Designer Systems .co.uk
Bespoke Electronic Design & Manufacturing in the UK

RS232 serial & I²C Servo Control IC

Technical Data

DS-SCXnS
Firmware revision 2

Features

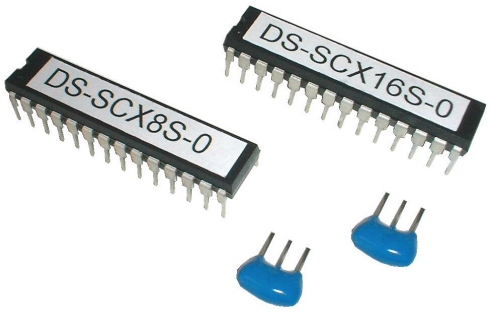
- **Simply plugs into the DS-RCB servo expansion module or can be used on it's own with the supplied 8MHz resonator.**
- **Eight (DS-SCX8S) or sixteen (DS-SCX16S) channel versions available.**
- **I²C or RS232 communication interface for simple connection to PC, OOPic, BASIC Stamp etc.**
- **Low power consumption from 5 Volt power supply.**
- **Sixteen (16) level speed control for each servo.**
- **Operate, reverse & soft-start (anti-jolt) control for each servo.**
- **RS232 Serial Control Protocol (SCP) fully supported.**
- **Movement complete and soft-start complete status for each servo.**
- **Global activation control ensures all servos start moving together.**
- **RS232 checksum register with confirmation feedback.**
- **0.50mS to 2.50mS output pulse width with 8uS accuracy.**
- **Eight (8) DS-SCXnS can be connected to the same RS232 or I²C bus allowing the control of 128 servos.**

Description

The Designer Systems DS-SCXnS Servo Control IC's (Version 2) provide a flexible control interface between a RS232 (i.e. PC) or I²C (i.e. OOPic, BASIC Stamp) compatible device and eight (DS-SCX8S) or sixteen (DS SCX16S) standard servos.

Once connected each servo can be positioned (with speed control), enabled, reversed and soft-started by simply writing a value to an internal register over the connected RS232 or I²C interface. The DS-SCXnS caters for the majority of servos by providing a wide pulse width range of 0.50mS to 2.50mS with 8uS per step accuracy. The new version 2 IC's also provide global activation of new servo position, soft-start & movement complete registers, a checksum register for RS232 control and support for the Savage Innovations Serial Control Protocol (SCP) on the RS232 interface.

Up to eight (8) DS-SCXnS IC's can be connected to the same controlling interface by specifying a version suffix when ordering. Versions run from 0 to 7 and allow a maximum of 128 servos to be controlled from one interface.



When used in conjunction with the OOPic controller and DS-RCB Servo Expansion Module the DS-SCX8S can provide a supplementary eight (8) servos that the OOPic can control without problem.

Both the DS-SCXnS IC's are supplied with an 8MHz resonator to allow the hobbyist to construct their own servo interface.

Applications

The DS-SCXnS IC's have applications in robotics, process control & sensor manipulation when used in conjunction with standard servos.
See the DS-RCB data-sheet for more information on this interface product.

INTERFACE MODULES

Selection Guide

Description	Part Number
Eight (8) Servo Control IC	DS-SCX8S-n
Sixteen (16) Servo Control IC	DS-SCX16S-n

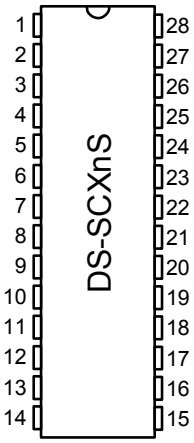
n – Version 0 to 7; specify during ordering.

Power requirements

The DS-SCXnS IC's require a regulated 5 Volt supply applied between Ground and VCC. *Warning: Misconnection of this connection may damage the DS-SCXnS.*

Pin connections

The DS-SCXnS IC's both have 28 pins in Dual In Line (DIL) format and numbered as below:



Note: the notch on the package indicates the pin 1 end.

