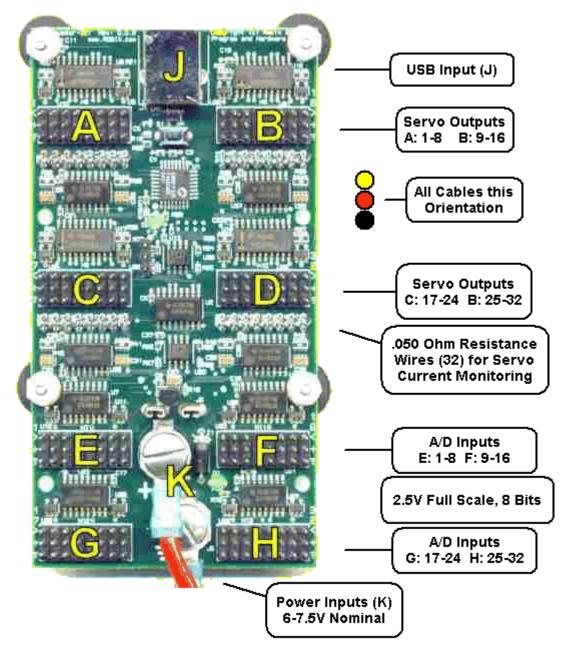
Robix Software, Part I: Quick Start Tutorial

This document may be copied or printed as desired.

Feedback: desk@robix.com. Home: www.robix.com Last mod: 2005-02-1

(Note: For the Tutorial on the *LPT* -connected Controller, See "Help" in that software)

Here's the Usbor servo controller. Connect 6 servos to positions 1-6 on connector 'A'. Make sure to have the yellow/orange wire on 'top', as shown in the **All Cables this Orientation** legend, below right.



Install the Robix Software

Install the Robix software according to the instructions on the CD that accompanies it. If you don't have the CD, you can download the software from <u>www.robix.com</u> by clicking on the download section and following the instructions there. Note that you may need to install the Java SDK if you don't have it already.

Plug in Power and USB

Once the software is installed, plug the power supply into the USB servo controller (called the "Usbor" for short) and use the USB cable to plug the controller into the computer.

Start the Software: Two Programs

Next, start two programs, called Nexus and Nexway. You can start them in either order.



In the Nexus window you should see an 11-character Usbor serial number. In this case it begins **3MU...**.

Usbor Nexus 0.0.12	2
e Help	
Listening for Nexway connections on TCP/IP port: 4575	
vices Nexways	
Detected Devices:	
MUCYKMVDH6	
am1	
ann	
nfo Configuration Firmware Calibration Camera Settings	1
Status: OK	
Status: OK Firmware Version: 0.0.11	
Firmware Version: 0.0.11	
Firmware Version: 0.0.11 Usbor Configuration	
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1	-
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 8	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 5	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 8	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 5 Pod: Pod2 Servos: 6 Digouts: 2 Sensors: 5	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod2 Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod3 Servos: 6 Digouts: 2 Sensors: 7	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod2 Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod3 Servos: 6 Digouts: 2 Sensors: 7	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod2 Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod3 Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod4 Servos: 6 Digouts: 2 Sensors: 7	8
Firmware Version: 0.0.11 Usbor Configuration Config Name: Classic ID: 1 Pod: Podl Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod2 Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod3 Servos: 6 Digouts: 2 Sensors: 7 Pod: Pod4 Servos: 6 Digouts: 2 Sensors: 7 Unused ROM bytes on Usbor: 481 of 1413	8

If Usbor show configuration **Config Name: Classic ID: 1** at the top of the lower window then you can skip the next page. Otherwise you'll have to load the Classic config.

If Needed: Loading the Classic Configuration into the Usbor

In the Nexus, click on the **Configuration** tab. Click on the **Select Configuration File** button and in the file dialog that follows you *should* see **Classic.rbxUsborConfig** listed.

