Parallax Propeller 2 Documentation

SMART PINS

Each I/O pin has a ‘smart pin’ circuit which, when enabled, performs some autonomous function on the pin. Smart pins free the cogs from needing to micro-manage many I/O operations by providing high-bandwidth hardware functions which cogs could not perform on their own by manipulating DIR and OUT bits and reading IN bits.

Normally, an I/O pin’s output enable is controlled by its DIR bit and its output state is controlled by its OUT bit, while the IN bit returns the read state. In smart pin modes, the DIR bit is used as an active-low reset signal to the smart pin circuitry, while the output enable state is set by a configuration bit. In some modes, the smart pin takes over driving the output state, in which case the OUT bit is ignored. The IN bit gets used as a flag to indicate to the cog(s) when the smart pin has completed some function or an event has occurred.

Smart pins have four 32-bit registers inside of them:

mode - smart pin mode, as well as low-level I/O pin mode (write-only)
X - mode-specific parameter (write-only)
Y - mode-specific parameter (write-only)
Z - mode-specific result (read-only)

These four registers are written and read via the following 2-clock instructions, in which S/# is used to select the pin number (0..63) and D/# is the data conduit:

```
WRPIN   D/#,S/#     - Set smart pin S/# mode to D/#, acknowledge pin
WXPIN   D/#,S/#     - Set smart pin S/# parameter X to D/#, acknowledge pin
WYPIN   D/#,S/#     - Set smart pin S/# parameter Y to D/#, acknowledge pin
RDPIN   D,S/# {WC}  - Get smart pin S/# result Z into D, flag into C, acknowledge pin
RQPIN   D,S/# {WC}  - Get smart pin S/# result Z into D, flag into C, no acknowledge

AKPIN   S/#         - Acknowledge pin S/#
```

Each cog has a 34-bit bus to each smart pin for WRPIN/WXPIN/WYPIN data and RDPIN/AKPIN acknowledge signalling. Each smart pin OR’s all incoming 34-bit buses from the cogs in the same way DIR and OUT bits are OR’d before going to the pins. Therefore, if you intend to have multiple cogs execute WRPIN/WXPIN/WYPIN/RDPIN/AKPIN instructions on the same smart pin, you must be sure that they do so at different times, in order to avoid clobbering each other’s bus data.

Each smart pin has an outgoing 33-bit bus which conveys its Z result and a special flag. RDPIN and RQPIN are used to multiplex and read read these buses, so that a pin’s Z result is read into D and its special flag can be read into C. C will be either a mode-related flag or the MSB of the Z result.

Any number of cogs can read a smart pin simultaneously, without bus conflict, by using RQPIN (‘read quiet’), since it does not utilize the 34-bit cog-to-smart-pin bus for acknowledge signalling, like RDPIN does.

