Propeller P8X32A Counters

Abstract: Each of the multicore P8X32A’s eight processors (cogs) has two independent hardware counter modules. Each counter is a configurable state machine for sensing or generating repetitive signals, potentially on every clock cycle. Use the counter modules as flexible subsystems that can often take the place of dedicated cogs or peripheral hardware, reducing code complexity and component count in an application.

Introduction

The Propeller chip has two counters within each cog for a total of sixteen. Every counter operates independently from each other and each one is an advanced module having 32 modes of operation. This application note will provide detailed information on these modes and examples of their use. Figure 1 is the block diagram of a single counter module applicable to all modes of operation. Simplified block diagrams of each subset of modes will be provided in the discussion of those modes.

Figure 1: Counter Block Diagram
Counter Registers

Each counter has three distinct registers that control its operation. In each cog these registers are named CTRA, FRQA and PHSA for the first counter; and CTRB, FRQB and PHSB for the second counter.

CTRA / CTRB is the control register for each counter. This register sets the mode in which the counter operates (CTRMODE field), the Phase-Locked Loop (PLL) division factor used in the PLL modes (PLLDIV field) and the first and second pins (APIN and BPIN fields). The two pins are used in certain modes as input or output to the counter, as will be explained for each mode. The locations of the fields are shown in Table 1.

Table 1: CTRA and CTRB Registers

<table>
<thead>
<tr>
<th>31</th>
<th>30..26</th>
<th>25..23</th>
<th>22..15</th>
<th>14..9</th>
<th>8..6</th>
<th>5..0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>CTRMODE</td>
<td>PLLDIV</td>
<td>-</td>
<td>BPIN</td>
<td>-</td>
<td>APIN</td>
</tr>
</tbody>
</table>

The CTRA / CTRB register’s fields are organized to make use of the assembly instructions MOVS, MOVD, and MOVI. The APIN field may be set using the MOVS instruction, the BPIN field using the MOVD instruction and the CTRMODE/ PLLDIV fields using the MOVI instruction. The BPIN and APIN fields are six bits each, the highest bit in each field (bits 5 and 14) are reserved for future use and are ignored by the Propeller P8X32A. Both counters have their own APIN and BPIN and are specified uniquely in their CTRA / CTRB register.

In every case, the actual output of a counter, as determined by the CTRMODE and APIN/BPIN fields, is OR’d with the corresponding output state bit in the output register (ex: OUTA) and AND’d with the corresponding direction bit in the direction register (ex: DIRA). For typical operation, the output state bit must be set to 0 and the direction bit set to 1; however, they can be set differently to effectively mask the counter as desired.

The PHSA / PHSB register is the heart of the counter; it is the accumulator that stores the counter’s current value. This value can be read from or written to by a program; though for many applications writing to the register is not necessary.

The FRQA / FRQB register holds the value which is added to the accumulator whenever an accumulate condition is true. The accumulate condition is specified by the mode of operation set in the CTRMODE field of the CTRA / CTRB register.

Whenever a cog is started, all six registers (three for each counter) are initialized to $0000_0000. Whenever a cog is stopped; all counter activity ceases and the registers are reset to $0000_0000.

Note: For the sake of clarity, the remainder of the document will refer to Counter A. The second counter’s operation is identical to the first counter; therefore what is explained for the first counter also applies to the second counter.

Counter Operation

The counter operates by adding the contents of FRQA to PHSA at each clock cycle the accumulate condition is true. For a Propeller clocked by a 5 MHz crystal with a 16x PLL multiplier, the counter will be conditionally accumulated 80 million times per second (16 x 5 MHz = 80 MHz). The condition for accumulation is specified by the CTRMODE field within the CTRA register.
Counter Modes

The 32 modes of operation for each counter are specified in Table 2. Table 3 provides application examples for each subset of modes.

