Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Структуры и базы данных»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
Магистр технических наук, ассистент
_____ А.Д. Сыс
___._____.2021

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
«БАЗА ДАННЫХ ДЛЯ ПОДДЕРЖКИ РАБОТЫ АВТОМАГАЗИНА»

БГУИР КП 1-39 03 02 004 ПЗ

Выполнил студент группы 913801
Гончар Виталий Витальевич
_____
(подпись студента)
Курсовой проект представлен на проверку ___._____.2021
_____
(подпись студента)

Минск 2021

РЕФЕРАТ

БГУИР КП 1-39 03 02 004 ПЗ

Гончар, В.В. База данных для поддержки работы автомагазина: пояснительная записка к курсовому проекту / В.В. Гончар. – Минск: БГУИР, 2021. – 52 с.

Пояснительная записка 52 с., 15 рисунков, 16 источников, 4 приложения.

АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ЕЁ ФОРМАЛИЗАЦИЯ ДЛЯ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ДЛЯ ОСНОВНОГО ВИДА ДЕЯТЕЛЬНОСТИ РАССМАТРИВАЕМОЙ ОБЛАСТИ, ПРИМЕНЕНИЕ РАЗРАБОТАННОЙ БАЗЫ ДАННЫХ

Цель проектирования: разработка веб-приложения, проектирование эффективной и безопасной базы данных для поддержания работы автомагазина.

Методология проведения работы: в процессе решения поставленных задач использованы теория проектирования базы данных, методы проектирования веб-приложений.

Результаты работы: изучены способы хранения информации в базе данных, проведен анализ реляционной модели данных, разработано веб-приложение для работы автомагазина.

Область применения результатов: разработанная база данных и приложение в будущем, возможно применить для создания полноценного коммерческого приложения, которое может быть использовано реальным автомагазином.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

БД – База данных
VIN – Идентификационный номер транспортного средства
СУБД – Система управления базами данных
PK – *Primary Key*
API – *Application Programming Interface*
ER – *Entity-Relationship*
CASE – Набор инструментов для проектирования программного обеспечения
ID – *Identifier*
НФ – Нормальная форма
DOM – *Document Object Model*
HTTP – Протокол передачи гипертекста
JS – *JavaScript*
VS – *Visual Studio*
GIT – Распределённая система управления версиями
SSL – Криптографический протокол

ВВЕДЕНИЕ

На сегодняшний день жизнь каждого человека сложно представить без автомобиля. Конечно, можно поспорить, сказав, что в крупных городах (где население близко к одному миллиону) существует развитая система общественного транспорта, такого как метро, автобусы, трамваи, и с этим сложно не согласиться. Однако, личный автомобиль предоставляет своему водителю такие преимущества, которые ни один общественный транспорт предоставить не может. В такие преимущества входят:

– комфорт поездки;

– свобода передвижения.

В связи с этим порождается высокий спрос на покупку личных автомобилей. Данная сфера остаётся актуальной и по сей день и поэтому важно, чтобы покупка автомобиля для будущего автовладельца проходила быстро и приятно. Для этих целей должна быть разработана как база данных для учета автомобиля и поддержки работы автомагазина в целом, так и веб-приложение для работы с базой данных.

При выполнении курсового проекта должны быть учтены особенности, с которыми должно производиться проектирование базы данных, а именно: проектируемая база данных должна отвечать требованиям надёжности, минимальной избыточности, целостности данных, содержать не менее десяти сущностей, а ее схема должна быть приведена к третьей нормальной форме. База данных должна поддерживать основные современные средства для работы и администрирования. Необходимо реализовать следующие объекты базы данных в количестве не менее трех каждый: индекс, триггер, хранимая процедура. Также требуется реализовать систему разграничения прав доступа к данным минимум для двух ролей пользователей.

Таким образом, целью данного курсового проекта является проектирование базы данных для поддержки работы автомагазина, предметной областью которой является реализация автомобилей, а также разработка программного средства.

1 ОБЩЕТЕХНИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРИБОРА

1.1 Анализ исходных данных

Проектируемым прибором является мобильная *IP*-видеокамера. В качестве самой видеокамеры будет использоваться *OV9655 CameraBoard* – простая цифровая камера, основанная на *OV9655*, которая может как делать фотографии, так и снимать видео, но только при использовании совместно с отладочной платой. В основе такой платы будет микроконтроллер на базе ядра *ARM Cortex-M4*. Прибор должен уметь считывать, сжимать, передавать поток данных по *Ethernet*-каналу на веб-сервер. Основным источником питания должно быть питание от аккумулятора напряжением 3.6 В. Потребляемый ток должен быть не более 40 мА. Также должна быть предусмотрена возможность зарядки аккумулятора от дополнительного источника питания через разъём *RJ*-45 (технология *PoE*).

1.2 Теоретические сведения и принципы функционирования отдельных узлов прибора

1.2.1 Понятие изображения

Изображение – это визуальное представление чего-либо. В информационных технологиях этот термин имеет несколько значений. В основном, изображение – это картинка, созданная или скопированная и сохраненная в электронном виде. Изображение может быть описано с точки зрения векторной графики или растровой графики. Изображение, хранящееся в растровом виде, иногда называют растровым.

Векторная графика также известна как масштабируемая векторная графика (*SVG*). Эта графика состоит из закрепленных точек, соединенных линиями и кривыми. Поскольку эта графика не основана на пикселях, она известна как независимая от разрешения, что делает ее бесконечно масштабируемой. Линии в векторной графике четкие, без потери качества или детализации, независимо от их размера. Эта графика также не зависит от устройства, что означает, что ее качество не зависит от количества точек, доступных на принтере, или количества пикселей на экране. Поскольку они состоят из линий и опорных точек, размер файлов относительно невелик.

