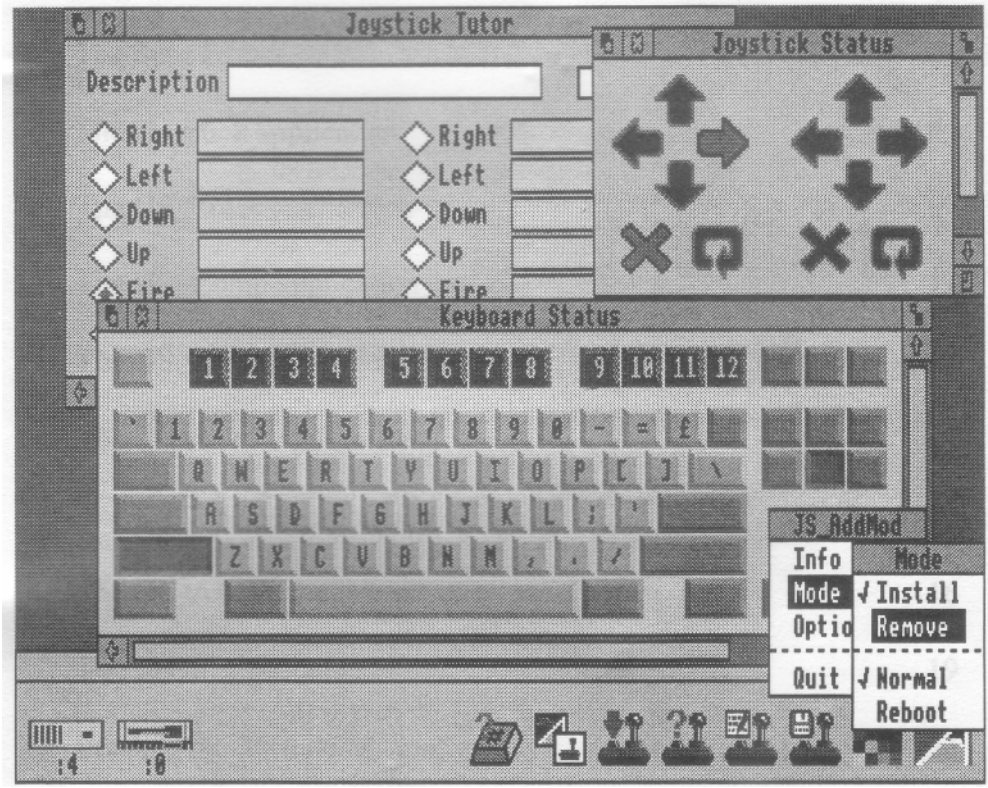
THE SERIAL PORT

JOYSTICK INTERFACE

The Joystick Interface

Release 2

Version 2.02



Vertical Twist
11-14 White Hays North
Westbury
Wiltshire
BA13 4JT
Tel: 0373 824200 Fax: 0373 824300

Contents

Introduction

The Serial Port Joystick Interface is a hardware/software package which allows one or two standard digital joysticks to be used with any of the Archimedes range of computers, including the A3000.

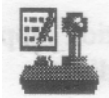
The hardware consists of a small interface box which plugs into the Archimedes' printer port, the printer is then plugged into the back of the interface. This allows the joystick interface to be fitted without having to take the computer apart or make any other purchases such as an expansion box or backplane - it can also be moved between computers easily.

The software consists of a relocatable module to control the joystick interface and a set of Risc OS applications to control the module's behaviour. The module allows joysticks to be used with programs which normally use the keyboard and/or mouse. It can be "programmed" to simulate sets of mouse movements and key presses when the joystick is operated. The " programming" of the module is performed using the Risc OS applications and the resulting joystick programs are stored in special files of type "JoyStick".

The functions of each program are :

Joystick - The control module hidden inside an application.

JS_Tutor - Creates simple joystick programs by learning key presses and joystick movements.



