Scon SB Products Description

Scon SB products are small microcontroller based PC boards specifically designed to drive RC type servos for robotic applications. A SconScript language interpreter is built into the Scon SB's microcontroller. Scon SBs contain a generous flash memory space for program storage; various inputs and additional outputs for non-servo devices. For programming, setup and download, these products communicate with PCs or other devices using standard RS-232 serial ports. To keep the size small, the serial connector may be a special type requiring a cable or adaptor to connect to the PC. Some devices contain Scon's LCP (Local Com Port), a single wire serial port that supports on-robot network operation to support more that eight servos.

Scon SB products are designed to be powered by batteries and require as little as 5.5 volts DC. This allows the use of the same battery supply for both servos and the controller. Some SB products may provide regulated 5volt power for the connected servos. For many applications, servo power may exceed the on-board regulator's heat sink or current limitations. For this reason most Scon SB products have separate power input to supply power for the connected servos.

The servo connectors are standard three pin with .1 inch spacing arranged with the ground (common) on the outside of the PC board; center is the power pin; and the pulse pin on the inside. On most Scon SBs the signal lead has a series resistor to protect the microcontroller in case of reverse connection or short circuit. SB products generate standard RC PWM servo pulses of 5 volts and a pulse width of between 0 and 6.5 milliseconds, specified in .1 microsecond units, accuracy is .5 microseconds.

Non-servo inputs and outputs are addressed starting with input 1 and output 1. Devices may have none or several inputs and outputs. Inputs on a three input Scon SB product will be inputs 1, 2 and 3. Scon language supports 254 inputs and 254 outputs.

Scon SB setup and downloading of programs is done using PScon, a Windows XP based program. PScon allows control of the Scon SB while it's running, including enabling motors, setting speed and position data.

SconScript programs are written with a text editor. PScon is then used to convert the text, and download it to the Scon SB board.

PScon also has other tools. The Position Manager allows live servo movement to facilitate creating positions and motions. Once all servos are at the desired rotations, simply click to store all the rotations in a Scon position memory element. After that, anytime that element is referenced with a move, all servos will rotate to those rotations. The Position Manager also allows stepping forward or backwards through position elements moving all or some servos to the set rotations with just a mouse click. This makes building movements easy. The speed editor allows viewing and changing speed elements. For the advanced user, the memory editor allows access to the first 256 bytes of the processor ram and/or EEPROM.

Once programs are loaded, Scon SBs can be controlled by one of four methods: PC control, onboard pushbutton control, or input pin activation. Additionally, and any device that contains a serial port can control various run oriented functions. This includes calling subroutines, allowing stamp like devices to control Scon very effectively.

The SconScript Language

SconScript is a sequential interpreter programming language that is similar to basic, but is specific to controlling motion in robotics using microcontrollers. The basic-like language is quick to learn and very effective. SconScript incorporates a very powerful multi-servo position and rate (speed) control system. The language supports continuous motion, such as walking or rolling, as well as movements like arm positioning, grip and one-time motions. The language accepts input conditions and supports output control for on/off devices.

Programs are written and edited using any standard text-file editor such as notepad. Once finished, the programs are packed, converted and downloaded into flash memory in the microcontroller hardware using a simple PScon utility.

The converter/down-loader will remove all spaces, tabs, commas, and anything after a comment character (/) is encountered. After this, the instruction is converted to its hex equivalent. Any data is added and the program is stored in a temporary file until downloaded. The Scon converter creates a list file for review of the program along with details of errors. All of this is done in one step using PScon.

SconScript can reside on many different types of microcontrollers and can be used to control many types of motors in various applications. This manual refers primarily to SconScript when used on Scon SB products.

SconScript programs consist of lines of program code starting at 0 and limited by memory space. Each line requires one memory “element”, typically element numbers range between 0 and 32,767. An element is a 16-byte block of memory that can contain an instruction, position information or other data. All elements are the same size and form on most hardware. SconScript normally starts executing at line zero (element 0); when finished with the function in element zero, Scon pulls line one from memory and executes it; then line two and so on. This functioning is done on the SconScript processor hardware; not on the PC, so the robot can independently of PC control.

