

```
//University of Tulsa Caterpillar ECM Power Loss Testing
//24 Aug 2015
//Jeremy Daily and Amila Perera

//Test Constants
unsigned long testDuration = 25000;
unsigned long speedChangeStartTime = 10000;//Milliseconds after st

unsigned long powerLossStartTime = 11500;
unsigned long powerLossDuration = 100;

//coolant level power loss
unsigned long CLpowerLossStartTime = 11000;//ms
unsigned long CLpowerLossDuration = 14000;

unsigned long boostPressStartTime = 12000;
unsigned long boostPressDuration = 13000;

unsigned long appStartTime = 13000;
unsigned long appDuration = 12000;

unsigned int pulsesPerMile = 29541;// pulses per mile
int quickStopRate = 10;//Miles per hour per second
int VSStimeChangeStep = 48;//ms

//Vehicle Speed Sensor
int speedAmp = 50;//tenths of a mph
float speedFreq= 0.2;//Hz
int originalVSSspeed = 500;//tenths of a mph

//Accelerator Pedal Position
int originalPedalPos = 24;//%
int pedalAmp = 10;//%
float pedFreq = .4;

//Oil Pressure
int originalboostPress = 25;
```

```

int oilAmp = 10;
float oilFreq = .6;

//Changing or calculated variables
int VSSfreq;
int VSSfreqChangeStep;//adjust for quickstop
int VSSspeed;
int pedValue;
byte appByte;
int appValue;
int boostPressValue;

//board setup constants
const int pinButton1 = 8;// Push Button Input Pin
const int powerRelayPin = 2;// ECM Power3
const int faultRelayPin = 3;// Sensor Power
const int LEDpin = 13;
const int appPin = 6;//Accelerator Pedal Position
const int boostPressPin = 9;
const int simulatedVSSpin = 5;

//These variables get reset
unsigned long lastSensorChangeTime = 0;
unsigned long VSSfreqChangeTime = 0;
unsigned long startTime = 0;
unsigned long CLstartTime = 0;
unsigned long currentMillis=0;
unsigned long displayTime = 0;

boolean testStarted =false;
boolean powerOff =false;
boolean faultRelayPinOn =false;
boolean sensorState =false;
boolean stateButton1 =false;

void setup() {

```

```
pinMode(appPin,OUTPUT);
pinMode(pinButton1, INPUT);
pinMode(powerRelayPin, OUTPUT);
pinMode(faultRelayPin, OUTPUT);
pinMode(simulatedVSSpin, OUTPUT);
pinMode(boostPressPin,OUTPUT);
pinMode(LEDpin, OUTPUT);
```

```
VSSfreqChangeStep = 30;//adjust for quickstop
VSSspeed = originalVSSspeed;
setVSSspeed(VSSspeed);
```

```
setboostPress(originalboostPress);
setAPP(originalPedalPos);
```

```
Serial.begin(115200);
Serial.println("University of Tulsa");
Serial.println("Engine Control Module (ECM) Power Loss Testing");
Serial.println("Test parameters are as follows:");
Serial.print("Speed Change Start Time:\t");
Serial.print(speedChangeStartTime);
Serial.println("\tmilliseconds");
Serial.print("Power Loss Start Time:\t");
Serial.print(powerLossStartTime);
Serial.println("\tmilliseconds");
Serial.print("Test Duration:\t");
Serial.print(testDuration);
Serial.println("\tmilliseconds");
Serial.print("boostPressStartTime:\t");
Serial.print(boostPressStartTime);
Serial.println("\tmilliseconds");
Serial.print("boostPressDuration:\t");
Serial.print(boostPressDuration);
Serial.println("\tmilliseconds");
Serial.print("StartingVSSfrequency:\t");
Serial.print(originalVSSspeed);
```

```

Serial.println("\tmph");
Serial.print("VSStimeChangeStep:\t");
Serial.print(VSStimeChangeStep);
Serial.println("\tmilliseconds");
Serial.print("VSSfreqChangeStep:\t");
Serial.print(VSSfreqChangeStep);
Serial.println("\tmph");

// Starting conditions
currentMillis =millis();

powerOff =false;
faultRelayPinOn =false;

digitalWrite(powerRelayPin, powerOff);
digitalWrite(faultRelayPin, faultRelayPinOn);
// VSSfreq = map(VSSspeed,0,100,0,820.58);
// tone(simulatedVSSpin,VSSfreq);
// analogWrite(boostPressPin,originalboostPress);
// analogWrite(appPin,originalPedalPos);

delay(5000);
Serial.println("Time [ms]\tVSSfreq. [Hz]\tSensorFreq\tpowerRelayP

}

int getSpeed() {
    int speedVal = originalVSSspeed + speedAmp*sin(2*speedFreq*3.141!
    return speedVal;
}

void setVSSspeed(int VSSspd) {
    int VSSfreq =map(VSSspd,0,1000,0,820.58);
    if (VSSfreq <= 31){//Constrain for negative VSSfrequencies.
        pinMode(simulatedVSSpin,INPUT);

