

DCC Dual Power Breaker – Technical Details

Introduction

The circuit schematic of the Dual Power Breaker (DPB) is shown in [1] on the next page, followed by a brief description of how everything operates and fits together. There are then details of the commands which you can use to set and read back the various DPB parameters and to control its various functions. The final section describes how the DPB current measurement can be calibrated using a RRampMeter.

Power Supplies

Power for the DPB is supplied from the DCC track output of your command station or booster which is normally set at around 14 volts. This is rectified by diode bridge BR1 to supply the voltage regulator VR1 which, in turn, supplies +9 volts to the Arduino Nano, to operational amplifiers IC1 and IC2, and to the DC-DC voltage converter IC3. In turn, the internal voltage regulator of the Arduino Nano supplies +5 volts to optoisolator OK1, and voltage converter IC3 supplies -9 volts (derived from the +9 volts input) to operational amplifiers IC1 and IC2.

The overall current consumption of the DPB varies between 30 and 90mA depending on the activity of the Arduino Nano.

The input DCC signal is also connected to the input of optoisolator OK1 via resistor R1. Capacitor C4 filters out any high-voltage spikes from the track, and diode D1 prevents the optoisolator input diode from being fatally reverse-biased by the negative-going part of the DCC signal. This DCC input circuitry is adapted from the design by Wolfgang Kuffer (<https://mrrwa.org/dcc-decoder-interface/>) and is the same as that used by Geoff Bunza as the basis for several of his projects.

The output from optoisolator OK1 is a replica of the DCC waveform, but at a safe +5 volt level, so that DCC command packets can be input to digital input D2 of the Arduino Nano module. Here they are decoded by the NmraDcc library functions, from which relevant DCC accessory and program-on-the-main commands can be passed to the DPB Arduino Nano sketch code.

