HIGH / LOW

In PBASIC the **HIGH** and **LOW** commands set a pin's state (1 for **HIGH**, 0 for **LOW**) and makes it an output.

PBASIC: **HIGH** *pin*
LOW *pin*

As Spin does not include methods comparable to **HIGH** and **LOW** you could code them.

```
pub high(pin)

  if (pin => 0) and (pin =< 31)                         ' if valid
    outa[pin] := 1                                      '   write 1 to pin
    dira[pin] := 1                                      '   make pin an output

pub low(pin)

  if (pin => 0) and (pin =< 31)                         ' if valid
    outa[pin] := 0                                      '   write 0 to pin
    dira[pin] := 1                                      '   make pin an output
```

In PBASIC:

```
  HIGH 0
  LOW 1
```

In Spin with the custom methods:

```
  high(0)
  low(1)
```

Where you want the most efficiency in your Spin program it is best to initialize a pin as an output with:

```
  dira[pin] := 1
```

…and then update the pin state (0 or 1) as required by writing to the **outa[]** register – there is no need to make a pin an output when it's already set to output mode.

One of the nice things about Spin is that we can isolate bits in the **dira[]** register. Let's say, for example that we wanted to define P3, P4, and P5 as outputs. As these pins are a contiguous group we can do this in one easy command and without affecting any of the other pins in **dira[]**.

```
  dira[5..3] := %111
```

Advanced programmers will tend to use the post-fix operators like this to make a group of pins outputs:

```
  dira[5..3]~~
```

The post-fix **~~** operator sets all pins in the group to "1."

The benefit of using the post-fix operators is that we can change the number of elements in a contiguous pin group without having to edit the value side of the assignment (%111 above).

As your programming skills improve you'll find it beneficial to move away from "magic numbers" (embedded constant numbers) in favor of using named constants. If these pins corresponded to green (P3), yellow (P4), and red (P5) LEDs a professional would start with enumerated constants.

```
con

  #3, LED_GRN, LED_YEL, LED_RED
```

With these constants we can set the LED pin group to outputs like this:

```
  dira[LED_RED..LED_GRN]~~
```

The advantage is two-fold: 1) this style spells out, functionally, which pins are being affected, and 2) the use of named constants and the post-fix **~~** operator means that if the location or size of the pin group changes the only area of the program that needs editing is in the **con** section at the top. The lesson here is to avoid embedded numeric constants for IO pins where possible; using named constants will make your program easier to read and maintain.

Note the order within the brackets: left-to-right, MSB to LSB. When using pin groups with **dira[]**, **outa[]**, and **ina[]** the order matters. For example:

```
  outa[LED_RED..LED_GRN] := %001
```

…will turn on the green LED while:

```
  outa[LED_GRN..LED_RED] := %001
```

…will turn on the red LED. Unless I have a very specific reason not to my programs will specify pin groups MSB to LSB; this makes sense to me as that would be the order of the bits if written as a binary number.