Растровые изображения состоят из пикселей или крошечных точек, которые используют цвет и тон для создания изображения. Пиксели выглядят как маленькие квадраты на миллиметровой бумаге, когда изображение увеличивается или увеличивается. Эти изображения создаются цифровыми камерами, путем сканирования изображений в компьютер или с помощью растрового программного обеспечения. Каждое изображение может

содержать только фиксированное количество пикселей; количество пикселей определяет качество изображения. Это известно как разрешение. Чем больше пикселей, тем лучше качество при том же или большем размере оригинала, но это также увеличивает размер файла и количество места, которое требуется для хранения файла. Чем меньше количество пикселей, тем ниже разрешение. Разрешение ограничивает размер изображения, которое может быть увеличено без возможности видеть пиксели. Однако изображение с высоким разрешением, напечатанное в маленьком размере, приведет к тому, что пиксели «втиснуты» друг в друга.

Общие форматы файлов изображений в Интернете включают:

1 *JPEG* – это файл графического изображения, созданный в соответствии со стандартом Объединенной группы экспертов по фотографии, группы экспертов *ISO/IEC*, которая разрабатывает и поддерживает стандарты для набора алгоритмов сжатия файлов компьютерных изображений. Файлы JPEG обычно имеют расширение *.jpg*.

2 *GIF* означает формат обмена графикой. *GIF* использует растровый тип данных *2D* и закодирован в двоичном формате. Файлы *GIF* обычно имеют расширение *.gif*

3 *GIF89a* – это анимированное изображение в формате *GIF*, отформатированное в соответствии с версией *GIF 89a*. Одним из главных преимуществ формата является возможность создания анимированного изображения, которое можно воспроизвести после передачи на страницу просмотра, которая движется – например, вращающаяся иконка или баннер с машущей рукой или буквы, которые волшебным образом увеличиваются. *GIF89a* также может быть указан для чересстрочного представления *GIF*.

4 *PNG – Portable Network Graphics*) – это формат файлов для сжатия изображений, который был разработан для обеспечения ряда улучшений по сравнению с форматом *GIF*. Как и *GIF*, файл *PNG* сжимается без потерь (это означает, что вся информация об изображении восстанавливается при распаковке файла во время просмотра). Файлы обычно имеют расширение *.png*.

5 *SVG* – это масштабируемая векторная графика, описание изображения как приложения *XML*. Любая программа, например браузер, распознающая *XML*, может отобразить изображение, используя информацию, предоставленную в формате *SVG*. Масштабируемость означает, что файл можно просматривать на дисплее компьютера любого размера и разрешения, будь то маленький экран смартфона или большой широкоформатный дисплей ПК. Файлы обычно имеют расширение *.svg*

6 *TIFF* (*Tag Image File Format*) – это распространенный формат для обмена растровыми графическими (растровыми) изображениями между прикладными программами, включая те, которые используются для изображений сканера. Файл *TIFF* можно идентифицировать как файл с суффиксом имени файла *.tiff* или «*.tif*».

7 *BMP* – формат несжатого растрового изображения, в заголовке которого записана информация об изображении – размер файла, ширина и высота рисунка, глубина пикселей, количество цветов. После заголовка может следовать палитра. Далее идет непосредственно набор данных о пикселях, который идентифицирует положение каждого пикселя и его цвет.

1.2.2 Принципы формирования и представления изображений

Как было отмечено ранее, векторный подход представляет изображение как совокупность простых элементов: прямых линий, дуг, окружностей, эллипсов, прямоугольников и других, которые называются графическими примитивами. Кодирование векторного изображения предполагает однозначное определение (задание) положения, формы, а также цветовых характеристик всех графических примитивов, составляющих рисунок.

Что касается растровой графики, то основным элементом растрового изображения является пиксель. Этот термин имеет несколько значений: отдельный элемент растрового изображения, отдельная точка на экране монитора, отдельная точка на изображении, напечатанном принтером. Поэтому на практике эти понятия часто обозначают так:

– пиксель – отдельный элемент растрового изображения;

– видеопиксель – элемент изображения на экране монитора;

– точка – отдельная точка, создаваемая принтером.

Растровое изображение состоит из пикселей. Каждый пиксель растрового изображения характеризуется координатами *x*, *y* и яркостью (только для черно-белых изображений). Поскольку пиксели имеют дискретный характер, то их координаты – это дискретные величины – целые или рациональные числа. В цветном изображении каждый пиксель характеризуется координатами x и y, и тремя яркостями: яркостью красного, яркостью синего и яркостью зеленого цветов (*VR, VB, VG*). Комбинируя данные три цвета, можно получить большое количество различных оттенков.

Если для представления каждого пикселя в чернобелом рисунке достаточно одного бита (бинарная форма записи), то для работы с цветом или полутоновым изображения этого явно недостаточно. Однако подход при кодировании цветных изображений остается неизменным. Любой рисунок разбивается на пиксели, т. е. небольшие части, каждая из которых имеет свой цвет.

Объем информации, описывающий цвет пикселя, определяет глубину цвета. Чем больше информации определяет цвет каждой точки в рисунке, тем больше вариантов цвета существует. Понятно, что для рисунков в естественном цвете требуется больший объем памяти. Чтобы представить более шестнадцати миллионов цветов, информация о каждой точке рисунка должна занимать четыре байта, что в тридцать два раза больше, чем для монохромного рисунка.

Важной характеристикой изображения является разрешение, которое измеряется в количестве точек на дюйм.

Разбив рисунок на пиксели, описав цвет каждого пикселя и задав разрешение, можно закодировать любое изображение. Таким образом, программы могут воспроизводить изображения.

1.2.3 Классификация цветовых моделей

Цветовая модель – термин, обозначающий абстрактную модель описания представления цветов в виде кортежей чисел, обычно из трёх или четырёх значений, называемых цветовыми компонентами или цветовыми координатами. Вместе с методом интерпретации этих данных множество цветов цветовой модели определяет цветовое пространство.