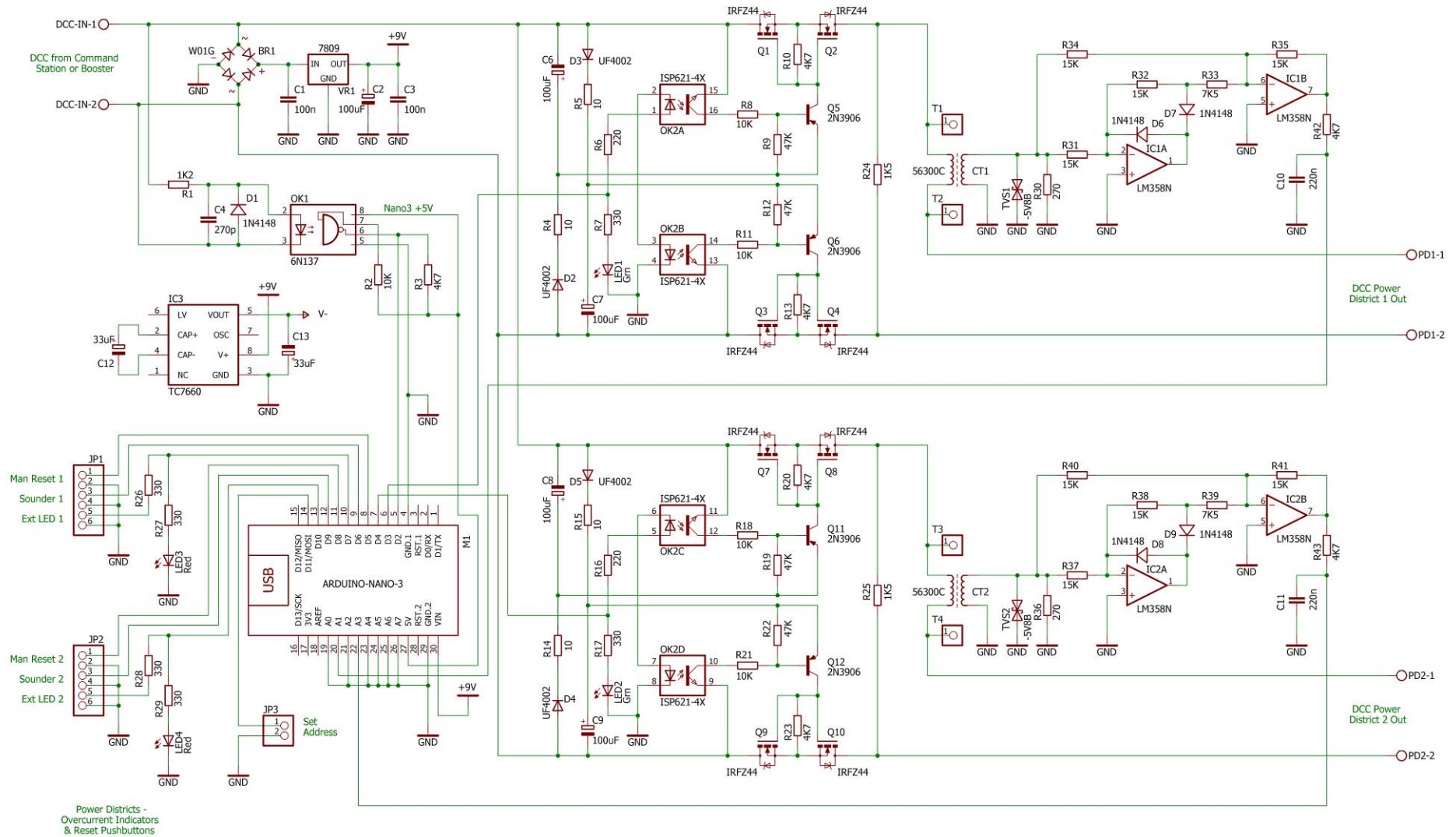
Arduino Nano Connections and Functions

All functions of the DPB are controlled from pins of the Arduino Nano –

- D3 – Enable Power District 1 DCC Output + LED1 (Green) lit when HIGH
- D4 – Enable Power District 2 DCC Output + LED2 (Green) lit when HIGH
- D5 – Manual Reset District 1 Input – LOW to activate (connect to 0V (GND))
- D6 – Sounder Alarm District 1 Output – set HIGH when Break Active
- D7 – Break Active District 1 – LED3 (Red) and External LED 1 lit when HIGH
- D8 – Manual Reset District 2 Input – LOW to activate (connect to 0V (GND))
- D9 – Sounder Alarm District 2 Output – set HIGH when Break Active
- D10 – Break Active District 2 – LED4 (Red) and External LED 2 lit when HIGH
- D11 – Set Address Input – LOW to activate (connect to 0V (GND) via jumper)
- A1 – Current Sense District 1 Input – from IC1B output
- A3 – Current Sense District 2 Input – from IC2B output

Pins A0, A2 and A4-A7 are all connected to 0V (GND) to minimise analog input noise and crosstalk.

The Arduino Nano monitors measurement of the current drawn by each District, and sets the timing of when to initiate a break and then when to attempt reconnection. See the software sketch listing for further details.



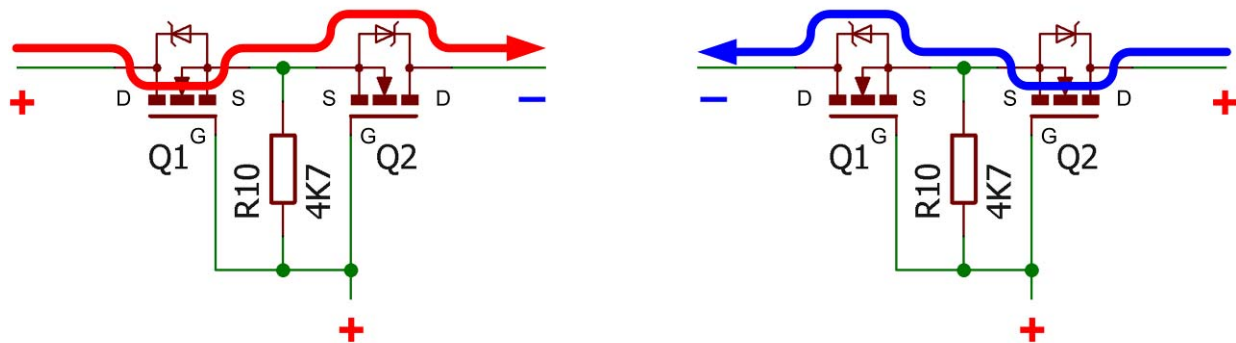
1. DCC Dual Power Breaker – Circuit Schematic

