# The Propeller counters, a dummies guide.
# By Graham Stabler

This document is a work in progress designed to bridge the gap between the Propeller manual and practical use of the counters. I'm not an expert but rather a frustrated beginner who hopes to relieve the frustrations of others. Special thanks go to Tracy Allen for helping me understand a bit more.

*Aside: if I write FRQx it means it could be FRQA or FRQB, it just depends on the counter being used.*

**The counter modules:**

The propeller has 8 cogs as we know and each one of these contains two counter modules, A and B. The counter modules do just that, they count; that is they add one number to another repeatedly. The simplest form of counting is to repeatedly add one, like you are counting sheep (much like reading this document), the propeller can do this sort of counting except you would be more likely to count some electrical event like a pulse edge. The propeller can also count up in numbers other than one; a real life example would be counting the legs of the sheep. You would want to add 4 to the count every time you saw a sheep, but (and this is important,) you would only want to add 4 to the count **if** you saw a sheep, not if saw a mongoose or an aardvark, just a sheep.

This is conditional accumulation and this is what the counter modules do. On every clock cycle they take a number (for example 4) stored in a register called FRQx and add it to the count, stored in a register called PHSx but only if certain conditions are met.

**What becomes of this number?**

It's all very well counting but what can PHSx be used for. Firstly your program can read it, for example if you were counting a simple event you could read it and display it. It might sound like you don't need a counter module for that task but as it is done by the hardware it is very fast and allows your program to do other things. There are some operating modes that associate the most significant bit of PHSx, PHSx[31] with an output pin, the pin goes high every time a 1 appears in PHSx[31], how often that happens depends on the value in FRQx and because of that it is described in the manual as a numerically controlled oscillator or NCO. Although it is easy to think of a bit as just part of a number that toggles you should try to think of it as an oscillating electrical signal like any other. Other modes feed the NCO (PHSx[31]) into a PLL multiplier, this multiplies the frequency by 16 before applying it to an output pin. You can also use taps off this X16 signal to produce other multiplication factors. The divide by two tap for example gives you X8 overall.

It should have become obvious that there are several things that must be configured for each counter, the mode of operation, the pins used for output and input (if any) and potentially the PLL tap. These are set up using the CTRx register than is well explained in the manual, I will however go through some of the operating modes for clarity.

**Counter modes:**

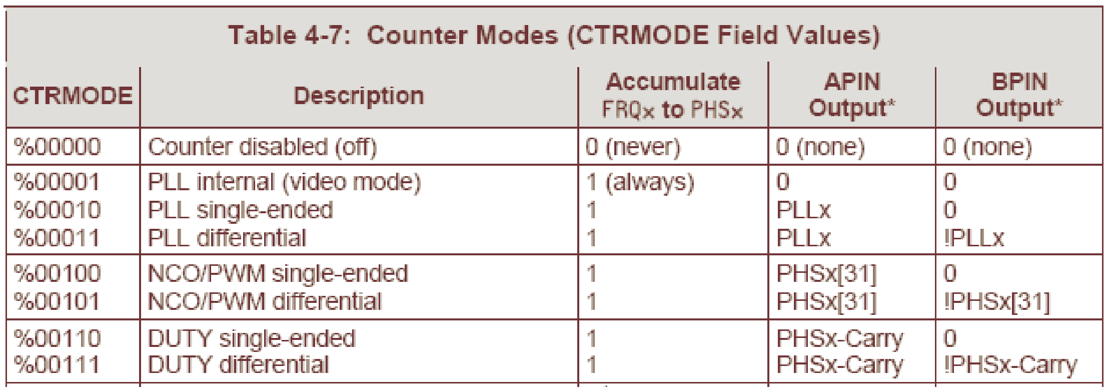

Table 4-7: Counter Modes (CTRMODE Field Values)

| CTRMODE | Description | Accumulate FRQx to PHSx | APIN Output* | BPIN Output* |
|---|---|---|---|---|
| %00000 | Counter disabled (off) | 0 (never) | 0 (none) | 0 (none) |
| %00001 | PLL internal (video mode) | 1 (always) | 0 | 0 |
| %00010 | PLL single-ended | 1 | PLLx | 0 |
| %00011 | PLL differential | 1 | PLLx | !PLLx |
| %00100 | NCO/PWM single-ended | 1 | PHSx[31] | 0 |
| %00101 | NCO/PWM differential | 1 | PHSx[31] | !PHSx[31] |
| %00110 | DUTY single-ended | 1 | PHSx-Carry | 0 |
| %00111 | DUTY differential | 1 | PHSx-Carry | !PHSx-Carry |