The SconScript programming concept can be described as a “move-to-position” system. Servo rotation angles for all servos are stored in memory elements (position elements). Each element can store the rotation information for eight servos. Once the position elements are set, the program moves the servos by referencing different position elements with the “move” instruction.

The user includes a number with the move instruction specifying the position memory element to be used; the referenced element contains pulse width specifications for all of the servos. There are many thousands of elements and possible rotation combinations. Here is an example with a robot arm having several servos: Each of the robot arm’s servos must be at a specific rotation for the arm to be in any specific position. One element could specify the arm’s home position this home position could be element 10,000 for example. Element 10,000 would contain the desired rotations for up to eight servos. The arm’s extended position could be defined in element 10,001. Each time the instruction Move 10,000 is encountered, the arm will go to the home position. Move 10,001 will cause the arm to move to the extended position. The arm could use several servos as joints or just a few. The position elements can be placed anywhere in memory and in any order. This allows for good memory usage because information for eight servos for any one

specific position need only be kept once, but can be accessed as often as needed. It is also an easy way to think about the movement of the arm because several points can be defined with movements between the points using the program.

For a walking robot or other continuous motion, a step can be built of several positions with SconScript causing movement between the positions to create the walking motion. These motions can be kept and referenced as subroutines. Do-loops can be used to specify a number of steps or movements.

Rolling robots are easily made using SconScript in combination with servos that have been modified for continuous motion. Once the servo is modified for continuous motion Scon's highly accurate pulse provides speed control of the modified servo. Once a position element is referenced, setting a servo in rotating motion, the servo motion will not change until a different value is sent, even if other position elements are referenced relating to other movements. Refer to the example papers for further information.

The move instruction is the primary method of movement control. SconScript also executes many other instructions such as goto, gosub, do, loop, call, and wait. For certain functions, SconScript supports pre-defined variables.

Controlling Rotation Rate (speed):

The speed that a servo driven device moves is determined by the rate at which the servo's pulse width changes. SconScript controls this rate by changing the pulse width in pre-determined time periods resulting in a uniform rotation rate. The rate can be specified by one of three methods as listed below:

Synchronized Rate: Scon calculates the move time based on the rate specified with the move instruction and the servo with longest move, then applies rates to each servo so that all servos complete the move at the same time. This is done automatically; the user simply specifies the desired rate.

Dynamic Synchronized Rate (DSR): The default method. Scon calculates the move in the same way it does with Synchronized rate, except that it is based on a rate stored in a variable not in the instruction. When Scon is initialized, the variables (ram locations) are loaded with the Default Synchronized Rate; this rate may be changed under program control by changing the values of the DSR variables. There are three DSR's; DSR1, DSR2, & DSR3 that can be specified in the move instruction. If no rate is specified in the move, the default variable DSR1 will be used. These rate controls allow the user's program to control the speed of the robotic system while it is running.

Table Rate: Rate calculations require processor time. In the case that time between instructions must be very short or rate control for individual servos is desired, servo travel rates may be pulled from an element. With this method, the user must specify a rate for each servo and no calculations are required so the move calculations require less time. This method may also be used in cases that it is desirable to specify a specific rate for each servo for other reasons.

Refer to the Controlling Speed and Rate document for further information.

The SconScript Instruction Set

The PScon converter will recognize and read in a standard text file using the instructions defined below. Scon SB products will execute some or all of the instructions depending on the product. There are two basic types of instructions: Loader instructions and executer instructions. Loader instructions direct the converter / down-loader while executer instructions are loaded onto the Scon product for execution. All labels are loader instructions except those specifically pre-defined as variables.

Executer instructions:

Goto — Causes the Scon program interpreter to start reading instructions from the line number immediately following the "Goto" key word.
Example: Goto 125

GoSub — Causes the program interpreter to start reading instructions from the line number immediately following the call key word. The current address is stored on a memory stack so that the interpreter can return back. 32 nested calls are allowed on Scon 017. Refer to the Return instruction.
Example: GoSub 125

Call — Call a machine code subroutine (future instruction)

Return — Pulls the address from the return address stack and causes the program interpreter to start reading instructions from that address. Refer to the "Gosub" instruction.
Example: Return

