СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc512238194)

[1 ОБЗОР ЛИТЕРАТУРЫ 7](#_Toc512238195)

[1.1 Обзор существующих аналогов 7](#_Toc512238196)

[1.2 Архитектура аппаратного обеспечения 8](#_Toc512238197)

[1.3 Архитектура программного обеспечения 12](#_Toc512238198)

[2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 15](#_Toc512238199)

[ЗАКЛЮЧЕНИЕ 20](#_Toc512238200)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc512238201)

[ПРИЛОЖЕНИЕ А 22](#_Toc512238202)

[ПРИЛОЖЕНИЕ Б 24](#_Toc512238203)

# ВВЕДЕНИЕ

Современную промышленность невозможно представить без встраиваемых систем. Они применяются повсюду – от управления промышленными роботами и автоматическими межпланетными станциями до смартфонов и фитнес-браслетов. Иногда пользователи могут не догадываться, что внутри какого-либо устройства находится компьютер, выполняющий специальную программу.

В связи с таким разнообразием применений, конструкторы встраиваемых систем часто сталкиваются со сложными проектными задачами. Встраиваемые системы должны быть надежными. Многие встроенные устройства не могут ломаться, и не могут быть перезагружены. Программное обеспечение невозможно обновить во многих встроенных устройствах. Многие устройства имеют жесткие конструкционные ограничения по производительности и потреблению энергии. Некоторым устройствам необходимо работать от батареи длительный период времени. Кроме того, потребительские устройства обычно очень быстро выходят на рынок с новыми продуктами и имеют жесткую ценовую конкуренцию. Во многих приложениях существуют ограничения реального времени и многие устройства имеют ограниченную память и вычислительную мощность.

Поскольку встраиваемая система представляет собой комбинацию аппаратного и программного обеспечения, при разработке и отладке новой системы очень часто возникает неопределённость, когда неясно где искать ошибку – в аппаратном или программном обеспечении. Ещё одной проблемой является то, что элементная база для построения таких систем быстро устаревает, но во многих случаях не может быть легко обновлена из-за того, что переход на новую элементную базу потребует полного переписывания программного обеспечения.

Целью данного дипломного проекта является разработка и сборка прототипа игровой консоли на микроконтроллере, а также создание кроссплатформенного приложения (в данном случае игры), способного без изменения кода (но с перекомпиляцией) работать как на ПК, так и на встраиваемой системе, к которой относится разрабатываемая консоль.

В соответствии с поставленной целью были определены следующие задачи:

* разработка аппаратного обеспечения консоли;
* разработка игрового движка;
* разработка игры на созданном движке;
* разработка библиотеки, необходимой для работы кода разработанной игры и движка на разработанной аппаратной платформе.

# 1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор существующих аналогов

Перед началом проектирования были изучены аналоги разрабатываемой консоли и ПО для неё. Одним из немногих и наиболее близких аналогов является DIY-консоль, описанная в статье [1]. На рисунке 1.1 приведена структурная схема консоли, взятая из данной статьи.

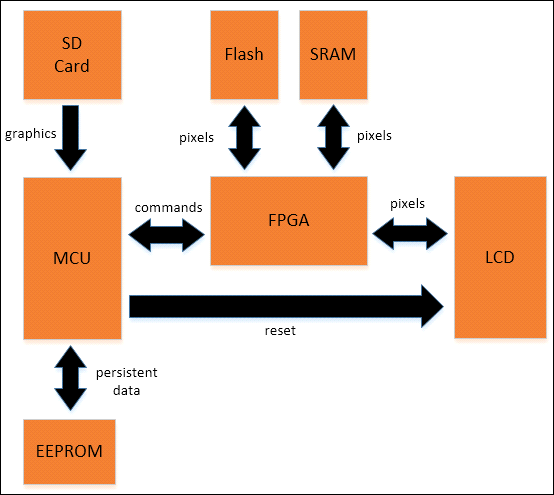


Рисунок 1.1 - Структурная схема консоли-аналога

Консоль построена на микроконтроллере STM32F429VIT6. Самой главной отличительной чертой этой консоли является использование аппаратного ускорителя 2D графики, выполненного на микросхеме FPGA. Сам же главный игровой цикл имеет стандартную архитектуру:

1. Прочитать состояние устройств ввода;
2. Сделать один шаг в игровой логике;
3. Обновить координаты всех спрайтов;
4. Отрендерить спрайты в задний буфер;
5. Произвести своп переднего и заднего буфера.

