599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Sales:sales@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com

Propeller Application Note Series

AN001 - Propeller Counters v1.0

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.

Block Diagram

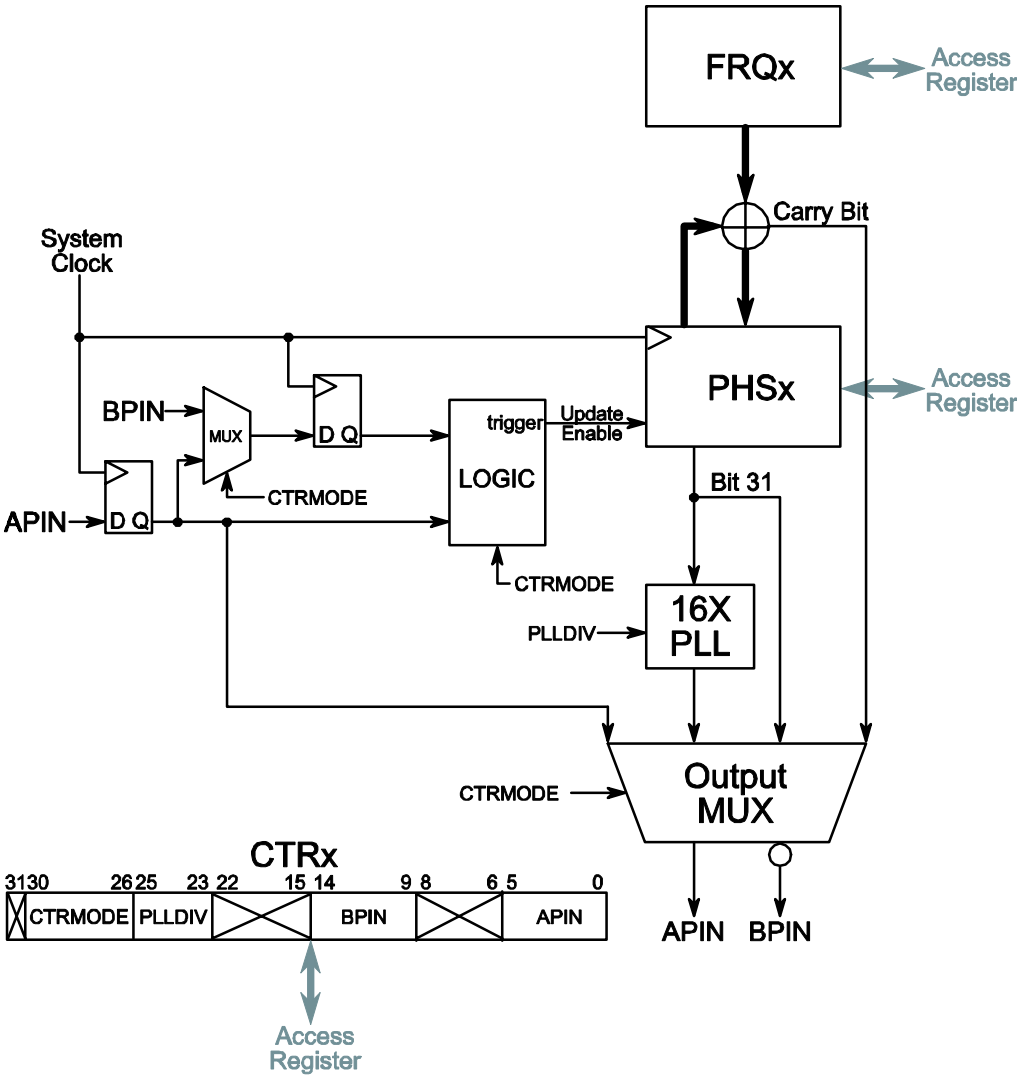

Figure 1: Counter Block Diagram

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

31	30..26	25..23	22..15	14..9	8..6	5..0
-	CTRMODE	PLLDIV	-	BPIN	-	APIN

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 P8X32A Propeller. Both counters have their own APIN and BPIN and are specified uniquely in their CTRA / CTRB register.

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.