The DS-SCX8S has the following pin connections:

Pin Description	Pin number
RESET (connect to VCC)	1
Not connected	2
Not connected	3
Not connected	4
Not connected	5
RS232 receive input 'TTL'	6
Not connected	7
Ground	8
Oscillator	9
Oscillator	10
Not connected	11
Not connected	12
Not connected	13
I2C clock (SCL)	14
I2C data (SDA)	15
Not connected	16
RS232 transmit output 'TTL'	17
Not connected	18
Ground	19
VCC (5V)	20
Servo 1 (SV9 on RCB)	21
Servo 2 (SV10 on RCB)	22
Servo 3 (SV11 on RCB)	23
Servo 4 (SV12 on RCB)	24
Servo 5 (SV13 on RCB)	25
Servo 6 (SV14 on RCB)	26
Servo 7 (SV15 on RCB)	27
Servo 8 (SV16 on RCB)	28

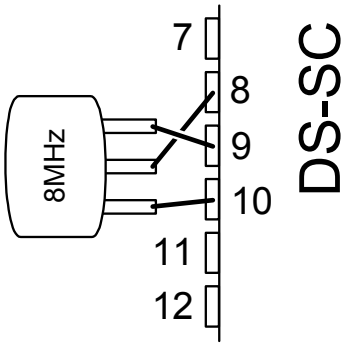
The DS-SCX16S has the following pin connections:

Pin Description	Pin number
RESET (connect to VCC)	1
Servo 9 (SV1 on RCB)	2
Servo 10 (SV2 on RCB)	3
Servo 11 (SV3 on RCB)	4
Servo 12 (SV4 on RCB)	5
RS232 receive input 'TTL'	6
Not connected	7
Ground	8
Oscillator	9
Oscillator	10
Servo 13 (SV5 on RCB)	11
Servo 14 (SV6 on RCB)	12
Servo 15 (SV7 on RCB)	13
I2C clock (SCL)	14
I2C data (SDA)	15
Servo 16 (SV8 on RCB)	16
RS232 transmit output 'TTL'	17
Not connected	18
Ground	19
VCC (5V)	20
Servo 1 (SV9 on RCB)	21
Servo 2 (SV10 on RCB)	22
Servo 3 (SV11 on RCB)	23
Servo 4 (SV12 on RCB)	24
Servo 5 (SV13 on RCB)	25
Servo 6 (SV14 on RCB)	26
Servo 7 (SV15 on RCB)	27
Servo 8 (SV16 on RCB)	28

Note: The Servo numbers referred to in brackets denote the servo connection numbers on the DS-RCB.

Resonator connection

The supplied 8MHz three terminal resonator must be connected (not required on the DS-RCB Servo Expansion Module) to pins 8, 9 and 10 in the following manner:



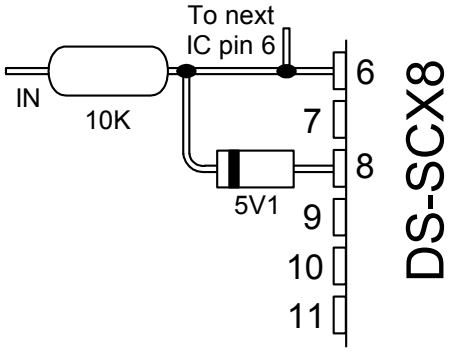
The outer two pins of the resonator may be connected to either pins 9 or 10 as the resonator is not polarity sensitive. Pin 8 must always be connected to power supply 'Ground' as must the centre resonator pin.

I^2C connection

If I^2C control is required then pins 14 and 15 must be connected to the I^2C bus and if not already present pull-up resistors of 4700R (4K7) connected from VCC to pin 14 (SCL) and from VCC to pin 15 (SDA).
Note: OOPic has these already.

