Scripting Language Reference

SimpleBGC 32bit

Actual firmware version: 2.70b5 and above
Updated: 13. Nov. 2026

© Basecamelectronics® 2013-2026

Overview

Scripting language is intended to control a gimbal by user-written program. This program is uploaded to controller and executed there without a connection to PC. Program can be started by a command from remote controller (RC), menu button or GUI button.

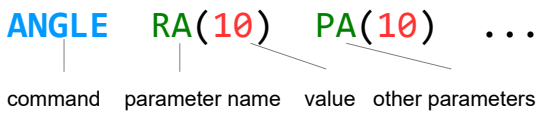
When script finishes, the system will be re-initialized – all parameters and modes of operation will be returned to default. To prevent this, use command `CONFIG INIT_SYSTEM_ON_FINISH(0)`

The SimpleBGC GUI includes a simple editor with syntax checking and execution tracing.

General rules on syntax:

- Each line can contain only one command
- Command may have up to 10 space-separated parameters
- All values must be in decimal notation with a fractional part separated by a dot. Fractional part can be omitted.
- You can add comments starting line with the # character, but comments are not saved to the board.

Command notation:



Some commands may have one-letter abbreviation, for example “ANGLE” has a shorthand “A”.

Parameters controlled in real-time *(frw. ver. 2.60b3)*

Values of parameters in any command can be dynamically updated during the execution of a program by linking them to any of RC signal sources. Just specify the signal source name and the desired range of the parameter's value, and RC signal will be captured and mapped to this range at the start of the command. Note that the value is not updated during the command execution.

All available RC signal source names are listed in the Appendix 3.

Usage:

```
<PARAMETER_NAME>(<RC_SOURCE_NAME>[<min_value>,<max_value>])
```

The advanced scripting version supports also variables as parameter values. Variables may be set by the command SET_ADJ_VAR.

Usage:

```
<PARAMETER_NAME>(<VARIABLE_NAME>[<offset>,<scale>])
```

, where parameter value is calculated as “<variable_value>*<scale> + <offset>”

Examples:

```
ANGLE RA(ADC1[-50,50]) PA(RC_VIRT_CH_1[0,90])
SPEED YS(TMP_VAR_1[0,1])
```

© Basecamelectronics® 2013-2026

Advanced functions

Some versions of SBGC32 controller having big FLASH and RAM memory size, support advanced functions which do not present in the basic version of script processor:

- Temporarily variables can be set by SET_ADJ_VAR command and used as a signal source in other commands;
- Basic arithmetic operations;
- IF .. GOTO command to implement control flow of the script (loops, conditional forks);
- Motion synchronization option to synchronize trajectory between axes in commands ANGLE, INC.

Advanced script functions are supported in the following controllers:

- “Extended” series: **“Extended”, “Extended long”, “Pro”, “CAN_MCU”, “OEM”**
- “Plus” series: **“Regular+”, “Tiny+”**

© Basecamelectronics® 2013-2026

Command reference

A, ANGLE — rotation to the absolute angle with a given speed

Command stops the program execution and waits until the angle reaches the given value. Rotation is done by the shortest path, so the angle can't exceed ±180°. If you want to turn to a greater angle, use INC command instead or set bit 1 in xMOD parameter.

I, INC – rotation by the relative angle with a given speed

The same as ANGLE but relative to the current position.

In both commands, you can specify a speed for each axis. If the speed is not set, default value is used, or the last value defined by the command "CONFIG".

Parameters:

```
RA(10.0) – target angle by the ROLL axis, in degrees*
PA(10.0) - target angle by the PITCH axis*
YA(10.0) - target angle by the YAW axis*
* Prior to 2.70b8 firmware, angle can't exceed ±720°

RS(10.0) – speed by the ROLL axis, °/s. Special value 0 is used to synchronize
motion with other axis.
PS(10.0) - speed by the PITCH axis
YS(10.0) - speed by the YAW axis

starting from firmware 2.59:
TIMEOUT(60.0) – wait timeout. If not specified, it is 60 seconds by default.

starting from firmware 2.73.2:
RMOD(flags), PMOD(flags), YMOD(flags) – modifier flags for each axis
      bit0: angle is frame-related
      bit1: don't wrap the distance by ±180° (frw.ver. 2.75.2)

starting from firmware 2.75.2:
M1A(10.0) … M7A(10.0) – target angle for auxiliary motors 1..7, in °
M1S(10.0) … M7S(10.0) – speed for auxiliary motors 1..7, in °/s
```

