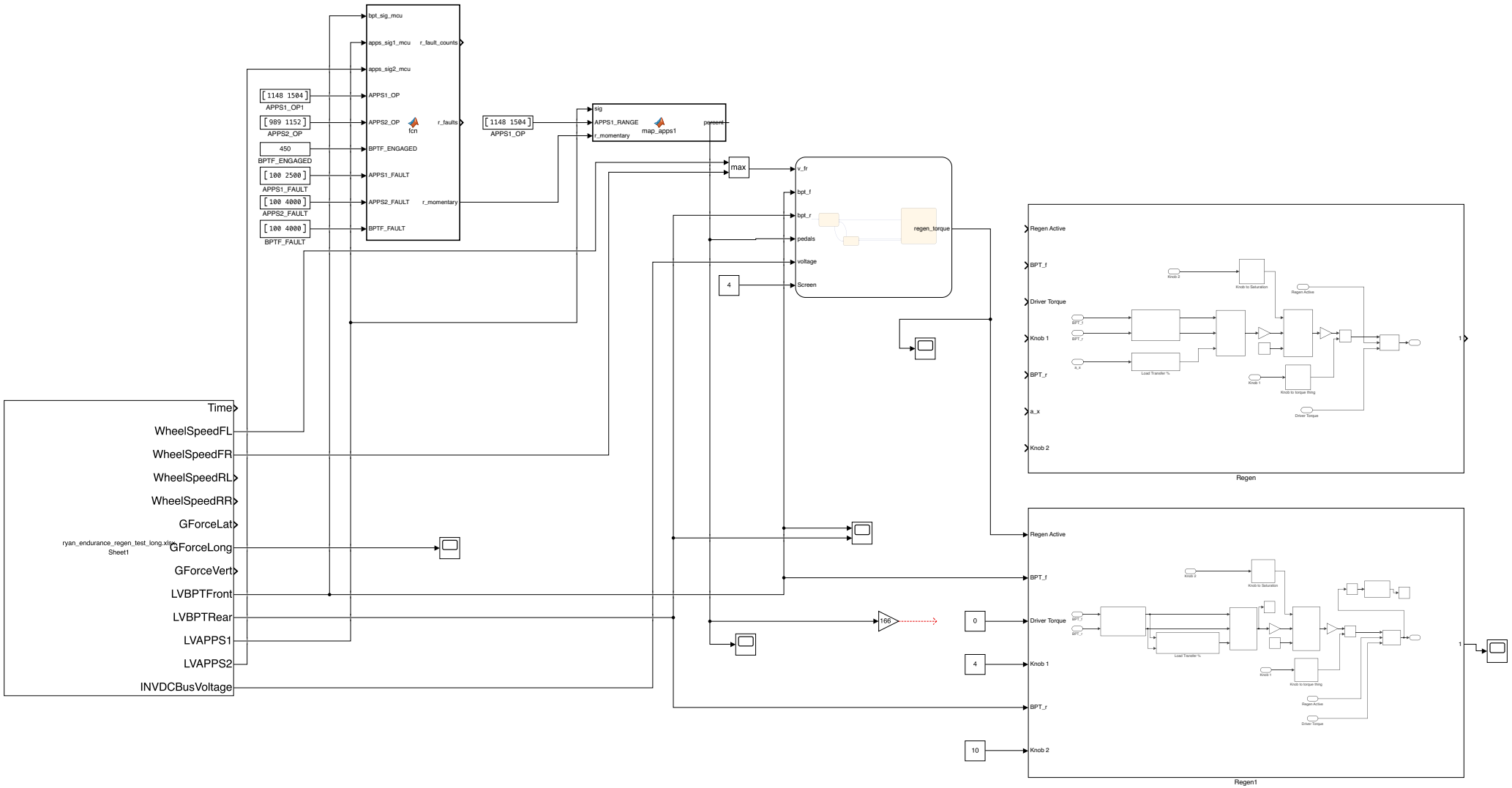
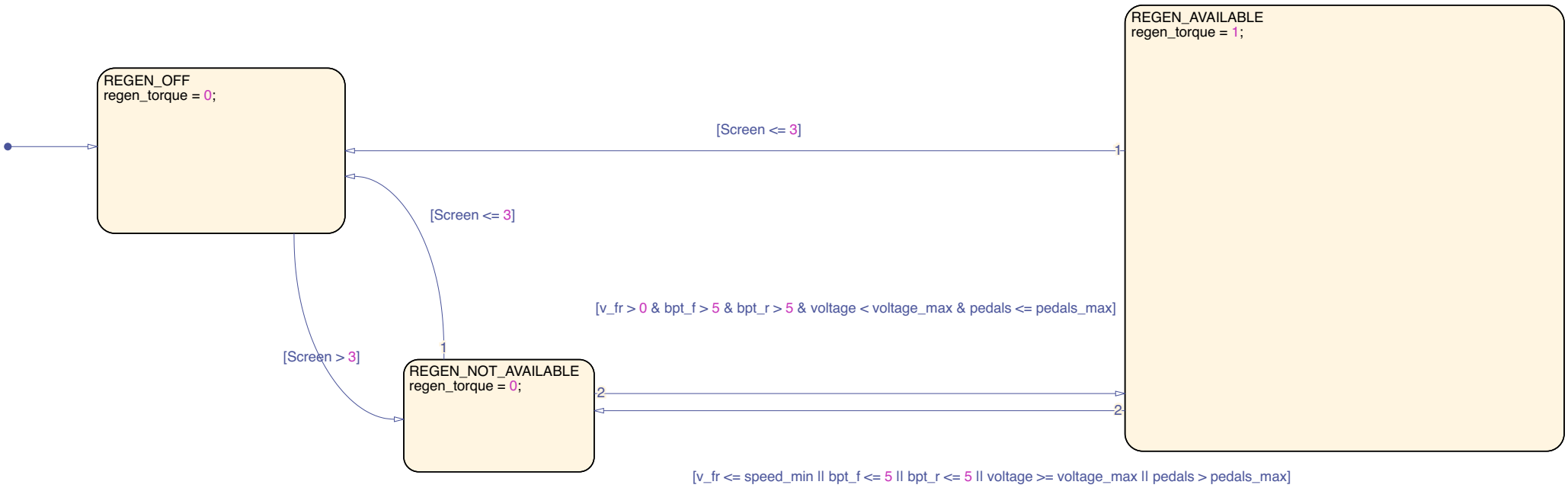
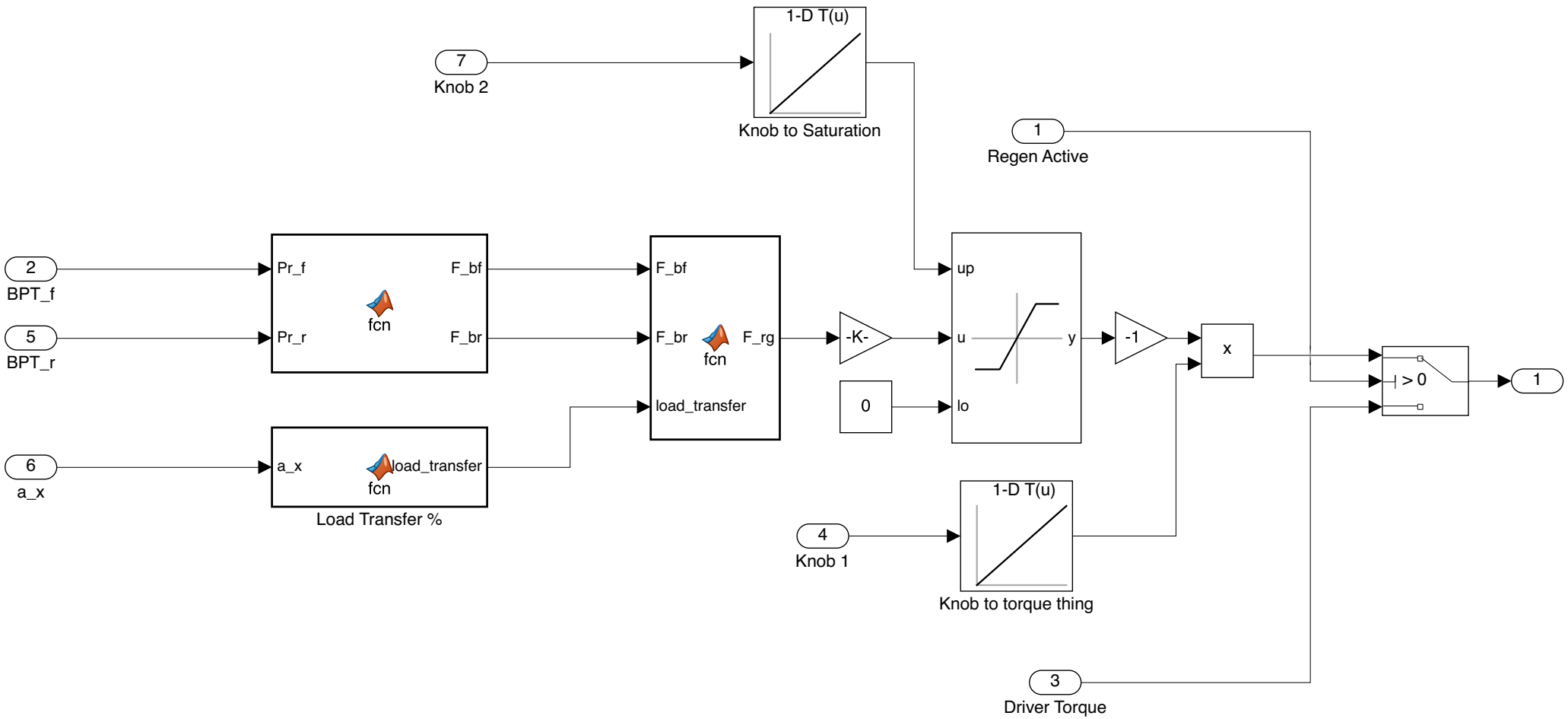

bpt_sig_mcu
apps_sig1_mcu
r_fault_counts
apps_sig2_mcu
APPS1_OP
APPS2_OP
fcn
r_faults
BPTF_ENGAGED
APPS1_FAULT
APPS2_FAULT
r_momentary
BPTF_FAULT
[1148 1504]
APPS1_OP1
[989 1152]
APPS2_OP
450
BPTF_ENGAGED
[100 2500]
APPS1_FAULT
[100 4000]
APPS2_FAULT
[100 4000]
BPTF_FAULT
[1148 1504]
APPS1_OP
sig
APPS1_RANGE
r_momentary
map_apps1
percent
max
v_fr
bpt_f
bpt_r
pedals
voltage
Screen
regen_torque
4
Time
WheelSpeedFL
WheelSpeedFR
WheelSpeedRL
WheelSpeedRR
GForceLat
ryan_endurance_regen_test_long.xlsx
Sheet1
GForceLong
GForceVert
LVBPTFront
LVBPTRear
LVAPPS1
LVAPPS2
INVDCBusVoltage
Regen Active
BPT_f
Driver Torque
Knob 1
BPT_r
a_x
Knob 2
