SST Programming Software

Introduction

Welcome to Saitek Smart Technology (SST) - a powerful and intuitive software to help you get the most functionality from your Saitek controller.

Most modern games do have their own control configuration screens, but by using the SST software you can:

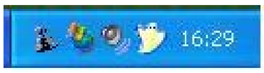
- Increase the number of assignable functions through shift states and multiple modes on your joystick or game pad;
- Create and save game profiles for your favorite PC games, which eliminates the need to go back and re-configure for that game every time you want to play it;

- Assign keyboard and mouse commands so that you can use your Saitek controller to play PC games that do not offer support for game pads and joysticks.

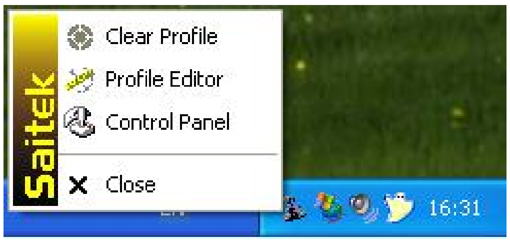
Getting Started

When you have installed the SST software and plugged in your controller for the first time, the Profile Editor will automatically appear so you can start programming.

After this, an icon will appear in taskbar next to your clock every time you plug in your controller. This is called the Profile Launcher and looks like this:



Right-click on the joystick, pad or wheel icon and you should see a pop-up menu like this:

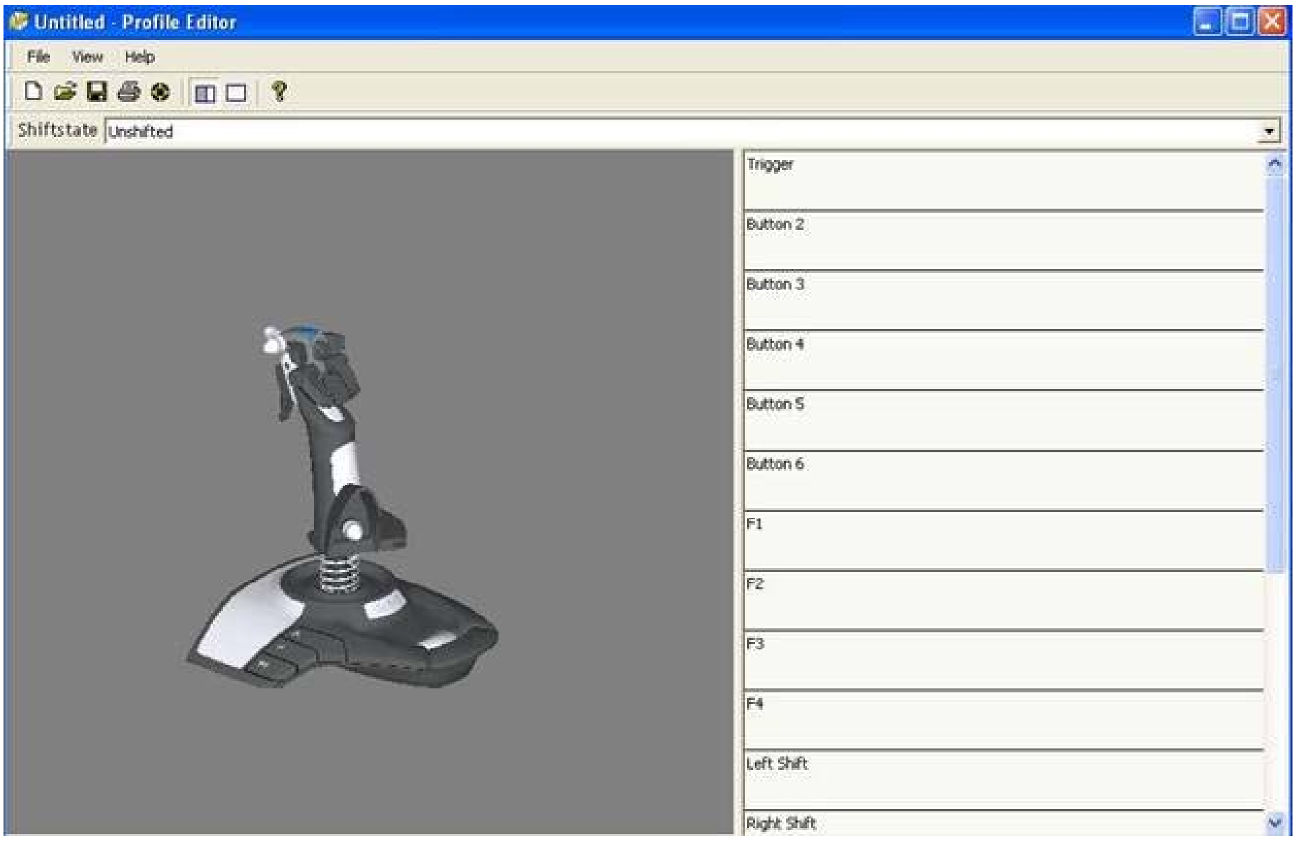


Clear Profile is used whenever you want to completely clear a profile from your controller;
Control Panel will bring up the test and calibration screens for your controller;

Profile Editor will present you with the following screen. Please note that this example uses the Cyborg stick but the procedure for programming your Saitek controller is exactly the same; the only difference is that each Saitek controller will have its own feature set designed to perform particular game functions.

Notes:
The Profile Editor can also be run by double clicking the Saitek Smart Technology Programming Software shortcut on your desktop or by clicking Start>Programs>Saitek>Profiler

If you have more than one Saitek controller plugged in at the same time, you will have a separate icon for each controller. When you hold your mouse pointer over the small icon a small pop-up notice tells you which controller the icon refers to.



The Profile Editor consists of a 3D model of your controller with a list of buttons down the right hand side of the screen. To assign game commands to the controls you must first press one of the buttons on your controller. This button will light up on the 3D model and the appropriate line in the list of buttons is highlighted by a slightly darker background to show you which button you are programming. This is called the 3D View.

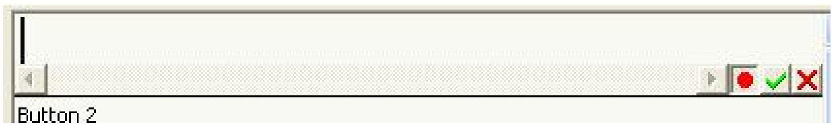
Note that there is another view you can use - this is described later in the manual.

Simple Commands or Keystrokes

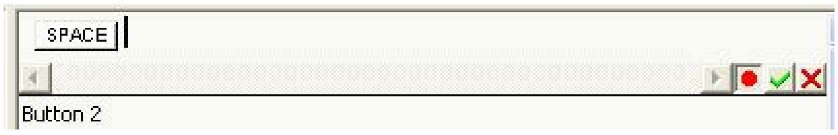
Every game has most in-game commands assigned to various buttons or combination of buttons on the keyboard. Using the Profile Editor, you can make the buttons on your controller pretend to be buttons on the keyboard.

Programming a keyboard command to a button
On the right side of the window in the list of buttons, click on the space just beneath the name of the button you wish to program (the mouse cursor will change shape to indicate that you can click it).

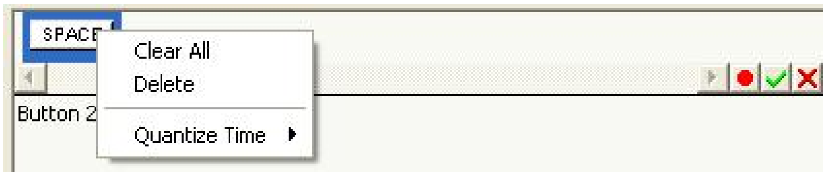
In the example below we have clicked on the space just beneath the trigger.



This makes the cursor flash indicating that the program is now waiting for you to input a keystroke. As an example, imagine you were programming the controller for a game and we wanted to use the trigger for firing a weapon. In the game, the key on the keyboard that does this is the Space key. So press that key on the keyboard and it will appear in the line onscreen; like this.

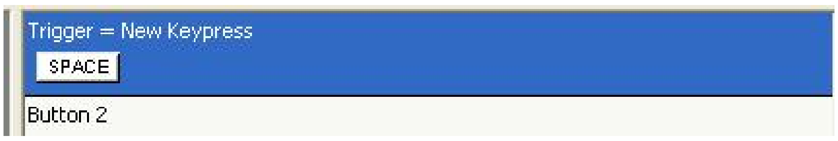


If this is okay then simply click the green tick at the bottom right of the line. If you have made a mistake or wish to clear a keystroke that you have assigned, right click on the keystroke you wish to remove and click *Delete* from the drop down list of options. Note that you can also click on the next button you wish to program (in the 3D view or in the list of buttons) and this will also save the keystrokes that you have just input.