For the WRPIN instruction, which establishes both the low-level and smart-pin configuration for each I/O pin:

```
D/# = %AAAA_BBBB_FFF_PPPPPPPPPPPPP_TT_MMMMM_0

%AAAA:      'A' input selector
            0xxx = true (default)
            1xxx = inverted
            x000 = this pin's read state (default)
            x001 = relative +1 pin's read state
            x010 = relative +2 pin's read state
            x011 = relative +3 pin's read state
            x100 = this pin's OUT bit from cogs
            x101 = relative -3 pin's read state
            x110 = relative -2 pin's read state
            x111 = relative -1 pin's read state

%BBBB:      'B' input selector
            0xxx = true (default)
            1xxx = inverted
            x000 = this pin's read state (default)
            x001 = relative +1 pin's read state
            x010 = relative +2 pin's read state
            x011 = relative +3 pin's read state
            x100 = this pin's OUT bit from cogs
            x101 = relative -3 pin's read state
            x110 = relative -2 pin's read state
            x111 = relative -1 pin's read state

 %FFF:      'A' and 'B' input logic/filtering (after 'A' and 'B' input selectors)
            000 = A, B (default)
            001 = A AND B, B
            010 = A OR  B, B
            011 = A XOR B, B
            100 = A, B, filtered 3-of-3 at clock/1
            101 = A, B, filtered 3-of-3 at clock/8
            110 = A, B, filtered 3-of-3 at clock/64
            111 = A, B, filtered 3-of-3 at clock/512

            The resultant 'A' will drive the IN signal in non-smart-pin modes.

%P..P:      low-level pin control (needs final silicon to fully operate)
            %0000CIOHHHLLL = digital mode (default = %0000000000000)
                     %C: 1 = clocked I/O (extra clock for IN and OUT)
                     %I: 1 = invert IN output
                     %O: 1 = invert OUT input
                   %HHH: 000 = drive high, other = float when driven high
                   %LLL: 000 = drive low,  other = float when driven low
            %101xxDDDDDDDD = DAC mode, %DDDDDDDD: DAC output level

  %TT:      pin DIR/OUT control (default = %00)
```

```
                For odd pins,  'OTHER' = NOT lower pin's output state (diff source).
                For even pins, 'OTHER' = unique pseudo-random bit (noise source).
                For all pins,  'SMART' = smart pin output which overrides OUT/OTHER.
                'DAC_MODE' is enabled when P[12:10] = %101.
                'BIT_DAC' overrides P[7:0] with $00 during 'low' output in DAC_MODE.

                for smart pin off (%MMMMM = %00000):
                    DIR enables output
                    for non-DAC_MODE:
                        0x = OUT drives output
                        1x = OTHER drives output
                    for DAC_MODE:
                        00 = OUT enables ADC, P[7:0] sets DAC level
                        01 = OUT enables ADC, cog DAC channel overrides P[7:0]
                        10 = OUT drives BIT_DAC
                        11 = OTHER drives BIT_DAC

                for all smart pin modes (%MMMMM > %00000):
                    x0 = output disabled, regardless of DIR
                    x1 = output enabled, regardless of DIR

                for DAC smart pin modes (%MMMMM = %00001..%00011):
                    0x = OUT enables ADC in DAC_MODE, P[7:0] overriden by smart pin
                    1x = OTHER enables ADC in DAC_MODE, P[7:0] overriden by smart pin

                for non-DAC smart pin modes (%MMMMM = %00100..%11111):
                    0x = SMART/OUT drives output or BIT_DAC if DAC_MODE
                    1x = SMART/OTHER drives output or BIT_DAC if DAC_MODE

%MMMMM:         00000   = smart pin off (default)
                00001   = long repository              (P[12:10] != %101)
                00010   = long repository              (P[12:10] != %101)
                00011   = long repository              (P[12:10] != %101)
                00001   = DAC noise                    (P[12:10]  = %101)
                00010   = DAC 16-bit dither, noise     (P[12:10]  = %101)
                00011   = DAC 16-bit dither, PWM       (P[12:10]  = %101)
                00100*  = pulse/cycle output
                00101*  = transition output
                00110*  = NCO frequency
                00111*  = NCO duty
                01000*  = PWM triangle
                01001*  = PWM sawtooth
                01010*  = PWM switch-mode power supply, V and I feedback
                01011   = periodic/continuous, A-B quadrature encoder
                01100   = periodic/continuous, inc on A-high
                01101   = periodic/continuous, inc on A-rise
                01110   = periodic/continuous, inc on A-high, dec on B-high
                01111   = periodic/continuous, inc on A-rise, dec on B-rise
```

```
10000   = time A states
10001   = time A-high states
10010   = time X A-highs
10011   = for X periods, count time
10100   = for X periods, count states
10101   = for periods in X+ clocks, count time
10110   = for periods in X+ clocks, count states
10111   = for periods in X+ clocks, count periods
11000*  = USB host, low-speed          (even/odd pin pair = DM/DP)
11001*  = USB host, high-speed         (even/odd pin pair = DM/DP)
11010*  = USB device, low-speed        (even/odd pin pair = DM/DP)
11011*  = USB device, high-speed       (even/odd pin pair = DM/DP)
11100*  = sync serial transmit         (A-data, B-clock)
11101   = sync serial receive          (A-data, B-clock)
11110*  = async serial transmit        (baudrate)
11111   = async serial receive         (baudrate)

* OUT signal overridden
```

When a mode-related event occurs in a smart pin, it raises its IN signal to alert the cog(s) that new data is ready, new data can be loaded, or some process has finished. A cog acknowledges a smart pin whenever it does a WRPIN, WXPIN, WYPIN, RDPIN or AKPIN on it. This causes the smart pin to lower its IN signal so that it can be raised again on the next event. Note that since the RQPIN instruction (read quiet) does not do an acknowledge, it can be used by any number of cogs, concurrently, to read a pin without bus conflict.

