Jaguar Console Hardware Release Notes

This document describes the Jaguar console hardware as far as software development is concerned. It is a companion to the **Jaguar Software Reference Manual - Tom & Jerry.**

General Guidelines For Software

Do not ever write to any of the following registers. The BOOTROM (in a standard retail console) or the STUBULATOR (in a development console) will set them up. Especially settings in CLK2, CLK3 and HP registers must be correct to make the hardware work at all and prevent dot crawl in particular. We really do mean it: DONT TOUCH THIS!

MEMCON1	$F00000	HVS	$F00036
MEMCON2	$F00002	HEQ	$F00054
CLK1	$F10010	VP	$F0003E
CLK2	$F10012	VBB	$F00040
CLK3	$F10014 (aka CHROMA_DIV)	VBE	$F00042
HP	$F0002E	VS	$F00044
HS	$F00034	VEB	$F0004A
HBE	$F00032	VEE	$F0004C
HBB	$F00030		

The VMODE register and object processor will be initialized and started after reset by the bootcode. Then the only object in the object list will be a stop object, which will effectively display a blank screen and send the correct video synchronization signals to the monitor or TV. This also allows the phase locked loop to settle, which takes about a second at start-up. Do not ever turn video off again! (i.e. by writing a zero to VMODE !!)

Specific Bits in Production Series Consoles

Audio is mute after reset. You have to turn it on by setting bit 8 in register JOYSTICK.

Jaguar cartridge normally contain a 128 byte serial EEPROM to be able to save highscores and other user specific information. For information how to access the EEPROM refer to EEPROM.ZIP of your developer software or from our BBS. EEPROM cartridges currently use bit 0 of JOYSTICK. Do not rely on the readable status of JOYSTICK bit 0 - it is random.

Jaguar Memory Map / Register List

The tables below show the Jaguar hardware register list. For each item in this list, we show the equate as given in the JAGUAR.INC include file (or other appropriate include files), the name of the register as given in the Jaguar Software Reference Manual, the address of the register in hexadecimal, and a two-letter code for how the register is to be used:

RW = Read/Write WO = Write Only RO = Read Only

Note: Those registers shown in BOLDFACE should never be modified by your programs. They are set up for you by the machine at boot-time. They are included here for informational purposes only.

System Setup Registers			
MEMCON1	**Memory Control Register 1**	**F00000**	**RW**
MEMCON2	**Memory Control Register 2**	**F00002**	**RW**
HC	Horizontal Count	F00004	RW
VC	Vertical Count	F00006	RW
LPH	Light Pen Horizontal	F00008	RO
LPV	Light Pen Vertical	F0000A	RO
OB[0-3]	Object Data Field	F00010-16	RO
OLP	Object List Pointer	F00020	WO
OBF	Object Flag	F00026	WO
VMODE	Video Mode	F00028	WO
BORD1	Border Colour (Red & Green)	F0002A-2C	WO
HP	**Horizontal Period**	**F0002E**	**WO**
HBB	**Horizontal Blanking Begin**	**F00030**	**WO**
HBE	**Horizontal Blanking End**	**F00032**	**WO**
HS	**Horizontal Sync**	**F00034**	**WO**
HVS	**Horizontal Vertical Sync**	**F00036**	**WO**
HDB1	Horizontal Display Begin 1	F00038	WO
HDB2	Horizontal Display Begin 2	F0003A	WO
HDE	Horizontal Display End	F0003C	WO
VP	**Vertical Period**	**F0003E**	**WO**
VBB	**Vertical Blanking Begin**	**F00040**	**WO**
VBE	**Vertical Blanking End**	**F00042**	**WO**
VS	**Vertical Sync**	**F00044**	**WO**
VDB	Vertical Display Begin	F00046	WO
VDE	Vertical Display End	F00048	WO
VEB	**Vertical Equalization Begin**	**F0004A**	**WO**
VEE	**Vertical Equalization End**	**F0004C**	**WO**
VI	Vertical Interrupt	F0004E	WO
PIT[0-1]	Programmable Interrupt Timer	F00050-52	WO
HEQ	**Horizontal equalization end**	**F00054**	**WO**
BG	Background Colour	F00058	WO
INT1	CPU Interrupt Control Register	F000E0	RW
INT2	CPU Interrupt resume register	F000E2	WO
CLUT	Colour Look-Up Table	F00400-7FE	RW
LBUF	Line Buffer	F00800-1D9E	RW

GPU Registers

G_FLAGS	GPU Flags Register	F02100	RW
G_MTXC	Matrix Control Register	F02104	WO
G_MTXA	Matrix Address Register	F02108	WO
G_END	Data Organisation Register	F0210C	WO
G_PC	GPU Program Counter	F02110	RW
G_CTRL	GPU Control/Status Register	F02114	RW
G_HIDATA	High Data Register	F02118	RW
G_REMAIN	Divide unit remainder	F0211C	RO
G_DIVCTRL	Divide unit Control	F0211C	WO

Blitter Registers

* Must be refreshed after a BLIT

** Must be refreshed if used to store dynamic data (i.e. an inner loop read occurs or GOURD or GOURZ is set).

*** Older versions of the **Jaguar Software Reference Manual** (v2.2 & earlier) reversed the order of these descriptions. The equates have not changed, so your source code should be unaffected.

A1_BASE	A1 Base Register	F02200	WO
A1_FLAGS	Flags Register	F02204	WO
A1_CLIP	A1 Clipping Size	F02208	WO
A1_PIXEL	A1 Pixel Pointer	F0220C F02204	WO RO*
A1_STEP	A1 Step Value	F02210	WO
A1_FSTEP	A1 Step Fraction Value	F02214	WO
A1_FPIXEL	A1 Pixel Pointer Fraction	F02218	RW*
A1_INC	A1 Increment	F0221C	WO
A1_FINC	A1 Increment Fraction	F02220	WO
A2_BASE	A2 Base Register	F02224	WO
A2_FLAGS	A2 Flags Register	F02228	WO
A2_MASK	A2 Window Mask	F0222C	WO
A2_PIXEL	A2 Pixel Pointer	F02230 F0222C	WO RO*
A2_STEP	A2 Step Value	F02234	WO
B_CMD	Command/Status Register	F02238	RW*
B_COUNT	Counters Register	F0223C	WO*
B_SRCD	Source Data Register	F02240	WO**
B_DSTD	Destination Data Register	F02248	WO**
B_DSTZ	Destination Z Register	F02250	WO**
B_SRCZ1	Source Z Register 1	F02258	WO**
B_SRCZ2	Source Z Register 2	F02260	WO**
B_PATD	Pattern Data Register	F02268	WO**
B_IINC	Intensity Increment	F02270	WO
B_ZINC	Z Increment	F02274	WO
B_STOP	Collision control	F02278	WO
B_I3	Intensity 3***	F0227C	WO
B_I2	Intensity 2***	F02280	WO
B_I1	Intensity 1***	F02284	WO

B_I0	Intensity 0***	F02288	WO
B_Z3	Z 3***	F0228C	WO
B_Z2	Z 2***	F02290	WO
B_Z1	Z 1***	F02294	WO
B_Z0	Z 0***	F02298	WO

