Stepper drive with cogging compensation

1 Introduction

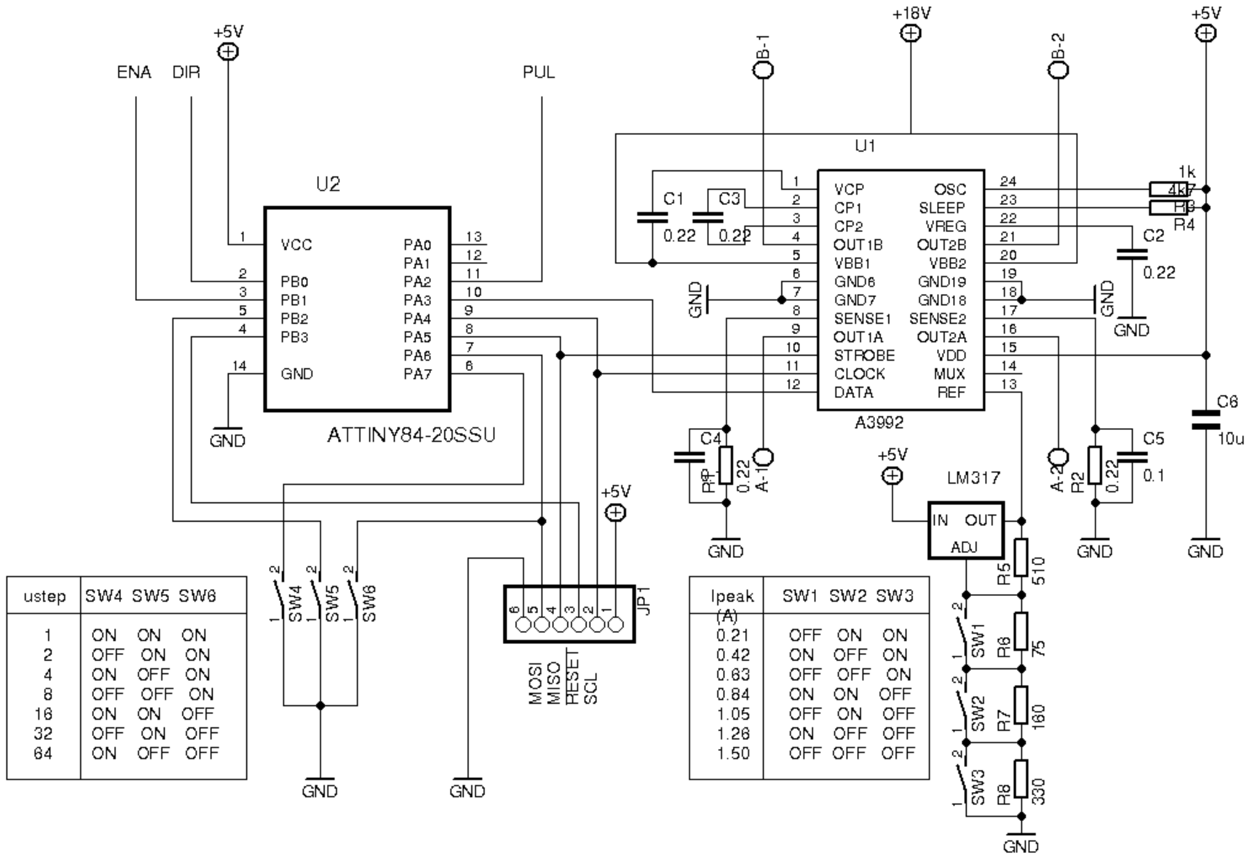
This project contains software for an Atmel ATTiny84 to control an Allegro A3992 DMOS Dual Full-Bridge Microstepping PWM Motor Driver chip. The ATtiny84 is programmed to produce a non-sinusoidal drive in order to compensate for stepper motor cogging. For my application it was necessary to produce very precise microstepping of the motor and a simple sinusoidal drive was not adequate with the motor I was using (a Wantai 1.7 amp 200-step NEMA-17 bipolar stepper motor). The software is written in C and is compiled and downloaded to the ATTiny84 by running the Makefile in this folder (avrdude is required).

2 Cannibalizing a 2M415

The stepper driver can be built from scratch, but a fairly convenient alternative for me was to cannibalize a 2M415 stepper driver (of Chinese manufacture and easy to find on ebay) and replace the original PIC16F630 with an ATTiny84, which has a similar pinout. A few modifications to the printed circuit were also required including disconnecting one track and adding a few extra bridging wires. I had to remove the PCB to do this. The existing 6 x 1 pin header on the board was wired to function as an ICSP port. The piano key switches for microstep and peak current selection were retained.