Do — Stores the number immediately following the "Do" key word. The current address is stored on a memory stack, so the interpreter can return back and repeat the instructions. 32 nested Do-Loops are allowed on Scon 018. Refer to Loop instruction.
Example: Do 10

Loop — Pulls the loop count from the loop stack and decrements it. If the result is 0, the program continues; if it's greater than 0 the interpreter pulls the associated "Do" address and returns to repeat the processes. Refer to the Do instruction.
Example: Loop

Move — The Move instruction pulls the servo rotations from the specified element and causes the servos to rotate to those angles. If no rate (speed) is specified the default synchronized rate will be used.
Example 1: Move 10,000
Example 2: Move 10,000 Rate=100
Example 3: Move 10,000 Rate=DSR2

Poke — Future instructions

Peek — Future instructions

Output — The Output instruction is used to turn a specific output on or off. The Output keyword is followed by the output number, then on or off.
Example 1: Output 1 on
Example 2: Output 1 off
Example 3: Output 15 off Note: Output 15 is the on board LED on Scon SB 018
Example 4: Output 15 on Note: Output 15 is the on board LED on Scon SB 018

If input — The "If input" instruction compares the selected input's value to the actual value and if true executes the specified instruction. Goto, Gosub, and Stop are allowed instructions. If the result was false, the instruction is skipped. Compound if is not allowed, for example: If input 1=1 If input 2=1 is an invalid statement.
Example 1: If input 1=1 Call 250
Example 2: If input 1=0 Goto 260
Example 3: If input 2=1 Stop

Wait — Causes the program interpreter to pause for the time period indicated in the number immediately following the Wait key word.
Example: Wait 100

Nop — No operation (Nop or No operation). The interpreter skips this instruction
Example 1: No operation
Example 2: Nop

Stop — Causes the program interpreter to stop reading and executing instructions. It does not affect outputs.
Example: Stop

Marker — Causes a text string to be placed in a Scon memory element limited to 15 bytes
Example: Marker Here I am

XXX= — Set variable absolute. See variables description for more information.
XXX+ — Add value to variable. See variables description for more information.
XXX- — Subtract a value from variable. See variables description for more information.

Loader Instructions:

/ — Comment anything on a line after the / character is ignored.
Example: /This is a comment

Address — Tells the loader where to start loading the program. If not specified, the loader will begin placing the program at element 0.
Example: Address 10,000

Position — Sets position memory (see the S1-S7 specifiers below)
Example: Position S1=15000 S2=15000 S3=15000 S8=15000

Speed — Sets speed memory (see the S1-S7 specifiers below) Note: Speed range limit for Scon 017 is 0 – 1023.

Example: Position S1=50 S2=50 S3=50 S8=50

S1-S7 Specifies a servo number within position or speed specifications. Used to specify rotation pulse width or speed for servos in element memory.

Example 1: Position S1=15000 S2=12500 S4=21000 specifies a 1.5 ms pulse for Servo 1 a 1.250 ms pulse for servo 2 and a 2.1 ms pulse for servo 4 in the position element.

Example 2: Speed S1=50 S2=10 Specifies 50 as the rotation rate for Servo 1 and 10 as the rotation rate for Servo 2.

Variables list:

DSR1 Dynamic Synchronized Rate #1. Sets the speed for moves that use DSR1
DSR2 Dynamic Synchronized Rate #2. Sets the speed for moves that use DSR2
DSR3 Dynamic Synchronized Rate #3. Sets the speed for moves that use DSR3

Example 1: DSR1=100
Example 2: DSR1=75
Example 3: DSR2=250
Example 4: DSR3=1,000
Example 5: DSR2+500
Example 6: DSR3-1,725

S1= Servo 1 absolute value
S2= Servo 2 absolute value
….
S8= Servo 8 absolute value

Example 1: S1=1,500 Set Servo 1 to 1.5 ms
Example 1: S3=2,255 Set Servo 2 to 2.255 ms

S1+ Add to Servo 1 absolute value
S2+ Add to Servo 2 absolute value
….
S8+ Add to Servo 8 absolute value

Example 1: S1+100 Add 100 to the absolute value of Servo 1
Servo 1 is at position 1,500 before instruction
Servo 1 is at position 1,600 after instruction