Jerry Registers			
CLK1	**Processor clock divider**	**F10010**	**WO**
CLK2	**Video clock divider**	**F10012**	**WO**
CLK3	**Chroma clock divider**	**F10014**	**WO**
JPIT1	Timer 1 Pre-scaler	F10000	WO
JPIT3	Timer 2 Pre-scaler	F10004	WO
JPIT2	Timer 1 Divider	F10002	WO
JPIT4	Timer 2 Divider	F10006	WO
J_INT	Interrupt Control Register	F10020	RW
SCLK	Serial Clock Frequency	F1A150	WO
SMODE	Serial Mode	F1A154	WO
LTXD	Left transmit data	F1A148	WO
RTXD	Right transmit data	F1A14C	WO
LRXD	Left receive data	F1A148	RO
RRXD	Right receive data	F1A14C	RO
L_I2S	Left I2S Serial Interface	F1A148	RW
R_I2S	Right I2S Serial Interface	F1A14C	RW
SSTAT	Serial Status	F1A150	RO
ASICLK	Asynchronous Serial Interface Clock	F10034	RW
ASICTRL	Asynchronous Serial Control	F10032	WO
ASISTAT	Asynchronous Serial Status	F10032	RO
ASIDATA	Asynchronous Serial Data	F10030	RW

Joystick Registers			
JOYSTICK	Joystick register	F14000	RW
JOYBUTS	Button register	F14002	RW

DSP Registers			
D_FLAGS	DSP Flags Register	F1A100	RW
D_MTXC	DSP Matrix Control Register	F1A104	WO
D_MTXA	DSP Matrix Address Register	F1A108	WO
D_END	DSP Data Organisation Register	F1A10C	WO
D_PC	DSP Program Counter	F1A110	RW
D_CTRL	DSP Control/Status Register	F1A114	RW
D_MOD	Modulo instruction mask	F1A118	WO
D_REMAIN	Divide unit remainder	F1A11C	RO
D_DIVCTRL	Divide unit Control	F1A11C	WO
D_MACHI	MAC High Result Bits	F1A120	RO

Jaguar Video & System Clocks

In the Jaguar console, the video clock is chosen to allow an inexpensive RF modulator system. This requires slightly different clock speeds for NTSC and PAL systems (but the difference is only about 0.01%). To be cost-effective, the GPU/DSP processor clock speed is the video clock speed, and the 68000 is 50% of this clock rate:

	NTSC	PAL
Video Clock GPU/DSP Clock Rate	26.590906 MHz	26.593900 MHz
68000 Clock Rate (50% of video clock)	13.295453 MHz	13.29695 MHz

The video system of Jaguar is programmable within the precision of the supplied video clock. From the video clock, the system produces the pixel (or dot) clock. The ratio between video and pixel clock is determined by high order bits of the VMODE register. The possible values for the ratio are shown in the table below, along with the number of pixels that will fit on screen overscanned or non-overscanned.
The numbers are the same for NTSC and PAL. For both PSL and NTSC the "safe' video area is about 40µs wide. The area required to guarantee of overscan is about 50µs. The table gives the number of pixels that can be displayed within these times for all available pixel clock dividers. Note that these numbers are not "nice" computer numbers like 320 or 256. Also, note that these numbers are simply rough guidelines to be used in deciding your artwork and object sizes, these numbers should not be used in calculating values for the video hardware registers. ***To properly initialize your program, including video, you must use the standardized Jaguar Startup Code described in the*** *Jaguar Libraries section*.

Pixel Divisor value for VMODE register	# of pixels Non-Overscanned	# of pixels Overscanned
1	1064	1330
2	532	665
3	355	443
4	266	332
5	213	266
6	177	222
7	152	190
8	133	166

We recommend that ALL software are for the Jaguar console overscan both vertically and horizontally so for the rest of this discussion we will restrict ourselves to the OVERSCAN column.

The first row (divisor of 1) requires that the object processor be started twice each line and produces a ridiculously high resolution for a TV, so it will be ignored.

A divisor of 3 gives a non overscanned resolution off about 355. This is a good match for many computer systems and programs designed around 320 pixel wide screens.

A divisor of four gives pixels that are about square. Square pixels are a great advantage for art creation and we recommend their use.

Let's look at the specific case of an overscanned game using square pixels. This uses a pixel divisor of 4. In both NTSC and PAL this allows for about 332 pixels to be displayed. Choosing a 320 pixel wide bitmap gives us a <4% error. Of these 320 pixels we should only count on the middle 266 being visible on most monitors and/or TV sets. This means that there is a border of about 27 pixels on each side that may be visible, but which should not contain essential game information.

The other pixel clock divisor that is of likely interest are is 5. In this case the number of overscanned pixels is useably close to a blittable width: 256.

To overscan vertically we suggest a screen height of 240 lines for NTSC and 288 lines for PAL. This will allow for both PAL and NTSC users to see a fully overscanned image both vertically and horizontally. The guaranteed visible region within which crucial game information is restricted is 200 lines for NTSC and 240 lines for PAL. Using 200 lines of critical video for both systems is a significant, and acceptable, simplification.

Video Ports

The information in this section is for informational purposes only. ***Do not attempt to change these timings or unpredictable results will occur!***

There are four versions of the Jaguar Console:

Video Standard	Where used
NTSC	USA/Canada
PAL-I	United Kingdom
PAL-B	Germany / other European countries
Peritel/Scart	France

The Jaguar console has an external video connector which supports Composite video, S-Video, and RGB. In addition, there is an RF Modulator output for all versions except the French Peritel/Scart version. The Peritel/Scart version is identical to PAL-B; except that there is no RF modulator.
Composite video, S-Video, and RGB are all available on the Peritel version, and have the same timings and characteristics of PAL-B.

The various specification timings are shown below:

RF And Composite

The information in this section is for informational purposes only. ***Do not attempt to change these timings or unpredictable results, will occur!***

	Chroma Clock	Subcarrier (MHz)	Sound subcarrier (MHz)
PAL-I	4.43361875	591.250	6 MHz
PAL-B	4.43361875	591.250	5.5 MHz
NTSC Channel 3	3.579545	61.25	4.5 MHz
NTSC Channel 4	3.579545	67.25	4.5 MHz

Video Timings

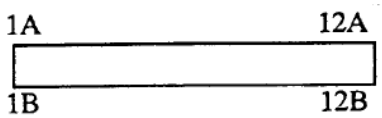
The information in this section is for informational purposes only. ***Do not attempt to change these timings or unpredictable results will occur!***

Parameter	PAL	NTSC	
Video master clock	26.593900MHz	26.590906MHz	
Horizontal period	64.0us	63.5555us	
Hsync width	4.7us	4.76us	
Hback porch	5.7us	4.45us	
Hfront porch	1.65us	1.27us	
Equalization pulse width	2.35us	2.54us	
Vertical sync pulse width	27.3us	29.26us	
Vertical lines (interlaced)	625	525	
Vertical lines (non interlaced)	624	524	
Vertical sync pulses	5	6	Non-interlaced
Vertical eq pulses before sync	5	6	Non-interlaced
Vertical eq pulses after sync	6	6	Non-interlaced
Vertical front porch	12 lines	12 lines	Non-interlaced
Vertical back porch	17 lines	12 lines	Non-interlaced

