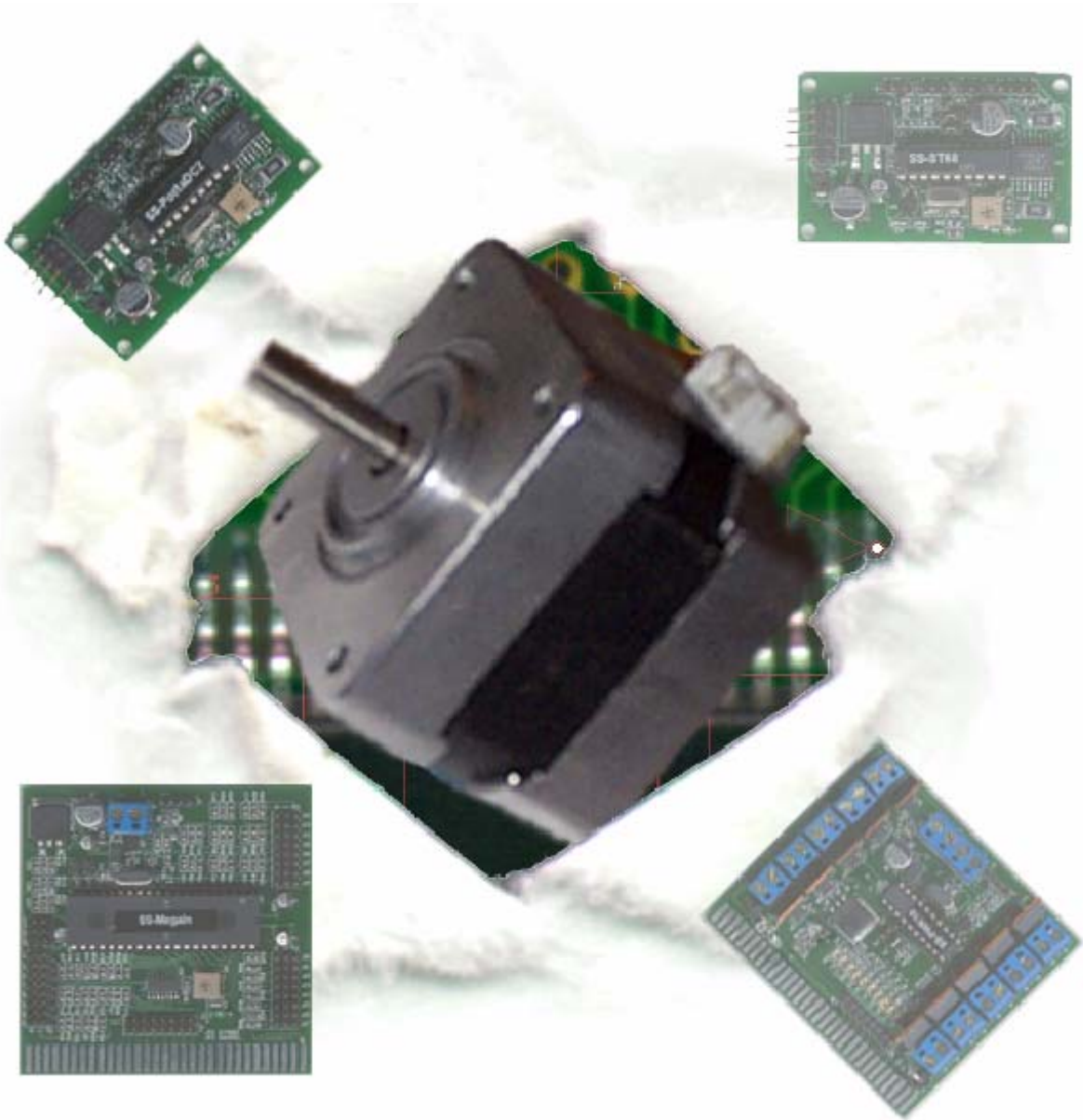


Super Stepper Architecture



Super Stepper Architecture

Introduction:

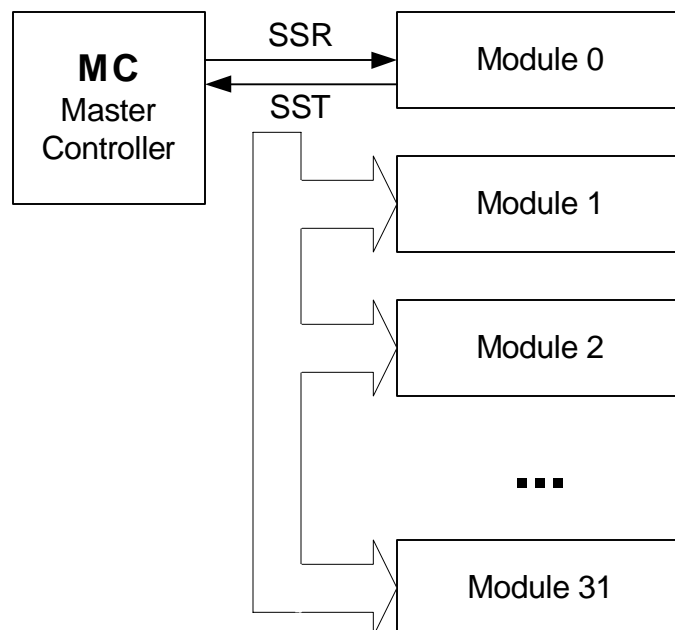
Welcome to the Super Stepper protocol and architecture, the most exciting modular control system on the market. Super Stepper is a serial command protocol capable of controlling not only stepper motors but DC motors, RC servo motors, inductive loads such as lamps, solenoids and relays and much more. Constructing robotic projects will never be easier!

What is a Super Stepper based system made of?

Any Super Stepper robot or automaton revolves around a Master Controller (MC) and up to 32 Slave units, or modules. The MC is any computer device with a conventional UART (Universal Asynchronous Receive Transmit) port. PC computer COM ports, Single Board Computer (SBC) UARTS, microcontroller UARTS or BASIC Stamp modules contain all you need to control a set of modules.

More than one module on the serial bus is called a Super Stepper Chain (SSC). This chain is connected to the MC via a three wire serial communication channel called the Super Stepper Bus (SSB). The SSB three wires are the Super Stepper Receive (SSR), Super Stepper Transmit (SST) and a ground for voltage level reference.

The SSR is the line from the MC to the module which carries Super Stepper commands. Once the module reacts to the command, it responds back with an acknowledge message through the SST line.



Super Stepper Architecture

How will each module know if it is being talked to?

The Super Stepper protocol allow multi-nodal access by means of a typical addressing scheme in which only modules with the same address react to a sent command. Modules with a different address remain dormant until a command with their own address is received.

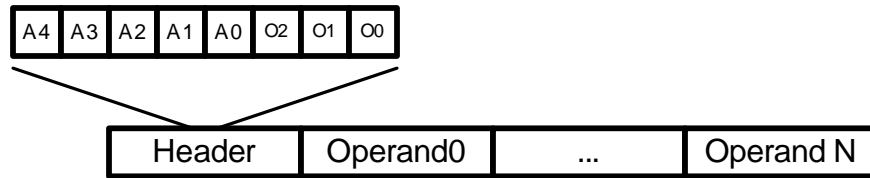
This Address is transmitted along with the command message in a special byte called the Header. The Header contains two very important pieces of information. First the address resides in the higher part of the byte and then an opcode fits the reaming portion.

The opcode is the actual command which will be executed upon entire message reception. Since a bit is 8 bits long, and the Address services up to 32 modules (taking the 5 most significant bits), the opcode is only 3 bits long. This allows for up to 8 different messages to be recognizable per module. Some fixed commands are Read from Memory, Write to Memory and Expanded commands. The other five commands are dependant on the particular SS device.

