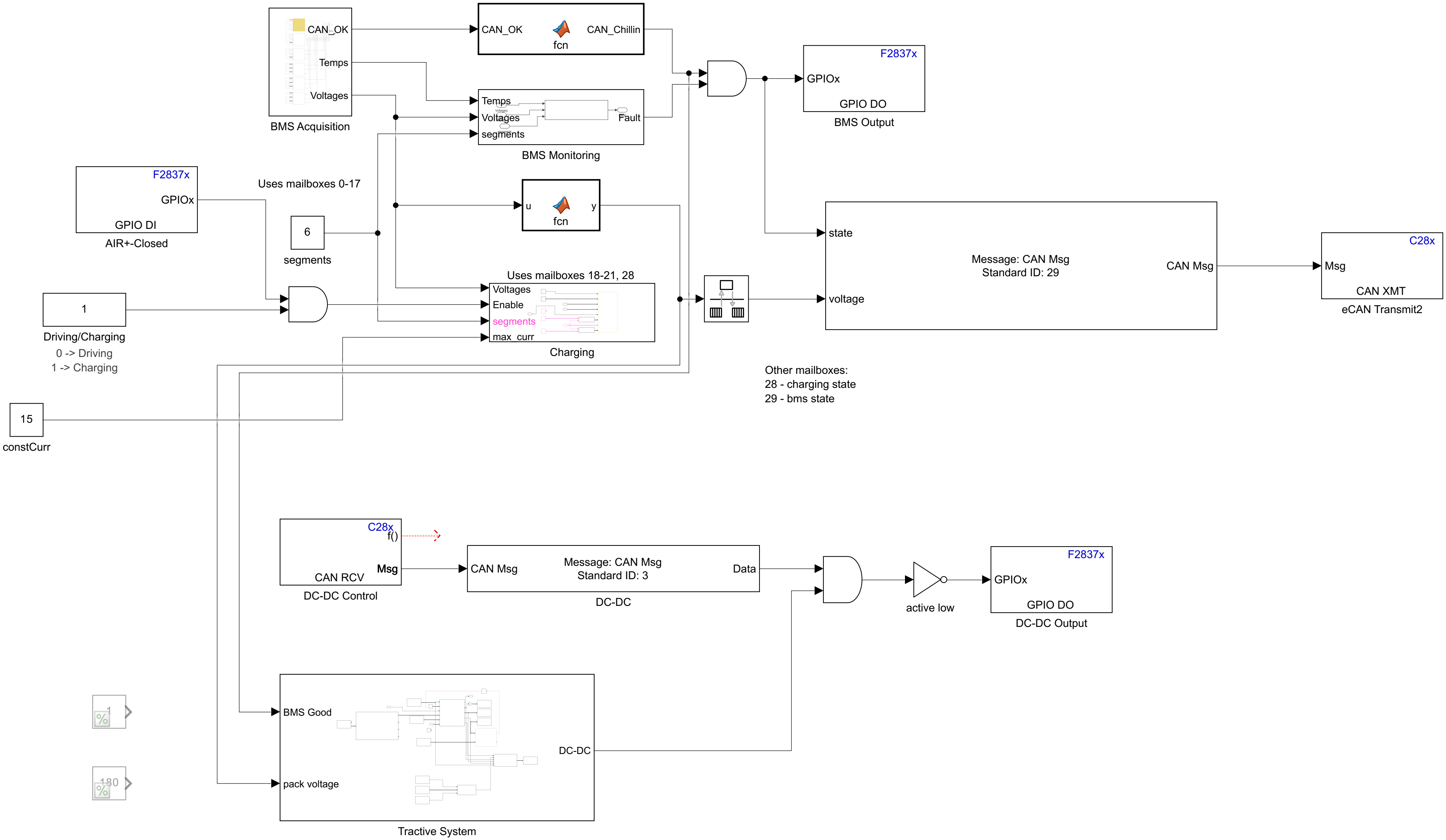
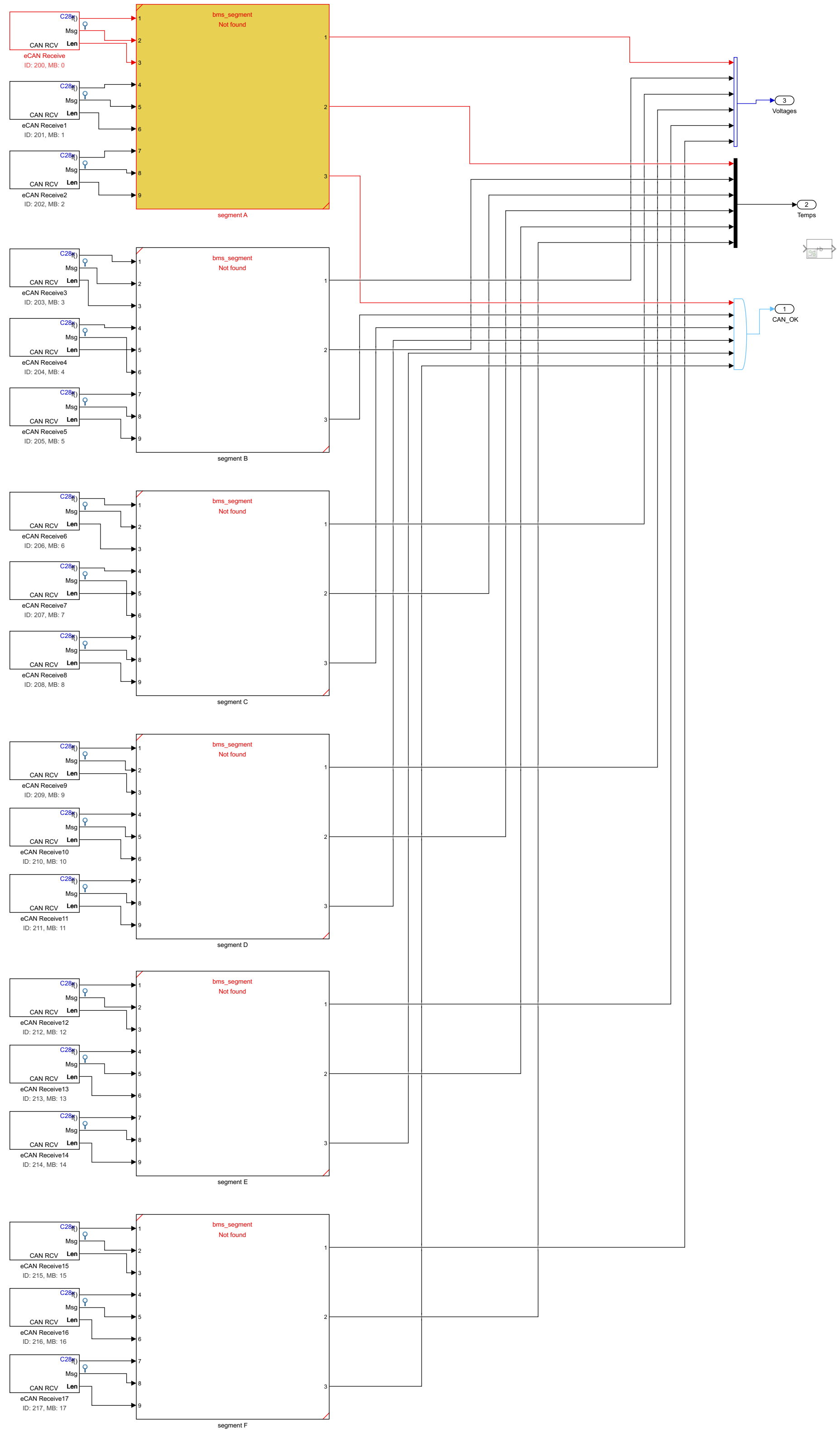
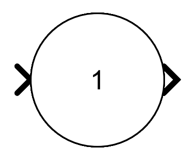
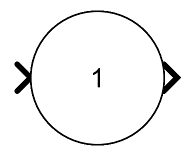
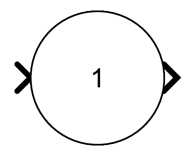