Jaguar Console Hardware Ports

Video Connector

The external video connector is a custom 24 pin, two row edge connector. The top row is row A, bottom row is row B. Pin 1 is on the left, pin 12 on the right when looking at the console from the rear:



Pin Number	Name	Description
1A	Audio_Left	EIAJ Line level, left audio
2A	Audio_Gnd	Audio Return (ground)
3A	Reserved	
4A	Video_Gnd	Video Return (ground)
5A	Blue	Blue video, 75 Ohm, 0.7V peak-to-peak
6A	Hsync	Horizontal Sync, 75 Ohm, 3V peak-to-peak
7A	Green	Green video, 75 Ohm, 0.7V peak-to-peak
8A	Chroma	S-Video Chroma, 75 Ohm, 1.0V peak-to-peak
9A	Reserved	
10A	Reserved	
11A	9V	9 Volts DC, 100mA max
12A	Reserved	
1B	Audio_Right	EIAJ Line level, right audio
2B	Audio_Gnd	Audio Return (ground)
3B	Video_Gnd	Video Return (ground)
4B	Red	Red video, 75 Ohm, 0.7V peak-to-peak
5B	VSL	Composite Sync, +5V, TTL levels
6B	Reserved	
7B	Video_Gnd	Video Return (ground)
8B	Luma	S-Video Luma, 75 Ohm, 1V peak-to-peak
9B	Reserved	
10B	Video_Gnd	Video Return (ground)
11B	Composite	Composite video, 75 Ohm, 1V peak-to-peak
12B	Reserved	

The Reserved signals should be left unconnected. They may be used in future versions of the Jaguar console, and therefore should be passed through on video adaptors. It is important to terminate the active signals correctly. Do not toad the 75 Ohm outputs with more than 75 Ohms.

A detailed mechanical drawing is available on request.

DSP Port

The external DSP connector is a custom 12-pin, two row edge connector. The top row is row A, the bottom row is row B. Pin 1 is on the left, pin 6 on the right when looking at the console from the rear:

Pin Number	Name	Description
1A	GND	
2A	SCK	Synchronous serial clock
3A	WS	Synchronous serial word strobe
4A	TXD	Synchronous serial transmit data (data out)
5A	RXD	Synchronous serial receive data (data in)
6A	GND	
1B	+5V	50mA maximum load
2B	UART_TXD	Asynchronous transmit data
3B	UART_RXD	Asynchronous receive data
4B	Reserved	Do not connect
5B	Reserved	Do not connect.
6B	GND	

All the active signals have 5 volt TTL levels. The SCK, WS, TXD and RXD signals are also connected to the cartridge expansion connector. They are used on the CD-ROM peripheral, therefore care must be taken to avoid contention (see the audio sub-system section below).

Cartridge / Expansion Port

Information on the Cartridge/Expansion Port of the Jaguar is available to hardware/accessory licensees. Hardware licensees should contact Atari regarding the connection of devices to this port.

Multi-console games

There are two types of Multi-Console games. The first type uses a special Local-Area-Network of multiple Jaguar consoles connected together via the console's asynchronous serial port. The second type uses the Jaguar modem to connect two Jaguar consoles via the telephone lines.

Jaguar Network

The low-level drivers required for networking multiple Jaguar consoles are currently in development. Contact Jaguar Developer Support for further information.

Jaguar Modem

The specification for using the Jaguar modem is described in the section titled **the Jaguar Voice Modem**.

Jaguar Controllers and Controller Ports

There are two controller ports on the Jaguar console: Controller Port 1 and Controller Port 2. Each has the following functions:

- Four bi-directional digital pins
- Six input only digital pins (split into 4 + 2 buttons)

Note: Early versions of the Jaguar console included an 8 bit ADC[1] on the motherboard. This has been deleted - analog controllers now require their own ADC chip.

Signals and Pinouts

Pin #	Port 1	Port 2	Description
1	J3	J4	Bi-directional signal. Used as output to specify to controllers which data to return
2	J2	J5	Bi-directional signal. Used as output to specify to controllers which data to return
3	J1	J6	Bi-directional signal. Used as output to specify to controllers which data to return
4	J0	J7	Bi-directional signal. Used as output to specify to controllers which data to return
5			Reserved
6	B0/LP	B2	Button input / Light gun on Port 1
7	+5V DC	+5V DC	Maximum 50mA load
8	n/c	n/c	Pulled up to +5V DC on 4 player adaptor
9	Gnd	Gnd	Ground
10	B1	B3	Button input
11	J11	J15	Input only signal
12	J10	J14	Input only signal
13	J9	J13	Input only signal
14	J8	J12	Input only signal
15			Reserved

Signals J0-J15, and B0-B3 are all TTL level digital inputs or outputs.

Controller Port 1 also has a light gun input in addition to the signals listed above. A TTL rising edge on the LP signal (pin 6 of port 1, shared with B0) causes the light pen registers (LPM and LPV) to be latched.

[1] Analog to Digital Converter — a device that converts analog signals such as a variable voltage level into a digital format suitable for processing by a computer.

Register Addressing – Digital Inputs

The table below shows the purpose of the individual bits of the JOYSTICK and JOYBUTS registers. Please note that some bits are used for non-controller related purposes.

JOYSTICK	$F14000 15.....8 7.....0	Read/Write	
Read	fedcba98 7654321q	f-1 q	Signals J15 to J1 Cartridge EEPROM output data
Write	exxxxxxm 76543210	e x m 7-4 3-0	1 = enable J7-J0 outputs 0 = disable J7-J0 outputs don't care Audio mute 0 = Audio muted (reset state) 1 = Audio enabled J7-J4 outputs (port 2) J3-J0 outputs (port 1)
JOYBUTS	**$F14002 15.....8 7.....0**	**Read Only**	
Read	xxxxxxxx rrdv3210	x r d v 3-2 1-0	don't care Reserved Reserved 1 = NTSC Video hardware 0 = PAL Video hardware Button Inputs B3 & B2 (port 2) Button Inputs B1 & B0 (port 1)

Device Addressing

All controller devices are addressed through the digital lines on the controller ports. Each controller port has 4 bi-directional pins and 6 input pins. We always use the bi-directional pins as outputs. By writing a 4-bit code to these outputs, 16 rows containing 6 bits of data each can be addressed. Each controller is allocated 4 rows of data, so up to 4 controllers may be connected to each port (via a 4-player adapter) for a maximum of 8 controllers total. Controllers may be connected to the Jaguar in two ways:

1) Directly to the controller port.

2) Via a multiplayer adaptor (usually a 4 player adaptor, or a pass-through connector on an advanced controller).

Advanced controllers typically provide a "pass-through" connector to allow a standard Jaguar controller to be connected at the same time as the advanced controller. Often this is a necessity, not a luxury, since the advanced controllers usually do not have as many buttons as the standard Jaguar controller (and may be missing such critical buttons as **Pause**).

Reading a Jaguar Controller

Reading a controller is done in two steps:

1) Write a 4 bit code to the port's output bits which specifies which row of controller data you want to read. Bits 0-3 of the JOYSTICK register contain the outputs bits for Port l. Bits 4-7 specify the output bits for Port 2. Note that the codes used for port 2 are a mirror image of the codes for port 1. (The bit order is reversed.) Bit 15 of JOYSTICK must also be set to enable the outputs. Bit 8 is also used to control audio muting, so you have to be careful not to clear this bit accidentally or you will disable your program's sound generation.

