spark - Cheat Sheet

docs.spark.io

Structure

`void setup(){ ... }`
Initialization function.

`void loop(){ ... }`
Main program loop.

Control Structures

`if(x<5){ ... } [else { ... }]`
Run a block of code only if a condition is true. If an else statement is present, run this block if condition is false.

`for(int i=0; i<255; i++){ ... }`
Loop for a set count.

`while(x<5){ ... }`
Loop while a condition is true.

`break`
Escape from a loop control structure,

`continue`
Escape from the current iteration of a control structure

General Operators

`=`
Assignment.

`+, -, *, /, %`
Plus, minus, multiply, divide, modulo.

`==, !=`
Equal to, not equal to.

`<, <=, >, >=`
Less than, less than or equal to, greater than, greater than or equal to.

Bitwise Operators

`&, |, ^, ~`
Bitwise AND, OR, XOR, NOT.

`<<, >>`
Bitwise left shift, bitwise right shift.

Compound Operators

`++, --, +=, -=, *=, /=, &=, !=`
Increment, decrement, compound addition, compound subtraction, compound multiplication, compound division, compound bitwise AND, compound bitwise OR.

Pointer Access

`&, *`
Reference operator, dereference operator.

Constants

`HIGH, LOW`
Pin value constants.

`OUTPUT, INPUT, INPUT_PULLUP, INPUT_PULLDOWN`
Pin mode constants.

`true, false`
Boolean constants.

Data Types

`void`
Function type declaration for functions that return no information.

`boolean`
eg, true or false

`char`
8-bit (1-byte) number from -128 to 127.

`byte`
8-bit (1-byte) unsigned number from 0 to 255.

`int`
32-bit (4-byte) value from -2,147,483,648 to 2,147,483,647.

`unsigned int`
32-bit (4-byte) value from 0 to 4,294,967,295.

`long`
32-bit (4-byte) value from -2,147,483,648 to 2,147,483,647.

unsigned long
32-bit (4-byte) value from 0 to 4,294,967,295.

short
16-bit (2-byte) value from 32,768 to 32,767.

float
32-bit (4-byte) floating point number.

double
64-bit (8-byte) floating point number.

Arrays

`int myArray[6];`
An unpopulated integer array with 6 slots.

`int myArray[] = {1,2,3,4,5,6};`
A populated integer array with 6 slots.

`int myArray[6] = {1,2,3,4};`
A partially populated integer array with explicitly 6 slots.

Strings

Basic strings are represented as char arrays, however the spark core also has a String class with many more helper methods.

```
char s1[15];
char s2[6] = {‘s’,’p’,’a’,’r’,’k’};
char s3[6] = {‘s’,’p’,’a’,’r’,’k’,’\0’};
char s4[] = “spark”;
char s5[6] = “spark”;
char s6[15] = “spark”;
```

Math

`min(x, y);`
Calculates the minimum of two numbers.

`max(x, y);`
Calculates the maximum of two numbers.

`abs(x);`
Computes the absolute value of a number.

`constrain(x, min, max);`
Constrains a number to be within a range.

`map(x, fromMin, fromMmax, toMin, toMax);`
Re-maps a number from one range to another.

`pow(base, exponent);`
Calculates the value of a number raised to a power.

`sqrt(x);`
Calculates the square root of a number.

Time

The spark core comes with basic timing methods, but also has a Time class with many more helper methods.

`millis();`
Returns the number of milliseconds since the Spark Core began running the current program.

`micros();`
Returns the number of microseconds since the Spark Core began running the current program.

`delay(ms);`
Pauses the program for the amount of time (in miliseconds) specified.

`delayMicroseconds(ms);`
Pauses the program for the amount of time (in microseconds) specified.

I/O

`pinMode(pin, mode);`
Configures the specified pin to behave either as an input or output.

`digitalWrite(pin, value);`
Write a HIGH or a LOW value to a digital pin.

`digitalRead(pin);`
Reads the value from a specified digital pin, either HIGH or LOW.

`analogWrite(pin, value);`
Writes an analog value (PWM wave) to a pin.

