EEW, EER commands –EEPROM Write, Read.

```
EEW -<addr>,<byte>,<byte>,<byte>,…,<byte><cr>    (address and data in decimal)
EER -<addr> ; <nbytes>    (returns binary data bytes)
```

Example: write 8 bytes of external EEPROM starting at address 256, then read the bytes back.

```
EEW -256, 123, 45, 67, 89, 1, 234, 56, 78<cr>
EER -256;8<cr>
➔ 123, 45, 67, 89, 1, 234, 56, 78  (binary)
```

Restrictions on EEPROM read/write:

- No EEW or EER commands allowed while sequences are playing
- Reads and writes limited to 32 bytes at a time
- EEW commands can take several milliseconds to complete. Do not send another EEW command until the previous one has completed. The best way to determine when an EEW command has completed is to do an EER command. When the EER command response is received, the EEW command is done. (Any command that receives a response from the SSC-32 can be used for this purpose.)

Memory Map – External EEPROM

Locations 0 – 255: Sequence Pointer Table

Note : only tested with a 32kB EEPROM for now (24LC256 I/P),
but a 64kB EEPROM (24LC512 I/P) should works fine too.
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

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

Servo/Speed List
The servo/speed list consists of 3 bytes per servo.

- 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

If there are M servos and N steps in a sequence, the time/pulse width list consists of 2*(M+1)*N + 2 bytes. The list alternates between a move time and move pulse widths for all the servos. It begins and ends with 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

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. Pulse widths are in us, just like move commands.
5. Pulse widths are 2 bytes each, big-endian.
6. 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. (Alignment does not matter for internal EEPROM.)
- 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.

Sequencer Commands

PL <number>

Setup player 0 to 1.

SQ <number>

Sequence 0 to 127

SM <number>

Speed Multiplier -100 to 100 (optional, default 100)

IX <number>

Starting index 0 to 255 (optional, default 0)

QPL <number>

Query a player. Returns 4 bytes

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

PA <number>

Pause between steps (index), 0-65535 ms

Valid Combinations of Sequencer Commands

PL <p>

Stops player ‘p’ immediately.

PL <p> SQ <s>

If player ‘p’ is not playing, starts it playing sequence ‘s’ at index 0 with speed multiplier 100. If player ‘p’ is playing, changes to sequence ‘s’, starting the new sequence at the same index as the previous one.

PL <p> SM <m>

If player ‘p’ is not playing, does nothing. If player ‘p’ is playing, changes the speed multiplier to ‘m’.
NOTE: If the speed multiplier is set to 0, the servos will stop moving, but the player is technically still playing the sequence. When the speed multiplier is set to a non-zero value again the sequence will pick up where it left off.

PL <p> SQ <s> SM <m>

If player ‘p’ is not playing, starts it playing sequence ‘s’ at index 0 with speed multiplier ‘m’. If player ‘p’ is playing, changes the sequence number and speed multiplier. It starts the new sequence at the same index as the previous one.

PL <p> SQ <s> IX <i>

This combination is only valid if the sequencer is not playing. It starts playing sequence ‘s’ at index ‘i’ with speed multiplier 100.

PL <p> SQ <s> SM <m> IX <i>

This combination is only valid if the sequencer is not playing. It starts playing sequence ‘s’ at index ‘i’ with speed multiplier ‘m’.

PL <p> PA <t>

cause a ‘t’ms pause to be inserted between all steps for player ‘p’.

The ***PA <t>*** command can be added to all of the previous combinations.

PL 1 SQ 25 PA 250 will start player #1 playing sequence #25 with a 250ms pause between steps, for example.