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 5MHz crystal with a 16x PLL multiplier, the counter will be conditionally accumulated 80 million times a second (16 x 5MHz = 80MHz). 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)				
CTRMODE	**Description**	**Accumulate FRQx to PHSx**	**APIN Output***	**BPIN Output***
%00000	Counter disabled (off)	0 (never)	0 (none)	0 (none)
%00001 %00010 %00011	PLL internal (video mode) PLL single-ended PLL differential	1 (always) 1 1	0 PLLx PLLx	0 0 !PLLx
%00100 %00101	NCO/PWM single-ended NCO/PWM differential	1 1	PHSx[31] PHSx[31]	0 !PHSx[31]
%00110 %00111	DUTY single-ended DUTY differential	1 1	PHSx-Carry PHSx-Carry	0 !PHSx-Carry
%01000 %01001 %01010 %01011	POS detector POS detector with feedback POSEDGE detector POSEDGE detector w/ feedback	A^1 A^1 A^1 & $!A^2$ A^1 & $!A^2$	0 0 0 0	0 !A1 0 !A1
%01100 %01101 %01110 %01111	NEG detector NEG detector with feedback NEGEDGE detector NEGEDGE detector w/ feedback	$!A^1$ $!A^1$ $!A^1$ & A^2 $!A^1$ & A^2	0 0 0 0	0 !A1 0 !A1
%10000 %10001 %10010 %10011 %10100 %10101 %10110 %10111 %11000 %11001 %11010 %11011 %11100 %11101 %11110 %11111	LOGIC never LOGIC !A & !B LOGIC A & !B LOGIC !B LOGIC !A & B LOGIC !A LOGIC A <> B LOGIC !A | !B LOGIC A & B LOGIC A == B LOGIC A LOGIC A | !B LOGIC B LOGIC !A | B LOGIC A | B LOGIC always	0 $!A^1$ & $!B^1$ A^1 & $!B^1$ $!B^1$ $!A^1$ & B^1 $!A^1$ A^1 <> B^1 $!A^1$ | $!B^1$ A^1 & B^1 A^1 == B^1 A^1 A^1 | $!B^1$ B^1 $!A^1$ | B^1 A^1 | B^1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

***Must set corresponding DIR bit to affect pin**
A^1 = APIN input delayed by 1 clock
A^2 = APIN input delayed by 2 clocks
B^1 = BPIN input delayed by 1 clock

Table 3: Counter Modes Application Examples		
CTRMODE*	**Description**	**Example Applications**
%0001X	PLL	RF carrier synthesis, clock generation
%0010X	NCO/PWM	Servo control, PWM motor control, LED dimming, Audio generation
%0011X	DUTY	Digital to Analog Conversion, Audio generation
%01X00	POS/NEG detector	Pulse width measurement, Duty cycle measurement
%01X01	POS/NEG detector w/ feedback	Analog to Digital Conversion
%01X1X	POSEDGE/NEGEDGE	Event counter, Frequency measurement
%1XXXX	LOGIC	Propagation Delay measurement, Long duration event timer

*** X = either a 0 or 1; multiple modes are applicable**

NCO/PWM modes of operation

Modes %00100 (NCO/PWM single-ended) and %00101 (NCO/PWM differential) specify the counter should operate in a Numerically Controlled Oscillator (NCO) or Pulse Width Modulation (PWM) 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.