Power Breaker

The two output lines to each power district are each controlled by a pair of back-to-back N-channel power MOSFETs (metal-oxide-silicon field effect transistors), Q1-Q4 for Power District 1, and Q7-Q10 for Power District 2. Use of pairs of MOSFETs is required because the DCC output is AC rather than DC, so that the polarity of each power district output line alternates between positive and negative at a frequency of around 6kHz.

The operation of this type of switch is best explained by looking at the connections of Q1 and Q2. The sources and gates of both MOSFETs are connected together, and both devices are switched on by applying a positive voltage to the common gate connection.

Nevertheless, it is important to note that a MOSFET only conducts when its drain is positive with respect to its source, even when the gate is held positive. Hence, in the left part of diagram [2], with the DCC input line (DCC-IN-1) positive, Q1 will conduct whereas Q2 will not, despite being nominally switched on. However, the internal construction of MOSFETs leads to them having what is known as a ‘body diode’ which allows current to flow past the device when the drain is reverse-biased – so that current flows through both Q1 and Q2 following the path indicated by the red line.



2. Conduction paths depending on DCC input line polarity

When the DCC-IN-1 input line changes to negative, as in the right part of diagram [2], Q2 switches on to conduct the track current while Q1 is off but allows the current to flow through its body diode, following the blue line.

When the positive gate voltage is removed, both Q1 and Q2 will switch off so that no current flows through the MOSFETs themselves. Current cannot flow through the body diodes either in this situation, since they are connected in opposite directions, hence DCC power to the district is completely removed.

Each pair of MOSFETs is provided with its own independent positive power supply for use by the gate drive circuitry. In the case of Q1 and Q2, this consists of capacitor C6 which is charged to around +14V from the opposite DCC input line (DCC-IN-2) via resistor R4 and fast rectifier diode D2. The current drain imposed by the gate control transistor Q5 and the associated optoisolator OK2A is sufficiently low that the voltage across C6 does not decrease appreciably when the DCC input polarity reverses and charging of C6 via R4 and D2 is paused.

A positive power supply for the other pairs of MOSFETs is provided by capacitors C7, C8 and C9 together with their associated charging components D3-D5 and R5, R14 and R15.

It is important to note that these positive power supplies are not referenced to 0V (GND) in any way, but float up and down in absolute voltage level as the DCC lines switch polarity. Hence the necessity to couple the power district enable signals from the Arduino Nano to the gate control transistors via optoisolators OK2A-OK2D.

If we consider the power district switch formed by Q1- Q4 and their associated circuitry again, the Power District 1 enable signal, provided by Arduino Nano pin D3 being set HIGH, activates the inputs to optoisolators OK2A and OK2B, which are connected in series, via resistor R6. The signal from D3 also switches on LED1 via R7.

With optoisolator OK2A activated, its output connects resistor R8 to DCC line DCC-IN-1 and so switches gate control transistor Q5 on. This brings the collector of Q5 close to the positive supply line provided by C6, and so supplies the necessary gate drive to Q1 and Q2, switching them on and enabling DCC power to output line PD1-1 of Power District 1.

Similar actions via OK2B, R11 and Q6 switch MOSFETs Q3 and Q4 on, which likewise enable DCC power to output line PD1-2 of Power District 1.

Independent control of Power District 2 is exercised similarly from Arduino Nano pin D4 to enable or disable the switching of MOSFETS Q7-Q10.

Resistors R24 and R25 provide small constant loads across the Power District outputs to minimise the feedthrough of switching spikes via the inherent capacitance of the MOSFETs.