3 Schematic

A schematic diagram of the modified drive is shown below.



ustep	SW4	SW5	SW6
1	ON	ON	ON
2	OFF	ON	ON
4	ON	OFF	ON
8	OFF	OFF	ON
16	ON	ON	OFF
32	OFF	ON	OFF
64	ON	OFF	OFF

Ipeak (A)	SW1	SW2	SW3
0.21	OFF	ON	ON
0.42	ON	OFF	ON
0.63	OFF	OFF	ON
0.84	ON	ON	OFF
1.05	OFF	ON	OFF
1.26	ON	OFF	OFF
1.50	OFF	OFF	OFF

In the 2M415 driver, the +5V supply is derived from the 18V input by a switch-mode power supply and the PUL, DIR and ENA inputs are protected by opto-couplers. These are not shown on the schematic (the circuit is complicated enough!) Schottky protection diodes placed across the outputs and between the outputs and ground have also been omitted. The voltage applied to the REF pin of the A3992 determines the maximum current delivered to the motor. This is set by means of switches SW1,SW2 and SW3. Microstepping is selected by SW4, SW5 and SW6. The motor windings are connected between terminals A-1 and A-2, and B-1 and B-2.

4 Programming

For in-circuit programming, the pin header JP1 is connected to a parallel port programming dongle. The compiling and programming functions are contained in the Makefile. Open a terminal and change to the directory containing the files, then enter the terminal commands:

```
make clean
make all
make program
```

5 Software description

A3992_driver.c

`int main (void)`

The main program sets the ATTiny84 to run at 8MHz, sets the TIMER1 clock frequency to 30.5Hz, sets the I/O pins, pull-ups and pin-change interrupts. It then writes the waveform look-up table and initiates an infinite `for` loop.

`ISR(TIM1_COMPA_vect)`

TIMER1 compare interrupt routine. The microstep select switches are read 30.5 times per second, see `read_uSTEP_switches` below.

`ISR(PCINT0_vect)`

Interrupt service routine for the PUL input. This checks that PUL is high 10 times (for better reliability), and sends WORD0 to the A3992.

`void read_uSTEP_switches(void)`

Reads the microstep select switches at every TIMER1 interrupt and sends WORD1 to the A3992. Strictly, this only needs to be done at boot up, but repeatedly sending WORD1 reduces the chance of data corruption by high current spikes.

`void setup_table(float a, float b, uint_t mode)`

Routine to set up the look-up table for the output waveform. Four different waveforms can be selected by `mode`. The A3992 contains two 6-bit linear DACs, so the waveform is normalised to a maximum of 63 with bit 7 functioning as the sign bit. There are 256 entries in the look-up table, representing 256 microsteps over a full step.

serial_write.S

The file `serial_write.S` contains assembler routines for writing the 19-bit serial words WORD0 and WORD1 to the A3992 (see the Allegro A3992 datasheet).

6 Waveforms

Several different waveforms were tried during development. The one which best compensates for the cogging of my stepper motor was arrived at by trial-and-error and a little guesswork. This gives the following drive level as a function of phase angle, ϕ.

$s = sin\phi + bsin4\phi \times exp(-3|sin\phi|) \times sgn(cos\phi)$,

with $b = 0.05$.

This function is plotted in the Figure 1 below, along with the original sinusoid. The second Figure 2 shows the correction (dashed line) added to $sin\phi$ to produce this final waveform (solid line).