Block Diagram

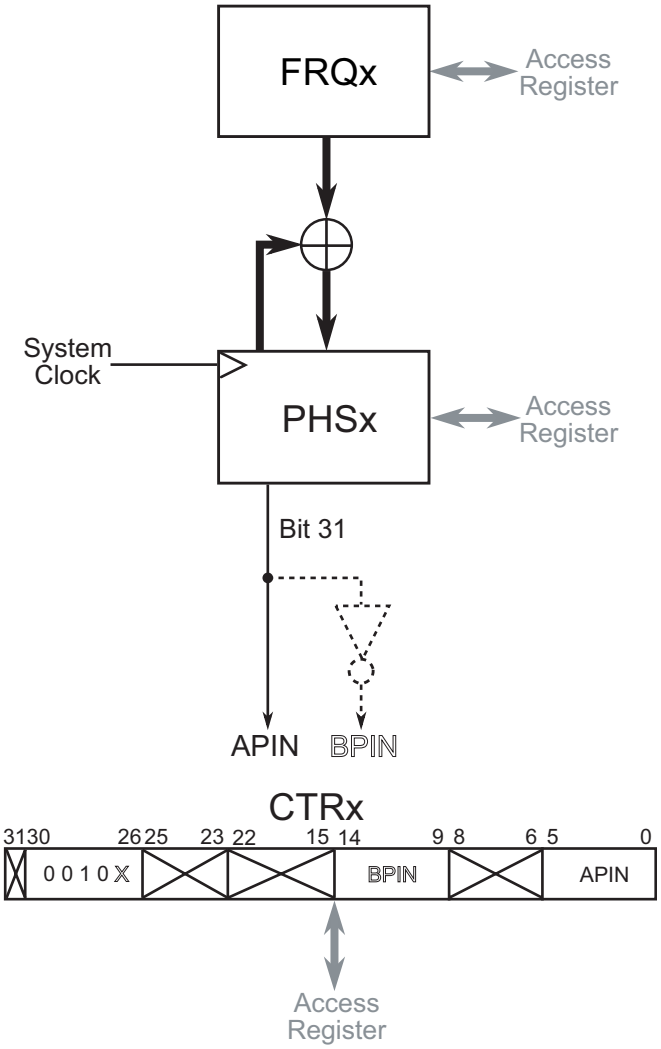

Figure 2: NCO/PWM Mode Block Diagram

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

```
''Demonstration of NCO/PWM 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/PWM State Progression

Clock Cycle	PHSA value	APIN
0	$0000_0000	0
1	$8000_0000	1
2	$0000_0000	0
3	$8000_0000	1
4	$0000_0000	0
5	$8000_0000	1
6	$0000_0000	0

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.

Oscilloscope of Single Output

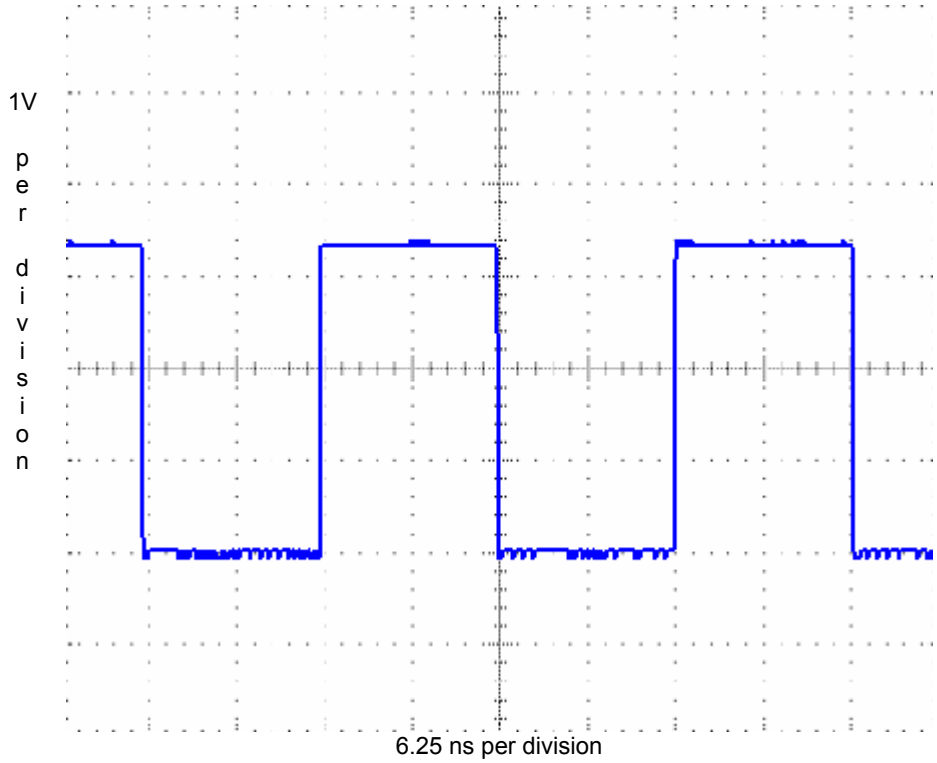

Figure 3: NCO/PWM Mode %00100 Output

If the mode were changed to %00101 (NCO/PWM 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.

Oscilloscope of Differential Output

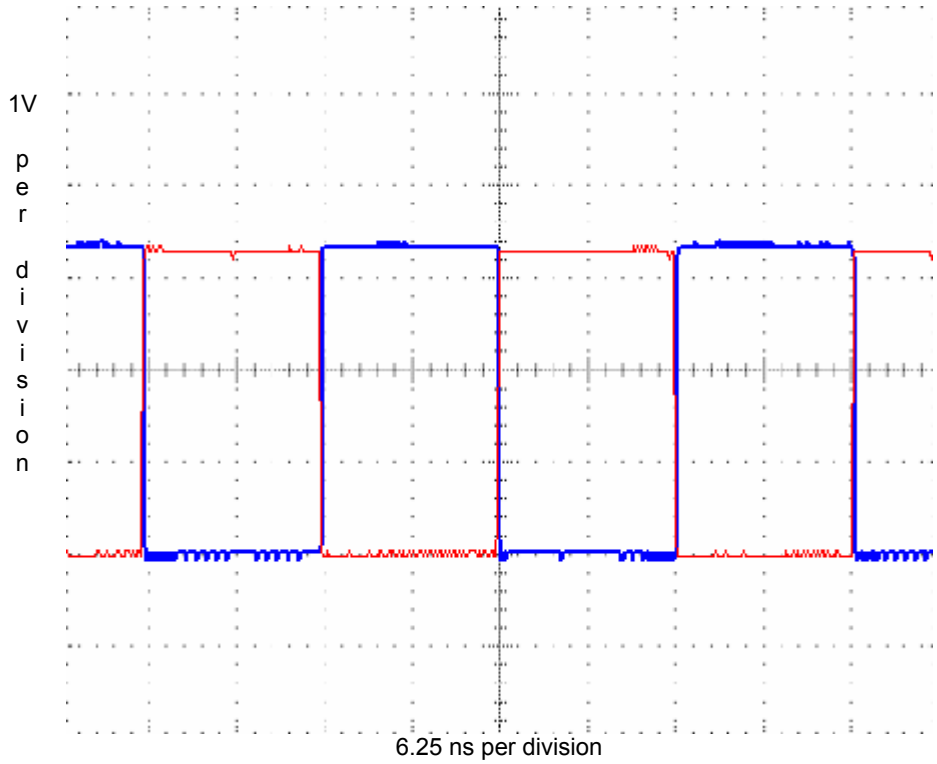

Figure 4: NCO/PWM Mode %00101 Output

If the master clock is running at 80MHz, the resultant output on APIN will be 40MHz or ½ 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 ¼ the system clock.

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

$$f_{Hz} = \frac{FRQA}{2^{32}} \cdot System\ Frequency \qquad \textbf{Equation 1}$$

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.

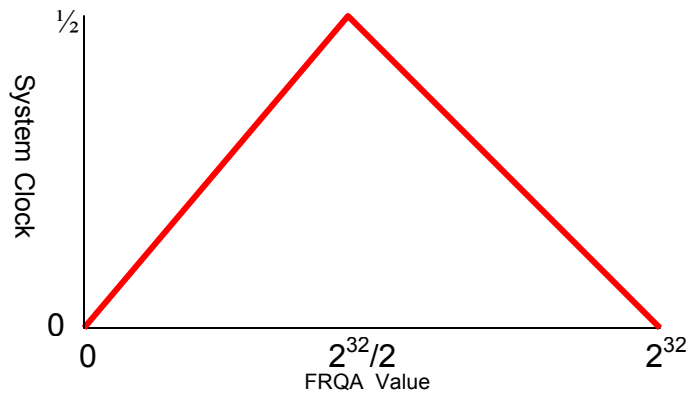

Figure 5: Output Frequencies as a Function of FRQA

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/PWM 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 perecentage
        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
period  long 100                        '800kHz period (_clkfreq / period)
time    res 1
value   res 1
```