Example:

```
A RA(0) PA(0) YA(0)          # go to 0 (ground-related) angle by all axes
A RA(0) PA(0) YA(0) YMOD(1)  # level roll and pitch and go to home position by yaw
A YA(70) YS(1)               # rotate YAW to 70 degree with the speed 1 degree/sec.
I YA(270)                    # rotate YAW by 270 degree from current position
```

Motion synchronization

The advanced scripting version in firmware 2.70b5 and above provides an option to synchronize a motion of any axis with the motion of another axis by specifying its speed parameter = 0. The last axis with the 'angle'

© Basecamelectronics® 2013-2026

parameter preceding the zero 'speed' parameter in the parameter string is selected as a *sample for synchronization*. Target angles should be specified for both axes and should differ from current angles.

Example:

```
A RA(0) PA(0) YA(0)                 # move to the initial position
A PA(20) PS(50) YA(90) YS(0)        # YAW axis is asked to be synchronized with the PITCH axis. YAW
```
travels 90°, but it takes exactly the same time as PITCH traveling 20°. Both axes have perfectly matched acceleration / deacceleration ramps.

Wrong examples:

```
A YA(90) YS(0)                      # error: sync sample is not defined
A PA(0) YA(90) YS(0)                # error: target angle in the sample should differ from its current angle
```

S, SPEED – turn with a given speed until it encounters another command SPEED, ANGLE, INC or the end of the program.

The camera starts moving and program execution continues without delay.

Parameters:

```
RS(10.0) – speed by ROLL axis, °/s
PS(10.0) - speed by PITCH axis
YS(10.0) - speed by YAW axis

starting from firmware 2.75.2:
M1S(10.0) … M7S(10.0) – speed for auxiliary motors 1..7, °/s
```

Example:

```
S YS(5.5)                           # panning by YAW with the speed 5.5 degree/sec
S RS(0) PS(0) YS(0)                 # full stop
```

R, RESET — set the YAW angle = 0 for the current position

Parameters:

no

T, TRIGGER — trigger the state of the output pins

Parameters:

```
1(0) – pin ID * and its state:
       0 – LOW (GND)
       1 – HIGH (+3.3V)
       2 – FLOATING (frw.ver. 2.66+)
       3 – FLOATING + PULLUP 3.3V (frw.ver. 2.66+)

TIMEOUT(0.01) (optional) – if specified, program execution is delayed by a given
time, in seconds.** At the end of the timeout the specified pins will be inverted
(HIGH → LOW, LOW → HIGH). Don't set this parameter if you need to keep pin's
```

© Basecamelectronics® 2013-2026

state after triggering. A necessary delay may be created by the separate 'DELAY' command.

* Mapping between Pin ID and physical port can be found in the Appendix 1. The pin should be free of other functions (for example, not be used as an input in the RC settings tab).

** Accuracy is ±1 millisecond. Specifying a timeout can be useful for example, to manually set the shutter speed when shooting HDR images.

Example:

```
T 1(1) 2(1)                 # switch outputs ID=1,2 to a HIGH state
T 1(1) TIMEOUT(0.020)       # switch output ID=1 to a HIGH state and return back to LOW after 20ms
```

SERVO — set PWM signal pulse width on the specified pin

This command is used to control the servos connected to the special pins that supports output of PWM signal. PWM rate can be changed in the GUI (RC - PWM Output - PWM Rate, Hz).

Parameters:

1(1500) – Servo ID* and pulse duration, microseconds. For regular servo values are in the range 900-2100. Special value -1 frees pin and makes it high-impedance input.

TIME(3)** – travel time to target position, in seconds

TIMEOUT(1) – when target is reached, program execution is delayed by a given time, in seconds.

* Mapping between Servo ID and physical port can be found in the Appendix 1. The pin should be free of other functions (for example, not be used as an input in the RC settings tab).

**Firmware ver. 2.5x and above

Example:

```
SERVO 1(1500) 2(1500)       # move two servos to a middle position
SERVO 1(1000) TIME(3)       # slowly move from 1500 to 1000 in 3 seconds
SERVO 1(2000) TIMEOUT(1)    # set new value and wait 1 second
SERVO 1(-1) 2(-1)           # free two outputs
```

D, DELAY - delay of program execution

Parameters:

TIMEOUT(10) – delay, in seconds. Accuracy ±1 milliseconds.

