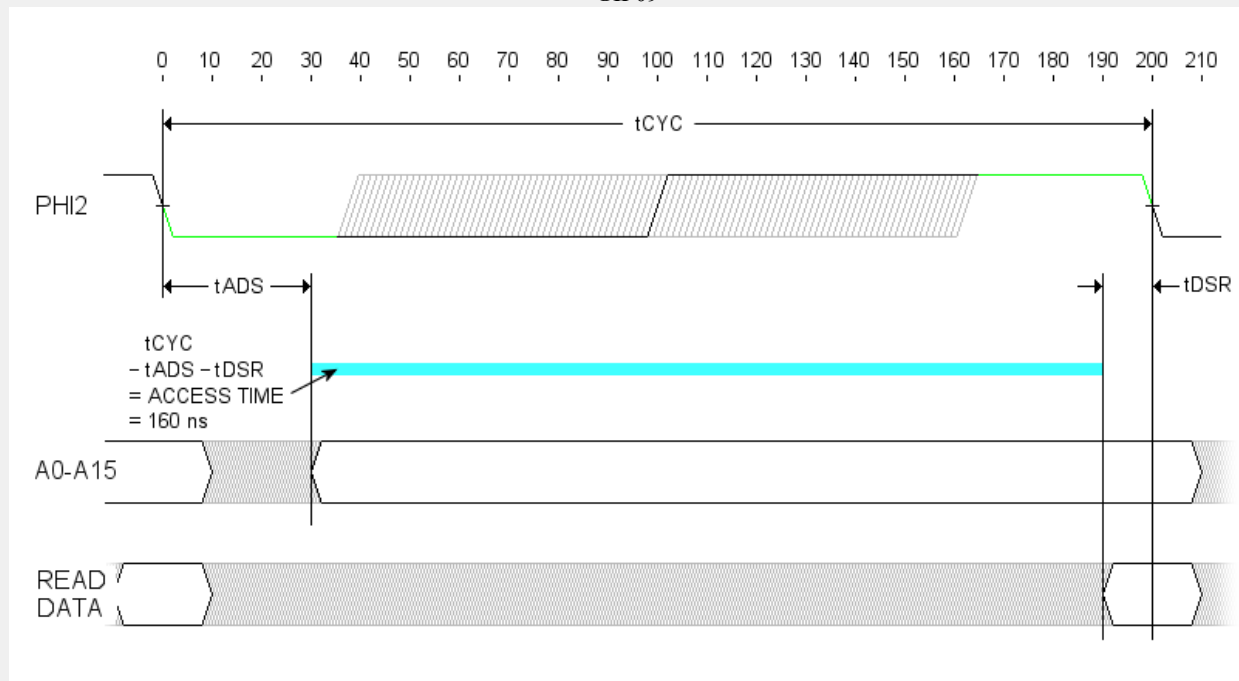
To show what happens when we operate the '816 at voltages below 5 volts, above we have another excerpt from Table 4-2. At 3.3 volts the **tPWL** & **tPWH** minimum spec almost doubles. I'm guessing I don't need to redo the GIF diagrams to illustrate this, so please just *imagine* that the colored bars in the diagrams have grown from 35 ns to about 62.5 ns. 🖼️:)

In the first example (GIF03, at 5 MHz) wider colored bars (increased minimum spec) would merely mean less tolerance regarding duty cycle. But in the later examples (GIF05, at 10 MHz), the wider bars would show that **tPWL** and **tPWH** violations are inescapable. 🖼️:: The limits on minimum **tPWL** & **tPWH** are "why" the '816 isn't guaranteed to go faster than 8 MHz at 3.3 volts (or 14 MHz at 5 volts).

(In practice it's usually possible to operate successfully at higher speeds. Following the specs assures success, timing-wise, but violating them doesn't necessarily assure failure. When you exceed the guaranteed spec you're relying on unspecified extra leeway — like how much margin the processor passed the tests with, the speed of devices connected, construction, temperature and other factors.)