The spin method `go` 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.

Oscilloscope of Scaling PWM

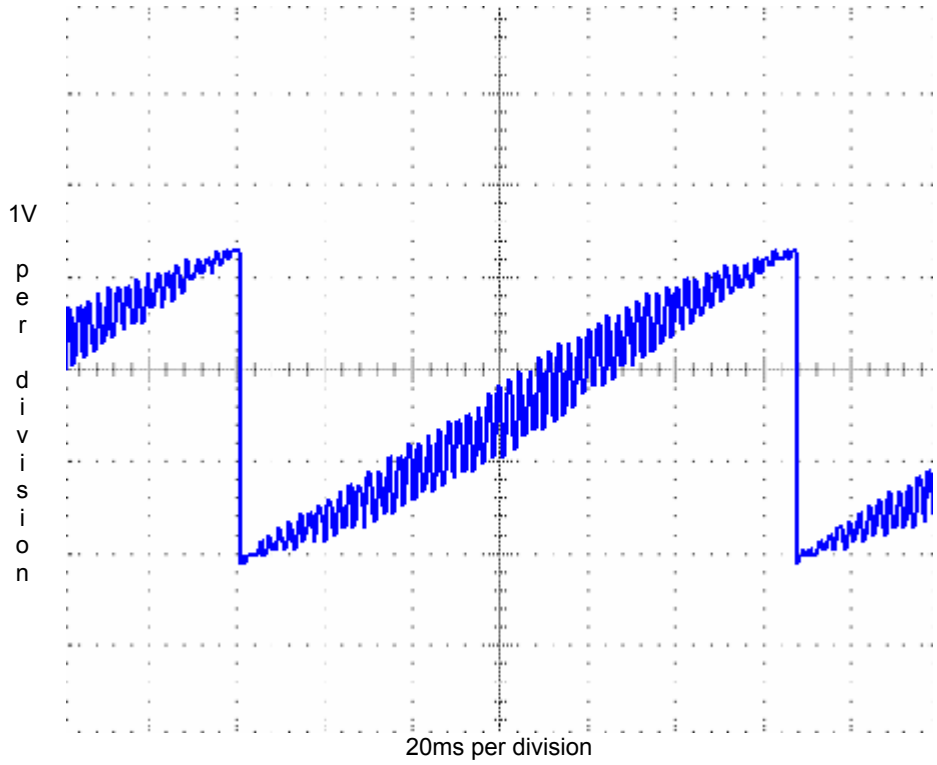

Figure 6: Scaling PWM Output

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/PWM 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/PWM single-ended) with the addition of the PLL, mode %00011 (PLL differential) is similar to mode %00101 (NCO/PWM 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.

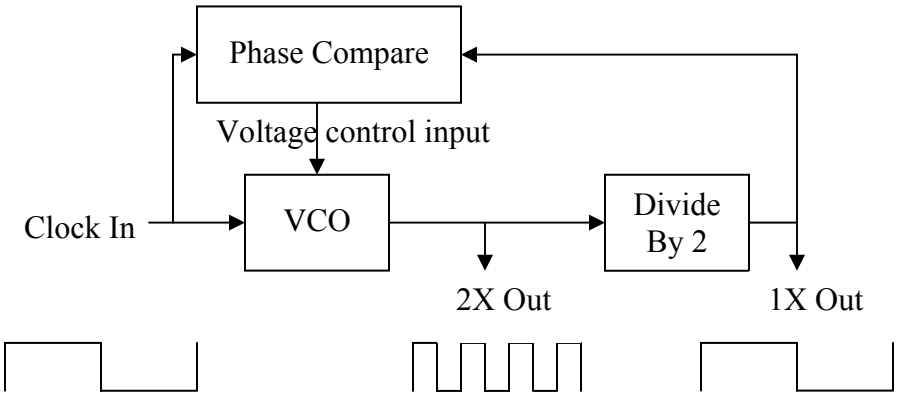

Figure 7: Block Diagram of a PLL

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, $\frac{1}{2}$ X, $\frac{1}{4}$ X and $\frac{1}{8}$ X 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

PLLDIV	%000	%001	%010	%011	%100	%101	%110	%111
Output	VCO ÷ 128	VCO ÷ 64	VCO ÷ 32	VCO ÷ 16	VCO ÷ 8	VCO ÷ 4	VCO ÷ 2	VCO ÷ 1

The block diagram for the mode is shown in Figure 8.