Example:

```
D TIMEOUT(0.300)            # delay for 300 milliseconds
```

W, WAIT – waiting for the specified angle or speed

Program execution is delayed until all specified conditions will be satisfied, or a given timeout will be expired.

Parameters:

© Basecamelectronics® 2013-2026

RA(10.0) – angle by the ROLL axis, in °
PA(10.0) – angle by the PITCH axis
YA(10.0) – angle by the YAW axis
RS(0) - speed by the ROLL axis, °/s
PS(0) - speed by the PITCH axis
YS(0) - speed by the YAW axis
TIMEOUT(10) – max. time of waiting, in seconds. Default is 60 sec.

starting from firmware 2.75.2:
M1A(10.0) … M7A(10.0) – target angle for auxiliary motors 1..7, in °

Example:

```
W YA(30)                    # wait for YAW angle=30 degree
W RA(0) PA(0) YA(0)         # wait for angle=0 for all axes
W YS(0) TIMEOUT(1)          # wait to stop motion by YAW axis, but not more than 1 second.
```

CONFIG — configure some parameters used in program execution

Parameters:

ACC_LIMIT_R(100), ACC_LIMIT_P(100), ACC_LIMIT_Y(100) – acceleration limit for the ROLL, PITCH, YAW axes, degrees/sec²

SPEED_R(10), SPEED_P(10), SPEED_Y(10) – speed for the ROLL, PITCH, YAW axes, degree/sec. Range is 0..2000. Initial value is taken from the "RC SPEED" parameters

INIT_SYSTEM_ON_FINISH(1) *(ver. 2.60b3+)* - if set to 1 (default), system will be re-initialized when script finishes (all parameters and modes of operation will be returned to default). If set to 0, system will be left in it's current state.

JERK_SLOPE_R(50), JERK_SLOPE_P(50), JERK_SLOPE_Y(50) *(ver. 2.66+)* – jerk limiter for the ROLL, PITCH, YAW axis, the rise/fall time in ms

The changed parameters are not saved to the persistent memory and will be reset to their original values when program is finished.

Example:

```
C ACC_LIMIT_R(100) SPEED_Y(5)  # set an acceleration limit for the ROLL axis to 100 degree/sec² and a speed for YAW axis to 5 degree/sec.
```

MENU_CMD — execute a menu command *(frw. ver. 2.60b3+)*

Parameters:

CMD_ID(10) - command ID to run. Full list of commands is listed in the Appendix 2. In case of any command is missed there, it can be found in the Serial API documentation. Script parser does not wait for command to be finished!

TIMEOUT(1) - program execution is delayed by a given time, in seconds.

© Basecamelectronics® 2013-2026

SET_ADJ_VAR — set the value of adjustable variable *(frw. ver. 2.61)*

Parameters:

NAME(GYRO_TRUST) – name of variable. All names are provided in the SBGC32 User Manual, "Adjustable variables" section.

VALUE(100) – new value to set.

You can specify up to 5 pairs NAME - VALUE in a single command

Example:

```
SET_ADJ_VAR NAME(RC_SPEED_PITCH) VALUE(60) NAME(RC_SPEED_ROLL) VALUE(50)
```

The advanced scripting version supports temporarily variables and arithmetic operations on them. Variables can be used as parameters in other commands, can be sent for external logging, or used for flow control (see IF .. GOTO command definition). System keeps their values between script executions, so you need to reset values at the start of the script, if needed.

Variables:

- **TMP_VAR_1 .. TMP_VAR_10** (floating point type) – may be used in a script only
- **API_VIRT_CH_1 .. API_VIRT_CH_32** (int16 type) – may be used as a source of control signal in all places where RC channels are assigned, value ±16384 encodes the full RC range.

Operations:

OPERATOR(=+-*/%) – do arithmetic operation on two values, where 1st value is result of all previous calculations and 2nd value is a value of the next parameter in sequence. If OPERATOR goes next to the NAME parameter, the current value of the variable is used.

