Microstepper Motor Controller

Circuit Cellar AVR Contest-2004. Entry no: unassigned

June 30, 2004

1 Abstract

Stepper motors are asynchronous motors wheerin the magnetic field is switched electronically to rotate the armature magnet. A stepper motor can be operated and controlelr very simply with the help of a suitable step sequencer and an appropriate driver. Typically, the stepper motor achieves 200 steps per rotation which is quite sufficient for many applications. More resolution can be attained by operating the stepper motor in the half-step mode to get 400 steps per rotation. For applications that require accurate positioning as well as a finer resolution over a wide range of speeds, a micro-stepping mode of operating the stepper motor can be utilized.

Micro-stepping mode of operation allows stepper motor to achieve finer resolutions upto 256 microsteps in a step, which is equivalent to about 50,000 steps per rotation.

The micro-stepping mode of operation requires that the current through the stepper motor windings be as a sinusiodal quadrature phase. If this sinusoidal current is generated digitally, then the step resolution of the sine wave would determine the number of microsteps per step.

We used an AVR Microcontroller AT90S8515, an 8-bit DAC (MAX521) to generate the required quadrature phase sine wave signal. Further, an ordinary audio amplifier TDA2030 was used to drive the stepper motor.

Step resolutions upto 64 microsteps per step were achieved. The system was tested and positional error of less than one microstep was observed.

The system can handle 4-wire, bi-polar stepper motors of upto 1A phase currents and upto 30V supply voltage. It can be modified to handle larger stepper motors.

Keywords: AT90S8515, MAX521, Stepper Motor, Micro-stepping operation, TDA2030 audio amplifier.

2 Introduction

Stepper motors are asynchronous motors wheerin the magnetic field is switched electronically to rotate the armature magnet. A stepper motor can be operated and controlelr very simply with the help of a suitable step sequencer and an appropriate driver. Typically, the stepper motor achieves 200 steps per rotation which is quite sufficient for many applications. More resolution can be attained by operating the stepper motor in the half-step mode to get 400 steps per rotation. For applications that require accurate positioning as well as a finer resolution over a wide range of speeds, a micro-stepping mode of operating the stepper motor can be utilized.

Micro-stepping mode of operation allows stepper motor to achieve finer resolutions upto 256 microsteps in a step, which is equivalent to about 50,000 steps per rotation.

The micro-stepping mode of operation requires that the current through the stepper motor windings be as a sinusiodal quadrature phase. If this sinusoidal current is generated digitally, then the step resolution of the sine wave would determine the number of microsteps per step.

We used an AVR Microcontroller AT90S8515 and an 8-bit DAC (MAX521) to generate the required quadrature phase sine and cosine wave signals. The microcontroller stores the internal flash memory with a table with 256 entries depicting the values of a sine wave. Similarly, another table stores the cosine wave values.

The MAX521 is an octal 8-bit DAC, with an I2C communicatin link. The microcontroller communicates with the DAC using bit-banging method, generating the I2C signals under software control.

The output of two of the DACs from MAX521 are connected to a pair audio amplifiers TDA2030. TDA2030 is a 30W class AB audio amplifier. The amplifier is operated with a gain of 4. However, care is taken to ensure that there is no gain offset between the two amplifiers.

3 Design Description

Figures 3 and 4 illustrate the schematic diagram of the motor controller. The output of the audio amplifiers feed the two phases of the bi-polar stepepr motor. The audio amplifiers operate with +12 and -12V power supply and the maximum swing at the output of the amplifier is 20V max.

The motor controller is operated in a network of many such controllers together. The motor controller receives motion commands from a master or a PC. The vasious motor controllers have their RxD pins of the UARTs tied together, while the TxD pins of the controllers are connected together using an open collector TTL inverter as shown in figure 2. All the motor controllers and the host occupy a common enclosure and communicate with each other on a back plane bus.

Each motor controller is given a unique identification address with the help of external DIP switches. Currently, upto 15 such motor controllers can be connected in a single system, allowing control of 15 motors simultaneously.