CAN_OK
Temps
Voltages
BMS Acquisition
CAN_OK
fcn
CAN_Chillin
Temps
Voltages
segments
Fault
BMS Monitoring
F2837x
GPIOx
GPIO DO
BMS Output
F2837x
GPIOx
GPIO DI
AIR+-Closed
Uses mailboxes 0-17
6
segments
u
fcn
y
state
Message: CAN Msg
Standard ID: 29
CAN Msg
voltage
C28x
Msg
CAN XMT
eCAN Transmit2
1
Driving/Charging
0 -> Driving
1 -> Charging
Uses mailboxes 18-21, 28
Voltages
Enable
segments
max_curr
Charging
Other mailboxes:
28 - charging state
29 - bms state
15
constCurr
C28x
f()
CAN RCV
Msg
DC-DC Control
CAN Msg
Message: CAN Msg
Standard ID: 3
Data
DC-DC
active low
F2837x
GPIOx
GPIO DO
DC-DC Output
BMS Good
pack voltage
DC-DC
Tractive System
1
180

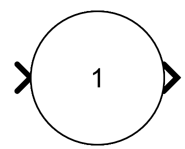


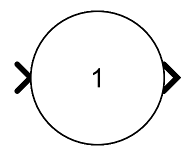
C28x
Msg
Len
CAN RCV
eCAN Receive
ID: 200, MB: 0
eCAN Receive1
ID: 201, MB: 1
eCAN Receive2
ID: 202, MB: 2
bms_segment
Not found
segment A
eCAN Receive3
ID: 203, MB: 3
eCAN Receive4
ID: 204, MB: 4
eCAN Receive5
ID: 205, MB: 5
segment B
eCAN Receive6
ID: 206, MB: 6
eCAN Receive7
ID: 207, MB: 7
eCAN Receive8
ID: 208, MB: 8
segment C
eCAN Receive9
ID: 209, MB: 9
eCAN Receive10
ID: 210, MB: 10
eCAN Receive11
ID: 211, MB: 11
segment D
eCAN Receive12
ID: 212, MB: 12
eCAN Receive13
ID: 213, MB: 13
eCAN Receive14
ID: 214, MB: 14
segment E
eCAN Receive15
ID: 215, MB: 15
eCAN Receive16
ID: 216, MB: 16
eCAN Receive17
ID: 217, MB: 17
segment F
3
Voltages
2
Temps
1
CAN_OK