Table 2: Counter Modes (CTRMODE Field Values)

<table>
<thead>
<tr>
<th>CTRMODE</th>
<th>Description</th>
<th>Accumulate FRQx to PHSx</th>
<th>APIN Output*</th>
<th>BPIN Output*</th>
</tr>
</thead>
<tbody>
<tr>
<td>%00000</td>
<td>Counter disabled (off)</td>
<td>0 (never)</td>
<td>0 (none)</td>
<td>0 (none)</td>
</tr>
<tr>
<td>%00001</td>
<td>PLL internal (video mode)</td>
<td>1 (always)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%00010</td>
<td>PLL single-ended</td>
<td>1</td>
<td>PLLx</td>
<td>0</td>
</tr>
<tr>
<td>%00011</td>
<td>PLL differential</td>
<td>1</td>
<td>PLLx</td>
<td>!PLLx</td>
</tr>
<tr>
<td>%00100</td>
<td>NCO single-ended</td>
<td>1</td>
<td>PHSx[31]</td>
<td>0</td>
</tr>
<tr>
<td>%00101</td>
<td>NCO differential</td>
<td>1</td>
<td>PHSx[31]</td>
<td>!PHSx[31]</td>
</tr>
<tr>
<td>%00110</td>
<td>DUTY single-ended</td>
<td>1</td>
<td>PHSx-Carry</td>
<td>0</td>
</tr>
<tr>
<td>%00111</td>
<td>DUTY differential</td>
<td>1</td>
<td>PHSx-Carry</td>
<td>!PHSx-Carry</td>
</tr>
<tr>
<td>%01000</td>
<td>POS detector</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%01001</td>
<td>POS detector with feedback</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%01010</td>
<td>POSEDGE detector</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &amp; !A&lt;sup&gt;2&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%01011</td>
<td>POSEDGE detector w/ feedback</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &amp; !A&lt;sup&gt;2&lt;/sup&gt;</td>
<td>0</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%01100</td>
<td>NEG detector</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%01101</td>
<td>NEG detector with feedback</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%01110</td>
<td>NEGEDGE detector</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt; &amp; !A&lt;sup&gt;2&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%01111</td>
<td>NEGEDGE detector w/ feedback</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt; &amp; !A&lt;sup&gt;2&lt;/sup&gt;</td>
<td>0</td>
<td>!A&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%10000</td>
<td>LOGIC never</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10001</td>
<td>LOGIC A &amp; !B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &amp; !B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10010</td>
<td>LOGIC A &amp; !B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &amp; !B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10011</td>
<td>LOGIC !B</td>
<td>B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10100</td>
<td>LOGIC A &amp; B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &amp; B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10101</td>
<td>LOGIC A</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10110</td>
<td>LOGIC A &lt;&gt; B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &lt;&gt; B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%10111</td>
<td>LOGIC A</td>
<td>!B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>!B&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%11000</td>
<td>LOGIC A &amp; B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; &amp; B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%11001</td>
<td>LOGIC A == B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt; == B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%11010</td>
<td>LOGIC A</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%11011</td>
<td>LOGIC A</td>
<td>!B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>!B&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%11100</td>
<td>LOGIC B</td>
<td>B&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>%11101</td>
<td>LOGIC A</td>
<td>B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>B&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%11110</td>
<td>LOGIC A</td>
<td>B</td>
<td>A&lt;sup&gt;1&lt;/sup&gt;</td>
<td>B&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>%11111</td>
<td>LOGIC always</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Must set corresponding DIR bit to affect pin

A<sup>1</sup> = APIN input delayed by 1 clock
A<sup>2</sup> = APIN input delayed by 2 clocks
B<sup>1</sup> = BPIN input delayed by 1 clock

Table 3: Counter Modes Application Examples

<table>
<thead>
<tr>
<th>CTRMODE*</th>
<th>Description</th>
<th>Example Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>%001X</td>
<td>PLL</td>
<td>RF carrier synthesis, clock generation</td>
</tr>
<tr>
<td>%010X</td>
<td>NCO</td>
<td>Servo control, PWM motor control, LED dimming, audio generation</td>
</tr>
<tr>
<td>%011X</td>
<td>DUTY</td>
<td>Digital to analog conversion, audio generation</td>
</tr>
<tr>
<td>%01X0</td>
<td>POS/NEG detector</td>
<td>Pulse width measurement, duty cycle measurement</td>
</tr>
<tr>
<td>%01X1</td>
<td>POS/NEG detector w/ feedback</td>
<td>Analog to digital conversion</td>
</tr>
<tr>
<td>%0X1X</td>
<td>POSEDGE/NEGEDGE</td>
<td>Event counter, frequency measurement</td>
</tr>
<tr>
<td>%1XXXX</td>
<td>LOGIC</td>
<td>Propagation delay measurement, long duration event timer</td>
</tr>
</tbody>
</table>

* X = either a 0 or 1; multiple modes are applicable
NCO Modes of Operation

Modes %00100 (NCO single-ended) and %00101 (NCO differential) specify the counter should operate in a Numerically Controlled Oscillator (NCO) mode. Applications in which these two modes are useful are motor control and audio generation. When operating in this mode, the value in FRQA is added to value in PHSA every clock cycle. In both modes the highest bit of PHSA is copied to APIN output, in mode %00101 the logical inverse of the highest bit of PHSA is copied to the BPIN output. Figure 2 is a block diagram of the mode.