JS_Comp - Compiles joystick programs from text files, allowing more complicated descriptions to be created than with JS_Tutor.

JS_Test - Displays the current status of the joystick(s), useful when creating complicated programs with JS_Comp.

Key_Test - Displays the current status of the keyboard, also useful when creating complicated programs with JS_Comp.

JS_Update - Updates joystick programs created with earlier versions of the joystick software, allows a one line description to be added to programs.

JS_AddMod - This allows the module to be added to your favourite games so that it is loaded automatically with the game.

The distribution disc will also contain the following files/ directories :

ReadMe - If present, this text file will contain information on any changes to the hardware and/or software since this guide was published.

Source - This directory contains a set of pre-written text files for joystick programs for many games and some programming examples given later in this guide.

Object - This directory contains the compiled joystick programs from the Source directory, for information on each program read the corresponding text file.

To read any of the text files use the !Edit application supplied with your Archimedes or A3000 on applications disc 1.

Choosing a joystick

Surprisingly enough, before you can use the Joystick Interface you need either one or two joysticks. The Joystick Interface can be used with any digital joystick conforming to the Atari/Commodore standard - by far the most common type available. Joysticks of this type simply contain five switches, four for the directions and one for the fire button.

(Many joysticks appear to have more than one fire button or trigger but they act as one switch and are provided for left or right handed use.)

Some models of joystick provide features in addition to the Atari/Commodore standard, such as "rapid-firing" so you don't have to keep romping the fire button. These features are not guaranteed to work with The Serial Port Joystick Interface - you should check that they can be switched OFF before making a purchase. This is not a disadvantage because most features can be provided using appropriate joystick programs, including rapid firing. This means that you can save your pennies and buy a cheaper joystick without losing out.

'Mere are several different styles of joystick on the market, the most common of which are the "pistol-grip" and "ball-ona-stick" varieties. Choosing between them will always depend on personal taste but a few points are worth remembering.

The ball-on-a-stick models are traditionally used for maze type games or those which require the best control. Both hands are required to operate one such joystick, one to move the stick and the other to hold it down and press the fire button on the base.

The pistol grip models have triggers or fire buttons on the top of the stick which can be operated with the thumb, they often

have suckers underneath so the joystick can be operated with just one hand. This leaves the other hand free to either press additional keys required by the game or to operate a second joystick. This makes games such as flight simulators much easier to use, one joystick could be used to control the plane while the other could be used for the throttle and rudder pedals.

Far more important than a joystick's styling, colour scheme and number of flashy features is its basic quality of construction. Some joysticks are very flimsy making games difficult to play and may break after a few hours use. Before parting with any money examine the actual joystick (rather than reading the description on the box) and ask yourself "how easy would it be to break this if I really wanted to '?". Some features to look out for are :

• Microswitches which you can actually feel going "click" as you move the stick or press the fire button. These provide the best feedback and are the least likely to break.

• Metal construction, such as a metal shaft running down the centre of the stick to provide strength.

Plugging in the Joysticks and Interface

The Joystick Interface plugs into the socket labelled "printer" on the back of your Archimedes or A3000, the interface will only fit one way round so there is no danger of plugging it in incorrectly. If you have a printer (or any other device which plugs into the printer port) then it can be plugged into the back of the Joystick Interface. A small switch on top of the interface chooses between joystick operation (away from the back of the computer) and normal printer operation (towards the back of the computer).

If you find that having the Joystick Interface and other devices plugged into the printer port cause a problem because they stick out too far then get in touch with us as we can supply, for only a few pounds, a cable that will allow you to put the interface somewhere more convenient.

The two joysticks are plugged into the top of the interface. The joystick control software refers to the left and right sockets as joysticks one and two, when sitting in front of the computer looking (or groping) over the top.

When the switch is in the normal printing operation position (towards the computer) then the Joystick Interface will have no effect on the printing. The joystick control module is smart enough to recognise when the switch is in this position and will behave sensibly. It can be instructed to output a warning message and/or wait until the switch is put into the joystick position.