The host sends a data packet containing the address of the slave and the desired command to the designated controller. This command is received by all the slaves together and the designated slave decodes the address in teh command with the address set on its card. It then acknowledges the command back to the host. Other slaves simply ignore the command till a command with an address matching their physical address is received.

The number of step size is also set on the PCB of the motor controller with the help of a 4-bit DIP switch. the user can select, 4, 8, 16 or 64 microsteps per step using this arrangement.

Step resolutions upto 64 microsteps per step were achieved. The system was tested and positional error of less than one microstep was observed.

The system can handle 4-wire, bi-polar stepper motors of upto 1A phase currents and upto 30V supply voltage.

In this particular implementation, the motor controller is used in a spectroscope meant for astronomy instrumentation. The system has four home positions, designated as H1, H2, H3 and H4. The motor controller can be given a command to seek any one of the four home positions. The home position is detected with the help of an opto-isolater (as illustrated in figure 4) which gets interrupted when the motor reaches that position.

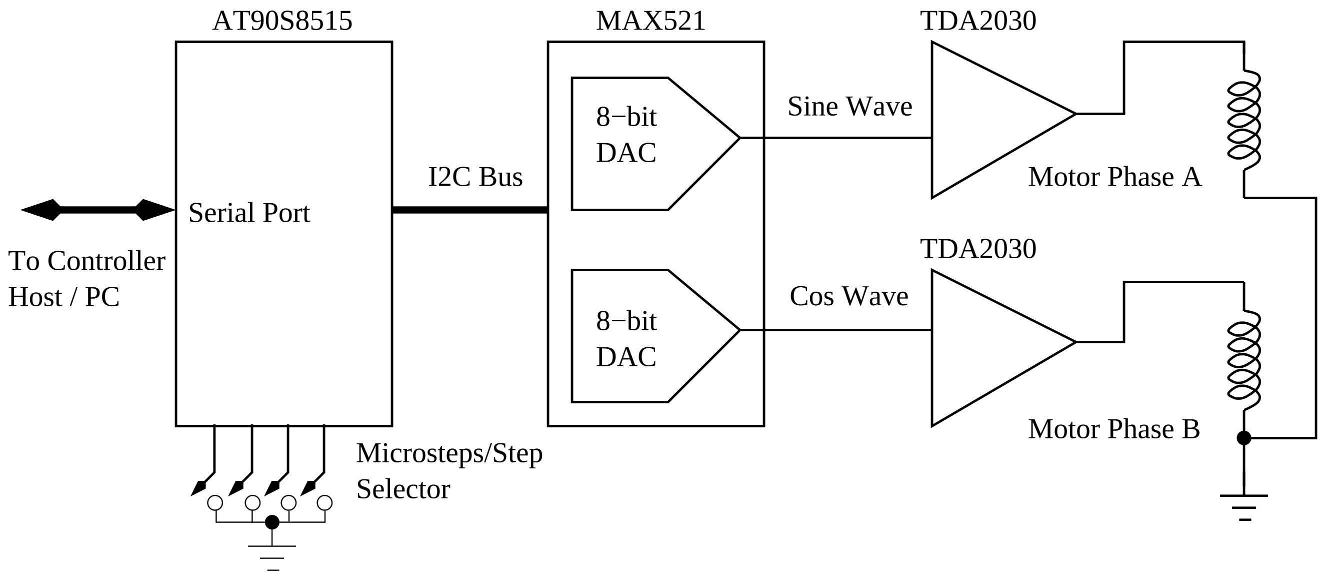

Figure 1: Microstepper Controller Block Diagram

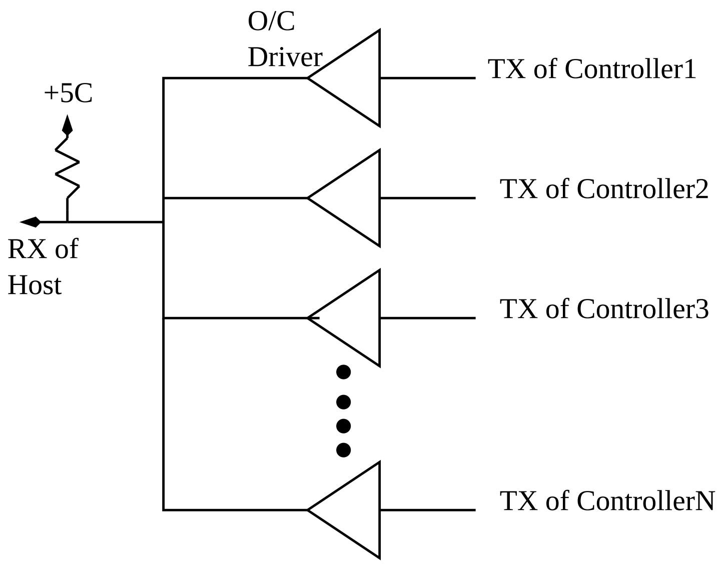

Figure 2: Connecting the Host and the Motor Controllers

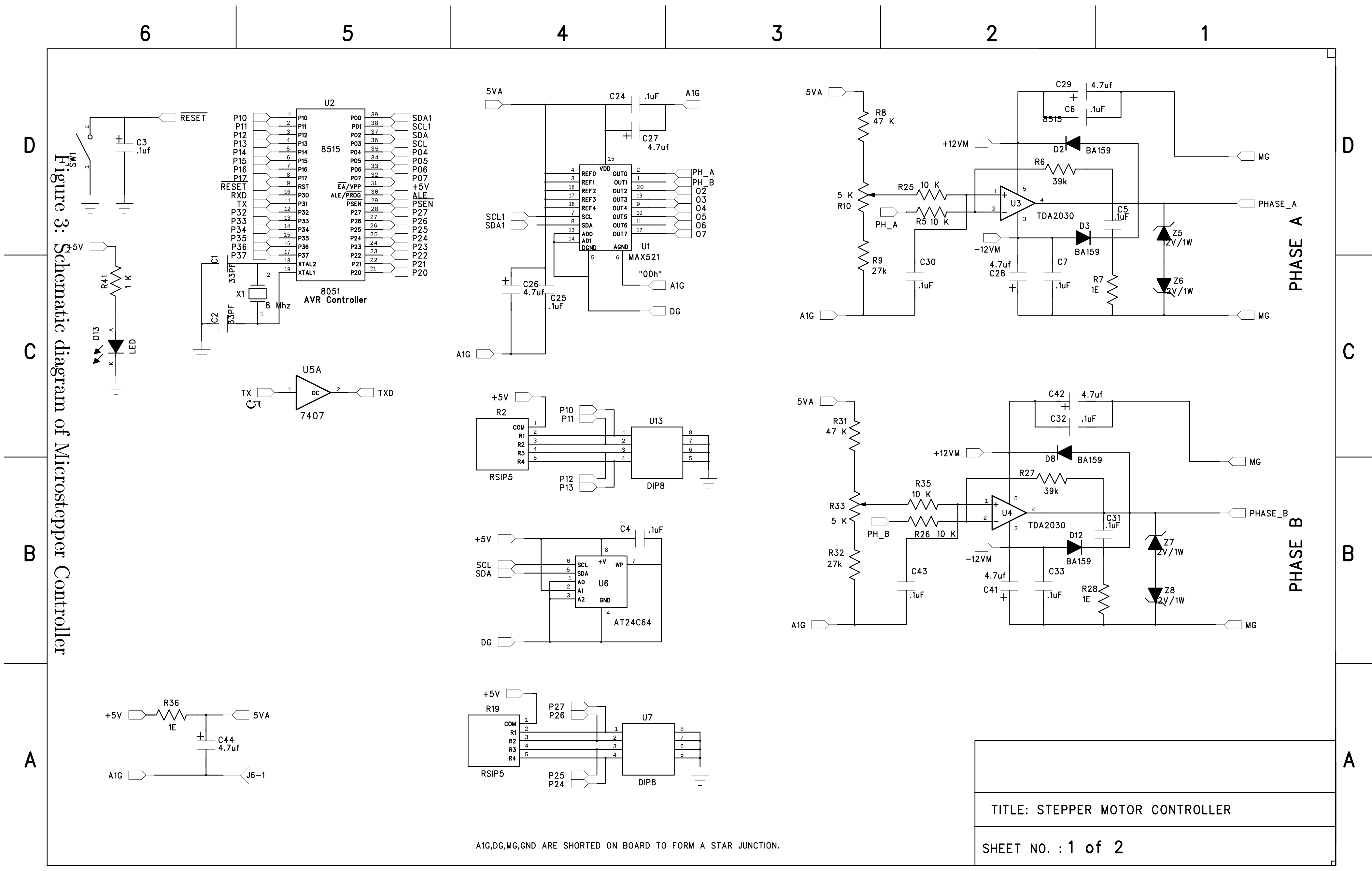

Figure 3: Schematic diagram of Microstepper Controller

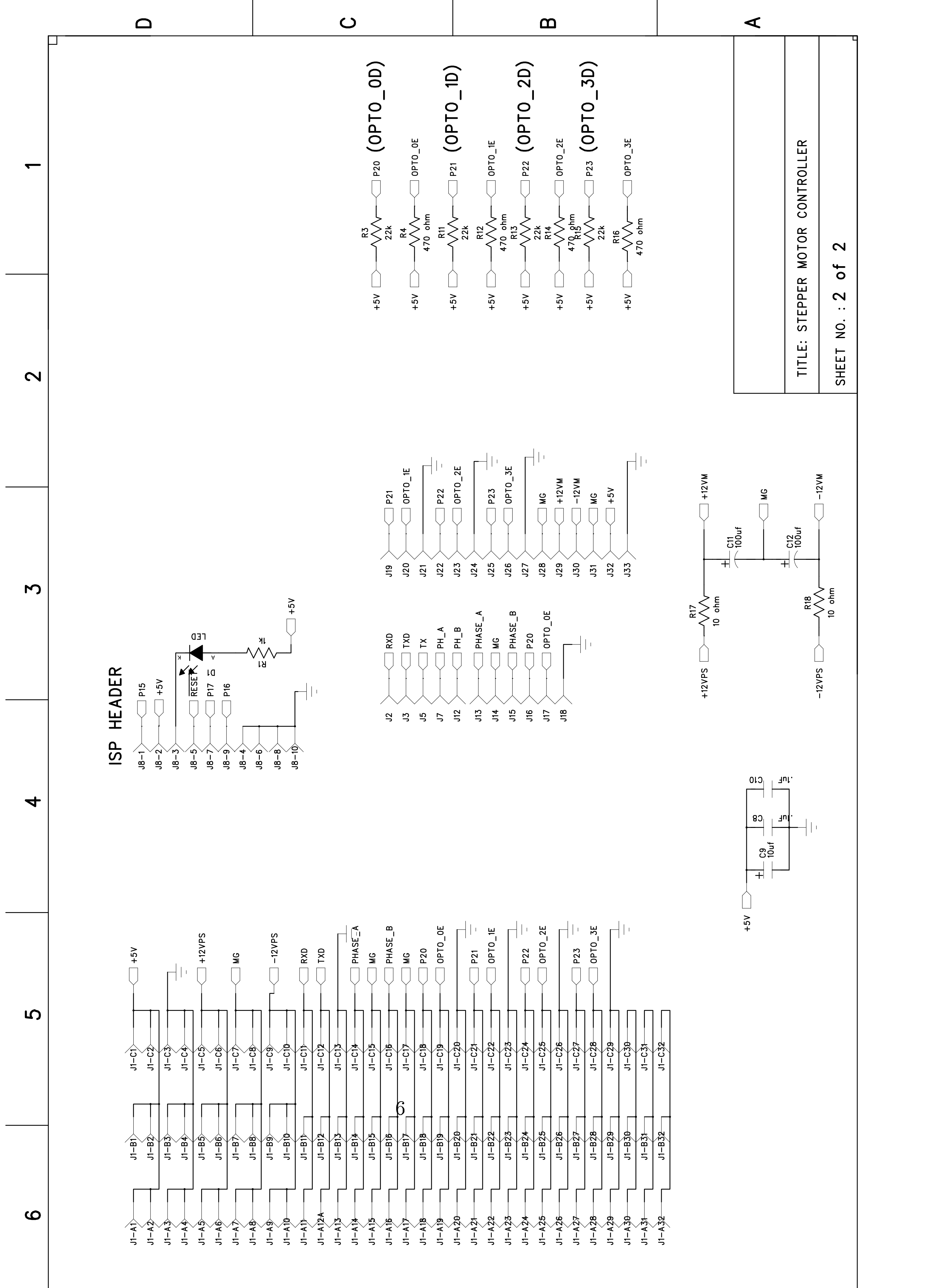

Figure 4: Schematic diagram of Microstepper Controller

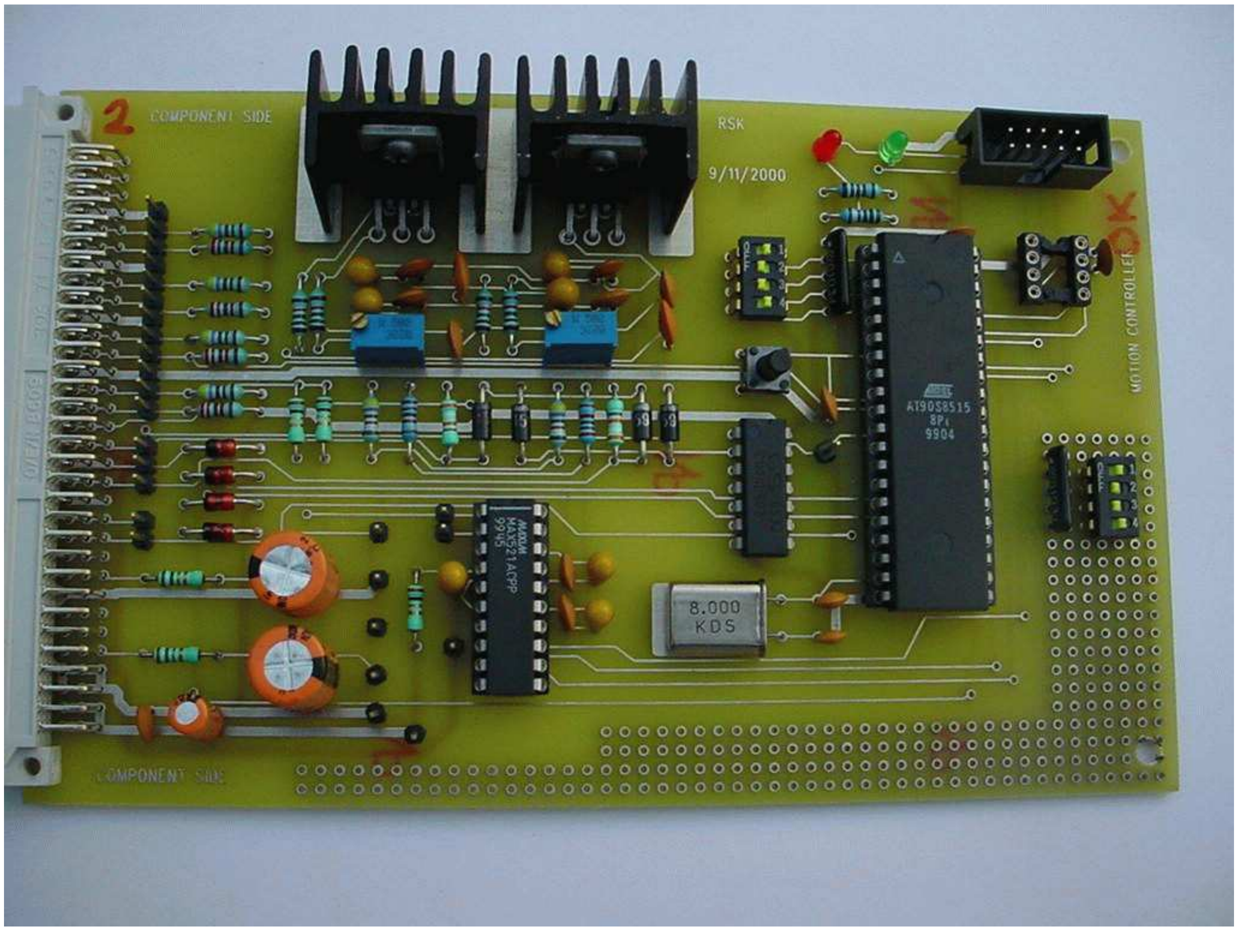

Figure 5: Picture of the Microstepper Controller PCB

4 Host and Motor Controller Communication Format

The communication between the host and the motor controller is at 19200 bps, 8 data bits, No parity and 1 stop bit.

