Ports

Sets of pins can be grouped in groups called ports – they can all be read from or written to simultaneously. This is done by reading into a register (each pin gets its own bit), or writing from it.

Why?

Mainly for speed. We'd have to use a separate read for each pin otherwise, which is very slow and expensive (using the arduino `digitalRead()`).

Also useful for the simultaneity – if trying to test people's reaction speeds against one another, you'll want to read in the values simultaneously so that you don't give an advantage to either person.

It also reduces the size of the program, which means we can do more within the memory restrictions.

Arduino Ports

Our device has 3 ports, two of which have 6 pins and one of which has 8 pins.

A special register is used to hold the pin direction for each pin in a port, you can use this register to control the pins. These are called DDRx (as in DDRD, DDRB, DDRC).

You can use a single command to control this:

```
DDRD = B10110001;
```

This will set the pins to output (1) or input (0), starting at the most significant bit.

You can also use hex values:

```
DDRD = 0xB1;
```

Port numbers are different on the arduino board (where port D goes from 0 to 7) and on the chip itself (where port D is 2 to 9).

Knight Rider Code

When written out in straight code using `digitalWrite()` and `pinMode()`, the compiled code is about 1300 bytes.

If you use ports instead (using for example the `PORTD` instruction), the size is cut down to about 700 bytes.

Using bit-shifting, you can get down to about 600 bytes.

- The << operator shifts all the bits one place to the left (multiplying the integer value by 2)

Port Output

Write out by assigning to `PORTD`, `PORTC` and `PORTB`.

If you write a 1 to a pin defined as an input, you get a pull-up resistor attached internally (so the pin is not floating).

Port Input

Uses the `PIND`, `PINC`, `PINB` commands (standing for "port input").

These and the `PORTD` etc. commands are macros.

Port Speed

Writing to ports is much faster than writing to individual pins using `digitalWrite()`.

Bitwise Operators

We'll use these for masks (accessing individual bits) and for reducing the amount of memory needed to store data.