Figure 2: NCO Mode Block Diagram

For example, if the APIN is set to 0, the mode is set to %00100 (NCO single-ended) and FRQA is set to $8000_0000 as in the following code.

```plaintext
Demonstration of NCO counter mode (%00100)
CON
  _clkmode  = xtal1 + pll16x
  _xinfreq  = 5_000_000
PUB NCO_single_ended_mode
  mode  PLL        BPIN     APIN
  ctra    := %00100_000 << 23 + 1 << 9 + 0  'Establish mode and APIN (BPIN is ignored)
  frqa    := $8000_0000                     'Set FRQA so PHSA[31] toggles every clock
  dira[0] := 1                              'Set APIN to output
  repeat                                    'infinite loop, so counter continues to run
```

The following table shows the sequence in the PHSA register and the resultant values on APIN:

**Table 4: NCO State Progression**

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>PHSA value</th>
<th>APIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0000_0000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$8000_0000</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$0000_0000</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$8000_0000</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$0000_0000</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>$8000_0000</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$0000_0000</td>
<td>0</td>
</tr>
</tbody>
</table>

The output signal as seen on an oscilloscope is shown in Figure 3. All oscilloscope figures are shown with the horizontal origin set to 2 volts and the 0 volt line 2 major divisions below the origin.

**Figure 3: NCO Mode %00100 Output**

![NCO Mode %00100 Output](image)
If the mode were changed to %00101 (NCO differential) the output would appear as Figure 4: the blue (dark) line is APIN output signal and the red (light) line is BPIN output signal.

**Figure 4: NCO Mode %00101 Output**

If the master clock is running at 80 MHz, the resultant output on APIN will be 40 MHz or \( \frac{1}{2} \) the system clock. If the contents of FRQA were changed to $4000_0000$, the sequence of PHSA would be $0000_0000$, $4000_0000$, $8000_0000$, $C000_0000$, $0000_0000$,... This would result in PHSA\[31\] to have the sequence 0,0,1,1,0,... or \( \frac{1}{4} \) the system clock.

The general formula for the frequency in the NCO mode is shown in Equation 1.

\[
 f_{Hz} = \frac{FRQA}{2^{32}} \cdot System\ Frequency \tag{1}
\]

The above equation works for values of FRQA between 0 and $8000_0000$; for values greater than $8000_0000$ the output frequency decreases according to Figure 5.
For values of FRQA which are not a power of 2, \((FRQA \neq 2^N)\), there will be jitter present on
the output signal since the most significant bit of PHSA will toggle at an inconstant rate.

Applications which require rapid changes to FRQA, such as audio generation, require the use
of assembly since Spin cannot alter the FRQA register fast enough for frequencies that
change rapidly.

Pulse Width Modulation is a Numerically Controlled Oscillator where the amount of high time
and low time of the signal may be unequal but the period, or sum of the two times, remains
equal. Below is a code example for PWM where the fraction of the period spent high is
linearly scaled from 0 to 100 percent.

```
''Demonstration of PWM version of NCO counter mode
CON _clkmode = xtal1 + pll16x
    _xinfreq = 5_000_000

VAR long parameter

PUB go | x
    cognew(@entry, @parameter)  'start assembly cog
repeat
    repeat x from 0 to period  'linearly advance parameter from 0 to 100
        parameter := x           'a constant here locks to value x percent high
        waitcnt(100_000 +cnt)    'wait a little while (1000 periods) before next update

DAT
''assembly cog fetches the value in parameter for PWM percentage
    org
    entry    mov dira, diraval                    'set APIN to output
    mov ctra, ctraval                    'establish counter A mode and APIN
    mov frqa, #1                        'set counter to increment 1 each cycle
    mov time, cnt                       'record current time
    add time, period                   'establish next period
:loop    rdlong value, par           'get an up to date pulse width
    waitcnt time, period             'wait until next period
    neg phsa, value                 'back up phsa so that it trips "value" cycles from now
    jmp #:loop                      'loop for next cycle
    diraval long |< 0                 'APIN=0
    ctraval long %00100 << 26 + 0     'NCO/PWM APIN=0
```
The spin method goes launches the assembly cog, then proceeds to modify the value in the variable parameter at a rate of 1 percent every 100,000 cycles. The assembly program sets up the counter then fetches the variable parameter. After the period expires, the PHSA register is set to 0 – parameter so PHSA’s most significant bit will transition from 1-to-0 parameter cycles later. The output on APIN appears shown in Figure 6.

