# Command : **servo**

## Target : rp2040 Microcontroller

## Description : Move a servo motor

## Command format :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| servo | port  P1 | command  P2 | motor  P3 | angle  P4 | [speed]  P5 |

## Parameters :

### P1 Define source task of command

* Specifies the return route of the command response. Will allow multiple tasks to send commands and for the responses to get back to the correct source.
* Software not implemented as yet, therefore use any value from 0->63

### P2 Command that can be executed by the RP2040 microcontroller

* Command list

|  |  |  |
| --- | --- | --- |
| Command | Code | Notes |
| ABS\_MOVE | 0 | Move servo to an absolute +/-position |
| ABS\_MOVE\_SYNC | 1 | Set ABS\_MOVE command, but wait for a RUN\_SYNC\_MOVES command to execute |
| SPEED\_MOVE | 2 | Move servo at a set speed |
| SPEED\_MOVE\_SYNC | 3 | Set SPEED\_MOVE command, but wait for a RUN\_SYNC\_MOVES command to execute |
| RUN\_SYNC\_MOVES | 4 | Run ALL pending SYNC commands |
| T\_DELAY | 5 | Delay a number of milliseconds |
| STOP | 6 | Stop current servo move |
| STOP\_ALL | 7 | Stop ALL servo moves |
| ENABLE | 8 | Enable/disable specified servo |

### P3 Motor code

* Motor list

|  |  |  |  |
| --- | --- | --- | --- |
| Motor | Code | Range (degrees) | Notes |
| Right\_EYE\_left/right | 0 | +/-25 | -25=fully left; +25=fully right |
| Right\_EYE\_up/down | 1 | +/-45 | -45=fully UP; +45=fully DOWN |
| Right\_EYE\_lid | 2 | +/-25 | -25=fully CLOSED; +25=fully OPEN |
| Right\_EYE\_brow | 3 | +/-40 | -40=angle UP; +40=angle DOWN |
| Left\_EYE\_left/right | 4 | +/-25 | -25=fully left; +25=fully right |
| Left\_EYE\_up/down | 5 | +/-45 | -45=fully UP; +45=fully DOWN |
| Left\_EYE\_lid | 6 | +/-25 | -25=fully CLOSED; +25=fully OPEN |
| Left\_EYE\_brow | 7 | +/-40 | -40=angle UP; +40=angle DOWN |
| Mouth | 8 | 0 -> +45 | 0 = mouth OFF, +45 = mouth ON |

### P4 Servo angle value

* Absolute angle specified in degrees
* Microcontroller will check that angle is within mechanical limits

### P5 Servo speed value (optional)

* Only used in TIMED\_MOVE command
* Timed value is in units of 100mS

## Returned

|  |  |
| --- | --- |
| Port  (int) | Status  (int) |

* Status value : signed integer
* Statue/Errors codes

|  |  |  |
| --- | --- | --- |
| Error | Code | Notes |
| OK |  | Success |
| BAD\_PORT\_NUMBER | -104 | Port out with range 0 to 63 |
| BAD\_NOS\_PARAMETERS | -105 | Incorrect number of parameters |
| BAD\_BASE\_PARAMETER | -106 |  |
| PARAMETER\_OUTWITH\_LIMITS | -107 | A parameter is out with set limits |
| BAD\_SERVO\_COMMAND | -108 | Unrecognised servo command |

# Command : **stepper**

## Target : rp2040 Microcontroller

## Description : Move a stepper motor

## Command format :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| stepper | port  P1 | command  P2 | motor  P3 | angle  P4 |

## Parameters :

### P1 Define source task of command

* Specifies the return route of the command response. Will allow multiple tasks to send commands and for the responses to get back to the correct source.
* Software not implemented as yet, therefore use any value from 0->63

### P2 Command that can be executed by the RP2040 microcontroller

* Command list

|  |  |  |
| --- | --- | --- |
| Command | Code | Notes |
| SM\_REL\_MOVE | 0 | Move steps from current position |
| SM\_ABS\_MOVE | 1 | Move to step value from origin position |
| SM\_REL\_MOVE\_SYNC | 2 | Set SM\_REL\_MOVE command, but wait for a servo RUN\_SYNC\_MOVES command to execute |
| SM\_ABS\_MOVE\_SYNC | 3 | Set SM\_ABS\_MOVE command, but wait for a servo RUN\_SYNC\_MOVES command to execute |
| SM\_CALIBRATE | 4 | Run motor calibrate process   * Run motor to maximum point and stop when limit switch is detected * Run motor to minimum point and stop when limit switch is detected * Update limit values and run motor to central position |

### P3 Motor

* “Pi the Robot” has only a single stepper motor (head rotation), therefore the only legal value is 0.