2) Read back the values contained in the JOYBUTS and JOYSTICK registers. These will contain the 6 data bits returned by each port.

For example, writing a value of $817E to JOYSTICK would allow you to read row 0 of the first controller connected to Port 1 and the first controller connected to Port 2. This value breaks down as:

```
$8000 = Enable JOYSTICK outputs J0-J7
$0100 = Enable Audio (bit 8 of JOYSTICK controls audio mute)
$0070 = Setup read of row 0 (code %0111) of controller 0, port 2
$000E = Setup read of row 0 (code %1110) of controller 0, port 1
-----
$817E = value to write to JOYSTICK register
```

Below is a table that shows how the 6 bits of data for each row are returned by the first controller connected on port 1 and the first controller returned on port 2. The meaning of the bits depends on which row is being read and what type of controller is connected (as defined later in the descriptions of each controller type).

Controller Port 1

Output Pin #					Input Pin #					
1 (J3)	2 (J2)	3 (J1)	4 (J0)		6 (B0)	10 (B1)	14 (J8)	13 (J9)	12 (J10)	11 (J11)
0	1	1	1	Row 3	C3	data	data	data	data	data
1	0	1	1	Row 2	C2	data	data	data	data	data
1	1	0	1	Row 1	C1	data	data	data	data	data
1	1	1	0	Row 0	data*	data	data	data	data	data

Controller Port 2

Output Pin #					Input Pin #					
1 (J7)	2 (J6)	3 (J5)	4 (J4)		6 (B2)	10 (B3)	14 (J12)	13 (J13)	12 (J14)	11 (J15)
1	1	1	0	Row 3	C3	data	data	data	data	data
1	1	0	1	Row 2	C2	data	data	data	data	data
1	0	1	1	Row 1	C1	data	data	data	data	data
0	1	1	1	Row 0	data*	data	data	data	data	data

* Bit B0 on Port 1 and bit B2 on Port 2 are used as a special "Bank 0" flag by bank switching controllers. See **Reading Bank Switching Controllers** for more information.

Identifying Controller Types

The basic type of controller is specified by the **C2, & C3** bits returned when you read the controller, as shown in the table. The currently defined controller type identifiers are: [1]

C2	C3	Controller Type
0	0	Reserved
0	1	Bank switching controller. (analog joystick, head-mounted tracker, etc.)
1	0	"Tempest" rotary controller
1	1	"Standard" Jaguar joypad controller (or nothing connected)

Software should scan all possible controller positions, including those on a 4-player adapter, to determine which types of controllers are currently connected. The game can then offer the user choice of which controller(s) to use.

Some advanced controllers use a special bank-switching technique to return more information than the 24 bits of data available from a standard controller. This makes a wide variety of controller types possible, so the specific controller type is identified by certain bits in the last bank of data returned by each controller.

Data Returned from Last Bank				
Row 3	Row 2	Row 1	Row 0	Bank Switching Controller Type
0	0	0	0	reserved
0	0	0	1	reserved
0	0	1	0	reserved
0	0	1	1	reserved
0	1	0	0	reserved
0	1	0	1	reserved
0	1	1	0	reserved
0	1	1	1	Head-mounted Tracker
1	0	0	0	reserved
1	0	0	1	reserved
1	0	1	0	reserved
1	0	1	1	reserved
1	1	0	0	reserved
1	1	0	1	Keyboard / Mouse
1	1	1	0	6D Controller
1	1	1	1	Analog Joystick or Driving Controller

See the descriptions of the individua1 controller types and the section **Reading Bank Switching Controllers** for additional information.

[1] Please note that the specification for identifying controllers was changed on March 31, 1995. The differences are important, but fairly minor from an implementation view, and do not affect any existing hardware on the market as of that date.

Standard Jaguar Controller Matrix

Below is a table showing the matrix for the standard joypad controller which is packed out with every Jaguar console. When plugged directly into the console, the matrix for this controller is as follows:

J4 J3	J5 J2	J6 J1	J7 J0	Port 2 Port 1	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1	Row 3	C3	Option	#	9	6	3
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1	Row 2	C2	C	0	8	5	2
1	1	0	0							
1	1	0	1	Row 1	C1	B	*	7	4	1
1	1	1	0	Row 0	Pause	A	Up	Down	Left	Right
1	1	1	1							

Reading a zero means the appropriate button is depressed.

4-Player Adaptor

The fact that 16 rows of data can be addressed allows a 4 controller adaptor to be connected to **each** console controller port (for a total of 8 controllers using two adaptors). The 4-player adapter is a device which expands either of the console controller ports to allow up to 4 controllers to be connected. It has 4 controller sockets (DB15 females, the same as on the console) for controllers to be connected, and a short cable with a DB15 male connector which plugs into the console.

The controller sockets on the adaptor have the 6 inputs wire OR'd together. The four output lines are an active low, 4 to 16 demultiplixed version of the 4 console outputs.

Each socket recognizes 4 unique row codes which are used to specify requests for data from that controller. The table below shows the row codes which must be output from the Jaguar to request data from controllers connected to specific sockets of the adapter. Note that socket 0 uses the same row codes as a single controller connected directly to one of the console controller ports.

Row Code Output From Jaguar:					Specifies which row of the controller connected to:			
Port 2 / Port 1	J4 / J3	J5 / J2	J6 / J1	J7 / J0	Socket 0	Socket 1	Socket 2	Socket 3
	0	0	0	0		Row 0		
	0	0	0	1		Row 1		
	0	0	1	0		Row 2		
	0	0	1	1		Row 3		
	0	1	0	0			Row 0	
	0	1	0	1			Row 1	
	0	1	1	0			Row 2	
	0	1	1	1	Row 3			
	1	0	0	0			Row 3	
	1	0	0	1				Row 0
	1	0	1	0				Row 1
	1	0	1	1	Row 2			
	1	1	0	0				Row 2
	1	1	0	1	Row 1			
	1	1	1	0	Row 0			
	1	1	1	1				Row 3

Except for socket 0, the row codes shown in the table are not the row codes seen by the controllers themselves. In order to make itself as transparent as possible to the controllers themselves, the adapter converts the row codes for sockets 1-3 so that those controllers will see only socket 0 row codes. In other words, when your program outputs the code %0101 that says it wants to read Row 1 of the controller connected to socket 2, the 4 player adapter will convert the code to %1101 and then pass it to socket 2. The controller connected to socket 2 will then see code %1101, the same code you would use to access a single controller connected directly to the Jaguar, and return the appropriate information.

4-Player Adapter and Advanced Controllers

Advanced controllers normally respond to row codes for socket 1 instead of the codes for socket 0 because they have a pass-through connector for a standard joypad controller, which sees socket 0 codes and responds as though it were connected directly to the Jaguar. However, when connected to a 4-player adapter, advanced controllers will never see codes for socket 1 because the adapter will convert them to socket 0 codes and then output only to the controller connected to socket 1.