Example 2: S3+225 Add 225 to the absolute value of Servo 1

S1- Subtract from Servo 1 absolute value
S2- Subtract from Servo 2 absolute value
….
S8- Subtract from Servo 8 absolute value

Example 1: S1-100 Subtract 100 from the absolute value of Servo 1

Servo 1 is at position 1,500 before instruction
Servo 1 is at position 1,400 after instruction

V1 User variable (double precision)
V2 User variable (double precision)
….
V9 User variable (double precision)

Reserved Keywords: Address, Call, Do, DSR1, DSR2, DSR3, Goto, Gosub, If, If input, Ifinput, Loop, Marker, Move, Nop, Peek, Poke, Position, Return, S1, S2, S3, S4, S5, S6, S7, S8, Speed, Stop, Wait, Slave, V1, V2, V3,V4, V5,V6, V7,V8, V9

Using PScon

PScon is a Windows XP compatible program that controls and downloads programs into a Scon SB product.

Setup of PScon:

PScon is the PC compatible program that controls Scon products. PScon runs under Windows XP and requires one serial port. Quality USB to serial converters may be used.

To setup PScon, run the install.exe file provided on the install disc, USB stick, or download. Refer to “Install Issues” at the end of this manual for Trouble Shooting.

If a Com port error occurs when PScon runs, refer to “Com Port Issues” in the “Trouble Shooting” section.

Once the serial port is functioning, verify that the Scon product is properly working by Selecting “Send Command” from the Setup and Tools drop down menu. Click the Request Version button and Scon will report back the version number. This verifies proper communication.

The default file locations are set in the System Setup screen from the Setup and Tools dropdown. Default locations are all set to C:\Irunner a required directory that contains the parameters. The default Scon and temporary file locations can be changed if desired.

Board Setup:

Select “Board Setup” from the Setup and Tools dropdown. The parameters are read when the display loads; clicking “Get Parameters” will re-load the parameters.

First, check the box for each servo that is to be used. All servos will receive a pulse and may be directly controlled; however, the Scon engines will only update and move the servos that are checked as “Active Servos”. If a servo is selected and not present, movement information will be generated for that servo. If a value is read for a servo that is not present it may appear that the board is not working when it is actually generating data for the servo that is not there. This could take over a minute to complete. This is why it is best to un-check the non-present servos in the “Active Servos” box.

“Report Controls” should be on in most cases. This causes the Scon board to generate reports that are read by PScon for display on the Main Control as Line number etc. This can be turned off to allow the Scon board to have more processing resources such as using the Memory Editor while Scon is in the run mode (not recommended) or faster speed calculations. The default value of 8 in the “Report Rate” box will cause data to be sent several times per second.

Program & Position storage should be set to “Separate”. If “Common/Packed” is set, positions for each instruction must be placed in the element immediately following the instruction. Non-move instructions waste a blank element after each instruction. This is a future option with limited functionality.

Run Controls selects the Start/Stop control method. PScon is always able to control Scon. If "Run on Power up" is selected, the board will execute "Run at line 0" as soon as power is applied. If "Push Button Control" is selected, the on-board push-button will toggle Scon into and out of the run mode. If "Hardware Control" is selected, Input 1 will start the board and Input 2 will stop the board. Line 0 is the start point for hardware and pushbutton control. Note: any combination of controls may be selected.

The "Pulse Width Limits" section limits the minimum and maximum pulse width that the board will generate. This is to protect the hardware and servos. Select the "Use limit controls" box to enable pulse limits. The values are in 10,000 microsecond units. Recommended defaults are 22500 (2.25 Milliseconds) for maximum and 9000 (.9 Milliseconds) for minimum. The same pulse limits apply to all servos.

Once all parameters have been set, click "Store Parameters", this will store the values into the board processors eeprom memory. At this point, the board continues to run with the old values. Power cycling the board or clicking the "Reload from EEPROM" button will reset the board and load the new values.

NOTE: If memory value become damaged or corrupted or changed inadvertently when the Memory Editor (discussed later) the Scon board product may be reset to factory default functions. This is accomplished by holding down the on-board pushbutton while the board is powered up.