Block Diagram

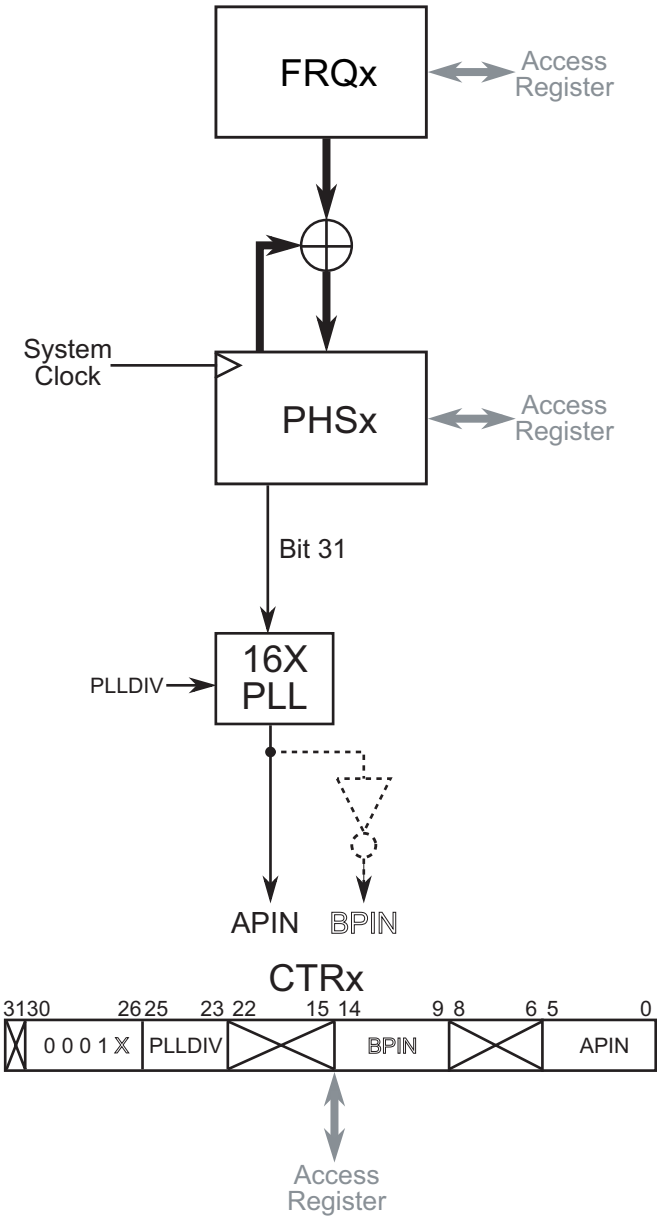

Figure 8: PLL Mode Block Diagram

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.[1]

[1] The number of bits that encompass the most significant bit and the least significant bit (leftmost 1 and rightmost 1) of FRQA value determines the amount of jitter. The PLL is capable of de-jittering the input signal up to a certain amount, which is determined by how spectrally pure the output signal must be for the application under development.

Duty Cycle modes of operation

Modes %00110 (DUTY single-ended) and %00111 (DUTY differential) on the surface appear similar to the NCO/PWM 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.