Все цветовые модели можно разделить на три группы, в зависимости от аппаратной платформы:

– аппаратно- зависимые модели, используемые в технических средствах ввода и вывода графической информации (*RGB, CMYK*);

– аппаратно-независимые модели, описывающие цвет в абстрактных колориметрических терминах (*XYZ*);

– интуитивные модели, построенные на основе субъективного восприятия цвета человеком (*HSB*).

Также в зависимости от объекта, на котором воспроизводится изображение, цветовые модели можно поделить на аддитивные и субтрактивные.

Модель *RGB* используется при работе с проектами на основе цифрового экрана, например, при просмотре на экране компьютера или дисплее телефона. В цветовой модели *RGB* каждому из основных цветов, красному, зеленому и синему, присваивается значение от 0 до 255, где 0 – темный, а 255 – яркий. Указав три значения для красного, зеленого и синего люминофоров, можно указать точный цвет, который получится в итоге.

Цветовая модель *RGB* представляет собой аддитивную цветовую систему, что означает, что цвета при смешивании становятся светлее. Когда каждый компонент света смешивается, комбинация становится новым цветом. Красный, зеленый и синий – это три аддитивных основных цвета.

Экраны телевизоров и компьютерные мониторы создают цвет, включая красный, зеленый и синий основные цвета в каждом пикселе.

Поскольку цветовая модель *RGB* способна воспроизводить только определенный диапазон (или гамму) цветов, некоторые цвета не могут быть точно воспроизведены монитором компьютера. Количество цветов, видимых на мониторе, еще больше уменьшается из-за ограничений видеооборудования компьютера, которое может отображать любое изображение, от черно-белого до 16,7 миллионов цветов.

Цветовая модель *CMYK* описывает цвета на основе процентного содержания голубого, пурпурного, желтого и черного. Многие компьютерные принтеры и традиционные «четырехцветные» печатные машины используют модель *CMYK*. В модели *CMYK*, смешивая голубые, пурпурные, желтые и черные чернила или краски, можно создать практически любой желаемый цвет.

CMYK – это субтрактивная цветовая модель, означающая, что при смешивании цвета становятся темнее. Каждая из смешанных красок или чернил поглощает различные компоненты света. Если смешать правильную комбинацию красок, все компоненты света поглощаются, и в результате получается почти черный цвет. Однако в реальном мире чернила, которые используют принтеры, не идеальны поэтому в результате смешивания цветов получается грязно-коричневый цвет, потому что основные цвета перекрываются и не полностью вычитают свет при смешивании. Поэтому букву *K* в аббревиатуре обозначает слово *Key* или чёрный цвет.

Модель *HSL* очень похожа на цветовую модель *RGB*. На самом деле, когда они выражены математически, они идентичны. Разница заключается в том, как цвета выражаются численно.

Оттенок определяет, какой это основной цвет. Красный, зеленый, синий, желтый, оранжевый и т. д. – это разные оттенки. Насыщенность и яркость больше говорят о вариациях этих основных цветов. Насыщенность – это яркость (или «чистота») цвета, т. е. то, сколько дополнительных цветов смешано. Наконец, легкость относится к «белизне» цвета. Его также можно назвать «яркостью», «значением» или «интенсивностью».

Другими моделями, связанными с моделью *HSL*, являются модели *HSB* (оттенок, насыщенность, яркость) и *HSI* (оттенок, насыщенность, интенсивность). Все эти термины похожи, но не взаимозаменяемы.

1.2.4 Разновидности цветовой модели RGB

Существует много разновидностей цветовой модели *RGB*, все они отличаются количеством бит, отведённых под каждый цвет. Например, в *9-bit RGB* на каждый цвет отведено по 3 бита, значит цветовая палитра включает в себя 512 цветов.

Наиболее используемым форматом является *RGB24*, где каждому цвету соответствует один байт. Таким образом, *RGB24* может передавать почти 17 миллионов цветов. Как можно заметить, каждый такой формат отличается определённым параметром, который напрямую влияет на количество передаваемых цветов и который называется глубиной цвета (Рисунок 1).

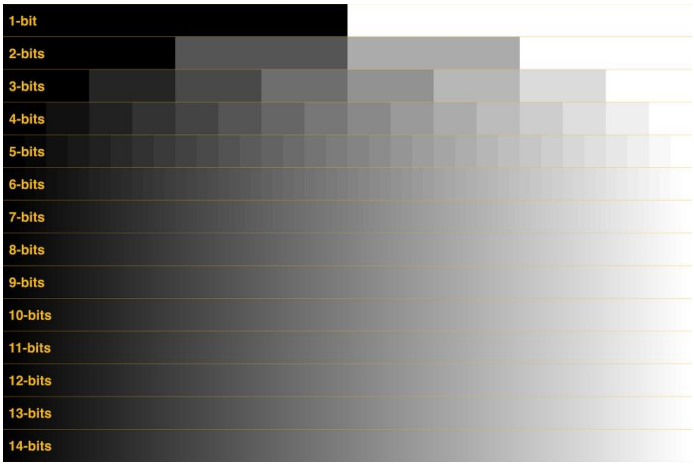

Рисунок 1 – Глубина цвета на примере черного и белого цвета

Также существует *RGB32*, который отличается от *RGB24* тем, что к нему добавляется ещё 1 байт под так называемый альфа-канал, который отвечает за прозрачность пикселя.

Битовая глубина (или глубина цвета) определяет количество возможных значений цвета, но также существует цветовое пространство, которое определяет максимальное значение или диапазон цвета. Что касается цветового пространства, то кроме стандартного *RGB* есть его модификации, такие как *Rec.709, AdobeRGB, DCI-P3, Rec.2020*. Как можно заметить на Рисунок 2, Rec.2020, например, имеет большой охват зелёного цвета, соответственно, он может передать больше цветов.