Особенность консоли проявляется в шаге 4. Вместо того, чтобы программно рендерить спрайты в задний буфер, процессор консоли формирует список команд для исполнения их на внешнем ускорителе. Сами текстуры спрайтов хранятся в памяти, подключённой непосредственно к внешнему ускорителю. Это позволяет процессору сразу же перейти к следующей итерации игрового цикла. Таким образом рендеринг текущего кадра и подготовка к рендерингу следующего будут идти параллельно.

Главный недостаток данной консоли заключается в её достоинстве - необходимость в микросхеме FPGA и изучении какого-либо из языков описания аппаратуры для реализации графического ускорителя на ней.

К другим недостаткам можно отнести:

* малое количество ресурсов, в частности быстродействие процессора (используется 180МГц ARM Cortex-M4) и объём оперативной памяти (у центрального процессора только 256Кб встроенной в микроконтроллер, у FPGA - 512 Кб в микросхеме статической памяти);
* использование нестандартного дисплея от мобильного телефона Sony Vivaz U5 с неизвестным интерфейсом и системой команд (автор статьи получил требуемую информацию путём реверс-инжиниринга);
* отсутствие интерфейса USB для подключения различных устройств ввода, отсутствие сенсорной панели на экране;
* отсутствие каких-либо возможностей вывода звука.

1.2 Архитектура аппаратного обеспечения

Аппаратное обеспечение разрабатываемой консоли будет строится на основе широко доступных комплектующих, недостатка в документации для которых нет.

Самый важный компонент консоли - микроконтроллер STM32F7. Поскольку это высокоскоростное устройство, критически важно правильно спроектировать принципиальную и монтажную схемы его подключения. Для этого этапа проектирования фирма STMicroelectronics выпускает набор документации, в частности [2] представляет собой подробное руководство по разработке аппаратного обеспечения с использованием микроконтроллеров STM32F7. Так, на рисунке 1.2 показана схема подключения источников питания и конденсаторов для развязки питания, взятая из данного руководства.