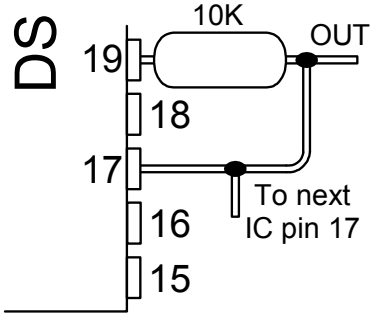
RS232 connection

If RS232 control is required then pins 6 & 17 must be connected to the controlling interface.
There are two popular forms of RS232 interfacing, standard and inverted TTL, the latter not actually being defined under the RS232 specification but being popular with the microcontroller community as no interface IC is required. The DS-SCXnS will support both, but a couple of interfacing components are required for connection to standard RS232 i.e. that available from a Personal Computer (PC) serial COM port.
The standard RS232 receive interface components are connected to pin 6 and 8 in the following manner:



A 10K (10,000 ohm) resistor and 5V1 (5.1 Volt) zener diode are used to reduce the RS232 voltage of +15 to –15 volts to TTL levels of +5 to 0.8 volts. *Warning: Damage may occur if RS232 levels are connected directly to the input pin 6.*

For both inverted TTL & standard RS232 the transmit interface is connected to pin 17 and 19 in the following manner:



Once connected to the external RS232 controlling interface additional SCX8S or SCX16S IC's (with different version numbers) may be networked together as shown without the need for further components.

Servo connection
Normal servos have three (3) connections 'Ground', 'Vsupply' & 'Control' which are normally colour coded BLACK, RED & YELLOW respectively. The servo power (4 to 7 Volts DC) is connected between 'Ground' and 'Vsupply' with the IC servo control connection (i.e. pin 21 for servo 1) being connected to 'Control'.
Tip: the most common wiring mistake is not to connect the servo 'Ground' to the IC 'Ground', so don't forget!

IC power connection
Finally a 5 Volt regulated supply with a 100nF decoupling capacitor connected close to the IC should be connected with pins 1 & 20 going to +5 Volts and pins 8 & 19 going to ground.

I²C communication

Up to eight DS-SCXnS IC's may be connected to the same I²C bus and accessed individually with their own address.
The version of the IC purchased (-n = 0 to 7) is used to define the device address i.e. if a DS-SCX8S-1 is purchased then the device address would be $1110001D_{binary}$, $E2_{hex}$.
The following table lists the IC version and the corresponding device address:

IC version	Device Address
-0	1110000D or E0
-1	1110001D or E2
-2	1110010D or E4
-3	1110011D or E6
-4	1110100D or E8
-5	1110101D or EA
-6	1110110D or EC
-7	1110111D or EE

The 'D' bit determines if a read or a write to the SCX is to be performed. If the 'D' bit is set '1' then a register read is performed or if clear '0' a register write.

To access the individual servo registers a device write must be undertaken by the OOPic / I²C Master which consists of a Start condition, device ID ('D' bit cleared), register to start write, one or more bytes of data to be written and a stop condition (see Figure 1.0 for I²C write protocol). The SCX also auto increments the register specified for every additional write requested by the Master I²C device, which allows more than one register to be written in one transaction.

Position / control registers
There are 17 or 33 individual registers (SCX8S and SCX16S respectively) that can be written within the SCX that control servo position, speed, operation, soft-start and reversal as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
SCX I2C address								
1.	1	1	1	0	X	X	X	0

XXX = SCXnS address purchased

Register address								
2.	U	U	B	B	B	B	B	B

B..B = 0 to 15 or 0 to 31, & 32
U..U = unused on this implementation

Servo 1 position								
R0	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 1 control								
R1	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 2 position								
R2	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 2 control								
R3	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 3 position								
R4	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 3 control								
R5	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 4 position								
R6	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 4 control								
R7	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 5 position								
R8	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 5 control								
R9	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 6 position								
R10	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 6 control								
R11	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 7 position								
R12	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 7 control								
R13	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 8 position								
R14	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 8 control								
R15	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 9 position								
R16	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 9 control								
R17	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 10 position								
R18	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 10 control								
R19	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 11 position								
R20	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 11 control								
R21	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 12 position								
R22	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 12 control								
R23	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 13 position								
R24	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 13 control								
R25	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 14 position								
R26	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 14 control								
R27	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 15 position								
R28	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 15 control

R29	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 16 position

R30	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 16 control

R31	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo global enable register

R32	X	X	X	X	X	X	X	X

X..X = Any value

Registers shown in *Italics* are only active on the SCX16S.