**Figure 6: Scaling PWM Output**

The jaggedness of the saw tooth is due to the switching of the signal and can be smoothed with the addition of a resistor and capacitor to the output.
PLL modes of operation

Modes %00001, %00010 and %00011 are just like the NCO mode of operation with the addition of a Phase-Locked Loop (PLL). A PLL multiplies an input clock signal by a fixed amount using a voltage controller oscillator (VCO) and locks the output onto the input clock so there is no difference between the phase of the input and output clocks. Mode %00010 (PLL single-ended) is similar to mode %00100 (NCO single-ended) with the addition of the PLL, mode %00011 (PLL differential) is similar to mode %00101 (NCO differential) with the addition of the PLL. Mode %00001 (PLL internal (video mode)) is a special PLL mode designed to be used in conjunction with broadcast television to generate the audio sub-carrier and is beyond the scope of this application note.

Figure 7: Block Diagram of a PLL

![Block Diagram of a PLL](image)

The PLL shown in Figure 7 doubles the input clock frequency and provides two frequencies as output, the input frequency and twice the input frequency. The PLL located in each counter multiplies the input frequency by 16 and provides 8 output frequencies (16x, 8x, 4x, 2x, 1x, ½x, ¼x and ⅛x the input frequency).

The PLLDIV field located in the CTRA register specifies which output frequency is output to the APIN according to the value shown in Table 5. The PLL modes of operation are the only modes which use the PLLDIV field.

Table 5: PLLDIV Field

<table>
<thead>
<tr>
<th>PLLDIV</th>
<th>%000</th>
<th>%001</th>
<th>%010</th>
<th>%011</th>
<th>%100</th>
<th>%101</th>
<th>%110</th>
<th>%111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>VCO ÷ 128</td>
<td>VCO ÷ 64</td>
<td>VCO ÷ 32</td>
<td>VCO ÷ 16</td>
<td>VCO ÷ 8</td>
<td>VCO ÷ 4</td>
<td>VCO ÷ 2</td>
<td>VCO ÷ 1</td>
</tr>
</tbody>
</table>

A PLL is designed to work over a range of frequencies. The range for the input clock on the PLL located within each counter is 4 to 8MHz, which results in an output range of 64 to 128 MHz. Frequencies as low as 500 kHz can be output to APIN given the range of output divisions available from the PLL. Therefore, any frequency from 500 kHz to 128 MHz can be generated using the PLL counter modes. PLL counter modes can be used to generate Radio Frequencies (RF) and can help reduce jitter in a non-power of 2 FRQA value.

The amount of jitter is determined by the number of bits that encompass the most significant high bit and the least significant high bit (leftmost 1 and rightmost 1) of the FRQA value. The PLL is capable of de-jittering the input signal up to a certain amount, the effectiveness of which is determined both by how spectrally pure the input signal is and how noise sensitive the application under development is.
The block diagram for the mode is shown in Figure 8.

**Figure 8: PLL Mode Block Diagram**

![PLL Mode Block Diagram](image-url)
Duty Cycle modes of operation

Modes %00110 (DUTY single-ended) and %00111 (DUTY differential) on the surface appear similar to the NCO modes of operation; however the waveforms they produce are very different. The output of the Duty Cycle modes is the carry output of the PHSA register, whenever the PHSA register overflows (wraps around from $FFFF_FFFF to $0000_0000) the APIN is set to 1. The block diagram for this mode of operation is shown in Figure 9.

If FRQA is $0000_0001, the carry bit of PHSA will be 1 only once every $2^{32}$ (4,294,967,296) cycles. At an 80MHz system clock, this will occur approximately once every 54 seconds. Similarly if FRQA is $FFFF_FFFF, the carry bit of PHSA will be 0 only once every $2^{32}$ cycles. Figure 10 shows a few examples of the output waveforms of the Duty Cycle modes.