```

```

        digitalWrite(simulatedVSSpin,0);
    }
    else tone(simulatedVSSpin,VSSfreq);

}

int getboostPress() {
    return originalboostPress + oilAmp*sin(2*oilFreq*3.1415927*(curr
}

void setboostPress(int boostPress) {
    int boostPressVal =map(boostPress,23,50,100,200);
    analogWrite(boostPressPin,boostPressVal);
    //analogWrite(boostPressPin, boostPress);
}

int getAPP() {
    return originalPedalPos + pedalAmp*sin(2*pedFreq*3.1415927*(curr
}

void setAPP(int appPercent) {
    int appVal =map(appPercent,10,72,70,150);
    analogWrite(appPin,appVal);
    //analogWrite(appPin,appPercent);
}

void loop() {
    currentMillis =millis();
    digitalWrite(LEDpin,testStarted);
    stateButton1 =digitalRead(pinButton1);

    //Start the test
    if(stateButton1 == HIGH && testStarted ==false) {
        testStarted =true;
        startTime = currentMillis;
        pinMode(simulatedVSSpin,OUTPUT);
        VSSspeed = originalVSSspeed;
    }
}

```

```

setVSSspeed(VSSspeed);

    powerOff =false;
    digitalWrite(powerRelayPin, powerOff);
}

if (testStarted) {

///Speed      Change
    if (currentMillis - startTime >= speedChangeStartTime) {
        if (currentMillis - VSSfreqChangeTime >= VSStimeChangeStep
            VSSfreqChangeTime = currentMillis;
            VSSspeed -= VSSfreqChangeStep;//change this number
            if (VSSspeed <= 0) VSSspeed =0;
            setVSSspeed(VSSspeed);
            Serial.println(VSSspeed);
        }
    }
    else{
        VSSspeed = getSpeed();
        setVSSspeed(VSSspeed);
    }

/*+++++ Start ECM Power Loss ++++++

    if (currentMillis - startTime >= powerLossStartTime && !powerOf
        powerOff =true;
        //Serial.println("Power Off.");
        digitalWrite(powerRelayPin, powerOff);
    }
    else if (currentMillis - startTime >= powerLossStartTime + powe
        powerOff =false; //Turn power back on
        digitalWrite(powerRelayPin, powerOff);
    }
    else {
        powerOff =false;
    }
}

```

```

/*+++++ Start Coolent Power Loss +++++

    if (currentMillis - CLstartTime >= CLpowerLossStartTime && !fau.
        faultRelayPinOn =true;
        //Serial.println("Coolant Off.");
        digitalWrite(faultRelayPin, faultRelayPinOn);
    }
    else if (currentMillis - CLstartTime >= CLpowerLossStartTime + (
        faultRelayPinOn =false; //Turn power back on
        digitalWrite(faultRelayPin, faultRelayPinOn);
    }
    else {
        faultRelayPinOn =false;
    }

/*+++++ Start Oil Pressure +++++

    if (currentMillis - startTime >= boostPressStartTime && boostPr
        pinMode(boostPressPin,INPUT);
        digitalWrite(boostPressPin,0);
        //boostPressValue = 0;
        //setboostPress(boostPressValue);
    }
    else if (currentMillis - startTime >= boostPressStartTime + boo
        pinMode(boostPressPin,OUTPUT);
        boostPressValue = getboostPress();
        setboostPress(boostPressValue);
    }
    else {
        boostPressValue = getboostPress();
        setboostPress(boostPressValue);
    }

/*+++++ End Oil Pressure +++++

/*+++++ Start Accelerator Pedal Pos
    if (currentMillis - startTime >= appStartTime && appValue > 0){
        pinMode(appPin,INPUT);

```

```

        digitalWrite(appPin, 0);
    }
    else if (currentMillis - startTime >= appStartTime + appDuratio
        pinMode(appPin, OUTPUT);
        appValue = getAPP();
        setAPP(appValue);
    }
    else {
        appValue = getAPP();
        setAPP(appValue);
    }

/*+++++ End Accelerator Pedal Posit

}

if (currentMillis - startTime > testDuration){
    testStarted=false;
    startTime = currentMillis;

    //pinMode(simulatedVSSpin, OUTPUT);
    Serial.println("Test Stopped.");

}

//    if (currentMillis - displayTime >= 100){
//        displayTime = currentMillis;
//        Serial.print(currentMillis-startTime);
//        Serial.print("\t");
//        Serial.print(VSSfreq);
//        Serial.print("\t");
//        Serial.print(appValue);
//        Serial.print("\t");
//        Serial.print(boostPressValue);
//        Serial.print("\t");
//        Serial.print(!powerOff);
//        Serial.print("\t");
//        Serial.println(!faultRelayPinOn);
//    }

```