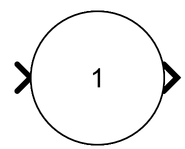

1

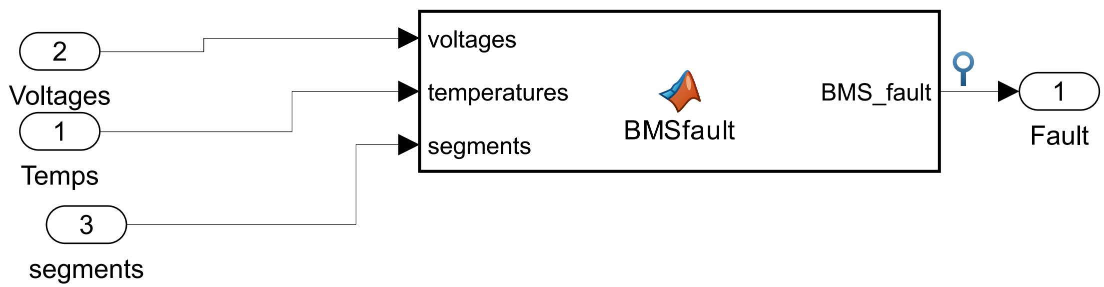

1


1

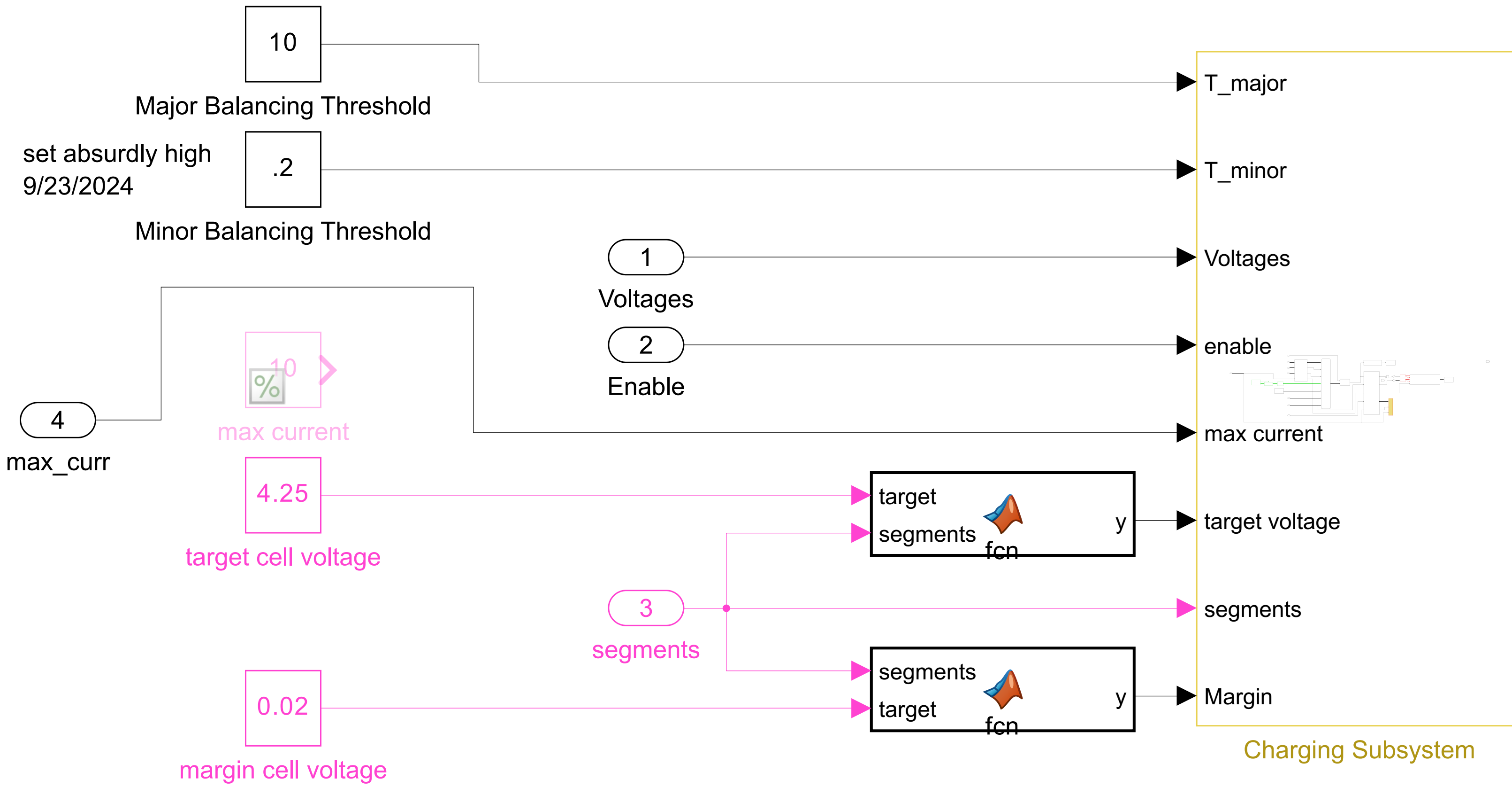

1

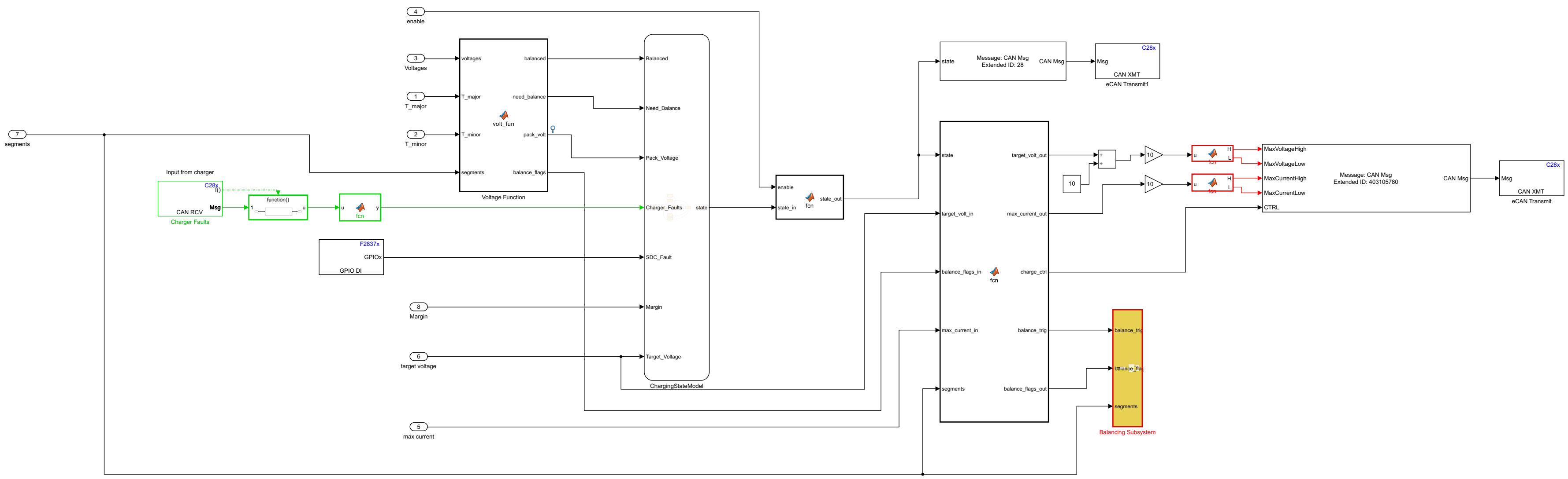

1


1

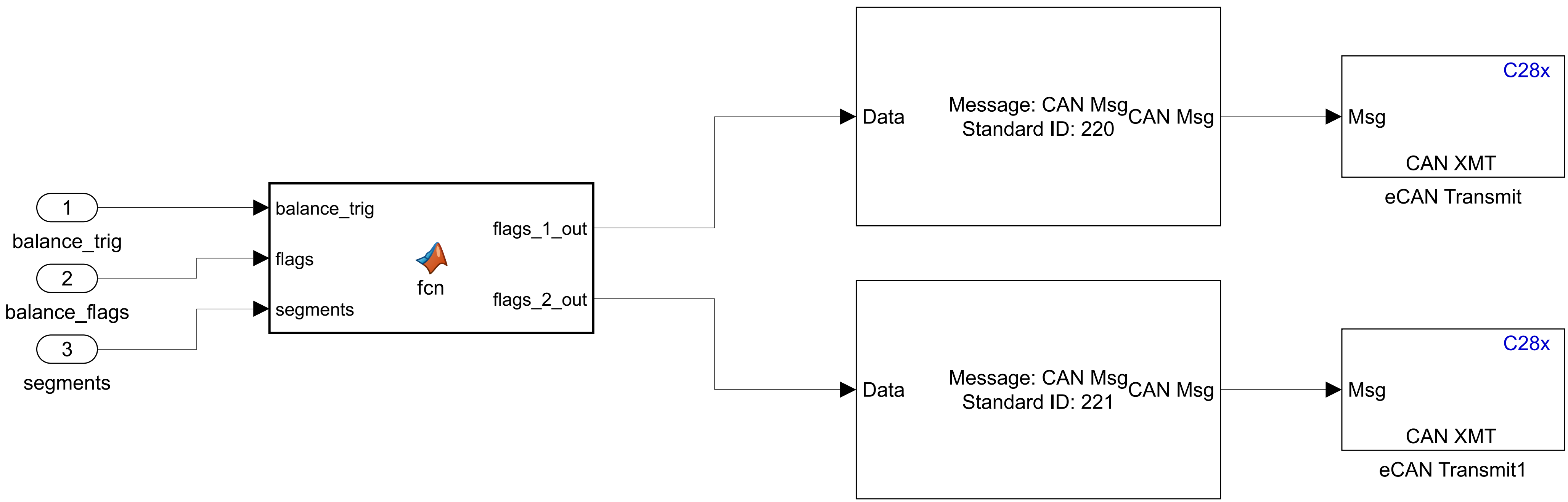

2
Voltages
1
Temps
3
segments
voltages
temperatures
segments
BMSfault
BMS_fault
1
Fault

```
 % checks if there are out of range temperatures or voltages for a segment's
% worth of battery modules
function BMS_fault = BMSfault(voltages, temperatures, segments)

% countVoltErrs will retain its value across multiple function calls
persistent countVoltErrs countTempErrs
if isempty(countVoltErrs)
    countVoltErrs = zeros(72,1);
    countTempErrs = zeros(72,1);
end

BMS_fault=true; % 1 means not faulted

voltErrThresh = 8; % the number of voltage faulse readings before an error is sent, was 3 changed to 8 by rosnel may 16
totalVoltErr = 0;
totalVoltErrThresh = 8; % the number of bad sensors accepted at once without fault, was 3 changed to 8 by rosnel may 16

% checks if there are multiple voltage errrors in a row
for i = 1:(segments*12)
    if(voltages(i)>4.25||voltages(i)<2.5)
        % voltage value transmitted on the CAN message is the actual voltages of the cells
        countVoltErrs(i) = countVoltErrs(i) + 1; % if there is an error, inc counter
    else