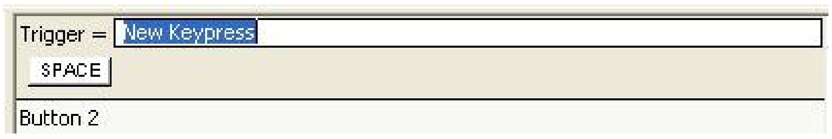
If you wish to clear all the keystrokes from the line then click *Clear All*.

Once you have decided on your keystroke and clicked the green tick, your screen should look like this.



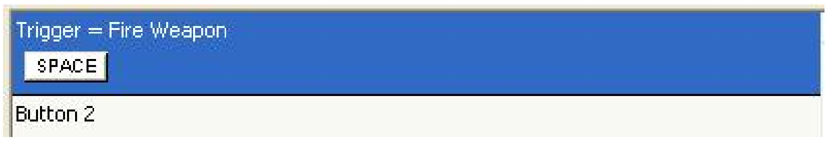
As you can see the software is now telling us that the trigger, when pressed, will initiate the Space bar command on the keyboard. This command has been named *New Keypress* as it was the first new keystroke that we have programmed.

If you wish to rename the keystroke to something more descriptive such as *Fire Weapon*, this is easily done by pointing the mouse cursor at the words *New Keypress* and clicking the left mouse button once. You will see that the words are now highlighted, as in this picture.



You can now type the name that you wish to call this command. In this case it’s been named *Fire Weapon* but you can call it whatever is appropriate to the command you are creating.

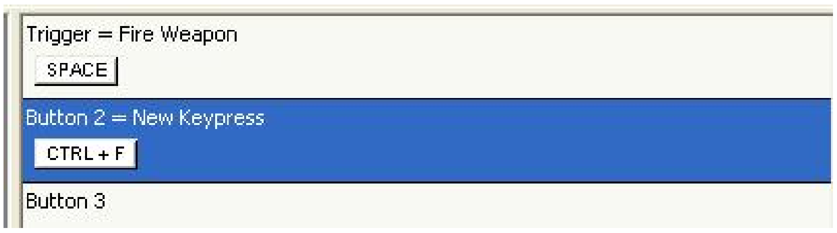
Simply press the *Enter/Return* key after typing your chosen command name in and it will update as below.



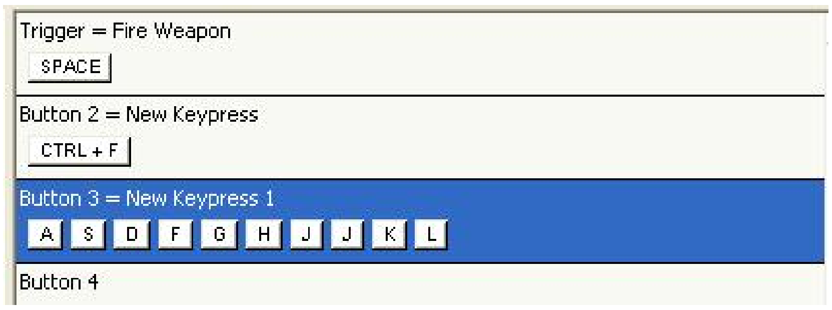
You have just created the first command for your controller.

Multiple keystrokes/Combined keystrokes

You are not limited to single keystrokes when programming commands. You can input as many keys into one command as you like, or combined keys, such as Ctrl+F, for example. The process is the same as inputting single keystrokes – just press the keys you wish to program into the command. In the example below we have input a Ctrl+F command.


Trigger = Fire Weapon
SPACE
Button 2 = New Keypress
CTRL + F
Button 3

You can see that it is a combined keystroke because both letters are on the same 'key' and there is a plus sign between them. Multiple, single keystrokes would appear like the line for Fire B in this picture.


Trigger = Fire Weapon
SPACE
Button 2 = New Keypress
CTRL + F
Button 3 = New Keypress 1
A S D F G H J J K L
Button 4

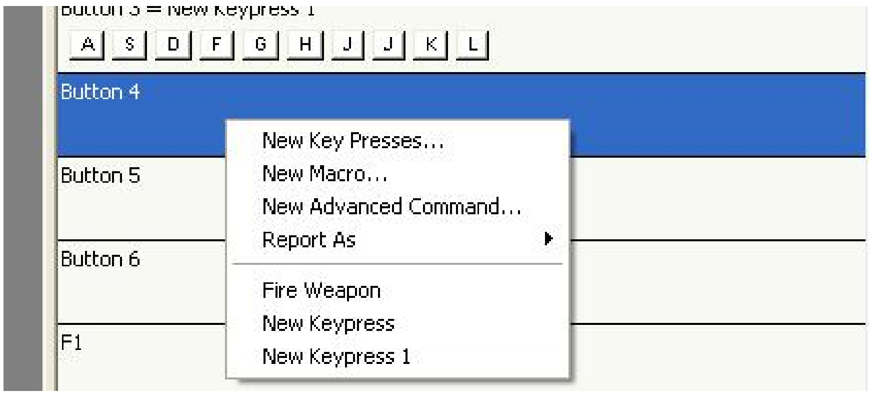
Macros

Having covered the basic types of keystroke, we need to introduce another type of keystroke command – a macro. A macro is a sequence of keystrokes that can be executed with the single press of a button. At first glance you might think that this is exactly the same as the multiple keystrokes, as assigned to the Fire B button in the above example. However this is not the case.

In order for the multiple keystrokes in the above example to actually happen you have to keep the assigned button held down until all of the commands have happened. If the above sequence had been assigned as a macro then you would just have to press the assigned button once and the keystrokes would then input themselves automatically.

You need both types of command because a macro can't be interrupted except by pressing another command on the stick. A simple sequence of keystrokes, like the one in the above example, can be interrupted by releasing the button. Depending on the in-game situation, this could be important.

Creating a macro is similar to creating a keystroke sequence but to access it we have to use the right mouse button. In the example below we have pointed to the *Fire C* line with the mouse and clicked the right mouse button (right-click).



Before proceeding, it's important to examine this drop-down menu. As you can see, we have a number of possible choices. The first is *New Key Presses* – clicking this would enable you to input keystrokes for a new command, just like we have been doing in the above examples.

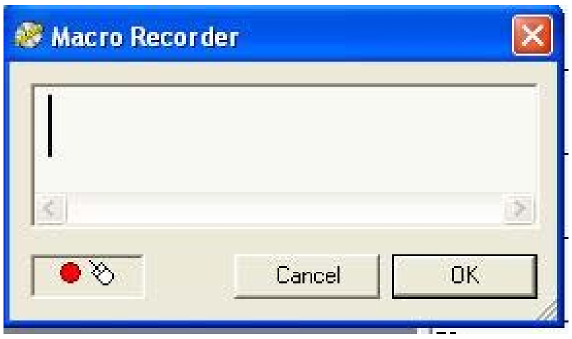
New Macro is the option we will be clicking next.

New Advanced Command offers more options involving repeat functions (amongst other things) which will be covered later.

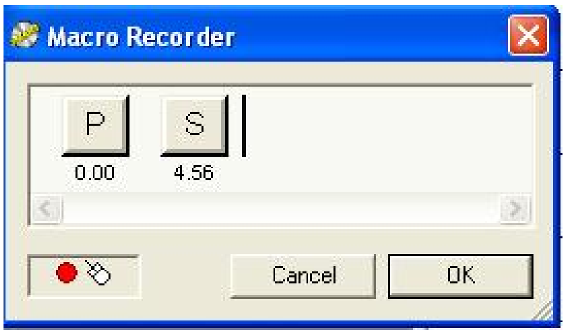
Report As, when you hold the mouse over it, brings up a list of the different button commands, allowing you to make the button pretend to be a different button on the controller.

Underneath these four options you can see listed the commands we have created previously. The *New Keypress and New Keypress 1* are still called that because we didn't rename them but, as you can see, the *Fire Weapon* command is there. If we wanted the *Fire C* button (as it is in this case) to also fire the weapons in our hypothetical game, all we would do is click *Fire Weapon* and that would assign the same command to *Fire C*.

However, for now, click *New Macro* and you will see a small window like this.



