C program which uses floats and which replaces Cosmic `putchar()` function

```
#include "hc12b32.h"
#include <stdio.h>
#include <float.h>
#include <math.h>

void main(void)
{
    float a;

    printf("hello, world!\n");
    a = 4.0;
    printf("a = %f\n", a);
}

int putchar(char c)
{
    while ((SC0SR1 & 0x80) == 0) ;
    SC0DRL = c;
    if (c=='\n')
    {
        while ((SC0SR1 & 0x80) == 0) ;
        SC0DRL = '\r';
    }
}
```

Cosmic .lkf file to linking libraries for floating point and printf()

- When linking the libraries for floating point the Cosmic manual says to link the floating point library *before* linking the integer library

```
#    link command file for test program
#
+seg .text -b 0x1000 -n .text    # program start address
+seg .const -a .text             # constants follow code
+seg .data -b 0x7000             # data start address
crts.o                           # startup routine
test.o                           # application program
c:/cx32/lib/libf.h12             # C floating point library
c:/cx32/lib/libi.h12             # C integer library
c:/cx32/lib/libm.h12             # machine library
+def __stack=0x0A00              # stack pointer initial value
```

Simple Algorithm for Motor Control

- A simple control algorithm (called proportional control) is

$$DC = \frac{1}{\alpha} S_d + k(S_d - S_m)$$

  where DC is the duty cycle, S_d is the desired speed, and S_m is the measured speed. α is the slope of the measured motor response (speed vs. duty cycle). k is a constant which can be found by trial and error. This algorithm will result in good behavior, but the speed of the motor will differ slightly from the desired speed.

- Another simple algorithm is

$$DC_{new} = DC_{old} + k(S_d - S_m)$$

  where DC is the duty cycle, S_d is the desired speed, and S_m is the measured speed. k is a constant which can be found by trial and error. This will not settle to the final value as quickly as the first algorithm but the actual speed of the motor will match the desired speed.

- The value of k determines the response of the system. If k is too large, the speed will have a great deal of overshoot, and may oscillate around the desired value. If k is too small, the motor will take a long time to reach the desired speed.

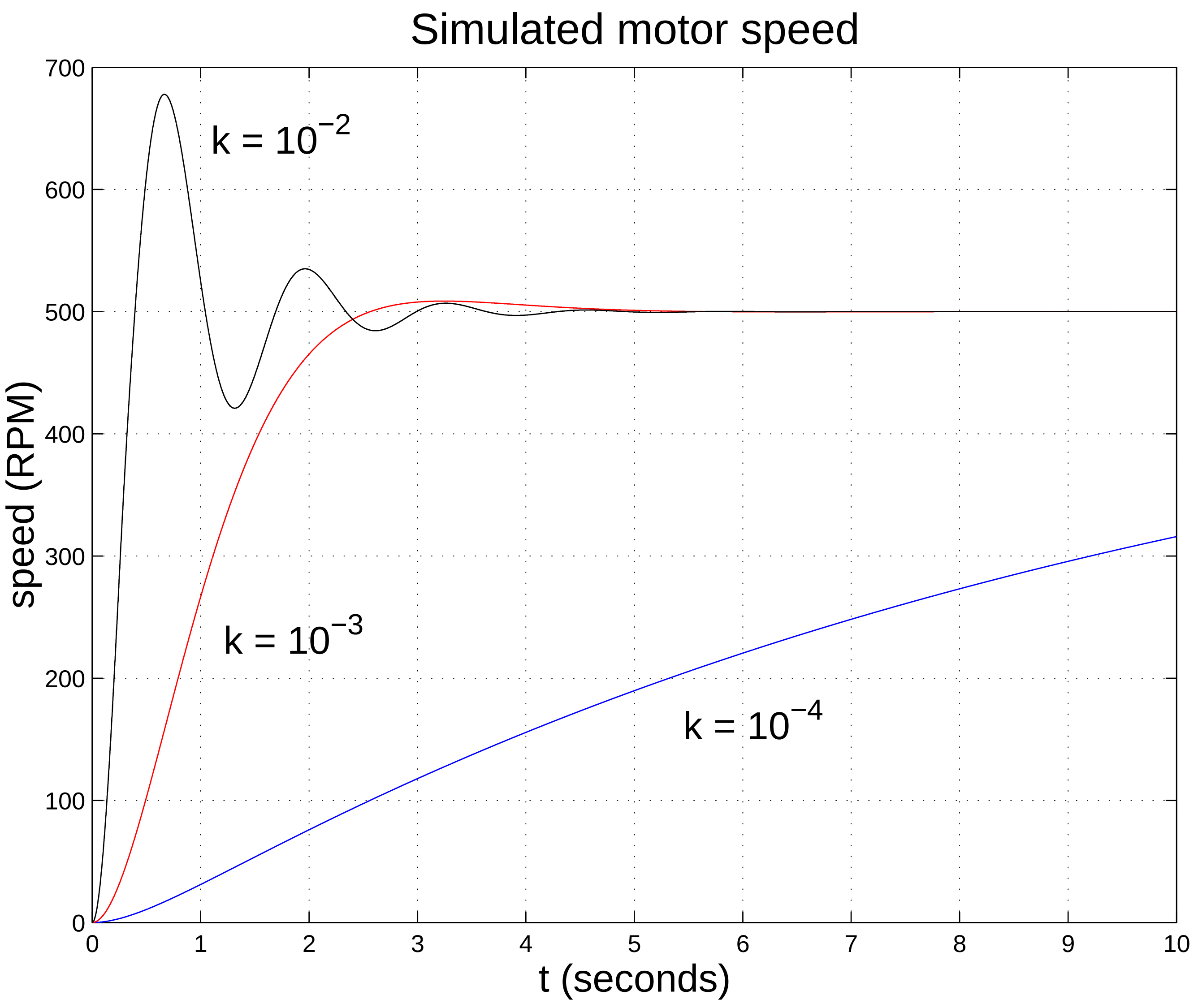
- The next page shows the theoretical step response of a motor for several values of k.

- In order for these to work effectively it is necessary to update the duty cycle at a regular rate.

- The rate should be fast compared to the time it takes the motor to respond to changes in duty cycle, but slow enough so that the HC12 has time to do the calculations

- A good rate for small motors is to do the calculation every few milliseconds

- You can use an real time interrupt or a timer overflow interrupt to perform the updates at a regular rate


Simulated motor speed
speed (RPM)
t (seconds)
$k = 10^{-2}$
$k = 10^{-3}$
$k = 10^{-4}$
0
100
200
300
400
500
600
700
0
1
2
3
4
5
6
7
8
9
10