## How the CPU Interacts With the Outside World

GIF09





Above we see a read cycle. This elementary occurrence is a back-and-forth transaction between the CPU and the outside world, and that means we need to accommodate both the CPU and whatever it's communicating with (typically memory, but possibly an IO device). We'll discuss the WDC W65C02S, which is highly similar to the '816. (Oct 2010 'C02 data sheet [here](#))

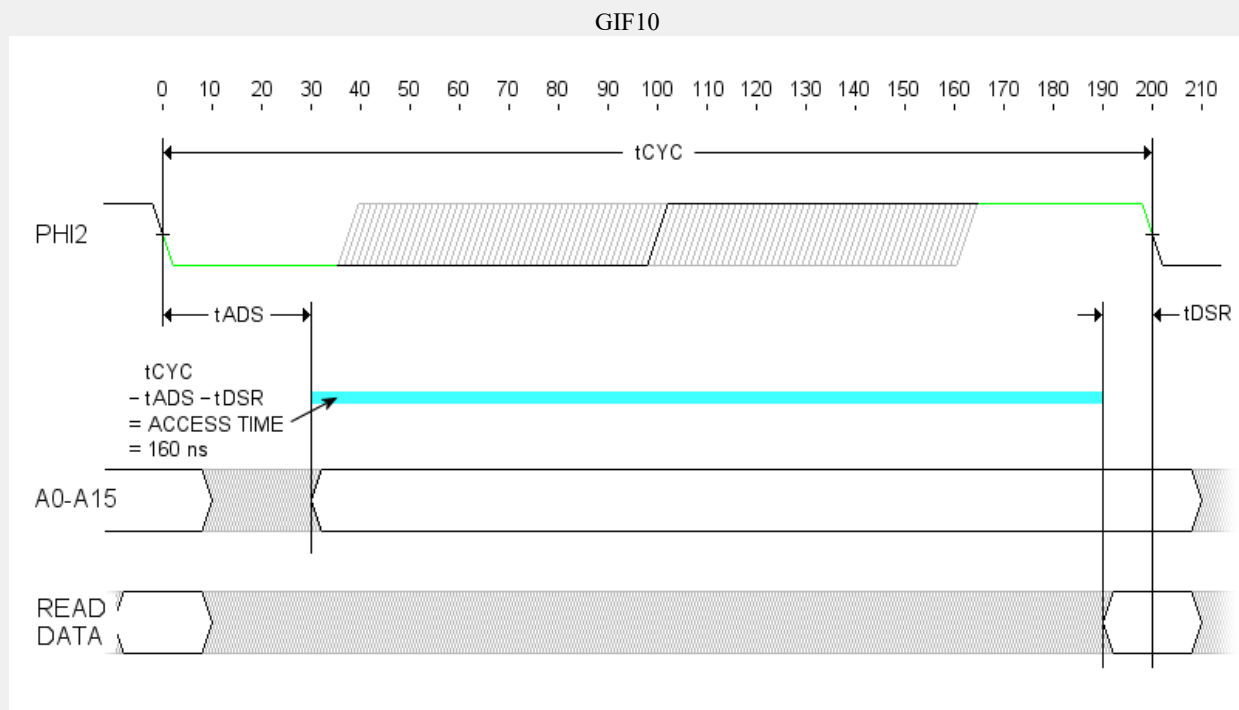
- at the beginning of the cycle, **tADS** (the A0-A15 Setup Time) must be endured. This is how long the CPU takes to output the address of the location that's to be read.
- next is the access time of the outside world (memory). The access can't begin until the address is valid — ie; after **tADS**.
- after the access, when memory has already returned data from the requested location, the CPU requires an extra timing margin prior to the end of the cycle. The spec for that margin is **tDSR** (the Read Data Setup Time).

The "man in the middle" is memory. To determine how fast it must respond, we take the CPU cycle time and subtract the two CPU delays. In other words,  $t_{CYC} - t_{ADS} - t_{DSR} = \text{access time}$ . With 5 volt operation those figures are 200 ns - 30 ns - 10 ns = 160 ns, as shown in the diagram. 160 ns is what the CPU leaves available, and *all non-CPU delays must fit within that* — not just the memory IC itself but all other delays too. A partial list might include the delays of the decoder, the address buffers IC's (if used), and the data-bus transceiver (if used). It's OK if the total of these delays (including the memory IC) is less than 160 ns, but if the total *exceeds* 160 ns then there'll be a violation of what happens next — **tDSR** (the Read Data Setup Time).