Message Length

Most serial command based protocols utilize a fixed number of bytes per message style. In other word, every command is, let say for example, 8 bytes long. Since this may be an impressive waste of real time resources, the Super Stepper protocol utilizes a non-fixed number of bytes per message format. Hence, there will be messages as short as 1 byte long. Such messages are quick to execute. Longer messages carry more information needed for proper execution.

Super Stepper Architecture



Super Stepper modules work by executing commands. These commands are transmitted into the control module from a PC computer, Single Board Computer (SBC) or any microcontroller with an UART. Communications are serial and the protocol is very easy to learn and put into practice.

The Protocol:

A message always starts with a Header Byte (HB). The HB contains the device address on the five most significant bits. The three least significant bits contain the opcode or instruction to be executed.

The address can be any number from 0 to 31. Devices with that address will receive the following bytes of the message and respond to the instruction. Devices with a different address will ignore the following bytes and wait for the next message to arrive.

The opcode can be any number from 0 to 7. SS Devices can only react to a maximum of 8 opcodes. If more functionality is needed, an expanded opcode is introduced which will include up to 256 extra opcodes as received in the Operand0 Byte.

After the Header byte is received, the device will expect N amount of operands to be sent. This is achieved thanks to the fact that the decoded opcode specifies how many bytes the device needs to receive, in order to acknowledge the entire command.

If all the bytes for a command are received, the device will positively acknowledge the message. This acknowledge includes information regarding possible frame errors happening during communication, returned data if any and a checksum. At this same time the device will commence command execution.

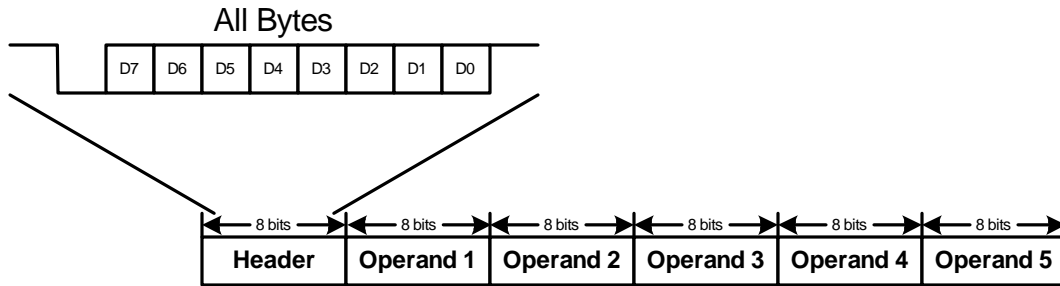
If a timeout occurs and not all the bytes have been received, the device will send a Non-Acknowledge byte indicating timeout and possible errors such as OverRuns, Noise and Frame Error.

8 bit vs 9 bit communications:

There are two data formats for which the SS devices can receive serial messages. The goal is to allow 8 bit UARTS, to interface to the device.

Avayan Electronics recommends using 8 or 9 bit communications on Uni-Node systems. Multi-Node systems should consider using 9 bit communications for improved reliability.

