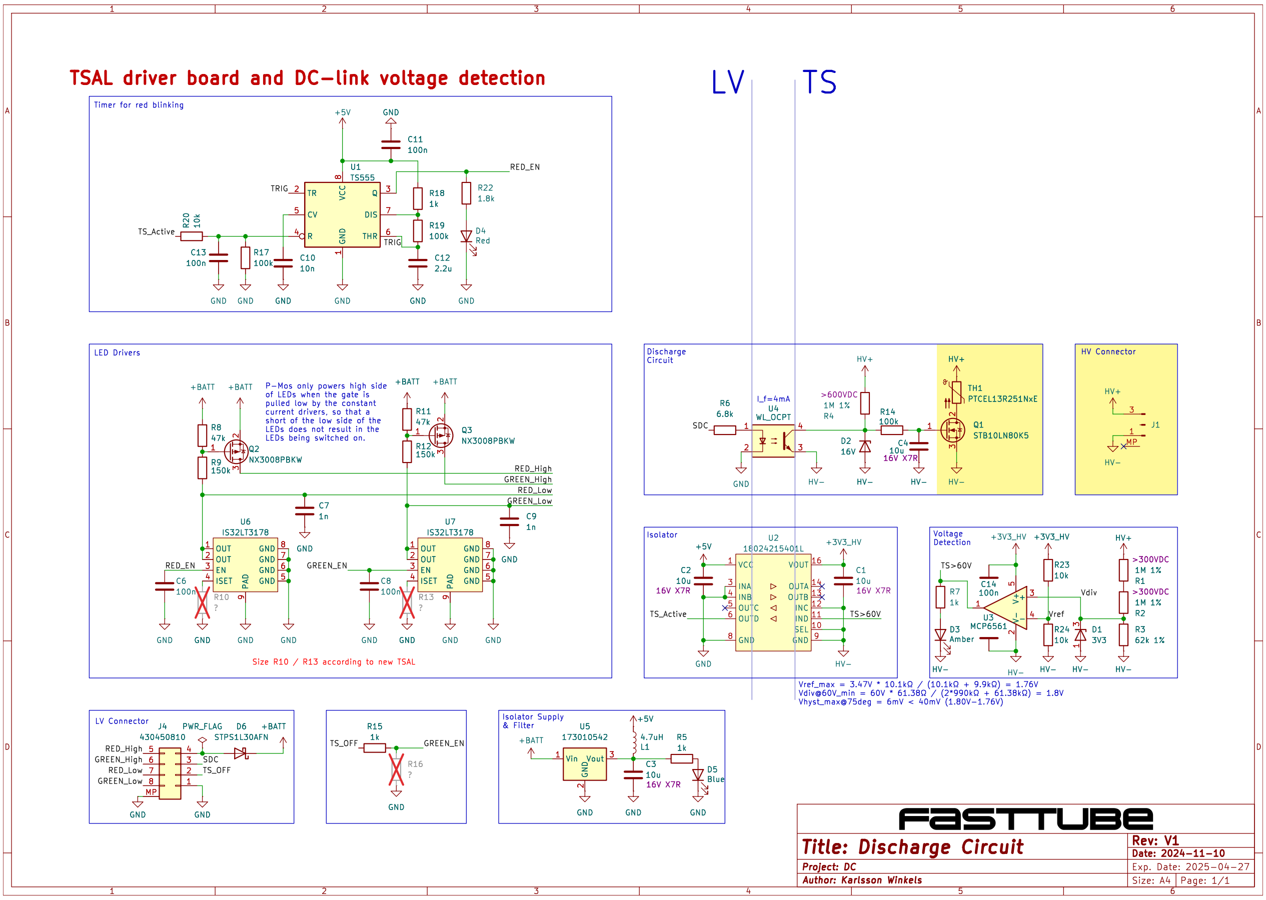

Figure 1: Schematic of the Discharge Circuit PCB

Discharge Time

As seen in the Schematic, for our discharge circuitry a PTC (PTCEL13R251NxE[2]) is used. The total capacitance of the DC-link capacitor from the two inverters (Emsiso emDrive H100[1]) that we are using is about 200 µF, and the maximum voltage of the accumulator is 403.2V. Using the RC discharging circuit equation, we obtain the highest resistance that the PTC can have so that we are still within the 5s discharge limit.

$$V_C = V_0 \cdot e^{-t/RC} \tag{1}$$

$$60\,\mathrm{V} = 403.2\,\mathrm{V} \cdot e^{-5\,\mathrm{s}/(R_{PTC} \cdot 200\,\mu\mathrm{F})} \tag{2}$$

$$R_{PTC} \approx 13123\,\Omega \tag{3}$$

To calculate how many discharge attempts can be made before the discharge time exceeds 5 s, we first determine the temperature at which the PTC has a resistance of 13123 Ω. This value can be obtained from the PTC's datasheet (see Fig. 2) [2].

From the graph, the corresponding temperature is approximately 165 °C. We assume the PTC reaches this temperature instantly after each discharge and that heat dissipation is negligible (since the thermal time constant τ_{th} is 130 s).