However if you attempt to print something when the switch is in the joystick position then the printer drivers may become confused (after sending some data to the printer and not receiving a reply) and may not be able to print- until the computer is reset.

The same applies to any non-standard devices plugged into the back of the joystick interface such as software protection " dongles", it may be necessary to reset the computer or switch it off and on again before normal operation can be resumed.

To check that the Joystick Interface is functioning correctly load the !JS Test application as described on page 11. The various icons should light-up correctly as you move the joysticks around and press the fire buttons. If they don't then make sure that the switch is correctly positioned before getting in touch with us.

Now that you know the interface is functioning you can try and get it wain with your games - the easiest way is by using the ' Start Files' as described in the next section. If you can't find your game in this list then read the rest of the manual for instructions on how to make your own module.

The Start Files

The Start Files' are the easiest method of getting the Joystick Interface to work with many of the popular games. They will automatically load up the joystick control module and then prompt you to select a game control module before loading in the game for you.

To use them double-click on the Start directory and look through the list to see if the game you want to use is listed. If it then double click on the file and then follow the on screen instructions. You may be offered a choice of game control modules complete with a short description of what they do or it may go straight on to the stage where it asks you to insert the games disk. Once you have done this the game will be loaded when you press a key and you will be able to control it using your joysticks.

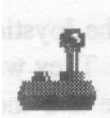
A complication arises with a few games that reset the computer half way through the loading (a process known as "rebooting"). For these games you will be asked to hold down the shift key during the loading. If you do this then the loading will stop half way and you must put the joystick software disc back into drive 0 and run the same 'Start' file again.

This may be done by double-clicking in the desktop or, if you are really unlucky and not in the desktop anymore, by typing *START.XXX at the '*' prompt where XXX is the name of the ' Start' file.

This may sound complicated right now but it isn't too difficult after a couple of goes.

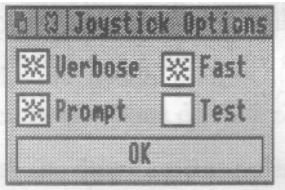
The !ReadMe file in the Start directory gives further details on the 'Start Files' - including how to create your own. To read it use !Edit as supplied on Applications Disc 1.

The !Joystick application



The !Joystick application contains the joystick control module as described on pages 19 and 20. It is contained within an application so that the application itself and the joystick program files have appropriate icons and also so that the desktop knows what to do when you double-click on such a file (it will make sure that the module has been loaded into memory and will then make the module load the file).

When you double-click on it you will be presented with a small window as shown below from which you can select which options will be active - see later for further details.

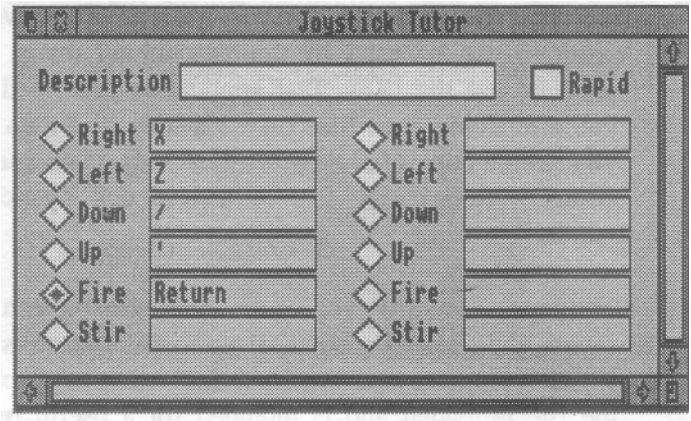


You should not often have to load the module manually as the various applications and start files will automatically load it if necessary.

In order reach the module inside the application double click on it while holding down the shift key.