        countVoltErrs(i) = 0; % if no errors, reset the given counter
    end
    if(countVoltErrs(i) > voltErrThresh) % if exceeding threshold, trigger error
        totalVoltErr = totalVoltErr + 1;
        if(totalVoltErr > totalVoltErrThresh)
            BMS_fault=false;
        end
    end
end

tempErrThresh = 8; % the number of false readings issues before an error is sent, was 3 but changed to 8 rosnel
totalTempErr = 0;
totalTempErrThresh = 12*segments*0.1;

for i = 1:(segments*12)
    if(temperatures(i)>=60) % T is in Celcius. changed to 60C by rosnel, may 31
        countTempErrs(i) = countTempErrs(i) + 1; % if there is an error, inc counter
    else
        countTempErrs(i) = 0; % if no errors, reset the given counter
    end
    if(countTempErrs(i) > tempErrThresh) % if exceeding threshold, trigger error
        totalTempErr = totalTempErr + 1;
        if(totalTempErr > totalTempErrThresh)
            BMS_fault=false;
        end
    end
end

end
```


10
Major Balancing Threshold
set absurdly high
9/23/2024
.2
Minor Balancing Threshold
1
Voltages
2
Enable
4
max_curr
10
max current
4.25
target cell voltage
3
segments
0.02
margin cell voltage
target
segments
y
fcn
segments
target
y
fcn
T_major
T_minor
Voltages
enable
max current
target voltage
segments
Margin
Charging Subsystem

