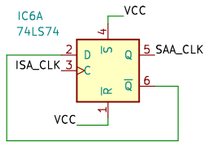
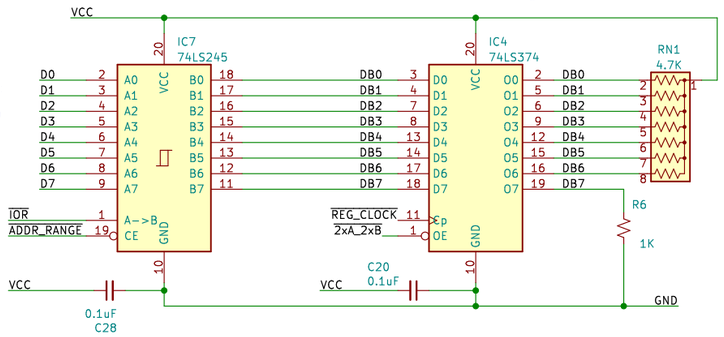
Clock Divider



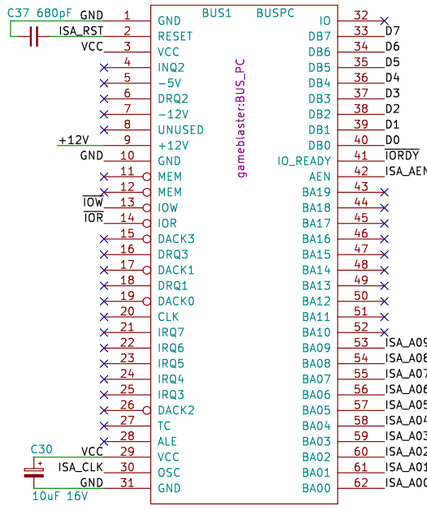
ISA Bus clock is 14.318MHz, nominal SAA1099 clock is about 8MHz, the Game Blaster runs the SAA1099s at 7.159MHz by dividing the ISA clock in half.

Bus transceivers are apparently good practice, so we take the ISA data bus D0–D7 and hook it up to a 74LS245 here. When IOR is pulled low (i.e. when we're reading from 2xA/2xBh or 2x4h on this card) we drive its direction pin to do B–>A; otherwise, we're writing, and we run it A–>B. CE is controlled by the output of the first decoder, which is low when an address within our base port range is accessed.

Detection Register

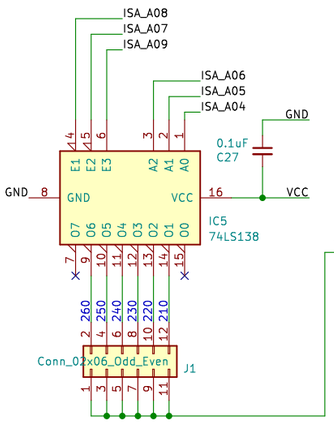


We use a 74LS374 octal flip–flop as a register to store the value written to 2x6/2x7h, and read it back on 2xA/2xBh, by putting DB0–DB7 on both the input and output. REG_CLOCK is the AND of IOW and 2x6_2x7, which means that we latch data into the register when we write there. OE is controlled by 2xA_2xB, so that when we read from those ports, the outputs are placed onto the transceiver bus, and back through to the ISA bus. When the flip–flop outputs are inactive, DB0–DB6 are pulled up through 4.7K resistors to VCC, and DB7 is pulled down through a 1K resistor to ground, which puts 0x7F on the bus if we read anywhere else (with the intent that we read it back on 2x4h).



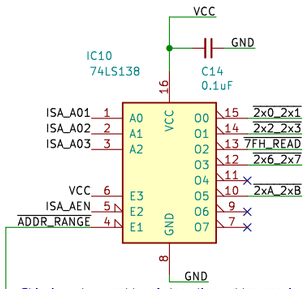
Address Decode

Port Range



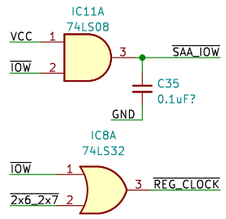
Base Address – 210h to 260h

Register/Addr



This is only capable of decoding addresses in pairs, which is fine because the SAA1099s each have two registers. 2x0–2x1h are the left SAA1099, 2x2–2x3h are the right SAA1099. 2x4 is used on the original card to read back a status bit from the DIP–40, but conventional wisdom says "just read 7Fh". Card detection requires accepting a write to 2x6h (or 2x7h), and reading the same value back on 2xAh or 2xBh.

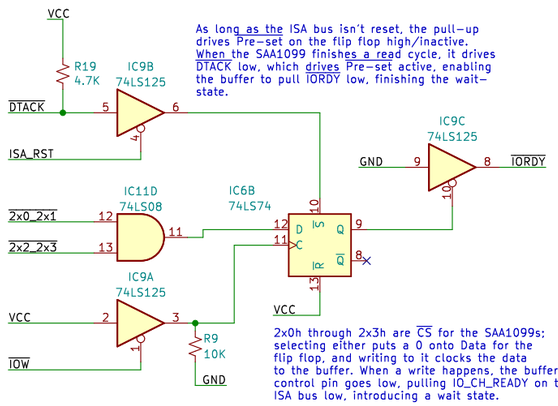
Write Logic



The original Game Blaster includes this AND of VCC and IOW to drive WR on the SAA1099. I suspect because the timing diagrams on the SAA1099 datasheet require that WR goes low slightly after CS goes low to start the data transfer. (Are these caps here to add delay? How do capacitors work?)