Regen
166
0
4
10
Regen1


REGEN_OFF
regen_torque = 0;
REGEN_AVAILABLE
regen_torque = 1;
REGEN_NOT_AVAILABLE
regen_torque = 0;
[Screen <= 3]
[Screen <= 3]
[Screen > 3]
[v_fr > 0 & bpt_f > 5 & bpt_r > 5 & voltage < voltage_max & pedals <= pedals_max]
[v_fr <= speed_min || bpt_f <= 5 || bpt_r <= 5 || voltage >= voltage_max || pedals > pedals_max]
1
1
1
2
2

```
function percent = map_apps1(sig, APPS1_RANGE, r_momentary)

DZ_BOT = .05; %rosnel changed to .05 was .1
DZ_TOP = .05;

APPS_BOTTOM = APPS1_RANGE(1) + DZ_BOT * (APPS1_RANGE(2) - APPS1_RANGE(1)) ;
APPS_TOP    = APPS1_RANGE(2) - DZ_TOP * (APPS1_RANGE(2) - APPS1_RANGE(1)) ;

if (sig < APPS_BOTTOM) || r_momentary(1) || r_momentary(2)
    percent = 0;
elseif sig > APPS_TOP
    percent = 1;
else
    percent = (double(sig) - APPS_BOTTOM) / (APPS_TOP - APPS_BOTTOM);
end
```

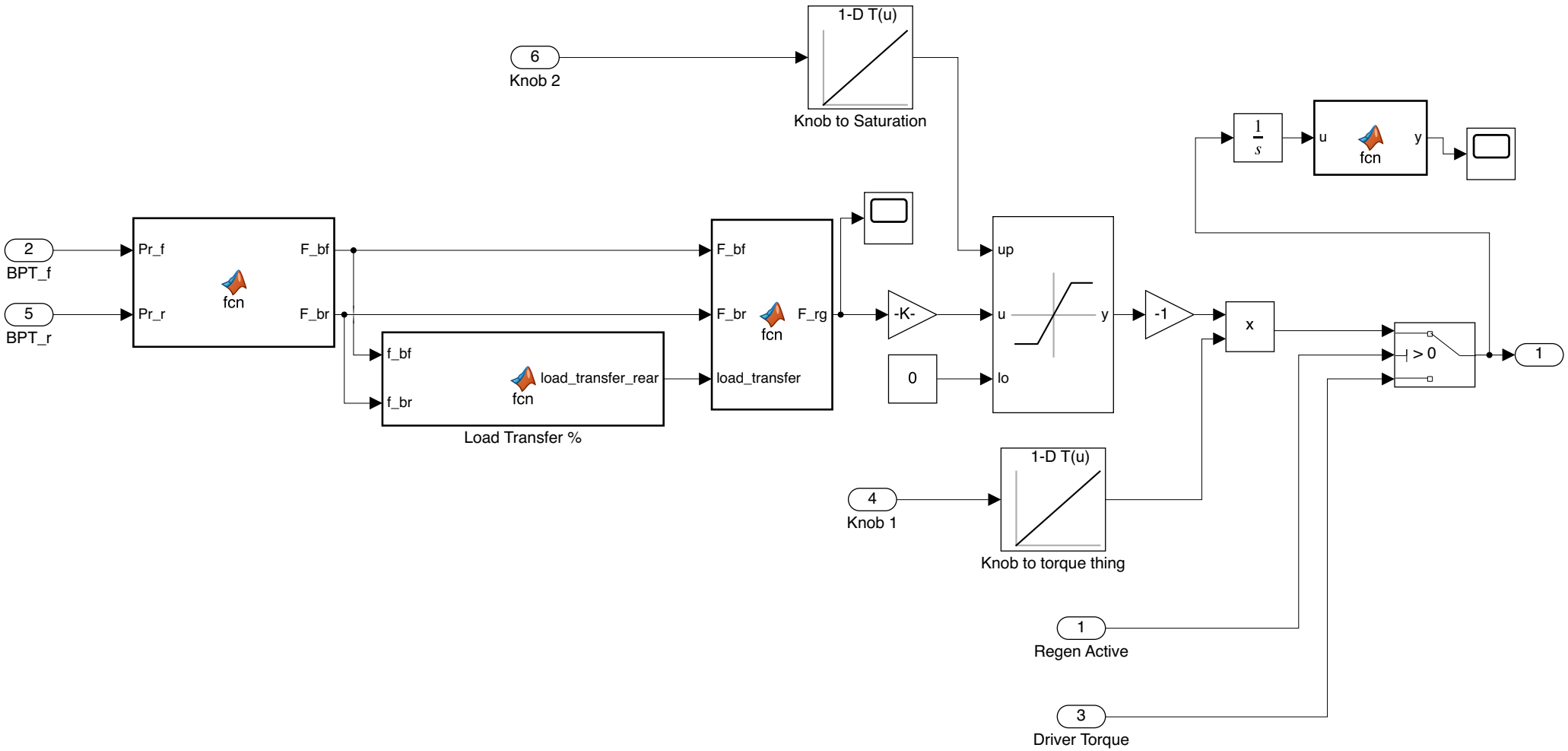
```
function [r_fault_counts, r_faults, r_momentary] = fcn(bpt_sig_mcu, apps_sig1_mcu, apps_sig2_mcu, APPS1_OP, APPS2_OP, BPTF_ENGAGED
%#codegen
%To map voltages / integers:
%Pre step down voltages (from sensors, range 0 to 5 V) * 2/3 -> Post step down voltages (0 to 3.3 V, used by mcu)
%Post step down voltages * (4096 / 3.3) -> uint16 (integer) value of variables
%Note: Some values are tweaked after testing to handle noise. 455 -> 435, 0.1 -> 0.13, etc. All values can and should be optimized

%Current operating range (MCU): 0.32 V to 2.67 V
%Brakes considered pressed: 0.40 V
%typical MCU noise: +- 0.03V

persistent fault_counts;
persistent faults;

FAULT_LIMITS = [10, 10, 10, 10, 10, 10, 10, 1];

% initialize persistents
if isempty(fault_counts)
    fault_counts = zeros(1,8);
end

if isempty(faults)
    faults = [false, false, false, false, false, false, false, false];
end

apps1_percent_pressed = (double(apps_sig1_mcu) - APPS1_OP(1))/(APPS1_OP(2) - APPS1_OP(1)); % current travel divided by range of tr
apps2_percent_pressed = (double(apps_sig2_mcu) - APPS2_OP(1))/(APPS2_OP(2) - APPS2_OP(1));

% Clear fault case 8 (two pedals) when apps goes below 5% travel
if (apps1_percent_pressed < 0.05)
    faults(8) = 0;
    fault_counts(8) = 0;
end

% Cases 1-6: APPS or BPT out of operating range (sensor failure)
out_of_bounds = [apps_sig1_mcu < APPS1_FAULT(1), apps_sig1_mcu > APPS1_FAULT(2), ...
