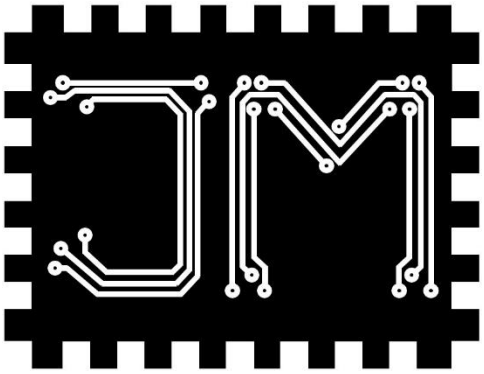

Jeff’s Laboratory

# NM09 – Servo Electro-Mechanical Actuator Databook

| Comments | Revision | Date | Author |
|---|---|---|---|
| Initial Release | A | January 7, 2025 | J. Mays |
| Update model wrapper in Simulink | B | February 26, 2025 | J. Mays |

## 1. References

1. https://www.servocity.com/hs-65mg-servo/

## 2. Purpose

This document describes the HS-65MG servo electro-mechanical actuator (servos for short) that is used to actuate the TVC and aft fins within the New Estes model rocket. First, there will be discussion of the servo hardware, and then the monte-carlo simulation model will be presented.

## 3. Hardware

The HiTec HS-65MG Electro-Mechanical Servo actuator (servo) is a micro class servo used for various hobbyist and robotic applications. This servo will be used to actuate the effectors on the New Mays rocket.

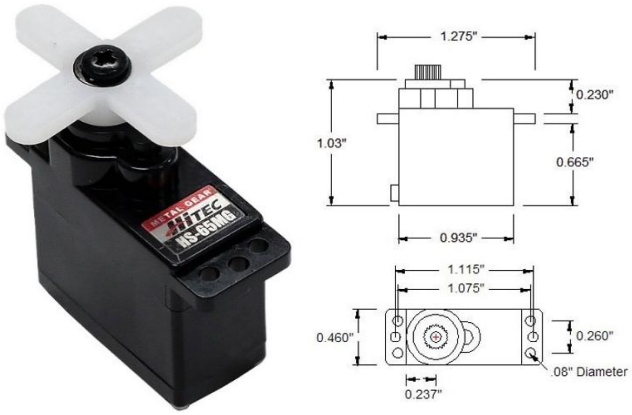


**Figure 1: HS-65MG Servo [1]**

The table below indicates some specs about this servo [1]. These specs were used to develop the model in Matlab / Simulink for monte-carlo simulations.

| Spec | Value |
|---|---|
| Voltage Range | 4.8V – 6.0V |
| No-Load Speed (4.8V) | 0.14sec/60° |
| No-Load Speed (6.0V) | 0.11sec/60° |
| Stall Torque (4.8V) | 25 oz-in (1.8 kg.cm) |
| Stall Torque (6.0V) | 31 oz-in (2.2 kg.cm) |
| Max PWM Signal Range | 610-2360µsec |
| Travel per µs (Stock) | .108°/µsec |
| Max Rotation (Stock) | 189° |
| Pulse Amplitude | 3-5V |
| Direction w/ Increasing PWM Signal | Clockwise (opposite our convention) |
| Deadband Width | 4µs |
| Feedback Style | 5KΩ Potentiometer |
| Weight | 0.39oz (11.2g) |
| Update Freq. | 50 Hz |

The HS-65MG servo is a closed system, which takes a pulse width modulated (PWM) signal as an input and turns it into a setpoint for a potentiometer to track. A typical internal control system for one of these servos appears in the following figure.

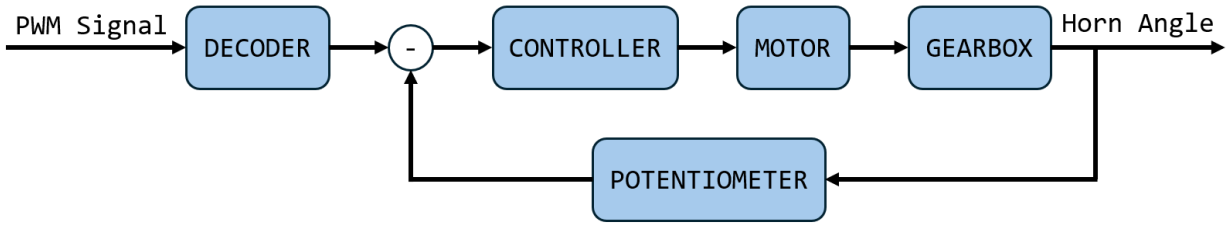


**Figure 2: Servo Internal Control System**

On the physical system, there are 3 wires that source the servo with everything it needs to operate. Other than the power and ground wires, the third wire provides the internal decoder with the analog PWM signal. For our application, this PWM signal is generated from the main processor on the main flight control board. By providing the correct PWM signal, we can expect the servo's horn to move to a pre-determined position.