Here is the top of the table showing the available counter modes in the manual. The first column provides the number to be placed in the CTRMODE field of CTRx, the second column gives the name of the mode although it must be stressed that they are really names of potential applications, mode 00100 does not by itself have the ability to provide PWM but with come maintenance from the cog it may be used to produce it. The third column shows the condition for accumulation, for all of these modes the accumulation happens on every single clock cycle except for mode 0 where the counter is disabled completely. The next two columns show how up to two output pins may be used, the mapping of APIN and BPIN is configured in the respective fields of CTRx. Although the table shows what you will get out of the pins in these modes you are not obliged to define pins if you don't want to, for example in mode 00011 if no pin were defined for BPIN you would get the same operation as mode 00010.

So if we consider mode 00010 we find that the value we place in FRQx is accumulated into PHSx on every clock cycle. The MSB of PHSx which acts as an NCO is fed into the X16 PLL and the multiplied signal (subject to the PLL tap selected) applied to APIN as defined in CTRA. In mode 00100 we find that the NCO is applied directly to APIN. In mode 00110 it is the carry of PHSx that is applied to APIN. I'll come back to how this might be used to produce PWM or "Duty" signals later but first I'll show some of the modes with conditions.

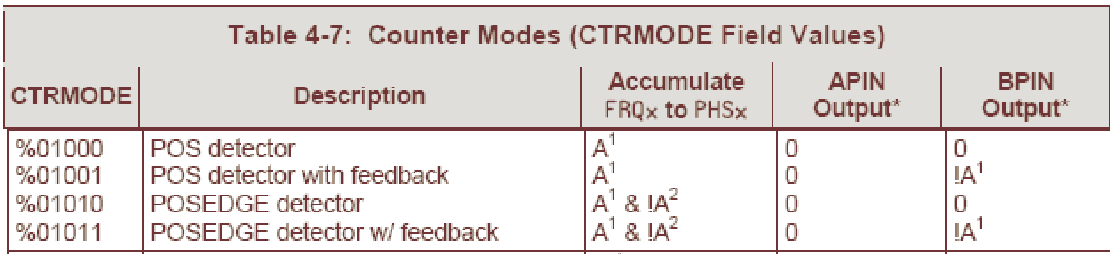

Table 4-7: Counter Modes (CTRMODE Field Values)

| CTRMODE | Description | Accumulate FRQx to PHSx | APIN Output* | BPIN Output* |
|---|---|---|---|---|
| %01000 | POS detector | A$^1$ | 0 | 0 |
| %01001 | POS detector with feedback | A$^1$ | 0 | !A$^1$ |
| %01010 | POSEDGE detector | A$^1$ & !A$^2$ | 0 | 0 |
| %01011 | POSEDGE detector w/ feedback | A$^1$ & !A$^2$ | 0 | !A$^1$ |

Looking at mode 01000 we see that it will accumulate only when A$^1$ is one. A$^1$ is the state of the APIN (which in this case is an input) one clock cycle ago. So in this instance accumulation occurs only when APIN is high, a simple example of its use would be to run the counter for a fixed number of clock cycles (using waitcnt say), if

1 was placed in the FRQx register then afterwards you could work out for what percentage of the time the pin was high for, if you did it for long enough you could measure the duty of a PWM signal.

Mode 01010 detects a positive edge, it only accumulates when the state of APIN two clocks ago was low and one clock ago was high, that will only occur on an edge so it will literally count edges, a simple example would be a digital frequency meter.  You would again load FRQx with one and run the counter for a fixed time period or at least read PHSx at regularly timed intervals.

*The next bit will be about binary stuff,  how you work out what number to put into FRQx etc and how outputting a carry bit is useful.*