Figure 9: Duty Cycle Mode Block Diagram
In the Duty Cycle modes of operation the clock signal will be high FRQA/2^32 percent of the time; this is the definition of duty cycle. By comparison, the NCO mode of operation has a duty cycle of 50%, or the number of cycles the APIN is high is equal to the number of cycles the APIN is low (for FRQA equal to a power of 2—FRQA which is not a power of two will have jitter in the signal and the duty cycle of the resultant wave form will not be 50% when analyzed within a small time frame.) Only when FRQA is $8000_0000 do the Duty Cycle and NCO modes generate identical waveforms.

The Duty Cycle is useful for Digital to Analog Conversion (DAC). With a resistor and capacitor connected to the APIN, the output will be averaged to produce an analog voltage according to Equation 2.

\[
V = 3.3 \times \frac{FRQA}{2^{32}} \quad \text{Equation 2}
\]

An example of Digital to Analog Conversion is provided in the following code.

```plaintext
{{
  Demonstration of scaling Duty Cycle
  10kΩ
  APIN ──► Out
  ┌ .1μF
  ▼

  Delta modulation has no fundamental freq but has quantization noise
}}

CON _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

VAR long parameter
```
When executed with the resistor and capacitor attached to the pin, the output on APIN appears as in Figure 11.

**Figure 11: Scaling Duty Cycle Output**

The voltage presented at Out of the circuit is \((3.3 \times x)/\text{period}\) due to scaling the input value by $20C49B.
Logic modes of operation

Modes %10000 through %11111 operate differently than the other modes of operation. In these Logic modes, APIN and BPIN are inputs into the counter which dictate when the FRQA register is added into the PHSA register. Only when the logic equation specified by the mode is true is the FRQA register added to the PHSA register. The %10000 (LOGIC never) mode is equivalent to mode %00000 (Counter off), since the FRQA register is never added to the PHSA register. The %11111 (LOGIC always) mode is similar to the system clock in that the FRQA register is added to PHSA register every cycle. The remaining modes accumulate when the mode's logic equation evaluates as true. These modes operate on buffered inputs, so the values present at the APIN and BPIN of the previous clock cycle are used in the equation for the present clock cycle. This is done to stabilize the input signal in a manner similar to the inputs of the SX microcontroller. Table 6 illustrates when accumulation occurs.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Accumulates each cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>%10001</td>
<td>APIN=0 and BPIN=0</td>
</tr>
<tr>
<td>%10010</td>
<td>APIN=1 and BPIN=0</td>
</tr>
<tr>
<td>%10011</td>
<td>BPIN=0</td>
</tr>
<tr>
<td>%10100</td>
<td>APIN=0 and BPIN=1</td>
</tr>
<tr>
<td>%10101</td>
<td>APIN=0</td>
</tr>
<tr>
<td>%10110</td>
<td>APIN ≠ BPIN</td>
</tr>
<tr>
<td>%10111</td>
<td>APIN=0 or BPIN=0</td>
</tr>
<tr>
<td>%11000</td>
<td>APIN=1 and BPIN=1</td>
</tr>
<tr>
<td>%11001</td>
<td>APIN=BPIN</td>
</tr>
<tr>
<td>%11010</td>
<td>APIN=1</td>
</tr>
<tr>
<td>%11011</td>
<td>APIN=1 or BPIN=0</td>
</tr>
<tr>
<td>%11100</td>
<td>BPIN=1</td>
</tr>
<tr>
<td>%11101</td>
<td>APIN=0 or BPIN=1</td>
</tr>
<tr>
<td>%11110</td>
<td>APIN=1 or BPIN=1</td>
</tr>
</tbody>
</table>

Logic modes are useful to keep running tallies on external events such as measuring pulse widths, measuring RC time constants, etc. These modes can be used to establish complex systems such as an event timer for long duration events. Such a system could be achieved by using two counters, the first in Duty Cycle mode configured to output a high once a millisecond and the second in Logic mode %11000 (LOGIC A & B). The PHSA register would contain the number of milliseconds BPIN has been high and in theory with an 80 MHz system clock would be able to measure the duration of an event on BPIN from 1 millisecond to nearly 50 days long with millisecond resolution. The accuracy of such a measurement depends on the accuracy of the clock used for the Propeller and the accuracy of the counter configured in Duty Cycle to output a high once every millisecond.