The !JS_Tutor application

The JS_Tutor application allows simple joystick programs to be created quickly and easily by teaching by example. Double click Select on the application icon in a directory viewer to install it on the icon bar. Then click Select on the application icon on the icon bar to open the main JS_Tutor window :



If you have the joystick interface plugged in and switched on then hold down the fire button on the joystick. This will select the radio icon labelled "Fire" in the window. (If you don't have a joystick handy then you can use the mouse to select the radio icon as usual.) Next hold down the space bar. This will make the text "Space bar" appear in the box next to the radio icon. You have just taught the application that you want to use the fire button instead of the space bar.

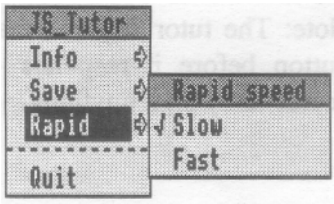
Note: The tutor requires that you press and hold down the button before it responds - this is to ensure you do not accidentally alter anything.

You now program another key either by clicking on the appropriate icon, or performing the action you wish to program next on either of the joysticks, and then pressing the key you need on the keyboard.

Once you have programmed as many keys as you like you can save the program in the normal Risc OS fashion. Click Menu on the application icon on the icon bar to open the main menu and move to the save window, enter a filename and drag the icon to a directory viewer. The saved joystick program can be used by double-clicking on it, to test the program use the !Key_Test application described below. (If you are saving your files on a separate disc then make sure you keep a copy of !Joystick on each disc you use - this will ensure you can simply double-click on any joystick data files you have saved.)

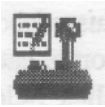
The input icon at the top of the main window can be used to save a one line description, of up to 64 characters, inside the joystick program file. This description can be displayed when the file is loaded and is intended for a useful reminder eg. "Frogsoft football, two player mode".

The option icon in the top right allows rapid firing to be enabled. When this has been selected then holding down a fire button on the joystick will have the effect of pressing and releasing the corresponding key very quickly. Two speeds of rapid firing are available and can be selected from the main menu - different games run at different speeds and may benefit from different rates of rapid firing.

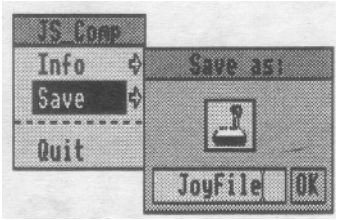


To delete a programmed key click Select on the box containing the key name.

The !JS_Comp application



The JS_Comp application allows much more complicated joystick programs to be developed by compiling text files containing the program in a special language described in a later section. Double click Select on the application icon in a directory viewer to install it on the icon bar. To compile a text file drag it from a directory viewer (or save it directly from another application such as !Edit) onto the JS_Comp application icon on the icon bar. After a brief pause a window will open from which the compiled joystick program can be saved.

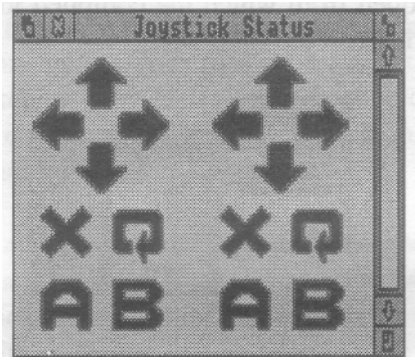


If there are any errors in the text file then the compilation will stop and an error window will give an explanation together with the number of the bad line in the text file.

The compiler is case insensitive and ignores white space between code words. Lines beginning with a "I" are also ignored and this character can be used to insert comments in the source text file. If the very first line in the text file is a comment then it will be saved inside the compiled joystick program file. This comment may be displayed when the file is loaded and is intended for a brief description/reminder.

The !JS_Test application

The JS_Test application displays the current status of both joysticks and is intended for use in "debugging" complicated joystick programs alongside JS_Comp. Double click Select on the application icon in a directory viewer to install it on the icon bar. Then click Select on the application icon on the icon bar to open the main window. Moving either joystick will highlight the corresponding direction icons in the window.