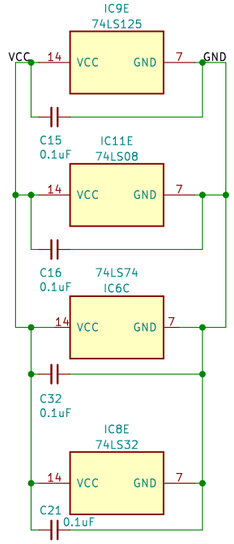
Since we want to save the contents of the bus into the detection register when a write is made to 2x6h or 2x7h, we clock the 74LS374 when IOW and 2x6_2x7 are both low. (i.e. when the host indicates a write, and when the address selected is the "ID write" port)

Wait State Latch

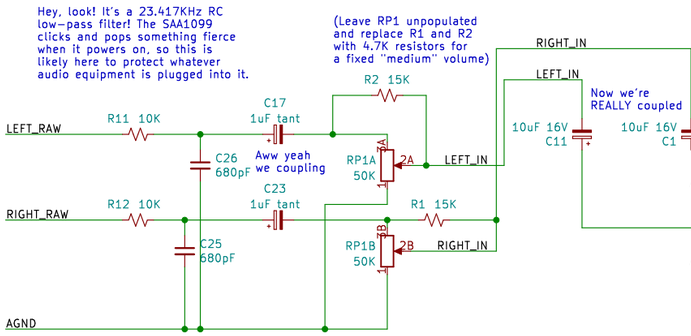


As long as the ISA bus isn't reset, the pull–up drives Pre–set on the flip flop high/inactive. When the SAA1099 finishes a read cycle, it drives DTACK low, which drives Pre–set active, enabling the buffer to pull IORDY low, finishing the wait–state.

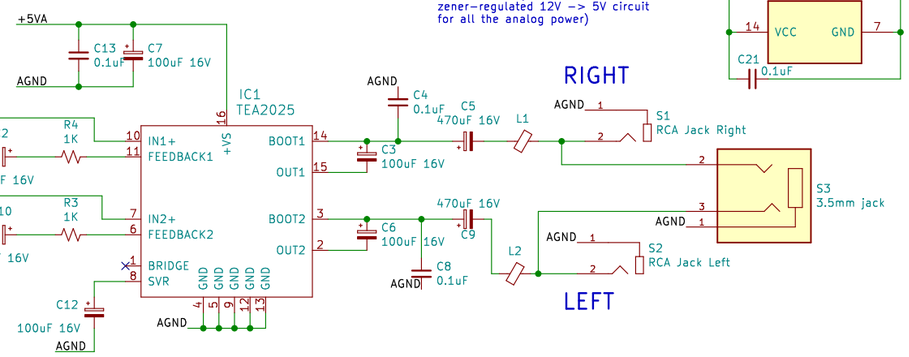
2x0h through 2x3h are CS for the SAA1099s; selecting either puts a 0 onto Data for the flip flop, and writing to it clocks the data to the buffer. When a write happens, the buffer control pin goes low, pulling IO_CH_READY on the ISA bus low, introducing a wait state.



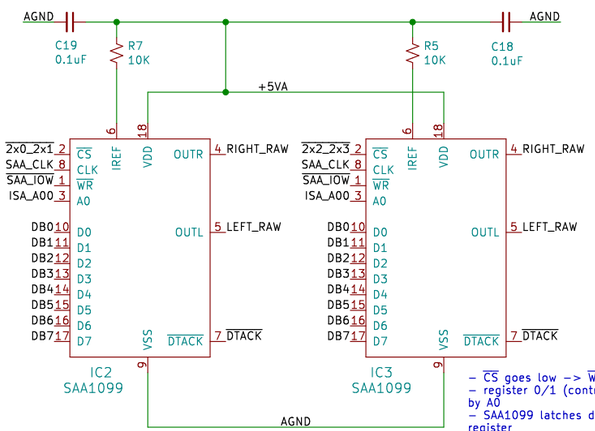
Volume Control



Audio Amplifier



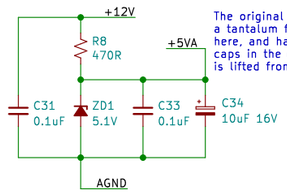
CMS Chips



Because CS is already guaranteeing we're accessing this specific SAA1099, we just use the A0 line of the ISA bus directly to determine which register to write to.

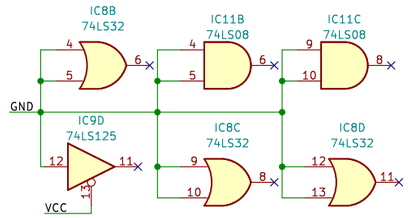
– CS goes low –> WR goes low
– register 0/1 (control/data) selected by A0
– SAA1099 latches data from D0–D7 into register
– SAA1099 pulls DTACK low when data trasnfer is complete

+12V to +5V Regulated

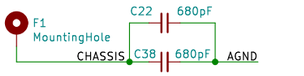


The original Game Blaster uses a tantalum for the polarized cap here, and has a couple extra 0.1uF caps in the circuit. This circuit is lifted from the Sound Blaster.

Unused



"EMI" "Protection"



Shame Master

Derivative Labs, Inc.

Sheet: /
File: gameblaster.kicad_sch

Title: CT–1300C

Size: A3	Date: 2023–02–07	**Rev: 1**
KiCad E.D.A. kicad (6.0.11)		Id: 1/1