# Leg Firmware Registers/Parameters

Registers/Parameters can be read/written whenever.

Servo\_a\_goal – goal for servo a

Servo\_b\_goal – goal for servo b

Fsr\_min – min value for the fsr

Fsr\_max – max value for the fsr

Fsr\_poll\_speed – speed in which to check for fsr values

fsr\_active – Which fsrs used. An array that will be bitmasked to determine which fsrs will be used.

z\_goal – the z position in which the “z\_go\_to” command attempts to go to

z\_touchdown\_enabled – sets the flag to continue going down pass the z\_goal value until fsr\_max is reached or greater

z\_count\_max – sets the absolute max z value, stops motor after max is reached

z\_count\_stabilization\_min – minimum value that z can have when stabilization is enabled

z\_count\_stabilization\_max – maximum value that z can have when stabilization is enabled

rot\_debounce\_time – sets the debounce time for rotation count

motor\_speed – sets the motor speed for used in operations

**Read ONLY**

Servo\_a - current value of servo a

Servo\_b – current value of servo b

Analog0 – value of analog0

Analog1 – value of analog1

Analog2 – value of analog2

z\_calibrated – holds the flags that keeps track if the leg was zero, thus calibrated. z related functions are disabled until z is calibrated using z\_zero.

z – current value of z position

z\_stabilization\_enabled – holds if stabilization is active or not

leg\_status\_check – register that holds if there is a pending leg interrupt, returns no interrupt(0) or the type(non-zero). Clear interrupt with clear\_pending\_interrupt.

leg\_status – register that holds the current status of the leg

vref – register that holds the vref value

# Leg Firmware Executable Commands/Actions

Commands/Actions will start a command/action on the leg firmware

z\_zero – finds the zero point of z

z\_go\_to – runs the motor to reach the z\_goal position

z\_stop – stops the motor from spinning. This turns of z\_stabilization also

z\_ stabilization\_start – starts the stabilization algorithm

servos\_detach – detaches the signal for servos. Use bit masking to determine which servos to detac h.

servos\_go\_to – runs the servos to reach the set servo\_a\_goal and servo\_b\_goal values. Use bit masking to determine which servos to go to their values.

stop – stops all currently running commands/actions and turns off stabilization, motors, servo movements, etc.

set\_vref – set the voltage reference for analog reads. Default: 2.5V

clear\_pending\_interrupt – clears pending interrupt, this sets the leg to “READY” mode.

update\_analog – this updates the analog read registers. Use bit masking to determine which ones to update. Perform a register read after to read the analog value.

# Program Outline

General Leg Functions

Loop that always keeps track of z value using interrupts for rising and falling.

Stabilization loop

(?)Servo incremental move

Control motor class

Control servo class

Read fsr / averaging class

I2C Commands

* Write Register
* Read Register
* Execute Command

I2C Packet Formats

General write/read to registers. Unsure on how to do it. Should it be:

Write to Device: [(0)I2CAddress\_Write] [CC\_Write\_To\_Register][Register][Value][Checksum]

Read from Device: [(0)I2CAddress\_Read] [(1)OK?] [Checksum(0…1)]

[Read][Register][Value]

Return [Value][Checksum]