Note that, due to the very low resistance of the MOSFETs when switched on (0.0175 ohm), they do not dissipate appreciable power even when carrying high currents. Even when carrying a maximum continuous current of 5A (definitely not a recommended operating condition) the dissipation in each device will only be 0.4375 watts which will lead to a temperature rise of less than 30 degC (54 degF) – which can be handled by the device's TO220 package without a heatsink. In normal operation, handling currents of up to 3A, dissipation would be less than 160mW with a correspondingly smaller temperature rise of less than 10 degC (18 degF).

Current Sense and Measurement

The current drawn by each power district is sensed using taking one of the district's output lines through the winding of a 300:1 current transformer, using a simple wire loop. This imposes no significant load or voltage drop on the power district output.

Considering Power District 1, for a current of 1 amp drawn from its output through the wire loop fitted as the current transformer CT1 primary (between terminals T1 and T2), the current through the CT1 secondary will be 3.33mA which will develop 0.9V across load resistor R30.

A bidirectional transient voltage suppressor TVS1 is also fitted across the current transformer secondary to protect the current sensor circuitry against any harmful voltage spikes fed through from the power district.

Since the voltage across R30 is essentially a small version of DCC, and therefore alternating, it needs to be rectified before its magnitude can be easily measured. Hence IC1A and IC1B form a precision rectifier where the operational amplifiers eliminate the voltage drop which would occur if a simple diode rectifier was used.

IC1A via resistors R31, R32 and diode D7 acts as a unity-gain half-wave precision rectifier which produces a positive output equal in magnitude to the negative half-cycles of the input DCC waveform. Diode D6 maintains IC1A in closed-loop operation during the positive half-cycles, preventing it going into saturation.

IC1B via resistors R33, R34 and R35 is another unity-gain amplifier which adds in the positive half-cycles of the DCC input resulting in full-wave rectification of the sensed current. The output of IC1B is, therefore, a voltage which is directly proportional to the current drawn in Power District 1 and which is transferred via the simple low-pass filter formed by R42 and C10 to an Arduino Nano analog-to-digital converter (ADC) connected to pin A1.

Current sensing for Power District 2 operates in exactly the same way, using current transformer CT2 and a precision rectifier formed by IC2A and IC2B.

Note that both IC1 and IC2 operate from ± 9 volt power supplies to provide sufficient headroom to handle the input DCC signal without their outputs limiting and producing a non-linear response.

Although the theoretical conversion figure of 0.9V/amp could be used by the Nano sketch software to compute the current drawn by each power district, in practice the performance of the precision rectifier does differ from the ideal, although still producing a linear response.

Hence, several DPBs were built and calibrated by measuring the current drawn by the power districts over the range of 0.25A to 5A, using a RRampMeter, and recording the corresponding values output by the Arduino Nano ADCs. Results were very consistent from several varieties of Arduino Nano (within 2%) and were used to provide a table within the sketch to set the desired break current for each district. Further details of the calibration procedure are given in the final section of this document.

Accessing CV Values

The DPB operates as a DCC accessory and is controlled by the set of configuration variables (CVs) listed in Table 1 [3].

CV Number	Default Value	Description	
41	31	Power Breaker Address LSB	Address can range from 0001 to 2043
42	0	Power Breaker Address MSB	
43	6	District 1 Current Limit	Set in 0.25 amp steps, from 0.25A to 5A Step values 1 to 20
44	6	District 2 Current Limit	
45	25	District 1 Trip Delay	Time before breaker acts, from 5 to 255 msec
46	25	District 2 Trip Delay	
47	12	District 1 Reconnect Delay	Time before breaker attempts to reconnect Steps of 250msec, 0.25s to 60s (1- 240)
48	12	District 2 Reconnect Delay	
49	0	Use Dual Power Breaker as Autoreverser if CV = 90	
50	0	Set all CVs to default values if CV50 = 0 – otherwise CV50 = 173	
51	0	Not used at present	
52	10	Auto reconnect District 1 if CV52=10, else reconnect with Manual pushbutton	
53	10	Auto reconnect District 2 if CV53=10, else reconnect with Manual pushbutton	
54	0	Enable output of current values measured by Nano ADCs to the Arduino IDE Serial Monitor if CV54 not zero	
55	0	Enable output of sketch debug messages to Serial Monitor if CV55 not zero	
56	0	Not used at present	

