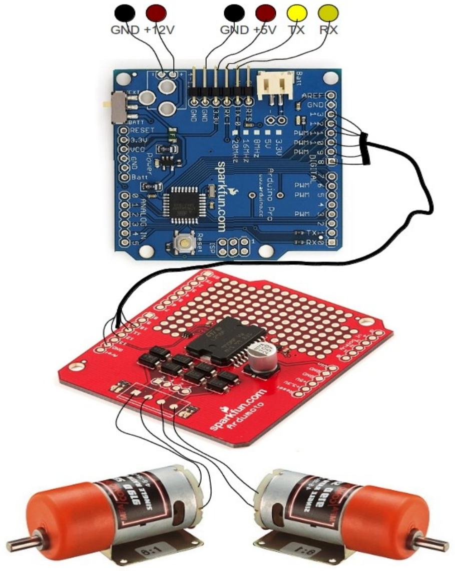
APEBOT
Motor Controller module

by Theo's Mechanic Ape
http://mechanicape.com

version 1.0
9 july 2011



Motordriver
Input: *+12V power, RX, TX, GND, RTS*
Output: 2 x 12V dc motor. Max 2A

Table of Contents

Motor controller

The motor controller is responsible for the following actions:

1. Turn the motors on/off
2. Turn the motors forward/backward

Hardware

The controller has the following external connections

1. +12V power (for the motors)
2. +5V power (for the communication)
3. GND - Ground (motors+communication)
4. TX - Send data (for sending serial data to host)
5. RX - Receive data (for receiving serial data from host)
6. CTS - Clear to send (unused)

Software API

The motordriver is communicating at 19200 baud. It accepts a command that is beginning with the character 'M' and followed by a commandbyte. The following commands are possible:

• “M” + char(1)	idle
• “M” + char(2)	forward
• “M” + char(3)	backward
• “M” + char(4)	rotate clockwise
• “M” + char(5)	rotate counter clockwise

The driver does not return information at this time. In the near future it will return 2 bytes indicating the actual left motor and right motor movement measured by a pulsedetector on the left and right axles.

Sourcecode

```
const int COMMAND_NONE=0;
const int COMMAND_IDLE=1;
const int COMMAND_MOVE_FW=2;
const int COMMAND_MOVE_BW=3;
const int COMMAND_TURN_CW=4;
const int COMMAND_TURN_CCW=5;


const int pinMotorLeftDirection=13;
const int pinMotorLeftPWM=3;
const int pinMotorRightDirection=12;
const int pinMotorRightPWM=11;

const boolean bMotorLeftForward=HIGH;
const boolean bMotorLeftBackward=LOW;
const boolean bMotorRightForward=LOW;
const boolean bMotorRightBackward=HIGH;
int receiveTimeout=0;
```

```
void setup()
{
  delay(3000);
  Serial.begin(19200);
  pinMode(pinMotorLeftDirection,OUTPUT);
  pinMode(pinMotorRightDirection,OUTPUT);
  pinMode(pinMotorLeftPWM,OUTPUT);
  pinMode(pinMotorRightPWM,OUTPUT);
  runCommand(COMMAND_IDLE);
}

void loop()
{
    int command=getSerialCommand();
    runCommand(command);
    delay(5);

}
```

