Nintendo 64 Controllers

While I was in Pittsburgh for an extended period of time for work, I got an urge to interface an N64 controller to one of my projects. Don't ask why, I get these weird urges sometimes. heh.

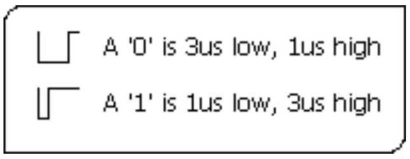
Before we get too far into this -- I am only interested in the hardware level control of the controller and devices attached to the controller. I'm not interested in emulation of the N64 or c0py1nG y3r 31337 r0m 1m4G35, so **DON'T ASK**. The following links are to places with similar technical or emulation interests. I doubt they are interested in swapping ROMs, either.

Dextrose
Steven Hans' N64 Stuff
Ken Kaarvik's Gameboy Stuff

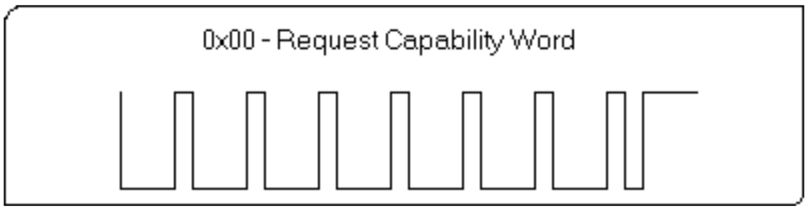
Anyway, upon further inspection, the N64 <--> Controller protocol is relatively simple to imitate. This page is dedicated to furthering the understanding of all aspects of the controller. Right now I can get it to return the button status, and with time I'll figure out how the RumblePak and MemPaks are talked to.

The electrical specification chosen by Nintendo seems very close to Apple's DeskTop Bus protocol, where all data transfer takes place on a single wire, and the duty cycle of the signal determining the data. Since I'm not familiar with the ADB protocol any more than that, I can't comment further.

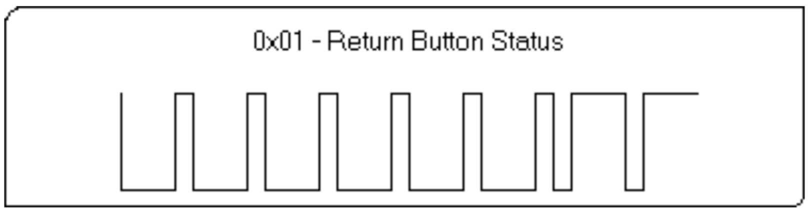
However, it is important that anyone who is interested in talking to this thing on a hardware level knows that there is a particular way in which you must interface -- **DO NOT ever drive the line high!** -- The N64 or the controller may bring the line low at any time and if you're trying to drive it high you may damage either the controller, the console, or whatever you're using to talk to it! The controller has a built-in pullup resistor -- either make damn sure that whatever you're using to play with it doesn't drive the line high or is incapable of driving it high. Open-collector outputs are handy for this kind of thing.



Here's the first step: understanding the encoding of bits on the 1-wire interface. It appears that all bits are 4us wide, and it's the amount of low time that determines the bit's value. Please note that the assignment of '1' and '0' has been done my myself; I could have them ass-backwards, but they seem to make sense to me. :-)
Also, the communications format includes a stop bit at the end of everything. All this involves is letting go of the line and waiting 4us before doing anything else.



This is sent out to the controller upon powerup, and some games (StarFox64 does this) send this every time they want to read button status as well. Why, I don't know. I don't have all the bits figured out yet, but I have 3 scope traces that give some insight... A controller with a MemoryPak gives a different response from one with a RumblePak and they both give different responses than one without any kind of cartridge plugged in. (*I'll put this up as soon as I get the scope software reinstalled.*)



Sending 0x01 to the controller makes it respond with 32 bits of button information. Please note that the controller responds VERY QUICKLY -- within 2 to 3us on my particular unit. Whatever you're using to talk to this thing has to be very quick. An 8MHz PIC would have a difficult time doing this. Better to use a 10-20MHz one and bit-bang it.

The information is in the following format:

(1 = button pressed, 0 = button released)

Bit	**Function**
0	A
1	B
2	Z
3	Start
4	Directional Up
5	Directional Down
6	Directional Left
7	Directional Right
8	*unknown (always 0)*
9	*unknown (always 0)*
10	L
11	R
12	C Up
13	C Down
14	C Left
15	C Right

The next 16 bits are for the analog joystick.

The analog joystick is an optical module and operates very much the same as most mice. If I recall correctly, it has 24 'positions' in each direction from center, with center returning 0. If you're not interested in using the whole controller, this joystick is a seperate module and can be removed from the controller. It's easy to use... 6 pin interface, 2 for power, then the remaining 4 are for the X and Y I/Q signals. Very nice. Not too bad in price, either.

The information returned in the last 16 bits from the controller is for the signed X and Y information, respectively. i.e. the first 8 bits are the signed X position, and the last 8 are for the signed Y position. From center: up and right is positive, down and left is negative.

28 July 1999 - I finally got around to updating this page. I had made a mistake in the first edition of this information -- The controller sends a stop bit which I was mistakenly including as data. The data above is now corrected. Thanks to those of you who emailled me with that correction.

The folks at the N64 Controller Interface link above have discovered how to make the rumble pak function. That information is coming here shortly for inclusion. I am also close to having my N64 controller reconnaissance unit working, so getting full info on the rumble and mempaks should be coming soon. :-)

Right now, here is the base info on how to do it:
To Init: send 03 80 01 followed by 34(!) 80's
To Start Rumble: 03 c0 1b followed by 32 01's
To Stop Rumble: 03 c0 1b followed by 32 00's
After starting or stopping a rumble, the controller returns with 3 bytes. I have not personally done this before, so I don't know what those three bytes are (yet). Thanks go to Ken Kaarvic for posting this info to the Gameboy development list, and further thanks go to Stephen Hans for providing Ken with the original information.

Ken Kaarvic has used this information here to successfully connect an N64 controller to a Gameboy through the link port. Head over to his page for more information on that.

When I get some time I want to get a little PIC connected up inbetween the controller and the console and start having it do reconnaissance so I can get more information regarding RumblePaks and MemPaks. After that I'd like to get the services of a skilled WinBlows 95 VXD writer to create a nice VXD so I can use the N64 as a gaming device, complete with functionality for RumblePaks and MemPaks. Who knows, this could be marketable. I know there are some out there now that have adapters but AFAIK nobody can talk to RumblePaks and MemPaks.

Ideas for N64 data:

- ROOMblePak - modify a standard Rumblepak to drive a small AC motor with an offset weight and place it under your couch :-)
- TerrorPak - Same idea, but give the user a mild electrical shock instead of rumbling

I've actually had emails regarding the latter idea! My suggestion: optically isolate and on the side where you connect to your body -- **USE A BATTERY** -- very important. You don't want any kind of link to AC or significant current. That keeps your masochistic tendencies at least safe. :-)

Ideas I'm using this information for:

- N64 Controller Reconnaissance - a PIC which intercepts and echoes console <--> controller communications and spits out the data to a PC through RS-232.
- N64-PC Interface - There are a number of people already doing it. Here is why mine's different:
  - uses VERY simple interface -- one or two ICs
  - interfaces to game port like any other joystick
  - useable by any game through DirectX style interface
  - open-source, open design model
  - works with any N64 controller. Jittery third party units will work okay
  - works with Rumble and Memory Paks through standard interface
- N64 Macro controller?
- N64 Keyboard?

Email me if you're interested.

Last updated: 28 July 1999