Advanced controllers 'need to detect the presence of a 4-player adapter and change their behaviour when one is present. Therefore, the 4-Player adapter provides a +5v DC signal on pin 8 of each socket, which is normally not connected when controllers are plugged directly into the console. Advanced controllers are expected to detect this signal when present, disable their pass-through connector, and then respond as socket 0 instead of socket 1.

To summarize these ideas, the table below shows the various socket and controller positions with and without a 4-player adapter. (Ports 1 & 2 are identical in these respects.)

Controller Port With 4-Player Adapter			
Socket 0	Socket 1	Socket 2	Socket 3
Adapter converts row codes sent by Jaguar program and routes them to the appropriate socket. Socket 0 is the same as a controller plugged directly into port. Standard and Advanced controllers respond only to socket 0 row codes. Pass-through connectors of advanced controllers are disabled.			
Controller Port Without 4-Player Adapter			
Standard controller plugged directly into port is the same as socket 0 of a 4-player adapter. Advanced controllers plugged directly into port respond to Socket 1 row codes. Pass-through connectors of advanced controllers are enabled, and addressed as socket 0.			

Bank Switching Controllers

Because there are 4 row codes allocated to each socket, the 4-player adaptor support 4 row controller devices. Without additional logic, each input supports up to 24 bits of **data** (4 rows of 6 bits). Three bits are reserved for the controller type identifier code, leaving 21 bits for data.

Intelligent controllers (i.e. ones which use a microcontroller) can multiplex even more data onto the same lines. One way this can be done is for the microcontroller to "Bank switch" whenever it sees a transition from row 3 back to row 0. Different bits of data are presented in each bank. See the section **Reading Bank Switching Controllers** later in this chapter for more information.

Detecting the 4-Player Adapter & Connected Controllers

To detect the presence of a 4-Player adapter, a program should inquire the status of Row 1 of controller socket #3. If a 4-Player adapter is present, the B0/B2 bit will be clear (0). Otherwise, it will be set (1).

The pseudocode below demonstrates the basic technique for detecting a 4 player adapter and the controllers connected to it, as well as any advanced controllers connected directly to the Jaguar:

```
for PORT = 1 to 2
      if PORT:SOCKET3:C1 = 0 then { 4-p1ayer adapter found }
            for SOCKET = 0 to 3
                  PORT:SOCkET:CONTROLLERTYPE = PORT:SOCKET:C2/C3
                  if PORT:SOCKET:CONTROLLERTYPE = BANK-SWITCHING then
                        PORT:SOCKET:BANKSWITCHTYPE = DETECT_BANK_SWITCH_TYPE
                  endif
            NEXT  SOCKET
      else
            PORT:SOCKET0:CONTROLLERTYPE = STANDARD
            if PORT:SOCKET1:C2/C3 = ROTARY then
                  PORT:SOCKET1:CONTROLLERTYPE = ROTARY
            else  if PORT:SOCKET1:C2/C3 = BANK-SWITCHING then
                  PORT:SOCKET:BANKSWITCHTYPE = DETECT_BANK_SWITCH_TYPE
            endif
NEXT  PORT

FUNCTION DETECT_BANK_SWITCH_TYPE
      DO
            READ  ROWS 0, 1, 2, 3
      UNTIL ROW0:B0/B2 = 0 {bank 0}
      BANKCOUNT = 0
```



```
      DO
            READ  ROWS 0, 1, 2, 3
            SAVE  ROWDATA( BANKCOUNT )
            BANKCOUNT = BANKCOUNT + 1
      UNTIL ROW0:B0/B2 = 0 {bank 0}
      return ROWDATA(BANKCOUNT - 1):ROWS0-3:B1/B3
END   FUNCTION
```

Caveats

The JOYSTICK and JOYBUTS registers return the same data in the same bits regardless of witch socket is being read. However, be aware that without a 4-player adapter, reading sockets 1-3 of a port may return an 'echo' of the standard joypad controller at socket 0. To avoid reading incorrect data, unless your program has detected that an advanced controller or a 4-player adapter is connected, it should not try to read from sockets 1-3 (except for the detection phase when the program is trying to detect what is connected).

Advanced Controllers

6D Controller

These controllers support 6 degrees of freedom: Pitch, Yaw, Roll, X, Y and Z. We refer to Pitch as Z Torque, Yaw as X Torque and Roll as Y Torque. Hence we have 6 values – X, Y, Z and TX, TY and TZ. We also define 7 buttons, A-G.

Three banks of data are required, since we define 55 bits of information: 8 bit values for each of 6 degrees of freedom (8*6=48 bits of information), plus 7 buttons:

Bank 0	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	D	X4	X5	X6	X7
Row 2	0 (C2)**	C	Z0	Z1	Z2	Z3
Row 1	1 (C1)	B	Y0	Y1	Y2	Y3
Row 0	0*	A	X0	X1	X2	X3

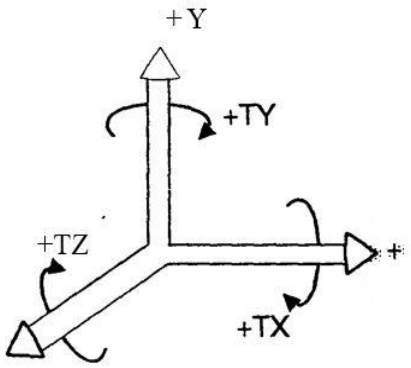
Bank 1	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	E	Y4	Y5	Y6	Y7
Row 2	0 (C2)**	F	TZ0	TZ1	TZ2	TZ3
Row 1	1 (C1)	G	TY0	TY1	TY2	TY3
Row 0	1*	Rezero	TX0	TX1	TX2	TX3

Bank 2	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	1**	Z4	Z5	Z6	Z7
Row 2	0 (C2)**	1**	TZ4	TZ5	TZ6	TZ7
Row 1	1 (C1)	1**	TY4	TY5	TY6	TY7
Row 0	1*	0**	TX4	TX5	TX6	TX7

* Bit B0/B2 of row 0 is used to synchronise the cycle of banks. It will always be zero in bank 0, while all other banks will return 1. Banks will cycle in the order Bank 0, Bank 1, Bank 2, Bank 0, etc. See **Reading Bank Switching Controllers** for more information.

** The C3 and G2 bits identify the basic controller type. The B1/B3 bits of the last bank of the controller are used to identify the specific bank switching controller type.

Value	Meaning
X(7:0)	X axis force
Y(7:0)	Y axis force
Z(7:0)	Z axis force
TX(7:0)	X axis, anticlockwise rotation torque
TY(7:0)	Y axis, anticlockwise rotation torque
TZ(7:0)	Z axis, anticlockwise rotation torque



X is positive right to left
Y is positive UP
Z is positive coming BACK (towards the user)

Torques are all positive in the COUNTER-CLOCKWISE direction, when facing the positive direction shown by the arrows above.

When connected directly to a Jaguar controller port, the controller will respond to socket 1 row codes (see **4-Player Adaptor**). A pass-through connector allows a second controller to be connected (usually a standard Jaguar Controller, for compatibility reasons), which will receive socket 0 row codes and appear as if it was directly connect to the Jaguar. When connected to a 4-player adaptor, the pass-through connector will not function and the controller will respond to socket 0 row codes.

Head Mounted Tracker

These devices provide three angular values, according to the orientation of the user's head.

