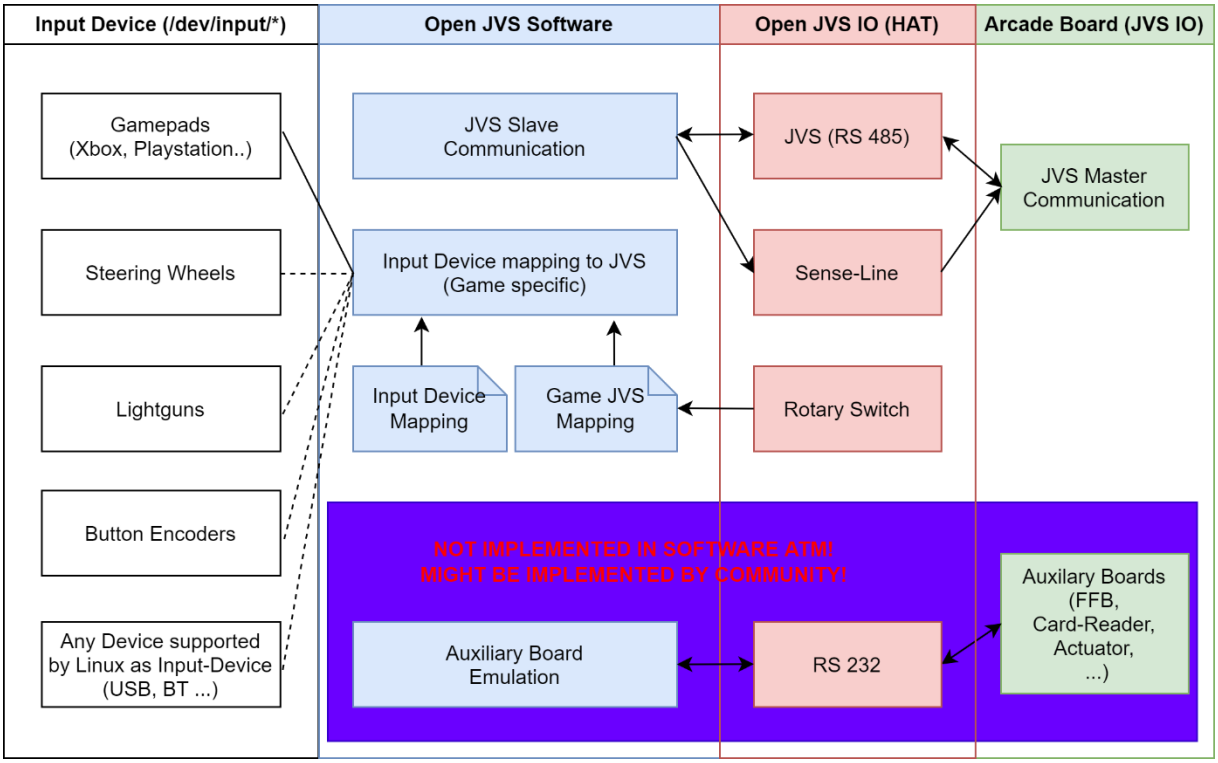
OpenJVS and OpenJVS IO Manual

By: RedOne and Bobby Dilley

1 Introduction

1.1 Concepts of OpenJVS



1.2 OpenJVS IO 1.1

OpenJVS IO is a HAT designed for the Raspberry PI series that will enable you to connect a JVS based Arcade board to the Raspberry PI. It takes care of the physical connections (RS485, Sense-Line) necessary for the JVS and also adds a RS232 transceiver and switches for varies purposes.