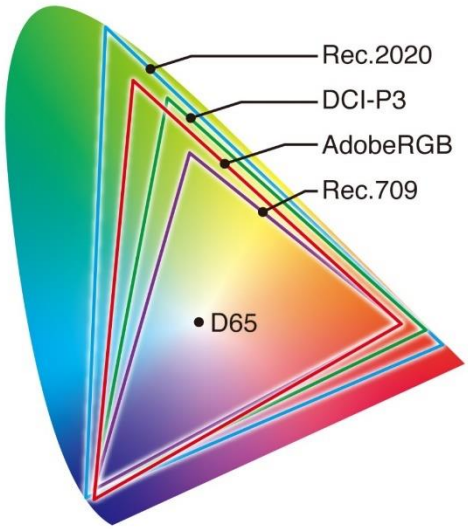

Рисунок 2 – Разница цветовых диапазонов

SRGB по-прежнему является стандартом для компьютерных изображений, большинства потребительских и средних фотокамер и

домашних принтеров. Для профессиональной печати и допечатной подготовки часто используется *Adobe RGB* с расширенным цветовым охватом, который можно воспроизвести с помощью профессиональной печати *CMYK*.

1.2.5 Понятие видеопотока

Потоковое видео – это технология, которая позволяет пользователю просматривать онлайн-видеоконтент через Интернет без необходимости предварительной загрузки медиафайлов. Потоковое видео относится конкретно к онлайн-видеоконтенту, такому как фильмы, телешоу, прямые трансляции событий и онлайн-видео, видеонаблюдение. Но потоковая передача также может включать аудиоконтент, такой как музыка, подкасты и многое другое

Поскольку пользователю не нужно загружать медиафайлы для использования контента, потоковая передача помогает экономить ресурсы хранения, время, затрачиваемое на загрузку или буферизацию видео, и обеспечивает удобство просмотра для пользователей. Потоковая передача видео (аудио) не была бы возможно без специальных протоколов.

Протоколы потоковой передачи видео – это специальные стандартизированные правила и методы, которые разбивают видеофайлы на более мелкие части (пакеты), чтобы их можно было доставить конечному пользователю для повторной сборки и просмотра

Файлы должны быть сжаты для транспортировки, этот процесс достигается с помощью «кодека», такого как наиболее распространенный H.264. Файлы также должны быть сохранены в «контейнерном формате», таком как *.mp4* или *.avi*, прежде чем их можно будет передать. Источником видеофайла может быть непосредственно камера вещателя в случае прямой трансляции или статические файлы в случае видео по запросу (*VoD*).

На сегодняшний день существует несколько протоколов потоковой передачи видео. Некоторые из них являются устаревшими стандартами, которые все еще используются в некоторых случаях, в то время как другие быстро развиваются, особенно благодаря среде с открытым исходным кодом.

На сегодняшний день *HLS* является наиболее часто используемым протоколом для потоковой передачи Этот протокол совместим с широким спектром устройств, от настольных браузеров, смарт-телевизоров, телевизионных приставок, мобильных устройств *Android* и *iOS* и до видеоплееров *HTML5*.

HLS также поддерживает потоковую передачу с адаптивным битрейтом. Единственным серьезным недостатком, связанным с протоколом *HLS,* могут быть связанные с ним большие задержки. Под задержкой понимается время, необходимое информации для перемещения от источника к месту назначения и обратно, когда через Интернет передаются большие объемы данных.

MPEG-DASH – один из последних протоколов потоковой передачи, разработанный Экспертной группой по движущимся изображениям (*MPEG*) в качестве альтернативы стандарту *HLS*. Это стандарт с открытым исходным кодом, который можно настроить для любого аудио- или видеокодека.

Как и *HLS, MPEG-DASH* поддерживает потоковую передачу с адаптивным битрейтом, позволяя зрителям получать видео самого высокого качества, которое может обрабатывать их сеть.

WebRTC – это проект с открытым исходным кодом, целью которого является доставка потоковой передачи с задержкой в реальном времени. Первоначально разработанный для приложений на основе исключительно чата и использования *VoIP*, он стал известен для использования в приложениях для видеочата и конференций после того, как был куплен *Google*.

SRT – это еще один протокол с открытым исходным кодом, разработанный поставщиком технологий потоковой передачи *Haivision*. Это предпочтительный протокол для членов *SRT Alliance*, группы компаний, в которую входят поставщики технологий и телекоммуникаций. Основными преимуществами, которыми славится *SRT*, являются безопасность, надежность, совместимость и потоковая передача с малой задержкой.

SRT может передавать потоковое видео высокого качества, даже если сетевые условия нестабильны. Он также не зависит от одного кодека, что позволяет использовать его с любыми аудио- и видеокодеками.

1.2.6 Назначение видеокамеры

Основное назначение цифровой видеокамеры – преобразование оптического изображения в электрический сигнал. Чтобы это было возможно, камере необходимо иметь в себе как минимум три компонента: объектив, матрицу, плату обработки.

В то время как свет отражается от объектов, он также может проходить сквозь объекты, но, когда это происходит, он может менять направление. Объектив камеры улавливает все световые лучи, отражающиеся вокруг, и использует стекло, чтобы перенаправить их в одну точку, создавая четкое изображение

Когда все эти световые лучи встречаются на сенсоре цифровой камеры или на куске пленки, они создают четкое изображение. Если свет не встречается в нужной точке, изображение будет выглядеть размытым или не в фокусе. Система фокусировки объектива перемещает линзу ближе или дальше от сенсора или пленки, позволяя фотографу настроить объектив так, чтобы объект был резким.

Объектив проецирует свет, который отражается от объектов, на матрицу. Качество детализации объекта зависит от расстояния до него. Чем большая степень детализации требуется, тем большую часть матрицы должно занимать интересующее нас изображение.

Регулируется это подбором фокусного расстояния объектива или углом его обзора. Эти два параметра находятся в непосредственной взаимосвязи – чем больше фокусное расстояние, тем меньше угол обзора.

Матрицей называется пластина из полупроводникового материала, которая состоит из множества светочувствительных участков (пикселей). Чем больше пикселей, тем лучше разрешение камеры и, соответственно, детализация. Однако при неизменной площади пластины увеличение количества пикселей приводит к потере светочувствительности камеры.