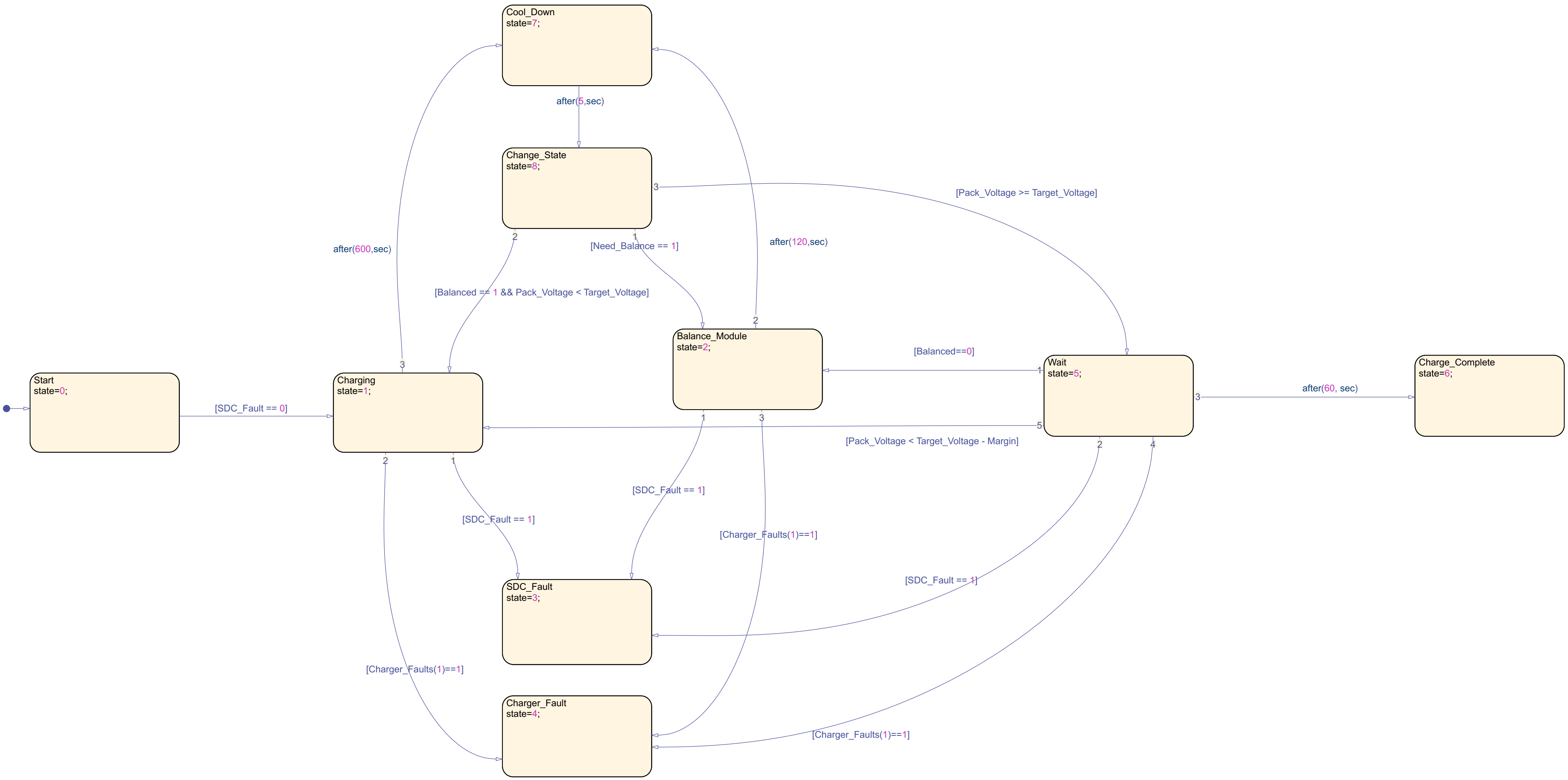


4
enable
3
Voltages
1
T_major
2
T_minor
7
segments
voltages
T_major
T_minor
segments
balanced
need_balance
pack_volt
balance_flags
volt_fun
Voltage Function
Input from charger
C28x
CAN RCV
Msg
Charger Faults
function()
fcn
F2837x
GPIOx
GPIO DI
8
Margin
6
target voltage
5
max current
Balanced
Need_Balance
Pack_Voltage
Charger_Faults
SDC_Fault
Margin
Target_Voltage
state
ChargingStateModel
enable
state_in
state_out
fcn
state
Message: CAN Msg
Extended ID: 28
CAN Msg
Msg
CAN XMT
eCAN Transmit1
state
target_volt_in
balance_flags_in
max_current_in
segments
target_volt_out
max_current_out
charge_ctrl
balance_trig
balance_flags_out
fcn
10
10
10
MaxVoltageHigh
MaxVoltageLow
MaxCurrentHigh
MaxCurrentLow
CTRL
Message: CAN Msg
Extended ID: 403105780
CAN Msg
Msg
CAN XMT
eCAN Transmit
balance_trig
segments
Balancing Subsystem