Features of OpenJVS IO

- JVS communication (RS485) with LED indicators
- JVS Sense-Line control with proper signal levels
- RS232 transceiver to communicate with various Arcade boards (FFB, Actuator Board) with LED indicators
- Rotary-Switch to change settings in “standalone” usage
- Reverse-Current protection for the PI
- Over-voltage protection for RS485
- As “plug and play” as is goes for Arcade stuff 😊

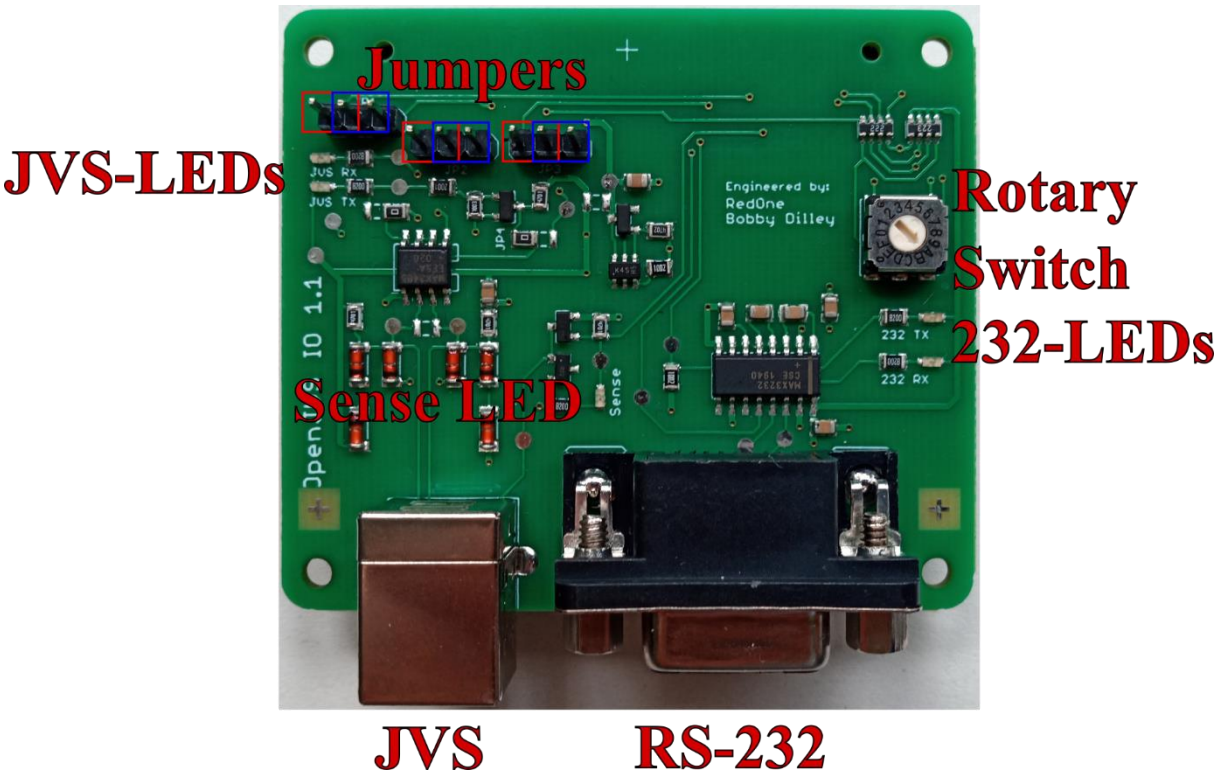
1.3 OpenJVS Software

OpenJVS is a software to emulate an actual JVS IO board opening the options for many different alternative input devices to be used with the JVS Arcade board. Virtually all (USB) input devices that are supported by Linux and show up as Input-Device are supported.
To check if your device is supported look for your device at */dev/input/* or use the tool *evtest*.