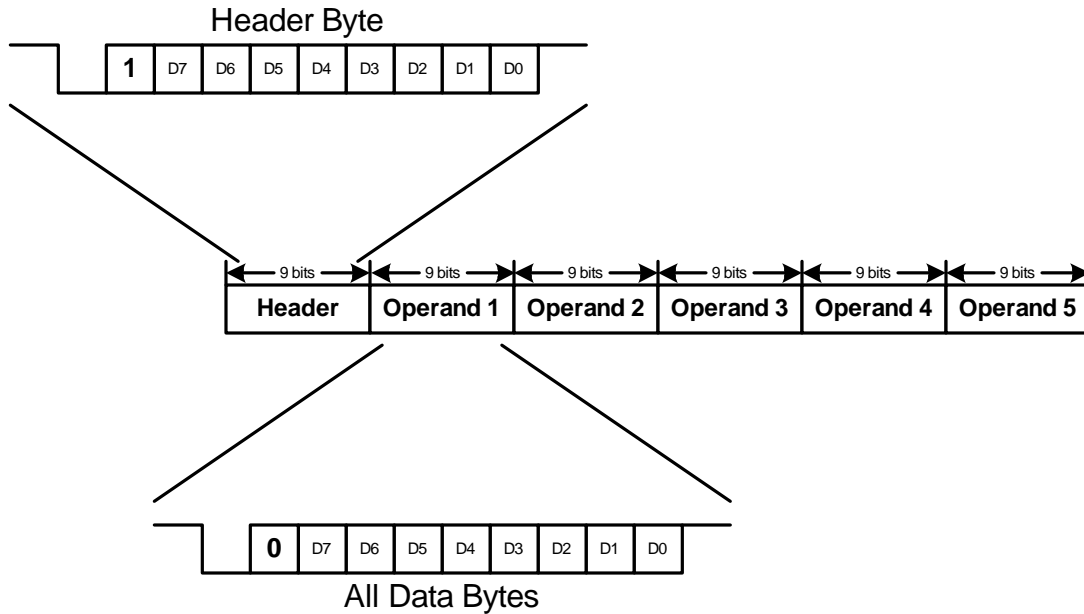
8 Bit Communication Mode



8 bit Data Format:

In the 8 bit data format, the byte is comprised of 8 data bits. Since there is no way to discriminate a header byte from an operand byte, a header byte is assumed to be any byte that reaches the receiver while on idle (no serial activity for more than 20 ms). Once the entire message is over, the communication channel must allow enough time for the receiver to timeout and go into idle mode. This measure has been put into place so that operand bytes do not get confused as Header Bytes.

9 Bit Communication Mode



9 bit Data Format:

The 9 bit data format is the most secure of both options since it incorporates true addressing. In this scenario, the byte is comprised of an address bit and 8 data bits. The address bit is 1 for a Header Byte and 0 for any Operand Byte. This makes it really easy for the receiver to discriminate between Header and Operand bytes. Although receiver time out is still available to report Non Acknowledge States, the communication channel does not need to wait time from one message to the next. This feature alone allows for faster communications.

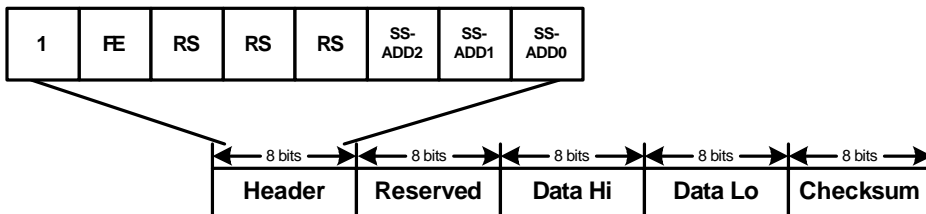
Super Stepper Architecture

Handshaking:

A very important aspect of any communication system is the response back the slave unit or node offers as a response of a received message. The Super Stepper protocol takes this detail in consideration as well in what is called an Acknowledge Message. There are two types of Acknowledge message where both are very similar. Both types of acknowledge messages consist of 5 bytes.

Positive acknowledge is sent by a module with the corresponding address, after the entire message has been received and executed. Negative Acknowledge is sent by a module with the corresponding address, after a partial message is received but a timeout has occurred. There is no message sent back if the message is sent to a non existing address module. Per example, if a Read RAM command is sent to SS-Address 6, but none of the modules in the chain have that address, silence will be observed in the SS transmit line.

Positive Acknowledge:



The positive acknowledge message signifies a completely received and totally executed command. The most significant bit is a one (for positive). Then the framing error bit specifies whether any byte on the message was received with poor synchronization (a frame error occurs when the stop bit, which is supposed to be high level or a mark symbol, is sampled to be the opposite state). The least significant bits are the responding device address.

FE (bit 6) : 1 for frame error in any received byte, 0 for no frame error received.