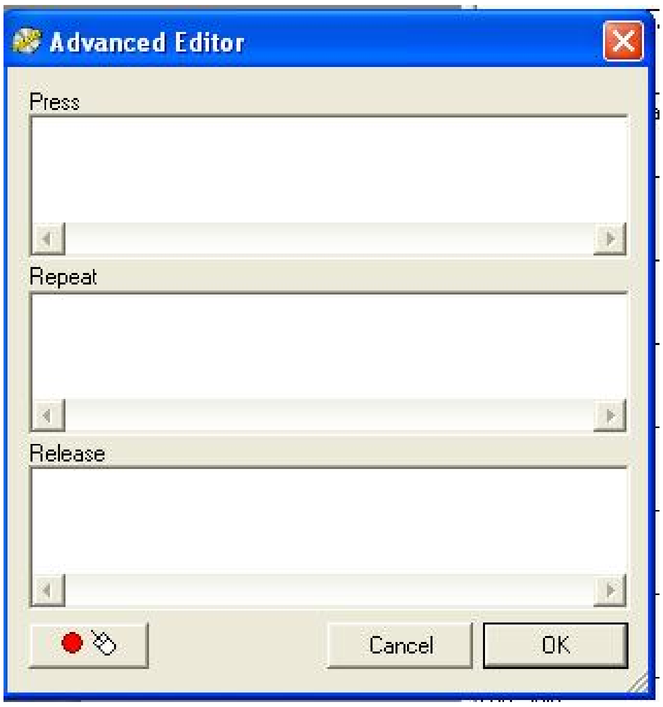
All you need to do now is input your keystroke sequence, exactly as you would input it in the game using the keyboard. As you press the keys you want you will notice a number appearing under each key in the Macro Recorder window. This represents the time, in seconds, since you pressed the first key in the macro sequence – therefore the first key in the sequence has 0.00 underneath it. In the example below you can see that there was a 4.56 second wait between pressing the P and S keys.



Click *OK* when you have finished inputting the macro sequence that you want. Just as with keystrokes, you will notice that the entire sequence of keys is shown in the bar for the button that you have just assigned this macro to. You can also rename this new macro, just as we did with the *Fire Weapon* command earlier.

Advanced Commands

Going back to the drop-down menu that appears when you right-click on a button line, click the *New Advanced Command* option. You will get the following window.

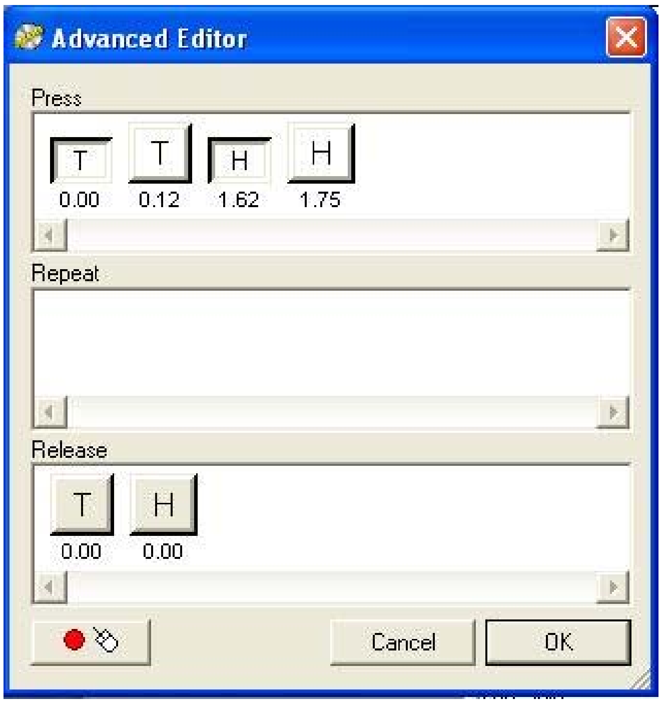


As you can see, this looks similar to the Macro Editor. Each window represents a different state of the button you are assigning the Advanced Command to. Any keystrokes in the Press column will happen when you press the button the command is assigned to. If you have multiple keystrokes you will need to keep the button held down until the commands have happened – this is exactly the way that normal multiple keystrokes work (ie, this doesn't function like a macro).

Any keystrokes that are in the repeat column will happen as long as you keep the button they are assigned to held down – again, these will only operate as normal multiple keystrokes.

Any commands in the release column will happen as soon as you release the button the Advanced Command is assigned to. However, the difference with the repeat column is that any multiple keystrokes will act like a macro.

You will notice that unlike the other keystroke input windows, when pressed each key places two instances of the key in the command input window – like below.



This is because when you press a key on a keyboard it actually produces two signals – one when you press it and another when you let go. In the above example you can see that the T key was held down for 0.12 seconds and the H key for 0.13 seconds.

Just like in the Macro Recorder, the numbers underneath the keystrokes represent the time in seconds from when you pressed the first key in the sequence; in the Advanced Editor you can adjust these timings. Simply point the mouse cursor at the time you wish to adjust and you will notice that the cursor changes to a clock with two arrows either side of it. Now click and hold down the left mouse button and drag the mouse right to increase the time or left to decrease. Once you are happy with the timing just let go of the mouse button and the new timing will be saved.

Therefore with the Advanced Editor you could set, for example, a command that launches a missile when you press the button. It might then switch to a camera view of the missile and then 4.5 seconds later switch to a camera view of the enemy. When you let go of the button it could switch back to your cockpit view.

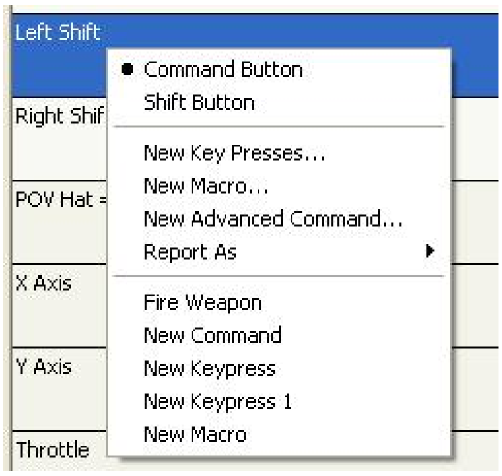
Obviously this is just a hypothetical example but it gives you an idea of the possibilities open to you with the Advanced Editor.

Shift Modes

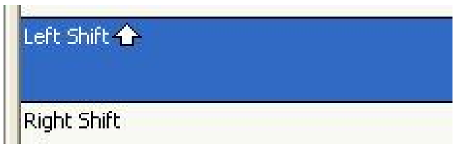
With the exception of the Cyborg 3D Force Stick, the controllers supported in this software release all have shift modes. This is a function that enables you to 'double up' the number of commands that you can assign to each button. For example, just as the period/full stop key on your keyboard can be made to print a '>' symbol when you hold down shift, you can assign a second command to each of the buttons on your controller.

On the Cyborg stick there are two shift buttons – the ones on the base of the stick with a double arrow symbol on them. On the X36 and X45 sticks, the shift button is the pinkie switch that your little finger rests on when gripping the stick.

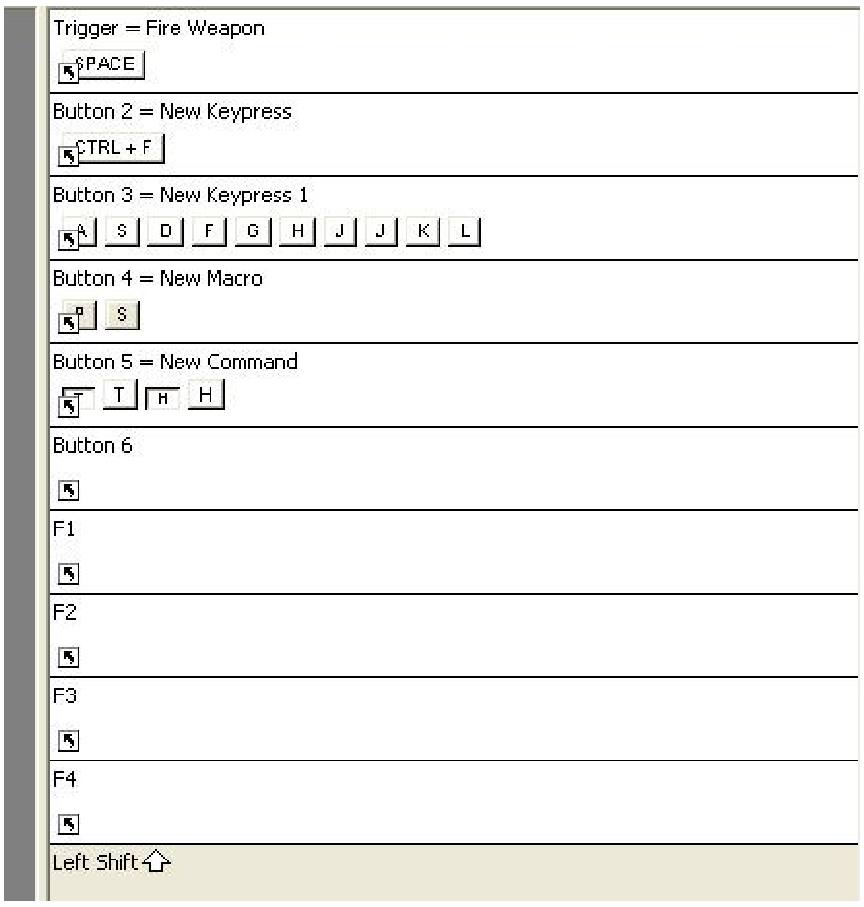
The process for enabling them to act as a shift button rather than a normal button is the same. Using the right click function again, point to the line referring to the shift button and right click.



