

ESP32-S3 Gimbal Controller - Quick Reference

Serial Commands (115200 baud)

Basic Movement

Command	Description	Example
P<angle>	Move pitch	P45 → Move to 45°
Y<angle>	Move yaw	Y120 → Move to 120°
M<p>,<y>	Move both	M45,90 → Pitch 45°, Yaw 90°
H	Home (center)	H → Move to 90°, 90°
S	Emergency stop	S → Stop all motion
?	Print status	? → Show current state

Motion Parameters

Command	Description	Example
V<speed>	Max velocity (deg/s)	V60 → 60°/s
A<accel>	Acceleration (deg/s²)	A150 → 150°/s²

Angle Range Configuration

Command	Description	Example
RP,<min>,<max>	Set pitch range	RP,30,150 → 30° to 150°
RY,<min>,<max>	Set yaw range	RY,0,270 → 0° to 270°

Calibration (Alpha/Beta Offsets)

Command	Description	Example
OP<offset>	Pitch offset (alpha)	OP10 → +10° pitch offset
OY<offset>	Yaw offset (beta)	OY-5 → -5° yaw offset
CP	Calibrate pitch	CP → Set current as reference
CY	Calibrate yaw	CY → Set current as reference
CR	Reset calibration	CR → Clear all offsets

Initialization

```
cpp

#include "GimbalController.h"

// Create controller (GPIO pins, update rate Hz)
GimbalController gimbal(1, 2, 100);

void setup() {
    Serial.begin(115200);
    gimbal.begin(); // Initialize
    gimbal.home(); // Move to center
}

void loop() {
    gimbal.update(); // REQUIRED every loop!
    gimbal.processSerial(); // Handle commands
}
```

Movement Methods

```
cpp

// Single axis
gimbal.moveAxis(PITCH, 45.0f);
gimbal.moveAxis(YAW, 90.0f);

// Both axes
gimbal.moveToPosition(45.0f, 90.0f);

// Relative movement
gimbal.getAxis(PITCH)->moveRelative(10.0f);

// Home position
gimbal.home();

// Emergency stop
gimbal.stopAll();
```

Angle Range Configuration

```
cpp

// Set ranges (min, max in degrees)
gimbal.setAxisAngleRange(PITCH, 30.0f, 150.0f);
gimbal.setAxisAngleRange(YAW, 0.0f, 270.0f);

// Or per-axis
gimbal.getAxis(PITCH)->setAngleRange(30.0f, 150.0f);
```

Calibration (Alpha/Beta)

```
cpp

// Set offsets for servo alignment
gimbal.setAlphaAngle(10.0f); // Pitch offset
gimbal.setBetaAngle(-5.0f); // Yaw offset

// Interactive calibration
gimbal.calibrateAxis(PITCH); // Current position = reference
gimbal.calibrateAxis(YAW);

// Reset
gimbal.resetCalibration();

// Get offsets
float alpha = gimbal.getAlphaAngle();
float beta = gimbal.getBetaAngle();
```

Motion Parameters

```
cpp

// Per-axis configuration
gimbal.getAxis(PITCH)->setMaxVelocity(60.0f); // deg/s
gimbal.getAxis(PITCH)->setAcceleration(120.0f); // deg/s²
gimbal.getAxis(PITCH)->setDeceleration(150.0f); // deg/s²
```

Status & Information

```
cpp
```

```
// Check motion status
if (gimbal.isMoving()) {
    // Still moving
}

// Get positions
float pitchPos = gimbal.GetAxis(PITCH)->getPosition();
float yawPos = gimbal.GetAxis(YAW)->getPosition();
float velocity = gimbal.GetAxis(PITCH)->getVelocity();

// Get limits
float minAngle = gimbal.GetAxis(PITCH)->getMinAngle();
float maxAngle = gimbal.GetAxis(PITCH)->getMaxAngle();

// Print full status
gimbal.printStatus();
```

Common Use Cases

1. Standard Gimbal (Full Range)

```
cpp

void setup() {
    gimbal.begin();
    gimbal.home();
    // Default 0-180° for both axes
}
```