`analogRead(pin);`
Reads the value from the specified analog pin. Values range between 0 and 4095.

spark - Cheat Sheet

Interrupts

`attachInterrupt(pin, function, mode);`
Specifies a function to call when an external interrupt occur

`detachInterrupt(pin);`
Turns off the given interrupt.

`noInterrupts();`
Disabled interrupts.

`interrupts();`
Re-enabled interrupts.

Tone

`tone(pin, frequency, duration);`
Generates a square tone on the given pin.

`noTone(pin);`
Stops the current tone playing on the given pin

RGB

`RGB.control(bool);`
Takes and gives back user control of the built in RGB LED.

`RGB.color(red, green, blue);`
Set the color of the RGB with three values, 0 to 255.

`RGB.brightness(val);`
Scale the brightness value of all three RGB colors with one value, 0 to 255.

Servo

`servo.attach(pin);`
Setup a servo on a particular pin.

`servo.detach();`
Detach the servo variable from its pin.

`servo.write(angle);`
Set the angle of the servo.

`servo.read();`
Reads the current angle of the servo.

TCPClient

```
client.connect(ip, port);
client.connect(url, port);
```
Connects to a specified IP address/URL and port. Returns true if connection succeeds, false if not.

`client.connected();`
Whether or not the client is connected. Returns true if the client is connected, false if not.

```
client.write(val);
client.write(buf, len);
```
Write data to the server the client is connected to.

```
client.print(data);
client.print(data, BASE);
```
Print data to the server that a client is connected to.

```
client.println();
client.println(data);
client.println(data, BASE);
```
Print data, followed by a carriage return and newline, to the server a client is connected to.

`client.available();`
Returns the number of bytes available for reading.

`client.read();`
Read the next byte received from the server the client is connected to

`client.flush();`
Discard any bytes that have been written to the client but not yet read

`client.stop();`
Disconnect from the server.

TCPServer

`TCPServer server = TCPServer(port);`
Create a server that listens for incoming connections on the specified port.

`server.begin();`
Tells the server to begin listening for incoming connections.

`server.available();`
Gets a client that is connected to the server and has data available for reading.

```
server.write(val);
server.write(buf, len);
```
Write data to all the clients connected to a server.

```
server.print(data);
server.print(data, BASE);
```
Print data to all the clients connected to a server.

```
server.println();
server.println(data);
server.println(data, BASE);
```
Print data, followed by a newline, to all the clients connected to a server.

Cloud Functions

`Spark.variable(name, var, type);`
Expose a variable through the Spark Cloud.

`Spark.function(name, function);`
Expose a function through the Spark Cloud.

`Spark.publish(name, data);`
Publish an event through the Spark Cloud.

`Spark.subscribe(name, function);`
Subscribe to events published by Cores.

Cloud API

```
HEADER Bearer {ACCESS_TOKEN},
?access_token={ACCESS_TOKEN}
```
Authenticates the request with the given access token.

`GET /v1/devices/{DEVICE_ID}/{VARIABLE}`
Read a variables exposed through the Spark Cloud from the given device.

`POST /v1/devices/{DEVICE_ID}/{FUNCTION}`
Call a method exposed through the Spark Cloud on the given device.

```
GET /v1/devices/{DEVICE_ID}/events/
[/{EVENT}]
```
Open an SSE connection to the given devices event stream.

Misc

`// ...`
Single line comment.

`/* ... */`
Multi-line comment.