Each control register consists of four control bits and a four bit speed control value defined as follows:

Bit (A) $128_{decimal}$ is the operate bit which when set activates the servo being controlled.
Bit (B) $64_{decimal}$ is the reverse bit which reverses the position value for the servo being controlled.
Bit (C) $32_{decimal}$ is the soft-start bit which when set on servo first activation, see operate bit above, feeds position pulses to the servo in a ramping manner until position is attained.
Bit (D) $16_{decimal}$ is the speed control enable bit which when set applies the speed value 0 to 15, contained in the four bits (SSSS) $1,2,4,8_{decimal}$, to the servo being controlled.

<u>Once all the required position & control registers have been set a write to the R32 (Global enable register) must be made to activate all the new positions.</u>

Example.
To set all eight servos on a SCX8S-0 to new positions with servos 1 to 4 running at speed 0 and servos 5 to 8 running at speed 5 in reverse mode, first write:

Byte 1 (SCX Adr) 11100000_{binary}
Byte 2 (Set register) $0_{decimal}$
Byte 3 (Register 0) $30_{decimal}$
Byte 4 (Register 1) $144_{decimal}$, 90_{hex}
Byte 5 (Register 2) $35_{decimal}$
Byte 6 (Register 3) $144_{decimal}$, 90_{hex}
Byte 7 (Register 4) $40_{decimal}$
Byte 8 (Register 5) $144_{decimal}$, 90_{hex}
Byte 9 (Register 6) $45_{decimal}$
Byte 10 (Register 7) $144_{decimal}$, 90_{hex}
Byte 11 (Register 8) $127_{decimal}$
Byte 12 (Register 9) $213_{decimal}$, $D5_{hex}$
Byte 13 (Register 10) $130_{decimal}$
Byte 14 (Register 11) $213_{decimal}$, $D5_{hex}$
Byte 15 (Register 12) $140_{decimal}$
Byte 16 (Register 13) $213_{decimal}$, $D5_{hex}$
Byte 17 (Register 14) $150_{decimal}$
Byte 18 (Register 15) $213_{decimal}$, $D5_{hex}$

then to activate write:

Byte 1 (SCX Adr) 11100000_{binary}
Byte 2 (Set register) $32_{decimal}$
Byte 3 (Register 32) $0_{decimal}$

The following code can be used on the OOPic, OOPic II for the above example:

```
Dim SERVO As New oi2c

Sub Main ()
'Set the SCX8S I2C address shifted right by 1 bit
SERVO.Node = &h70     'Device address
SERVO.Width = cv8bit  'Control Info is 1-byte
SERVO.Mode = cv10bit 'I2C mode is 10-Bit
SERVO.NoInc = cvFalse 'Increment on each write
SERVO.Location = 0    'Point to register R0
SERVO = 30            'Register 0
SERVO = 144           'Register 1
SERVO = 35            'Register 2
SERVO = 144           'Register 3
SERVO = 40            'Register 4
SERVO = 144           'Register 5
SERVO = 45            'Register 6
SERVO = 144           'Register 7
SERVO = 127           'Register 8
SERVO = 213           'Register 9
SERVO = 130           'Register 10
SERVO = 213           'Register 11
SERVO = 140           'Register 12
SERVO = 213           'Register 13
SERVO = 150           'Register 14
SERVO = 213           'Register 15
'
SERVO.Location = 32   'Point to register R32
SERVO = 0             'and activate position
End Sub
```

To read the status registers a device write then read must be undertaken by the OOPic / I^2C Master. The write consists of a Start condition, device ID ('D' bit cleared), register to start read and a Stop condition. This is followed by a read, which consists of a Start condition, device ID ('D' bit set), followed by data from the status register and terminated with a Stop condition (see Figure 1.1 for I^2C read protocol).