You can see that there is an option at the top that isn't present on other buttons – the *Shift Button* option. If you click that then you can see that the programming software now reports that the shift button is acting as a shift button.



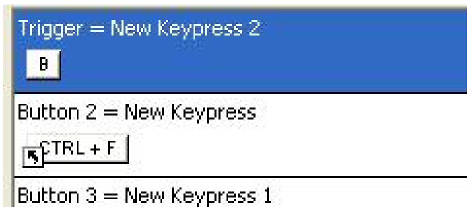
From here you have two choices to assign commands to your buttons in a shift state. At the top of the Profile Editor window you will see that there is a white bar that says *Shiftstate* next to it. In the bar should be the word, *Unshifted*. If you click that bar you can see that there are two other options – *Left Shift* and *Right Shift*. Click the *Left Shift* option and you will now be looking at the list of commands for the buttons when you have the left shift button held down.



Trigger = Fire Weapon
SPACE
Button 2 = New Keypress
CTRL + F
Button 3 = New Keypress 1
A S D F G H J J K L
Button 4 = New Macro
S
Button 5 = New Command
T H H
Button 6
F1
F2
F3
F4
Left Shift

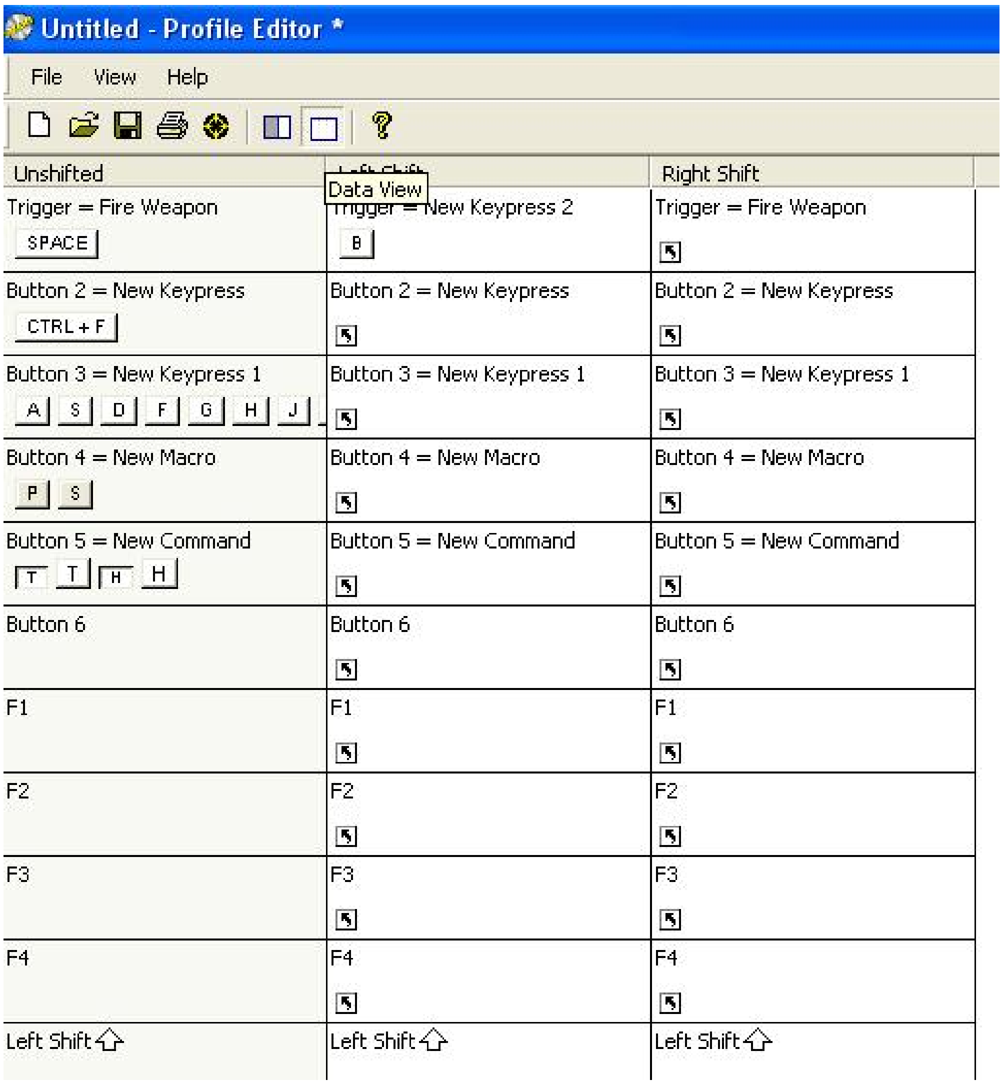
You can see that the commands created in the unshifted mode are carried over into this mode too – the little arrow in the box at the bottom left corner of each command line indicates this. You can now assign the commands you wish to the different buttons, exactly as you have been doing already. You will notice that when you enter a new keystroke onto a button the little arrow in the box disappears, indicating that this command only happens in this shift mode and isn't carried over from the unshifted mode. In the picture below you can see that we have put a B keystroke onto the trigger and that arrow is no longer there.

Now when we press the shift button on the stick when in the game, the trigger will give us a B keystroke rather than the space bar command that it does in the unshifted mode.



Trigger = New Keypress 2
B
Button 2 = New Keypress
CTRL + F
Button 3 = New Keypress 1

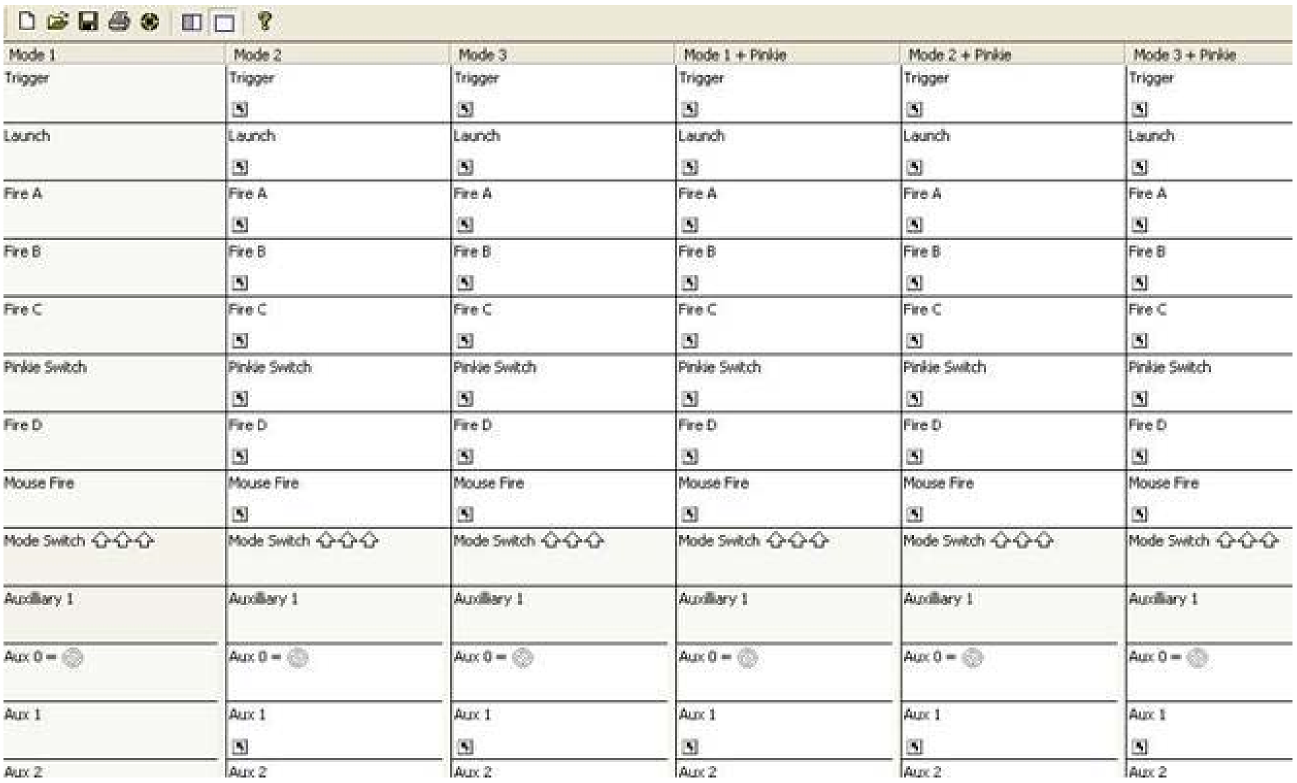
You can also assign a third set of commands to each button if you enable the right shift button as a shift button. The second way to input commands into a different shift mode is to switch the view of the Profile Editor to the *Data View*. The default mode with the 3D model of your controller on the left is called the 3D mode. To switch modes you simply press the Data View button at the top of the Profile Editor, as indicated in the picture below, and you will see the same view as in the picture.