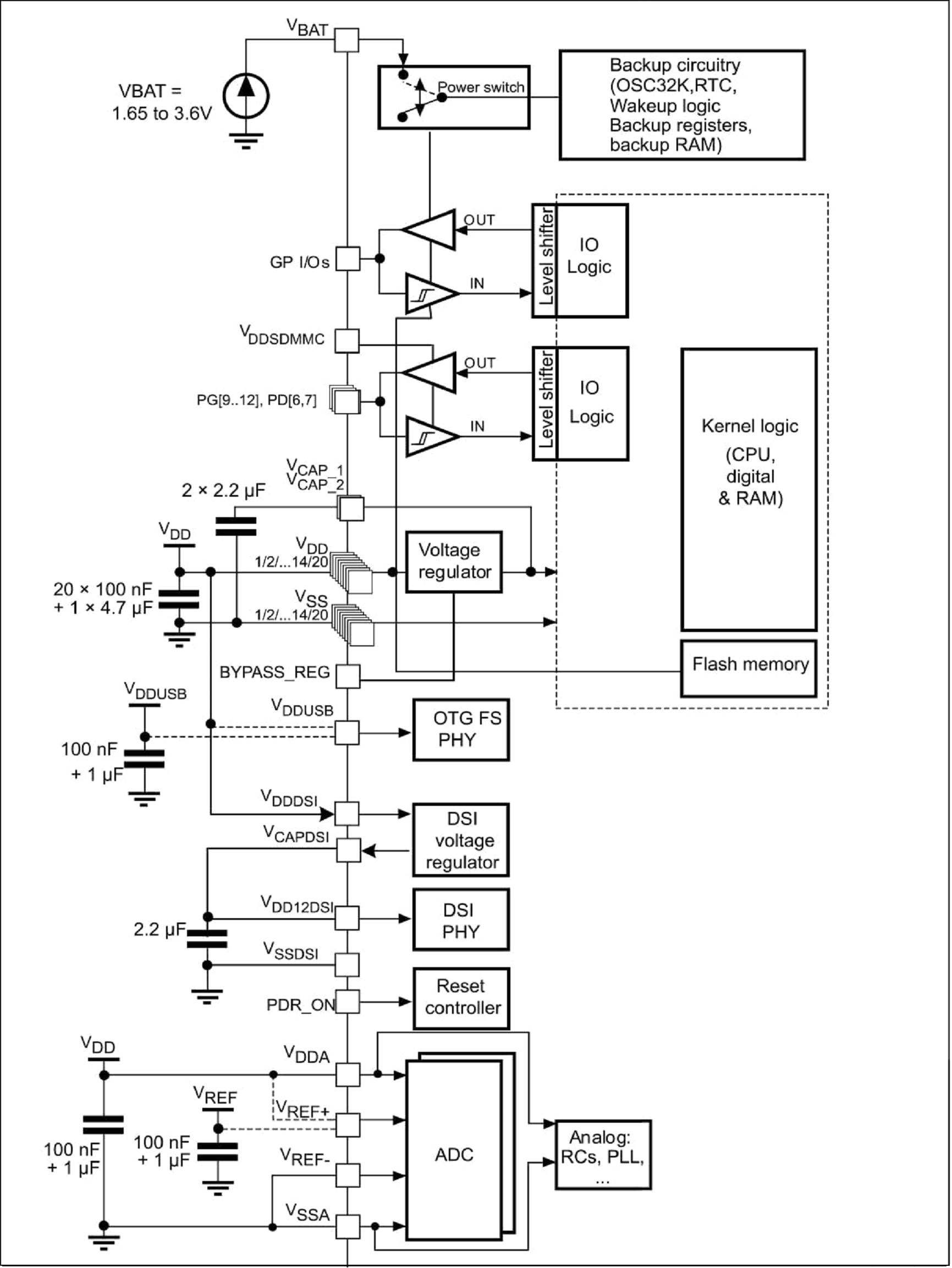


Рисунок 1.2 - Подключение источников питания к микроконтроллеру

В этом же документе приведены рекомендации по разводке дорожек высокоскоростных шин, к которым относятся шина памяти SDRAM, шина ULPI и шина SDMMC.

Документ [3] содержит подробную информацию обо всех аппаратных интерфейсах используемого микроконтроллера. Сведения из него будут использованы для разработки драйверов некоторых интерфейсов, необходимых для реализации требуемого функционала. К таким интерфейсам относятся:

* FMC - для работы с дисплеем и памятью SDRAM;
* SPI - для работы с аудиопроцессором и сенсорной панелью;
* SDMMC - для работы с картой памяти MicroSD;
* USB OTG - для работы с устройствами ввода, подключаемыми по шине USB;
* GPIO - для работы со встроенными клавишами.

Из [3] известно, что для возможности использования режима HS одного из USB OTG модулей необходим внешний PHY, подключаемый к микроконтроллеру при помощи шины ULPI. Сама фирма STMicroelectronics выпускает демо-платы с микроконтроллерами STM32, которые можно использовать не только как платформу для отладки кода и изучения работы с микросхемами, но и как образец проектирования аппаратного обеспечения на основе этих микроконтроллеров. Для реализации модуля USB OTG HS на этих платах используется микросхема USB3300 фирмы Microchip. На рисунке 1.3 показана структурная схема модуля USB OTG HS, построенная на основе этой микросхемы, взятая из [4].



Рисунок 1.3 - Структурная схема модуля USB OTG HS

Также в [4] приводится рекомендованная принципиальная схема использования данного PHY.

Для использования памяти SDRAM помимо правильной схемы её включения необходима информация о таймингах работы её шины. Эту информацию, вместе с рекомендованной последовательностью инициализации можно найти в [5]. На рисунке 1.4 показана структурная схема используемой памяти, взятая из [5].