Notice that **the rising edge of PHI2 has no relevance to Access Time**. In that regard PHI2 simply doesn't matter! But of course as a general requirement the minimums for **tPWL** and **tPWH** (shown in green) need to be observed.



Many computers use the PHI2 signal as an input to the logic that generates the read enable for memory. But for this job there's nothing magic about PHI2 — alternatives are acceptable. Some machines have clock-generation circuitry that can easily provide a suitable read-enable signal, and such a signal needn't closely mimic PHI2 (although it does need to enable memory soon enough that tDSR will be satisfied).

**Aside:** sharp-eyed readers will notice a difference in how transitions are marked in this diagram as compared to those before. The earlier diagrams, modeled on Figure 4-1 from the '816 data sheet, mark the start and end of each transition. The interval between is the rise or fall time, during which the voltage is neither a valid high nor a valid low. In contrast, WDC's W65C02S datasheet uses a different style. There, the timing diagram marks only the midpoint of each transition. That results in a cleaner diagram...  and a quibblesome loss of precision.  ( But if rise & fall times are known to be short then you'll probably feel comfortable just ignoring the tiny discrepancy.



Above we see the effect on access time when Cycle Time is cut in half. The "overhead" of **tADS** plus **tDSR** is constant at 40 ns. Halving the Cycle Time means...

- the CPU is running faster by a factor of 2
- memory is required to run faster by a factor of 2.666

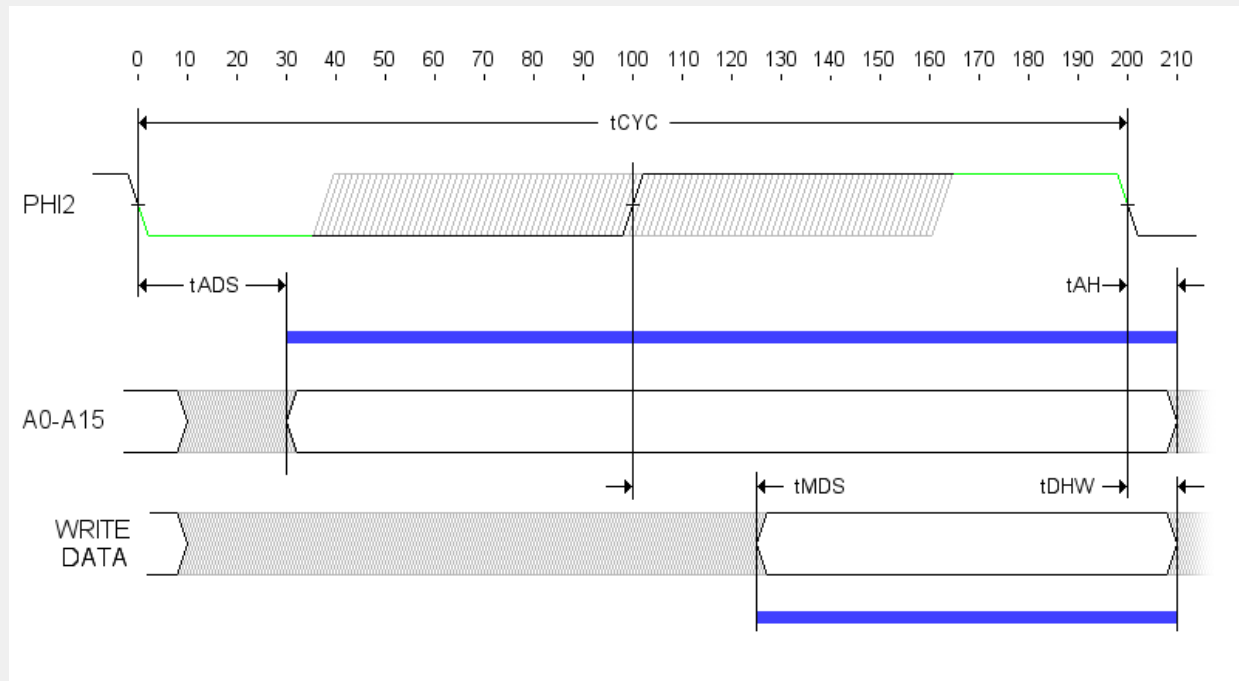
If that seems surprising, better look again. :  The proportions are different at a different operating frequency.