After WRPIN/WXPIN/WYPIN/RDPIN/AKPIN, it will take two clocks for IN to drop, before it can be polled again:

```
WRPIN/WXPIN/WYPIN/RDPIN/AKPIN   'acknowledge smart pin, releases IN from high
NOP                             'elapse 2 clocks (or more)
TESTIN  pin     WC              'IN can now be polled again
```

Smart pins should be configured while their DIR signal is low, holding them in reset. During that time, WRPIN/WXPIN/WYPIN can be used to establish the mode and related parameters. Once configured, DIR can be raised high and the smart pin will begin operating. After that, depending on the mode, you may feed it new data via WXPIN/WYPIN or retrieve results using RDPIN/RQPIN. These activities are usually coordinated with the IN signal going high.

To return a pin to normal mode, do a 'WRPIN #0,pin'.

SMART PIN MODES

%00001..%00011 and not DAC_MODE = long repository

This mode turns the smart pin into a long repository, where WXPIN sets the long and RDPIN/RQPIN returns the long.

Upon each WXPIN, IN is raised.

%00001 and DAC_MODE = DAC noise

This mode overrides P[7:0] to feed the pin's 8-bit DAC unique pseudo-random data on every clock. P[12:10] must be set to %101 to configure the low-level pin for DAC output.

X[15:0] can be set to a sample period in clock cycles, in case you want to mark time with IN raising at each period completion. If a sample period is not wanted, set X[15:0] to zero (65,536 clocks), in order to maximize the unused sample period, thereby reducing switching power.

RDPIN/RQPIN can be used to retrieve the 16-bit ADC accumulation from the last sample period.

During reset (DIR=0), IN is low.

%00010 and DAC_MODE = DAC 16-bit pseudo-random dither

This mode overrides P[7:0] to feed the pin's 8-bit DAC pseudo-randomly-dithered data on every clock. P[12:10] must be set to %101 to configure the low-level pin for DAC output.

X[15:0] establishes the sample period in clock cycles.

Y[15:0] establishes the output value which gets captured at each sample period and used for its duration.

On completion of each sample period, Y[15:0] is captured for the next output value and IN is raised. Therefore, you would coordinate updating Y[15:0] with IN going high.

Pseudo-random dithering does not require any kind of fixed period, as it randomly dithers the 8-bit DAC between adjacent levels. So, if you would like to be able to update the output value at any time and have it take immediate effect, set X[15:0] to one (IN will stay high).

RDPIN/RQPIN can be used to retrieve the 16-bit ADC accumulation from the last sample period.

During reset (DIR=0), IN is low and Y[15:0] is captured.

%00011 and DAC_MODE = DAC 16-bit PWM dither

This mode overrides P[7:0] to feed the pin's 8-bit DAC PWM-dithered data on every clock. P[12:10] must be set to %101 to configure the low-level pin for DAC output.

X[15:0] establishes the sample period in clock cycles. The sample period must be a multiple of 256 (X[7:0]=0), so that an integral number of 256 steps are afforded the PWM, which dithers the DAC between adjacent 8-bit levels.

Y[15:0] establishes the output value which gets captured at each sample period and used for its duration.

On completion of each sample period, Y[15:0] is captured for the next output value and IN is raised. Therefore, you would coordinate updating Y[15:0] with IN going high.

PWM dithering will give better dynamic range than pseudo-random dithering, since a maximum of only two transitions occur for every 256 clocks. This means, though, that a frequency of Fclock/256 will be present in the output at -48dB.

RDPIN/RQPIN can be used to retrieve the 16-bit ADC accumulation from the last sample period.

During reset (DIR=0), IN is low and Y[15:0] is captured.

%00100 = pulse/cycle output

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical cycle time.

X[31:16] establishes a value to which the base period counter will be compared to on each clock cycle, as it counts from X[15:0] down to 1, before starting over at X[15:0] if decremented Y > 0. On each clock, if the base period counter > X[31:16] and Y > 0, the output will be high (else low).

Whenever Y[31:0] is written with a non-zero value, the pin will begin outputting a high pulse or cycles, starting at the next base period.

Some examples:

If X[31:16] is set to 0, the output will be high for the duration of Y > 0.

If X[15:0] is set to 3 and X[31:16] is set to 2, the output will be 0-0-1 (repeat) for the duration of Y > 0.

IN will be raised when the pulse or cycles complete, with the pin reverting to low output.

During reset (DIR=0), IN is low, the output is low, and Y is set to zero.