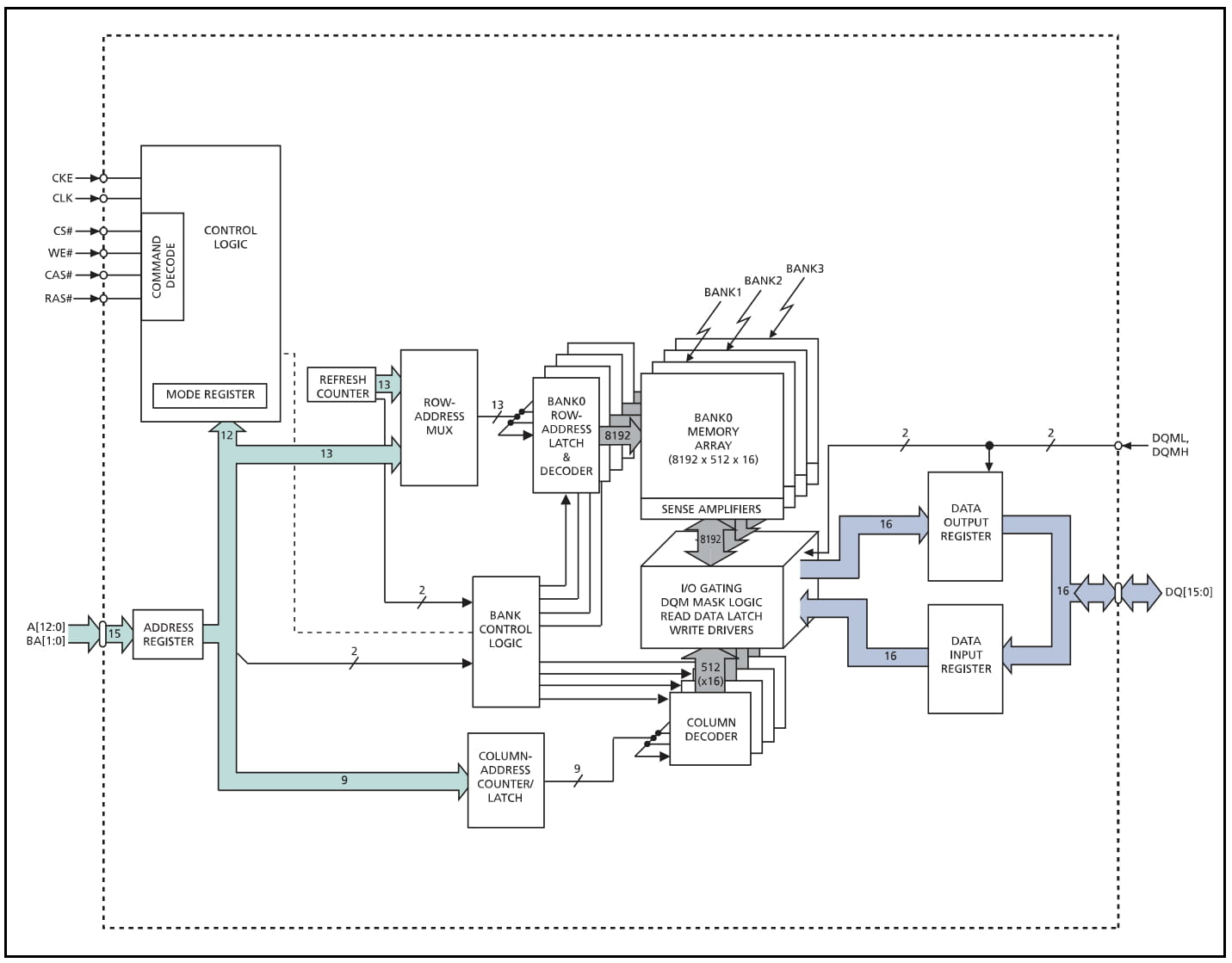


Рисунок 1.4 – Структурная схема памяти SDRAM

Применение аудиопроцессора для воспроизведения звука обуславливает необходимость в документации на используемую микросхему, аналогично всем вышеперечисленным микросхемам. Фирма VLSI Solution предоставляет необходимую документацию [6] на DSP-процессор VS1011E. В этом документе наряду с принципиальной схемой включения данного DSP-процессора дана подробная информация по его программированию. На рисунке 1.5 приведена структурная схема DSP-процессора, взятая из [6].