                 apps_sig2_mcu < APPS2_FAULT(1), apps_sig2_mcu > APPS2_FAULT(2), ...
                 bpt_sig_mcu   < BPTF_FAULT(1),  bpt_sig_mcu   > BPTF_FAULT(2)];
% Case 7: Both APPS sensors differ by more than 10% for more than 100ms.
apps_disagree = abs(apps1_percent_pressed - apps2_percent_pressed) > 0.1;
% Case 8: If APPS is more than 25% pressed and brakes are pressed.
two_pedal = (apps1_percent_pressed >= 0.25) && (bpt_sig_mcu >= BPTF_ENGAGED);

conditions = [out_of_bounds, apps_disagree, two_pedal];

r_momentary = conditions;

conditions(8) = conditions(8) || faults(8); % Case 8 is sticky

% if condition is met but haven't yet triggered fault, increment fault_count.
% if condition is met but already in fault, leave fault_count the same (to
% prevent overflow)
% if condition is not met, zero the count
fault_counts = conditions .* (fault_counts + not(faults));
faults = fault_counts >= FAULT_LIMITS;

r_fault_counts = fault_counts;
r_faults = faults;
```


7
Knob 2
1-D T(u)
Knob to Saturation
1
Regen Active
2
BPT_f
5
BPT_r
Pr_f
Pr_r
fcn
F_bf
F_br
F_bf
F_br
fcn
F_rg
load_transfer
-K-
0
up
u
y
lo
-1
x
> 0
1
6
a_x
a_x
fcn
load_transfer
Load Transfer %
4
Knob 1
1-D T(u)
Knob to torque thing
3
Driver Torque

```
function load_transfer = fcn(a_x)
m = 250+68; %Mass with driver weight
a = 1.530/2; %Distance cg to front axle with load distribution of 0.5
h = 0.249; %Height of CG
l = 1.530; %Length of wheelbase
load_transfer = abs(a_x*h/l);
```

```
function [F_bf,F_br] = fcn(Pr_f,Pr_r)
mu = 0.410;
r_rotor = 76.96; %mm, effective rotor radius
A_r = 1816; %mm^2
psi_to_bar = 1/14.503;
%maybe ((1/BB_e)-1), MIT paper, now is just some variable 0 to 1 minus 1
F_br = (2*mu*r_rotor*A_r*Pr_r*psi_to_bar)/(10000);
F_bf = (2*mu*r_rotor*A_r*Pr_f*psi_to_bar)/(10000);
```

```
function F_rg = fcn(F_bf,F_br,load_transfer)
gain = (0.5 - load_transfer)/(0.5+load_transfer);
F_rg = F_bf*gain-F_br;
```


6
Knob 2
1-D T(u)
Knob to Saturation
1
s
u
fcn
y
2
BPT_f
5
BPT_r
Pr_f
Pr_r
fcn
F_bf
F_br
f_bf
f_br
load_transfer_rear
fcn
Load Transfer %
F_bf
F_br
fcn
F_rg
load_transfer
-K-
0
up
u
lo
y
-1
x
> 0
1
4
Knob 1
1-D T(u)
Knob to torque thing
1
Regen Active
3
Driver Torque

```
function load_transfer_rear = fcn(f_bf, f_br)
m = 250+68; %Mass with driver weight
a = 1.530/2; %Distance cg to front axle with load distribution of 0.5
h = 0.249; %Height of CG
l = 1.530; %Length of wheelbase
total_f = f_bf + f_br;
g = 9.81;
load_transfer_rear = abs((a*m*g)/l-(total_f*h)/l)/(m*g);
```

```
function [F_bf,F_br] = fcn(Pr_f,Pr_r)
mu = 0.410;
r_rotor = 76.96; %mm, effective rotor radius
A_r = 1816; %mm^2
psi_to_bar = 1/14.503;
%maybe ((1/BB_e)-1), MIT paper, now is just some variable 0 to 1 minus 1
F_br = (2*mu*r_rotor*A_r*Pr_r*psi_to_bar)/(10000);
F_bf = (2*mu*r_rotor*A_r*Pr_f*psi_to_bar)/(10000);
```

```
function F_rg = fcn(F_bf,F_br,load_transfer)
F_rg = (F_bf+F_br)*load_transfer-F_br;
```

```
function y = fcn(u)
persistent e;
if isempty(e)
    e = 0;
end
e = e + u;
y = e/(3600*1000);
```