G Programming Language

Copyright 2009, Geordie and Chris Tilt

G to C compiler Usage, version 0.1

## G Language Examples

**Example 1.** This G program will execute the actions inside the **do** loop in parallel for 20 times. “Parallel” means that it will execute each action at the same time! An LED connected to pin3 will be “on” all the time. An LED connected to pin4 will blink fast for 15 seconds, but at the same time pin5’s LED will blink slowly. Pin2’s LED will blink at the default rate. It should look like a Christmas tree – all blinking at the same time. The do loop always means “run everything inside the loop at the same time”.

**do**

**turn on pin3**

**fast blink pin4 for 15 secs**

**slow blink pin5 for 15 secs**

**blink pin2**

**until 20 times**

**Example 2.** This G program shows a lot of new things. The **repeat** loop is used to execute things in a sequence – that means the things inside the loop happen one after the other, not all at the same time. The do loop that is contained inside the repeat loop happens after the slow blink of pin2 and everything inside it happens at the same time. The loop runs long enough so that the things inside it can run twice. After the do loop, there is another repeat loop and everything in side that loop happens in a sequence, one after the other, and the whole loop repeats 2 times.

**repeat**

**blink pin1 fast 50 times**

**slow blink pin2 1 times**

**do**

**fast blink pin1 50 times**

**blink fast pin1 50 times**

**until 3 times**

**repeat**

**medium blink pin3 2 sec**

**blink pin4**

**blink pin4 slow**

**until 2 times**

**if detect pin5 blink pin3 5 times**

**if detect pin5 fast blink pin3 5 times and turn off pin7**

**repeat**

**if detect pin5 fast blink pin3 3 seconds**

**turn on pin7**

**wait 2 sec**

**turn off pin7**

**wait 25 msec**

**until 3 secs**

**until 3 times**

Also, note that you can **detect** input on a pin, like the **detect** on pin5, and you can take action when input on the pin goes high. In our example we blink pin3 when pin5’s input changes.

You can also **wait** for some time to pass, which is really only useful inside a repeat loop when things are happening in sequence. It doesn’t make sense to wait in parallel with other actions inside a do loop.

## G Language Reference

In the following, <number> means any valid number. So, “pin<number>” could be *pin3* or *pin9*. And “until <number> msecs” could be “*until 50 msecs”*. And **[is]** means that the keyword **is** is optional. *On* and *is on* mean the same thing, so you can put the optional word in your statement if you want to. Any keyword that appears inside brackets, like this, **[** keyword **]**, is optional.

## A G Program

A G program is just a list of G statements. They will all get executed one after the other in a sequence, as if they had been surrounded with a do loop. But it’s good programming practice to write your top level program with a loop that specifies when it should stop. It can even be empty, but that’s not very interesting.

**do**

**blink pin3 fast**

**forever**

### Do loops:

A **do** loop runs all of it’s statements concurrently, which means “at the same time”. It’s like a sand box of kids all playing together; you give them all instructions and say, “go”. They all play at the same time. You have to tell the loop how long to run by adding a guard. The “...” is where you put your other G program statements. The world runs concurrently, so we like to be able to program that way ☺

**do** … **forever**

**do** … **until** <number> **times**

**do** ... **until detect pin**<number> **[is] on**

**do** … **until detect pin**<number> **[is] off**

**do** … **until detect pin**<number> **[is] high**

**do** … **until detect pin**<number> **[is]** **low**

**do** … **until** <number> **secs**

**do** … **until** <number> **msecs**

**do** … **while detect pin**<number> **off**

## Repeat loops:

A **repeat** loop is exactly like a **do** loop, except that the statements inside the loop are executed one after the other, in the order they appear in the loop. This is called “sequential programming” and it’s what most old programmers are used to. But, sometimes you really need stuff to happen in a sequence one after the other. The “...” is where your statements go and they can of course contain more loops inside your top loop as shown in Example 2.

**repeat** … **forever**

**repeat** … **until** <number> **times**

**repeat** ... **until detect pin**<number> **[is] on**

**repeat** … **until detect pin**<number> **[is] off**

**repeat** … **until detect pin**<number> **[is] high**

**repeat** … **until detect pin**<number> **[is]** **low**

**repeat** … **until** <number> **secs**

**repeat** … **until** <number> **msecs**

**repeat** … **while detect pin**<number> **off**

## Simple actions

For example, the statement “*turn on pin8*” sets the output of pin 8 to the high voltage. And “*wait 2 secs*” delays program execution for 2 seconds.

**turn on pin**<number> turn on an output pin

**turn off pin**<number> turn off an output pin

**turn pin**<number> **on** turn on an output pin

**turn pin**<number> **off** turn off an output pin

**wait** <number> **msecs** delay the program

**wait** <number> **secs** delay the program

**blink pin**<number>blink the pin at medium rate

**blink pin**<number> **fast** blink the pin twice per second

**blink pin**<number> **slow** blink the pin twice every tree seconds

**blink pin**<number> **medium** blink the pin once per second

**blink pin**<number> **every** <number> **msec**

blink the pin every <number> msecs

**blink pin**<number> **[for]** <number> **secs** blink pin for <number> seconds

**blink pin**<number> [**for**] <number> **times** blink pin for <number> times

The blink rate can appear before or after the pin:

**blink pin**<number> **slow every** <number> msecs

**blink pin**<number> **fast for** <number> **secs**

**blink medium pin**<number>

Also, you can control both the blink rate and how long it does it.

**blink pin**<number> **every** <number> **msec for** <number> **sec**

**blink pin**<number> **fast** **for** <number> **sec**

And finally, you can put the blinking rate at the beginning of the statement.

**Fast blink pin**<number> **for** <number> **sec**

Examples**:**

**fast blink pin4 for 15 secs**

**blink slow pin5 for 15 secs**

**blink pin2**

**blink pin9 every 300 msecs for 42 secs**

**medium blink pin8**

**turn on pin7**

**turn pin7 off**

## If statements

Sometimes you only do something **if** something else happens. For that, we use a conditional if statement.

**if detect pin**<number> **turn on pin**<number>

You can detect both on and off as well as high and low (on is the same as high, off is the same as low). And the word “is” is optional in there too. You can execute multiple actions when your **if** condition is detected. So it looks like this:

**If detect pin**<number> [**is**] **on** <simple action list>

**If detect pin**<number> [**is**] **off** <simple action list>

A <simple action list> looks like several actions connected by the word “**and**”.

<action> **and** <simple action> **and** <action> ....

For example, you could write “*if detect pin3 turn on pin9 and turn off pin8 and wait 2 secs”* which will set the output voltage on pin9 to high and pin8’s voltage to low and then wait for 2 seconds only if the input voltage on pin3 is high.