3. Table 1 - Dual Power Breaker configuration variables

A large part of the Arduino Nano sketch is devoted to providing the command interface by which the various parameters used by the DPB can be set and read back via the Nano's USB interface and the Arduino IDE Serial Monitor. The command set, as shown in Table 2 [4], lets you set all operational parameters simply by typing these character strings into the Arduino Serial Monitor Message box and pressing the 'Enter' key. The Serial Monitor must be set to a data rate of 38400 baud, and the termination character should be left at the default of 'New Line'.

DCC Dual Power Breaker – Technical Details

Description	Command	Notes	Response
NOP	N	Clears Debug and Output ADC Flags	"OK"
Return CV Status	S	Displays 16 values, comma separated	"CVs - " + CV41–CV56 values
Return Address	A		"Address = nnnn"
Read Version	V	Displays Major.Minor version	eg. "3.8"
Enable Debug Messages	G		"OK"
Enable ADC Display	K	Displays ADC values, 5 times/sec	"OK" followed by "Dist 1 ADC = " xxx – "Dist 2 ADC = " xxx
Set Address	Tnnnn	Replace 'nnnn' with up to 4 digits for new address – error message is displayed if outside range 1 - 2043	"Address = nnnn"
Set Trip Current District 1	C1nn	Replace 'nn' with Index value 1 - 20 (Trip current = Index x 0.25A)	"Dist 1 Trip Current = x.xx A"
Set Trip Current District 2	C2nn		"Dist 2 Trip Current = x.xx A"
Set Trip Delay District 1	D1nnn	Replace 'nnn' with delay value in range 5 to 255msec	"Dist 1 Trip Delay = xxxmsec"
Set Trip Delay District 2	D2nnn		"Dist 2 Trip Delay = xxxmsec"
Set Reconnect Delay Dist 1	R1nnn	Replace 'nnn' with Index value 1 - 240 (Delay = Index x 0.25sec)	"Dist 1 Reconnect Delay = xx.xxsec"
Set Reconnect Delay Dist 2	R2nnn		"Dist 2 Reconnect Delay = xx.xxsec"
Set Auto Reconnect Dist 1	M1A		"Dist 1 Reconnect = Automatic"
Set Manual Reconnect Dist 1	M1M		"Dist 1 Reconnect = Manual"
Set Auto Reconnect Dist 2	M2A		"Dist 2 Reconnect = Automatic"
Set Manual Reconnect Dist 2	M2M		"Dist 2 Reconnect = Manual"
Reset CVs to Factory Default	F		"CVs Reset"
Break Power District 1	B1	If in Autoreverser Mode these commands just generate an error message – with no change of which power district is enabled	"Dist 1 Switched Off"
Break Power District 2	B2		"Dist 2 Switched Off"
Break Power Both Districts	BB		"Dists 1 & 2 Switched Off"
Resume Power District 1	P1		"Dist 1 Switched On"
Resume Power District 2	P2		"Dist 2 Switched On"
Resume Power Both Districts	PB		"Dists 1 & 2 Switched On"
Enable Autoreverser Mode	UE	Dist 1 enabled, Dist 2 disabled	"AutoReverser Mode Enabled"
Disable Autoreverser Mode	UD	Leaves both Districts disabled	"AutoReverser Mode Disabled - Both Districts Switched Off"