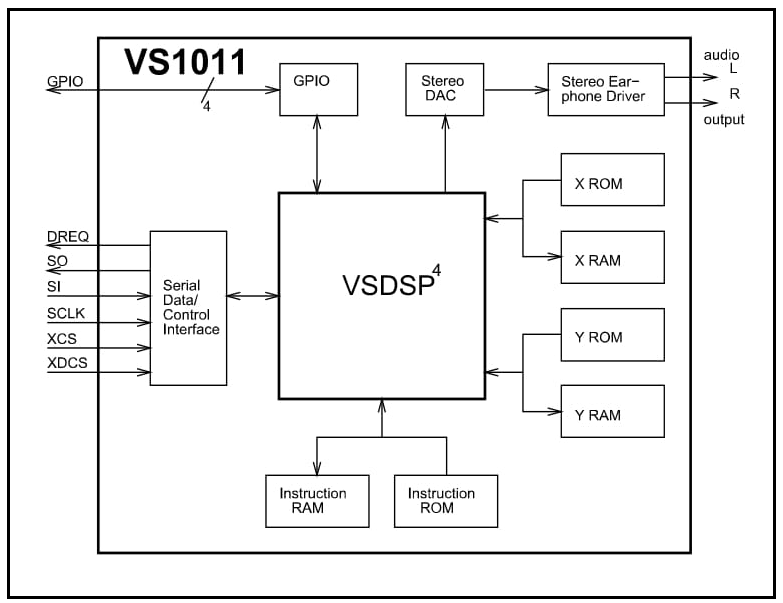


Рисунок 1.5 – Структурная схема DSP-процессора VS1011

1.3 Архитектура программного обеспечения

Разработке и архитектуре игр посвящено большое количество литературы, в основном зарубежной. В целом всю литературу по данной теме можно разделить на 2 типа - литература, подробно рассматривающая одну конкретную тему в контексте разработки игр (например, контроль столкновений в реальном времени, моделирование физики в реальном времени, рендеринг в реальном времени и прочее) и литература, описывающая в целом архитектуру игровых движков и игр, иногда на примере разработки конкретной игры. При этом информация в каждом разделе книг 2-го типа носит обзорный характер, ввиду большого количества тем подлежащих рассмотрению для разработки реальной игры, и большому объёму каждой из таких тем. Книга [7] как раз относится ко 2-му типу. В ней полностью описывается архитектура простого игрового движка (называемого авторами GameCode4) и простой 3D игры на нём (Teapot Wars), для демонстрации работы движка.

Книга [7] состоит из следующих основных глав. Информация из этих глав будут использоваться при разработке игрового движка в данном дипломном проекте.

*What’s in a Game?* В этой главе даётся информация о самой важной теме в разработке игрового движка - его высокоуровневой архитектуре. Авторы представляют архитектуру, похожую на паттерн MVC. На рисунке 1.6 приведена структурная схема движка, взятая из этой главы. Архитектура движка, разрабатываемого в данном дипломном проекте, будет примерно соответствовать данной архитектуре, по причине её простоты и гибкости.

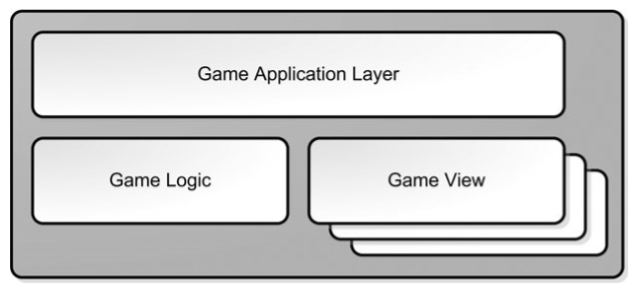


Рисунок 1.6 - Высокоуровневая архитектура движка из [7]

Так же в этой главе приведено краткое описание последующих глав и приведены структурные схемы каждого из показанных на рисунке 1.6 модулей.

*Building Your Game*. В этой главе даётся пример организации файлов проекта и настройки его сборки. Поскольку движок и игра, создаваемая в данном дипломном проекте, будет компилироваться под несколько программно-аппаратных платформ, правильная организация проектов и автоматизация их сборки необходима.

*Game Actors and Component Architecture*. В данной главе рассматриваются варианты построения классификации всех объектов игрового мира - акторов, и даётся обоснование почему для таких задач композиция подходит лучше, чем наследование. На рисунке 1.7 приведена структура актора из данной главы.

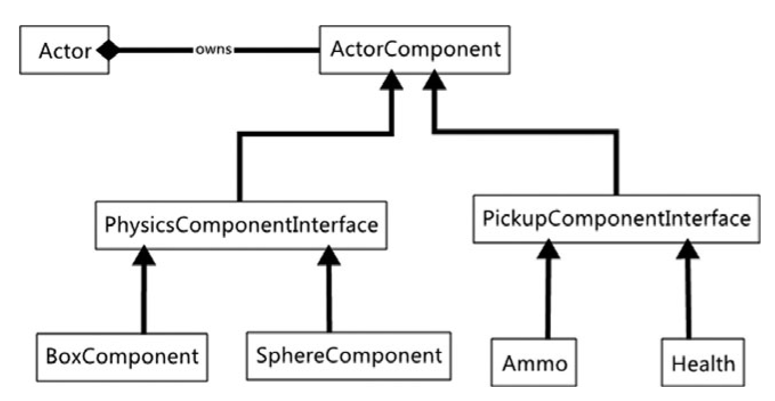


Рисунок 1.7 - Структура игрового актора из [7]

*Controlling the Main Loop.* В данной главе даётся описание одного из вариантов организации кода, ответственного за обновление состояния акторов в каждой итерации игрового цикла.