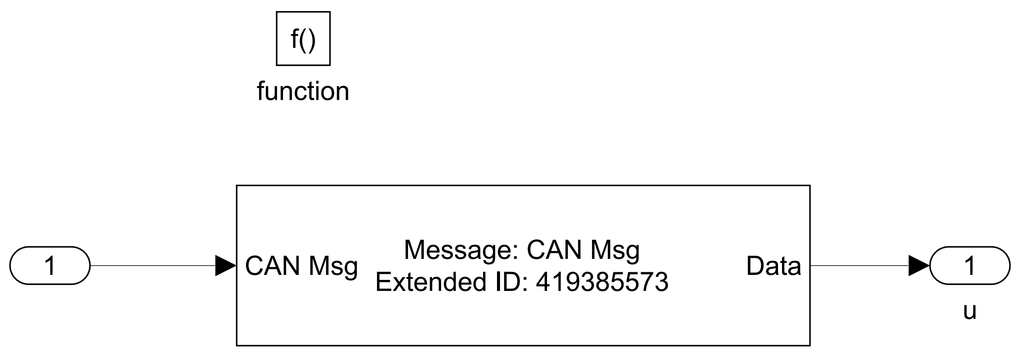


1
balance_trig
2
balance_flags
3
segments
balance_trig
flags
segments
fcn
flags_1_out
flags_2_out
Data
Message: CAN Msg
Standard ID: 220
CAN Msg
Data
Message: CAN Msg
Standard ID: 221
CAN Msg
C28x
Msg
CAN XMT
eCAN Transmit
C28x
Msg
CAN XMT
eCAN Transmit1

```
function [flags_1_out, flags_2_out] = fcn(balance_trig, flags, segments)
    flags_1 = zeros(64, 1, 'uint8');
    flags_2 = zeros(64, 1, 'uint8');

    if balance_trig == 1
        for segment = 0:3
            if segment < segments
                for cell = 1:12
                    flags_1(segment * 16 + cell) = flags(segment * 12 + cell);
                end
            end
        end
        for segment = 4:5
            if segment < segments
                for cell = 1:12
                    flags_2(segment * 16 + cell) = flags(segment * 12 + cell);
                end
            end
        end
    end

    flags_1_out = zeros(8, 1, 'uint8');
    flags_2_out = zeros(8, 1, 'uint8');
    for byte = 1:8
        for bit = 1:8
            flags_1_out(byte) = flags_1_out(byte) + flags_1((byte - 1) * 8 + bit) * 2^(8-bit);
            flags_2_out(byte) = flags_2_out(byte) + flags_2((byte - 1) * 8 + bit) * 2^(8-bit);
        end
    end
```



Cool_Down
state=7;
after(5,sec)
Change_State
state=8;
[Pack_Voltage >= Target_Voltage]
after(120,sec)
after(600,sec)
[Need_Balance == 1]
[Balanced == 1 && Pack_Voltage < Target_Voltage]
Balance_Module
state=2;
[Balanced==0]
Wait
state=5;
Charge_Complete
state=6;
after(60, sec)
Start
state=0;
[SDC_Fault == 0]
Charging
state=1;
[Pack_Voltage < Target_Voltage - Margin]
[SDC_Fault == 1]
[SDC_Fault == 1]
[Charger_Faults(1)==1]
[SDC_Fault == 1]
SDC_Fault
state=3;
[Charger_Faults(1)==1]
Charger_Fault
state=4;
[Charger_Faults(1)==1]



f()
function
1
CAN Msg
Message: CAN Msg
Extended ID: 419385573
Data
1
u

```
function y = fcn(u)
    if u(5)>0
        y = 1;
    else
        y = 0;
    end
end
```

```
function [target_volt_out, max_current_out, charge_ctrl, balance_trig, balance_flags_out] = fcn(state, target_volt_in, balance_fla
    % charge_ctrl 0 means charging, 1 means stop
    if state == 1
        target_volt_out = target_volt_in;
        max_current_out = max_current_in;
        charge_ctrl = 0;
    else
        target_volt_out = 0;
        max_current_out = 0;
        charge_ctrl = 1;
    end

    if state == 2
        balance_trig = 1;
    else
        balance_trig = 0;
    end

    persistent balance_flags
    if isempty(balance_flags) %initialize persistant variable
        balance_flags = zeros(72, 1);
    end
    balance_flags_out = zeros(72, 1);

    if state == 8 % change state
        for i = 1:72
            balance_flags(i) = balance_flags_in(i);
        end
    end

    for i = 1:72
        balance_flags_out(i) = balance_flags(i);
    end
```

```
function [H,L] = fcn(u)

    if u <= 255
        H = 0;
        L = u;
    else
        H = floor(u/256);
        L = mod(u, 256);

    end

end
```

```
function [H,L] = fcn(u)

if u <= 255
    H = 0;
    L = u;

else
    H = floor(u/256);
    L = mod(u, 256);

end


end
```

```
function state_out = fcn(enable, state_in)
    if enable == 0
        state_out = 6;
    else
        state_out = state_in;
    end
```