Status registers
There are 17 registers that can be read within the SCX as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0

SCX I2C Address

1.	1	1	1	0	X	X	X	1

XXX = SCXnS address purchased

Servo 1 status

R0	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 2 status

R1	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 3 status

R2	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 4 status

R3	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 5 status

R4	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 6 status

R5	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 7 status

R6	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 8 status

R7	A	B	C	D	0	0	0	0

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 9 status

R8	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 10 status

R9	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 11 status

R10	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 12 status

R11	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 13 status

R12	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 14 status

R13	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 15 status

R14	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

Servo 16 status

R15	*A*	*B*	*C*	*D*	*0*	*0*	*0*	*0*

A = Operation (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start (0 – Complete 1 – In-progress)
D = Movement (0 – Complete 1 – In-progress)

DS-SCXnS Status

R16	U	U	U	U	V	V	V	V

V..V = Firmware revision number 1-15

Registers shown in *Italics* are only active on the SCX16S.

Bit (A) $128_{decimal}$ is the operate bit which when set indicates that the servo is operational.
Bit (B) $64_{decimal}$ is the reverse bit which when set indicates that position values written to the servo will be reversed.
Bit (C) $32_{decimal}$ is the soft-start bit which when set indicates that soft-start is in progress.

Movement complete determination
Bit (D) $16_{decimal}$ is cleared to indicate if the current servo movement has completed.
This indication is not derived from mechanical or electrical feedback from the servo being controlled but is a function of the current servo speed selected and position.
When the slowest servo speed (0) is selected the determination of movement completion is at its best. This is because the positional change of the servo between its current and final position has been split into many sub-positions which must be attained before the final position is reached. These many sub-positions ensure that the mechanical position closely relates to the position requested by the pulse width and therefore the determination of final position (movement complete) will closely relate to mechanical position. As servo speed is increased the error between mechanical position and pulse width position increases and movement completion accuracy is degraded.

RS232 communication

The serial (RS232) protocol used for communication with the SCXnS is as follows:

9600 baud (bps)
8 Data bits
1 Stop Bit
No Parity
No handshaking (if configurable)

Up to eight DS-SCXnS IC's may be connected to the same RS232 port and accessed individually with their own address. The version of the IC purchased (-n = 0 to 7) is used to define the device address i.e. if a DS-SCX8S-1 is purchased then the device address would be $1110001D_{binary}$, $E2_{hex}$.
The following table lists the IC version and the corresponding device address:

IC version	Device Address
-0	1110000D or E0
-1	1110001D or E2
-2	1110010D or E4
-3	1110011D or E6
-4	1110100D or E8
-5	1110101D or EA
-6	1110110D or EC
-7	1110111D or EE

The 'D' bit determines if a read or a write to the SCX is to be performed. If the 'D' bit is clear '0' a register write is performed, a register read is not supported.

To access individual servos a device write must be undertaken by the RS232 device which consists of a Prefix character, device ID ('D' bit cleared), register to start write, one or more bytes of data to be written and two terminator characters.

There are 17 or 33 individual registers (SCX8S and SCX16S respectively) that can be written within the SCX that control servo position, speed, operation, soft-start and reversal as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0

RS232 command prefix

1.	0	1	0	1	1	0	1	1

ASCII character ' ['

SCX RS232 address

2.	1	1	1	0	X	X	X	0

XXX = SCXnS address purchased

Register address

3.	U	U	B	B	B	B	B	B

B..B = 0 to 15 or 0 to 31, & 32, 33
U..U = unused on this implementation

Servo 1 position

R0	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 1 control

R1	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 2 position

R2	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 2 control

R3	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 3 position

R4	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 3 control

R5	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 4 position

R6	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 4 control

R7	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 5 position

R8	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 5 control

R9	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 6 position

R10	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 6 control

R11	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 7 position

R12	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 7 control

R13	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 8 position

R14	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 8 control

R15	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 9 position

R16	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 9 control

R17	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 10 position

R18	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 10 control

R19	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 11 position

R20	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 11 control

R21	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 12 position

R22	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 12 control

R23	*A*	*B*	*C*	*D*	*S*	*S*	*S*	*S*

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 13 position

R24	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 13 control

R25	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 14 position

R26	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 14 control

R27	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 15 position

R28	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 15 control

R29	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Servo 16 position

R30	P	P	P	P	P	P	P	P

P..P = Servo position 0 to 255 (0.5mS to 2.50mS)

Servo 16 control

R31	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

Internal checksum register

R32	C	C	C	C	C	C	C	C

C..C = XOR value of all values received so far.