Features of OpenJVS

- Emulation of different JVS IO boards
- Many profiles for Arcade Games are available and new profiles can be added
- Supports any Input-device supported by Linux (*/dev/input)*
- Mapping layer to map arbitrary Input device to “Linux Gamepad Specification”
- Mapping layer to map “Linux Gamepad Specification” to Arcade Game profile
- Configurable PINs, UART-devices and Sense-Line implementations
- Configurable switch supported to change Game profiles
- Input-Devices and game profiles can be changed during runtime (hardware switch required)
- Open Source written in C

2 OpenJVS IO 1.1 Board



Picture 1: OpenJVS IO 1.1 board

2.1 JVS Connector

2.1.1 Description

Connect to JVS master of an Arcade board.
On physical level this is a standard RS-485 signal with an additionally line for the JVS Sense-Line

WARNING: DO NOT CONNECT TO A USB PORT FOR THIS MIGHT DAMAGE JVS-IO BOARD OR THE USB PORT.

2.1.2 Connector and Pinout

Connector: USB-B female

Pin	Function	Comment
D+	RS485-A	
D-	RS485-B	
VBUS	Sense-Line	
GND	GND	

Table 1- JVS USB Pinout

2.2 RS-232

2.2.1 Description

Standard RS232 signal that can be used to communicate with several Arcade boards like “Actuator boards” or “FFB boards”.

2.2.2 Connector and Pinout

Connector: DB9 Female

Note: To be used with a straight DB9 male to DB9 female cable. (Switching the connection TX->RX and vice-versa is already done on the board).

Pin	Function	Comment
2	RS232 TX	
3	RS232 RX	
5	GND	

Table 2-RS232 Connector Pinout

2.3 Rotary Switch

The rotary switch serves as input to be able to interface with the software to reduce the necessity to manually log into the Raspberry to change a setting.

For detailed explanation how the switch is used in OpenJVS please see 3.2.5 Configuration of Rotary Switch.

NOTE: Please use suitable sized screwdriver (or similar item) to turn the switch. With the right tool the switch can be rotated very easily and is durable.

2.4 JVS Jumpers

2.4.1 Description

Jumpers are used to select which Raspberry PI Pins are connected to the RS-485 transceiver for JVS. All three jumpers (JP1, JP2, JP3) must be either left or right aligned to connect the signals to the Raspberry-PIs UART.

Please see Picture 1: OpenJVS IO 1.1 board for visual definition of “Left Aligned” (in RED) and “Right Aligned” (in BLUE).

Note: When using a Raspberry PI4 it is strongly recommended to use UART3 with the “Left Aligned (in RED)” jumper position for multiple reasons. Only use UART0 with the Right Aligned (in BLUE) jumper position when using an older Raspberry PI or you want to make your own UART remapping and know how to do so.

2.4.2 Pins affected by Jumpers

Jumper Setting	Pin at Raspberry (BCM / Connector)	Function / UART
Left Aligned (in RED)	BCM4 / Pin 7	TXD3 / UART 3
	BCM5 / Pin 29	RXD3 / UART3
	BCM7 / Pin 26	RTS3 / UART3
Right Aligned (in BLUE)	BCM15 / Pin 10	RXD0 OR RDX1 / UART0 OR UART1
	BCM14 / Pin 8	TDX0 OR TXD1 / UART0 OR UART1
	BCM17 / Pin 11	RTS0 OR RTS1 / UART0 OR UART1

Table 3 - Jumper Settings

Please note that the RTS Signals (in Purple) are OPTIONAL. Currently these are only reserved and thus can be used for other purposes.

2.5 Overview of Raspberry-PI Pins used by OpenJVS IO

Pin at Raspberry (BCM / Connector)	Usage for OpenJVS IO	Comment
BCM4 / Pin 7	JVS Communication (RS485-TX)	See 2.4 JVS Jumpers
BCM5 / Pin 29	JVS Communication (RS485-RX)	See 2.4 JVS Jumpers
BCM7 / Pin 26	JVS Communication (RS485 output enable)	OPTIONAL
BCM15 / Pin 10	JVS Communication (RS485-RX)	See 2.4 JVS Jumpers
BCM14 / Pin 8	JVS Communication (RS485-TX)	See 2.4 JVS Jumpers
BCM17 / Pin 11	JVS Communication (RS485 output enable)	OPTIONAL
BCM12 / Pin 32	JVS Sense-Line Control (Output to Pull Sense-Line down)	
BCM18 / Pin 12	Rotary Switch (Bit 0)	
BCM19 / Pin 35	Rotary Switch (Bit 1)	
BCM20 / Pin 38	Rotary Switch (Bit 2)	
BCM21 / Pin 40	Rotary Switch (Bit 3)	
BCM8 / Pin 24	RS-232 TX (UART4 – Pi4 only)	
BCM9 / Pin 21	RS-232 TX (UART4 – Pi4 only)	
Pin 6	GND	
Pin 39	GND	
Pin 17	3.3 V	