%00101 = transition output

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical high and low times.

Whenever Y[31:0] is written with a non-zero value, the pin will begin toggling for Y transitions at each base period, starting at the next base period.

IN will be raised when the transitions complete, with the pin remaining in its current output state.

During reset (DIR=0), IN is low, the output is low, and Y is set to zero.

%00110 = NCO frequency

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical high and low times.

Y[31:0] will be added into Z[31:0] at each base period.

The pin output will reflect Z[31].

IN will be raised whenever Z overflows.

During reset (DIR=0), IN is low, the output is low, and Z is set to zero.

%00111 = NCO duty

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical high and low times.

Y[31:0] will be added into Z[31:0] at each base period.

The pin output will reflect Z overflow.

IN will be raised whenever Z overflows.

During reset (DIR=0), IN is low, the output is low, and Z is set to zero.

%01000 = PWM triangle

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical high and low times.

X[31:16] establishes a PWM frame period in terms of base periods.

Y[15:0] establishes the PWM output value which gets captured at each frame start and used for its duration. It should range from zero to the frame period.

A counter, updating at each base period, counts from the frame period down to one, then from one back up to the frame period. Then, Y[15:0] is captured, IN is raised, and the process repeats.

At each base period, the captured output value is compared to the counter. If it is equal or greater, a high is output. If it is less, a low is output. Therefore, a zero will always output a low and the frame period value will always output a high.

During reset (DIR=0), IN is low, the output is low, and Y[15:0] is captured.

%01001 = PWM sawtooth

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical high and low times.

X[31:16] establishes a PWM frame period in terms of base periods.

Y[15:0] establishes the PWM output value which gets captured at each frame start and used for its duration. It should range from zero to the frame period.

A counter, updating at each base period, counts from one up to the frame period. Then, Y[15:0] is captured, IN is raised, and the process repeats.

At each base period, the captured output value is compared to the counter. If it is equal or greater, a high is output. If it is less, a low is output. Therefore, a zero will always output a low and the frame period value will always output a high.

During reset (DIR=0), IN is low, the output is low, and Y[15:0] is captured.

%01010 = PWM switch-mode power supply with voltage and current feedback

This mode overrides OUT to control the pin output state.

X[15:0] establishes a base period in clock cycles which forms the empirical high and low times.

X[31:16] establishes a PWM frame period in terms of base periods.

Y[15:0] establishes the PWM output value which gets captured at each frame start and used for its duration. It should range from zero to the frame period.

A counter, updating at each base period, counts from one up to the frame period. Then, the ‘A’ input is sampled at each base period until it reads low. After ‘A’ reads low, Y[15:0] is captured, IN is raised, and the process repeats.

At each base period, the captured output value is compared to the counter. If it is equal or greater, a high is output. If it is less, a low is output. If, at any time during the cycle, the ‘B’ input goes high, the output will be low for the rest of that cycle.

Due to the nature of switch-mode power supplies, it may be appropriate to just set Y[15:0] once and let it repeat indefinitely.

During reset (DIR=0), IN is low, the output is low, and Y[15:0] is captured.

%01011 = A/B-input quadrature encoder

X[31:0] establishes a measurement period in clock cycles.

If zero is used for the period, the measurement operation will not be periodic, but continuous, like a totalizer, and the current 32-bit quadrature step count can always be read via RDPIN/RQPIN.

If a non-zero value is used for the period, quadrature steps will be counted for that many clock cycles and then the result will be placed in Z while the accumulator will be set to the 0/1/-1 value that would have otherwise been added into it. This way, all quadrature steps get counted across measurements. At the end of each period, IN will be raised and RDPIN/RQPIN can be used to retrieve the last 32-bit measurement.

It may be useful to configure both ‘A’ and ‘B’ smart pins to quadrature mode, with one being continuous (X=0) for absolute position tracking and the other being periodic (x<>0) for velocity measurement.

The quadrature encoder can be “zeroed” by pulsing DIR low at any time. There is no need to do another WXPIN.

During reset (DIR=0), IN is low and Z is set to the adder value (0/1/-1)

%01100 = Count A-input highs

X[31:0] establishes a measurement period in clock cycles.

If zero is used for the period, the measurement operation will not be periodic, but continuous, like a totalizer, and the current 32-bit high count can always be read via RDPIN/RQPIN.