The HS-65MG servo has a PWM range of 610 to 2360 µsec, with a center µsec of 1485. We can set this as the nominal PWM command such that we center the servo control horn within each effector in the vehicle. We can also use this PWM to correspond to 0 degrees in our future model.

## 4. Modeling & Simulation

The modeling of one of these servos is no trivial feat. Each servo from different manufacturers is different, and the system is difficult to characterize without extensive testing. All we can really do is check the available datasheets from the servo manufacturer and hope it is relatively correct. Some slow-motion videos and small tests were performed, but ultimately it is very difficult to correctly model the expected behavior of these servos without a massive test campaign.

Ultimately, I decided to model the servos in Simulink with the datasheet I had in an attempt to be "close enough." Later efforts could be made to attempt and model the elements in Figure 2, but ultimately it would be an exercise of randomly updating parts of the model to try and match the datasheet. Figure 3 illustrates the high-level library block of the servo model, which will be referenced many times in the future fully integrated simulation.

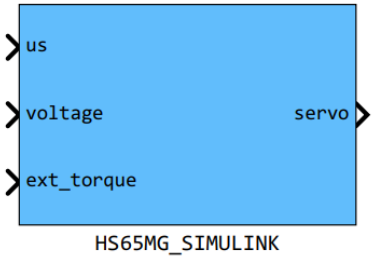


**Figure 3: HS-65MG Simulink Library Block**

The above library block contains a parameterization file that is run for each instance of the servo library block. This not only sets constant parameters for the simulation, but also updates tunable parameters for the monte-carlo simulation such that we disperse the performance of the servo to ensure system robustness. The initialization code for the model is shown below.

```
function [servo, servo_tunable] = loadHS65MGServoParameters(mc, servo_num)
% Load Servo parameters
%
% Author: Jeff Mays

% Voltages
servo.min_voltage = 4.8 * C_V;
servo.max_voltage = 6.0 * C_V;

% Minimum operating voltage
servo.minimum_operating_voltage = 4.8 * C_V;

% Min/Max us
servo.min_us = 610;  % Min microsecond accepted value
servo.max_us = 2360; % Max microsecond accepted value

% Nominal us
servo.nominal_us = 1485;

% Servo computational delay
servo.computational_delay = 1/50; % sec

% rad of rotation per microsecond
servo.rot_per_us = 0.108 * C_DEG; % 189.0 deg / (2360-610 us) = 0.108 deg/us

% us deadband
servo.rot_cmd_deadband = 4; % Deadband of servo controller

% Servo dynamics
wn = 20 * C_HZ;
zeta = sqrt(2)/2;
servo.dynamics.num = [0 0 wn^2];
servo.dynamics.den = [1 2*zeta*wn wn^2];

% Rate and accel filters (used to get rid of numeric issues. Breaks down
% the meaning of the signal, but this will be a known deficiency in the
% model...)
w = 5 * C_HZ;
z = sqrt(2)/2;
servo.filter.num = [0 0 w^2];
servo.filter.den = [1 2*z*w w^2];

% Servo angle limits
servo_tunable.min_angle = mc.disperse(['min_angle' num2str(servo_num)], ...
                                      'Dispersion', 'Normal', ...
                                      'Nominal',    -189.0/2 * C_DEG, ...
                                      'StdDev',     1 * C_DEG);
servo_tunable.max_angle = mc.disperse(['max_angle' num2str(servo_num)], ...
                                      'Dispersion', 'Normal', ...
                                      'Nominal',    +189.0/2 * C_DEG, ...
                                      'StdDev',     1 * C_DEG);

% Stall torques
servo_tunable.stall_torque_5V = mc.disperse(['stall_torque_5V' num2str(servo_num)], ...
                                            'Dispersion', 'Normal', ...
                                            'Nominal',    0.1765 * C_N*C_M, ...
                                            'StdDev',     0.01 * C_N*C_M);
servo_tunable.stall_torque_6V = mc.disperse(['stall_torque_6V' num2str(servo_num)], ...
                                            'Dispersion', 'Normal', ...
                                            'Nominal',    0.2189 * C_N*C_M, ...
                                            'StdDev',     0.01 * C_N*C_M);

% Rotation speed of unloaded servo
servo_tunable.no_load_rot_speed_5V = mc.disperse(['no_load_rot_speed_5V' num2str(servo_num)], ...
                                                 'Dispersion', 'Normal', ...
                                                 'Nominal',    60.0/0.14 * C_DEG/C_SEC, ... % 60 degrees in 0.14 sec
                                                 'StdDev',     15.0 * C_DEG/C_SEC);
servo_tunable.no_load_rot_speed_6V = mc.disperse(['no_load_rot_speed_6V' num2str(servo_num)], ...
                                                 'Dispersion', 'Normal', ...
                                                 'Nominal',    60.0/0.11 * C_DEG/C_SEC, ... % 60 degrees in 0.11 sec
                                                 'StdDev',     15.0 * C_DEG/C_SEC);

servo_tunable.angle_bias = mc.disperse(['angle_bias' num2str(servo_num)], ...
                                       'Dispersion', 'Normal', ...
                                       'Nominal',    0.0 * C_DEG, ...
                                       'StdDev',     0.33 * C_DEG); % This is bias in our tare accuracy

end
```