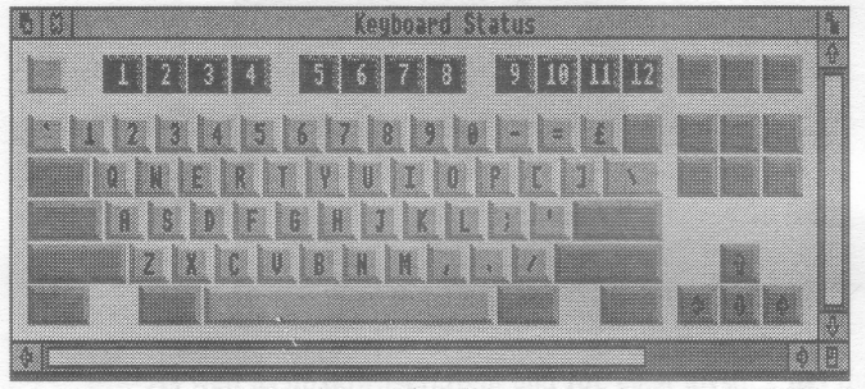
As well as up,down,left,right and fire each joystick has three other icons displayed as a circular arrow, a letter A and a letter B. The circular arrow can be highlighted by "stirring" the stick quickly. This effectively provides a sixth switch which can be used by JS_Tutor or JS_Comp just like the other five switches. " Stirring" takes some practice but can be invaluable when game playing for quickly releasing a weapon of awesome destruction rather than having to dive for the correct key.

(The module's stir sensitivity can be reduced to allow stirs to be performed much more easily, but it may become possible to stir accidentally. See the sections on JS_AddMod and the joystick module for details.)

The A and B icons display the status of hidden "flags" stored inside the joystick module. These can be used to provide many useful features such as rapid firing. See the sections on the joystick module and the joystick programming language for details.

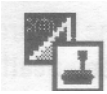
The !Key_Test application

The Key_Test application displays the current status of the keyboard and is intended for use in "debugging" complicated joystick programs alongside JS_Comp. Double click Select on the application icon in a directory viewer to install it on the icon bar. Then click Select on the application icon on the icon bar to open the main window. Pressing any key on the keyboard (or any simulated key using the joystick) will highlight the corresponding key icon in the window.



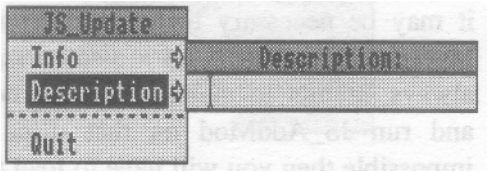
This application is very useful for testing joystick programs where moving a joystick in one direction affects more than one key. Without the Key_Test application the only way to test that a joystick program worked correctly would be to run the game.

The !JS_Update application



The JS_Update application is provided to allow users of previous versions of the joystick control software to update their data files. Double click Select on the application icon in a directory viewer to install it on the icon bar. Then drag a program file from a directory viewer onto the application icon on the icon bar to update it. The file will be checked for validity then it will be changed to the correct filetype to be recognised by the desktop.

JS_Update also allows comments to be added to joystick programs. This allows existing users to add descriptions to their existing uncommented programs and also allows the descriptions to be changed in program files generated using the current version of the software. To enter a comment click Menu on the application icon on the icon bar and move to the description submenu, up to 64 characters can be entered.



The !JS_AddMod application

The JS_AddMod application allows you to add the control module to your games discs so that it is loaded automatically when you play the game. Double click Select on the application icon in a directory viewer to install it on the icon bar. To add the module to a game with a Risc OS application icon just drag it from the directory viewer onto the JS_AddMod application icon on the icon bar. A window will appear asking for confirmation, click Select on the OK icon and the module will be added. Finally a new directory viewer window will open, you should save an appropriate joystick program file for the game into this directory.