If a non-zero value is used for the period, highs will be counted for that many clock cycles and then the result will be placed in Z, while the accumulator will be set to the 0/1 value that would have otherwise been added into it, beginning a new measurement. This way, all highs get counted across measurements. At the end of each period, IN will be raised and RDPIN/RQPIN can be used to retrieve the 32-bit measurement.

During reset (DIR=0), IN is low and Z is set to the adder value (0/1)

%01101 = Count A-input positive edges

X[31:0] establishes a measurement period in clock cycles.

If zero is used for the period, the measurement operation will not be periodic, but continuous, like a totalizer, and the current 32-bit positive-edge count can always be read via RDPIN/RQPIN.

If a non-zero value is used for the period, positive edges will be counted for that many clock cycles and then the result will be placed in Z while the accumulator will be set to the 0/1 value that would have otherwise been added into it, beginning a new measurement. This way, all positive edges get counted across measurements. At the end of each period, IN will be raised and RDPIN/RQPIN can be used to retrieve the 32-bit measurement.

During reset (DIR=0), IN is low and Z is set to the adder value (0/1)

%01110 = Count, A-input highs increment and B-input highs decrement

X[31:0] establishes a measurement period in clock cycles.

If zero is used for the period, the measurement operation will not be periodic, but continuous, like a totalizer, and the current 32-bit competing-highs count can always be read via RDPIN/RQPIN.

If a non-zero value is used for the period, competing highs will be counted for that many clock cycles and then the result will be placed in Z while the accumulator will be set to the 0/1/-1 value that would have otherwise been added into it, beginning a new measurement. This way, all competing highs get counted across measurements. At the end of each period, IN will be raised and RDPIN/RQPIN can be used to retrieve the 32-bit measurement.

During reset (DIR=0), IN is low and Z is set to the adder value (0/1/-1)

%01111 = Count, A-input positive edges increment and B-input positive edges decrement

X[31:0] establishes a measurement period in clock cycles.

If zero is used for the period, the measurement operation will not be periodic, but continuous, like a totalizer, and the current 32-bit competing-positive-edges count can always be read via RDPIN/RQPIN.

If a non-zero value is used for the period, competing positive edges will be counted for that many clock cycles and then the result will be placed in Z while the accumulator will be set to the 0/1/-1 value that would have otherwise been added into it, beginning a new measurement. This way, all competing positive edges get counted across measurements. At the end of each period, IN will be raised and RDPIN/RQPIN can be used to retrieve the 32-bit measurement.

During reset (DIR=0), IN is low and Z is set to the adder value (0/1/-1)

%10000 = Time A-input states

Continuous states are counted in clock cycles.

Upon each state change, the prior state is placed in the C-flag buffer, the prior state's duration count is placed in Z, and IN is raised. RDPIN/RQPIN can then be used to retrieve the measurement. Z will be limited to $80000000.

If states change faster than the cog is able to retrieve measurements, the measurements will effectively be lost, as old ones will be overwritten with new ones. This may be gotten around by using two smart pins to time highs, with one pin inverting its 'A' input. Then, you could capture both states, as long as the sum of the states' durations didn't exceed the cog's ability retrieve both results. This would help in cases where one of the states was very short in duration, but the other wasn't.

During reset (DIR=0), IN is low and Z is set to $00000001.

%10001 = Time A-input high states

Continuous high states are counted in clock cycles.

Upon each high-to-low transition, the previous high duration count is placed in Z, and IN is raised. RDPIN/RQPIN can then be used to retrieve the measurement. Z will be limited to $80000000.

During reset (DIR=0), IN is low and Z is set to $00000001.

%10010 = Time X A-input highs

Time is measured until X highs are accumulated.

X[31:0] establishes how many highs are to be accumulated.

Time is measured in clock cycles until X highs are accumulated from the input. The measurement is then placed in Z, and IN is raised. RDPIN/RQPIN can then be used to retrieve the measurement. Z will be limited to $80000000.

During reset (DIR=0), IN is low and Z is set to $00000001.

%10011 = For X periods, count time
%10100 = For X periods, count states

X[31:0] establishes how many A-input rise/edge to B-input rise/edge periods are to be measured.

Y[1:0] establishes A-input and B-input rise/edge sensitivity:

%00 = A-input rise to B-input rise
%01 = A-input rise to B-input edge
%10 = A-input edge to B-input rise
%11 = A-input edge to B-input edge

Note: The B-input can be set to the same pin as the A-input for single-pin cycle measurement.