If you *do not* see **Classic.rbxUsborConfig** listed then create it as follows: Close the file dialog if still open, then click **Open Usbor Configuration Editor**. Click **File** | **New Configuration** | **'Classic'**. Then click **File** | **Save Configuration** and in the file dialog click **Save** (the filename **Classic** should already be filled in for you in the file dialog). Close the Usbor Configuration Editor. Now you can return to the top of this page and again follow the instructions there.

<mark>₹</mark> Usbor Nexus 0.0.12	_ 🗆 🗵
File Help	
Listening for Nexway connections on TCP/IP port: 4575	
Devices Nexways	
Detected Devices:	
3MUCYKMVDH6	
Cam1	
Info Configuration Firmware Calibration Camera Settings	
Open Usbor Configuration Editor	
To download a config without the help of the config editor, use the buttons below.	
Select Configuration File Classic.rbxUsborConfig	
Config Name: Classic ID: 1	<u> </u>
Pod: Podl Servos: 6 Digouts: 2 Sensors: 8	
Pod: Pod2 Servos: 6 Digouts: 2 Sensors: 8	
Pod: Pod3 Servos: 6 Digouts: 2 Sensors: 8	
Pod: Pod4 Servos: 6 Digouts: 2 Sensors: 8	
	- I
Download Configuration to Usbor	

Open Classic.rbxUsborConfig to see:

Click the **Download Configuration to Usbor** button and the configuration is re-loaded into the Usbor. The process will take a few seconds.

The Nexway, first time Starting

When you open the Nexway for the very first time, you see a blank window: (If this is not the first time the Nexway has been opened then you'll skip the rest of this page and the next one.

Cusbor Nexway 0.0.12	
File Connect Control	Help
New Nexus Connection	Open Pod GUI
I	
last.rbxNexwaySession	

If your Nexway is blank, click on the New Nexus Connection button to see:

New Nexus Connection
Enter an IP address, host name, or 'localhost':
localhost
Enable Auto-Reconnect
Retry Interval (in seconds): Retry Limit:
5 - 0 -
OK Cancel

Since we are working **locally**, that is, with the **Nexway** and **Nexus** on the **same computer**, we select the default of **localhost** by clicking **OK**.

Next we click on **3MU...** (or whatever your Usbor's serial number is) and we see:

The Nexway

Click on the + signs to open the folders to see:

Vsbor Nexway 0.0.12	
File Connect Control H	elp
New Nexus Connection	Open Pod GUI
⊡-⊖ localhost	
Pod1	
Pod2	
Pod3	
Pod4	
Cam1	
Usbor Detail	
Firmware Version:	0.0.11
Serial Number:	3MUCYKMVDH6
Config Name:	Classic
Pod Count:	4
Avg Rep Response Time:	2 msec
Property Heartbeat Interval:	
Sensor Heartbeat Interval:	200 msec
last.rbxNexwaySession	

The Pod's Graphical User Interface (GUI)

Click on **pod1** in the Nexway to open that pod's gui ("gooey").

Rod1 - Us	bor: 3	MUCYKM	1¥DH6					_ 🗆 ×
File Edit C	ontrol	Gui He	elp					
$\overline{\mathbf{A}}$			📕 As	sign P	ower Of	f Power	On	.
local assign	ned his	story						
Untitled Scrip	rt 🛛					L	1 C:	1 Main Script
		Servo P	ropertie	s				
• •	1	2	3	4	5	6		
pos	0	0	0	0	0	0		
maxspd	100	100	100	100	100	100		
accel	10	10	10	10	10	10		
decel	10	10	10	10	10	10		
absi	8	7	7	8	7	7		
Pod1 3MUC	YKMVE	H6 loc	alhost	idle		defau	lt.rbxPodGui	(modified)

Adjusting the Pod GUI

Right click anywhere in the **Servo Properties** panel and select **Split This Pane** and then **▼Down**. Then right click in the new panel and select **Replace With** and then **Sensors**,

Now right click again in the **Servo Properties** panel and select **Split This Panel** and then ▶ **Right**. Then right click in the new panel and select **Replace With** and then **Digouts**.

