# SSC-32 Binary Commands.

Version: 1.00
Date: Apr 25, 2011

## Table of Contents



---

## Binary Command Overview

Some of the more recent General Purpose Sequencer versions of the SSC-32 firmware have added an additional set of commands to support some of the more commonly used operations of the SSC-32, but in a shorter more concise format.

These commands are in a binary non-human readable format. That is up till now all of the commands that were sent to the SSC-32 were such that all of the characters in the command were in the normal human readable range of a space character, which has the value 32 decimal (hex 20) up to the "~" character which has the value 126 (hex 7E) and the commands were variable length and were executed when the user sent the Carriage Return, which has a value 13 decimal (D hex). This format is great for being able to enter commands through programs like Lynxterm. However having a textual commands set causes the programs to have to send (or receive) several characters over the relatively slow communication port and some programs needed a faster mechanism.

So a few new commands were added that supported a fixed binary format. These commands are distinguished from other commands as their first character starts out side of the human readable range. In particular they are in the range 128-255 (Hex 80-FF).

---

## Binary Servo Move or Group Move

0x80 - 0x9F are the pulse width commands, and specify a servo number 0-31 (0x80=servo0, 0x81=servo1, etc.). The next 2 bytes must be the desired pulse width in microseconds, MSB first.

0xA0 is the optional speed command for a servo. It must immediately follow the pulse width command for the servo, and the next 2 bytes must be the max speed in microseconds/second.

0xA1 is the move time. It is required for a group move, and the move will not begin until the move time is received. The next 2 bytes must be the move time in milliseconds. If you want the move to be as fast as possible, just specify a move time of 0.

As an example, the group move "#0P1000 #1P1500S1000 #3P2000 T1000 <cr>" is equivalent to the following (hex) bytes in an extended binary command:

Code: 80 03 E8 81 05 DC A0 03 E8 83 07 D0 A1 03 E8

Like the standard group move command, you can combine the speed and time commands if desired. The speed for each servo will be calculated according to the following rules:

1. All channels will start and end the move simultaneously.
2. If a speed is specified for a servo, it will not move any faster than the speed specified (but it might move slower if the time command requires).
3. If a time is specified for the move, then the move will take at least the amount of time specified (but might take longer if the speed command requires).

Any move that involves more than one servo is considered a Group Move, and all servos will start and stop moving at the same time. If you require moving several servos at different speeds, you must issue the commands separately.

---

## Query Pulse Width

The binary query command is 5 bytes. The servos are specified using a bitmap, with each servo being represented by a bit. The 5 bytes are:

- byte 1 = command byte, and servos 0-3
- byte 2 = servos 4-10

- byte 3 = servos 11-17
- byte 4 = servos 18-24
- byte 5 = servos 25-31

In more detail:

- byte 1 = (binary) 1 0 1 1 s3 s2 s1 s0
- byte 2 = (binary) 0 s10 s9 s8 s7 s6 s5 s4
- byte 3 = (binary) 0 s17 s16 s15 s14 s13 s12 s11
- byte 4 = (binary) 0 s24 s23 s22 s21 s20 s19 s18
- byte 5 = (binary) 0 s31 s30 s29 s28 s27 s26 s25

The numbers s0 through s31 represent servos 0-31. A 1 bit requests the value for the servo.

Byte 1 starts with bits 1011, or 0xB, which is the start of binary query commands. The other bytes all start with a 0 bit.

So, for example, to query servos 0, 3, 4, 12, 21, and 31, you would use the following bytes:

10111001 = 0xB9
00000001 = 0x01
00000010 = 0x02
00001000 = 0x08
01000000 = 0x40

serout p0,i38400,[0xB9,0x01,0x02,0x08,0x40]

The result of this command will be 12 bytes, 2 bytes per servo requested. They will be in order starting with S0 through S31. There is no prefix byte.

You can request any number of servos, and it will take 5 bytes to make the request regardless of the number of servos requested. The number of bytes returned will be 2x the number of servos requested. Servo pulse widths are returned most significant byte first.

The SSC-32 will send the servos as fast as it can, honoring the delay and pacing values that have been set.

WARNING: After sending a query command, do not send any other commands until all of the servo pulse widths have been received back from the SSC-32. If you do, this may corrupt the servo pulse widths being sent, and you might get some garbage values.

---

## Stop All Command

Hex 0xA2 (Dec 162) is the stop all command, which immediately stops all servos where they are. It does not require any additional bytes.

---

## Atom / SSC-32 Binary Test Example

```
' Atom / SSC-32 Extended Binary Command Test
'****************************************************************
'Extended binary commands.
'
'Commands 0x80-0x9F are group move servo number 0-31.Â  The next 2 bytes
'must be pulse width for the servo:
'Â Â Â  0x80+servoNum, pwHigh, pwLow
'
'Command 0xA0 is the group move servo speed, which must immediately
'follow the pulse width:
'Â Â Â  0xA0, spdHigh, spdLow
'
'Command 0xA1 is the group move time, which must follow all of the
'pulse widths and speeds:
'Â Â Â  0xA1, timeHigh, timeLow
'
'Command 0xA2 is the stop all command, 1 byte
'Â Â Â  0xA2

'****************************************************************

servonum var byte
servopw var word
servospd var word
movetime var word

'**** Move to initial position ****
'Same as #0P2000 #31P1600 T0
```

```
      '2000 = 0x07D0
      '1600 = 0x0640
      serout p0,i38400,[0x80,0x07,0xD0,0x81,0x06,0x40,0xA1,0x00,0x00]

start:

      'Same as #0P1000 #31P1400 T2000

      servonum=0
      servopw = 1000
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      servonum=31
      servopw = 1400
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      movetime = 2000
      serout p0,i38400,[0xA1,movetime.highbyte,movetime.lowbyte]
      pause 2500

      'Same as #0P2000 #31P1600 T1500
      servonum=0
      servopw = 2000
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      servonum=31
      servopw = 1600
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      movetime = 1500
      serout p0,i38400,[0xA1,movetime.highbyte,movetime.lowbyte]
      pause 2500

      'Same as #0P1000S500 #31P1400 T100
      '(Even though the move time is 100ms, this move will take 2 seconds because
      'servo0 needs to move 1000us and is limited to 500us/second.)

      servonum=0
      servopw = 1000
      servospd = 500
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      serout p0,i38400,[0xA0,servospd.highbyte,servospd.lowbyte]
      servonum=31
      servopw = 1400
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      movetime = 100
      serout p0,i38400,[0xA1,movetime.highbyte,movetime.lowbyte]
      pause 2500

      'Same as #0P2000 #31P1600 T2000
      servonum=0
      servopw = 2000
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      servonum=31
      servopw = 1600
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      movetime = 2000
      serout p0,i38400,[0xA1,movetime.highbyte,movetime.lowbyte]
      'Pause only 500ms, so the move will not be complete after the pause
      pause 500

      '**** Stop the previous move part way ****
      'Same as STOP
      serout p0,i38400,[0xA2]
      pause 2500

      '**** Finish the move ****
      'Same as #0P2000 #31P1600 T1500
      servonum=0
      servopw = 2000
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      servonum=31
      servopw = 1600
      serout p0,i38400,[0x80+servonum,servopw.highbyte,servopw.lowbyte]
      movetime = 1500
      serout p0,i38400,[0xA1,movetime.highbyte,movetime.lowbyte]
      pause 2500
      goto start
```