Плата обработки сигнала формирует выходной сигнал и в зависимости от вида этого сигнала (аналоговый и цифровой) существуют аналоговые и цифровые камеры.

Аналоговые видеокамеры умеют только передавать сигнал, их функциональные возможности позволяют этот сигнал обрабатывать с целью улучшения и устранения возможных помех, искажений.

Цифровые видеокамеры также называются *IP*-видеокамерами или сетевыми видеокамерами. В отличие от аналоговых, сетевые видеокамеры могут работать в локальных сетях как отдельные устройства. Также сетевые видеокамеры способны как записывать видео на встроенные карты памяти, так и передавать видеопоток на сервер.

1.2.7 Ethernet в контексте модели *OSI*

Модель *OSI* – эталонная модель, которая описывает архитектуру и принципы работы компьютерных сетей. Всего данная модель имеет 7 уровней. Так как речь идёт про *Ethernet*, то будут удостоены внимания два первых уровня в данной сетевой модели – физический и канальный.

Физический уровень в *OSI* отвечает за передачу непосредственно электрических сигналов (битов) по какой-либо среде и также представляет собой физическую среду, по которой передаются сигналы между узлами. Формат таких сигналов варьируется в зависимости от среды передачи. В случае *Ethernet* биты передаются в виде электрических импульсов. В случае *Wi-Fi* биты передаются в виде радиоволн. В случае оптоволокна биты передаются в виде световых импульсов.

Второй уровень называет канальным, и он решает проблему адресации при передаче информации. Канальный уровень получает биты и превращает их в кадры – *frames*. Канальный уровень формирует кадры с адресом отправителя и получателя, после чего отправляет их по сети. Кадр – это единица данных, которыми обмениваются компьютеры в сети *Ethernet*. Кадр имеет фиксированный формат и наряду с полем данных содержит различную служебную информацию, например адрес получателя и адрес отправителя. После того как адаптер отправителя поместил кадр в сеть, его начинают принимать все сетевые адаптеры. Каждый адаптер проводит анализ кадра, и если адрес совпадает с их собственным адресом устройства (*MAC*–адрес), кадр

помещается во внутренний буфер сетевого адаптера, если же не совпадает, то он игнорируется.

У канального уровня есть два подуровня – это *MAC* и *LLC*. *MAC* расшифровывается как *Media Access Control* и отвечает за присвоение физических *MAC*-адресов, а *LLC – Logical Link Control* – отвечает за контроль логической связи и занимается проверкой и исправлением данных, управляет их передачей.

Ethernet работает на первых двух уровнях модели *OSI*. На втором уровне *Ethernet* разделяет функции канального уровня передачи данных на два отдельных подуровня: подуровень управления логическим каналом (*LLC*) и подуровень управления доступом к среде (*MAC*). Использование этих подуровней значительно способствует совместимости между различными конечными устройствами. Подуровень *Ethernet MAC* выполняет две основные функции: инкапсуляция данных и контроль доступа к данным. *MAC* включает в себя инициирование передачи кадров и восстановление после сбоя передачи из–за коллизий. *LLC* управляет связью между верхними уровнями и сетевым программным обеспечением, а также нижними уровнями, как правило, аппаратным обеспечением.

Основной принцип работы, используемый в данной технологии, заключается в следующем. Для того чтобы начать передачу данных в сети, сетевой адаптер компьютера «прослушивает» сеть на наличие какого–либо сигнала. Если его нет, то адаптер начинает передачу данных, если же сигнал есть, то передача откладывается на определенный интервал времени. Время монопольного использования разделяемой среды одним узлом ограничивается временем передачи одного кадра.

1.2.8 Протокол *TCP/IP*

Интернет-протокол (*IP*) – это адресная система Интернета, основная функция которой заключается в доставке пакетов информации от исходного устройства к целевому устройству. *IP* – это основной способ установления сетевых соединений, и он закладывает основу компьютерных сетей. *IP* не обрабатывает порядок пакетов или проверку ошибок. Для такой функциональности требуется другой протокол, обычно *TCP*.

IP – это протокол без установления соединения, что означает, что каждая единица данных адресуется и маршрутизируется от исходного устройства к целевому устройству индивидуально, а целевое устройство не отправляет подтверждение обратно источнику. Вот тут появляются такие протоколы, как протокол управления передачей (*TCP*). *TCP* используется в сочетании с *IP* для поддержания соединения между отправителем и получателем и для обеспечения порядка пакетов.

TCP/IP определяет, как устройства передают данные между собой, причём важно, чтобы эти данные никак не изменялись во время своего пути.

Чтобы гарантировать, что каждое сообщение достигает адресата без изменений, модель *TCP/IP* разбивает данные на пакеты, а затем повторно собирает пакеты в полное сообщение на другом конце. Отправка данных небольшими пакетами упрощает поддержание точности по сравнению с отправкой всех данных сразу.

После разделения одного сообщения на пакеты эти пакеты могут перемещаться по разным маршрутам, если один маршрут перегружен. Это как отправить по почте несколько разных поздравительных открыток одному и тому же дому.

Таким образом, *TCP/IP* – сетевая модель, которая описывает процесс передачи цифровых данных, и которая была названа в честь двух главных протоколов в себе на время создания. Как и модель *OSI*, данная модель имеет уровни, но их 4, а не 7: прикладной, транспортный, сетевой и канальный. На каждом уровне работают свои протоколы (Рисунок 3), поэтому *TCP/IP* является стеком протоколов.