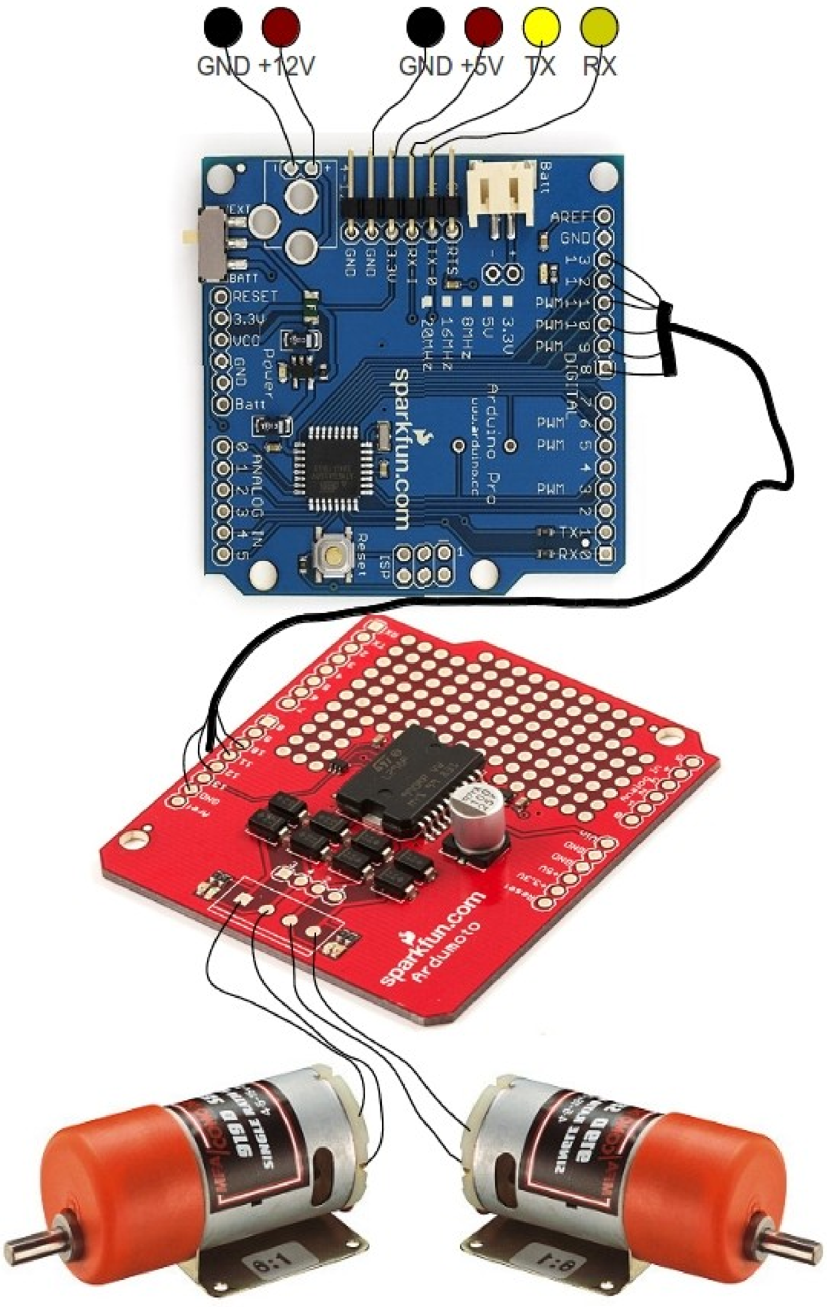
```
int getSerialCommand()
{
  int validatedCommand=COMMAND_NONE;
  if (Serial.available()>0)
  {
     receiveTimeout=0;
     int userCommand=Serial.read();
     Serial.flush();
     //Serial.println(char(48+userCommand));
     switch(userCommand)
     {
       case COMMAND_IDLE:     validatedCommand=COMMAND_IDLE;     break;
       case COMMAND_MOVE_FW:  validatedCommand=COMMAND_MOVE_FW;  break;
       case COMMAND_MOVE_BW:  validatedCommand=COMMAND_MOVE_BW;  break;
       case COMMAND_TURN_CW:  validatedCommand=COMMAND_TURN_CW;  break;
       case COMMAND_TURN_CCW: validatedCommand=COMMAND_TURN_CCW; break;
       case COMMAND_NONE:     validatedCommand=COMMAND_NONE;     break;
       default:               validatedCommand=COMMAND_IDLE;     break;
     }
  }
  else
  {
     receiveTimeout++;
  }
  return validatedCommand;
}
```

```
void runCommand(int command)
{
  switch(command)
  {
    case COMMAND_IDLE: doIdle();break;
    case COMMAND_MOVE_FW: doForward(); break;
    case COMMAND_MOVE_BW: doBackward(); break;
    case COMMAND_TURN_CW: doTurnCW(); break;
    case COMMAND_TURN_CCW: doTurnCCW(); break;
    case COMMAND_NONE:  doIdle(); break;
  }
}


void doIdle()
{
  doStop();
}
```

```
void doForward()
{
    setMotorLeft(bMotorLeftForward,255);
    setMotorRight(bMotorRightForward,255);
}

void doBackward()
{
    setMotorLeft(bMotorLeftBackward,255);
    setMotorRight(bMotorRightBackward,255);
}

void doTurnCW()
{
    setMotorLeft(bMotorLeftForward,255);
    setMotorRight(bMotorRightBackward,255);
}

void doTurnCCW()
{
    setMotorLeft(bMotorLeftBackward,255);
    setMotorRight(bMotorRightForward,255);
}

void doStop()
{
    setMotorLeft(bMotorLeftForward,0);
    setMotorRight(bMotorRightForward,0);
}
```

```
void setMotorLeft(boolean dir,int power)
{
   digitalWrite(pinMotorLeftDirection,dir);
   digitalWrite(pinMotorLeftPWM,power);
}

void setMotorRight(boolean dir,int power)
{
   digitalWrite(pinMotorRightDirection,dir);
   digitalWrite(pinMotorRightPWM,power);
}
```