The Block diagram for modes %10000 through %11111 is shown in Figure 12.
Figure 12: Logic Mode Block Diagram

Pin State Detection Modes of Operation

Modes %01000 through %01111 are used to track the state of APIN according to Table 7. Modes %01X0X, where X means either 0 or 1, operate on the buffered input of APIN. Modes %01X1X operate on the buffered input of APIN and the double buffered input of APIN in order to detect a transition between states on APIN.

Table 7: Pin State Equations

<table>
<thead>
<tr>
<th>Mode</th>
<th>Accumulates each cycle</th>
<th>Feedback to BPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>%01000</td>
<td>APIN=1</td>
<td>no</td>
</tr>
<tr>
<td>%01001</td>
<td>APIN=1</td>
<td>yes</td>
</tr>
<tr>
<td>%01010</td>
<td>APIN = ___*</td>
<td>no</td>
</tr>
<tr>
<td>%01011</td>
<td>APIN = ___*</td>
<td>yes</td>
</tr>
<tr>
<td>%01100</td>
<td>APIN=0</td>
<td>no</td>
</tr>
<tr>
<td>%01101</td>
<td>APIN=0</td>
<td>yes</td>
</tr>
<tr>
<td>%01110</td>
<td>APIN = ___**</td>
<td>no</td>
</tr>
<tr>
<td>%01111</td>
<td>APIN = ___**</td>
<td>yes</td>
</tr>
</tbody>
</table>

* Rising edge  
** Falling edge

Mode %01000 (POS detector) is identical to mode %11010 (LOGIC A) and mode %01100 (NEG detector) is identical to mode %10101 (LOGIC !A) in function. Feedback is where the inverse of APIN is output to BPIN. The block diagram for modes %01000 through %01111 are shown in Figure 13.
Figure 13: Pin State Detection Mode Block Diagram

Modes %01001 (POS detector with feedback) and %01101 (NEG detector with feedback) are designed to make Sigma-Delta Analog to Digital Conversion (ΣΔ-ADC) a simple process. A four-component external circuit is required as shown in Figure 14.

Figure 14: ΣΔ-ADC External Circuit

The two 1 nF capacitors and the 100 kΩ resistor must be placed within 1 inch (2.5 cm) of the Propeller chip for stable operation of the circuit and any excess leads must be cut away. This circuit will not work properly by placing it on a breadboard. The component marked as an asterisk is chosen according to the application. If an AC voltage is to be measured, a capacitor should be used and a value of 0.1 μF is sufficient for most applications. If a DC voltage is to be measured, a resistor should be used. For full scale measurements a value of 150 kΩ will provide a wide range of values without clipping either end. This value can be fine tuned by tying the Analog In to both the highest voltage to be measured and the lowest voltage to be measured, and assuring that both values are as close to the maximum and minimum values without clipping the measured value. The precise values of all...
components may change according to the requirements of the application. The following program demonstrates the \(\Sigma\Delta\)-ADC using mode %01001 (POS detector with feedback).

This program demonstrates the use of the counter in POS detector with feedback to perform ADC calculations.

CON

\[\text{_clkmode = xtal1 + pll16x}\]
\[\text{_xinfreq = 5_000_000}\]

At 80MHz the ADC/DAC sample resolutions and rates are as follows:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2.5 MHz</td>
</tr>
<tr>
<td>6</td>
<td>1.25 MHz</td>
</tr>
<tr>
<td>7</td>
<td>625 KHz</td>
</tr>
<tr>
<td>8</td>
<td>313 KHz</td>
</tr>
<tr>
<td>9</td>
<td>156 KHz</td>
</tr>
<tr>
<td>10</td>
<td>78 KHz</td>
</tr>
<tr>
<td>11</td>
<td>39 KHz</td>
</tr>
<tr>
<td>12</td>
<td>19.5 KHz</td>
</tr>
<tr>
<td>13</td>
<td>9.77 KHz</td>
</tr>
<tr>
<td>14</td>
<td>4.88 KHz</td>
</tr>
</tbody>
</table>

bits = 12  'try different values from table here
fbpin = 2   'feedback pin (BPIN)
adcpin = 7  'feedin pin (APIN)

OBJ
txt : "VGA_Text"