Opening the library block reveals the following Simulink code, shown in Figure 4 which uses the parameterized variables in the initialization script.

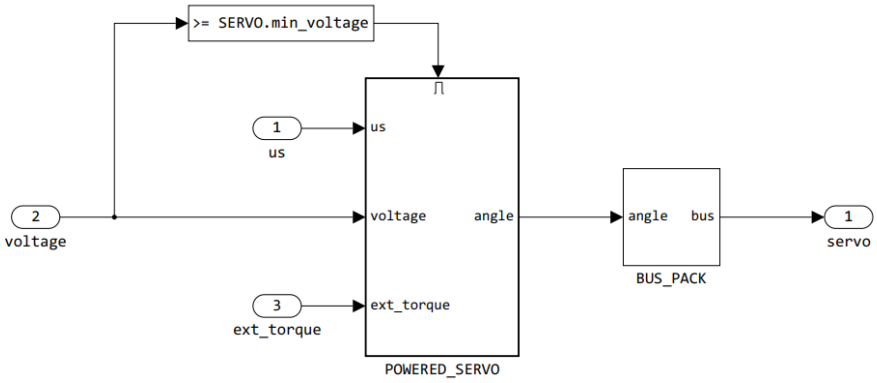


**Figure 4: HS-65MG Simulink Library Block Contents**

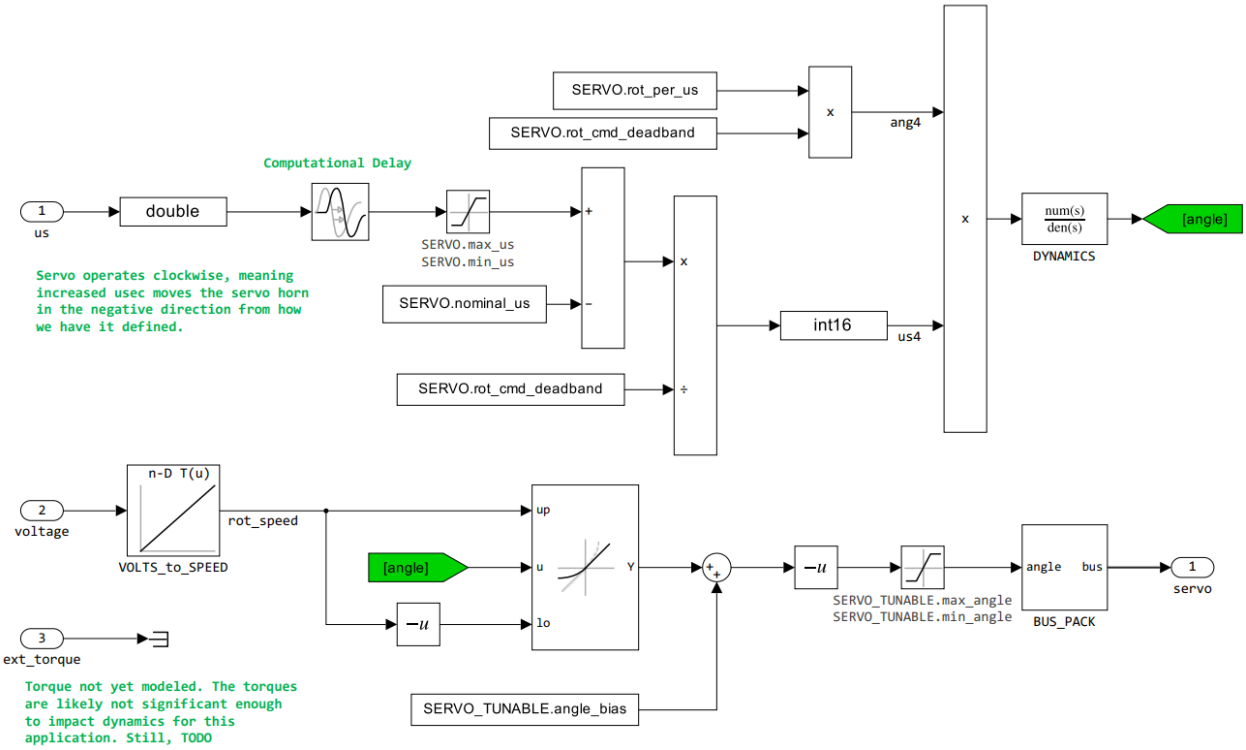


**Figure 5: HS-65MG Simulink Library Block/POWERED_SERVO**

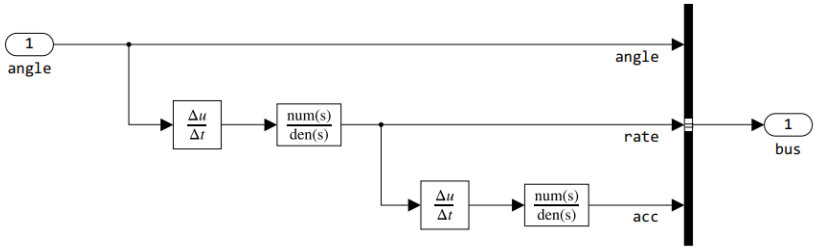


**Figure 6: BUS_PACK**

### 4.1. Model Validation

Before using the model explicitly for the full simulation, we must first develop a test environment to prove to ourselves that the model works as intended. The model validation harness is shown in Figure 7.