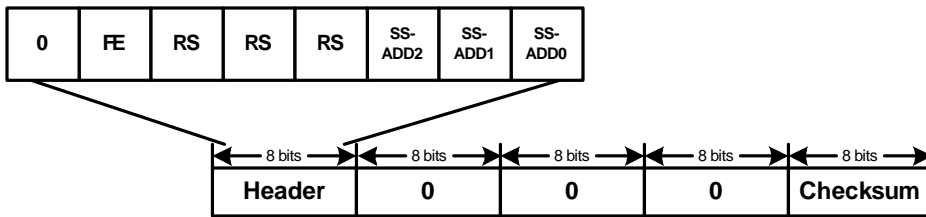
RS (bit3 to bit5) : Reserved bits

SS-ADDR (bit0 to bit2) SS Module address

The next bytes sent are data bytes. The second byte transmitted, is reserved at this time. Data Hi contains the High Side of a word read from memory as with the Read from Memory commands. Data Lo contains the Low Side of a word read with the same memory read command. Finally a checksum is sent. The checksum is computed by adding all previous four bytes. Carry bits are ignored so the resulting 8 bit number is sent as the checksum.

Super Stepper Architecture

Negative Acknowledge:



The negative acknowledge message signifies an incompletely received and of course not executed command. The most significant bit is a zero (for negative). Then the framing error bit specifies whether any byte on the message was received with poor synchronization (a frame error occurs when the stop bit, which is supposed to be high level or a mark symbol, is sampled to be the opposite state). The least significant bits are the responding device address.

FE (bit 6) : 1 for frame error in any received byte, 0 for no frame error received.

RS (bit3 to bit5) : Reserved bits

SS-ADDR (bit0 to bit2) SS Module address

The next bytes sent are all zeroes. Finally a checksum is sent. The checksum is computed by adding all previous four bytes. Carry bits are ignored so the resulting 8 bit number is sent as the checksum.

A negative acknowledge occurs when a timeout happens before all bytes are received or when than less than needed bytes are sent to the SS Module.

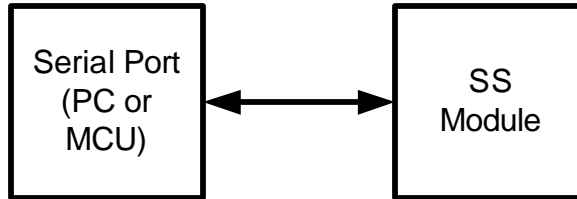
Timeout is a security measure to ensure that the SS-Module does not stay on the unwanted mode of waiting forever for a command to arrive. If this measure were not in place, an incomplete command would be completed by the next arriving bytes, which most likely were sent as part of a second command. The execution then would be totally random, generating many errors and leaving the module in the same state as before since the remaining second message bytes would be sampled as a new incoming message who was never finalized.

The SS protocol requires all bytes to be sent 20 ms one from another. If more than 20 ms elapse within two bytes, the first bytes will be discarded and the new bytes will form a totally new message.

NOTE: This problem only exists on 8 bit communications as 9 bit communications always starts with an Address Header Byte. Still, more than 20 ms from one byte to the next is considered a timeout and clears the transmission buffer.