RS232 command terminator 1

n.	0	1	0	1	1	1	0	1

ASCII character '] '

RS232 command terminator 2

n.	0	1	0	1	1	1	0	1

ASCII character '] '

Registers shown in *Italics* are only active on the SCX16S.

On receipt of the last terminator character '] ' the new positions are activated.

Example:
To set all eight servos on a SCX8S-0 to new positions with servos 1 to 4 running at speed 0 and servos 5 to 8 running at speed 5 in reverse mode, write:

[#224 #0 #30 #144 #35 #144 #40 #144 #45 #144 #127 #213 #130 #213 #140 #213 #150 #213]]

Where #nn is a decimal value, not ASCII character i.e. #40 is actually a single ASCII character '('

Checksum register (optional)
During receipt of new position / control values from the external RS232 device an internal checksum is kept that can be used to determine if the values written to the servo control & position registers is correct. This register is cleared upon receipt of the RS232 prefix character '[' and then Exclusive-OR^{ed} (XOR^{ed}) with each value received e.g. decimal values 224 $_{XOR}$ 0 $_{XOR}$ 30 $_{XOR}$ 144 $_{XOR}$ 35 $_{XOR}$ 144 would equal 221, until register 31 is reached. The next value received, destined for register 32, from the RS232 device must then be a checksum value calculated by the RS232 device which is compared with the internal value calculated by the SCXnS. If the two values match then the new control/position values are activated upon receipt of the last terminator character '], if they do not match then the new values are ignored and the following response is sent back to the RS232 device:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0

RS232 command prefix

1.	0	1	0	1	1	0	1	1

ASCII character ' ['

SCX RS232 address

2.	1	1	1	0	X	X	X	0

XXX = SCXnS address purchased

SCX Status register

3.	I	H	G	F	E	D	B	A

A = Checksum (0 – No error 1 – Error)

RS232 command terminator

4.	0	1	0	1	1	1	0	1

ASCII character '] '

Note: If the SCXnS is fitted to the DS-RCB V1.00 PCB then the above response is not possible due to hardware limitations. A V1.00 PCB modification may be undertaken to support RS232 responses, see document 'RCB V1.00 PCB modification for SCXnS V2 IC' for more details.

Example:
To set all eight servos on a SCX8S-0 to new positions with servos 1 to 4 running at speed 0 and servos 5 to 8 running at speed 5 in reverse mode, with checksum, write:

[#224 #0 #30 #144 #35 #144 #40 #144 #45 #144 #127 #213 #130 #213 #140 #213 #150 #213 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #0 #63]]

Where #nn is a decimal value, not ASCII character i.e. #40 is actually a single ASCII character '('. #63 is checksum.

Note that registers R16-R31 must be written, even on the SCX8S, so that register 32 can be accessed.

SCP communication

The Savage Innovations SCP allows a remote PC, Pocket PC, Palm Pilot, or any other device with a serial port to control the SCXnS. The serial protocol is as follows:

9600 baud (bps), 8 Data bits
1 Stop Bit, No Parity
No handshaking (if configurable)

Up to eight DS-SCXnS IC's may be connected to the same RS232 port and accessed individually with their address. The version of the IC purchased (-n = 0 to 7) is used to define the node address i.e. if a DS-SCX8S-1 is purchased then the node address would be 1.
The entire character set used by SCP is composed of human readable characters so that a serial terminal program can be used to manually control servo positions and read status flags. The following commands are only briefly described as the full SCP is not within the scope of this data sheet, a full explanation being available from Savage Innovations.

