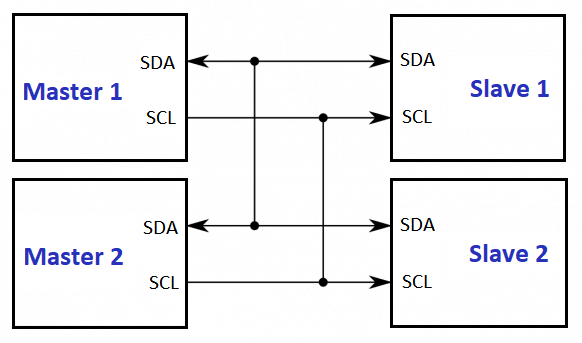
**I2C multi-master**



Marijn Verschuren 12-05-2023

# Table of contents

[1. Table of contents 2](#_Toc1)

[2. Table of figures 2](#_Toc2)

[3. Introduction 3](#_Toc3)

# Table of figures

**No table of figures entries found.**

# 3. Introduction

For this project i have chosen to make Tetris, this is a very simple game where blocks are dropped and the player is tasked to fill in rows in the grid which when filled are removed. The main challenge in this project was the fact that both players (micro-controllers) share a single screen, this screen is controlled via a I2C bus that is shared among the players. The issue with a single I2C bus is that only one micro-controller can write to it at the time, this is only made worst by the fact that the screen takes a 400 byte screen buffer so updating the screen takes a significant amount of time. If this issue is not properly dealt with the screen would not show anything useful.

# 4. Design

Bus arbitrage (two micro-controllers writing to the bus at once) is already detected by the Arduino library. Once the micro-controller notices that what it is sending is not actually appearing on the bus it will stop sending, this will however mess up the data that the other master was sending. This is why another level of arbitrage prevention is added, we have done this by sending a ‘token’ back and forth, only the micro-controller with the token is allowed to write to the bus. In this application no actual token is sent, we simply send the game state over to the other micro-controller which sets the token flag once it receives the game state. Sending the game state was the next problem because an instance of the game is way bigger than 16 bytes, this is an issue because the Arduino library only allows you to send 16 bytes of data in one go. This would mean that i have to send my game instance in pieces making it incredibly hard to receive and put together on the other micro-controller. So we have solved this by only sending the block that is currently moving, this does however open up possibilities for desynchronisation but that is a fair tradeoff for simpler code (also avoiding possible receive errors).