Flash memory

When a Scon file is downloaded, the onboard flash memory is overwritten with the new data; only the elements that are used in the new program will be written to. This allows downloading small sections while leaving blocks of subroutines in memory. This is desirable due to the required flash load time on large programs. It also allows commonly used subroutines to be left as part of the program.

The entire flash memory may be erased if desired. Select "Flash tools" from the setup and tools menu, and then click "Erase flash memory".

The "Flash tools" utility also allows reading and writing Scon's flash memory. For reading, a page address range must be selected. Each page contains 16 elements. The entire range of pages (0-2047) may be selected; however reading the entire flash memory may take several minutes. Writing will take longer due to the flash memory write cycle. The flash file name is listed on the "Flash tools" utility screen. Scon flash file format specifies a page address for each page to be written so only what is read into the file will be written back into Scon with a flash write. This is an excellent tool for reading and writing positional data blocks or "cloning" Scon modules.

Trouble Shooting PSCON

Install Issues:

PScon.exe is a complete program that will run on most PC's without full installation. However it does require mscomm32.ocx (the serial port driver) to be installed in the windows system path. This will be done automatically by the install utility if used.

If a problem occurs during installation, copy mscomm32.ocx into the computers Windows system directory, then click on PScon to run it without a full install.

PScon requires one directory (C:\irunner) for data files. The directory will be created automatically when PScon runs the first time.

Com Port Issues:

If a serial port error occurs the first time the program runs; the pre-set default serial port may not be present on the computer. To set or change the serial port from the main screen "Main Control", select the "Setup and Tools" dropdown menu. Set the port to the desired port number, the baud rate should be 057600, parity n, length 8, and stop bits 1. The parameters MUST be saved BEFORE selecting "Re-Connect Com port". If the error occurs again, try selecting port numbers until the correct port is located. The comport number can be located using Windows Desktop or Start tab by right clicking "My Computer", selecting "Properties", and then "Hardware", and then "Device Manager", and then "ports (COM & LPT)". This will list all of the ports present on the computer. Once a valid port is selected, PScon will run without error even if the Scon board is not present. However; the Scon board must be plugged into the selected port to operate properly.

Program Run Errors / Issues:

PScon does not list many errors. If the program does not work as expected check the Scon program carefully. Misspelled keywords may be ignored.

Stack underflow or overflow errors can be caused by a misspelled "call" or "return" keyword or by calling or looping more than the supported stack memory allows.

Get started - an example program

Here is a little program that can be copied into a notepad or a similar text file editor. Do not use a word processor unless files can be saved as simple text files. Copy everything including the ++++ lines. All PC's have notepad or WordPad that work fine as text editors. Sconcon recommends PFE (Programmers File Editor) available free on the web.

```
/++++++++++++++++++++++++++++
Address 0       /Always specify a starting address
Begin           /Just a label that defines a point in the program, you could call it anything
Move 10000      /This line moves the servos to the rotations in element 10000
Wait 20         /This is a short delay
Move 10001      /This line moves the servos to the rotations in element 10001
Output 15 on    /This line turns on the LED on the board
Wait 20         /This is a short delay so the light stays on
Output 15 off   /This line turns off the LED on the board
Goto begin      /This line sends the interpreter back to the start you may also write "Goto 0"
/Stop is not necessary because it's a continuous loop
/Now tell the loader put data starting at element 10000
Address10000            /Notice that spaces are optional
Position        s1=10000 s2=12500    /That's 1MS for servo 1 and 1.25 for servo2
/The loader automatically moves to the next line (10001)
Position        s1=20000 s2=17500    /That's 2MS for servo 1 and 1.75 for servo2
/++++++++++++++++++++++++++++
```

Save the file as Scon.txt in the c:\Irunner directory. This directory was created when PScon was first run. Programs can be stored elsewhere if desired.

This program uses only servo 1 and Servo 2. Do not attach any hardware to the servos at this time; just a small lever such as a servo arm so that movement can be seen. Plug the servos into the first and second servo connectors. The servos will rotate to the center position as soon as the Scon SB is powered up. If not, verify that the servos are connected correctly. Watch the LED on the Scon board flash when power is applied this verifies that it is operating correctly.