To enable SCP:

Send "**0V**"	replace 0 with node address 0 - 7
Receive "v"	indicates SCX is functional.

To set Memory type:

Send "128H"	128 + number of registers to access 0 = 1, 1=2
Receive "h"	confirms set.

To set register location to start access:

Send "15J"	15 is register number 0 to 31.
Receive "j"	confirms set.

To write to previously selected register:

Send "80N"	80 is sample hexadecimal value to write to register (must be in two character notation)
Receive "n"	confirms write.

To read previously selected register:

Send "M"	Request register read.
Receive "80m"	80 is sample hexadecimal value.

To read register location:

Send "I"	Request register location.
Receive "16i"	16 is current register location.

To read memory type:

Send "G"	Request memory type.
Receive "128g"	128 is current memory type.

To reset SCXnS:

Send "W"	SCXnS is reset.

No response generated.

Use the I^{2}C register set for SCP.

To query SCP buffer:

Send "Q" Request SCP buffer contents.

Receive "124q" Characters in buffer are returned e.g. 124 followed by "q". This command does not effect the buffers contents.

To disable SCP:

Send "X" Request exit from SCP.

Receive "x" SCP has exited.

If command format or value is not correct then a "!" response will be received and the command will not be executed.

Example:

To enable SCP, setup memory type register, location register, values to set servo 1 to mid-position with soft-start and enable new position:

Send "\0V"	Enable SCP.
Receive "v"	SCP enabled.
Send "129H"	Set memory type to 128 + 1=129 (2 registers to be accessed).
Receive "h"	Command confirmed.
Send "0J"	Set register location to 0 to allow access to R0.
Receive "j"	Command confirmed.
Send "7fa0N"	Set servo 1 position to mid-point $7f_{hex}$ 127 $_{decimal}$ and enable servo with soft-start $a0_{hex}$ $160d_{ecimal}$.
Receive "n"	Command confirmed.
Send "128H"	Set memory type to 128 + 0=128 (1 register to be accessed).
Receive "h"	Command confirmed.
Send "32J"	Set register location to 32 to allow access to R32.
Receive "j"	Command confirmed.
Send "00N"	Global enable all new servo positions.
Receive "n"	Command confirmed.

Electrical Characteristics ($T_A = 25^oC$ Typical)

Parameter	Minimum	Maximum	Units	Notes
Supply Voltage (+5V) (Vcc)	4.5	5.5	V	1
Supply Current	2	5	mA	
Operating frequency	-	8	MHz	
RS232 data input level (TTL)	0.8	VCC + 0.2	V	
RS232 speed	-	9600	bps	
I^2C speed	-	400	kHz	
Servo output pulse duration	500	2500	uS	
Servo output pulse repetition	-	20	mS	
Servo output pulse resolution	-	8	uS	
Servo speed to position	1	16	Ipr	2
Servo soft-start speed	-	3	Seconds	

Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units
Supply Voltage (+5V)	-0.5	+6	V

Environmental

Parameter	Minimum	Maximum	Units
Operating Temperature	0	70	**^oC**
Storage Temperature	-10	80	**^oC**
Humidity	0	80	**%**
Dimensions	Standard 28pin DIL IC (0.3" width)		
Immunity & emissions	EMC compliance to 89/336/EEC		

Notes:

1. Vcc is supply rail from DS-RCB or any other +5V supply.
2. 'Ipr' is Increment per repetition and indicates the value added per pulse repetition to the last position to get to the new position.

WEEE Consumer Notice

This product is subject to Directive 2002/96/EC of the European Parliament and the Council of the European Union on Waste of Electrical and Electronic Equipment (WEEE) and, in jurisdictions adopting that Directive, is marked as being put on the market after August 13, 2005, and should not be disposed of as unsorted municipal/public waste. Please utilise your local WEEE collection facilities in the disposition and otherwise observe all applicable requirements. For further information on the requirements regarding the disposition of this product in other languages please visit www.designersystems.co.uk

RoHS Compliance

This product complies with Directive 2002/95/EC of the European Parliament and the Council of the European Union on the Restriction of Hazardous Substances (RoHS) which prohibits the use of various heavy metals (lead, mercury, cadmium, and hexavalent chromium), polybrominated biphenyls (PBB) and polybrominated diphenyl ethers (PBDE).