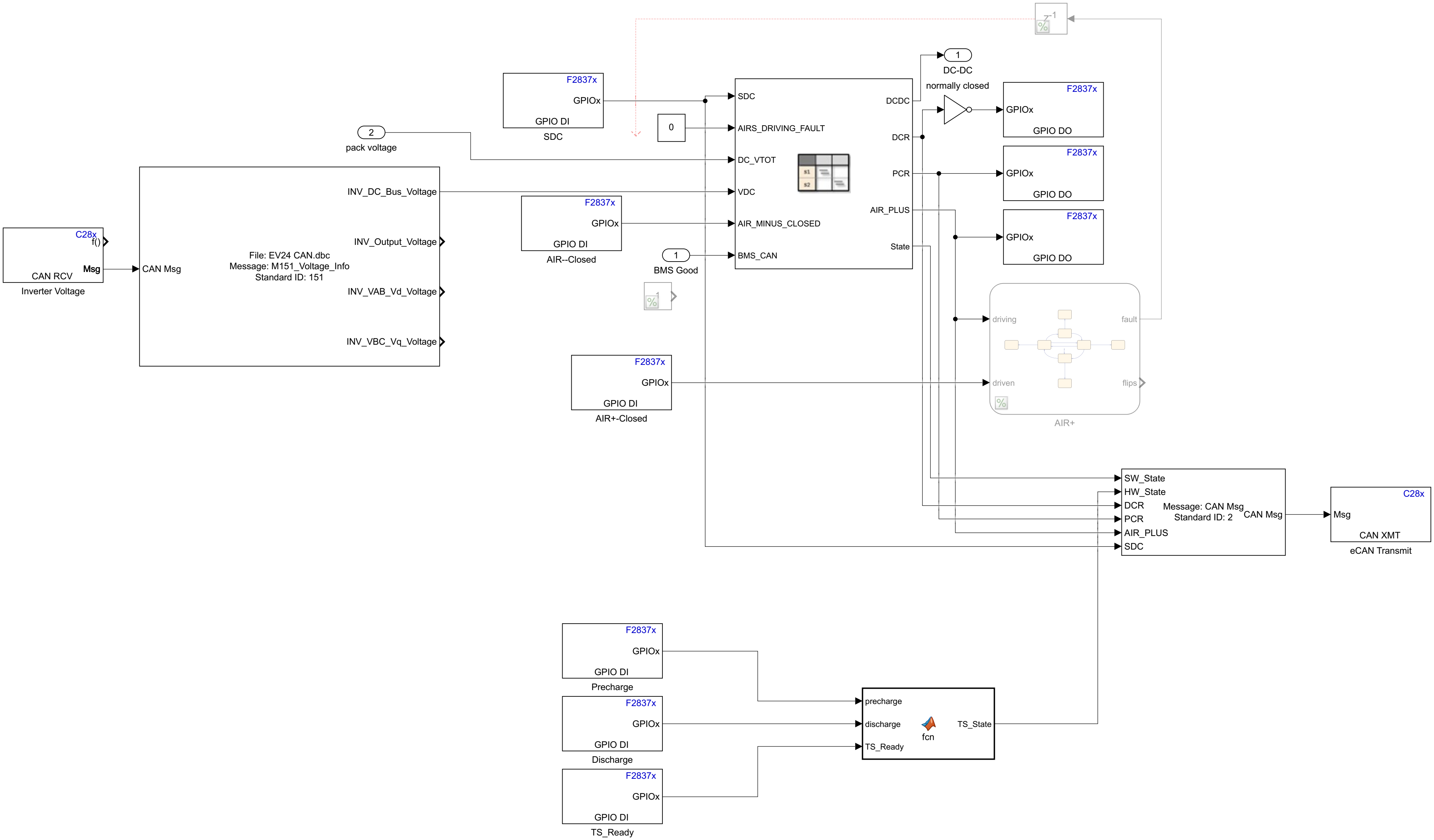
```
function [balanced, need_balance, pack_volt, balance_flags]  = volt_fun(voltages, T_major, T_minor, segments)

balanced = 1; % 1 means balanced, all of the voltages are within Tminor from min voltage
need_balance = 0; % need_balance=1 means any of the voltages is at least Tmajor above min voltage

minVoltage = min(voltages);
balance_flags = zeros(72, 1);

for i = 1:(segments*12)
    if(voltages(i) - minVoltage > T_major)
        need_balance = 1;
    end
    if(voltages(i) - minVoltage > T_minor)
        balanced = 0;
        balance_flags(i) = 1;
    end
end

% now calculate the voltage of the entire pack
% simply sum all individual voltages

pack_volt = sum(voltages);
% the summed voltage should be comparable to target voltage
```

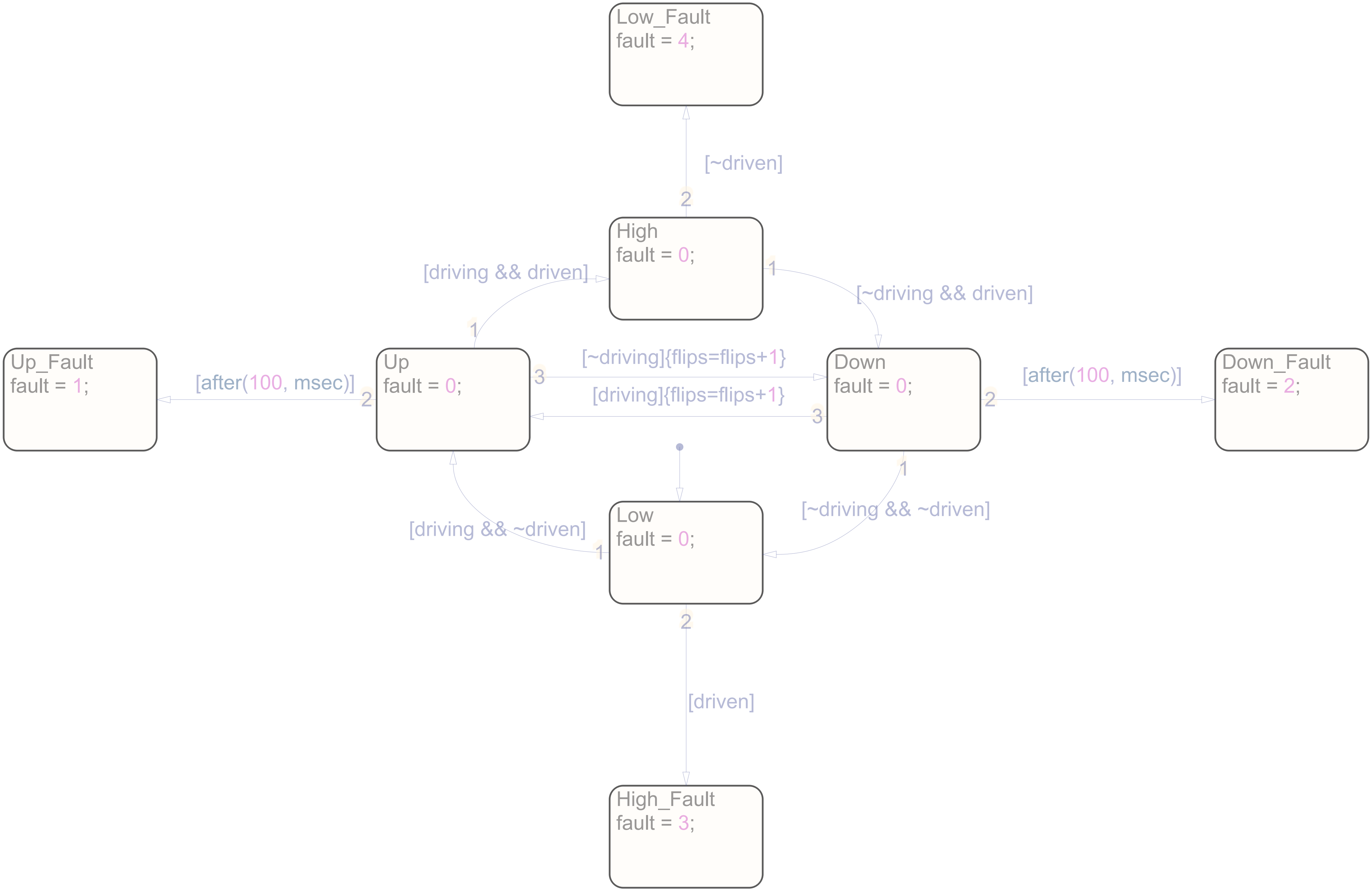
```
function y = fcn(target, segments)
    y = 12*segments*target;
```

```
function y = fcn(segments, target)
    y = 12*segments*target;
```

```
function y = fcn(u)
y = sum(u);
```

```
function CAN_Chillin = fcn(CAN_OK)

persistent countCANTimeout %CAN_SAD
if isempty(countCANTimeout) %initialize persistant variable
    countCANTimeout = 0;
end

CAN_Chillin = true; % 1 means not faulted

CANThresh = 8; % the number of CAN problems before an error is sent, was 2 changed to 8 by rosnel may 30

% checks if there are multiple CAN timeout errrors in a row
if(CAN_OK == 0)
    countCANTimeout = countCANTimeout + 1; % if there is an error, inc counter
else
    countCANTimeout = 0; % if no errors, reset the counter
end

if(countCANTimeout > CANThresh) % if exceeding threshold, trigger error
    CAN_Chillin=false;
end

end
```