The motor controller receives the commands on its serial port. Each motor controller is an AT90S8515. The motor controller maintains a 2 member command queue which means at any time, the slave can hold upto 2 commands. Any more command will not be received. Each command that is received will be acknowledged immediately. It will then be put in an execution queue. The acknowledgment will contain information as to whether the command can be actioned or whether the command reception had a problem or whether the command had a wrong format as specified in the next section. If the slave is executing a command and it gets a Status command, the Status command will be ignored. The status will indicate the current status of the slave.It will be always @ 0E TYPEXY OK HLT 87.

5 Command Structure

The command structure employed by the motor controller is generic. It shares the command structure with other device controllers such as Relay drivers, shutter controller etc. These other devices can be connected together for a given system as the requirement may be.

```
command:
@ NUM TypeXY command CS

Type:     M | R | E (M stands for Motor, R stands for Relay, E stands
                    for EEPROM constants)

XY:       number of the card. XY is a hex byte encoded as two ASCII
          characters.

          X will be always zero as only four way dip switch is used
          for address.
command:
```

```
(in the context of Type = M)

MA S3 S2 S1 S0
(move by S3:S2:S1:S0 microsteps.A is direction which
will be F for clockwise direction and R for counter
clockwise direction.
S3..S0 is a hex byte encoded as 2 ASCII characters.)

H1 Move to Home 1 position.

H2 Move to Home 2 position.

H3 Move to Home 3 position.

H4 Move to Home 4 position.

SS report the status of the motor.

(in the context of Type =R)

AO XXXX
(All relays ON for time XXXX. XXXX is a 16 bit number
encoded as 4 ASCII characters. Time unit is 100 ms.)

AF
(All relays Off)

O1 XXXX
(relay1 ON for time XXXX)

O2 XXXX
(relay2 ON for time XXXX)

O3 XXXX
(relay3 ON for time XXXX)

O4 XXXX
```

```
          (relay4 ON for time XXXX)

          F1
          (relay1 Off)

          F2
          (relay2 Off)

          F3
          (relay3 Off)

          F4
          (relay4 Off)

          SS
          (report the status of the relays)

response:
@ NUM TypeXY result values CS

NUM, Type and XY as defined previously.

result:
         OK
         (Command is accepted and actioned)

         CHKSUMERR
         (Error in the command reception)

         NOK
         (Error in the command like wrong combination etc.)
values:

(When the command is a SS (Status) command for the type M or R and
result is OK)

         HLT
         (specified motor is halted)


         O1|F1 O2|F2 O3|F3 O4|F4

(when the command is E for the type R)
         X1, X2..XN (as desired in the command)

(for other commands)

the response contains the argument of the command.
```

6 Design Code

The system was coded in C and the course code and other header files are included in the submission.