Unshifted	Left Shift	Right Shift
Trigger = Fire Weapon SPACE	Trigger = New Keypress 2 B	Trigger = Fire Weapon
Button 2 = New Keypress CTRL + F	Button 2 = New Keypress	Button 2 = New Keypress
Button 3 = New Keypress 1 A S D F G H J	Button 3 = New Keypress 1	Button 3 = New Keypress 1
Button 4 = New Macro P S	Button 4 = New Macro	Button 4 = New Macro
Button 5 = New Command T T H H	Button 5 = New Command	Button 5 = New Command
Button 6	Button 6	Button 6
F1	F1	F1
F2	F2	F2
F3	F3	F3
F4	F4	F4
Left Shift⇧	Left Shift⇧	Left Shift⇧

This enables you to see what commands you have assigned to each button in all modes. You can also input keystrokes, macros etc. in this view, exactly as you did in the 3D view.

For the X36 and X45 users, note that while these controllers have only one shift button you do have the three different modes made available to you by the mode switch on the throttle. These are programmed in exactly the same way as the Cyborg – in fact it’s probably best to use the Data View for these two controllers because, as you can see, it’s far easier to keep track of which commands you have programmed.

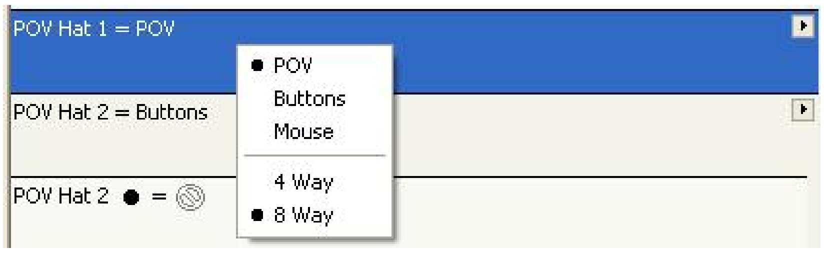


Mode 1	Mode 2	Mode 3	Mode 1 + Pinkie	Mode 2 + Pinkie	Mode 3 + Pinkie
Trigger	Trigger	Trigger	Trigger	Trigger	Trigger
Launch	Launch	Launch	Launch	Launch	Launch
Fire A	Fire A	Fire A	Fire A	Fire A	Fire A
Fire B	Fire B	Fire B	Fire B	Fire B	Fire B
Fire C	Fire C	Fire C	Fire C	Fire C	Fire C
Pinkie Switch	Pinkie Switch	Pinkie Switch	Pinkie Switch	Pinkie Switch	Pinkie Switch
Fire D	Fire D	Fire D	Fire D	Fire D	Fire D
Mouse Fire	Mouse Fire	Mouse Fire	Mouse Fire	Mouse Fire	Mouse Fire
Mode Switch	Mode Switch	Mode Switch	Mode Switch	Mode Switch	Mode Switch
Auxiliary 1	Auxiliary 1	Auxiliary 1	Auxiliary 1	Auxiliary 1	Auxiliary 1
Aux 0 =	Aux 0 =	Aux 0 =	Aux 0 =	Aux 0 =	Aux 0 =
Aux 1	Aux 1	Aux 1	Aux 1	Aux 1	Aux 1
Aux 2	Aux 2	Aux 2	Aux 2	Aux 2	Aux 2

Programming Hat/POV (Point of View) Switches

All of the controllers supported by SST have a Hat, or POV switch. On the joysticks and pads a POV switch is usually used as a device for looking through the different views from a cockpit in a flight simulator, but you can assign it to do whatever you want. On the GM2/GM3 it is very important as it's the switch that movement commands are assigned to for first person shooter games like Medal Of Honor.

If left unprogrammed, the POV switch will just act as either a default POV or, in the X36/X45's case, as a set of four buttons, depending on in which direction it is pushed. This is easily changed and to start programming the POV simply right click on it or click the little arrow icon next to it in the profile editor and you will get the following drop-down menu.



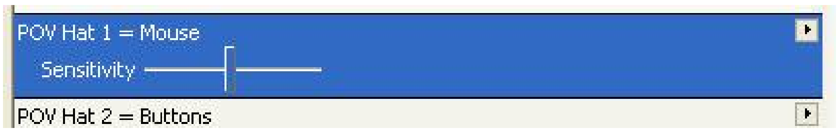
POV Hat 1 = POV

POV Hat 2 = Buttons

POV Hat 2 ● =

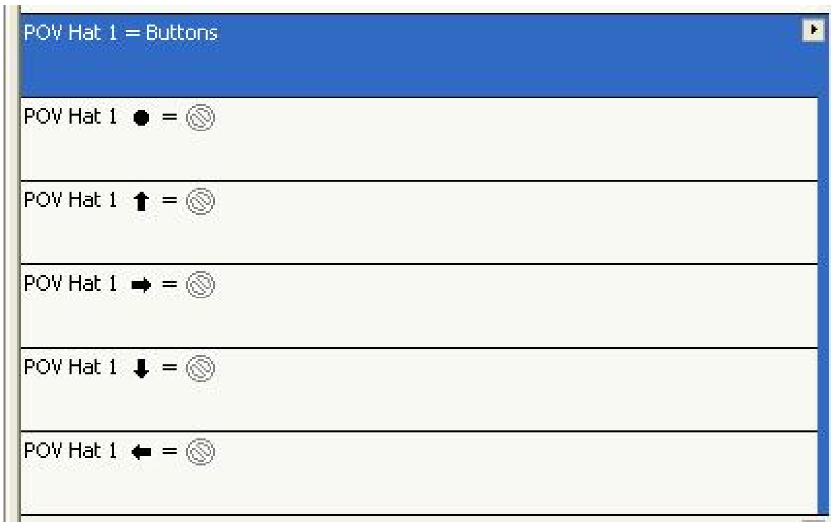
You can see that you have three possible functions for the POV switch, the default option for which (POV) is currently selected. The other two options are to set it to operate as buttons or to emulate the mouse. You can also set whether the POV operates in 8-way or 4-way mode here. For Mouse and POV mode it's recommended that you leave it as 8-way but for buttons mode it's definitely recommended to set it to 4-way, simply because it can be very hard to select the diagonal positions uniquely in the heat of gameplay (the only exception to this is if you are configuring the hat to move something in 8 directions, like you would with the GM2/GM3 for a first person shooter game).

Setting it to *Mouse* will simply give you a Sensitivity slider bar like below:


POV Hat 1 = Mouse
Sensitivity
POV Hat 2 = Buttons

And adjusting that will simply change the speed of the mouse cursor when you push the POV in the appropriate direction, slow being to the left of the scale and fast being to the right.

However, clicking the *Buttons* option adds more options to configure, one for each position of the hat switch.


POV Hat 1 = Buttons
POV Hat 1 ● =
POV Hat 1 ↑ =
POV Hat 1 → =
POV Hat 1 ↓ =
POV Hat 1 ← =

It’s then simply a case of programming each position of the hat with the keystroke, macro or advanced command that you wish to assign, in the same manner that you program the buttons. Note that the centre position of the POV hat should generally be left unprogrammed otherwise the POV will continually issue any command that you have assigned to it when it is at rest in its centre position.