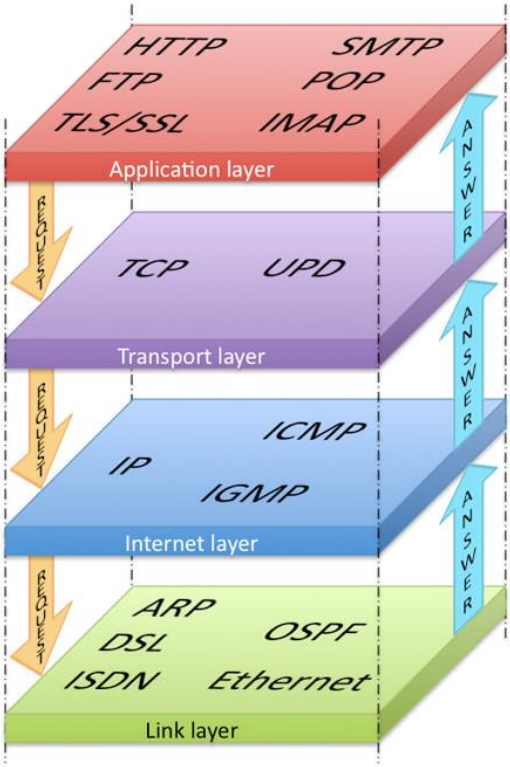

Рисунок 3 – Модель *TCP/IP*

Канальный уровень обрабатывает физический обмен электрическими сигналами. На канальном уровне, как правило, работает протокол *Ethernet*. По этому стандарту каждое устройство имеет свой уникальный *MAC*-адрес, который затем инкапсулируется в пакет данных на сетевом уровне.

Сетевой уровень формирует пакеты данных, добавляет *IP*-адреса источника и получателя к заголовкам пакетов.

Далее данные инкапсулируются на транспортный уровень, где пакеты становятся сегментами (если речь идёт про *TCP*) или дейтаграммами (*UDP*). Выделяются порядковые номера, в заголовок добавляются номера портов источника и получателя. В случае с *TCP*, который является протоколом гарантированной доставки, доставка подтверждается, а то, что потерялось, отсылается повторно.

Уровень приложения использует протоколы самого верхнего уровня, чтобы каким-либо образом отображать, изменять данные в понятном человеку виде. Схема инкапсуляции, декапсуляции, а также содержимого пакетов представлено на Рисунок 4.

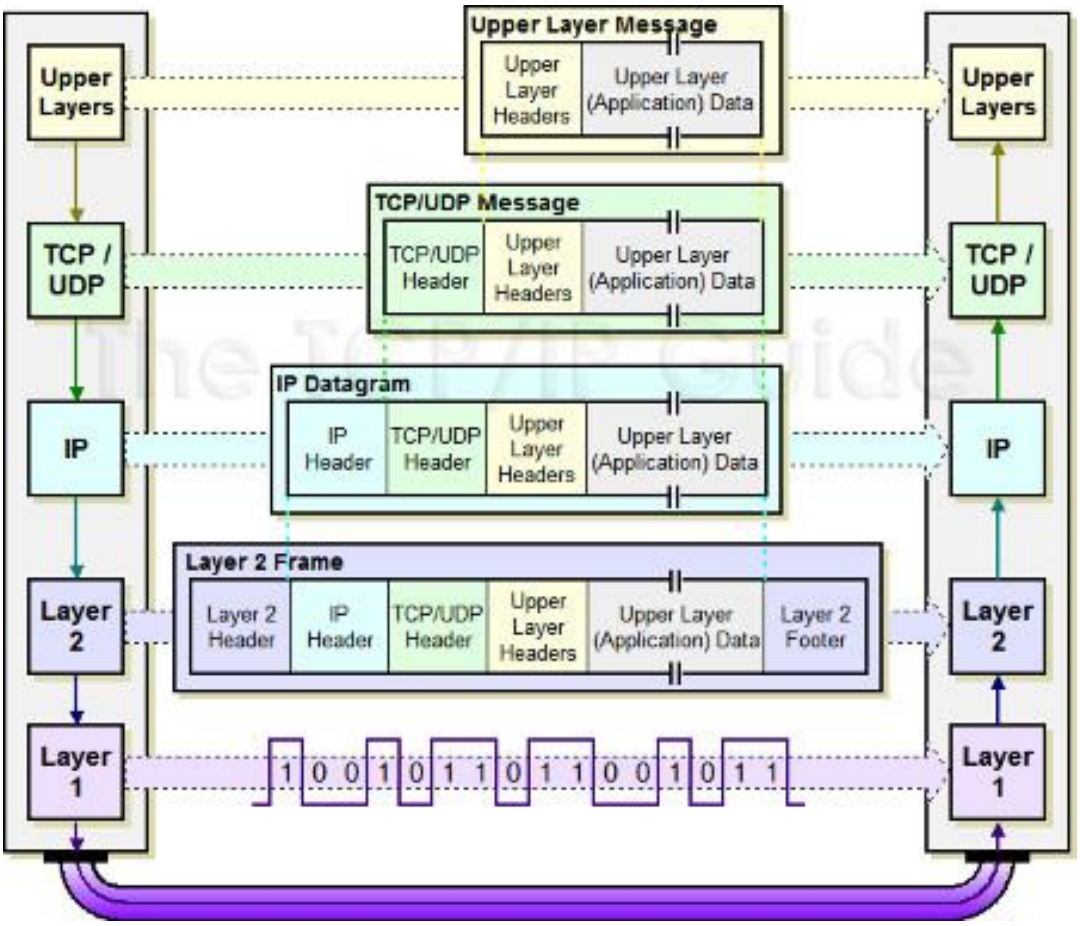

Рисунок 4 – Обработка пакетов в *TCP/IP*

Таким образом, модель *TCP/IP* остается неизменной, в то время как стандарты протоколов могут обновляться, что еще дальше упрощает работу с *TCP/IP*. Благодаря всем преимуществам стек *TCP/IP* получил широкое распространение и использовался сначала в качестве основы для создания глобальной сети, а после для описания работы интернета.

1.2.9 Обзор современных архитектур и микропроцессорной базы *IP*-видеокамер

В 1974 году компания *Texas Instruments* выпустила первый микроконтроллер, представляющий собой тип микропроцессора с оперативной памятью и устройствами ввода-вывода, интегрированными с ЦП на одном кристалле. Вместо того, чтобы соединять отдельные компоненты, их объединили на одном чипе.

