N64 Controller

Dmitry Bondarenko
Willie Chen
Erik Meade
Henry Wang

Agenda

- Nintendo 64 Background & Usage
- Interface Introduction
- Communication Protocol Overview
- Controller Peripherals
- Possible Implementation Scheme

N64 Console Background

- Gaming console released by Nintendo in 1996
- 64-bit processor @ 93.75 MHz, 562.5 MB/s bus, 640x480 px, 4MB D-RAM
- Used cartridges to store and load games
- One of the first consoles to use analog stick for navigation

https://commons.wikimedia.org/wiki/User:Evan-Amos/VOGM/N64

Example Projects

Example Projects

Example Projects

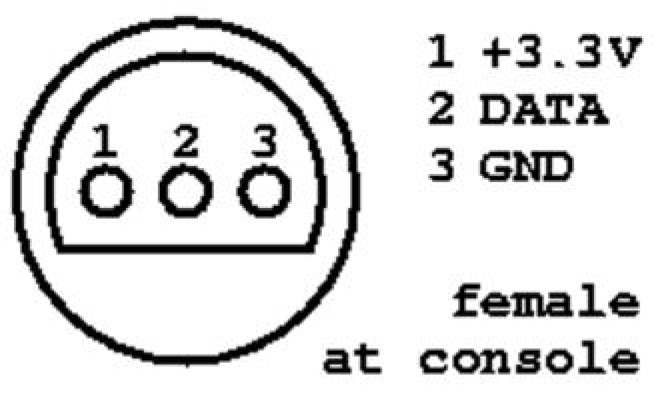
N64 Controller Motivation

- Project involving mobile robots
- Intuitive and convenient controller with analog stick
- Ample control options
- Less complex and more documented than modern controllers like PS and XBox
- Familiarity and well-known interface

http://www.gratuitousscience.com/?p=3033

N64 Interface Introduction

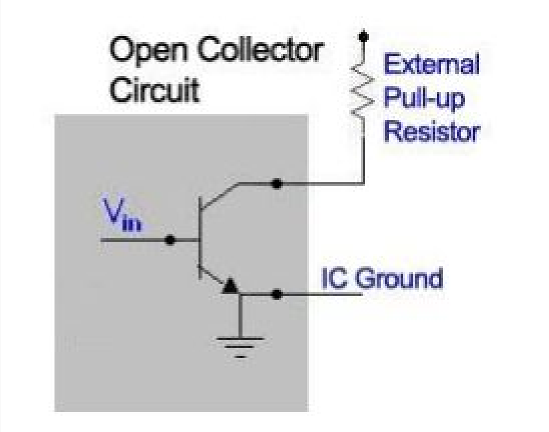
- 3 Pins
  - Vref (3.3V)
  - Ground
  - Data/Clock Line, open collector
- Single data wire interface
  - Every falling edge initiates a bit transaction
    - Self clocking
    - Similar to uart, goes at a defined rate
  - ~2 us after the falling edge, read bit (0/1) from the line



http://ezhid.sourceforge.net/n64pad.html

N64 Interface Introduction

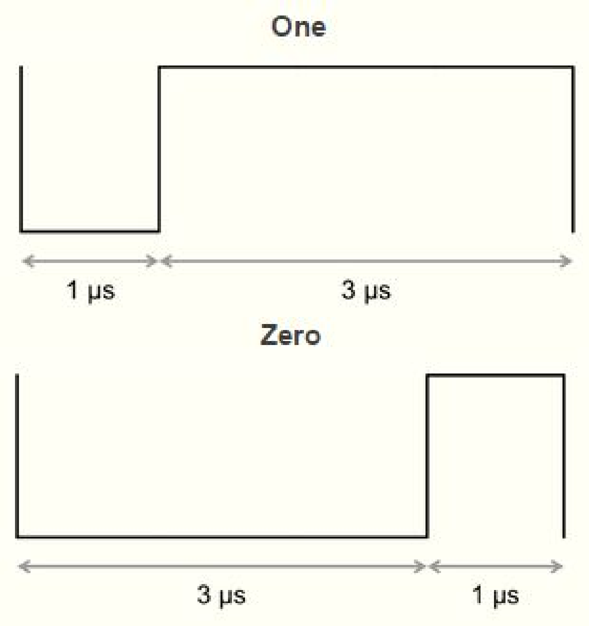
- Open collector
  - Allows data to travel through the data wire to and from the controller
  - Used instead of a tri-state