4. Table 2 – Dual Power Breaker interface command set

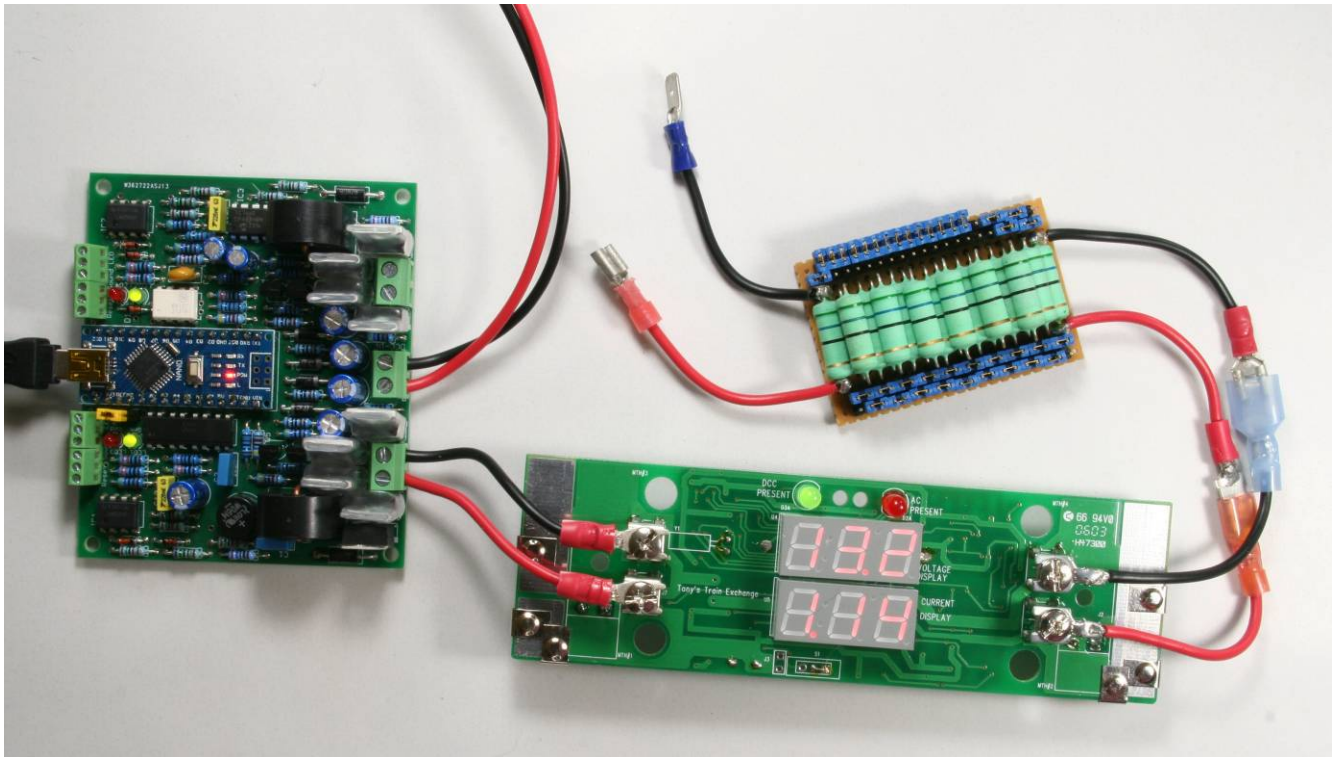
You can use either upper or lower case letters for all alphabetic commands and parameters, and all numeric parameters and result values are decimal. Note also that, in the case of the numeric parameters which follow the T, C, D, and R commands, any leading zeros are ignored.

Current Measurement - Calibration

The current drawn from a DPB power district is determined by measuring the voltage output by the district's precision rectifier circuitry using the Arduino Nano's inbuilt analog-to-digital converters (ADCs), as described earlier in this note.

Rather than rely on theoretical values calculated from the relevant DPB component values, a table was created of ADC output values for a range of district currents measured using a proprietary RRampMeter. The design of the RRampMeter is optimised to give accurate measurements of voltage and current in a DCC system operating with a square waveform at around 6kHz, unlike standard multimeters which are designed to measure sinusoidal AC voltages and currents at 50 or 60Hz (and which cannot be relied upon to give accurate results in a DCC system).