Calculating binary bit values:

The registers used above use the binary notation to allow the control of servo operation, reversal, soft-start & speed selection. Each register is made up of eight (8) bits, which can be set or cleared to produce the desired operation, the individual bits having a value associated with them as follows:

128	64	32	16	8	4	2	1

If we take for example one of the servo control registers we can see it is made up of four (4) separate bits A, B, C & D plus a four bit value SSSS:

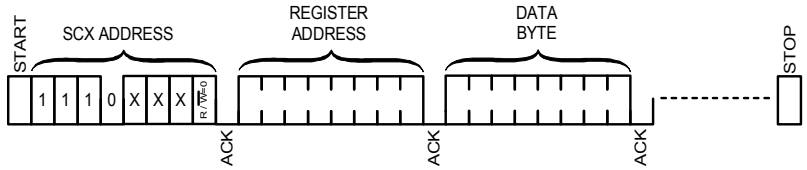
Servo 1 control

R1	A	B	C	D	S	S	S	S

A = Operate (0 – Servo disabled 1 – Servo enabled)
B = Reverse (0 – Servo normal 1 – Servo reversed)
C = Soft-start control (0 – Disabled 1 – Enabled)
D = Speed control (0 – Disabled 1 – Enabled)
S..S = Servo speed value 0 to 15 (0 = slowest)

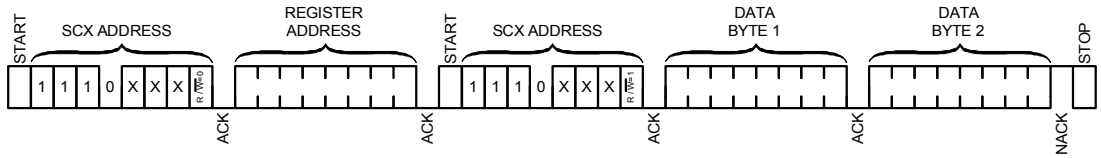
Each bit is defined to control a particular function for the servo it controls, so if for example we wanted to enable servo 1 we would need to set bit 'A' which controls the servo operation. We know from the bit values defined above that the value associated with the 'A' bit is 128, so by writing this value to register 1 we can enable servo 1. If we need to enable additional functions such as the speed control - 'D' - as well as the servo enable, the value of this bit is added to the value written to the register i.e. 128 + 16 = 144. In addition we could also add a speed value of 5 that would make the total value 128 + 16 + 5 = 149.

Figure 1.0 (I^2C write protocol)



Multiple bytes may be written before the 'STOP' condition. Data is written into registers starting at 'REGISTER ADDRESS', then 'REGISTER ADDRESS' +1, then 'REGISTER ADDRESS' +2 etc.
Each byte transfer is acknowledged 'ACK' by the SCX until the 'STOP' condition.

Figure 1.1 (I^2C read protocol)



'DATA BYTE 1 & 2' are register values returned from the SCX. Each byte written is acknowledged 'ACK' by the SCX , every byte read is acknowledged 'ACK' by the I^2C Master. A Not-acknowledge 'NACK' condition is generated by the I^2C Master when it has finished reading.

Declaration of Conformity
Apparatus name / model number DS-SCXnS
Conformity via Generic Standard EN50081-1
Generic Standard EN50082-1
Conformity criteria For use only within commercial, residential and light industrial applications
We certify that the apparatus identified above conforms to the requirements of Council Directive 89/336/EEC & 73/23/EEC
Signed. **Date** 1/6/02

Having made this declaration the CE mark is affixed to this product, its packaging, manual or warranty.

Manufacturer Designer Systems, 15 Andrew Place, Truro, Cornwall TR1 3HZ, United Kingdom
Description of apparatus Robotic interface peripheral