Rod1 - Us	sbor: 3N	MUCYKN	1VDH6									
File Edit (Iontrol	Gui He	elp									
			📕 As	sign P	ower Of	f	r On					
local assign												
local assig	nea nis	story										
										46 1 0 11		
Untitled Scrip	nt j					JL		1 C:	1	Main Script		
		Servo F	ropertie	s				Į	Digou	ts		
▲ ▶	1	2	3	4	5	6			12			
pos	0	0	0	0	0	0			••			
maxspd	100	100	100	100	100	100						
accel	10	10	10	10	10	10						
decel absi	10 8	10 8	10 6	10 8	10 7	10 6						
ausi	0	0			<u> </u>	0						
			Sens		_		_					
	1	2	3	4	5	6	7	8				
<u> </u>	3	4	3	4	4	_4	4	6				
Pod1 3MUC	CYKMVD	H6 loc	alhost	idle		defa	ult.rb×	PodGu	i (mod	lified)		

Click Gui in the menu and save as default.

Your First Script

If you don't have links or at least link horns on the 6 servos you plugged in at the beginning of this tutorial, add them now so that servo motion is readily visible.

Stretch the pod gui vertically to increase the size of the blue scripting panel. Then type in these two lines.

```
move all to -1000; wait 5; digout 1 on, 2 off; wait 5
move all to 1000; wait 5; digout 1 off, 2 on; wait 5
```

Rod1 - Us	sbor: 3N		1¥DH6							×	
File Edit (Iontrol	Gui He	elp								
	V V III D Assign Power Off Power On										
local assign	ned his	tory									
	move all to -1000; wait 5; digout 1 on, 2 off; wait 5 move all to 1000; wait 5; digout 1 off, 2 on; wait 5										
Untitled Scrip	nt 🗍					L	:	2 C:	54 Main Sc	ript	
		Servo P	ropertie	s					Digouts		
• •	1	2	3	4	5	6			12		
pos	0	0	0	0	0	0					
maxspd	100	100	100	100	100	100					
accel	10	10	10	10	10	10					
decel absi	10 7	10 7	10 9	10 8	10 6	10 5					
			Sens	sors							
	1	2	3	4	5	6	7	8			
	4	4	3	4	4	4	4	6			
Pod1 3MUC	YKMVD	H6 loc	alhost	idle		defa	ult.rb:	xPodGu	ii (modified)		

Now **run the script by clicking the 'run' button** at upper left.

Click \blacksquare again to run the script again. Notice that the servos move back and forth, the **pos** values change, and the digouts change color as they are turned on and off.

Your First Macro

Stretch the pod gui vertically to make more room in the scripting area. Then turn the two script lines into a script **macro** named **osc** by altering the script to this form:

Pour - u	Isbor: 3	MUCYKI	M¥DH6							_ 0	×
File Edit	Control	Gui H	lelp								
	: 11 D		📕 A:	ssign F	Power O	ff Powe	r On				
local assig	gned hi	story									
osc 4 macro osc move all move all end	to -10										
Untitled Scri	ipt						L:	1 C:	6	Main Scr	ipt
Untitled Scr	ipt 🛛	Senio I	Pronertie				L:	1 C:			ipt
Untitled Scr	ipt [Servo I 2	Propertie 3	es 4	5	6	L:	1 C:	6 Digou 1 2	ts	ipt
Untitled Scr					5 1000		L:	1 C:	Digou	ts	ipt
pos maxspd	1 1000 100	2 1000 100	3 1000 100	4 1000 100	1000 100	6 1000 100	L:	1 C:	Digou 1 2	ts	ipt
pos maxspd accel	1 1000 100 10	2 1000 100 10	3 1000 100 10	4 1000 100 10	1000 100 10	6 1000 100 10	L:	1 C:	Digou 1 2	ts	ipt
pos maxspd accel decel	1 1000 100 10 10	2 1000 100 10 10	3 1000 100 10 10	4 1000 100 10 10	1000 100 10 10	6 1000 100 10 10	L:	1 [C:	Digou 1 2	ts	ipt
pos maxspd accel	1 1000 100 10	2 1000 100 10	3 1000 100 10 10 8	4 1000 100 10 10 7	1000 100 10	6 1000 100 10	L:	1 C:	Digou 1 2	ts	ipt
pos maxspd accel decel	1 1000 100 10 10 6	2 1000 100 10 10 7	3 1000 100 10 10 8 <i>Sen</i>	4 1000 100 10 10 7 sors	1000 100 10 10 10 7	6 1000 100 10 10 6			Digou 1 2 •	ts	ipt
pos maxspd accel decel	1 1000 100 10 10	2 1000 100 10 10	3 1000 100 10 10 8	4 1000 100 10 10 7	1000 100 10 10	6 1000 100 10 10	L: 7 3	1 (C:	Digou 1 2 •	ts	ipt