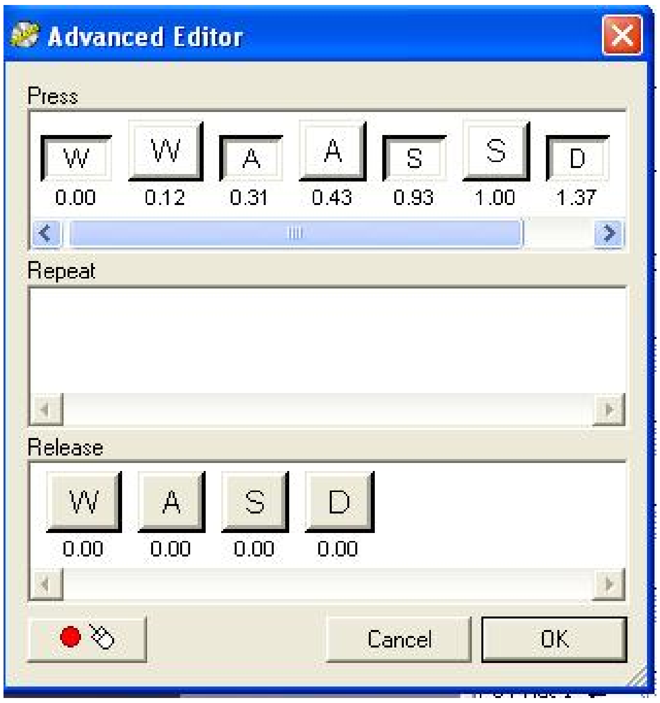
One slight problem with programming the hat is that you may end up with a problem with ‘sticky keys’. This is where a programmed function on a hat position doesn’t ‘let go’ when you release the hat and the computer thinks that the assigned keystroke is still being issued. This can usually be solved by using the tips in the following section, which also describes how to configure the POV on the GM2/GM3 for movement.

Hat Programming Tricks

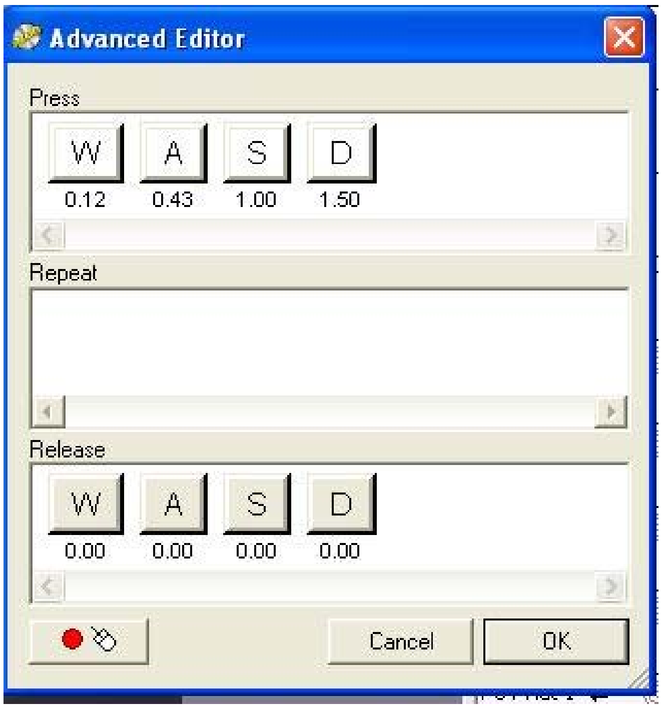
One thing that you can do to ensure that the hat ‘lets go’ of a command assigned to one of its directions is to program the centre position of the hat with a ‘release’ command for any of the keystrokes assigned to any of its positions. This sounds complicated but is actually quite straightforward. As an example we will take the programming of the GM2/GM3’s hat switch for movement in a first person shooter game.

For these games, movement is usually controlled by four keys on the keyboard, typically W, A, S and D (respectively forward, step left, backward and step right). What may happen if you just assign simple keystrokes to the appropriate directions is that when you switch from moving forward to backward the software may ‘forget’ to release the forward key command, leaving you constantly walking forwards. To stop this from happening assign a ‘release key press’ to the middle position of the POV using an advanced command and then make sure that there is a release key press for every one of the keys that are used by any of the hat positions. Then when you return from any hat position it will release that key press.

To do this, right-click on the POV middle position option and click *New Advanced Command*. Now enter all the keys that are in the other positions of the hat switch into the press row and you will end up with something like this:

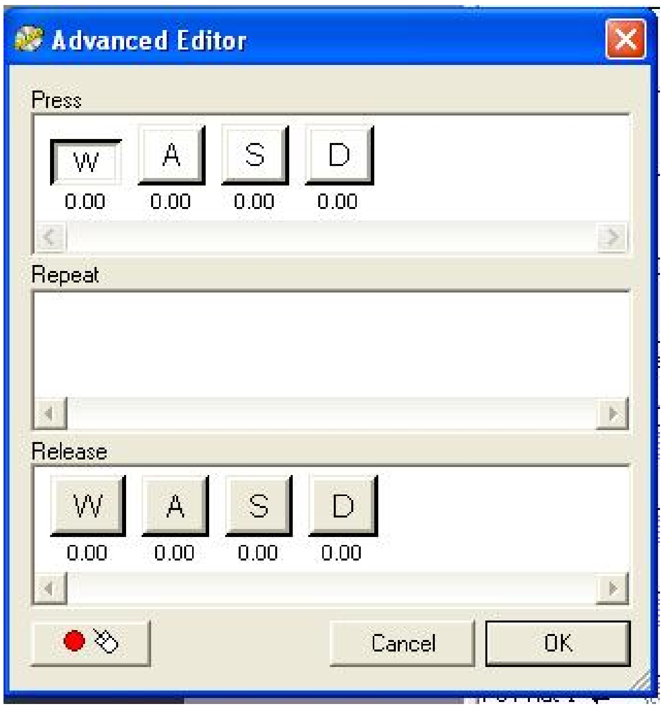


Now simply click on each of the press instances of the keys in the *Press* row (in the above example these are the W’s with the 0.00 underneath, the A’s with the 0.31 underneath etc), then right-click on them and click *Delete* from the drop down list of options.

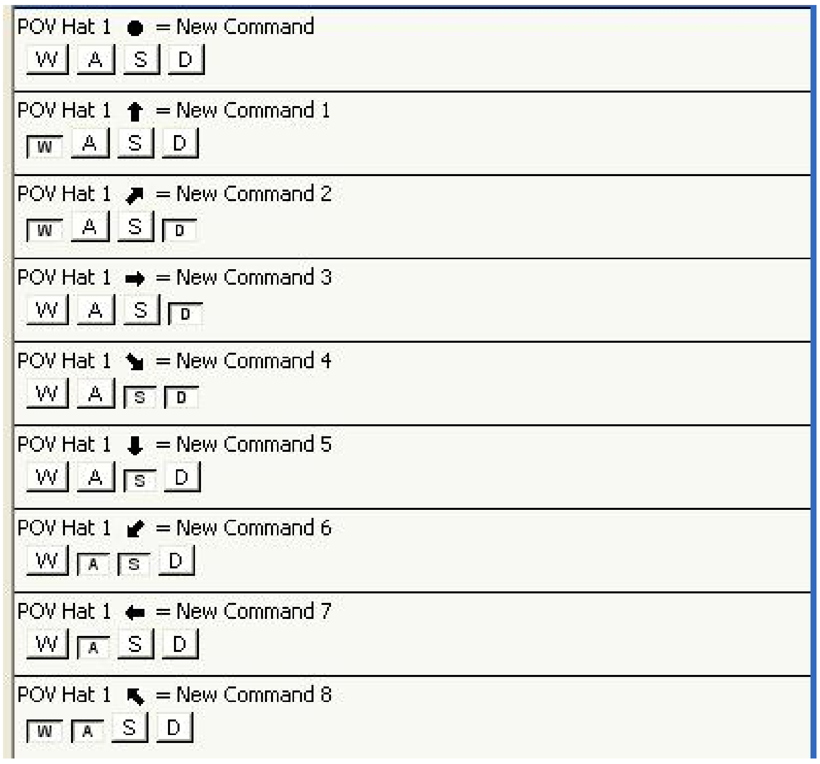


Before you click *OK* to finish make sure that you right-click on the keys in the Press row, point to *Quantize Time* and then select 0.00. This will have the effect that when you let go of the POV and it returns to its centre position it will issue a ‘release command’ for those keys to ensure that the program thinks that those keys have been ‘let go’.

To complete this you must program each direction of the hat with the appropriate press commands for that direction, as well as release commands for the other positions of the POV. This would look something like the following for the forward command on the hat of the GM2.