The configuration used is shown in [5] –



5. Using a RRampMeter to measure current output from the DPB

A set of 20 56ohm 5watt resistors was used to provide a load for each power district. Resistors were successively connected in parallel across each DPB district output to draw currents over a range of approximately 0.15 to 4.30A. In [5], five 56ohm resistors are connected, drawing 1.14A at 13.2V from power district 1.

While the resistors were connected, the corresponding ADC values were displayed continuously (every 200msec) in the Arduino Serial Monitor window by using the 'K' command, as shown in [6]. Note that the connection to the resistor assembly was only maintained for the few seconds it took for the ADC values to be seen as reasonably stable and recorded, since the resistors become very hot. The DCC output from the power district does include a fair amount of electrical noise so that, as can be seen in [6], the ADC values do vary by a small amount (from 223 to 226). Here, an average (or most frequent) figure of 225 would be taken.



5. Output of DPB ADC values in Arduino Serial Monitor

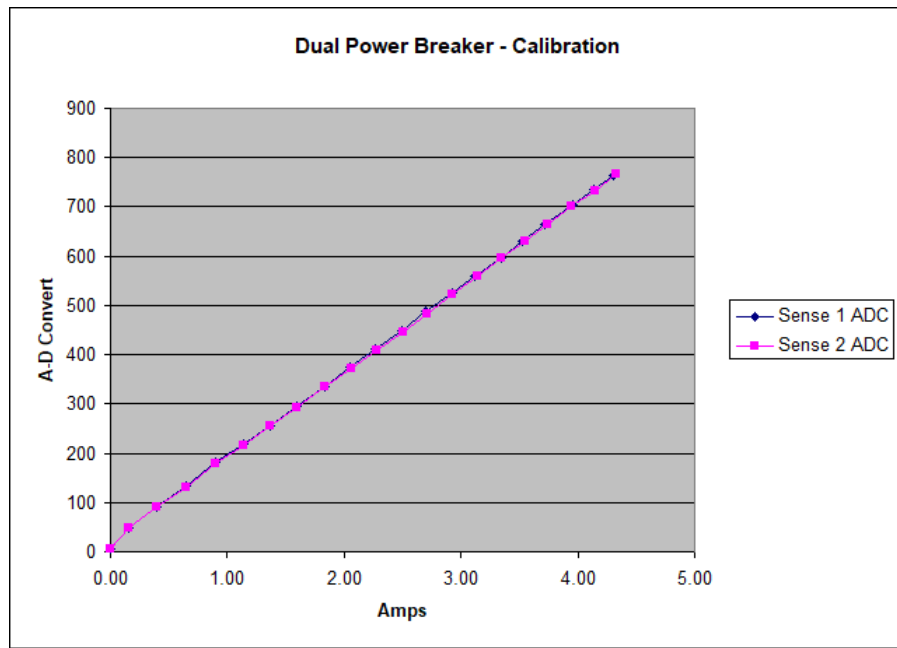
A complete set of measurements for both power districts was then entered into a spreadsheet as shown in [6] in columns A-D. The responses are essentially linear, as can be seen when they are plotted graphically as in [7], and the next step is to generate a table of expected ADC values for a set of fixed trip currents which have been entered in column F.

	A	B	C	D	E	F	G	H	I	J
1										
2	Load (A)	Sense1 ADC	Load (A)	Sense2 ADC						
3	0.00	5	0.00	5						
4	0.15	48	0.15	48						
5	0.40	91	0.40	90						
6	0.65	134	0.65	132						
7	0.90	181	0.90	179						
8	1.14	218	1.14	216						
9	1.37	256	1.37	255						
10	1.60	296	1.60	293						
11	1.83	336	1.83	334						
12	2.05	374	2.06	373						
13	2.27	412	2.28	410						
14	2.49	450	2.50	446						
15	2.70	488	2.71	483						
16	2.93	526	2.93	522						
17	3.12	560	3.14	559						
18	3.34	595	3.34	596						
19	3.53	631	3.55	630						
20	3.72	665	3.74	665						
21	3.96	705	3.95	700						
22	4.14	734	4.15	732						
23	4.30	765	4.33	766						
24	5.00		5.00							
25										
26	x	y	a	b						
27	0.65	134.00	171.52	22.52						
28	3.95	700.00								
29										