Note that the **osc 4** line could also have come after the macro definition instead of before it.

Run this script as before (\checkmark) and see that **osc** runs 4 times. If you change the **4** to **0** and run the script, you'll see that **osc** runs indefinitely. After **osc** has repeated several times, **stop the script** by clicking **anywhere in the script area**, or by clicking on the red stop button \blacksquare .

Note that in general, you can have multiple commands on single line separated by ;'s. A ; is allowed at the end of a line but is not required. Commands may not 'span' lines; that is, a command needs to start and end on the same line.

A script may contain **any number of macros**, and macros can call other macros.

Servo Current and Torque

Now **apply gentle rotational pressure** on the output shaft of servo 1 and watch the **absi** value of servo 1 increase as the servo draws more current to oppose the force you are exerting. The **absi** value is the absolute value of the short-term (~.1 second) average of the servo current. This value, in general, indicates the torque being exerted by the servo and are approximately equal to the milliamps being drawn by the servo.

Remember: **Don't leave servos straining** with **absi** over about 500-600 for more than a couple of minutes at a time; otherwise your servos can overheat, shortening their lives.

Teach Mode

Next, **position the text cursor on the word 'end'** at the end of the macro and **open the Teach Window** by clicking on the teach button 'B'. You should notice an arrow, '->', appear in a new blank line above 'end'. In addition, the teach window will appear:

Teach - Pod1 - Usbor: 3M 🗙										
Servos										
• •	1	2	3	4	5	6				
+ Coarse	1	2	3	4	5	6				
- Coarse	Q	W	E	R	Т	Y				
+ Fine	A	s	D	F	G	H				
- Fine	Ζ	Х	С	V	В	N				
 All Serv Moved 3 	5	1	Mo Ju	ove mp						
Ad	ld te	o Si	crip	rt						

While the **Teach Window** has focus your keyboard is acting as a **teach pendant**. That is, holding down various keys, as indicated in the Teach Window, will move the corresponding servo by coarse or fine increments in the positive or negative direction. You will see the position servos move and their **pos** values change in the pod gui.

After you have moved the pod's servos to new positions, click on the 'Add to Script' button in the Teach window, and a move command will be inserted on the line with the '->' arrow and the arrow will move down a line. Repeat these steps to add several new lines to the **osc** macro, then run the script again to see your added move commands in action.

Robix Software, Part II: Tutorial in Depth

Why Two Programs?



Why **two** programs? Because the Robix software has the advanced capability to run robots *remotely*, over a network, or even over the internet. But if two machines are involved then they each need to be running a program. One program, the Nexus, "talks" directly to the Usbor via USB. The second program (the Nexway) talks over the network to the Nexus, sending commands and receiving continual status reports.

In the case when you're running "locally", that is, on just one machine, then the Nexus and Nexway run on that one machine.

And what's so valuable about operating robots remotely? To begin with, it allows two or even more student groups on separate but adjacent computers to share the same Usbor controller, reducing expenses:. The controller can run 32 servos at once, and these can be divided between the groups or what we call "pods".