Block Diagram

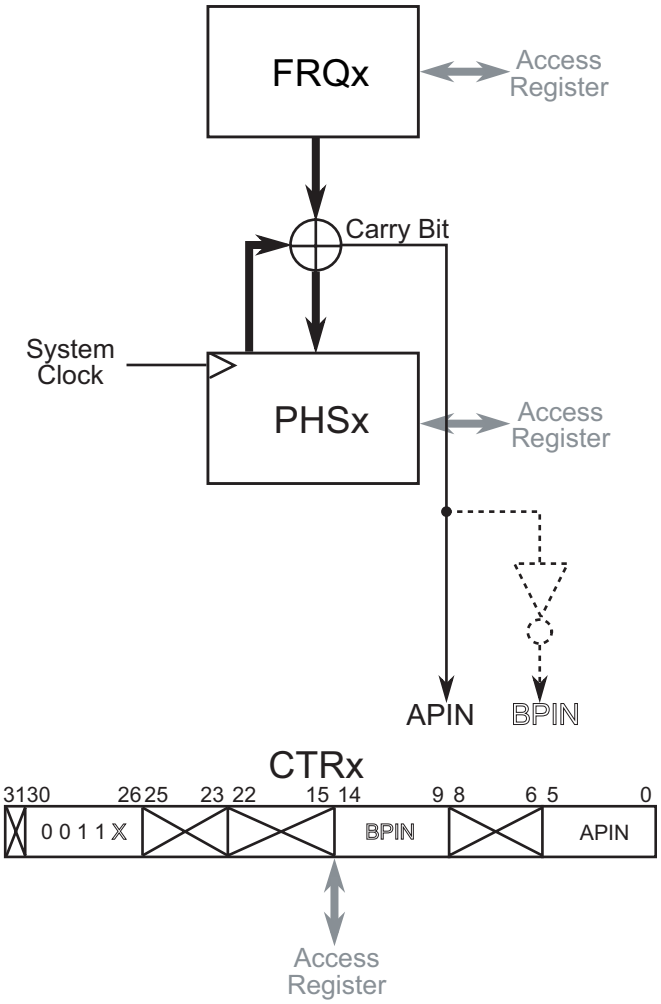

Figure 9: Duty Cycle Mode Block Diagram

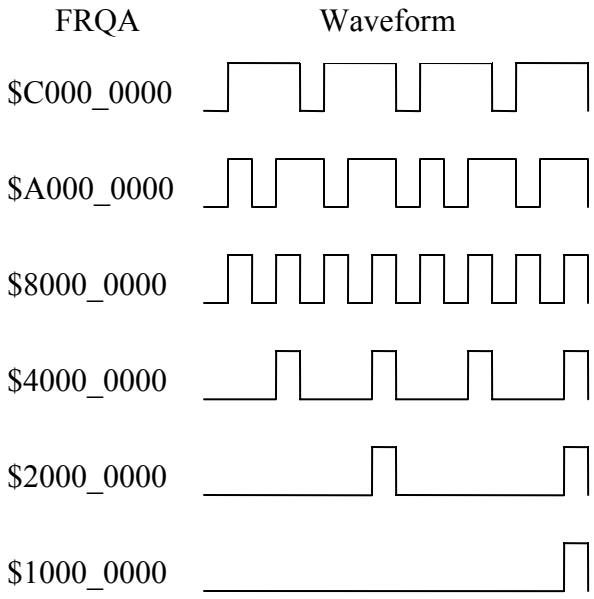

Figure 10: Examples of Duty Cycle Mode Output

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[2]. Only when FRQA is $8000_0000 do the Duty Cycle and NCO/PWM 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.

$$V = 3.3 \times \frac{FRQA}{2^{32}}$$

Equation 2

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

```
{{
   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
```

[2] 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.

```
PUB go | x
  cognew(@entry, @parameter)          'startup DAC cog and point to DAC value
  repeat
    repeat x from 0 to period         'loop over the entire scale
      parameter := $20C49B * x        '$1_0000_0000 / period * x <- provides full scale voltage
      waitcnt(1000 +cnt)              'wait awhile before changing the value

DAT
        org

entry   mov dira, diraval                       'set APIN to output
        mov ctra, ctraval                       'establish counter A mode and APIN

        mov time, cnt                           'record current time
        add time, period                        'establish next period

:loop   rdlong value, par                       'get an up to date duty cycle
        waitcnt time, period                    'wait until next period
        mov  frqa, value                        'update the duty cycle
        jmp #:loop                              'do it again

diraval long |< 0                               'APIN direction
ctraval long %00111 << 26 + 0                   'NCO/PWM APIN=0 {BPIN=1} <-not used
period  long 2000                               '40kHz period (_clkfreq / period)
time    res 1
value   res 1
```