Formulas and Annotations:

- Cell G2: $=($D$27*F5)+$E27
- Cell I2: $=G5-5$
- Cell G24: $=($D$27*F24)+$E27
- Cell I24: $=G24-5$
- Cell B28: $=(B28-B27)/(A28-A27)$
- Cell D28: $=B28-(((B28-B27)/(A28-A27))*A28)$

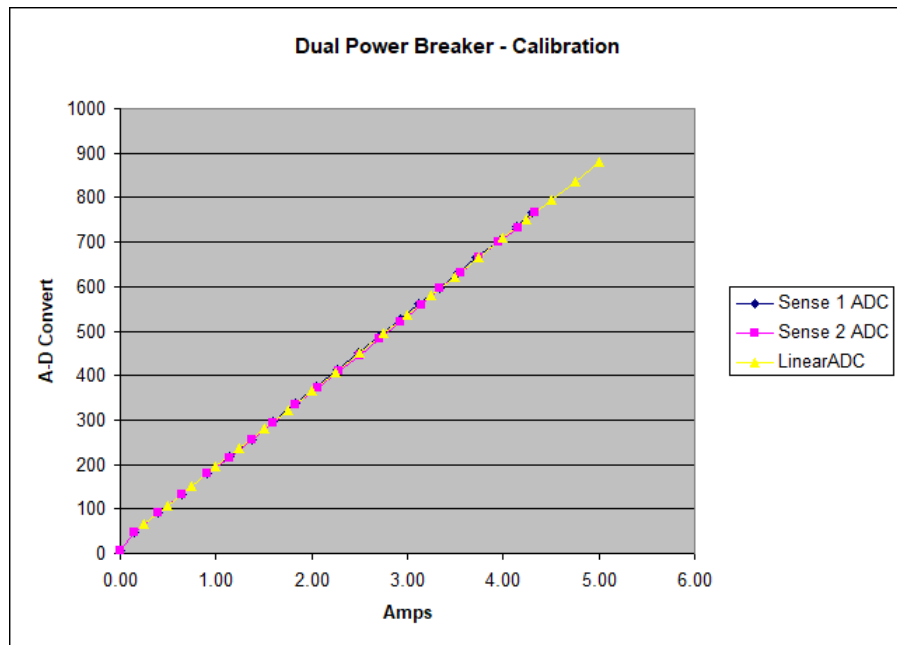
6. Recorded DPB ADC values entered to spreadsheet



7. Plot of recorded DPB ADC values

To generate the expression to calculate the trip current table, the method I used was to pick a pair of values (one from each ADC set), at low and high currents, which appeared to lie on the best straight line through all the recorded points. These were copied to cells A27 – B28 on the spreadsheet, as shown in [6] – labelled x and y. Formulae on the spreadsheet then calculate two values in cells D27 and E27, as shown. These values, a and b, are used, in turn, to calculate the expected ADC values, in column G, for the 20 values of load current entered in column F.

For the mathematically inclined, the spreadsheet simply solves a pair of simultaneous equations to generate a solution to the straight line represented by the expression $y = ax + b$, which is plotted against the measured points in [8].



8. Plot of straight line through recorded DPB ADC values

Note that, when the RRampMeter records zero load current, the Nano ADCs show a count of 5 which represents the current taken by the RRampMeter internal electronics (about 30mA). Hence, to produce the final ADC values for the trip current table (in column I), 5 was subtracted from the values in column G, and the 20 column I values then copied to the trip_slct[] array in the DPB sketch (DualPowerBreaker_3-8.ino) at line 41.

Rather than just use a single set of measurements from one DPB, I recorded sets from a number of DPBs constructed using Arduino Nanos from different batches, then averaged the results before calculating the final trip_slct[] values to be used.

You can find a copy of the basic spreadsheet, together with all of the other relevant files) on the DPB download page of my website at <https://www.a-train-systems.co.uk/dpb-download>.