In addition, and at least as important, remote monitoring and control of robots reflects the reality of today's factory floors.

The Nexus, continued

The word "**nexus**" is generally defined as "**a means of connection**". The Nexus in our case connects hardware such as Usbor **controllers** and their **servos**, and **webcams** used in the V-1.0 vision system **to user interfaces and user programs**. The Nexus is needed because, as mentioned, the user interfaces and programs may be running on other computers on a network or on the internet.

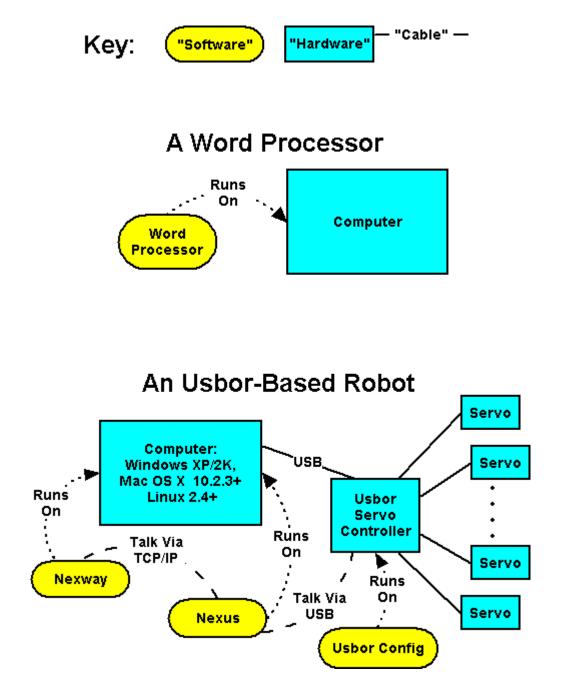
In case you're wondering, the Nexus communicates to these other programs using TCP/IP, the primary protocol(s) of the internet.

The Nexway, continued

The Nexway (think "Gateway to Nexus") is the last link in the chain joining the user interface (and user programs) through the Nexus to the Usbor and its pods and finally to the pods' servos, digouts (digital outputs) and sensors.

Summary So Far...

To sum up, here's a diagram so far of the software and hardware. We contrast a conventional program (a word processor) with our remote robotic software:



Editing Scripts to Change Servo Parameters

Commands that alter servo properties need to be edited into the script by hand. In the macro **osc**, change the speed of the motion, add the command

maxspd all 5

as the first line in the macro, and then run the script again. You'll see that the motion is slower than before.

By the way, instead of running the script with the \square button, you can also double-click on the **osc 0** line to run the osc macro indefinitely.

You can also adjust other servo properties, assuming that these servo properties were defined as 'variable' (the default case) in the pod's **configuration**. For example, change the **-1000** in your script to **minpos** and change the **1000** to **maxpos**. And right-click on the **Servo Properties** panel and choose **Select Properties**. Then check **minpos** and click **OK**.

Now run osc again to see that the range of motion has been increased.

Pod1 - I	Ishor: 3	мнсук	MVDH6								
	File Edit Control Gui Help										
	🔽 🛫 🔢 🗅 🔳 🗒 Assign Power Off Power On										
local assid	local assigned history										
osc O	grioù l'ri	0.01)									
080 0											
macro os	-										
maxspd al											
maxspu a. move all		nnos.	wait P	. dia	out 1	on 2	off.	wai	+ 5		
move all								wai	τs		
<pre># lines #</pre>	crom y	our te	acn mo	ode exi	ercise	are i	iere				
# etc.											
end											
Untitled Scr	ipt						L:	4 C:	13	Main Script	
		Servo I	Propertie	es e					Digou	ts	
	1	2	3	4	5	6			1 2		
pos	1000	1000	1000	1000	1000	1000			00		
minpos	-1500	-1500	-1500	-1500	-1500	-1500					
maxspd	100	100	100	100	100	100					
accel	10	10	10	10	10	10					
decel	10	10	10	10	10	10					
absi	7	8	8	6	8	6					
			Sen	sors							
		-	3		_	~	7		8		
	1	2	3	4	5	6	ſ		D		
	า 4	2 4	э З	4 4	5 4	б 4	4		0 6		
		4	3	4		4	-		- 6	122 - 15	

Next, see what happens when you add the line

minpos all -800; maxpos all 800

to the top of the macro **osc** and then run the macro again. Now the motion is more restricted.