Examples:

```
SET_ADJ_VAR NAME(TMP_VAR_1) VALUE(1) NAME(TMP_VAR_2) VALUE(1)
```

Result: TMP_VAR_1 = 1, TMP_VAR_2 = 1

```
SET_ADJ_VAR NAME(TMP_VAR_1) OPERATOR(+) VALUE(10)
```

Result: TMP_VAR_1 = 11

```
SET_ADJ_VAR NAME(TMP_VAR_3) VALUE(TMP_VAR_1[0,1]) OPERATOR(+) VALUE(TMP_VAR_2[-5,4])
OPERATOR(*) VALUE(10)
```

*Result: TMP_VAR_3 = (11 + (1*4 - 5))*10 = 100*

PLAY_SOUND — plays standard sound and blinks LED *(frw. ver. 2.68)*

Parameters:

CMD_ID(1) – sound to play:

- 1 – calibration
- 2 – confirmation
- 4 – error
- 8 – alarm
- 16 – single click
- 32 – action complete
- 64 – startup melody

Modifiers (can be combined with the ID by OR operator):

- 16384 – silent mode: do not emit sound, just blink LED

© Basecamelectronics® 2013-2026

TIMEOUT(1) – program execution is delayed by a given time, in seconds. If no timeout is specified, program goes to a next command, playing sound in the background.

IF — conditional flow control of the script *(frw. ver. 2.69b7)*

The advanced scripting version supports a flow control using IF .. GOTO command together with the SET_ADJ_VAR command and temporarily variables.

Parameters:

NAME(<VARIABLE_NAME>) – name of the variable to compare, one of the:

- TMP_VAR_1..TMP_VAR_10 – floating point type
- API_VIRT_CH_1..API_VIRT_CH_32 – int16 type
- all adjustable variables listed in “User Manual”

VALUE(<value | RC source[min,max] | variable[offset,scale]>) – value to compare. It may be a regular decimal value or RC signal source/variable.

COND(<condition>) – condition to be used in a comparison of variable and the following VALUE parameter. Both values are rounded to integers before comparing, excepting >> and << operations.

- = - equal
- != - not equal
- > - greater
- >= - greater or equal
- < - less
- <= - less or equal
- >> - greater, not rounded
- << - less, not rounded

GOTO(<line number | label_name>) – line number in script to jump if condition is true. It's recommended to use labels instead. You can place label before any command or on empty line, in a form of “LABEL_NAME: ..command..”

Command expects two values, provided by the NAME or VALUE parameters, one COND parameter and one GOTO parameter. In case of (NAME|VALUE)x2 and COND parameters are missed, GOTO will do unconditional jump.

Example: repeat INC command 10 times to rotate YAW axis by 10-degree steps, sending a short pulse to pin AUX1 after each step.

```
SET_ADJ_VAR NAME(TMP_VAR_1) VALUE(0)
LOOP1: SET_ADJ_VAR NAME(TMP_VAR_1) OPERATOR(+) VALUE(1)
INC YA(10)
TRIGGER 16(1) TIMEOUT(0.5)
IF NAME(TMP_VAR_1) COND(<) VALUE(10) GOTO(LOOP1)
```

© Basecamelectronics® 2013-2026

Appendix 1. Mapping between pin IDs and physical ports

Labeling on the board	Trigger pin ID	Servo ID
RC_ROLL	1	-
RC_PITCH	2	3
RC_YAW	5	-
EXT_FC_ROLL	3	1
EXT_FC_PITCH	4	2
AUX1	16	4
AUX2	17	-
AXU3	18	-
BUZZER	32	-
CAN_DRV1_AXU1	64*	5
CAN_DRV1_AXU2	65*	-
CAN_DRV1_AUX3	66*	6
...	...	...
CAN_DRV7_AUX1	82*	17
CAN_DRV7_AUX2	83*	-
CAN_DRV7_AUX3	84*	18
CAN_IMU_AUX1	85*	-
CAN_IMU_AUX2	86*	-
CAN_IMU_FRAME_AUX1	87*	-
CAN_IMU_FRAME_AUX2	88*	-