3 OpenJVS Software

3.1 Configuration of Raspberry PI

- Download the Raspberry Pi Imager from https://www.raspberrypi.org/software/. This will download a program available for Ubuntu, Mac and Windows.
- Insert your SD card, and run the Imager. For the OS, select Other Raspberry Pi OS, and then select Raspberry Pi OS Lite. Then select the SD card you want to write to, remembering this will remove everything on the Pi.
- Once the image has burnt, safely remove the card and re-insert it into your computer.
- Create a blank file called 'ssh' in the boot partition to enable SSH.
- Next you need to modify config.txt, adding the following to the bottom of the file.
  If you're on a Pi 3 add:
  dtoverlay=disable-bt
  If you're on a Pi 4 add:
  dtoverlay=uart3
  dtoverlay=uart4
- Next you must modify *cmdline.txt*
  Remove the initial serial console line, which will look like the following. Remove from *console* up to the first space.
  console=serial0,115200
- Now we need to setup WiFi. Add a file called wpa_supplicant.conf to the boot partition, containing the following. You will need to add your ssid, and password as well as country code. More information can be found here: https://www.raspberrypi.org/documentation/configuration/wireless/headless.md.
  ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
  update_config=1
  country=<Insert 2 letter ISO 3166-1 country code here>
  network={
  ssid="<Name of your wireless LAN>"
  psk="<Password for your wireless LAN>"
  }
- Safely remove the SD card, place it into the Pi and boot the pi
- You now need to find the Pi's IP Address. You can either do this by looking on your router's web page, or by downloading a network scanning application. Another way is to type the following into a terminal:
  ping raspberrypi.local
  You should see the IP address in the ping command
  Now you need to ssh into the Pi
  ssh pi@<ip_address>
  You should now be logged in and you should be able to see a prompt.
- If you're using a Pi3, theres more to do to disable bluetooth. Run the following:
  sudo systemctl disable hciuart
- Now you need to install the required dependencies
  sudo apt install git cmake evtest
- Clone openjvs
  git clone https://github.com/openjvs/openjvs
- Make and install OpenJVS
  cd openjvs

```
make
sudo make install
```

Please refer to section 3.2.4 Autostart OpenJVS as service for instructions on how to start OpenJVS.

3.2 Configuration of OpenJVS

3.2.1 Configuration of Uart to be used for JVS

To tell OpenJVS which uart to use for JVS communication please set the correct value for *DEVICE_PATH in /etc/openjvs/config*

If you were following these instructions and use a Pi4 use the following settings:
DEVICE_PATH /dev/ttyAMA1

If you were following these instructions and use a Pi3 (or older) use the following settings:
DEVICE_PATH /dev/ttyAMA0

3.2.2 Configuration of Uart to be used for RS232

Auxiliary board support like FFB, Card-Reader, Actuator-Boards ARE NOT IMPLEMENTED yet.
We welcome every developer wanting to contribute a feature on Github and create a Pull-request one finished for merge.

3.2.3 Configuration of Sense-Line

In */etc/openjvs/config* set the correct value for *"SENSE_LINE_TYPE"*

Note: When using OpenJVS IO always use: `SENSE_LINE_TYPE 2`

Value	Description
0	Disable Sense-Line Control
1	Direct-Sense-Line mode (GPIO will be driven low when Sense-Line is engaged. Otherwise GPIO will float leaving the Sense-Line as is it)
2	OpenJVS IO mode (GPIO will be driven high when Sense-Line is engaged, otherwise GPIO is driven low.)