Clock cycles or A-input trigger states are counted from each A-input rise/edge to each B-input rise/edge for X periods. If the A-input rise/edge is ever coincident with the B-input rise/edge at the end of the period, the start of the next period is registered. Upon completion of X periods, the measurement is placed in Z, IN is raised, and a new measurement begins. RDPIN/RQPIN can then be used to retrieve the completed measurement. Z will be limited to $80000000.

The first mode is intended to be used as an oversampling period measurement, while the second mode is a complementary duty measurement.

During reset (DIR=0), IN is low and Z is set to $00000000.

%10101 = For periods in X+ clock cycles, count time
%10110 = For periods in X+ clock cycles, count states
%10111 = For periods in X+ clock cycles, count periods

X[31:0] establishes the minimum number of clock cycles to track periods for. Periods are A-input rise/edge to B-input rise/edge.

Y[1:0] establishes A-input and B-input rise/edge sensitivity:

%00 = A-input rise to B-input rise
%01 = A-input rise to B-input edge
%10 = A-input edge to B-input rise
%11 = A-input edge to B-input edge

Note: The B-input can be set to the same pin as the A-input for single-pin cycle measurement.

A measurement is taken across some number of A-input rise/edge to B-input rise/edge periods, until X clock cycles elapse and then any period in progress completes. If the A-input rise/edge is ever coincident with the B-input rise/edge at the end of the period, the start of the next period is registered. Upon completion, the measurement is placed in Z, IN is raised, and a new measurement begins. RDPIN/RQPIN can then be used to retrieve the completed measurement. Z will be limited to $80000000.

The first mode accumulates time within each period, for an oversampled period measurement.

The second mode accumulates A-input trigger states within each period, for an oversampled duty measurement.

The third mode counts the periods.

Knowing how many clock cycles some number of complete periods took, and what the duty was, affords a very time-efficient and precise means of determining frequency and duty cycle. At least two of these measurements must be made concurrently to get useful results.

During reset (DIR=0), IN is low and Z is set to $00000000.

%11000 = USB host, low-speed
%11001 = USB host, full-speed
%11010 = USB device, low-speed
%11011 = USB device, full-speed

This mode requires that two adjacent pins be configured together to form a USB pair, whose OUTs will be overridden to control their output states. These pins must be an even/odd pair, having only the LSB of their pin numbers different. For example: pins 0 and 1, pins 2 and 3, pins 4 and 5, etc., can form USB pairs. They can be configured via WRPIN with identical D data of %1_110xx_0. Using D data of %0_110xx_0 will disable output drive and effectively create a USB 'sniffer'. A new WRPIN can be done to effect such a change without resetting the smart pin. **NOTE: in the current FPGA, there are no built-in 1.5k and 15k resistors, which the final silicon smart pins will contain, so it is up to you to insert these yourself on the DP and DM lines.**

The upper (odd) pin is the DP pin. This pin's IN is raised whenever the output buffer empties, signalling that a new output byte can be written via WYPIN to the lower (even) pin. No WXPIN/WYPIN instructions are used for this pin.

The lower (even) pin is the DM pin. This pin's IN is raised whenever a change of status occurs in the receiver, at which point a RDPIN/RQPIN can be used on this pin to read the 16-bit status word. WXPIN is used on this pin to set the NCO baud rate.

These DP/DM electrical designations can actually be switched by swapping low-speed and full-speed modes, due to USB's mirrored line signalling.

To start USB, clear the DIR bits of the intended two pins and configure them each via WRPIN. Use WXPIN on the lower pin to set the baud rate, which is a 16-bit fraction of the system clock. For example, if the main clock is 80MHz and you want a 12MHz baud rate (full-speed), use 12,000,000 / 80,000,000 * $10000 = 9830. Then, set the pins' DIR bits. You are now ready to read the receiver status via RDPIN/RQPIN and set output states and send packets via WYPIN, both on the lower pin.