https://www.eecs.umich.edu/courses/eecs270/lectures/270L23NotesF14.pdf

N64 Interface Introduction

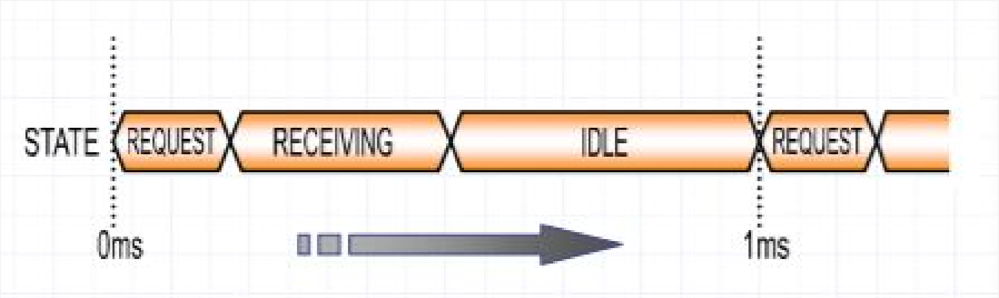
- For each bit, the transaction always begins/ends with a falling edge
- The one bit is low for 1 us after the falling edge then goes high for 3 us
- The zero bit stays low for 3 after the falling edge then goes high for 1 us



http://www.pieter-jan.com/node/10

N64 Communication

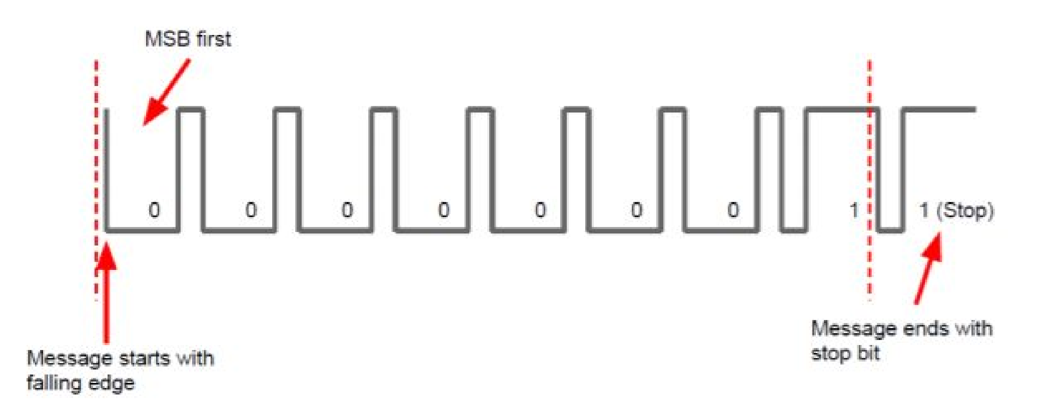
- Three phases in the communication sequence
  - Requesting, receiving, idling
- Console is the master, controller is the slave
- Request: the console requests data from controller
- Receive: either the controller or the console writes on the data line
- Idle: the data line then becomes idle



http://www-inst.eecs.berkeley.edu/~cs150/sp01/Labs/lablecckpt1.pdf

Sending commands to the N64 controller

- The console initiates transactions
- Commands are either a read or a write
- Example of initiating a button status read transaction (0x01)
- Ends with Stop bit



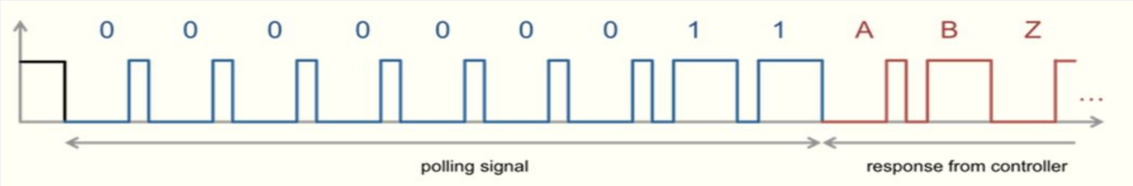
https://www.eecs.umich.edu/courses/eecs270/lectures/270L23NotesF14.pdf

Commands

Command	Description
0x00	Request info (Rumble Pak or mem card?)
0x01	Read button values
0x02	Read from memory pack
0x03	Write to memory pack
0x04	Read EEPROM
0x05	Write EEPROM
0xFF	Reset

Controller Response

- Controller responds with button statuses
- Bits 0-7, 10-15: push buttons, active-high
- Bits 16-31: analog stick coordinates, represented as a two's complement integer between -80 and 80
- Transaction ends with a stop bit