Finally, change this last line added to

minpos all default; maxpos all default

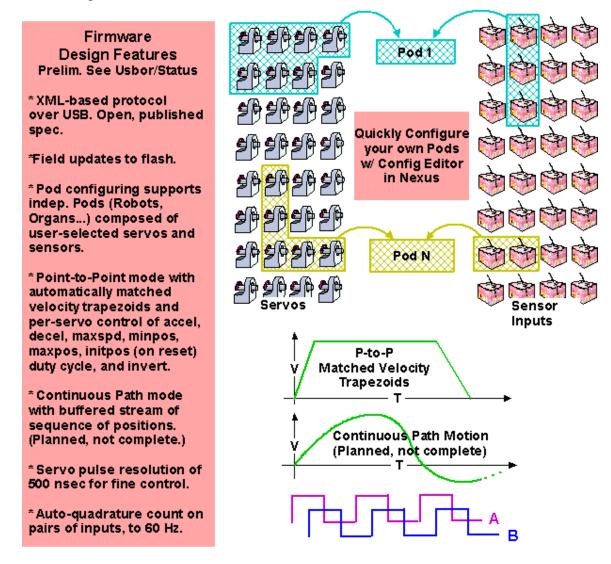
to return the values to their defaults of -1400 and 1400.

For a listing of all script commands and their usage go to

http://www.robix.com/Robix Scripting Reference.pdf

Understanding the Usbor Configuration

The Usbor configuration is stored in the flash memory of the Usbor itself and is retained even when power is removed. The configuration defines how the 32 servo outputs (each of which can alternately be a digital output or 'digout') and the 32 analog inputs are assigned to various pods in the Usbor.



Creating Your First Configuration and Loading it into the Usbor

In this example we will use 18 of the 32 servos, and make three pods of 10, 5 and 3 servos each. We begin by clicking the **Configuration Tab** on the **Nexus**.

Cusbor Nexus 0.0.12	_ 🗆 ×
File Help	
Listening for Nexway connections on TCP/IP port: 4575	
Devices Nexways	
Detected Devices:	
3MUCYKMVDH6	
Cam1	
Info Configuration Firmware Calibration Camera Settings	
Open Usbor Configuration Editor	
To download a config without the help of the config editor, use the buttons below.	
Select Configuration File	
Download Configuration to Usbor	
L	

Next click Open Usbor Configuration Editor and get

Reprint Configuration Editor 0.0.	9			- U ×
File Configuration Help				
Download				
🔄 Usbor Configuration (0 Pods)		Par	ent Servo	
E-Carent Servos	Name:		Syster	m_All_Var
System_All_Var	Est. RC	initpos Var C minpos Var C M Cost: 26		58
Unsaved Usbor Config	Unused ROM bytes: 1	413 of 1413 Un	used RAM bytes: 18	324 of 1824

Now we click Configuration and float over New and we see

<mark>₹</mark> Us	sbor Configuration Edit	or 0.0.9			_ 🗆 🗵	
File	Configuration Help					
	New	۱.	Pod	Ctrl+Shift+P		
50	Duplicat <u>e</u> Pod	Ctrl+Shift+E	VT <u>S</u> ervo	Ctrl+Shift+S		
	<u>R</u> emove	Delete	Jump Servo	Ctrl+Shift+J	System_All_Var	
	Download to Usbor		Digout	Ctrl+Shift+D		
		ADC Sensor	Ctrl+Shift+A	<u> </u>		
		Quadrature Sensor	Ctrl+Shift+Q	0		
I			Pa <u>r</u> ent Servo	Ctrl+Shift+R		
			Est. ROM Cost: 26	6 Est. RA	AM Cost: 58	
Unsa	ved Usbor Config	Unused	ROM bytes: 1413 of 141	3 Unused RAM	bytes: 1824 of 1824	

We will be making new **pods** (with **Ctrl+Shift+P**) and adding **servos** (with **Ctrl+Shift+S**) to the **pods**.