В последнее время развитие мобильных систем подтолкнуло интеграцию даже дальше, чем микропроцессоры или микроконтроллеры. В результате получается система на чипе, которая может упаковать многие элементы современной компьютерной системы (графический процессор, сотовый модем, ускорители ИИ, *USB*-контроллер, сетевой интерфейс) вместе с процессором и системной памятью в одном корпусе. Это еще один шаг в непрерывной интеграции и миниатюризации электроники, которая, вероятно, будет продолжаться и в будущем.

SoC – это аббревиатура от *System on Chip*, это основа каждой *IP*-камеры (а также *DVR* и *NVR*). Для *IP*-камер *SoC* объединяет процессор сигналов изображения (*ISP*), модуль видеокодека (кодирование), процессор цифровых сигналов (*DSP*), сетевые периферийные интерфейсы, интеллектуальный механизм анализа видео и т. д.

Если знать, какая *SoC* использовалась, можно сравнивать *IP*-камеры разных брендов, которые имеют схожие функции. Однако многие производители *IP*-камер не раскрывают *SoC*, который они используют в своих продуктах. *Amabrella*, *Hisilicon*, *Texas Instruments* – три известных и распространенных производителя *SoC* для *IP*-камер. Известные бренды оборудования для *IP*-наблюдения, такие как *Hikvision* и *Dahua*, в основном *используют SoC Ambarell*а, а *Uniview* и *Tiandy* используют *SoC Hisilicon*.

Сегодня многие *SoC* основаны на процессорных ядрах *Arm*, таких как ядра *Cortex-A*, *Cortex-M* и *Cortex-R*. Другие специализированные ядра, используемые в архитектурах *SoC*, включают процессоры *Synopsys ARC*, процессоры *Cadence Tensilica Xtensa* и процессорные ядра, основанные на архитектуре набора инструкций *RISC-V*.

Архитектуры *SoC* все чаще основаны на нескольких ядрах. При симметричной многопроцессорной обработке вычисления распределяются по нескольким процессорным ядрам. В асимметричной многопроцессорной обработке у ядер могут быть совершенно разные роли: одни выполняют задачи управления вводом-выводом в реальном времени, а другие выполняют исполнительные функции.

Архитектуры *SoC* могут включать в себя различные типы памяти и конфигурации. Статическая оперативная память (*SRAM*) может использоваться для регистров процессора и быстрых кэшей уровня 1 (или *L1*),

в то время как динамическая оперативная память (*DRAM*) часто составляет основную память нижнего уровня *SoC*.

Многие периферийные устройства были включены в архитектуры *SoC*, часто для работы с часто используемыми протоколами связи. Популярные интерфейсы включают *GPIO, PCI-Express, Gigabit Ethernet, CAN, SPI, USB, UART* и *I2C*.

1.2.10 Структура микроконтроллерного ядра *ARM Cortex–M4*

Аббревиатура *ARM* известна как *Acorn RISC Machines*, но со временем она была переименована на *Advanced RISC Machines*. Важным моментом здесь является то, что *ARM* не разрабатывает кремниевые чипы для микроконтроллеров, а только предоставляет ядро для микропроцессора и других строительных блоков микроконтроллера. Затем *ARM* предоставляет его различным производителям посредством лицензирования.

Микроконтроллеры *Cortex-M* представляют собой одно из направлений развития микропроцессорных ядер, предлагаемых фирмой *ARM*. Фактически, под общей торговой маркой *Cortex* можно увидеть три типа процессоров, обозначаемых буквами *A, R, M*. Профиль *A* характеризуется достижением большой вычислительной мощности. Профиль *R* нацелен на использование во встраиваемых системах, поэтому эти процессоры модернизированы для исполнения задач в реальном времени. Основной задачей профиля *M* заявлена простота и низкая стоимость. Технически *Cortex-M* представляют сильно упрощенные варианты старших моделей. Тем не менее, даже такие «урезанные» контроллеры обладают вычислительной мощностью, значительно превышающей многие аналоги.

Семейство *Cortex-M* состоит из *Cortex-M0, Cortex-M0+, Cortex-M1, Cortex-M3, Cortex-M4 и Cortex-M7*. Рассматриваемое ядро *Cortex-M4* основано на архитектуре *ARMv7*.