R I Henderson 2016-5-30

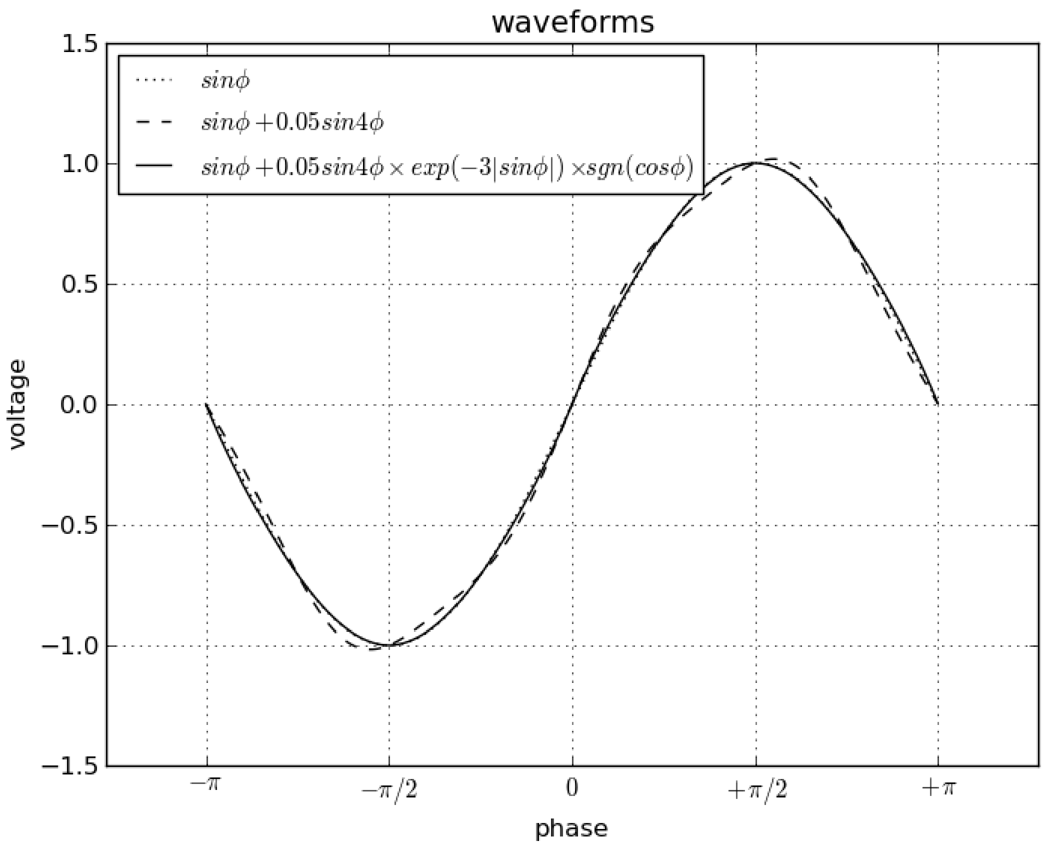


Figure 1: Driving waveforms

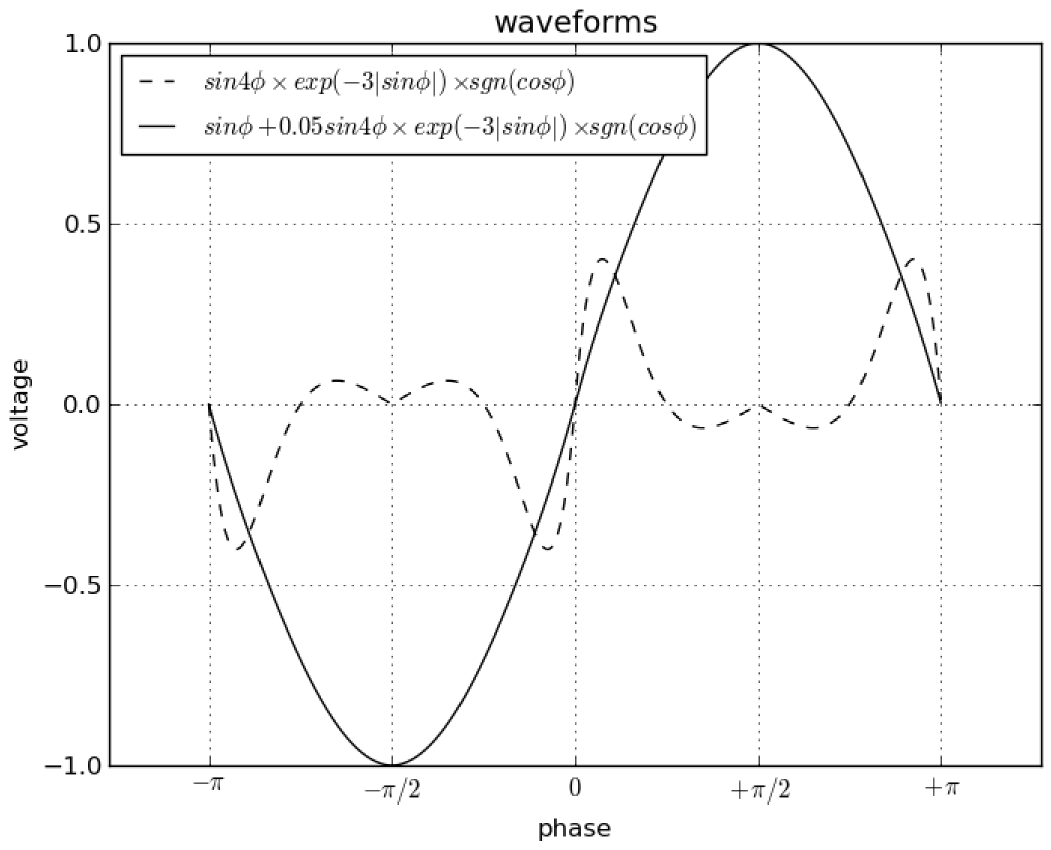


Figure 2: Difference and total waveforms