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

Oscilloscope of Scaling Duty Cycle

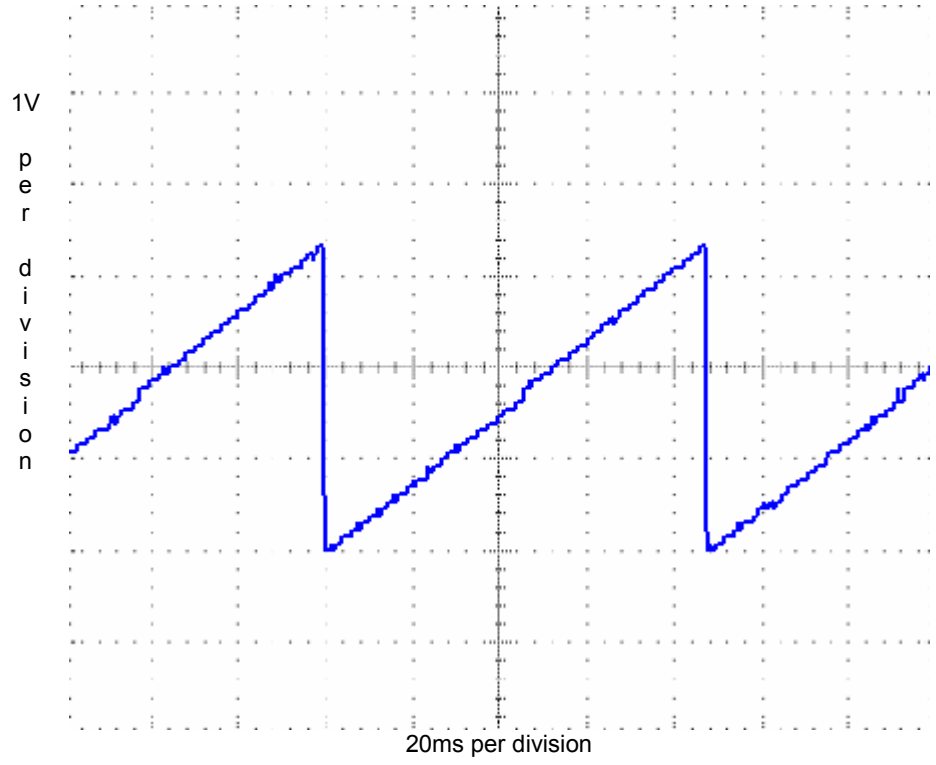

Figure 11: Scaling Duty Cycle Output

The voltage presented at `Out` of the circuit is `(3.3 × x)/period` due to scaling the input value by $20C49B.

Logic modes of operation

Modes %10000 through %11111 operate differently that 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 6: Logic Mode Equations

Mode	Accumulates each cycle
%10001	APIN=0 and BPIN=0
%10010	APIN=1 and BPIN=0
%10011	BPIN=0
%10100	APIN=0 and BPIN=1
%10101	APIN=0
%10110	APIN ≠ BPIN
%10111	APIN=0 or BPIN=0
%11000	APIN=1 and BPIN=1
%11001	APIN=BPIN
%11010	APIN=1
%11011	APIN=1 or BPIN=0
%11100	BPIN=1
%11101	APIN=0 or BPIN=1
%11110	APIN=1 or BPIN=1

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 80MHz 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.[3]

The Block diagram for modes %10000 through %11111 is shown in Figure 12.

[3] 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.

Block Diagram

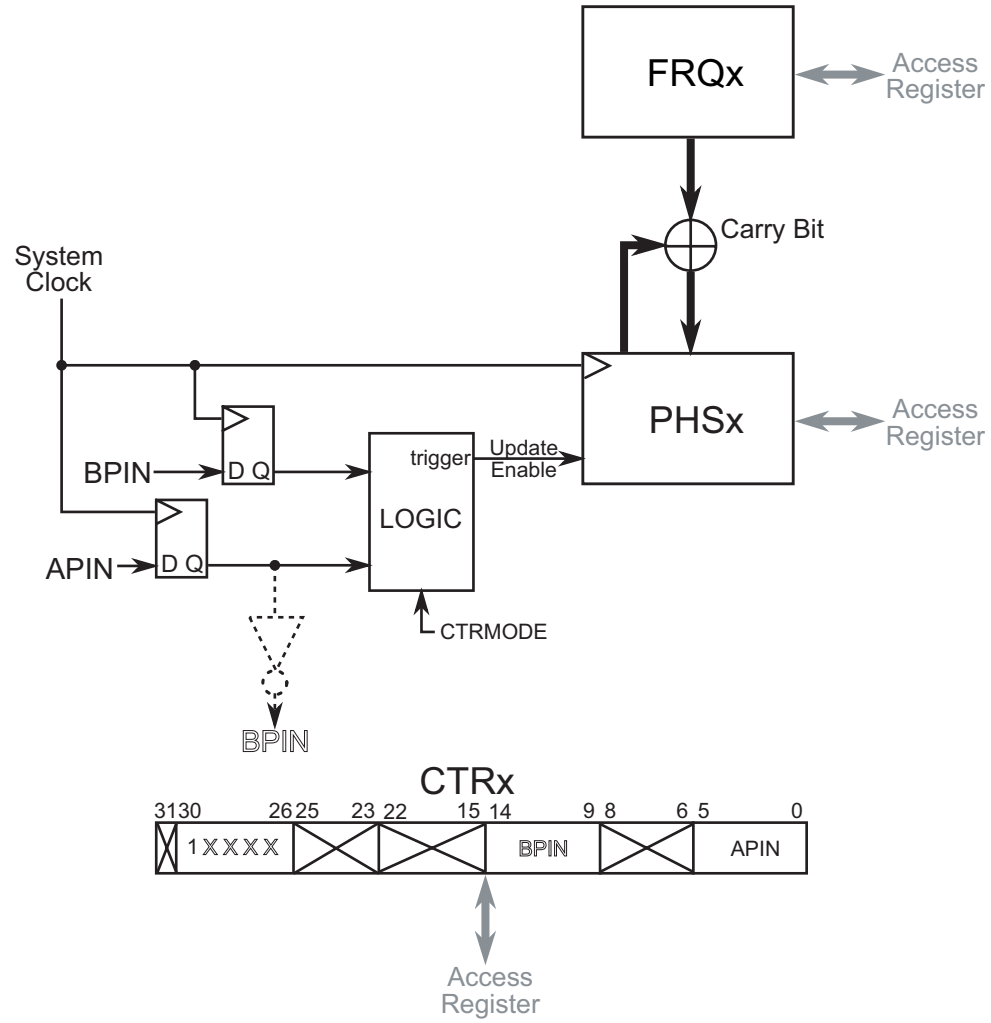

Figure 12: Logic Mode Block Diagram

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		
Mode	**Accumulates each cycle**	**Feedback to BPIN**
%01000	APIN=1	no
%01001	APIN=1	yes
%01010	APIN = _/‾ *	no
%01011	APIN = _/‾ *	yes
%01100	APIN=0	no
%01101	APIN=0	yes
%01110	APIN = ‾_ **	no
%01111	APIN = ‾_ **	yes