Schema

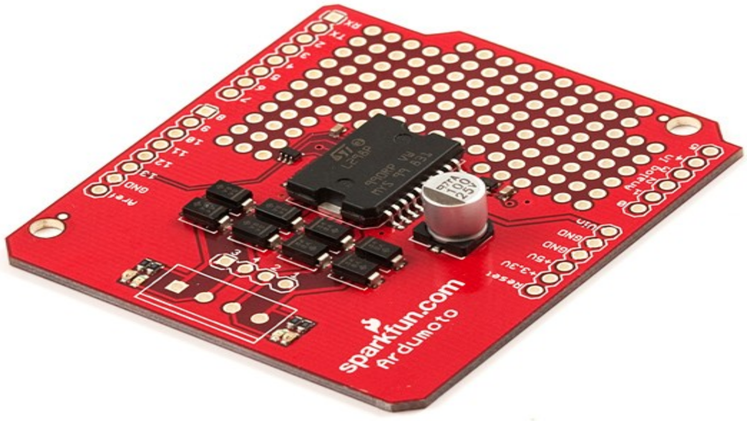


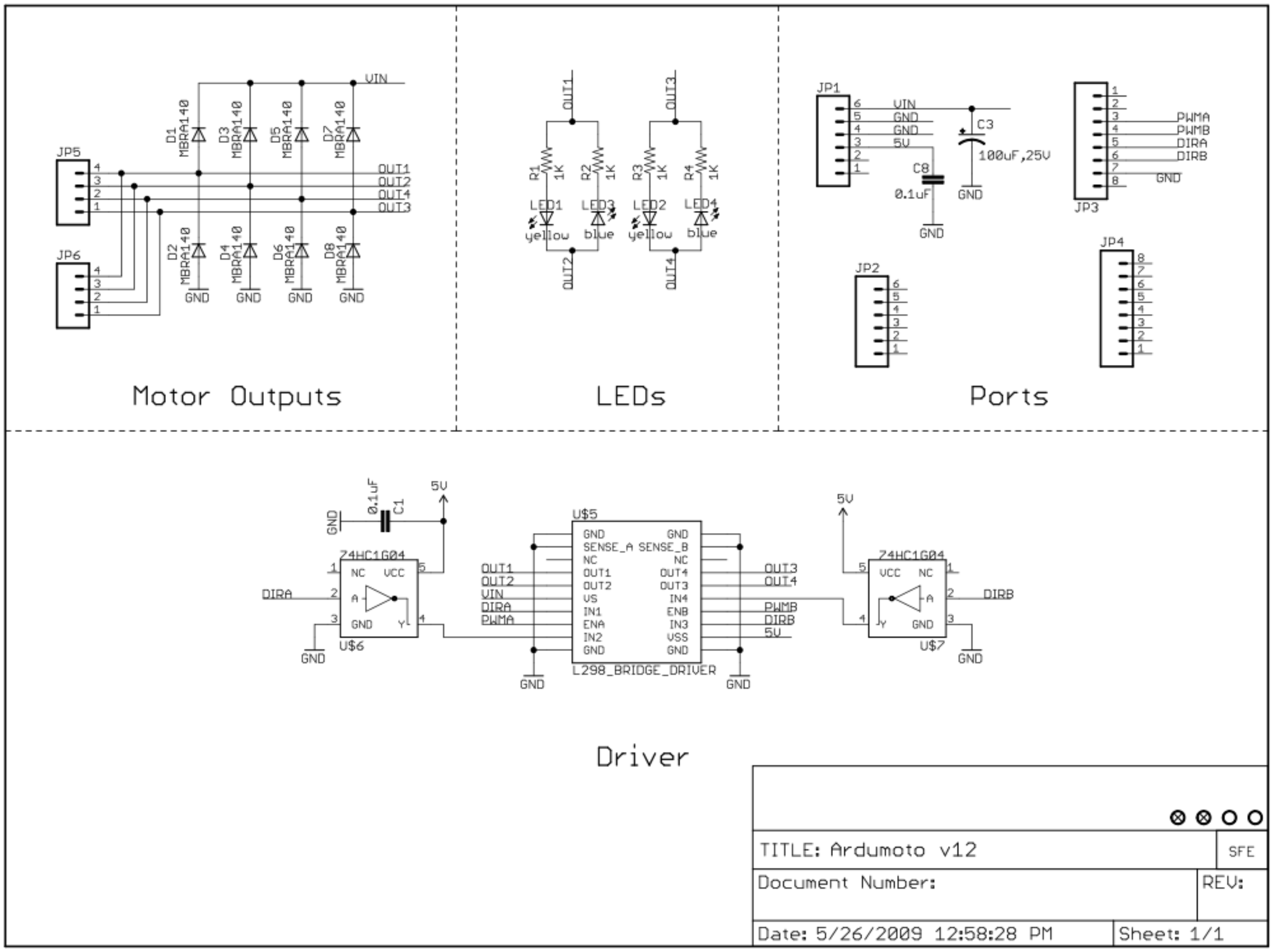
Motordriver

Input: *+12V power, RX, TX, GND, RTS*

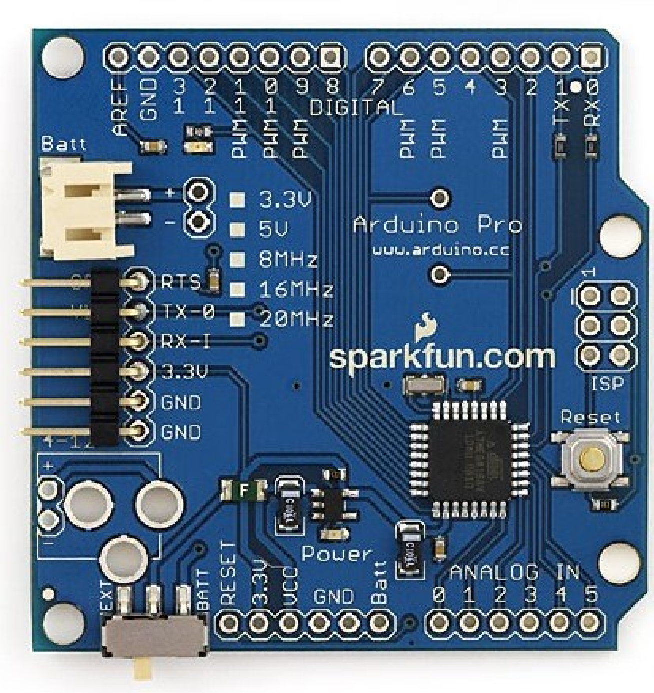
Output: 2 x 12V dc motor. Max 2A

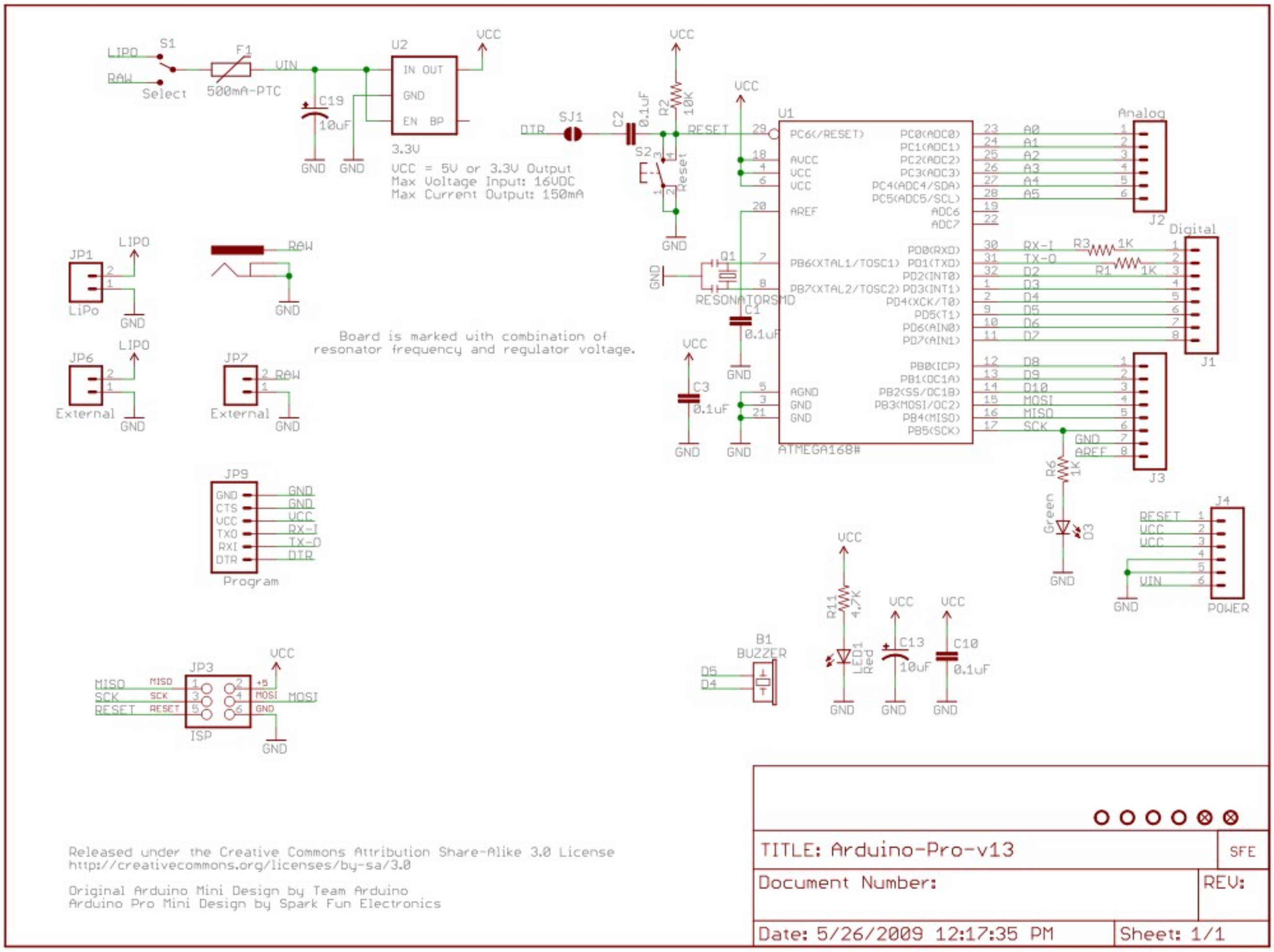
Sparkfun ArduMoto



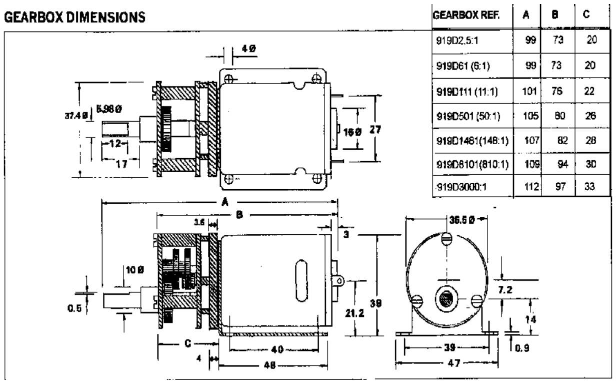


Sparkfun Arduino Pro



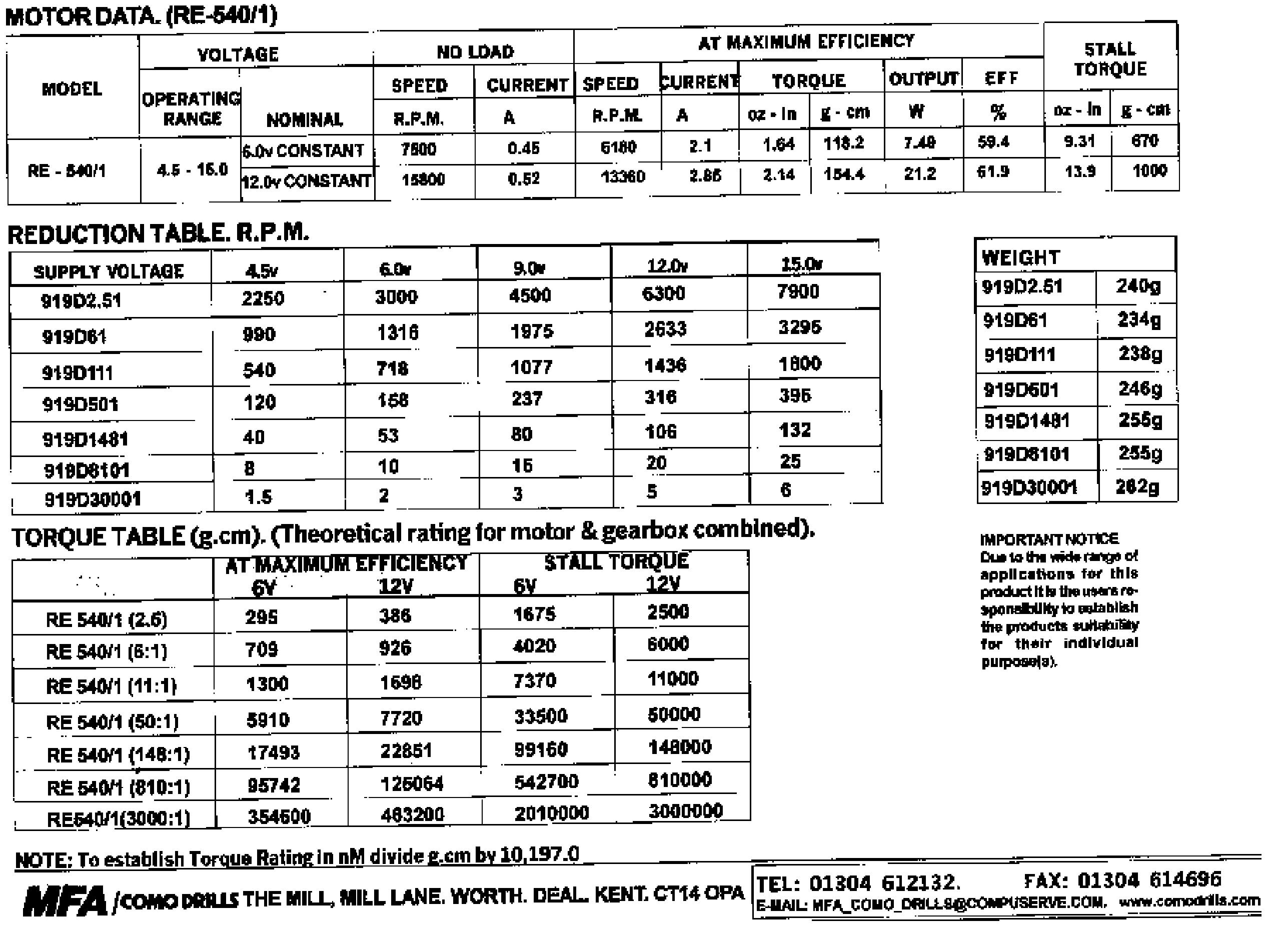


MFA/KOMO DC transmissionmotors