After everything is connected; and the file above is created and stored, open PScon and click the tab "Scon File Download". Click "Read Scon File" if the file contains errors or is not located in the correct directory an error will appear, otherwise the file will load and a list file similar the one below will be created. The file is called list.txt and is also in the C:\Irunner directory. Now open it in a text editor and compare it to Scon file already created. It will look something like this:

```
0 Label: BEGIN
0 Move To Position: 10000
1 Wait 0020
2 Move To Position: 10001
3 Output 0015 On
4 Wait 0020
5 Output 0015 Off
6 Goto Line number: 00000
```

10000 Load Position S1= 10000 S2= 12500
10001 Load Position S1= 20000 S2= 17500

Back to “Scon File Download”; after the file is read in, simply click the red “Write to Flash” button and the file will be downloaded into the Scon board. When it’s finished, “OK” will appear in the status box.

Now click the “Run” button on the “Main Control”. This will cause the servos to begin moving in accordance with the program. It will run continuously until stopped. Clicking the “Stop” button will cause the program to stop. If the “Pause” button is clicked, the program will stop but will not reset; clicking “Run” will re-start it at its current position.

Examine each line of the Scon file. Comments (any thing after /) will be ignored and not explained because they do nothing and do not load onto the Scon board.

The first line *“Address 0”* tells the loader to begin placing data at element 0. Everything after this until another “Address” keyword is encountered will be placed sequentially starting with element 0. Because it is not actually part of your program, this line does not show up in the compacted file or the list file but.

The next line *“begin”* is a label, and indicates that you want to name the current address “begin”, after this, each time the word “begin” that address, in this case 0 will be used. In the middle of a larger program it is difficult to keep track of the line number so labels may be necessary.

The next line *“Move 10000”* is stored in element 0 because it’s the first instruction following the “Address 0” instruction. This Move instruction tells the Scon board to get the servo rotations from element 10000 and drive the servos to those rotations. Note that later in the manual, information is stored at element 10000.

The next line *“Wait 20”* causes a pause of about .36 seconds. This line is stored in element 1.

The next line *“Move 10001”* tells the Scon board to get the servo rotations from element 10001 and drive the servos to those rotations. Note that later in the manual, information is stored at element 10000.

The line *“Output 15 on”* turns on output #15. Output 15 is the onboard LED. Note: Accessing unsupported outputs will not generate errors.

The next line *“Wait 20”* causes a short pause of about .36 seconds.

The next line *“Output 15 off”* causes output #15 to turn off.

The next line is the last line in the actual program. Its *“Goto begin”* and causes Scon to return back to the start. This program is an endless loop that will continue until stopped manually.

The next line *“Address10000”* sets the address to element 10000.

The next line *“Position s1=10000 s2=12500”* tells Scon to generate a 1.0 Millisecond (MS) pulse for Servo 1 and a 1.250 MS pulse for Servo 2.

The last line *"Position s1=20000 s2=17500"* tells Scon to generate a 2.0 Millisecond (MS) pulse for Servo 1 and a 1.750 MS pulse for Servo 2.

Placing position movement data into elements as shown in the example is an easy way to test servos and write programs, however in practice the position manager provides a much better way to create movements with multiple servos.

Change the positions of the program

Click the stop button on the main control screen to stop the program. Select "Move and Position control" then click "Position Manager". Set the Position Memory box to 10000; this is element 10000. Click "Get" in the same box. Now look at the positions for each servo; servo 1 should be 10000 and servo 2 should be 12500 as set by the program when loaded. Click the big "+" button for servo 1 and watch the servo move and the rotation value change. Now experiment with +/- for both servos to see how they operate. Note that un-used servos are at 65535. If not set to this value, the Scon board will generate pulse movement data for the un-used servos. Move the servos to arbitrary new positions then click the "Put" button to save the position. Run the program again. The position manager is very useful when creating multiple servo arm or walking movements.

With the position manager open and the program stopped and the Position Memory set to 10000, locate the "Move to" box on the position manager screen and click the "Bump & Move" button. All servos will move to position 10001, now click "Dec & Move" and notice them move back.

The last item on the Position Manager screen is the manual "Output Control" box. Set the output number to 15 and click the on and off buttons. Note that the Scon onboard LED is changing in accordance with the selection. This will work with any onboard driven output or connected slave board output.