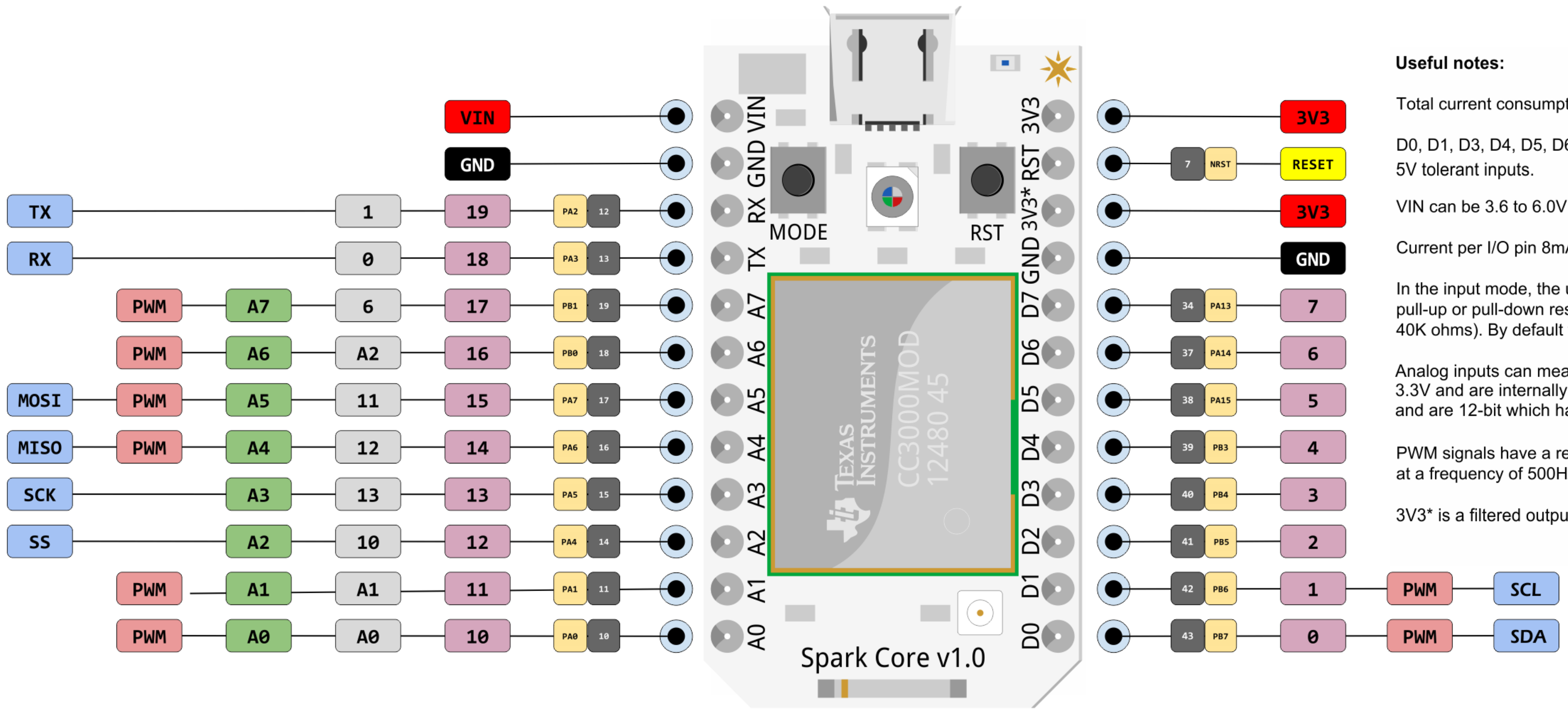
`#define ANSWER 42`
Constant variable declaration.

`#include <myLib.h>`
Includes a third party library. .

Links

https://www.spark.io/
http://docs.spark.io/
http://community.spark.io/
https://www.spark.io/build

Pinout Diagram



Useful notes:

Total current consumption is 300mA

D0, D1, D3, D4, D5, D6 and D7 are 5V tolerant inputs.

VIN can be 3.6 to 6.0V

Current per I/O pin 8mA min, 20mA max

In the input mode, the user can activate internal pull-up or pull-down resistors (typically equal to 40K ohms). By default these are deactivated.

Analog inputs can measure voltages of up to 3.3V and are internally referenced to 3V3*, and are 12-bit which have a max value of 4095.

PWM signals have a resolution of 8 bits and run at a frequency of 500Hz.

3V3* is a filtered output used for analog sensors.

■	GND
🟥	Power
🟨	Control
■	Physical Pin
🟨	Port Pin
□	Pin Function
□	Arduino Equiv.
🟩	Analog Pin
🟥	PWN Pin
🟦	Serial Pin
🟪	IDE
● ●	Source Total 150ma

Ref
https://github.com/spark/core/blob/master/Pin%20mapping/core-pin-mapping-v1.xlsx
http://docs.spark.io/#/shields

Inspired by
http://goo.gl/gvIUsz

Diagram Created by Jonathan Beri / BDub
http://google.com/+JonathanBeri