GEARBOX DIMENSIONS

GEARBOX REF.	A	B	C
919D2,5:1	99	73	20
919D61 (6:1)	99	73	20
919D111 (11:1)	101	76	22
919D501 (50:1)	105	80	26
919D1481(148:1)	107	82	28
919D8101(810:1)	109	94	30
919D3000:1	112	97	33



MOTOR DATA. (RE-540/1)

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY						STALL TORQUE	
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	EFF		
			R.P.M.	A	R.P.M.	A	oz - in	g - cm	W	%	oz - in	g - cm
RE - 540/1	4.5 - 15.0	6.0v CONSTANT	7500	0.45	6180	2.1	1.64	118.2	7.48	59.4	9.31	670
		12.0v CONSTANT	15800	0.52	13360	2.85	2.14	154.4	21.2	61.9	13.9	1000

REDUCTION TABLE. R.P.M.

SUPPLY VOLTAGE	4.5v	6.0v	9.0v	12.0v	15.0v
919D2.51	2250	3000	4500	6300	7900
919D61	990	1316	1975	2633	3295
919D111	540	718	1077	1436	1800
919D501	120	158	237	316	395
919D1481	40	53	80	106	132
919D8101	8	10	15	20	25
919D30001	1.5	2	3	5	6

WEIGHT	
919D2.51	240g
919D61	234g
919D111	238g
919D501	246g
919D1481	255g
919D8101	255g
919D30001	262g

TORQUE TABLE (g.cm). (Theoretical rating for motor & gearbox combined).

	AT MAXIMUM EFFICIENCY		STALL TORQUE	
	6V	12V	6V	12V
RE 540/1 (2.5)	295	386	1675	2500
RE 540/1 (6:1)	709	926	4020	6000
RE 540/1 (11:1)	1300	1698	7370	11000
RE 540/1 (50:1)	5910	7720	33500	50000
RE 540/1 (148:1)	17493	22851	99160	148000
RE 540/1 (810:1)	95742	125064	542700	810000
RE540/1(3000:1)	354600	463200	2010000	3000000

IMPORTANT NOTICE
Due to the wide range of applications for this product it is the users responsibility to establish the products suitability for their individual purpose(s).

NOTE: To establish Torque Rating in nM divide g.cm by 10,197.0

MFA/COMO DRILLS THE MILL, MILL LANE. WORTH. DEAL. KENT. CT14 OPA
TEL: 01304 612132. FAX: 01304 614696
E-MAIL: MFA_COMO_DRILLS@COMPUSERVE.COM. www.comodrills.com