### P4 angle

* Limited to +/-60 degrees.
* Angle is converted to steps and checked to ensure the move is within limits
  + Very important for relative moves

## Returned

|  |  |
| --- | --- |
| Port  (int) | Status  (int) |

1. Status/Errors codes

|  |  |  |
| --- | --- | --- |
| Error | Code | Notes |
| OK | 0 | Sucess |
| LETTER\_ERROR | -100 | Parse cmd : Letter in a number |
| DOT\_ERROR | -102 | Parse cmd : extra point in real value |
| PLUSMINUS\_ERROR | -103 | Parse cmd : +,- symbol error |
| QUOTE\_ERROR | -131 | Parse cmd : quote symbol (“) error |
|  |  |  |
| BAD\_PORT\_NUMBER | -104 | Port out with range 0 to 63 |
| BAD\_NOS\_PARAMETERS | -105 | Incorrect number of parameters |
| PARAMETER\_OUTWITH\_LIMITS | -107 | A parameter is out with set limits |
| STEPPER\_CALIBRATE\_FAIL | -109 |  |
| BAD\_STEPPER\_COMMAND | -110 |  |
| BAD\_STEP\_VALUE | -111 |  |
| MOVE\_ON\_UNCALIBRATED\_MOTOR | -112 |  |
| EXISTING\_FAULT\_WITH\_MOTOR | -113 |  |
| SM\_MOVE\_TOO\_SMALL | -114 |  |
| LIMIT\_SWITCH\_ERROR | -115 |  |
| UNKNOWN\_STEPPER\_MOTOR\_STATE | -116 |  |
| STEPPER\_BUSY | -117 |  |

# Command : **display**

## Target : rp2040 Microcontroller

## Description : Interact with touch screen

## Notes :

1. There are two types of display command ; API and HIGH LEVEL. There are four API commands that map to the four relevant commands understood by the physical hardware. These use a ‘global index’ access pointer to specify the object.and should not (in most circumstances) be used the high level control software. Users should use the HIGH LEVEL commands to access display hardware. .. One of the main differences is that these HIGH LEVEL command use a ‘local index (see Note 2) to access screen objects. Also they are more descriptive of what effect you want to achieve.
2. High l(see Note 2) level commands use ‘local index’ to access screen objects. ‘local index’ has a value of 0 to ‘n’ for each form. For example, the first winbutton screen object of a form will have a ‘local index’ value of 0. The microcontroller connected to the display will convert the ‘local index’ value to a ‘global index’ value that is need to access the object on the display hardware.

## High level commands

## Command format :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| display | port  P1 | command  P2 | value\_1  P3 | value\_2  P4 |

**display port command form local\_index param1 … paramN**

|  |  |  |
| --- | --- | --- |
| Sub-command | Notes | Value |
| **display** | Specify a command to interact with the system LCD touch screen |  |
| **port** | Specifies the return route of the command response. Will allow multiple tasks to send commands and for the responses to get back to the correct source. | 0 -> 63  (use 0) |
| **command** | Sub-command understood by the RP2040 microcontroller  () | refer to sub-command list |
| **form** | Screen form | 0 -> MAX\_FORM |
| **local\_index** |  | 0 -> MAX\_INDEX |
| **param1** |  |  |
| **paramN** |  |  |

## Display sub-commands – ASCII string

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Port **(1)** | Display sub-command code |  |  |  |  |  |
| P[1] | P[2] **(2)** | Code | P[3] | P[4] | P[5] | Nos parameters **(3)** |
| 0 | SET\_uLCD\_FORM | 0 | form | ------- | ------- | 4 |
| 0 | GET\_uLCD\_FORM | 1 | ------- | ------- | ------- | 3 |
| 0 | SET\_uLCD\_CONTRAST | 2 | Contrast value | ------- | ------- | 4 |
| 0 | READ\_uLCD\_BUTTON | 3 | form | local index | ------- | 5 |
| 0 | READ\_uLCD\_SWITCH | 4 | form | local index | source **(8)** | 6 |
| 0 | READ\_uLCD\_OBJECT **(7)** | 5 | Object type | global index |  | 5 |
| 0 | WRITE\_uLCD\_STRING | 6 | form | local index | string **(4)** | 6 |
| 0 | WRITE\_uLCD\_OBJECT **(7)** | 7 | Object type | global index | value **(5)** | 6 |
| 0 | SCAN\_uLCD\_FORM\_BUTTON\_PRESSES **(6)** | 8 | form | ------- | ------- | 4 |
| 0 | SCAN\_uLCD\_SWITCHES **(9)** | 9 | form | ------- | ------ | 4 |