Table 4 - Sense-Line configuration

3.2.4 Autostart OpenJVS as service

Function	Command
Autostart OpenJVS on boot	`sudo systemctl enable openjvs` `sudo systemctl start openjvs`
Stop OpenJVS when it was started automatically on boot	`sudo systemctl stop openjvs`
Disable OpenJVS to start on boot	`sudo systemctl disable openjvs`
View OpenJVS Logs	`sudo journalctl -u openjvs`

Table 5 - Autostart OpenJVS as service

3.2.5 Configuration of Rotary Switch

In OpenJVS the rotary switch is used for profile selection. To use this features it must be enabled in the global config and a list of profiles must be specified in the file */etc/openjvs/rotary*.

OpenJVS will automatically load the profile selected by the rotary switch when it starts but also supports “hot-switchting” and will reinit itself with the new profile.

To enable the Rotary Switch for OpenJVS please modify */etc/openjvs/config* and set the following value:
DEFAULT_GAME rotary

Note: Reinit will interrupt the JVS communication for a brief time and a running game may show an JVS communication error.

3.2.6 Configuration of Mapping

The new mapping system is based on a two stage mapping process. The first stage will map your device to the controller seen below. The second stage will map the controller seen below to an actual game on an arcade system. This means that when you want to change controller, you only have to create one map file for your new controller, and likewise if you want to change game you only have to make one map file for your new game.

To create mapping you should become familiar with the program evtest which can be installed with *sudo apt install evtest*. When you run it with *sudo evtest* it will show you all of the input devices you have connected, and if you select one it will list their capabilities and show you the input events they are producing when you press buttons.

3.2.6.1 Configuration of Input Devices

Files for mapping your own controller to this virtual controller live in */etc/openjvs/devices/* and should be named as the same name the controller comes up in when you run *sudo evtest* with the spaces replaced with - symbols.

The file consists of multiple lines with a FROM and TO mapping seperated with a space like below.

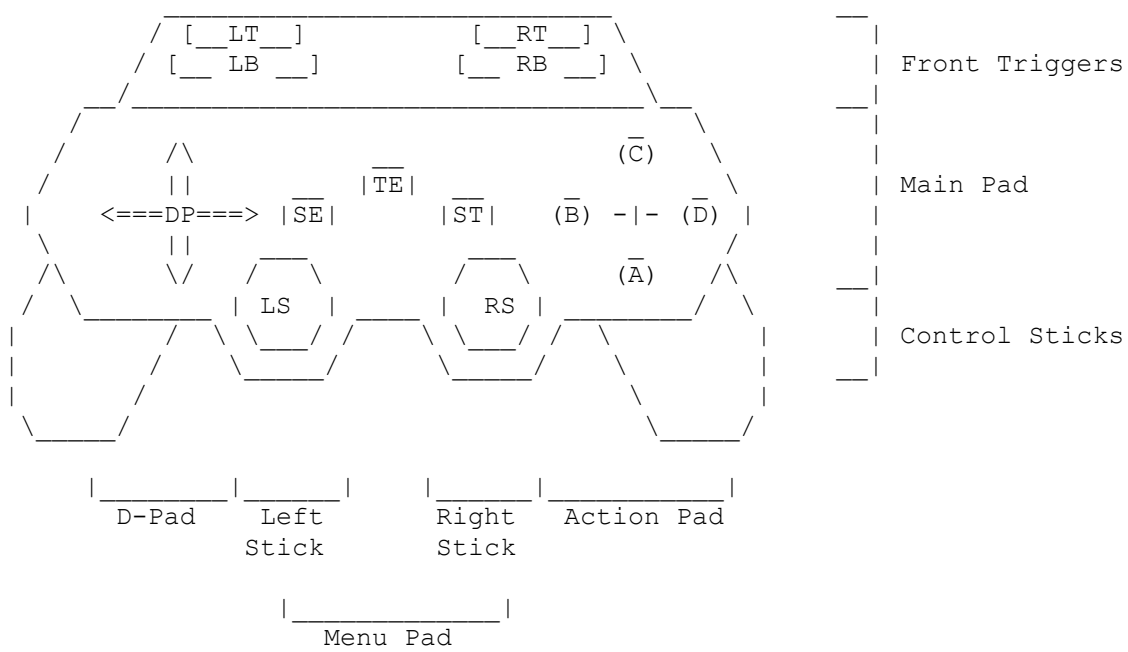
Note extra modifiers can be used on analogue channels:

- *REVERSE* can be added to the end of a line to reverse the direction of the input
- *SENSITIVITY 1.9* can be used to multiply the sensitivity of the devices analogue axis.

```
# Example Mapping File
# Author: Bobby Dilley

# Map the button section
# <FROM> <TO>
BTN_SOUTH CONTROLLER_ANALOGUE_A
BTN_NORTH CONTROLLER_ANALOGUE_C

# Map the analogue section
ABS_X CONTROLLER_ANALOGUE_X
ABS_Y CONTROLLER_ANALOGUE_Y REVERSE
ABS_Z CONTROLLER_ANALOGUE_R REVERSE SENSITIVITY 1.5
ABS_RZ CONTROLLER_ANALOGUE_L SENSITIVITY 0.9

# Map a HAT controller
ABS_HAT0X CONTROLLER_BUTTON_LEFT CONTROLLER_BUTTON_RIGHT
```

As above you can map the HAT controls which are sometimes used for DPADS. This should take controller button outputs on a single line. This can map any analogue channel into a digital one so analogue hats as well as thumb sticks can be converted into digitals!

FROM keywords are selected from the list of linux input event keywords. These are the same as the ones shown when you run *sudo evtest*. TO keywords are selected from the pre defined virtual controller mapping keywords list. Below is the virtual controller that the mapping is based upon, along with the mapping keywords used to reference this controller.



MAPPING KEYWORD	DIAGRAM LABEL	PURPOSE
CONTROLLER_BUTTON_TEST	TE	Test Button,
CONTROLLER_BUTTON_TILT		Tilt Button,
CONTROLLER_BUTTON_COIN	RS CLICK	Coin Button,
CONTROLLER_BUTTON_START	ST	Start Button,
CONTROLLER_BUTTON_SERVICE	SE	Service Button,
CONTROLLER_BUTTON_UP	DP UP	Joystick Up Button,
CONTROLLER_BUTTON_DOWN	DP DOWN	Joystick Down Button,
CONTROLLER_BUTTON_LEFT	DP LEFT	Joystick Left Button,
CONTROLLER_BUTTON_LEFT_BUMPER	LB	Gear Down Button,
CONTROLLER_BUTTON_RIGHT	DP RIGHT	Joystick Right Button,
CONTROLLER_BUTTON_RIGHT_BUMPER	RB	Gear Up Button,

```
CONTROLLER_BUTTON_A,                    A
Button 1 / Trigger Button,
CONTROLLER_BUTTON_B,                    B
Button 2 / Screen In / Out,
CONTROLLER_BUTTON_C,                    C
Button 3 / Action Button 1,
CONTROLLER_BUTTON_D,                    D
Button 4 / View Change Button,
CONTROLLER_BUTTON_E,                    LS CLICK
Button 5,
CONTROLLER_BUTTON_F,
Button 6,
CONTROLLER_BUTTON_G,
Button 7,
CONTROLLER_BUTTON_H,
Button 8,
CONTROLLER_BUTTON_I,
Button 9,
CONTROLLER_BUTTON_J,
Button 10,
CONTROLLER_ANALOGUE_X,                  LS X
Analogue Joystick X / Steering Wheel,
CONTROLLER_ANALOGUE_Y,                  LS Y
Analogue Joystick Y / Aeroplane Pitch,
CONTROLLER_ANALOGUE_Z,                  RS X
Analogue Joystick Z,
CONTROLLER_ANALOGUE_R,                  RT
Accelerator,
CONTROLLER_ANALOGUE_L,                  LT
Breaks,
CONTROLLER_ANALOGUE_T,                  RS Y
Analogue Joystick T,
```