Change the speed (rate) of movements in the program
Add the following to line 1

Rate=200

The line should now look like this

Move 10000 rate=200 /This line moves the servos to the rotations in element 10000

The servo with the greatest rotation during the move will now rotate at this new rate on this specific move. In the next move, the servos will move at the default rate as before.

The program now looks like this:

```
/++++++++++++++++++++++++++
Address 0       /Always specify a starting address
begin           /Just a label that defines a point in the program, you could call it anything
Move 10000 rate=200/This line moves the servos to the rotations in element 10000
Wait 20         /This is a short delay
```

```
Move 10001    /This line moves the servos to the rotations in element 10001
Output 15 on  /This line turns on the LED on the board
Wait 20       /This is a short delay so the light stays on
Output 15 off /This line turns off the LED on the board
Goto begin    /This line sends the interpreter back to the start you may also write "Goto 0"
/Stop is not necessary because it's a continuous loop
/Now tell the loader put data starting at element 10000
Address10000          /Notice that spaces are optional
Position      s1=10000 s2=12500   /That's 1MS for servo 1 and 1.25 for servo2
/The loader automatically moves to the next line (10001)
Position      s1=20000 s2=17500   /That's 2MS for servo 1 and 1.75 for servo2
/++++++++++++++++++++++++++++
```

Save this program, read it in and load it into the Scon board. The results of the rate selections will be evident when the program is run.

Now stop the program and remove the rate specification (rate=200) the first move.

Add the line "DSR1=25" to the line preceding the "Goto begin" line.

The program now looks like this:

```
/++++++++++++++++++++++++++++
Address 0     /Always specify a starting address
begin         /Just a label that defines a point in the program, you could call it anything
Move 10000    /This line moves the servos to the rotations in element 10000
Wait 20       /This is a short delay
Move 10001    /This line moves the servos to the rotations in element 10001
Output 15 on  /This line turns on the LED on the board
Wait 20       /This is a short delay so the light stays on
Output 15 off /This line turns off the LED on the board
DSR1=25
Goto begin    /This line sends the interpreter back to the start you may also write "Goto 0"
/Stop is not necessary because it's a continuous loop
/Now tell the loader put data starting at element 10000
Address10000          /Notice that spaces are optional
Position      s1=10000 s2=12500   /That's 1MS for servo 1 and 1.25 for servo2
/The loader automatically moves to the next line (10001)
Position      s1=20000 s2=17500   /That's 2MS for servo 1 and 1.75 for servo2
/++++++++++++++++++++++++++++
```

Save this program, read it in and load it into the Scon board then run it. Note that the speed changes when line "DSR1=25" is executed. This line changes the Dynamic System Rate #1. All Move instructions that do not specify another speed control method use DSR1. Unless a speed is specified otherwise, this new rate will be used for all moves. DSR1 is not reset when the program is stopped. Power-up, reset or changing DSR1 will cause it to change. Refer to "Controlling Speed and Rate" document for further information.

Scon instruction list with encoding

Executer Instructions:

Goto	Instruction:10 Speed:NA Value:0-65,535 (to address)
Call	Instruction:20 Speed:NA Value:0-65,535 (to address)
Return	Instruction:22 Speed:NA Value:NA
Do	Instruction:30 Speed:NA Value:0-65,535 (loop count)
Loop	Instruction:32 Speed:NA Value:NA
Move	Instruction:0 Speed:0-255 (speed address) Value:0-65,535 (rotations address)
Output	Instruction:64 (on) Speed:NA Value:0-65,535 (address of output)
	Instruction:65 (off) Speed:NA Value:0-65,535 (address of output)
Wait	Instruction:40 Speed:NA Value:0-65,535 (wait period)
Nop	Instruction:255 Speed:NA Value:NA
Stop	Instruction:128 Speed:NA Value:NA
If input	Instruction:50 Speed:0-255 (input number) Value:0-65,535 (input value)
	Specify1: 0-255 (Instruction if true) Data1: 0-65,535 (address or data if true)
Marker	Instruction:254 Speed:NA Value:NA

Loader Instructions:

/	Comment
Address	#########
Position	S1#### S2#### etc..
Speed	S1#### S2#### etc..