ZR User API

This is a quick guide to the functions used to control a SPHERES satellite in Zero Robotics. These functions do not change from game to game. All of them except DEBUG are accessed as members of the api object; that is, they are called as api.function(arguments).

BASIC

void setPositionTarget(float posTarget[3])	Sets a point as the position target Argument: array of three floats—x, y, and z position Return value: None
void setAttitudeTarget(float attTarget[3])	Sets a unit vector direction for the satellite to point toward Argument: array of three floats—x, y, and z components of unit vector Return value: None
void setVelocityTarget(float velTarget[3])	Sets the closed-loop x, y, and z components of the target velocity vector Argument: array of three floats—x, y, and z velocity Return value: None
void setAttRateTarget(float attRateTarget[3])	Sets the closed-loop target rotation rate components on the body frame Argument: array of three floats—rotation rates about the x, y, and z axes Return value: None
void setForces(float forces[3])	Sets the open-loop x, y, and z forces to be applied to the satellite Argument: array of three floats—x, y, and z forces Return value: None
void setTorques(float torques[3])	Sets the open-loop x, y, and z torques to be applied to the satellite Argument: array of three floats—torques about the x, y, and z axes Return value: None
void getMyZRState(float myState[12])	Gets the current state of the satellite in the following format: Places/indices 0-2: Position 3-5: Velocity 6-8: Attitude vector 9-11: Rotation rates Arguments: Array of 12 floats to store the state Return value: None
void getOtherZRState(float otherState[12])	Same as getMyZRState but gets the state of the opponent's satellite
unsigned int getTime()	Gets the time (in seconds) elapsed since the beginning of the game NOTE: This function is new for the 2013 season. Arguments: None Return value: Unsigned int containing time in seconds

DEBUG((“Some text!”))	Prints the supplied text to the console. Accepts formatted strings in the same format as the standard C printf function. NOTE: Make sure to use double parentheses. Do not type api. before this function. Arguments: String to be printed Return value: None

ADVANCED

void setQuatTarget(float quat[4])	Specifies a SPHERES quaternion attitude target for the satellite. Note that the scalar part of the quaternion Argument: array of four floats—quaternion components Return value: None
void getMySphState(float myState[13])	Gets the current SPHERES state (with quaternion attitude) for the satellite in the following format: Places/indices 0-2: Position 3-5: Velocity 6-9: Attitude quaternion 10-12: Rotation rates Arguments: Array of 13 floats to store the state Return value: None
void getOtherSphState(float otherState[13])	Same as getMySphState but gets the state of the opponent’s satellite
void spheresToZR(float stateSph[13], float stateZR[12])	Converts a 13-element state SPHERES state to a 12-element ZR state Arguments: Array of 13 floats containing a SPHERES state and an array of 12 floats to store the ZR state Return value: None
void attVec2Quat(float refVec[3], float attVec[3], float baseQuat[4], float quat[4])	Finds the quaternion that rotates refVec to attVec. This function determines the quaternion rotation from a user unit vector in the global frame. baseQuat defines the orientation of the satellite when refVec points in the desired direction. Setting baseQuat to something other than {0,0,0,1} allows the satellite to be rotated around the reference vector. In ZR, baseQuat is typicaly {1,0,0,0} to point the tank toward global +Z. When using this function to find the minimal rotation from the current attitude to a target attitude, it is advised to supply the current pointing direction in refVec, the desired attitude in attVec, and the current quaternion attitude in baseQuat. Since one of the degrees of freedom is unconstrained, using another approach can result in unexpected rotations about the pointing direction. Arguments: refVec—unit vector that specifies the body direction

	corresponding to no rotation. In ZR this is typcially the velcro (-X) face of the satellites, so refVec is {-1,0,0}. attVec—unit vector specifying the desired pointing direction baseQuat—quaternion specifying if there should be an initial rotation applied to the reference frame before calculating the output quaternion. For a tank-down nominal attitude, this should be {1,0,0,0} for a 180 degree rotation about X. quat—quaternion converted from attVec Return value: None
void quat2AttVec(float refVec[3], float quat[4], float attVec[3])	Converts a quaternion into a ZR attitude vector by rotating the supplied unit vector refVec using quat to determine the direction of attVec. NOTE: refVec is not copied to local storage, so it should be a different variable from attVec. Arguments: refVec unit vector that specifies the body direction corresponding to no rotation. In ZR this is typically the velcro (-X) face of the satellites, so refVec is {-1,0,0}. quat—quaternion to convert to ZR attitude vector attVec—converted attitude vector
void setPosGains(float P, float I, float D)	Sets the position controller gains Arguments: float P (proportional gain), float I (integral gain), float D (derivative gain) Return value: None
void setAttGains(float P, float I, float D)	Sets the attitude controller gains Arguments: float P (proportional gain), float I (integral gain), float D (derivative gain) Return value: None
void setCtrlMeasurement(float myState[13])	Sets the state measurement to be used in the standard ZR controllers instead of the default getMySphState() Arguments: float state[13] Return value: None
void setControlMode(CTRL_MODE posCtrl, CTRL_MODE attCtrl)	Sets the control mode for position and attitude control. The default is PD for position and PID for attitude. Arguments: Each of the two arguments should be one of the two macros CTRL_PD and CTRL_PID Return value: None
void setDebug(float values[7])	Adds an array of 7 user-defined debugging values to the satellite telemetry. The data can then be plotted with the ZR plotting tools. Arguments: Array of 7 floats Return value: None