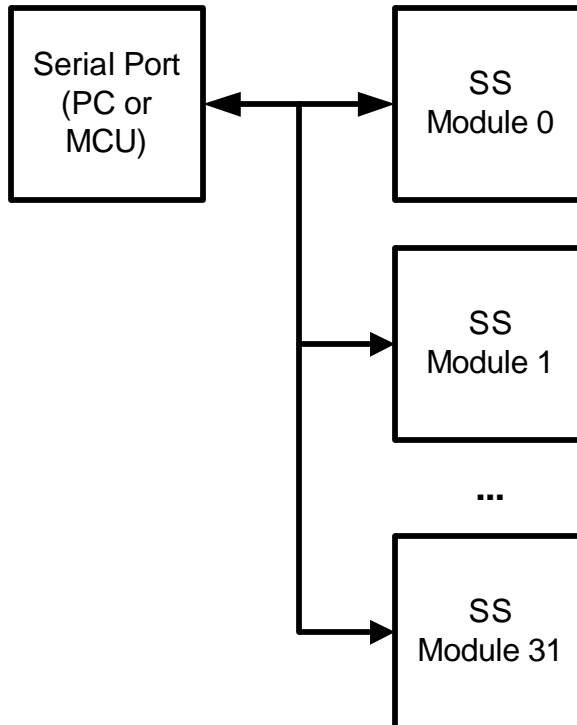
Super Stepper Architecture

Hardware Configurations:



Single Board Configuration (Uni-Node):

In this configuration, the Master Processor communicates with only one SS-Module. This mode is particularly powerful for adding control modules to control applications with simple UARTS that does not have 9 bit UART capabilities.



Multiple Board Configuration (Multi-Node):

In this configuration, the Master Processor communicates with more than one SS-Module. This mode is particularly powerful for adding many control modules to applications with a powerful UART capable of handling 9 bit communications and a powerful CPU with large memory to store plenty of commands.

Super Stepper Architecture

Mode Bits:

There are two mode hardware bits (MOD0 and MOD1 pins) that when configured allow one of four possible serial control scenarios.

8/9 bits Serial Communication (MOD0): This bit specifies whether the SS chip UART will transmit and receive data commands on 8 bit data format or 9 bit data format. Pulling this bit low, configures the SS chip to work as an 8 bit data format UART, while pulling it high configures it to work as a 9 bit data format UART.

Serial Comm Port Configure (MOD1): This bit specifies whether the SS Chip will configure the UART with default values of 2400 BAUD rate and SS-Address 0, when pulled high; or will fetch the parameters from EEPROM locations 0 (BAUD Rate) and 1 (SS-Address), when pulled low.

MOD0: Selects 8 bit (MOD0 = 0) or 9 bit (MOD 0 = 1) communications.

MOD1: Selects BAUD rate and device address. MOD1 = 0 selects 2400 BAUD rate and Device Address 0 while MOD1 = 1 selects BAUD rate and Device address programmed on EEPROM addresses 00h and 01h.

MOD1	MOD0	Description
0	0	8 bit communications with 2400 BAUD rate on address 0
0	1	9 bit communications with 2400 BAUD Rate on address 0
1	0	8 bit communications with BAUD rate and address as programmed on EEPROM
1	1	9 bit communications with BAUD rate and address as programmed on EEPROM

BAUD Rate Selection:

The Baud Rate programmed on EEPROM must meet the particular device BAUD Rate Selection table. BAUD Rates are dependant on the system clock. Some SS devices work with 4 MHz crystals, while others will use higher frequency oscillators. Once the Selection Table is found, select the Register Value and program it into the corresponding EEPROM memory location. After reset, this will become the new BAUD Rate for the device.

NOTE: All SS Devices in the bus must have the same BAUD Rate or else, modules will not be able to communicate properly.

Super Stepper Architecture

Programming BAUD Rate and SS-Address in EEPROM

To properly configure the Super Stepper Peripheral UART, there are a few easy steps to be accomplished. They only need to take place once as the board is prepared to be used on a multiple board system or if a faster BAUD rate is desired under single board configuration. Once the two parameters have been programmed into SS-peripheral EEPROM, there is no need to reprogram them again.

1. Connect the SS-peripheral to a PC serial port. Make sure the SS-peripheral is connected in a single board system.
2. Configure the board to 8 bit mode and 2400/address mode by removing Bit Mode jumpers (usually JMP1 and JMP2).
3. Apply power to the SS-peripheral.
4. Configure the PC Serial Port with 2400 BAUD Rate and 8 bit data format.
5. Send a Write EEPROM Byte command with address 0 and BAUD Rate as data. Select BAUD Rate data from table 1.
6. Send a Write EEPROM Byte command with address 1 and Address of your selection (a number from 0 to 31).
7. Remove power from SS-Peripheral.
8. Place the Bit Mode Jumpers Back depending on your options.
 - ✓ Place the 8/9 Data Format Mode Bit Jumper of you wish to communicate with the board with 9 bit data format.
 - ✓ Place the Serial Comm Port Mode Bit Jumper if you want the system to operate with programmed BAUD Rate and address.
9. Repeat process 1 to 8 for all SS-peripherals in your system or whenever a new peripheral is added.
10. Your SS-Peripherals are ready for use. Enjoy!!!