IMPORTANT NOTICE

You should NEVER use JS_AddMod on an original games disc because something may go wrong rendering the disc useless. It is always good practice to make a backup copy of original discs - this may prove difficult with certain discs and it may be necessary to use a disc utility program such as Investigator that includes a disc backup facility. You should always attempt to make a working copy of the games disc and run JS_AddMod on that disc. If this proves to be impossible then you will have to load the module "manually" before running the game, as described later.

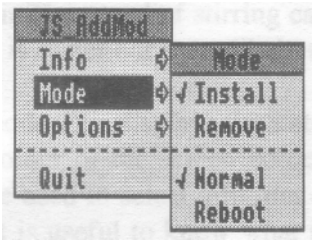
JS_AddMod can also be used on old style games with !Boot files instead of Risc OS applications. Simply drag the !Boot file onto the JS_AddMod icon instead.

You can access a selection of different joystick programs, necessary if for example the game uses different keys in one player and two player modes, by saving multiple joystick program files into the special directory viewer. When the game is loaded you will be asked which of the files you wish

to use, simply press a key to make a choice and the game will load as usual. (If there is only one file in the directory then you will not be offered a choice.)

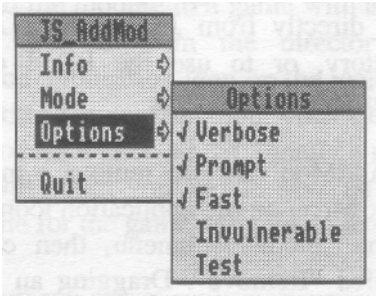
If you only have one disc drive then it may be difficult to copy a joystick program file into the game's special directory viewer without juggling a lot of discs. It is much easier to save directly from JS_Comp or JS_Tutor into the special directory, or to use the RAM disc as described in the Archimedes user guide.

JS_AddMod can also remove a module from a games disc. Click Menu on the application icon on the icon bar and move to the " mode" submenu, then click Select on the entry labelled " Remove". Dragging an application icon onto the JS_AddMod icon whilst in "Remove" instead of "Install" mode will remove the module.



The other option in the "mode" submenu (labelled "Normal" or "Reboot") is for use with old style games which "reboot" the computer while loading. You can recognise such games because there is a long pause immediately after starting the load, the screen will clear to black and the Acorn startup message will appear (maybe only briefly). If you wish to add multiple programs to such a game then you should use JS_AddMod in " Reboot" mode. This will prevent the prompt for choosing a joystick file from appearing twice during the load (although it would be harmless).

The joystick module can be loaded with any of the usual options, as described in the joystick module section below. These can be enabled/disabled using the "options" submenu. If you used JS_AddMod on a game which has been treated before then the current set of options will be used to update the game (including the normal/reboot mode).



The Joystick Module

The joystick module itself is hidden inside the !Joystick application directory and can be loaded from the command line, if need be, by using the command :

*RMLoad !Joystick.Joystick <options>

Where <options> consists of :

V - Verbose mode The module will display a startup message and the descriptive text in each program file loaded.

P - Prompt for interface The module will wait until the interface is switched on, a message will be displayed unless verbose mode is disabled.

F - Fast stir The speed of stirring can be either fast or slow, where slow is easier but more likely to happen by accident.

T - Test mode This displays the status of flags B_1 and B_2 using the colour of the screen border, it can be useful when the flags are used to select different control "modes" inside a game and it is useful to know what the current mode is. It is also a method of checking if the module is still resident after a game has loaded - try invulnerable mode if it isn't.

H - Help Displays a list of the options with their current states.

~ - Invert one of the above options

These options can also be set when loading the module from the desktop by double-clicking on the !Joystick application and setting the buttons in the small window that appears as required - see page 8.

By default, only verbose and fast modes are switched on. Thus using the option string ~V~FH would load without displaying a startup message, without prompting for the interface, with slow stirring selected and with the help text displayed.