We're using VT (velocity trapezoid) servos which give smooth motion. Jump servos save configuration memory but can only execute jump commands which result in sharp, uncoordinated motions.

Let's add the pod with 10 servos first. It's very quick and easy. Press Ctrl+Shift+P once to start a new pod. Press Ctrl+Shift+S 10 times to add 10 servos. That's it!

Configuration Editor 0.0.9							
File Configuration Help							
Download	_						
Usbor Configuration (1 Pod)	Velocity Trapezoid (VT) Servo						
🕀 💼 Parent Servos	Name: Servo10						
E-OP Pod1	Phy Port (Pin):						
Servo1	Parent Servo: System_All_Var						
Servo2	,						
	initpos						
	💿 inherit 💿 Var 🔿 Const 🛛 🕦						
- Servo6	O override						
- • Servo7							
Servo8	minpos						
Servo9	ⓒ inherit ⓒ Var C Const -1400						
Servo10							
	Coverride C Var C Const -1400						
	ID: 10 ROM Cost: 26 RAM Cost: 58						
Unsaved Usbor Config Unused F	ROM bytes: 1136 of 1413 Unused RAM bytes: 1241 of 182						

Page 16 of 19

And if you make a mistake, just delete the pod (s) and start over.

Now add another pod and 5 servos and finally a third pod and 3 servos. This gives us:

Usbor Configuration Editor 0.0.9		×				
File Configuration Help						
Download						
Configuration (3 Pods)	Velocity Trapezoid (VT) Servo					
🔄 🕀 💼 Parent Servos	Name: Servo	3				
È⊖ Pod1	Phy Port (Pin):	E				
Servo1	Parent Servo: System_All_Var	i.				
Servo2		1				
Servo3	_ initpos					
Servo4	inherit Var Const 0					
Servo7	C override C Var C Const 0					
Servo8						
Servo9						
Servo10	💿 inherit 💿 Var 🔿 Const 🛛 -1400					
E Pod2	C override					
Servo1						
Servo2	maxpos					
Served	ⓒ inherit					
Servo4						
	Coverride C Var C Const 1400					
L → Servo1	p0pos					
- Servo2						
Servo3	Inherit	-				
	ID: 3 ROM Cost: 26 RAM Cost: 58	-				
l						
Unsaved Usbor Config Unused ROM bytes: 894 of 1413 Unused RAM bytes: 771 of 1824						

Now we need to give this configuration a name. We'll call it 10-5-3. Click on File / Save and save our configuration with that name.

Finally, we have to download this configuration to the Usbor.

Click **Download**. Then select your Usbor (if you have more than one connected) for the download.

When complete you can close the download selector and the configuration editor.

Take a look at the gui for pod1. It has automatically detected the change to its pod and adjust its contents accordingly. Since there are no digital outputs or sensors in this pod we have removed the digout and sensor windows (by right clicking and selecting **Remove This Panel**)

Select **Gui** from the menu and click **Save**. Then save the gui layout as **default**. Any other pod gui's that you open will now have this layout. You could also have saved this gui under some other name. This would enable you to switch layouts easily by selecting **Gui** again and then **Open Pod Gui...**.