2. Limited Range Gimbal

```
cpp
```

```
void setup() {  
  gimbal.begin();  
  
  // Mechanical stops at 30° and 150°  
  gimbal.setAxisAngleRange(PITCH, 30.0f, 150.0f);  
  gimbal.setAxisAngleRange(YAW, 45.0f, 315.0f);  
  
  gimbal.home();  
}
```

3. Offset-Mounted Servos

```
cpp  
  
void setup() {  
  gimbal.begin();  
  
  // Servo mounted 10° forward  
  gimbal.setAlphaAngle(10.0f);  
  
  // Servo mounted 5° to right  
  gimbal.setBetaAngle(5.0f);  
  
  gimbal.home();  
}
```

4. Interactive Calibration

```
cpp
```

```

void setup() {
    gimbal.begin();

    Serial.println("Position gimbal level, then send CP");
    Serial.println("Position gimbal center, then send CY");

    // Wait for user to send CP and CY commands
}

void loop() {
    gimbal.update();
    gimbal.processSerial(); // Handles CP, CY commands
}

```

5. Save/Load Calibration

```

cpp

#include <Preferences.h>

void saveToFlash() {
    Preferences prefs;
    prefs.begin("gimbal", false);
    prefs.putFloat("alpha", gimbal.getAlphaAngle());
    prefs.putFloat("beta", gimbal.getBetaAngle());
    prefs.putFloat("p_min", gimbal.getAxis(PITCH)->getMinAngle());
    prefs.putFloat("p_max", gimbal.getAxis(PITCH)->getMaxAngle());
    prefs.end();
}

void loadFromFlash() {
    Preferences prefs;
    prefs.begin("gimbal", true);
    gimbal.setAlphaAngle(prefs.getFloat("alpha", 0.0f));
    gimbal.setBetaAngle(prefs.getFloat("beta", 0.0f));
    gimbal.setAxisAngleRange(PITCH,
        prefs.getFloat("p_min", 0.0f),
        prefs.getFloat("p_max", 180.0f));
    prefs.end();
}

```

Performance Profiles

Smooth Cinematic

```
cpp

gimbal.getAxis(PITCH)->setMaxVelocity(40.0f);
gimbal.getAxis(PITCH)->setAcceleration(80.0f);
```

Fast Action

```
cpp

gimbal.getAxis(PITCH)->setMaxVelocity(150.0f);
gimbal.getAxis(PITCH)->setAcceleration(300.0f);
```

High Precision

```
cpp

GimbalController gimbal(1, 2, 200); // 200Hz
gimbal.getAxis(PITCH)->setMaxVelocity(20.0f);
gimbal.getAxis(PITCH)->setAcceleration(40.0f);
```

Troubleshooting

Issue	Solution
Servo jitters	Add capacitor, lower update rate
Hits mechanical stop	Set angle range with (R) command
Wrong starting angle	Use alpha/beta offsets
Not smooth	Increase acceleration
Commands ignored	Check 115200 baud, add newline
Inverted motion	Use negative offset values

Example Session

```
> H      ← Home
Homing to center position...
```

> RP,20,160 ← Set pitch limits

Pitch axis range: 20.0° to 160.0°

> OP5 ← Set pitch offset

Alpha angle (pitch offset) set to 5.0°

> P45 ← Move pitch to 45°

Pitch -> 45.0°

> M60,120 ← Move both axes

Move -> Pitch: 60.0°, Yaw: 120.0°

> ? ← Check status

=== Gimbal Status ===

Pitch: 60.0° -> 60.0° (0.0 deg/s) [IDLE]

Yaw: 120.0° -> 120.0° (0.0 deg/s) [IDLE]

Calibration:

Alpha (Pitch offset): 5.0°

Beta (Yaw offset): 0.0°

Angle Ranges:

Pitch: 20.0° to 160.0°

Yaw: 0.0° to 180.0°

=====

> V80 ← Increase speed

Max velocity set to 80.0 deg/s

> A200 ← Increase acceleration

Acceleration set to 200.0 deg/s²

Library Version: 1.0.0

Compatible: ESP32-S3, ESP32

Dependencies: ESP32Servo

License: MIT