VAR long value

PUB go
txt.start(16)
txt.out(@asm_entry, @value)  'launch assembly program into a COG
txt.out($00)  'clear the screen
repeat
  waitcnt(40_000_000 + cnt)  'wait 1/2 second until updating
  txt.out($00)  'clear the screen
  txt.dec(value)  'write the value to the screen

DAT
  Assembly program
  org
asm_entry outputs
  mov   dira,asm_dira  'make pins 8 (ADC) and 0 (DAC)
movs  ctra,#adcpin  'POS W/FEEDBACK mode for CTRA
movd  ctra,#fbpin
movi  ctra,#%01001_000
mov   frqa,#1

mov   asm_cnt,cnt  'prepare for WAITCNT loop
add   asm_cnt,asm_cycles

:loop
  waitcnt asm_cnt,asm_cycles  'wait for next CNT value
  mov   asm_sample,phsa  'capture PHSA and get difference
  sub   asm_sample,asm_old
add       asm_old,asm_sample
wrlong    asm_sample, par                 'write the value to main memory
jmp       #:loop                          'wait for next sample period

' Data
asm_cycles long |< bits - 1                     'sample time
asm_dira  long |< fbpin                        'output mask
asm_cnt   res 1
asm_old   res 1
asm_sample res 1

The program establishes the mode and pins to be used and simply waits the number of
cycles equal to the maximum value obtainable for the measured voltage \(2^{\text{bits}}-1\) and takes
the difference of PHSA from the last time it was read to obtain the value. This value is
written to main memory which is used by the Spin method \text{go} to display the value on a VGA
monitor. Bases other than powers of two can be achieved by waiting the appropriate
number of cycles, for instance waiting 100 cycles will yield a result which represents a
percentage. The counter was set up using a different means than in the previous examples
by using movs, movd and movi. This is to illustrate another means for setting up the counter.
If this method is used, be sure to leave movi as the last instruction of the sequence since
the counter will begin immediately after this instruction.

Modes %0101X (POSEDGE) and %0111X (NEGEDGE) detect the edge of a signal on APIN.
This is useful in frequency counting applications as shown in the following example.

```
''Demonstration of the counter used as a frequency counter

CON
   _clkmode = xtal1 + pll16x
   _XinFREQ = 5.000.000

OBJ
   txt : "VGA_Text"

VAR
   long ctr, frq

PUB Go | freq
   txt.start(16)
   cognew(@entry, @freq)
   repeat
      txt.out($00)                   'clear the screen
      txt.dec(freq)                 'display the value (in Hz)

DAT
   org

entry mov     ctra, ctra_             'establish mode and start counter
mov     frqa, #1                'increment for each edge seen
mov     cnt_, cnt               'setup time delay
add     cnt_, cntadd
:loop   waitcnt cnt_, cntadd            'wait for next sample
mov     new, phsa               'record new count
mov     temp, new               'make second copy
sub     new, old                'get delta
mov     old, temp               'set next delta's base
```
The program counts every positive edge on APIN and displays the number of edges seen in one second, so the output displayed on the VGA monitor is the frequency of the signal in Hz.

## Conclusion

As this document and examples have shown, the counters contained within the Propeller are very powerful and capable of simplifying many counter based functions. With the 32 modes of operation, each individual counter may act as a:

- Waveform generation (square, saw-tooth, sinusoid, audio)
- PWM driver for servo and motor control, LED fading
- Digital to Analog conversion
- Analog to Digital conversion
- Frequency counting
- Event counting (measuring pulse widths such as servo to PWM conversion)
- RF carrier generation up to 128 MHz
- and many other applications

The counters permit the design of complex systems by offloading computation from the cog. This enables the cog to perform other tasks when dynamic manipulation of the counter is not necessary, and achieves higher bandwidth when there is dynamic manipulation of the counter.

## Resources

Download the example code listed here from this application note’s web page: www.parallaxsemiconductor.com/an001. The CTR.spin and VGA_Text.spin objects are included with the Propeller Tool software library. The Propeller Tool is a free download from www.parallaxsemiconductor.com/software.

## Revision History

Version 2.0: Title updated from “Propeller Counters.” Updated to Parallax Semiconductor contact information and disclaimer. Updated Figure 1, Figure 2, Figure 8, Figure 9, Figure 12 and Figure 13. NCO/PWM mode now referred to as simply NCO mode. Footnotes unbound and placed inline with text.