*Game Event Management.* В этой главе приводится пример устройства системы движка, ответственную за связь между Game Logic и Game View - системы сообщений.

*Collision and Simple Physics.* Здесь приводится пример встраивания в движок стороннего физического движка и его использования.

*A Game of Teapot Wars*! Эта глава показывает, как используя системы, разработанные в предыдущих главах, создать конкретную игру.

# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После исследования литературы по проектированию отдельных составных частей разрабатываемого проекта, а также составления списка требований к проекту, разбиваем разрабатываемую систему на составные части, с целю перейти к практической реализации.

В разрабатываемой аппаратной платформе можно выделить следующие составные части:

* микроконтроллер с обвязкой;
* память SDRAM;
* дисплей;
* сенсорная панель;
* аудиопроцессор с обвязкой
* модуль USB OTG FS;
* модуль USB OTG HS;
* слот для карт памяти MicroSD;
* встроенные клавиши;
* источник питания.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.2014 Э1.

Разрабатываемое программное обеспечение можно поделить на следующие крупные составные части:

* игровой движок;
* игровое приложение;
* библиотека аппаратных абстракций.

Рассмотрим составные части аппаратной платформы.

*Микроконтроллер с обвязкой* представляет собой ядро разрабатываемой платформы. Именно он ответственен за исполнение игрового кода, его хранение, а также управление всеми остальными составными частями платформы. В разрабатываемом устройстве будет применятся микроконтроллер STM32F765ZG фирмы STMicroelectronics. Как известно, игры — это очень требовательные к ресурсам приложения, а среди всех микроконтроллеров этой фирмы (и не только этой) серия F7 отличается наибольшей производительностью процессора (выше только у серии H7, находящейся в разработке), наибольшими объёмами встроенной памяти и наибольшим количеством периферийных модулей, при помощи которых к данному микроконтроллеру будут подключаться остальные составные части платформы. Подробное описание всех характеристик выбранного микроконтроллера приведено в [3].

Выбранный микроконтроллер обладает значительным объёмом встроенной оперативной памяти (по меркам микросхем данного класса) - 512Кб, однако для игровых приложений может потребоваться ещё большие объёмы памяти для хранения текстур, звуков, карт уровней и прочих данных. По этой причине есть необходимость в установке дополнительной микросхемы *памяти SDRAM*. Выбранный микроконтроллер оснащён контроллером памяти данного типа, который берёт на себя весь функционал по трансляции адресов с внутренних шин на протокол доступа к памяти SDRAM, а также периодическое обновление данных в микросхеме (операция, специфичная для динамической памяти). Подробное описание всех характеристик выбранной памяти приведено в [4].

*Дисплей* служит для отображения игрового процесса человеку-игроку (игрокам-ботам физический дисплей не требуется). Выбранный микроконтроллер оснащён несколькими контроллерами дисплеев, как пассивных, так и активных, однако наибольшее распространение для DIY-проектов (к которым относится разрабатываемая консоль) получили дисплейные модули на основе активных матриц, оснащённых контроллерами, аналогичными контроллеру SSD1289. Эти модули подразумевают подключение к управляющему микроконтроллеру по параллельной 16-ти битной шине i8080, а данная шина так же поддерживается выбранным микроконтроллером аппаратно. Описание работы с контроллером выбранного дисплея приведено в [5].

Как известно, самым важным в любой игре является возможность в неё играть, т.е. каким-либо образом влиять на то, что в ней происходит. Для этого любой игровой системе требуются различные устройства ввода, дизайну которых всегда уделялось огромное внимание разработчиков. Это могут быть как универсальные устройства (клавиатуры, мыши, джостики, геймпады), подходящие практически для любой игры, так и устройства, предназначенные для управления отдельным жанром игр или даже конкретным тайтлом (например, гитара для игры Guitar Hero). Помимо всего этого, современные мобильные игровые системы так же оснащены сенсорным экраном, который удобнее других устройства ввода в некоторых игровых жанрах. Поскольку разрабатываемая консоль близка по формату к мобильным игровым системам, она так же должна быть оснащена *сенсорной панелью,* и выбранный дисплейный модуль как раз ей оснащён. Контроллер сенсорной панели подключается к микроконтроллеру по шине SPI.