Notes

1. The ‘**port’** parameter allows the reply from a command to be directed to the task that initiated the command. Not implemented at present so set port parameter to zero.
2. P[n] refers to the nth parameter. Parameter P[0] is the ‘**display’** command string
3. Total number of items in a command equals the number of parameters plus the initial ‘**display’** ( **P[0]** ) command
4. ASCII string terminated by a NEWLINE (‘\n’) character
5. 16-bit value
6. Scans the buttons on the active form. Returns the **local** **index** of the first button it finds that has a PRESSED state. The command is terminated at this point. Returns a value of -1 if no pressed button is detected.
7. Provide raw access to the READ/WRITE command of the display. These only require the object type and the global index. No reference is made to the form.
8. **source** = 0 : Read switch from display; **source** = 1 : read from in memory **form\_data** structure
9. Read all switches on a form. Data is returned as a series of bits in a 16-bit value

## Values returned from display sub-commands – ASCII string

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Display sub-command **(1)** | P[0] | P[1] | P[2] | P[3] | P[4] | Nos parameters |
| SET\_uLCD\_FORM | port | status | ------- | ------- | ------- | 2 |
| GET\_uLCD\_FORM | port | status | form | ------- | ------- | 3 |
| SET\_uLCD\_CONTRAST | port | status | ------- | ------- | ------- | 2 |
| READ\_uLCD\_BUTTON | port | status | value | state **(2)** | press length **(3)** | 5 |
| READ\_uLCD\_SWITCH | port | status | value | ------- | ------- | 2 |
| READ\_uLCD\_OBJECT | port | status | ------ | ------ | ------- | 2 |
| WRITE\_uLCD\_STRING | port | status | ------ | ------ | ------- | 2 |
| WRITE\_uLCD\_OBJECT | port | status | ------ | ------ | ------- | 2 |
| SCAN\_uLCD\_FOR\_PRESSED\_BUTTON | port | status | local index | press length | ------- | 4 |
| SCAN\_uLCD\_SWITCHES | port | status | value **(4)** |  |  | 3 |

Notes

1. All returned values are integer values.
2. state is either PRESSED or NOT\_PRESSED
3. press length is in units of 100mS
4. 16-bit value that encodes the binary values of the switches scanned

## Display sub-command list

|  |  |  |
| --- | --- | --- |
| Command | Code | Notes |
| SET\_FORM | 0 | Display selected form on screen |
| GET\_FORM | 1 | Get number of current active form |
| SET\_CONTRAST | 2 | Set display contrast |
| READ\_uLCD\_BUTTON | 3 | Read specified button from current displayed form |
| READ\_uLCD\_SWITCH | 4 | Read specified switch from current displayed form |
| WRITE\_uLCD\_STRING | 5 | Write string to current form |
| SCAN\_FOR\_PRESSED\_BUTTON | 6 | Check for button press in current form. |

### P3 and P4

|  |  |  |  |
| --- | --- | --- | --- |
| Command | P3 | P4 | Notes |
| SET\_FORM | Form object code >=0 | unused |  |
| GET\_FORM | unused | unused | Return value >=0 |
| SET\_CONTRAST | Contrast value 0->100% | unused |  |
| READ\_BUTTON | Button object code >=0 | unused | Return value 0 or 1 |
| WRITE\_STRING | String object code >=0 | string |  |

## Returned

* SET\_FORM, SET\_CONTRAST, WRITE\_STRING commands

|  |  |
| --- | --- |
| Port  (int) | Status  (int) |

* GET\_BUTTON command

|  |  |  |
| --- | --- | --- |
| Port  (int) | Status  (int) | button data  (int) |

* Status error codes

|  |  |  |
| --- | --- | --- |
| Error | Code | Notes |
| OK | 0 | Sucess |
| LETTER\_ERROR | -100 | Parse cmd : Letter in a number |
| DOT\_ERROR | -102 | Parse cmd : extra point in real value |
| PLUSMINUS\_ERROR | -103 | Parse cmd : +,- symbol error |
| QUOTE\_ERROR | -131 | Parse cmd : quote symbol (“) error |
|  |  |  |
| GEN4\_uLCD\_NOT\_DETECTED | -119 |  |
| GEN4\_uLCD\_WRITE\_OBJ\_FAIL | -120 |  |
| GEN4\_uLCD\_WRITE\_OBJ\_TIMEOUT | -121 |  |
| GEN4\_uLCD\_WRITE\_CONTRAST\_FAIL | -122 |  |
| GEN4\_uLCD\_WRITE\_CONTRAST\_TIMEOUT | -123 |  |
| GEN4\_uLCD\_READ\_OBJ\_FAIL | -124 |  |
| GEN4\_uLCD\_READ\_OBJ\_TIMEOUT | -125 |  |
| GEN4\_uLCD\_CMD\_BAD\_FORM\_INDEX | -126 |  |
| GEN4\_uLCD\_WRITE\_STR\_TOO\_BIG | -127 |  |
| GEN4\_uLCD\_WRITE\_STRING\_FAIL | -128 |  |
| GEN4\_uLCD\_WRITE\_STRING\_TIMEOUT | -129 |  |
| GEN4\_uLCD\_BUTTON\_FORM\_INACTIVE | -130 |  |