Super Stepper Architecture

Instructions:

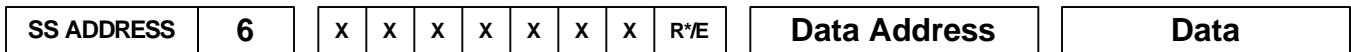
Super Stepper							
Opcode	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Description	# of Bytes
0	-	-	-	-	-	Device Dependant	Device Dependant
1	-	-	-	-	-	Device Dependant	Device Dependant
2	-	-	-	-	-	Device Dependant	Device Dependant
3	-	-	-	-	-	Device Dependant	Device Dependant
4	-	-	-	-	-	Device Dependant	Device Dependant
5	Dev Dep	Dev Dep	Dev Dep	Dev Dep	Dev Dep	Expanded	Device Dependant
6	Mem Sel	Address	Data	-	-	Write	4
7	Mem Sel	Address	-	-	-	Read	3

Most Super Stepper Protocol instructions are device dependant. That is, those instructions found on a stepper controller may not be relevant to the DC motor controller. Three instructions which you should expect to see on every Super Stepper module are the Expanded Opcode, Write to Memory and Read from Memory. They are next detailed.

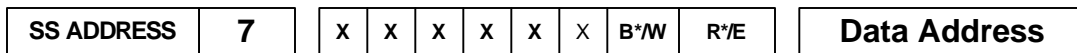
Expanded (Opcode 5): The Expanded opcode allows up to 256 extra commands to be coded per application. These 256 extra commands are called XOpCodes. The first byte after the Expanded Opcode is the XOpcode number. Any byte following will be used to parameterize the actual command.

The Expanded Opcode is always the same length in number of bytes per application device. Hence, short format commands must be filled with zeroes to meet the specified length. Not all devices utilize the Expanded Opcode. For more information on the Expanded opcode, refer to the particular device data sheet.

Write RAM/EEPROM (Opcode 6): Writes to internal RAM or EEPROM memory. The MEM SEL parameter specifies which type of memory will be accessed. Logic Low (0) accesses RAM while LogicHigh (1) accesses EEPROM. All writes are done to byte memory spaces. Writing to a word is not supported, so user must do two consecutive word writes to write to a word memory allocation, if needed.



Read RAM/EEPROM (Opcode 7): Reads from internal RAM or EEPROM memory. The MEM SEL parameter specifies which type of memory will be read and if the read request will return a word or a byte. Only reads to RAM can return a word. EEPROM reads always returns a byte.



B*/W	R*/E	Description
0	0	Read a RAM Byte from ADDRESS
0	1	Read an EEPROM Byte from ADDRESS
1	0	Read a RAM Word (16bit) from ADDRESS
1	1	Read an EEPROM Byte from ADDRESS

Super Stepper Architecture

Disclaimer:

Super Stepper is a trademark of Avayan Electronics. Protocol may not be used by third party providers either partially or totally without Avayan Electronics' consent.

The material in this data sheet reflects the technology as developed by Avayan Electronics Electrical Engineering and Research Laboratories. It may change at any time without prior notice.

Super Stepper is an architecture for motion control, automation and modular robotic design. The architecture does not comprise redundancy or error check and correction enough to make it robust against faults which may put in danger human life as in medical appliance applications, toxic waste handling control, space exploration, etc. Use of the protocol for such tasks may result in serious injury in the event of communication failure. Avayan Electronics can not be made responsible for such occurrences.