Ultimately it should look like the following (this is how a GM2 hat switch looks when configured for a first person shooter game):



POV Hat 1 ● = New Command
W A S D

POV Hat 1 ↑ = New Command 1
W A S D

POV Hat 1 ↗ = New Command 2
W A S D

POV Hat 1 → = New Command 3
W A S D

POV Hat 1 ↘ = New Command 4
W A S D

POV Hat 1 ↓ = New Command 5
W A S D

POV Hat 1 ↙ = New Command 6
W A S D

POV Hat 1 ← = New Command 7
W A S D

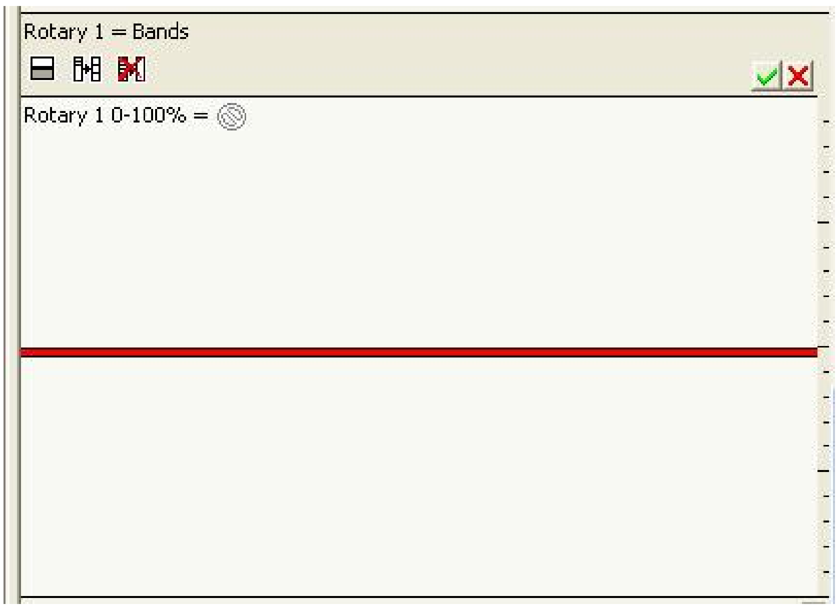
POV Hat 1 ↖ = New Command 8
W A S D

Axis Programming (rotaries, thumb wheels etc)

The final parts of your controller that can be programmed are the axes. These are basically anything that has a range of movement on your controller, be it the joystick, a throttle, rudder, thumb wheel (GM2/GM3) or a rotary (X36/X45).

You can program the axes on your controller with keyboard commands. Generally speaking, the main axes of a joystick should be left to work as an axis, simply because most games detect these by default anyway. However, some games don’t support joysticks at all (so you could then configure the stick to pretend to be a part of the keyboard) and this would allow you to make the stick work in the game where otherwise it wouldn’t. GM2/GM3 and X36/X45 owners will mainly use the axis programming for the thumb wheel and rotaries.

To start programming an axis we first have to set it to banded mode. This is done by right-clicking on the appropriate axis (or left clicking the small arrow on the right of the axis listing) in the Profile Editor and choosing *Bands* from the drop-down list. You will see something like this:


Rotary 1 = Bands

Rotary 1 0-100% =

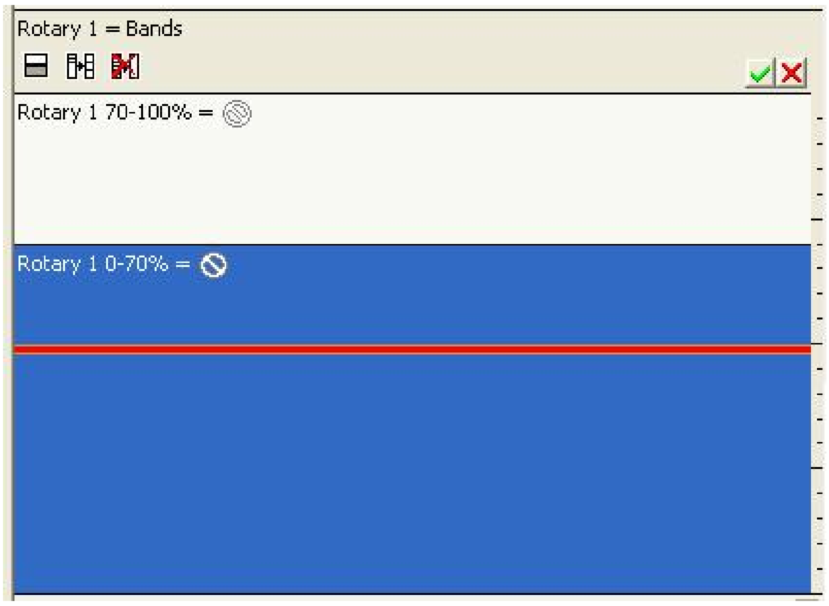
Your mouse cursor will change so that it looks like a horizontal line with a vertical, double-headed arrow running through it – don’t click anything yet!

The red line in the window represents the current position of the axis that you are programming. In this case it is the rotary 1 on the X45 and if you move that axis on the actual controller it will change the position of the red line.

What we are going to do is split the axis up into banded areas so that we can then assign keyboard commands to those areas.

In the above example we are just going to create a simple bit of programming where moving the rotary in one direction will give us an ‘A’ key press and moving it in the other will give us a ‘B’ key press. First we have to create our banded areas and this is done with the mouse. That horizontal line with the arrow running through it indicates that where you next click in that rotary area will set a ‘split point’.

You can see in the picture below that we have created a split point at the 70% mark on the axis’ scale.

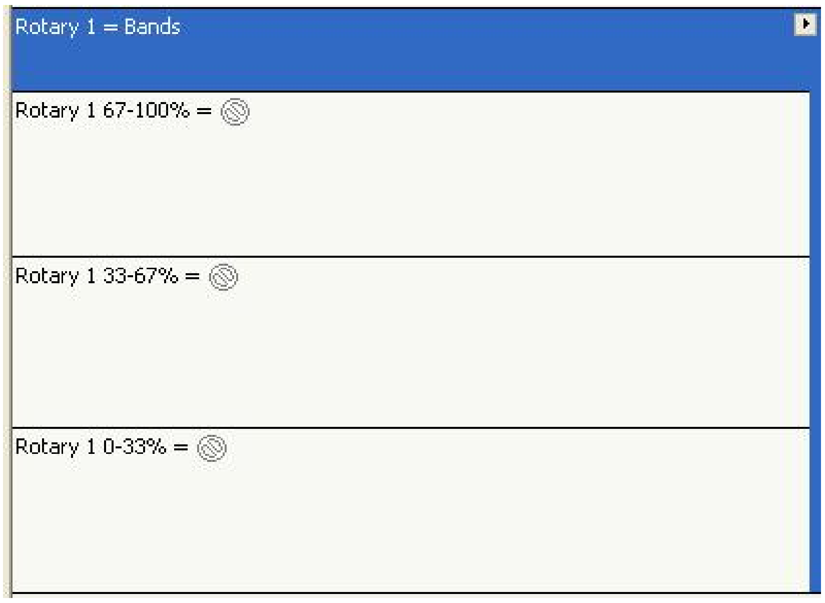

Rotary 1 = Bands

Rotary 1 70-100% =

Rotary 1 0-70% =

We could also have done this by actually physically moving the axis to that point (so that the red line was at that point) and then clicking the left icon just underneath where it says *Rotary 1= Bands*. We should now also create a second split and then click the middle icon which spaces the areas out evenly.

We can then click the green tick to set those split positions in place. This leaves us with something like this.


Rotary 1 = Bands

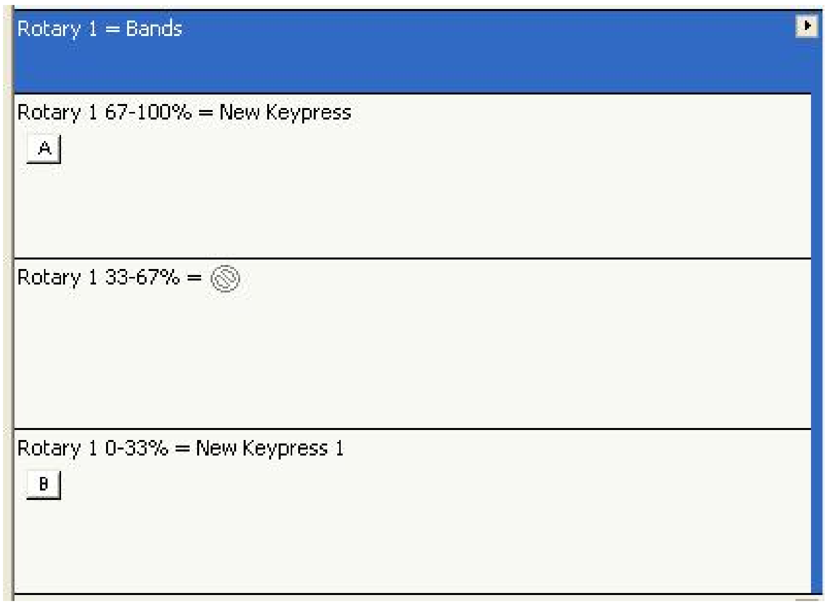
Rotary 1 67-100% =

Rotary 1 33-67% =

Rotary 1 0-33% =

Please note that you cannot start entering keystrokes into the band until you have saved your programmed bands by clicking on the green check mark. You can then program keystrokes, macros or advanced commands to the areas just like any button on the controller.

