# SSC-32 General Purpose Sequencer

Author: Mike Dvorsky
Version: 1.00
Date: Jan 15 2008

Printable version by: Daniel Washington (dwsmartins)
Date: October 16, 2012

# Table of Contents

# EEW, EER commands: EEPROM Write, Read

The EEW and EER commands are used to write and read external EEPROM data. In order for these commands to be used, an EEPROM chip must be installed in the 8-pin socket on the SSC-32. The chip used for testing is a Microchip 24LC256-I/P which has 32768 bytes of storage. Other I2C serial EEPROMs with comparable pinout should work, but have not been tested.

The primary use of external EEPROM is to store sequences for the General Purpose sequencer. The first 256 bytes of external EEPROM are reserved exclusively for the GP sequencer, but the other bytes may be used for other purposes.

EEPROM read and write commands are issued with arguments in decimal format, but the response to an EEPROM read command is in binary format. This is in keeping with the SSC-32 convention of receiving commands in decimal format and providing responses in binary format.

NOTE: The current firmware version (SSC32-V2.01GP) requires a '-' character before the address in EEPROM read and write commands. If this character is omitted, the EEPROM access will be performed on internal EEPROM of the ATMega168 processor. Future firmware versions will not require the '-' character for external EEPROM access.

NOTE: No external EEPROM read or write accesses should be performed while a sequence is executing out of external EEPROM.

## EEW Instruction Format

EEW -<addr>,<byte>,<byte>,<byte>,...,<byte><cr>

- <addr> = the starting address of the data to be written, in decimal format
- <byte> = a byte of data to be written to external EEPROM, in decimal format

The EEW command writes up to 32 bytes of EEPROM. To write more than 32 bytes, issue multiple commands. The command may take several milliseconds to complete. When performing multiple successive EEW commands, it is recommended that you wait for each command to complete before starting the next command. There are several ways to do this:

- Delay 10ms between EEW commands.
- Use an EER command to read back the data just written. The EER command will not return until the EEW has finished.
- Issue any command that receives a response from the SSC-32. The SSC-32 will not respond to any commands until the EEW command has finished.

Example: write 8 bytes of external EEPROM starting at address 256. Write the values 12, 34, 56, 78, 90, 98, 76, 54.

EEW -256, 12, 34, 56, 78, 90, 98, 76, 54<cr>

NOTE: The EEW command will complete faster if the starting address is a multiple of 32. If you have a large block of data to write, begin by writing any unaligned portion; then write 32-byte aligned blocks; finally write any remaining unaligned portion.

# EER Instruction Format

EER <addr> ; <nbytes>

- <addr> = the starting address of the data to be written, in decimal format
- <nBytes> = the number of bytes to read, in decimal format

The EER command reads up to 32 bytes of EEPROM. To read more than 32 bytes, issue multiple commands. The return value from the EER command is a string of bytes, in binary format.

Example: read 8 bytes of external EEPROM starting at address 256. (Assume the values in EEPROM are the values programmed in the previous example.)

EER -256;8<cr>

result: 12 34 56 78 90 98 76 54 (binary)

# Memory Map:
# Sequences in External EEPROM

Sequences stored in external EEPROM must follow a defined format, as described below. The first 256 bytes are reserved for the Sequence Pointer Table, but the remaining bytes may be used for other purposes, as long as they are not part of a sequence pointed to by an entry in the Sequence Pointer Table.

## Locations 0 - 255: Sequence Pointer Table

The Sequence Pointer Table contains the addresses of the sequence starting locations in EEPROM. Valid starting addresses are 256 - 32767. If there is no sequence for a particular sequence number, the Sequence Pointer Table entry should be 0 or 65535. The addresses are stored in big-endian format, with the high byte in the lower address.

@0:1 = pointer to sequence 0 (high byte in address 0, low byte in address 1)
@2:3 = pointer to sequence 1
@4:5 = pointer to sequence 2
@6:7 = pointer to sequence 3
Etc., up to sequence 127

## Locations 256 - 32767: Sequence Data

The sequence pointer table entries point to sequence data. Sequences may be stored at any location, but the only way a sequence can be accessed is through the Sequence Pointer Table.

### *Sequence format*

Each sequence has 3 contiguous sections

- Header - containing the sequence number, number of servos, and number of steps
- Servo/speed list - containing a list of all the servos and the maximum move speed for each servo
- Time/pulse width list - containing the move time and pulse widths for each step

### *Header*

The header consists of 3 bytes

- Sequence number
- Number of servos
- Number of steps

The sequence number must match the sequence number of the Sequence Pointer Table entry. The number of servos may be from 1 to 32. The number of steps may be from 1 to 255.

## *Servo/Speed List*

The servo/speed list consists of 3 bytes per servo. The size of the servo/speed list must match the number of servos in the header.