** Available in main frw. ver. 2.74.3, CAN_DRV frw. 1.48, CAN_IMU frw. 1_55*

© Basecamelectronics® 2013-2026

Appendix 2: Menu commands

```
MENU_CMD_PROFILE1 = 1
MENU_CMD_PROFILE2 = 2
MENU_CMD_PROFILE3 = 3
MENU_CMD_SWAP_PITCH_ROLL = 4
MENU_CMD_SWAP_YAW_ROLL = 5
MENU_CMD_CALIB_ACC = 6
MENU_CMD_RESET = 7
MENU_CMD_SET_ANGLE = 8
MENU_CMD_CALIB_GYRO = 9
MENU_CMD_MOTOR_TOGGLE = 10
MENU_CMD_MOTOR_ON = 11
MENU_CMD_MOTOR_OFF = 12
MENU_CMD_FRAME_UPSIDE_DOWN = 13
MENU_CMD_PROFILE4 = 14
MENU_CMD_PROFILE5 = 15
MENU_CMD_AUTO_PID = 16
MENU_CMD_LOOK_DOWN = 17
MENU_CMD_HOME_POSITION = 18
MENU_CMD_RC_BIND = 19
MENU_CMD_CALIB_GYRO_TEMP = 20
MENU_CMD_CALIB_ACC_TEMP = 21
MENU_CMD_BUTTON_PRESS = 22
MENU_CMD_RUN_SCRIPT1 = 23
MENU_CMD_RUN_SCRIPT2 = 24
MENU_CMD_RUN_SCRIPT3 = 25
MENU_CMD_RUN_SCRIPT4 = 26
MENU_CMD_RUN_SCRIPT5 = 27
MENU_CMD_CALIB_MAG = 33
MENU_CMD_LEVEL_ROLL_PITCH = 34
MENU_CMD_CENTER_YAW = 35
MENU_CMD_UNTWIST_CABLES = 36
MENU_CMD_SET_ANGLE_NO_SAVE = 37
MENU_HOME_POSITION_SHORTEST = 38
MENU_CENTER_YAW_SHORTEST = 39
MENU_ROTATE_YAW_180 = 40
MENU_ROTATE_YAW_180_FRAME_REL = 41
MENU_SWITCH_YAW_180_FRAME_REL = 42
MENU_SWITCH_POS_ROLL_90 = 43
MENU_START_TIMELAPSE = 44
MENU_CALIB_MOMENTUM = 45
MENU_LEVEL_ROLL = 46
MENU_LOAD_PROFILE_SET1 = 48
MENU_LOAD_PROFILE_SET2 = 49
MENU_LOAD_PROFILE_SET3 = 50
MENU_LOAD_PROFILE_SET4 = 51
MENU_LOAD_PROFILE_SET5 = 52
```

© Basecamelectronics® 2013-2026

```
MENU_LOAD_PROFILE_SET_BACKUP = 53
MENU_INVERT_RC_ROLL = 54
MENU_INVERT_RC_PITCH = 55
MENU_INVERT_RC_YAW = 56
MENU_SNAP_TO_FIXED_POSITION = 57
MENU_CAMERA_REC_PHOTO_EVENT = 58
MENU_CAMERA_PHOTO_EVENT = 59
MENU_MOTORS_SAFE_STOP = 60
MENU_CALIB_ACC_AUTO = 61
```

© Basecamelectronics® 2013-2026

Appendix 3: RC signal source names

Hardware inputs for PWM format:

```
RC_ROLL_PWM
RC_PITCH_PWM
RC_YAW_PWM
FC_ROLL_PWM
FC_PITCH_PWM
```

RC input should be enabled (i.e. assigned to control of any axis or CMD channel in the RC settings).

Analog inputs (joystick connection):

```
ADC1
ADC2
ADC3
```

Sum-PPM or serial protocols (Spektrum, s-bus):

```
RC_VIRT_CH_1
RC_VIRT_CH_2
..
RC_VIRT_CH_31
```

Desired serial protocol should be selected in the RC settings;
Channels 20..31 have special meaning: they are linked to the sin, cos of the angles of motors.

Serial API virtual channels (can be set by the external devices or mobile applications connected by SBGC Serial API protocol):

```
API_VIRT_CH_1
API_VIRT_CH_2
..
API_VIRT_CH_31
```

Motor angles:

INNER_MOTOR_ANGLE	MIDDLE_MOTOR_ANGLE	OUTER_MOTOR_ANGLE
COS_INNER_MOTOR	COS_MIDDLE_MOTOR	COS_OUTER_MOTOR
ABS_COS_INNER_MOTOR	ABS_COS_MIDDLE_MOTOR	ABS_COS_OUTER_MOTOR
SIN_INNER_MOTOR	SIN_MIDDLE_MOTOR	SIN_OUTER_MOTOR
ABS_SIN_INNER_MOTOR	ABS_SIN_MIDDLE_MOTOR	ABS_SIN_OUTER_MOTOR