The module provides a single * command for loading joystick program files, for example :

*JoystickLoad $.MyProg

Loading the Module Manually

If you cannot safely use the JS_AddMod application to force a game to load the joystick module automatically then you will have to do it by hand every time the game is loaded as outlined below:

A) Well behaved games

Luckily most games written since Risc OS was introduced are well behaved and do not remove any modules from memory. With such games the module can be loaded simply by double-clicking on a program file before starting the game as usual.

B) Badly behaved games

If a game is badly behaved then it may attempt to clear the relocatable module area. The "I" option can be used to make the joystick module invulnerable to most attempts to clear the module area. Thus to load the module you must press F12 in the desktop and issue the following commands with a disc containing the !Joystick application in the current drive :

1) *Mount
2) *RMLoad !Joystick.Joystick I
3) *JoystickLoad <whatever your joystick file is called>

You can then press return to return to the desktop and start the game as usual.

C) Games which need to be "booted"

Many old games are started by "booting" the disc (ie. holding down shift and pressing break). Because the modules will be cleared as the machine is reset during the reboot it is necessary to perform the boot a different way. Load the module and joystick program file as for (B) above, insert the games disc and enter the following commands:

1) *Mount 0
2) *Cat
3) *Run !Boot if the "Option" field of the catalogue is 02 (Run) or *Exec !Boot otherwise

D) Games which "reboot" themselves

The most badly behaved games will actually make the computer reset itself and then re-execute the game. In these cases it is necessary to perform the following drastic actions:

1) Start the game loading as usual
2) As soon as the game has started loading hold down shift together with * on the numeric keypad
3) When the computer reboots it will produce a * prompt
4) Now do (B) then (C) again to load the joystick module and then start the game re-loading

The Joystick Programming Language

The joystick programming language is used to instruct JS_Comp how to translate joystick movements into key presses or mouse movements. A program consists of a series of expressions such as this:

```
return = fire_1
```

This will simulate a press of the return key when the fire button on joystick one is pressed.

The expressions can include the boolean operators:

~ NOT

. AND

+ OR

for example:

```
return = fire_1 . fire_2
space_bar = fire_1 + fire_2
enter = fire_1 . ~fire_2
```

This means that holding fire 1 AND fire 2 will press return, holding fire 1 OR fire 2 will press the space bar and holding fire 1 AND releasing fire 2 will press enter.

By using these expressions together with each joystick's "stir" ability a large number of actions can be performed with basic joystick movements.

For example, the following program allows joystick one to perform the functions of the seven keys used during editing :

```
| Move stick without fire to move cursor
left_arrow = left _1 . ~fire 1
right_arrow = right_1 .~fire_1
up_arrow = up_1 . ~fire_1
down_arrow = down_1 . ~fire_1
| Move stick with fire pressed for return,copy,delete
return = down_1 . fire_1
copy = right_1 . fire_1
delete = left_1 . fire_1
```

The left hand side of each expression does not have to be a key, it can also be a mouse movement with a speed from 0 to 15, for example:

```
| Move stick without fire to move pointer
right_pointer_4 = right_1 . fire_1
left_pointer_4 = left_1 . ~fire_1
down_pointer_4 = down_1 . ~fire_1
up_pointer_4 = up_1 . ~fire_1
| Move stick with fire pressed for left,middle,right buttons
left_button = left_1 . fire_1
middle button = down_1 . fire_1
right_button = right_1 . fire_1
```

A "memory" capability is provided in the form of a set of flags which can be included in each expression. There are four flags named A and B for joysticks 1 and 2. They can be set and reset as follows:

setA_1 = fire_l

resetA_1 = fire_1

return = flagA_1

This will make flag A_1 follow the state of fire 1 since when fire is pressed the flag will set and when fire is released the flag is reset. The return key is pressed then flag A_1 is set so , the net effect is that the fire button acts like the return key again.

setB_1 = ~flagB_1

resetB_1 = flagB_1

return = flagB_1

This will make flag B_1 oscillate between set and unset states. This will happen every time the expressions are reevaluated : 50 times per second. Thus flagB_1 will oscillate at 25Hz. This is how rapid firing is provided.