Bank 0	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	1	1	1	1	1
Row 2	0 (C2)**	1	AZ0	AZ1	AZ2	AZ3
Row 1	1 (C1)	1	AY0	AY1	AY2	AY3
Row 0	0*	1	AX0	AX1	AX2	AX3

Bank 1	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	0**	1	1	1	1
Row 2	0 (C2)**	1**	AZ4	AZ5	AZ6	AZ7
Row 1	1 (C1)	1**	AY4	AY5	AY6	AY7
Row 0	1*	1**	AX4	AX5	AX6	AX7

* Bit B0/B2 of row 0 is used to synchronise the cycle of banks. It will always be zero in bank 0, while all other banks will return 1. Banks will cycle in the order Bank 0, Bank 1, Bank 2, Bank 0, etc. See **Reading Bank Switching Controllers** for more information.

** The C3 and C2 bits identify the basic controller type. The B1/B3 bits of the last bank of the controller are used to identify the specific bank switching controller type.

Value	Meaning
AX(7:0)	Rotation angle around x (=roll=head titled) axis
AY(7:0)	Rotation angle around y (=yaw=looking left/right) axis
AZ(7:0)	Rotation angle around z (=pitch=looking up/down) axis

Zero is facing straight ahead. Positive values are tilt left/look left/look up. Values are linear angle values, where +180 degrees = $7F, -179 degrees = $80.

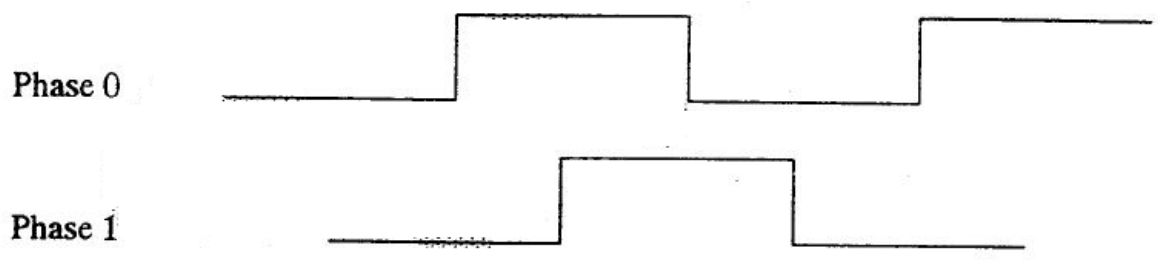
When connected directly to a Jaguar controller port, the controller will respond to socket 1 row codes (see **4-Player Adaptor**). A pass-through connector allows a second controller to be connected (usually a standard Jaguar controller, for compatibility reasons), which will receive socket 0 row codes and appear as if it was directly connected to the Jaguar. When connected to a 4-player adaptor, the pass-through connector will not function, and the controller will respond to socket 0 row codes.

Rotary "Tempest" Controller

This device is similar to the original Tempest arcade controller. It uses a two optical switch, which can be read by software to determine the direction of rotation.

	B2 B0	B3 B1	J12 J8	J13 J9	J14 J10	J15 J11
Row 3	0 (C3)					
Row 2	1 (C2)					
Row 1	1 (C1)					
Row 0					Phase 0	Phase 1

The phase signals (Phase 0 and Phase 1) specify which direction to rotary wheel is turning. They look like this when the wheel is turning anticlockwise:



In other words:

Anticlockwise sequenz	J10	(pin 12)	0	1	1	0	0	1	1	
	J11	(pin 11)	0	0	1	1	0	0	1	...
Clockwise sequence	J10	(pin 12)	0	0	1	1	0	0	1	
	J11	(pin 11)	0	1	1	0	0	1	1	...

When connected directly to a Jaguar controller port, the controller will respond to socket 1 row codes (see **4-Player Adaptor**. A pass-through connector allows a second controller to be connected (usually a standard Jaguar Controller, for compatibility reasons), which will receive socket 0 row codes and appear as if it was directly connected to the Jaguar. When connected to a 4-player adaptor, the pass-through connector will not function, and the controller will respond to socket 0 row codes.

Analog Joystick and "Driving" Controllers

These devices typically require 8 bits of analog resolution in 2 dimensions (X and Y). Two 100Kohm linear potentiometers are typically used, with a +5volt potential across the ends. The center wiper will then read a voltage between 0V and +5V.

To read this voltage requires an analog to digital converter (ADC). A good solution is to use the Motorola 68HC05P9 microcontroller. This part has four 8 bit ADC channels and 16 general purpose digital I/O lines. The four controller row outputs would be used to select one of our 6 bit addresses. The two 8 bit ADC values use 16 addresses, leaving room for 5 switches and 3 device identifier codes.

In the example below, we have used bank switching to support even more switches. The bank is switched when the 68HC05 sees a transition from Row 3 to Row 0. Bank identification is achieved by reading bits B0/B2 of Row 0. See **Reading Bank Switching Controllers** for more information.

Bank 0	B2 B0	B3 B1	J12 J9	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	D	Y(4)	Y(5)	Y(6)	Y(7)
Row 2	0 (C2)**	C	Y(0)	Y(1)	Y(2)	Y(3)
Row 1	1 (C1)	B	X(4)	X(5)	X(6)	X(7)
Row 0	0	A	X(0)	X(1)	X(2)	X(3)

Bank 1	B2 B0	B3 B1	J12 J9	J13 J9	J14 J10	J15 J11
Row 3	1 (C3)**	1**	1	1	1	1
Row 2	0 (C2)**	1**	1	1	1	1
Row 1	1 (C1)	1**	1	1	1	1
Row 0	1	1**	Up	Down	Left	Right

* Bit B0/B2 of row 0 is used to synchronise the cycle of banks. It will always be zero in bank 0, while all other banks will return 1. Banks will cycle in order Bank 0, Bank 1, Bank 2, Bank 0, etc. See **Reading Bank Switching Controllers** for more information.

** The C3 and C2 bits identify the basic controller type. The B1/B3 bits of the last bank of the controller are used to identify the specific bank switching controller type.

	"Stick" Controller	"Driving" Controller
X(7:0)	Roll Right = Positive delta values from centered position Left = Negative delta values from centered position	Steering Right = Positive delta values from centered position Left = Negative delta values from centered position
Y(7:0)	Pitch Forward = Positive delta values from centered position Back = Negative delta values from centered position	Acceleration/Brake Accelerate = Positive delta values from centered position Brake = Negative delta values from centered position
Up	Hat switch "Up"	Gear shift Up
Down	Hat switch "Down"	Gear shift Down
Left	Hat switch "Left"	Spare 1
Right	Hat switch "Right"	Spare 2
A	Top switch	Spare 3
B	Trigger switch	Spare 4
C	Middle switch	Spare 5
D	Lower switch	Spare 6

The range of possible X and Y values is 0-255, but not all controllers will use this entire range, and the range they do use is not pre-defined. Do not assume that certain constant values can always be used for the center, hard right, and hard left position. Analog devices are different from controller to controller, and even from day to day as temperature and humidity conditions change. For example, a driving controller may return values of 160 (steering wheel centered), 245 (turned hard right), and 75 (turned hard left). A different controller of the same type from the same company (or the same controller under different temperature and/or humidity conditions) may return values of 150 (center), 240 (hard right) and 55 (hard left). The center position is different, and the value ranges are also different. Your software needs to be able to account for this.