© Basecamelectronics® 2013-2026

Example 1: Shooting 3-row spherical panorama with increment of 60 degrees in 20 seconds, AUX1 controls the shutter.

```
# Setup a high speed for a quick completion of the panorama
CONFIG SPEED_R(200) SPEED_P(200) SPEED_Y(200) ACC_LIMIT_R(500) ACC_LIMIT_P(1000) ACC_LIMIT_Y(500)
# Reset the origin of the YAW: Start recording with the current azimuth
R
# Turn off the shutter pin
T 16(0)

# Make 1st shot in the zenith position: raise the camera vertically
A RA(0) PA(-90) YA(0)
# Make a shot
T 16(1) TIMEOUT(0.100)

# Tilt 45 degrees up to make the 1st row
A RA(0) PA(-45) YA(0)
T 16(1) TIMEOUT(0.100)
# Panning with increment of 60 degrees clockwise
A YA(60)
T 16(1) TIMEOUT(0.100)
A YA(120)
T 16(1) TIMEOUT(0.100)
A YA(180)
T 16(1) TIMEOUT(0.100)
A YA(240)
T 16(1) TIMEOUT(0.100)
A YA(300)
T 16(1) TIMEOUT(0.100)

# Camera is tilted 0 degrees to make 2nd row
A RA(0) PA(0) YA(0)
T 16(1) TIMEOUT(0.100)
# Panning with increment of 60 degrees counterclockwise (avoid double-twisting of the wires)
A YA(-60)
T 16(1) TIMEOUT(0.100)
A YA(-120)
T 16(1) TIMEOUT(0.100)
A YA(-180)
T 16(1) TIMEOUT(0.100)
A YA(-240)
T 16(1) TIMEOUT(0.100)
A YA(-300)
T 16(1) TIMEOUT(0.100)

# Tilt 45 degrees down to make the 3rd row
A RA(0) PA(45) YA(0)
T 16(1) TIMEOUT(0.100)
# Panning with increment of 60 degrees clockwise
A YA(60)
T 16(1) TIMEOUT(0.100)
```

© Basecamelectronics® 2013-2026

```
A YA(120)
T 16(1) TIMEOUT(0.100)
A YA(180)
T 16(1) TIMEOUT(0.100)
A YA(240)
T 16(1) TIMEOUT(0.100)
A YA(300)
T 16(1) TIMEOUT(0.100)


# Shoot in the nadir position
A YA(0) PA(90)
T 16(1) TIMEOUT(0.100)

# Returns the camera to its original position (untwist wires)
A RA(0) PA(0)
I YA(-360)

### End of program ###
```

© Basecamelectronics® 2013-2026

Example 2: Pan at 5 deg/sec by 90 degrees. AUX1 operates recording

```
# Reset the origin of the YAW: Start recording with the current azimuth
R
# Tilt the camera 30 degrees down and level it
A RA(0) PA(30) YA(0)
# Start recording
T 16(1)
# Writing freeze for 3 seconds.
D TIMEOUT(3)
# Setup low acceleration for smooth start and stop of the motion
CONFIG ACC_LIMIT_Y(5)
# Panning with a speed of 5 degree/sec. clockwise
S YS(5)
# Wait until turned by 90 degrees.
W YA(90)
# Stop panning (de-acceleration starts here)
S YS(0)
# Wait until de-acceleration is finished
W YS(0)
# Writing freeze for 3 seconds.
D TIMEOUT(3)
# Stop recording
T 16(0)
### End of program ###
```

Example 3: Time-lapse shooting with minimized gyroscope drift

```
# Let system to know that the frame is still, to compensate a drift of gyroscope;
SET_ADJ_VAR NAME(FRAME_HEADING_ANGLE) VALUE(0)
# Set the 'gyro trust' parameter low enough to better compensate drift of gyroscope
SET_ADJ_VAR NAME(GYRO_TRUST) VALUE(60)
# (Optional) move camera to the desired initial position. Skip this command to start from the current
position
#ANGLE PA(0) RA(0)
# Pan left with the speed 0.1 degrees/sec and tilt up with the speed half slower.
SPEED YS(0.1) PS(-0.05)
# Wait 10 minutes
DELAY TIMEOUT(600)
```

© Basecamelectronics® 2013-2026