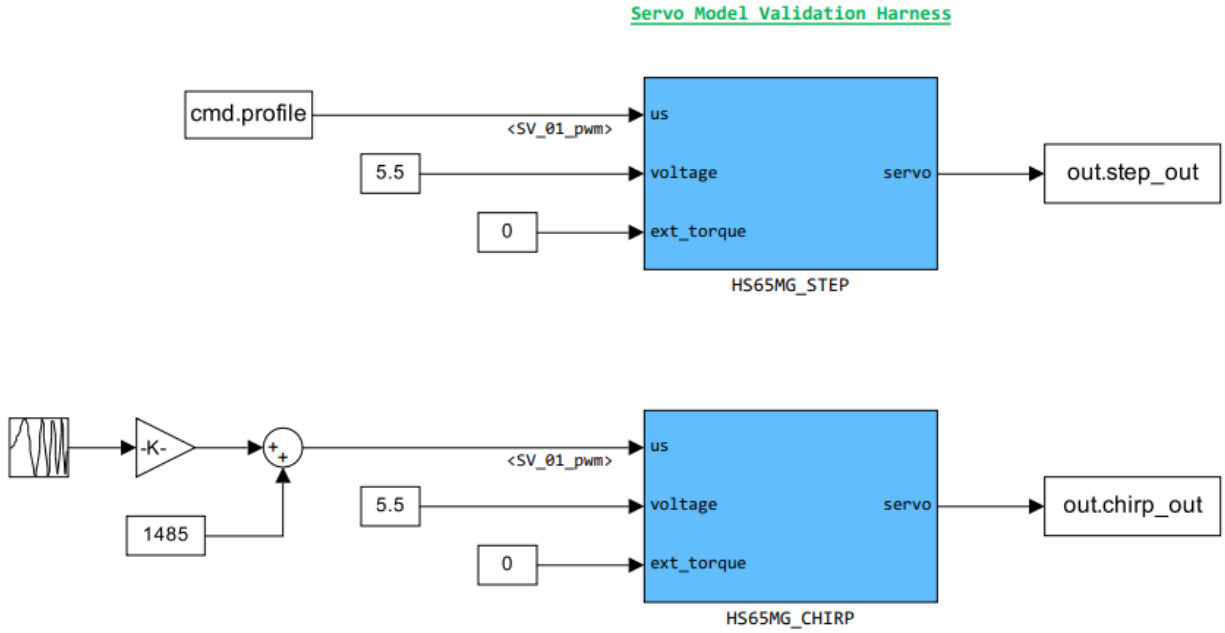


**Figure 7: Validation Harness**

Using the monte-carlo tooling, I performed 301 simulations of the above harness and generated step and chirp profiles to prove to myself that the behavior and tunable monte-carlo parameters were all behaving as intended. The