*** 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.

Block Diagram

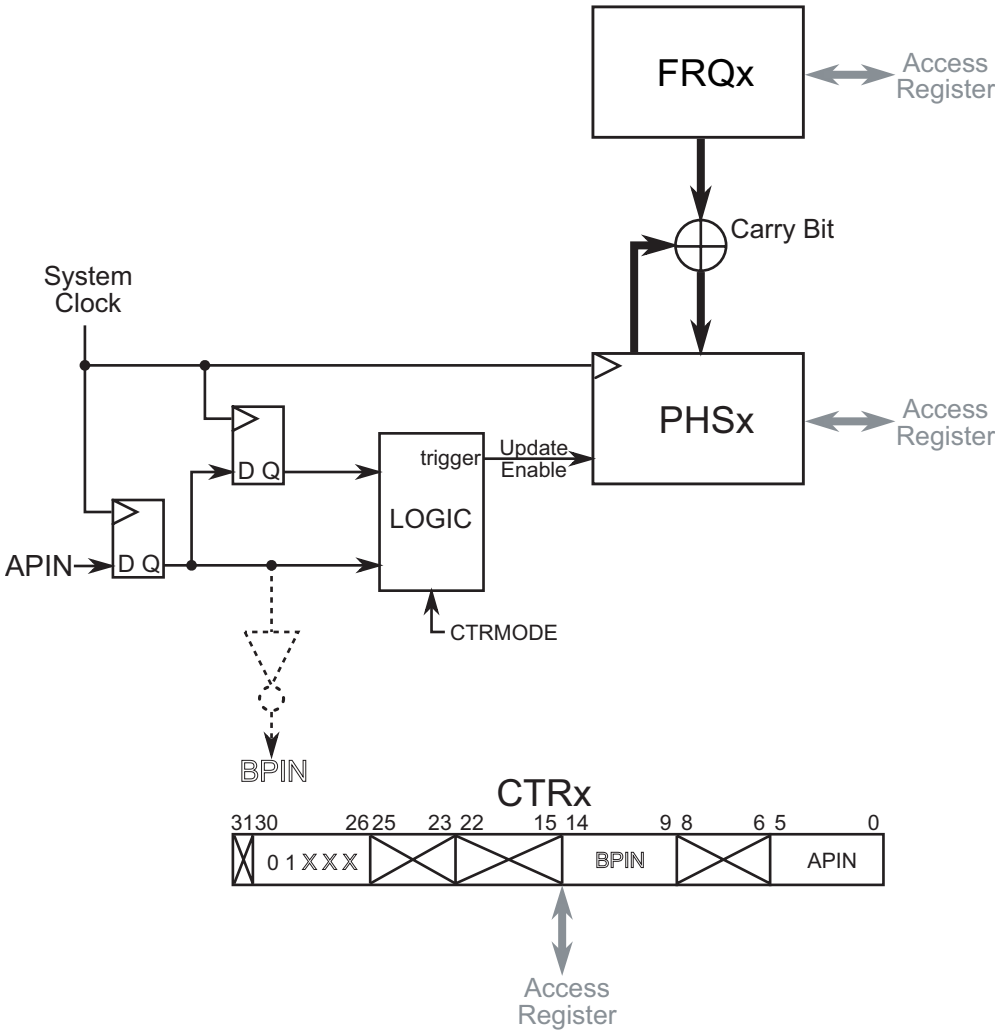

Figure 13: Pin State Detection Mode Block Diagram

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.

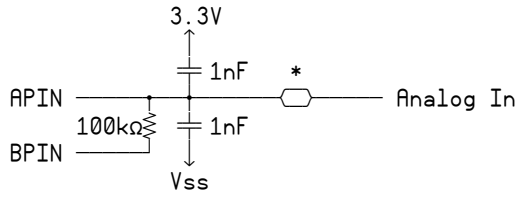

Figure 14: ΣΔ-ADC external circuit

The two 1nF capacitors and the 100kΩ 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 ΣΔ-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
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

' At 80MHz the ADC/DAC sample resolutions and rates are as follows:
' sample   sample
' bits       rate
' ----------------
' 5       2.5 MHz
' 6      1.25 MHz
' 7       625 KHz
' 8       313 KHz
' 9       156 KHz
' 10       78 KHz
' 11       39 KHz
' 12     19.5 KHz
' 13     9.77 KHz
' 14     4.88 KHz

  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)
  cognew(@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     mov       dira,asm_dira                  'make pins 8 (ADC) and 0 (DAC) outputs

              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^{nbits}-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 `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

        wrlong  new, par
        jmp     #:loop

ctra_   long    %01010 << 26 + 7        'mode + APIN
cntadd  long    80_000_000              'wait 1 second, answer in Hz
cnt_    res     1                       'next count to wait on
new     res     1
old     res     1
temp    res     1
```

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 128MHz
- 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.