Поскольку звук в играх порой не менее важен, чем видео, консоль должна обладать возможностью его воспроизводить. На сегодняшний день основным способом воспроизведения звука для микроконтроллеров является подключение к ним аудио кодеков по шине I2S, однако такие кодеки представляют собой цифро-аналоговый преобразователь и могут воспроизводить только несжатый звук. Ещё одним способом воспроизведения звука является использование специализированных DSP-процессоров, обладающих возможностью проигрывать как несжатый, так и сжатый звук в формате MP3. Одним из таких *аудиопроцессоров* является VS1011E фирмы VLSI Solution. Это распространённая микросхема, с лицензией на воспроизведение MP3 (чем обуславливается её сравнительно высокая цена). Данная микросхема и используется в разрабатываемой консоли.

Как говорилось выше, для игры крайне важны устройства ввода. Поскольку сенсорная панель не удобна для значительного количества игр, необходимо иметь возможность подключать к консоли и другие устройства ввода. В современных игровых (и не только игровых) системах почти единственным интерфейсом для подключения всех устройств ввода (и не только ввода) является USB. Поскольку данная шина распространена крайне широко, очень большое количество микроконтроллеров оснащено аппаратными модулями для работы с ней. Не исключение и выбранный микроконтроллер, который оснащён сразу 2-мя модулями *USB OTG 2.0*, что означает возможность работы не только в режиме устройства, но и в режиме хоста, что и требуется для работы с внешними устройствами ввода. Один из этих контроллеров поддерживает работу только в режиме FS, а другой может работать как в режиме FS, так и в режиме HS. Для работы в режиме HS второму контроллеру необходим внешний PHY, подключаемый по шине ULPI. В разрабатываемой консоли используются оба контроллера USB OTG, в качестве PHY для второго используется микросхема USB3300 фирмы Microchip.

Для того, чтобы создать игру необходим не только код, но и ресурсы, к которым относятся текстуры, звуки, карты уровней, скрипты, таблицы локализированных строк, и прочее. Все эти ресурсы необходимо где-то хранить. Для этого в разрабатываемой консоли имеется слот для широко распространённых *карт памяти MicroSD.*

*Встроенные клавиши* могут пригодится для выполнения действий по управлению самой консолью, например - регулировка уровня громкости, блокировка/разблокировка экрана и подобных.

Хороший *источник питания* необходим для правильной и стабильной работы всех аппаратных частей консоли.

Теперь рассмотрим составные части разрабатываемого ПО консоли.

*Игровой движок* необходим по той причине, что у всех игр довольно большое количество одинакового функционала. Чтобы не создавать его заново каждый раз, имеет смысл вынести его в отдельный модуль. Движок, разрабатываемы в рамках данного проекта будет состоять из следующих блоков:

* Application Layer;
* Game View;
* Game Logic.

*Application Layer* содержит в себе весь платформо-зависимый код, необходимый для работы игры на выбранной платформе. В его задачи входит:

* инициализация и завершение игры;
* взаимодействие с очередью сообщений платформы;
* создание окна игры, контекста для рендеринга;
* работа с сетевыми сервисами платформы.

При переносе игры на другую платформу этот код должен быть полностью переписан. Однако разрабатываемый движок в качестве основы для построения этого блока будет использовать широко известную библиотеку SDL2, которая используется во многих играх, например Amnesia: The dark descent, Dying Light, VVVVVV, linux-версия half-life и множество других. Использование данной библиотеки означает отсутствие необходимости переписывать Application Layer для тех платформ, для которых доступна SDL2, а заодно открывает дорогу для портирования игр на разрабатываемую в рамках данного проекта аппаратную платформу, о чём пойдёт речь ниже.

*Game View* отвечает за всё, что касается взаимодействия игры с игроком. Можно выделить 3 типа игроков:

* человек;
* искусственный интеллект;
* удалённый игрок.

Соответственно может быть 3 типа Game View - Game Human View, Game AI View, и Networked Game View. Игровое меню является подвидом Game Human View - Game Menu View. В игре, разрабатываемой в рамках данного проекта для демонстрации работы всего аппаратного и программного обеспечения потребуется только Game Human View и Game Menu View. В задачи этих модулей будет входить:

* для Game Human View - отображение игрового мира (графика и звук) и отправка команд управления от игрока в Game Logic;
* для Game Menu View - отправка в Game Logic команд, соответствующим виртуальным элементам управления.

*Game Logic* представляет собой модуль, где непосредственно реализованы правила игры, игровой мир, игровые механики и прочее. В нём находится контейнер с игровыми акторами, физический мир, если игре требуется физика, и менеджер процессов для постоянного обновления всех акторов. Большая часть кода конкретной игры принадлежит данному модулю.