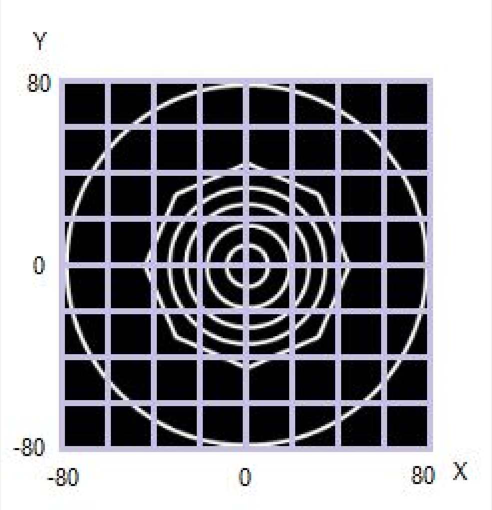
http://www.pieter-jan.com/node/10

0	A
1	B
2	Z
3	Start
4	Up
5	Down
6	Left
7	Right
8	"Joystick Reset"
9	(0)
10	L
11	R
12	C-Up
13	C-Down
14	C-Left
15	C-Right
16-23	X-Axis
24-31	Y-Axis
32	Stop bit (1)

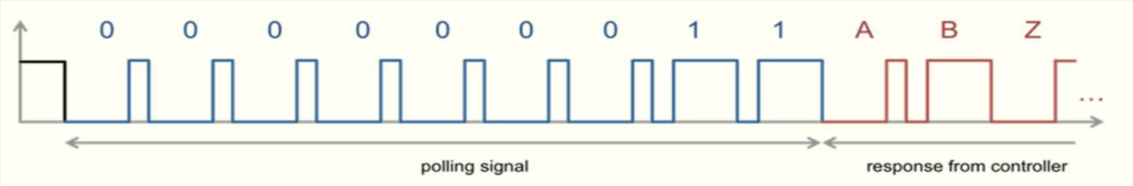
https://www.eecs.umich.edu/courses/eecs270/lectures/270L23NotesF14.pdf

Controller Response

- Controller responds with button statuses
- Bits 0-7, 10-15: push buttons, active-high
- Bits 16-31: analog stick coordinates, represented as a two's complement integer between -80 and 80
- Transaction ends with a stop bit



12	C-Up
13	C-Down
14	C-Left
15	C-Right
16-23	X-Axis
24-31	Y-Axis
32	Stop bit (1)



http://www.pieter-jan.com/node/10

https://www.eecs.umich.edu/courses/eecs270/lectures/270L23NotesF14.pdf

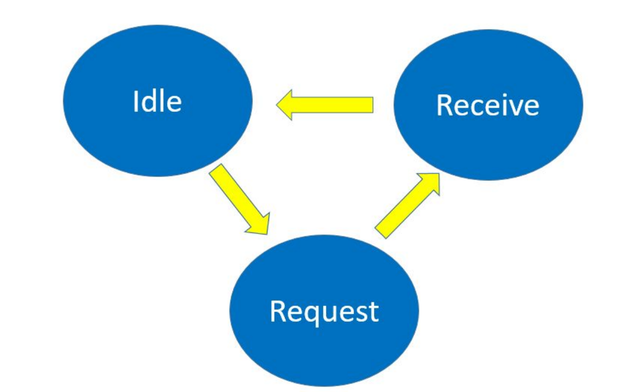
Controller "Peripherals"

- Memory slot can have either Rumble Pak or memory card
- Memory slot:
  - Read: 0x02
  - Write: 0x03
- Send 0x00 to know if it's a Rumble Pak or memory card
- If Rumble Pak
  - Send 32 byte block of 01's to turn on
  - Send 32 byte block of 00's to turn off

https://en.wikipedia.org/wiki/Rumble_Pak

Interfacing with the N64

- The FPGA periodically sends a request byte to the controller (polling)
- After sending the request byte, FPGA will process incoming data
  - Use the single data wire as a clock to synchronize the reading of the bits sent by the controller
  - Use the system clock to specify reading exactly 2 us after the falling edge of the data wire
- Then enter idle state and wait for the next time to poll the controller



Interfacing with the N64

- After receiving data from the controller, check for changes
- Check by comparing with previous button status
- If yes, trigger an interrupt
- Otherwise, enter idle state
- Button status data will be available to the Cortex M3 via MMIO

Questions?