## A Write cycle

A write cycle is similar to a read, but the sequence is all one-way — ie; it's all CPU to outside-world. We don't have to get anything back. But now we have two items to send out: the address (as before) but also the write data.

The diagrams below show how to determine the start and end of the period during which the address is valid, and likewise the period during which the write data is valid. In a future post maybe we'll run an example, and see whether a specific memory chip of our choice can accommodate such operation. But right now let's just be aware that, **"this is what the CPU gives the outside world to work with."** Same deal with the read cycle's Access Time — it's what the CPU expects the outside world to work with.

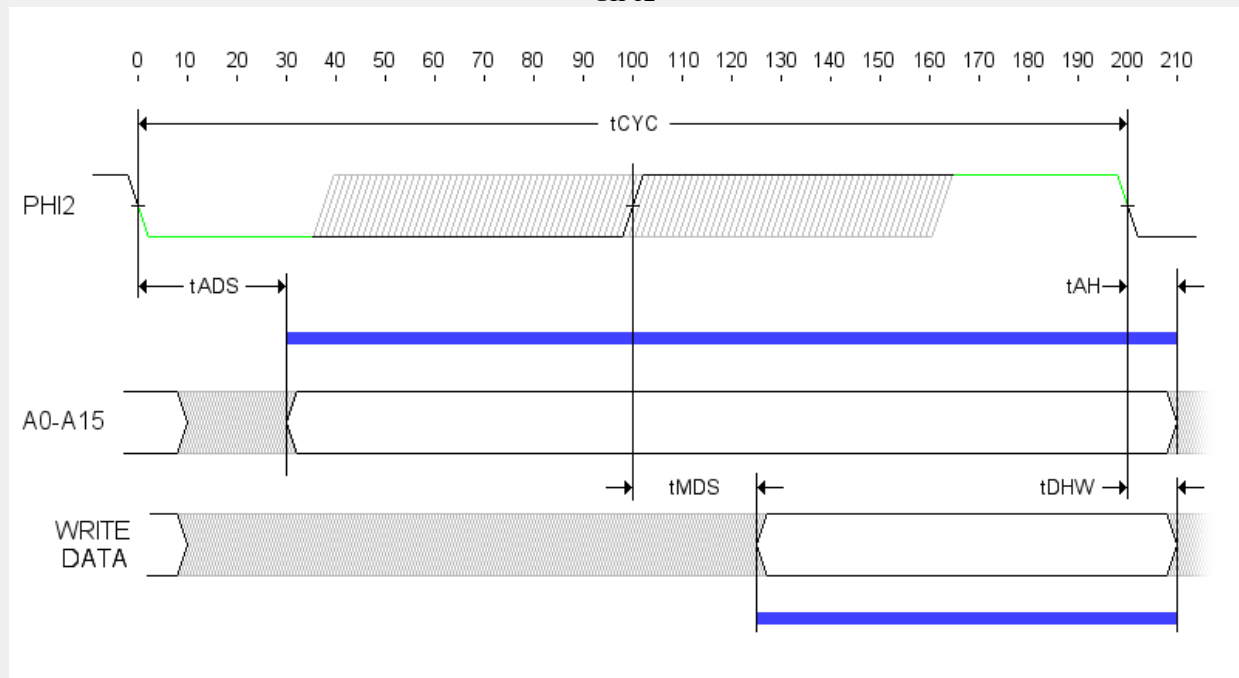
GIF11



As with the previous examples, we see (below) that changing the cycle time alters the shape (proportions) of the waveforms. As noted, that's because delays specified in ns remain at the specified ns figure regardless of changes in operating frequency.

With writes, the rising edge of PHI2 merits more attention, because PHI2 is the signal that tells the CPU to enable its tristate buffers and drive the data bus. In certain cases it might be advantageous to change the duty cycle so the PHI2 rising edge occurs sooner. That means the buffer-enable delay,  $t_{MDS}$ , would be completed sooner, resulting in a more generous data-valid window for the outside world to work with.

GIF12



LAUGHTON ELECTRONICS

NAVIGATE

[Projects](#)

[Servicing the Unserviceable](#)

[Main index](#) —» [for registerheads only](#)

© Jeff Laughton

View my profile on **Linked in**