It will be necessary to provide some sort of calibration routine where your program will ask the user to move the controller to certain positions, in order to read the values at those positions[2]. This should be an option on your controller configuration screen. It would also be nice if the user could choose to recalibrate the controller while paused in the middle of a game. It would be another nice touch if you stored the current calibration values into the cartridge EEPROM. That way, if the user is using the same controller under the same basic conditions most of the time, they won't be forced to recalibrate each time they play.

Analog controllers require a certain amount of processing time from the time the row code is written to the JOYSTICK register until the data read back from the JOYSTICK or JOYBUTS registers will be valid. With a typical analog controller, this delay is normally about 25 microseconds (worse case is about 40 microseconds) when going from row to row within the same bank (this delay applies to all

[2] If you've ever played a game on a PC that uses an analog joystick, then you've probably seen examples of such calibration screens.

bank-switching controllers), and approximately 200 microseconds in between banks.[4] There are two ways to handle this. You can do a small delay loop while waiting for the data to be available (do this in a way that uses the bus as little as possible, i.e. avoid memory accesses). Or if your program has a timer interrupt of some kind, you could write out the row code on one interrupt, and then wait for another interrupt before reading the value back. You could also use GPU interrupts in a similar way.
Whichever way you choose, try to avoid wasting CPU time and bus bandwidth just waiting to read the controller(s) when there is other processing you could be doing.

When connected directly to a Jaguar controller port, the controller will respond to socket 1 row codes (see **4-Player Adaptor**). A pass-through connector allows a second controller to be connected (usually a standard Jaguar Controller, for compatibility reasons), which will receive socket 0 row codes and appear as if it was directly connected to the Jaguar. When connected to a 4 player adaptor, the pass-through controller will not function, and the controller will respond to socket 0 row codes.

Keyboard/Mouse Interface

Note: The specification for this controller type is still in the preliminary stages and is subject to change without notice. Contact Jaguar Developer Support or turner information if your project requires this type of controller.

[MF3]

Reading Bank Switching Controllers

One subject that has been discussed a number of times through this section is bank switching, a technique which allows a controller to return more information that would otherwise be possible with a single controller.

Bank switching is done automatically when the controller sees a transition from row 3 to row 0 (of the same controller socket) It is not possible to read only a particular bank or set of banks and ignore the other ones; you must always read all banks even if you don't really need all of the information.
Program must always read an entire bank from a controller at once. However, it is not required that you read all banks from a single controller in a single pass. It is acceptable to read a bank from one controller, followed by a bank or multiple banks from other controller, and then come back to read the next bank from the first controller. Controllers are expected to ignore any requests for rows on other controllers. Such requests must not cause the controller to lose synchronization or perform any bank switching.

The rows of each bank of a controller must be read in sequence: Row 0, Row 1, Row 2, Row 3. The controller relies on the rows being read in sequence so that it can start processing the data for the next row in advance. The results of reading rows out of sequence are undefined; the data returned by the

[4] These numbers were arrived at using a sample prototype analog driving controller using the Motorola 68HC05 microcontroller.

controller may be invalid. For example, your program would read data from an analog joystick controller like this:

Bank 0: **Row 0, Row 1, Row 2, Row 3,**
(controller will automatically bank switch here)
Bank 1: **Row 0, Row 1, Row 2, Row 3.**

It is not necessary to know in advance which bank is active when you start reading. If you read all banks of a controller into a table, you can look at the data afterwards to figure out where **Bank 0** is, and from there you can figure out where the data for the other banks must be. For example, if you were reading a driving controller, the data you read would end up in a table that looks like this:

Bank 0							
Row 0		**Row 1**		**Row 2**		**Row 3**	
word 0	word 1	word 2	word 3	word 4	word 5	word 6	word 7
joystick	buttons	joystick	buttons	joystick	buttons	joystick	buttons
Bank 1							
Row 0		**Row 1**		**Row 2**		**Row 3**	
word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
joystick	buttons	joystick	buttons	joystick	buttons	joystick	buttons

The bottom row of the table would be an array of WORD values read from the JOYSTICK and JOYBUTS registers. You could store these values into separate arrays if you prefer, and it is not necessary to read both the JOYSTICK register and the JOYBUTS register for each row, but this example assumes you are always reading both registers and storing all the results into a single table for further processing.

In this example, Bank 0 came first, but that won't always be the case. You need to examine the data in the table to determine the location of each bank of data. Bank switching controllers always indicate **Bank 0** by setting bit 0 (**B0** of controller port 1) or bit 2 (**B2** of controller port 2) of the value read from the JOYBUTS register from Row 0. The bit will be 0 for **Bank 0** and 1 for all other banks. Because the banks are always read in sequence, once you find **Bank 0** in the table, then you know where to find the data for all the other banks.

In the example above, because bit 0 of word 1 was clear (assuming controller port 0), then you would know that the data for **Bank 0** was in words 0-7. Since we only have two banks, that means the data for **Bank 1** must be in words 8-15.

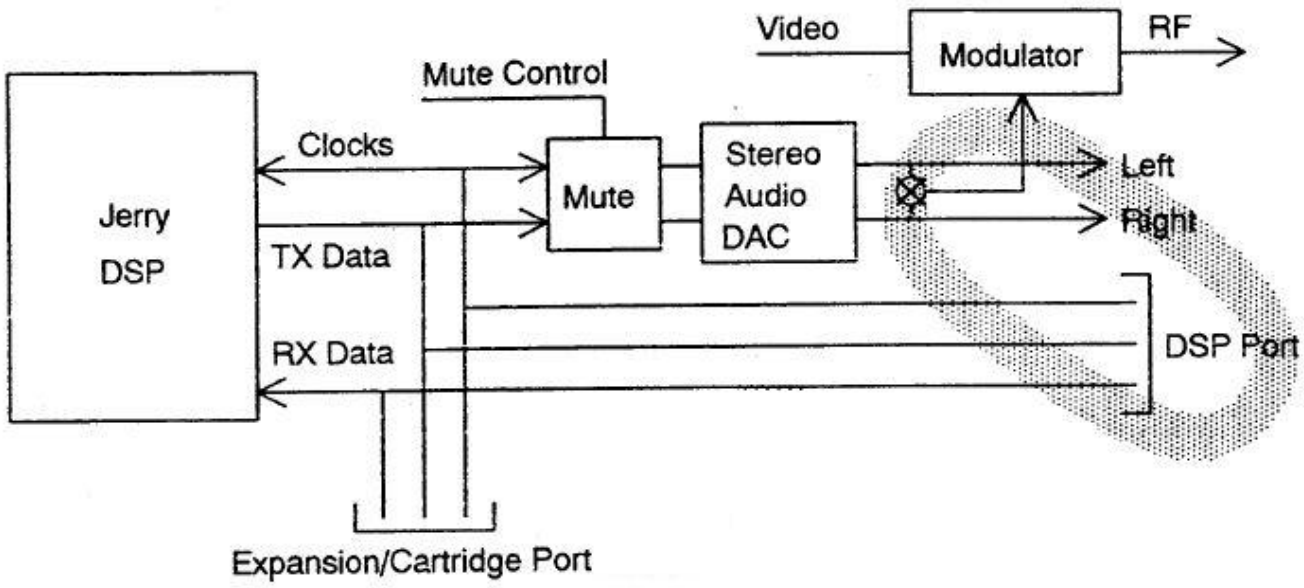
Suppose you had a 6D Controller, which has 3 different banks of information, connected to port 0. After reading 3 banks' worth of information from this controller, you might end up with a buffer that looks like this:

Bank 2							
Row 0		**Row 1**		**Row 2**		**Row 3**	
word 0	word 1	word 2	word 3	word 4	word 5	word 6	word 7
joystick	buttons	joystick	buttons	joystick	buttons	joystick	buttons
Bank 0							
Row 0		**Row 1**		**Row 2**		**Row 3**	
word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
joystick	buttons	joystick	buttons	joystick	buttons	joystick	buttons
Bank 1							
Row 0		**Row 1**		**Row 2**		**Row 3**	
word 16	word 17	word 18	word 19	word 20	word 21	word 22	word 23
joystick	buttons	joystick	buttons	joystick	buttons	joystick	buttons

The first thing you need to do is find the data for **Bank 0**. First you would look at bit 0 of word 1, then word 9. In this example, word 9 would have bit 0 clear to indicate **Bank 0**. Therefore, words 8-15 contain the data for **Bank 0**. Once you know that, then you also know that **Bank 1** is contained in words 16-23 and **Bank 2** must be in words 0-7.

Note that there is a certain amount of processing time required when switching from one row to the next, because the microcontroller inside the controller has to put a different set of data on the outputs. This is normally approximately 25 microseconds (worse case is about 40 microseconds) when going from row to row within the same bank. Analog controllers typically also require an additional 200 microseconds when going from one bank to the next (so that the analog inputs may be digitized). See the **Analog Joystick And Driving Controllers** section for ideas about how to deal with this.

Audio Subsystem



The Jaguar console includes a stereo 16 bit audio subsystem. Digital audio data can only be sourced from the Jerry DSP. This data can also be monitored at the expansion or DSP ports, on the TXD serial data line. Jerry can also read serial digital audio data on its RXD pin. The bit clock and word strobe signals can be sourced by Jerry, the expansion port or the DSP port. If the clock source is not Jerry, then software must force the Jerry clock lines tristate, by clearing bit 0 of SMODE.

The Audio mute function has been added to allow non-audio data to be transmitted by Jerry, without making a horrible noise on the audio outputs. When serial peripherals are connected to the DSP port, and are in use, the audio should be muted by writing zero to bit 8 of the JOY1 joystick register ($F14000). Take great care not cause the J4-J7 outputs to all go low (by writing a 1 to bit 15 and 0 to bits 4-7 in the same register). This will inadvertently cause multi-player adaptors to go into extended mode.

Cartridges And NVRAM

The Jaguar console cartridge port supports up to 6 Megabytes of address space. Cartridges can be 8, 16 or 32 bits wide[5]. Special support is also included for serial EEPROMs. Reading and writing the EEPROM **must** be done through the Atari supplied routines. (See the sample program for accessing NVRAM.) This is the only way to ensure reliable operation.

Bit 0 of the JOYSTICK register, when read, represents the data output bit of the EEPROM, and **not** the J0 input from the joystick. Since J0 has always been used as an output only so far, this should not cause problems. But bear in mind that this data bit is now random when read, and not equal to the J0 output bit as before.

It should be noted that the EEPROM uses addresses in the GPI00 and GPI01 range ($F14800-$F15FFF). Any inadvertent access (reads or writes) to these address ranges will cause subsequent EEPROM reads and writes to fail. So don't do it ...

When you build your own 32-bit test cartridges using Atari's 4-chip EPROM cartridge blanks, the ordering of data in the chip is as follows:

Chip	Bytes	Bits in 32-bit long
U1	$800003, $800007, $80000B, etc.	(d0-d7)
U2	$800002, $800006, $80000A, etc.	(d8-d15)
U3	$800001, $800005, $800009, etc.	(d16-d23)
U4	$800000, $800004, $800008, etc.	(d24-d31)

In a non-encrypted cartridge, locations $800000 to $801FFF should have values of $FF. Your program code should always start at $802000; in both encrypted and non-encrypted cartridges.

Burning Your Own Cartridge EPROMs

For those wanting to use an EPROM burner to create their own non-encrypted test cartridges, any EPROM burner capable of handling 4 megabit EPROM chips should be acceptable.

If you would like a specific recommendation for a particular EPROM burner, Atari has had good success with the **Pilot** EPROM Burner, manufactured by **Advin**. This burner is relatively fast, and can handle an entire set of EPROM chips at once. The table below shows the model numbers, a description, and the price of the base unit and accessories:

Pilot 832D	Base unit plus Gang Faceplate 832D for up to DIL-32 Pin EPROM / 4 megabit	$1500.00 (includes base unit and software)
Pilot 844D	Replacement Gang Faceplate for up to DIL-44 Pin EPROM / 16 megabit	$1095.00 (upgrades Pilot 832D to Pilot 844D)

[5] At this time, the Stubulator ROM used in development machines currently only supports the use of 32-bit wide cartridges.

Pilot 844D complete package	Base unit plus Gang Faceplate 844D for up to DIL-44 Pin EPROM / 16 megabit (Note: this unit does not include the 832D faceplace, and CANNOT handle 32 Pin EPROMs !!)	$1795.00 (including base unit and software)

This burner can burn a 4 megabit EPROM in approximately 3:08 minutes, or a 16 megabit EPROM in under 15 minutes.

Please note that all prices shown are based on the latest information obtained by Atari, and are subject to change without notice. These EPROM burners are not available directly from Atari. Plea contact Advin to inquire about purchasing these products. To contact Advin from North America:

Advin
1050-L East Duane Ave.
Sunnyvale CA 94086
TEL 408-243-7000 FAX
408-736-2503

Technical questions: ask for Edwin
Sales information: ask for Susan

Advin's USA office can handle out of country delivery if necessary, but they may have a local distributor. The distributor in England is (to obtain information about distributors for other countries in Europe, please contact Advin):

Quarndon Electronics Ltd.
Slack Lane
Derby DE3 3ED
England
Tel.: (+44) 332-32651

EPROMs For Making Test Cartridges

The following EPROM types have been successfully used in Atari's test department:

For a 4x4 EPROM cartridge with 128 byte EEPROM, a cartridge uses (4) 512kBit x 8 (4 megabit) chips.

Manufacturer	Chip Code
Macronix	MX27C4000DC-12 or MX27C4000-15
Toshiba	TC574000AD-120 or TC574000AD-150
AMD	AM27C040-150DC

For a 16x2 EPROM cartridge with 128 Byte EEPROM, a cartridge uses a single 1024kBit x 16 (16 megabit) chip:

Manufacturer	Chip Code
Toshiba	TC5716200 (Atari is currently looking for compatible parts)

Chips with access speeds slower those shown above are not recommended. Similar chips from other manufacturers may work, but have not been tested by Atari. Try them at your own risk. However, if you do find other chips that work, please contact Atari's Developer Support department and let them know so that they can be added to the list.