# Command : **ping**

## Target : rp2040 Microcontroller

## Description : check microcontroller is responding

## Command format :

|  |  |  |
| --- | --- | --- |
| ping | port  P1 | ping value  P2 |

## Parameters :

### P1 Define source task of command

* Specifies the return route of the command response. Will allow multiple tasks to send commands and for the responses to get back to the correct source.
* Software not implemented as yet, therefore use any value from 0->63

### P2 Ping value

* Positive integer value sent
* If all is well, ping command will return the value sent plus 1

## Returned

|  |  |  |
| --- | --- | --- |
| Port  (int) | Status  (int) | Ping value + 1  (int) |

* Status/error codes

|  |  |  |
| --- | --- | --- |
| Error | Code | Notes |
| OK | 0 | Sucess |
| LETTER\_ERROR | -100 | Parse cmd : Letter in a number |
| DOT\_ERROR | -102 | Parse cmd : extra point in real value |
| PLUSMINUS\_ERROR | -103 | Parse cmd : +,- symbol error |
| QUOTE\_ERROR | -131 | Parse cmd : quote symbol (“) error |
|  |  |  |

# Command : **speak**

## Target : PC, Rasp-Pi

## Description : Output TTS string

## Command format :

speak

t

f

w

c

“text”

“filename”

P1

P2

P3

## Parameters :

### P1 Define source of text to be spoken

* **t** = text is part of the command
  + quoted string as parameter 3 (P3)
* **f** = text in a file
  + quoted filename as parameter 3 (P3)

### P2 Define action when text is spoken

* **w**  = return from command after speech is complete
* **c** = return from command when speech start

## Returned

* Status/error codes

|  |  |  |
| --- | --- | --- |
| Error | Code | Notes |
| OK | 0 | Sucess |
| TEXTFILE\_NOT\_FOUND | -100 | Parse cmd : Letter in a number |

# Command : **neopixel**

## Target : rp2040 Microcontroller

## Description : Control string of Neopixel RGB LEDS

## Command format :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| neopixel | port  P1 | command  P2 | [par1]  P3 | [par2]  P4 | [par3]  P5 | [par4]  P6 |

## Parameters :

### P1 Define source task of command

* Specifies the return route of the command response. Will allow multiple tasks to send commands and for the responses to get back to the correct source.
* Software not implemented as yet, therefore use any value from 0->63

### P2 Command that can be executed by the RP2040 microcontroller

* Command list

|  |  |  |
| --- | --- | --- |
| Command | Code | Notes |
| NP\_SET\_PIXEL\_ON | 0 | Set pixel to its ON colour |
| NP\_SET\_PIXEL\_OFF | 1 | Set pixel to its OFF colour |
| NP\_SET\_PIXEL\_FLASH | 2 | Set pixel to flashing between two colours |
| NP\_BLANK\_ALL | 3 | Set ALL pixels to OFF (BLACK) |

### P3-P6 Command parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Command | P3 | P4 | P5 | P6 | Notes |
| NP\_SET\_PIXEL\_ON | colour | N/A | N/A | N/A | **a** |
| NP\_SET\_PIXEL\_OFF | colour | N/A | N/A | N/A | **a** |
| NP\_SET\_PIXEL\_FLASH | on-colour | on-time | off-colour | off-time | **a,b** |
| NP\_BLANK\_ALL | N/A | N/A | N/A | N/A | **-** |

### Notes

1. The system is set to understand the RAINBOW colours.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Colour | Code | Name | R-value | G-value | B-value |
| White | 0 | N\_BLACK | 0xFF | 0xFF | 0xFF |
| Red | 1 | N\_RED | 0xFF | 0x00 | 0x00 |
| Orange | 2 | N\_ORANGE | 0xFF | 0x7F | 0x00 |
| Yellow | 3 | N\_YELLOW | 0xFF | 0xFF | 0x00 |
| Green | 4 | N\_GREEN | 0x00 | 0xFF | 0x00 |
| Blue | 5 | N\_BLUE | 0x00 | 0x00 | 0xFF |
| Indigo | 6 | N\_INDIGO | 0x4B | 0x00 | 0x82 |
| Violet | 7 | N\_VIOLET | 0x94 | 0x00 | 0xD3 |
| Black | 8 | N\_BLACK | 0x00 | 0x00 | 0x00 |

1. Times are set in units of 10mS.