following two plots show evidence that the model works as intended. Additional validation tests could be generated in the future to further validate the model.

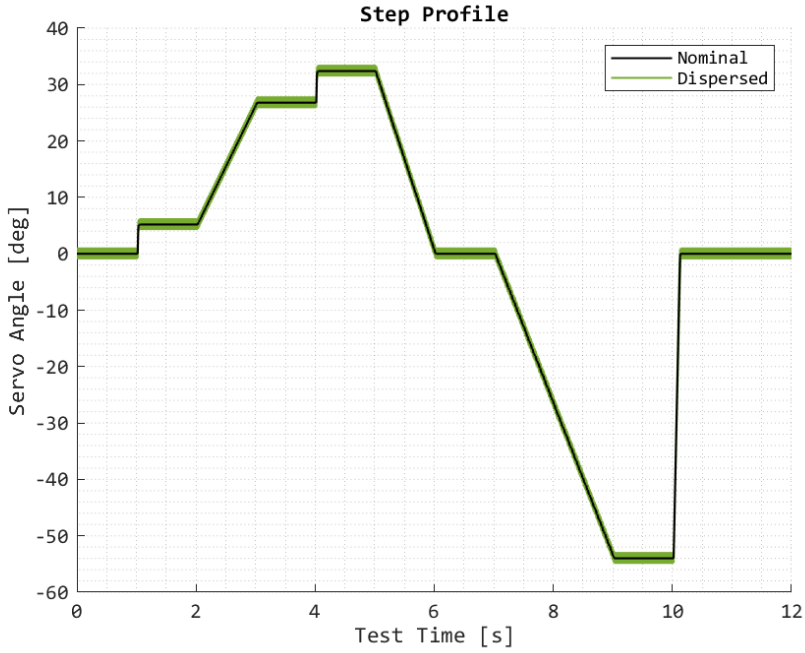


**Figure 8: Step Profile**

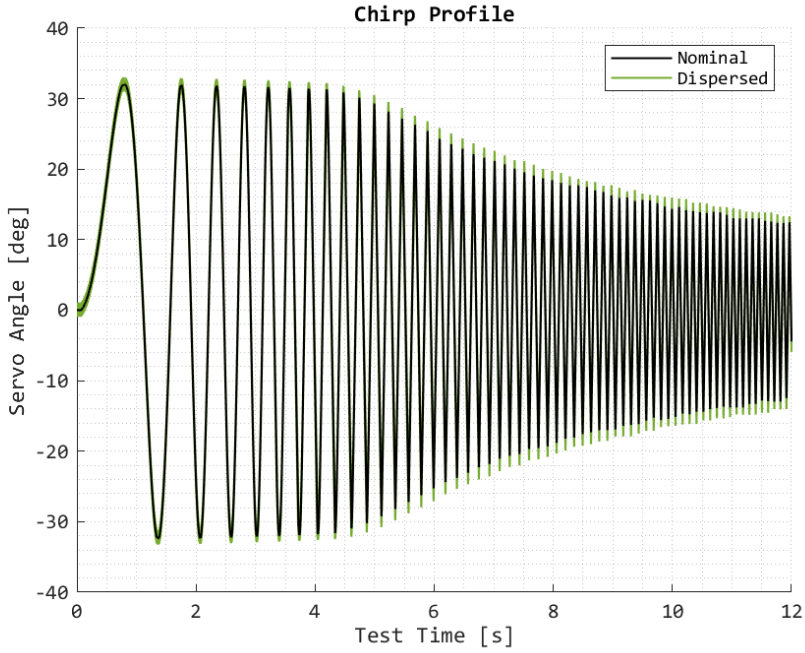


**Figure 9: Chirp Profile**