F2837x
GPIOx
GPIO DI
SDC
2
pack voltage
0
1
DC-DC
normally closed
SDC
AIRS_DRIVING_FAULT
DC_VTOT
VDC
AIR_MINUS_CLOSED
BMS_CAN
DCDC
DCR
PCR
AIR_PLUS
State
GPIO DO
C28x
f()
Msg
CAN RCV
Inverter Voltage
CAN Msg
File: EV24 CAN.dbc
Message: M151_Voltage_Info
Standard ID: 151
INV_DC_Bus_Voltage
INV_Output_Voltage
INV_VAB_Vd_Voltage
INV_VBC_Vq_Voltage
AIR--Closed
BMS Good
AIR+-Closed
driving
fault
driven
flips
AIR+
SW_State
HW_State
AIR_PLUS
Message: CAN Msg
Standard ID: 2
CAN XMT
eCAN Transmit
Precharge
Discharge
TS_Ready
precharge
discharge
fcn
TS_State


Low_Fault
fault = 4;
[~driven]
2
High
fault = 0;
[driving && driven]
1
[~driving && driven]
1
Up_Fault
fault = 1;
[after(100, msec)]
2
Up
fault = 0;
3
[~driving]{flips=flips+1}
[driving]{flips=flips+1}
3
Down
fault = 0;
2
[after(100, msec)]
Down_Fault
fault = 2;
1
[~driving && ~driven]
[driving && ~driven]
1
Low
fault = 0;
2
[driven]
High_Fault
fault = 3;

```
function TS_State = fcn(precharge, discharge, TS_Ready)
%#codegen
a = precharge + discharge + TS_Ready;
if ((precharge && discharge) || (discharge && TS_Ready))
    TS_State = 0;
else
    if(precharge && TS_Ready)
        TS_State=2;
    elseif(discharge==1)
        TS_State=1;
    else
        TS_State=3;
    end
end
```

<table>
<tr><th colspan="7">State Transition Table</th></tr>
<tr><th colspan="7">Block: accumulator/Tractive System/State Transition Table</th></tr>
<tr><td rowspan="3">1</td><td rowspan="3"></td><td rowspan="3">Start_Discharge
State = 0;
DCR = 1;
PCR = 0;
AIR_PLUS = 0;
DCDC = 0;</td><td>[AIRS_DRIVING_FAULT]</td><td>[SDC && AIR_MINUS_CLOSED]</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Fault_Discharge</td><td>Precharge</td><td></td><td></td></tr>
<tr><td rowspan="3">2</td><td rowspan="3"></td><td rowspan="3">Discharge
State = 1;
DCR = 0;
PCR = 0;
AIR_PLUS = 0;
DCDC = 0;
on after(DEAD_MSEC, msec):
DCR = 1;</td><td>[AIRS_DRIVING_FAULT]</td><td>[flip_counter > FLIP_THRESHOLD]</td><td>[SDC && AIR_MINUS_CLOSED]</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Fault_Discharge</td><td>Fault_Discharge</td><td>Precharge</td><td></td></tr>
<tr><td rowspan="3">3</td><td rowspan="3"></td><td rowspan="3">Precharge
State = 2;
DCR = 0;
PCR = 0;
AIR_PLUS = 0;
on after(DEAD_MSEC, msec):
PCR = 1;</td><td>[AIRS_DRIVING_FAULT]</td><td>[~SDC]</td><td>after(PC_SEC, sec)[VDC > 0.95*DC_VTOT && BMS_CAN]</td><td>after(PC_SEC*32, sec)</td></tr>
<tr><td></td><td>{flip_counter = flip_counter + 1;}</td><td></td><td></td></tr>
<tr><td>Fault_Discharge</td><td>Discharge</td><td>TS_Ready</td><td>Fault_Discharge</td></tr>
<tr><td rowspan="3">4</td><td rowspan="3"></td><td rowspan="3">TS_Ready
State = 3;
DCR = 0;
PCR = 1;
AIR_PLUS = 1;
DCDC = 1;</td><td>[AIRS_DRIVING_FAULT]</td><td>[~SDC]</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Fault_Discharge</td><td>Discharge</td><td></td><td></td></tr>
<tr><td rowspan="3">5</td><td rowspan="3"></td><td rowspan="3">Fault_Discharge
State = -1;
DCR = 0;
PCR = 0;
AIR_PLUS = 0;
DCDC = 0;
on after(DEAD_MSEC, msec):
DCR = 1;</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>