To determine the maximum allowable thermal energy before the PTC cools down, we assume an ambient temperature of 45 °C. Given the thermal capacity $C_{th} = 1.45\,\mathrm{J\,K^{-1}}$, the maximum thermal energy that can be absorbed is:

$$E = \Delta T \cdot C_{th} = (165\,°\mathrm{C} - 45\,°\mathrm{C}) \cdot 1.45\,\mathrm{J\,K^{-1}} = 174\,\mathrm{J}$$

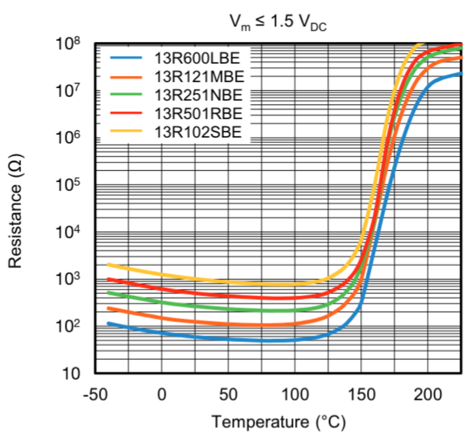

Figure 2: Resistance vs. Temperature for PTCEL13 (typical)

Next, we calculate the energy dissipated in one discharge:

$$E = \frac{1}{2} \cdot C \cdot V^2 \tag{4}$$
$$= \frac{1}{2} \cdot 200\,\mu\text{F} \cdot (403.2\,\text{V})^2 = 16.26\,\text{J} \tag{5}$$

Therefore, the number of discharges possible before the discharge time exceeds 5 s is:

$$\frac{174\,\text{J}}{16.26\,\text{J}} \approx 10.7 \Rightarrow \textbf{10 discharges}$$

Permanent TS Voltage

We can find the equilibrium temperature by finding the temperature at which the heat loss is equal to the power emitted. To find that, we first convert the graph provided in the datasheet (fig. 2) to a Look Up Table (LUT), a python script is then created with the two function listed below to find the equilibrium point.
(DF: dissipation factor. For the used PTC: 19.5 mW/K).

$$(T_{eq} - T_{amb}) \cdot DF = P_{dissipated} \tag{6}$$
$$V_{TS}^2 / R_{PTC} = P_{created} \tag{7}$$

After the execution of the script, we can see that the power dissipation at equilibrium is about 1.84 W. The equilibrium temperature and the corresponding resistance calculated is then 139 °C and 88.5 kΩ accordingly. We can see that this is smaller then the maximum temperature rated at 165 °C.

To find whether the MOSFET STB10LN80K5 can survive the permanent TS voltage, we first have to calculate the current going through it.

$$I = V/R = 403.2\,\text{V}/88.5\,\text{k}\Omega \tag{8}$$
$$= 4.56\,\text{mA} \tag{9}$$

Since the MOSFET drain current I_D is rated for 8A, it will work under permanent TS voltage. [3]

Python script

```
import pandas as pd
import numpy as np

data = pd.read_csv('PTC_LUT.csv', comment='#')

temp = data['Temperature']
resist = data['Resistance']

volt = 403.2
dissiFac = 0.0195
ambTemp = 45

powerLoss = -1
powerCreate = 0
i = 0

# Find the zone where the lines intersect
while powerCreate>powerLoss:
    powerLoss = (temp[i] - ambTemp)*dissiFac
    powerCreate = np.square(volt) / resist[i]
    i = i + 1

# put zone into points to solve for intersection
p1 = [i-1, (temp[i-1] - ambTemp)*dissiFac]
p2 = [i, (temp[i] - ambTemp)*dissiFac]

p3 = [i-1, np.square(volt) / resist[i-1]]
p4 = [i, np.square(volt) / resist[i]]

# Line 1 dy, dx and determinant
a11 = (p1[1] - p2[1])
a12 = (p2[0] - p1[0])
b1 = (p1[0]*p2[1] - p2[0]*p1[1])

# Line 2 dy, dx and determinant
a21 = (p3[1] - p4[1])
a22 = (p4[0] - p3[0])
b2 = (p3[0]*p4[1] - p4[0]*p3[1])

# Construction of the linear system
# coefficient matrix
A = np.array([[a11, a12],
              [a21, a22]])

# right hand side vector
b = -np.array([b1,
               b2])
# solve
try:
    intersection_point = np.linalg.solve(A,b)
    print('Intersection point detected at:', intersection_point)
    print('Result:')
    print('Temperature: ', intersection_point[1]/dissiFac+ambTemp)
    print('Resistance: ', np.square(volt)/intersection_point[1])
except np.linalg.LinAlgError:
    print('No single intersection point detected')
```

Reference

[1] *emDrive HXXX Datasheet*. emsisosi.sharepoint.com

[2] *Vishay PTCEL13R251NxE Datasheet*. www.vishay.com, 09.2024

[3] *ST STB10LN80K5 Datasheet*. www.st.com, 02.2016