Взаимодействие Game Logic и Game View будет происходить через *систему событий*. Она служит для полной развязки этих двух модулей. При использовании такого подхода, всё что требуется делать Game Logic это посылать в систему событий события изменения состояния акторов, и принимать из системы команды управления. Таким образом, для Game Logic нет разницы с каким именно Game View он взаимодействует - Human, AI или Networked. Это может быть даже список из нескольких Game View.

*Игровое приложение* будет использовать сервисы, предоставляемые игровым движком для реализации меню и правил конкретной игры.

Одной из особенностей данного проекта будет являться то, что код игры и движка будет способен без изменений компилироваться и работать как на ПК, так и на разрабатываемой консоли, при условии, что ресурсов консоли будет достаточно. Достигаться это будет за счёт создания специальной *библиотеки аппаратных абстракций*. Как уже говорилось выше, в качестве основы для Application Layer движка будет использована библиотека SDL2. Из всех используемых движком сторонних библиотек только она является платформо-зависимой, и для компиляции и запуска игры будет достаточно реализовать несколько функций из этой библиотеки для разрабатываемой консоли. Помимо этого, в библиотеке аппаратной абстракции будет находится:

* операционная система реального времени, для возможности параллельно исполнять потоки-контроллеры некоторых аппаратных устройств;
* Реализация стандартных функций языка Си для доступа к файловой системе, таких как fopen(), fread() и других.

# ЗАКЛЮЧЕНИЕ

Во время работы над преддипломной практикой была изучена предметная область, произведён поиск и анализ литературы, необходимой для разработки проекта. В частности, были получены знания по проектированию электронных устройств на основе микроконтроллеров STM32, изучено применение микросхем памяти SDRAM и специализированных микросхем, таких как аудиопроцессор и контроллер PHY для USB OTG HS. Так же подробно изучены вопросы, касающиеся разводки печатных плат современных высокоскоростных электронных устройств. Была изучена работа с современным САПР для разработки аппаратного обеспечения, работа со средствами разработки и отладки программного обеспечения встраиваемых систем на основе микроконтроллеров STM32. На демо-плате фирмы STMicroelectronics с микроконтроллером STM32 были исследованы примеры драйверов от производителя и написаны прототипы драйверов, необходимых для работы проектируемого устройства. Был изучен пример архитектуры игрового движка, и работа со сторонними библиотеками, необходимыми для создания движка и игры на нём.

Также в ходе преддипломной практики была разработана электрическая структурная схема реализуемого устройства. Благодаря системному подходу к проектированию возможно дальнейшее его улучшение и расширение функциональности в целом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] An FPGA sprite graphics accelerator with a 180MHz STM32F429 controller and 640 x 360 LCD [Электронный ресурс]. – Режим доступа: http://andybrown.me.uk/2014/06/01/ase/

[2] Getting started with STM32F7 Series MCU hardware development [Электронный ресурс]: Datasheet/STMicroelectronics. – Режим доступа: http://www.st.com/content/ccc/resource/technical/document/application\_note/b0/5c/ba/b2/cb/10/41/dd/DM00164549.pdf/files/DM00164549.pdf/jcr:content/translations/en.DM00164549.pdf

[3] STM32F76xxx and STM32F77xxx advanced Arm-based 32-bit MCUs [Электронный ресурс]: Datasheet/STMicroelectronics. – Режим доступа: http://www.st.com/content/ccc/resource/technical/document/reference\_manual/group0/96/8b/0d/ec/16/22/43/71/DM00224583/files/DM00224583.pdf/jcr:content/translations/en.DM00224583.pdf

[4] USB3300 Hi-Speed USB Host, Device or OTG PHY with ULPI Low Pin Interface [Электронный ресурс]: Datasheet/Microchip. – Режим доступа: http://ww1.microchip.com/downloads/en/DeviceDoc/00001783C.pdf

[5] SDR SDRAM [Электронный ресурс]: Datasheet/Micron. – Режим доступа: https://www.micron.com/~/media/Documents/Products/Data%20Sheet/DRAM/256Mb\_sdr.pdf

[6] VS1011e - MP3 AUDIO DECODER [Электронный ресурс]: Datasheet/VLSISolution. – Режим доступа: http://www.vlsi.fi/uploads/media/vs1011\_01.pdf

[7] Game Coding Complete, Fourth Edition/M. McShaffry, D. Graham – Boston, 2012

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Вводный плакат

# ПРИЛОЖЕНИЕ Б

*(обязательное)*

Схема структурная электрическая