- Servo number for 1st servo
- Speed high byte for 1st servo
- Speed low byte for 1st servo
- Servo number for 2nd servo
- Speed high byte for 2nd servo
- Speed low byte for 2nd servo
- Etc.

The servo numbers must be 0-31. Speeds are 0-65535 us/second just like move commands.

## *Time/Pulse Width List*

The Time/Pulse Width List specifies the servo positions (a.k.a. pulse widths) for each step in the sequence, and the amount of time to move from step to step. The list alternates between servo pulse widths and move times. It begins and ends with a move time.

- Step (N-1) -> 0 Time
- Step 0 PW, PW,..., PW (one PW for each servo)
- Step 0 -> 1 Time
- Step 1 PW, PW,..., PW
- Step 1 -> 2 Time
- . . .
- Step (N-1) PW, PW,..., PW
- Step (N-1) -> 0 Time

If there are M servos and N steps in a sequence, the time/pulse width list consists of 2*(M+1)*N + 2 bytes.

There are several things to note about the time/pulse width list:

1. The Step (N-1) -> 0 move time is duplicated, appearing at the beginning and the end of the list. This makes it easier for the SSC-32 to play sequences in reverse.
2. Move times are in ms, just like move commands.
3. Move times are 2 bytes each, with the high byte appearing first (i.e. big-endian).
4. The move times should be the times for a normal move speed. The speed can be adjusted faster or slower as the sequence is played. The range of adjustment is from 0 to 200% of the programmed speed, either forward or reverse.

5. Pulse widths are in us, just like move commands.
6. Pulse widths are 2 bytes each, big-endian.
7. The pulse width entries for each step must exactly match the list of servo numbers in the servo/speed list.

# General notes on programming EEPROM.

- Write no more than 32 bytes at a time. For the fastest write speed with external EEPROM, make as many writes as possible start on a 32-byte boundary. For example, to write a 1000-byte block starting at address 500, begin by writing bytes 500-511 in a single 12-byte write. Then do 30 writes of 32 bytes each--these writes will all start on 32-byte boundaries. Finally, write the last 28 bytes in a single write.
- After writing sequences to external EEPROM, it is recommended that you verify the contents of external EEPROM by reading back the bytes written.
- Bytes of external EEPROM can be used for other purposes than storing sequences, as long as no Sequence Pointer Table entries point to these bytes. For example, a portion of EEPROM could be reserved for storage of text strings describing the sequences. This way, when the sequences are read out of EEPROM, there will be some clue to what the sequence is intended to do.

# Sequence Examples

Here is a simple example of a sequence. The sequence will move 2 servos through 3 steps. The servos will be servo #9 and servo #10. The steps will be:

Step 0: Servo #9 = 1500us, servo #10 = 1500us
Step 1: Servo #9 = 1000us, servo #10 = 1500us
Step 2: Servo #9 = 1000us, servo #10 = 2000us

Moving from step 0 to step 1 will take 600ms, from step 1 to step 2 will take 1200ms, and from step 2 to step 0 will take 2400ms.

The sequence number will be 5, and it will be stored beginning at EEPROM address 500.

## Sequence Pointer Table

First, the Sequence Pointer Table must be set so that sequence 5 begins at address 500. Sequence 5 begins at EEPROM address 10. When the number 500 is split into 2 bytes, the high order byte is 1 and the low order byte is 244 (500 = 1*256 + 244), so the Sequence Pointer Table value starting at address 10 will be

@10 = 1
@11 = 244

The EEW command to write this portion of the Sequence Pointer table is
EEW -10, 1, 244

## Sequence Header

The sequence header is 3 bytes that specify the sequence number (5), number of servos (2), and number of steps (3). The sequence header begins at EEPROM address 500:

@500 = 5
@501 = 2
@502 = 3

The EEW command to write the sequence header is
EEW -500, 5, 2, 3

## Servo/Speed List

The servo/speed list has 3 bytes per servo (1 byte servo number, 2 bytes maximum speed). For this sequence, the maximum speeds will be 65535 = 255*256 + 255. The servo/speed list begins immediately after the sequence header, at address 503:

@503 = 9
@504 = 255
@505 = 255
@506 = 10

@507 = 255
@508 = 255

The EEW command to write the sequence header is
EEW -503, 9, 255, 255, 10, 255, 255

# Time/Pulse Width List

The time/pulse width list has the servo pulse widths for each step, and the times to move between the steps. The servo pulse widths in this example are

- 1000 = 3*256 + 232
- 1500 = 5*256 + 220
- 2000 = 7*256 + 208

The move times in this example are

- 600 = 2*256 + 88
- 1200 = 4*256 + 176
- 2400 = 9*256 + 96

The time/pulse width list begins immediately after the servo/speed list, at address 509:

@509 = 9 Step 2 to step 0 time = 2400
@510 = 96
@511 = 5 Step 0, servo 9 = 1500
@512 = 220
@513 = 5 Step 0, servo 10 = 1500
@514 = 220
@515 = 2 Step 0 to step 1 time = 600
@516 = 88
@517 = 3 Step 1, servo 9 = 1000
@518 = 232
@519 = 5 Step 1, servo 10 = 1500
@520 = 220
@521 = 4 Step 1 to step 2 time = 120
@522 = 176
@523 = 3 Step 2, servo 9 = 1000
@524 = 232
@525 = 7 Step 2, servo 10 = 2000
@526 = 208
@527 = 9 Step 2 to step 0 time = 2400
@528 = 96

The EEW command to write the time/pulse width list is
EEW -509, 9,96,5,220,5,220,2,88,3,232,5,220,4,176,3,232,7,208,9,96

# Sequencer Commands

These are the commands that allow you to play programmed sequences. The SSC-32 has two "players" that can each play a sequence. The players may be used simultaneously, and other servo move commands may be sent to the SSC-32 while players are playing.

NOTE: Do not simultaneously play two sequences that use some of the same servos. The results will be unpredictable.

NOTE: Do not issue servo move commands that use servos in a playing sequence. The results will be unpredictable.

## Start a player

The following command starts a player playing a specified sequence, with several parameters.

(The notation <x> indicates a number that is one of the parameters to the command. The notation "[xxx]" indicates an optional part of the command.)

- PL <p> SQ <s> [SM <m>] [IX <i>] [PA <pa>] [ONCE]
- PL <p> Specifies player p, either 0 or 1
- SQ <s> Specifies sequence number s, from 0 to 127
- SM <m> Specifies speed multiplier m, from -200 to 200
- IX <i> Specifies starting index i (a.k.a. step number), from 0 to 255
- PA <a> Specifies pause between steps, a, in milliseconds from 0 to 65535
- ONCE Specifies that the sequence is played only one time

The SM value is a percentage of the speed based on the move times in the programmed sequence. The larger the number the faster the sequence plays, up to 2 times the speed of the programmed sequence (for a speed multiplier of 200%). If the SM value is negative, the sequence is played in reverse. The SM value does NOT affect the maximum speeds for the servos. If SM is not specified, it defaults to 100%.

The IX value specifies which step should be first when the sequence is started. The player will attempt to move the servos gradually to the starting position. If IX is not specified, it defaults to 0.

The PA value specifies the length of a pause that is inserted between steps in the sequence. The value is in milliseconds. If PA is not specified, it defaults to 0.

The ONCE option specifies that the sequence is played only one time, and then the player stops. The servos will start and end at the position specified by the IX parameter (or 0 if IX is not specified). If ONCE is not specified, the sequence will repeat continuously until the player is stopped by a user command.

## Stop a player

The following command stops a player immediately.

PL <p>

# Change the speed of a player

The following command changes the speed of a player. The player must already be playing a sequence for this command to be valid.

It is possible to set the speed to 0. In this case, the servos will stop moving but the sequence will still be playing and can be restarted by changing the speed to a non-zero value.

PL <p> SM <m>

# Change the pause value of a player

The following command changes the pause value of a player. The player must already be playing a sequence for this command to be valid.

PL <p> PA <pa>

# Go to a step in a sequence

The following command causes a group of servos to move to one of the steps in a sequence. The step number and move time can be optionally specified. This command is equivalent to a normal group move command, except that the servos and maximum speeds are obtained from the specified sequence. The T parameter in this command works exactly like the T parameter in a normal group move command.

NOTE: This command is independent of any player, and may be issued even if both players are playing sequences. Like any servo move command, this command should not be used if it will cause servos to move that are part of a currently playing sequence. The results will be unpredictable.

SQ <p> [IX <i>] [T <t>]

# Query a player

The following command causes the SSC-32 to return information about the state of a player.

QPL <p>

The QPL command returns 4 bytes of binary information:

1. The sequence number being played (or 255 if no sequence is being played)
2. The index moving from in the sequence (0 through the maximum step number)
3. The index moving to in the sequence
4. The remaining time in the step, 100ms per bit (e.g. if there are 700ms remaining, the value will be 7)

# Examples of Sequencer Commands

Start player 0 playing sequence 5. Start at step 0, 100% forward speed, no pause between steps, play continuously.
PL 0 SQ 5

Change the speed of player 0 to 50% reverse speed.
PL 0 SM -50

Pause player 0 (set the speed to 0)
PL 0 SM 0

Change the speed of player 0 to 200% forward speed.
PL 0 SM 200

Change the pause between steps for player 0 to 1000ms (1 second).
PL 0 PA 1000

Stop player 0.
PL 0

Go to step 3 of sequence 20.
SQ 20 IX 3

Go to step 5 of sequence 20, taking 2000ms (2 seconds).
SQ 20 IX 5 T 2000

Start player 1 playing sequence 15. Start at step 2, 70% reverse speed, play one time only.
PL 1 SQ 15 IX 2 SM -70 ONCE