Tour c	Isbor: 3	MUCYK	MVDH6							
File Edit Control Gui Help										
<pre>local assigned history osc 0 macro osc maxspd all 5 minpos all default; maxpos all default; move all to minpos; wait 5; digout 1 on, 2 off; wait 5 move all to maxpos; wait 5; digout 1 off, 2 on; wait 5 # lines from your teach mode exercise are here # etc. end</pre>										
Untitled Scr	ipt						L: :	5 C:	40 <i>Mai</i> .	n Script
Untitled Scr	ipt			Servol	Propertie		L: :	sja	40 <i>Mai</i>	n Script
Untitled Scr	ipt 1	2	3	Servo i 4	Propertie 5		L: : 7	8 S J C:	40 <i>Mai</i> . 9	n Script 10
pos	1 0	0	0	4 0	5 0	98 6 0	7 0	8 0	9 0	10 0
pos minpos	1 0 -1400	0 -1400	0 -1400	4 0 -1400	5 0 -1400	es 6 0 -1400	7 0 -1400	8 0 -1400	9 0 -1400	10 0 -1400
pos minpos maxspd	1 0 -1400 100	0 -1400 100	0 -1400 100	4 0 -1400 100	5 0 -1400 100	es 6 0 -1400 100	7 0 -1400 100	8 0 -1400 100	9 0 -1400 100	10 0 -1400 100
pos minpos maxspd accel	1 0 -1400 100 10	0 -1400 100 10	0 -1400 100 10	4 0 -1400 100 10	5 -1400 100 10	es 6 0 -1400 100 10	7 0 -1400 100 10	8 0 -1400 100 10	9 0 -1400 100 10	10 0 -1400 100 10
pos minpos maxspd	1 0 -1400 100	0 -1400 100	0 -1400 100	4 0 -1400 100	5 0 -1400 100	es 6 0 -1400 100	7 0 -1400 100	8 0 -1400 100	9 0 -1400 100	10 0 -1400 100

Run the script again. Note the error **Number is out of range** – 1 and that the "1" in **digout 1** is highlighted. This is because the pod now has no digital outputs. Remove **digout 1 on, 2 off** and **digout 1 off, 2 on** and run again. This time execution should proceed without error.

Special Subject: Setting Initpos

When servos first power up and "come to attention" they immediately track as quickly as possible to their **Initpos** (initial position) which is 0 by default, or about at the middle of their travel. This can cause problems depending on what's in the way.

Consider the situation where you have prepared a "work cell", that is, some fixtures surrounding your robot with which it will interact. However, when the robot starts up it smacks into your fixtures and dislodges them. This problem can be addressed in the configuration editor by changing the **Initpos** of some or all of the servos in your robot.

First use teach mode to move the robot to the desired starting position. Then open the configuration editor and for each servo whose initial position needs to be adjusted click the appropriate servo in the configuration list and change its **Initpos** radio buttons from **Inherit** and **Var** to **Override** and **Const**. Then note the **Pos** value of the servo in the pod gui and copy this value into the text area to the right of **Const**. Below we have changed **Initpos** of **Servo5** on **Pod1** to **-355**.

When you're done, save and download the configuration. The robot will relax for a moment during reset then come to attention at the desired initial position.

Usbor Configuration Editor 0.0.9		×
File Configuration Help		
Download		
Solution (Section (Section (Section 2)	Velocity Trapezoid (VT) Servo	
🔁 💼 Parent Servos	Name: Servo	5
Pod1	Phy Port (Pin):	Ę
Servo1	Parent Servo: System_All_Var	1
Servo2	Parent Servo. [System_All_var	1
Servo3	initpos	*
Servo4	Cinherit C Var C Const 0	
Servo5		
Servo6	· Override C Var O Const -355	
Servo7		
Servoo	minpos	
Servol0	⊙ inherit ⊙ Var C Const -1400	
E-@ Pod3	C override ⊙ Var C Const -1400	_
Servo1		Ľ
- Servo2	D: 5 ROM Cost: 26 RAM Cost: 56	
10-5-3.rbxUsborConfig	Unused ROM bytes: 894 of 1413 Unused RAM bytes: 773 of 18	24

Setting Other Servo Parameters

Except for **Initpos** it is rarely necessary to preset servo parameters. In the unusual case where the configuration you choose is too large to fit the Usbor controller's memory you can save a few bytes by changing some addition parameters to Const from Var. Otherwise there is little reason to make any adjustments.