Using our ‘A’ and ‘B’ example above we get the following.


Rotary 1 = Bands

Rotary 1 67-100% = New Keypress

A

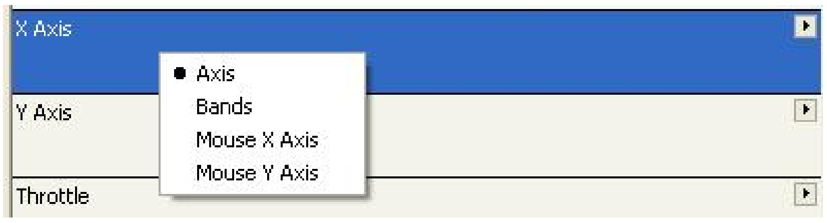
Rotary 1 33-67% =

Rotary 1 0-33% = New Keypress 1

B

You may wonder why we didn’t just create one split so that the area was split in half and then just assign the keystrokes to the two areas? It’s simply because you need a position where the axis is ‘at rest’ and isn’t issuing any keyboard commands – you will notice that no commands are assigned to the middle position of the area in the above example. No matter which controller you have or which axis you are programming, you must always have a band over the middle of the axis movement with no command assigned to it. You can create as many splits as you want but too many can become very complicated.

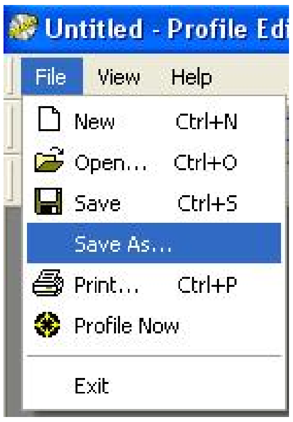
Mouse Emulation
You can also program an axis to emulate the movements of the mouse cursor if you wish. To do this simply right click the axis you want to become a mouse movement and you will get the following menu:



Clicking the *Mouse X Axis* option will make that axis control the left/right movement of the mouse and the *Mouse Y Axis* option makes it control the up/down movement. Once selected you will see a slider bar which controls the sensitivity of the mouse movements; the left-most setting is the slowest and the right-most setting the fastest.

Saving the Profile

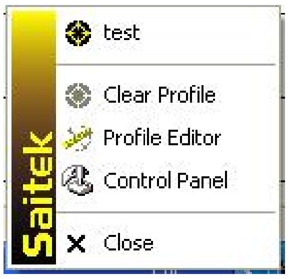
Once you have finished assigning all the commands you want you must save the profile so that you can access it again later. Simply click *File* at the top of the Profile Editor and then click *Save*.



Just as when you save a document in on your computer, the Profile Editor will ask you where you wish to save the profile and what you wish to call it. Do not change the location of the profile – it must be saved in the directory that is already in the save window. The name of the profile should ideally be the name of the game that you are making this profile for. Once you have done this, click Save to save the profile.

Activating the Profile

Activating the profile couldn't be easier. Simply right click on the little joystick icon next to your clock and you will now notice that the profile that you saved is at the top of the pop-up menu. In the example below you can see that we called our profile 'Test'.



If you then click your profile you will notice that the little joystick icon now has a green square behind it, indicating that a profile has been loaded into the controller.

At any point, if you wish to clear the profile from your controller, simply right click the joystick icon and click *Clear Profile*. The green circle will disappear indicating that the controller is now cleared of any commands.

Testing the Profile

Once you have activated the profile you can test it. A good way to do this is to open up Wordpad in Windows – when you then press the buttons on your controller the assigned keystrokes will appear in Wordpad. Please note that this will only work with basic keys that you would normally type into a word processor ie, letters, numbers and punctuation. The function keys (F1, F2 etc) shift, alt and ctrl keys won't print anything in a word processor either. We will be including a profile tester in a later release of the programming software so that you can test all keystrokes that you input into your profiles.

Printing the Profile

Once everything is programmed into your pad you may find yourself forgetting which commands you have assigned to each button, especially if you have used shift modes. Therefore we have included a printing facility in the software which produces a couple of pages showing the buttons/hats/axes of the controller and which commands are assigned. This will only be useful if you have named you commands as described in the simple keystrokes section of this guide.

To print your profile out, simply load it into Profile Editor, click *File* at the top of the editor window and then *Print*.

Product Specific Features

X36/X45
The X36 and X45 flight controllers include several unique features that are nonetheless programmable using the same principles as functions on the Cyborg. The mode switch simply allows you to program all the other functions of the controller so that they can do something different depending on which mode the switch is in. This is easily programmed using the Data view for the Profile Editor, as described earlier in the manual. Once in the Data view you will see that each button/POV/axis has a keystroke entry box for Mode 1, Mode 1 +Pinkie, Mode 2, Mode 2 + Pinkie, Mode 3 and Mode 3 + Pinkie.

The pinkie switch is the X36/X45's shift button and has to be set to work as a shift button before those '+Pinkie' commands are usable. The rotaries are an axis and are programmable in the manner described in the axis programming section of the manual. The Aux switch is just a series of three buttons in a row, programmable in exactly the same way as any other button on the controller. There are a couple of notes to remember when it comes to assigning commands onto the Aux switch positions. First, any command assigned to an Aux position will be 'played' continually as long as the Aux switch remains in that position. This will become even more complicated if that command in any way involves the Ctrl, Alt or Shift keys on the keyboard; if you have an Aux position continually issuing a command with a Ctrl key in it then it will modify any other keypress output by any other button on the controller, creating all kinds of unintended commands. The best practice is to leave the middle position of the Aux switch free of any command and to then assign commands to the other positions. When you move the switch to initiate those commands, always ensure that you return the switch to the middle position so that the Aux switch is 'at rest'.

P880, P2500 and P3000
The new pads have several features that are especially important to console users (Playstation 2, X Box and Gamecube), who use these pads to play first person shooter games like Medal Of Honor, 007 Nightfire and Unreal Tournament 2003. On consoles, the sticks on the pads are used for movement and looking around; however, most PC games of this type don't include support for game controllers at all, or if they do then it's fairly basic. With the SST software you can set up the sticks with the equivalent keyboard and mouse commands from these games to make them work as they do on consoles. This is covered in the Axis Programming section of the manual or you can find profiles for your pad for these games in the profile download section on our website. As these type of games typically use the same keys for movement (W, S, A and D), then even if there isn't a profile for your particular game, you can download another one and adapt if for use with your game using the Profile Editor.

The pads do feature a button that is entirely unique to them - this is the red Smart button just next to button number 2. This button can have three functions; the first is for it to be programmed as a normal button; the second is for it to become a shift button like that on the Cyborg so that you can then assign a second command to the other buttons for a shifted mode; and the third is to set it as a Smart Button.

If set as a Smart button, you can program in keystroke sequences to the other buttons on the pad 'on the fly' without having to use the profile editor. The process for doing this is simple. Open up the Profile Editor and right click on the *Smart Button* entry in the list of buttons. Choose *Smart Button* from the drop-down list of options and then save the profile (call it what you want to) and load it into the pad.

To program a keystroke (or set of keystrokes) ‘on the fly’, simply press the Smart button on your pad, press the button you want to assign the keystroke(s) to and then press the key(s) on the keyboard. To save this, press the Smart button on the pad again. Those keystroke(s) should now happen when you press the button you programmed. This can be done in-game or in Windows.

PC Dash 2/P8000
The PC Dash 2/P8000 has no unique features - it has a shift button, a POV switch and the rest of the buttons on it are programmable in the same way as buttons on the Cyborg. However, the Dash 2 does have an ‘unlabelled’ second shift button, or rather one of the buttons can be configured to be a second shift button, therefore enabling another set of commands to be assigned to the rest of the features on the Dash. The button that can be set as a shift button is the one in the bottom left corner of the face of the Dash - E1.

If you have any queries or questions regarding this version of the Saitek Smart Technology (SST) software, please contact your local support representative or go to the user forum on our web site: http://www.saitek.com