3.2.6.2 Configuration of Game Profiles

Files for mapping your own controller to this virtual controller live in */etc/openjvs/games/* and should be named either by their function such as *generic-driving* or by a specific game name *outrun*.

This file will be selected using the *DEFAULT_MAPPING* config keyword, or by a parameter passed to the program *sudo openjvs outrun2*.

The file consists of multiple lines with a *CONTROLLER_INPUT CONTROLLER_PLAYER ARCADE_INPUT ARCADE_PLAYER* mapping as shown below. Note *REVERSE* can be added to the end of a line to reverse the direction of the input. Note a secondary *BUTTON_** can be added to the end of the line as a secondary output which will be enabled with the first. This allows the emulating of H shifters in games like Wangan Midnight Maximum Tune.

```
# Example Mapping File
# Author: Bobby Dilley

# Map the button section
# <CONTROLLER_INPUT> <CONTROLLER_PLAYER> <ARCADE_INPUT> <ARCADE_PLAYER>
CONTROLLER_BUTTON_A CONTROLLER_1 BUTTON_1 PLAYER_1
CONTROLLER_BUTTON_B CONTROLLER_1 BUTTON_2 PLAYER_1

# Player Two
CONTROLLER_BUTTON_A CONTROLLER_2 BUTTON_1 PLAYER_2
CONTROLLER_BUTTON_B CONTROLLER_2 BUTTON_2 PLAYER_2

# Map the analogue section
```

```
CONTROLLER_ANALOGUE_X CONTROLLER_1 ANALOGUE_0
CONTROLLER_ANALOGUE_Y CONTROLLER_1 ANALOGUE_1
```

You can also now map an Analogue to a digital button using the DIGITAL modifier at the start of the line:

```
DIGITAL CONTROLLER_ANALOGUE_X CONTROLLER_1 BUTTON_1 PLAYER_1
```

The list of ARCADE_INPUT keywords are listed below:

```
MAPPING KEYWORD
---------------
BUTTON_TEST,
BUTTON_TILT,
BUTTON_START,
BUTTON_SERVICE,
BUTTON_UP,
BUTTON_DOWN,
BUTTON_LEFT,
BUTTON_RIGHT,
BUTTON_1,
BUTTON_2,
BUTTON_3,
BUTTON_4,
BUTTON_5,
BUTTON_6,
BUTTON_7,
BUTTON_8,
BUTTON_9,
BUTTON_10,
ANALOGUE_1,
ANALOGUE_2,
ANALOGUE_3,
ANALOGUE_4,
ANALOGUE_5,
ANALOGUE_6,
ROTARY_1,
ROTARY_2,
ROTARY_3,
ROTARY_4,
ROTARY_5,
ROTARY_6,
```

Other mapping files can be included, so for example if a game is almost exactly the same as a generic mapping with one small difference, the generic mapping can be included and the one small difference added.

```
# Example Outrun 2 Mapping File
# Author: Bobby Dilley

# Set the IO to emulate
EMULATE sega-type-3

# Include the generic driving file
INCLUDE generic-driving

# Make the additions
CONTROLLER_BUTTON_RIGHT_BUMPER CONTROLLER_1 BUTTON_UP PLAYER_2
CONTROLLER_BUTTON_LEFT_BUMPER CONTROLLER_1 BUTTON_DOWN PLAYER_2
```

As well as this the IO that should be emulated can be specified in the map file with the `EMULATE` keyword, these IOs are defined in the */etc/openjvs/ios* directory.

4 Document History

Document Version	Changes
1.2	Initial Document Version