To affect the line states or send a packet, use WYPIN on the lower pin. Here are its D values:

0 = output IDLE	- default state, float pins, except possible resistor(s) to 3.3V or GND
1 = output SE0	- drive both DP and DM low
2 = output K	- drive K state onto DP and DM (opposite)
3 = output J	- drive J state onto DP and DM (opposite), like IDLE, but driven
4 = output EOP	- output end-of-packet: SE0, SE0, J, then IDLE
$80 = SOP	- output start-of-packet, then bytes, automatic EOP when buffer runs out

To send a packet, first do a WYPIN #$80,lowerpin'. Then, after each IN rise on the upper pin, do a 'WYPIN byte,lowerpin' to buffer the next byte. The transmitter will automatically send an EOP when you stop giving it bytes. To keep the output buffer from overflowing, you should always verify that the upper pin's IN was raised after each WYPIN, before issuing another WYPIN, even if you are just setting a state. The reason for this is that all output activity is timed to the baud generator and even state changes must wait for the next bit period before being implemented, at which time the output buffer empties.

There are separate state machines for transmitting and receiving. Only the baud generator is common between them. The transmitter was just described above. Below, the receiver is detailed. Note that the receiver receives not just input from another host/device, but all local output, as well.

At any time, a RDPIN/RQPIN can be executed on the lower pin to read the current 16-bit status of the receiver, with the error

flag going into C. The lower pin's IN will be raised whenever a change occurs in the receiver's status. This will necessitate A WRPIN/WXPIN/WYPIN/RDPIN/AKPIN before IN can be raised again, to alert of the next change in status. The receiver's status bits are as follows:

[31:16]	<unused>		- $0000
[15:8]	byte		- last byte received
[7]	byte toggle		- cleared on SOP, toggled on each byte received
[6]	error		- cleared on SOP, set on bit-unstuff error, EOP SE0 > 3 bits, or SE1
[5]	EOP in		- cleared on SOP or 7+ bits of J or K, set on EOP
[4]	SOP in		- cleared on EOP or 7+ bits of J or K, set on SOP
[3]	SE1 in	(illegal)	- cleared on !SE1, set on 1+ bits of SE1
[2]	SE0 in	(RESET)	- cleared on !SE0, set on 1+ bits of SE0
[1]	K in	(RESUME)	- cleared on !K, set on 7+ bits of K
[0]	J in	(IDLE)	- cleared on !J, set on 7+ bits of J

The result of a RDPIN/RQPIN can be bit-tested for events of interest. It can also be shifted right by 8 bits to LSB-justify the last byte received and get the byte toggle bit into C, in order to determine if you have a new byte. Assume that 'flag' is initially zero:

```
        SHR     D,#8    WC      'get byte into D, get toggle bit into C
        CMPX    flag,#1 WZ      'compare toggle bit to flag, new byte if Z
IF_Z    XOR     flag,#1         'if new byte, toggle flag
IF_Z    <use byte>              'if new byte, do something with it
```

%11100 = synchronous serial transmit

This mode overrides OUT to control the pin output state.

Words of 1 to 32 bits are shifted out on the pin, LSB first, with each new bit being output two internal clock cycles after registering a positive edge on the B input. For negative-edge clocking, the B input may be inverted by setting B[3] in WRPIN's D value.

WXPIN is used to configure the update mode and word length.

X[5] selects the update mode:

> X[5] = 0 sets continuous mode, where a first word is written via WYPIN during reset (DIR=0) to prime the shifter. Then, after reset (DIR=1), the second word is buffered via WYPIN and continuous clocking is started. Upon shifting each word, the buffered data written via WYPIN is advanced into the shifter and IN is raised, indicating that a new output word can be buffered via WYPIN. This mode allows steady data transmission with a continuous clock, as long as the WYPIN's after each IN-rise occur before the current word transmission is complete.
>
> X[5] = 1 sets start-stop mode, where the current output word can always be updated via WYPIN before the first clock, flowing right through the buffer into the shifter. Any WYPIN issued after the first clock will be buffered and loaded into the shifter after the last clock of the current output word, at which time it could be changed again via WYPIN. This mode is useful for setting up the output word before a stream of clocks are issued to shift it out.

X[4:0] sets the number of bits, minus 1. For example, a value of 7 will set the word size to 8 bits.

WYPIN is used to load the output words. The words first go into a single-stage buffer before being advanced to the shifter for output. Each time the buffer is advanced into the shifter, IN is raised, indicating that a new output word can be written via WYPIN. During reset, the buffer flows straight into the shifter.

If you intend to send MSB-first data, you must first shift and then reverse it. For example, if you had a byte in D that you wanted to send MSB-first, you would do a ‘SHL D,#32-8’ and then a ‘REV D’.

During reset (DIR=0) the output is held low. Upon release of reset, the output will reflect the LSB of the output word written by any WYPIN during reset.