setA_1 = fire_1

resetA_1 = ~fire_l

setB_1 = ~flagB_1 . ~flagA_1 . fire_1

resetB_1 = flagB_1 . ~flagA_1 . fire_1

return = flagB_1

This example is a combination of the above two. Again, flag A_1 follows the state of the fire button. flagB_1 will oscillate again but can only do so when -flagA_1.fire_1 is set. At the instant when the fire button is pressed then fire_1 is set but flagA_1 (the previous state of fire_1) is unset, thus

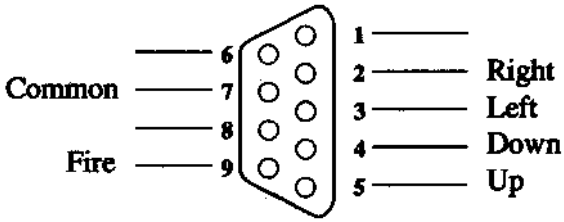
~flagA_1.fire_1 is only true when the fire button goes from an unpressed state to a pressed state.

The result of all this can be seen if this program is run and observed using !JS_Test, pressing the fire button toggles flagB_1 between unset and set states.

A complete list of the "keywords" used in the joystick programming language is given in Appendix B.

Appendix A

Pinouts of the Atari/Commodore standard joystick



Appendix B

Joystick programming language summary

Keywords which can appear on right hand side of expressions where N=1 or 2

right_N left_N down_N up_ N fire_N stir_N
flagA_N flagA_N

Keywords which can appear on left hand side of expressions, num_ prefix means the key is on the numeric keypad:

Keyboard buttons:
esc f1 f2 f3 f4 f5 f6 f7 f8 f9 ft10 f11 fl2 print
scroll_lock break (also shift_fl, ctrl_shift_f1 etc)
' 1 2 3 4 5 6 7 8 9 0 - = £ backspace
insert home page_up
num_lock num_/ num_* num_# tab q w
e r t y u i o p [ ] \
delete copy page_down
num_7 num_8 num_9 num_-
left_ctrl a s d f g h j k 1 ; ' return
num_4 num_5 num_6 num_+
left_shift z x c v b n m ./ right_shift
up_arrow left_arrow down_arrow right arrow
num_1 num_2 num_3 num_0 num_. enter
caps_lock left_alt space_bar right_alt right_ctrl
shift ctI alt
Mouse buttons:
left_button middle_button right_button
Pointer movement with speed N:
right_pointer_N left_pointer_N down_pointer_N up_pointer_N
Flag setting/resetting, N=1 or 2:
resetA_N setA_N resetB_N setB_N

Appendix C

Guidelines for applications programmers

We at The Serial Port would welcome any programmers who wish to provide support for our joystick interface in their programs. The joystick module provides a SWI call to test the joystick directly named "Joystick_Status", it returns a word with the following bits :

0 Joystick 1 right
1 left
2 down
3 up
4 fire
5 stir
6 flagA
7 flagB

8 Joystick 2 right
9 left
10 down
11 up
12 fire
13 stir
14 flagA
15 flagB

You should definitely NOT copy our joystick module into the boot file for your program for three reasons:

1) The module may change because of hardware changes

2) The very act of accessing the printer port may disturb such non-standard devices such as dongles, it should be up to the user to decide when to load the joystick module

3) It is copyright The Serial Port(!)

The easiest way to support our joystick interface at the moment is to write programs according to Acorn's guidelines ie. don't clear the RMA, don't access hardware directly etc... and let the module simulate the keyboard.

If you wish to use the "Joystick Status" SWI then you should test for the presence of the module and only when present call the SWI. It should be left up to the user to double-click on the module before running the game.

SERIAL PORT PRODUCTIONS SUPPLIED BY
QD ENTERPRISES LTD.
TEL (0243) 531194 FAX: (0243) 531196