%11101 = synchronous serial receive

Words of 1 to 32 bits are shifted in by sampling the A input around the positive edge of the B input. For negative-edge clocking, the B input may be inverted by setting B[3] in WRPIN’s D value.

WXPIN is used to configure the sampling and word length.

X[5] selects the A input sample position relative to the B input edge:

> X[5] = 0 selects the A input sample just before the B input edge was registered. This requires no hold time on the part of the sender.
>
> X[5] = 1 selects the sample coincident with the B edge being registered. This is useful where transmitted data remains steady after the B edge for a brief time. In the synchronous serial transmit mode, the data is steady for two internal clocks after the B edge was registered, so employing this complementary feature would enable the fastest data transmission when receiving from another smart pin in synchronous serial transmit mode.

X[4:0] sets the number of bits, minus 1. For example, a value of 7 will set the word size to 8 bits.

When a word is received, IN is raised and the data can then be read via RDPIN/RQPIN. The data read will be MSB-justified.

If you received LSB-first data, it will require right-shifting, unless the word size was 32 bits. For a word size of 8 bits, you would need to do a ‘SHR D,#32-8’ to get the data LSB-justified.

If you received MSB-first data, it will need to be reversed and possibly masked, unless the word size was 32 bits. For example, if you received a 9-bit word, you would do ‘REV D’ + ‘TRIML D,#8’ to get the data LSB-justified.

%11110 = asynchronous serial transmit

This mode overrides OUT to control the pin output state.

Words from 1 to 32 bits are serially transmitted on the pin at a programmable baud rate, beginning with a low “start” bit and ending with a high “stop” bit.

WXPIN is used to configure the baud rate and word length.

X[31:16] establishes the number of clocks in a bit period, and in case X[31:26] is zero, X[15:10] establishes the number of fractional clocks in a bit period. The X bit period value can be simply computed as: (clocks * $1_0000) & $FFFFFC00. For example, 7.5 clocks would be $00078000, and 33.33 clocks would be $00215400.

X[4:0] sets the number of bits, minus 1. For example, a value of 7 will set the word size to 8 bits.

WYPIN is used to load the output words. The words first go into a single-stage buffer before being advanced to a shifter for output. This buffering mechanism makes it possible to keep the shifter constantly busy, so that gapless transmissions can be achieved. Any time a word is advanced from the buffer to the shifter, IN is raised, indicating that a new word can be loaded.

Here is the internal state sequence:

1. Wait for an output word to be buffered via WYPIN, then set the ‘buffer-full’ and ‘busy’ flags.
2. Move the word into the shifter, clear the ‘buffer-full’ flag, and raise IN.
3. Output a high for one bit period (guarantees a whole STOP bit).
4. Output a low for one bit period (the START bit).
5. Output the LSB of the shifter for one bit period, shift right, and repeat until all data bits are sent.
6. Output a high (begins the STOP bit).
7. If the ‘buffer-full’ flag is set due to an intervening WYPIN, loop to (2). Otherwise, clear the ‘busy’ flag and loop to (1).

RDPIN/RQPIN with WC always returns the ‘busy’ flag into C. This is useful for knowing when a transmission has completed.

During reset (DIR=0) the output is held high.

%11111 = asynchronous serial receive

Words from 1 to 32 bits are serially received on the A input at a programmable baud rate.

WXPIN is used to configure the baud rate and word length.

X[31:16] establishes the number of clocks in a bit period, and in case X[31:26] is zero, X[15:10] establishes the number of fractional clocks in a bit period. The X bit period value can be simply computed as: (clocks * $1_0000) & $FFFFFC00. For example, 7.5 clocks would be $00078000, and 33.33 clocks would be $00215400.

X[4:0] sets the number of bits, minus 1. For example, a value of 7 will set the word size to 8 bits.

Here is the internal state sequence:

1. Wait for the A input to go high (idle state).
2. Wait for the A input to go low (START bit edge).
3. Delay for half a bit period.
4. If the A input is no longer low, loop to (2).
5. Delay for one bit period.
6. Right-shift the A input into the shifter and delay for one bit period, repeat until all data bits are received.
7. Capture the shifter into the Z register and raise IN.
8. Loop to (1).

RDPIN/RQPIN is used to read the received word. The word must be shifted right by 32 minus the word size. For example, to LSB-justify an 8-bit word received, you would do a ‘SHR D,#32-8’.