Структура *ARM Cortex-M4* представлена на Рисунок 5 – *Структура* ARM Cortex-M4

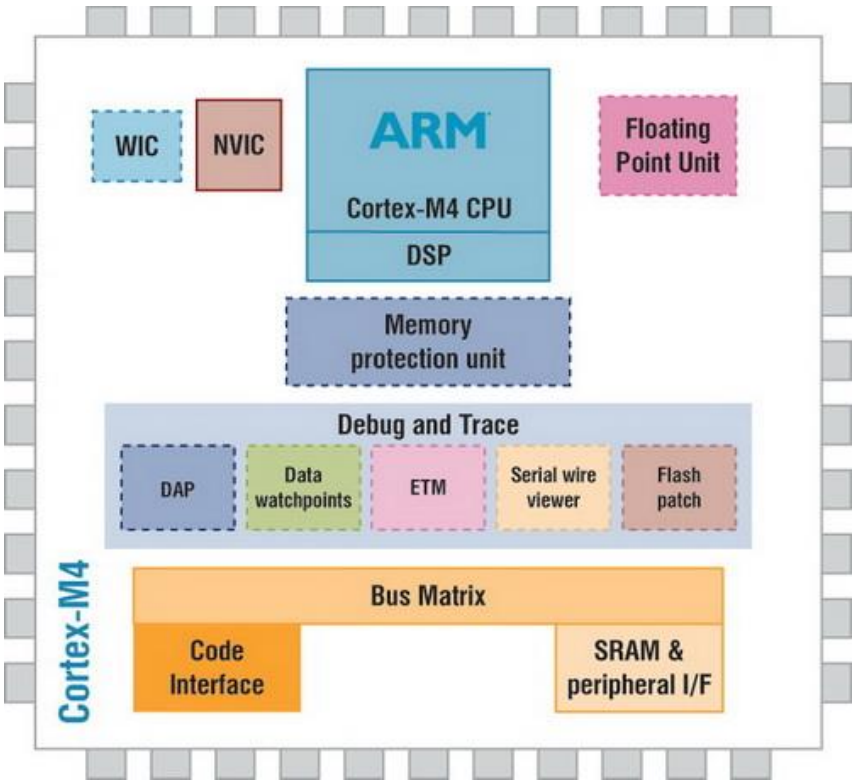

Рисунок 5 – Структура *ARM Cortex-M4*

ARM Cortex-M4 состоит из следующих компонентов:
– ядро процессора;
– контроллер вложенных векторных прерываний;
– блок с системой отладки;
– система шин и матрица шин;
– память;
– блок операций с плавающей точкой;

Микроконтроллер *ARM Cortex-M4* поддерживает 240 системных и периферийных прерываний, а *NVIC* осуществляет управление прерываниями с помощью таблицы векторов прерываний. Также микроконтроллеры *ARM Cortex-M4* обычно основаны на 32-разрядной архитектуре *RISC*.

1.2.11 Регистровая модель портов ввода–вывода общего назначения микроконтроллера с ядром *ARM Cortex–M4*

GPIO – это сигнальный контакт на интегральной схеме или плате, который можно использовать для выполнения функций цифрового ввода или вывода. По своей конструкции он не имеет заранее определенной формы и может использоваться разработчиком аппаратного или программного обеспечения для выполнения выбранных им функций. Типичные области применения включают в себя управление светодиодами, переключателями считывания и управление различными типами датчиков.

GPIO может быть реализован с помощью выделенных интегральных схем или поддерживаться устройствами системы на кристалле (*SoC*) или системой на модуле (*SoM*).

Что касается *ARM Cortex-M4* и *GPIO*, то важно упомянуть про разделение пространства памяти на блоки, часть которых представлена на Рисунок 6 – *Разметка участка памяти*.

0x4002 2C00 - 0x4002 2FFF	Reserved
0x4002 2800 - 0x4002 2BFF	
0x4002 2400 - 0x4002 27FF	
0x4002 2000 - 0x4002 23FF	
0x4002 1C00 - 0x4002 1FFF	GPIOH
0x4002 1800 - 0x4002 1BFF	GPIOG
0x4002 1400 - 0x4002 17FF	GPIOF
0x4002 1000 - 0x4002 13FF	GPIOE
0X4002 0C00 - 0x4002 0FFF	GPIOD
0x4002 0800 - 0x4002 0BFF	GPIOC
0x4002 0400 - 0x4002 07FF	GPIOB
0x4002 0000 - 0x4002 03FF	GPIOA

Рисунок 6 – Разметка участка памяти

Как можно увидеть, *GPIO* разделён на *GPIO A, GPIO B, GPIO C, GPIO D, GPIO E, GPIO F, GPIO G, GPIO H*.

Каждый порт в контроллерах с ядром *ARM Cortex-M4* состоит из 16 контактов. С каждым выводом связаны различные регистры, изменяя содержимое регистров, можно управлять поведением конкретного вывода.

В контроллерах с ядром *ARM Cortex-M4* поведение каждого вывода можно контролировать с помощью:

– *GPIO Mode Register* (регистр режима *GPIO*);

– *GPIO Output Type Register* (регистр типа вывода *GPIO*);

– *GPIO Speed Registe*r (регистр скорости *GPIO*);

– *GPIO Pull-up/Pull-down Register* (подтягивающий/вытягивающий регистр *GPIO*);

– *GPIO Input data Register* (регистр входных данных *GPIO*);

– *GPIO Output data Register* (регистр выходных данных *GPIO*);

– *GPIO bit set/reset Register* (установка/сброс бита);

– *GPIO Configuration Lock Register* (регистр блокировки конфигурации *GPIO*);

– *GPIO Alternate Functionality Register* (регистр альтернативной функциональности *GPIO*).

Регистр режима *GPIO* используется для выбора режима вывода. В этом регистре можно запрограммировать четыре режима: режим ввода, режим вывода общего назначения, режим альтернативной функции и аналоговый режим. Режим вывода общего назначения используется, когда нужно что-то подать на выход. Каждый вывод имеет соответствующий выходной буфер, который может быть записан программным обеспечением. Альтернативный функциональный режим используется, когда нужно назначить конкретный вывод любому другому периферийному устройству, например: если используется *I2C*, то понадобятся два контакта, для *SDA* и *SCL*. Аналоговый режим работы если нужны возможности аналого-цифрового или цифро-аналогового преобразователя.

Регистр типа вывода *GPIO* – этот регистр записывается программным обеспечением для настройки типа вывода. Возможны два типа выхода: выход двухтактный и выход с открытым стоком.

Регистр скорости *GPIO* – это параметр скорости, который определяет, насколько быстро он может перейти от уровня 0 к уровню логической единицы и наоборот.

Регистр *GPIO Pull-up/Pull-down*. Используя этот регистр, можно включать/отключать управлять током проводящей линии на месте контакта.

Регистр входных данных *GPIO*. Этот регистр можно использовать для установки состояния чтения на выводе. Может считывать до 4 байтов.

Регистр выходных данных *GPIO* – этот регистр используется для установки значения 0 или 1 на выходном контакте.

Регистр установки/сброса битов *GPIO*. Этот регистр позволяет устанавливать и сбрасывать каждый отдельный бит в регистре выходных данных *GPIO*.

Регистр блокировки конфигурации *GPIO* – этот регистр используется для блокировки конфигурации вывода.

Регистр альтернативных функций *GPIO*. Этот регистр используется для настройки контактов в качестве альтернативных функций, т. е. эти контакты могут использоваться другими периферийными устройствами в микроконтроллере. Однако конкретный вывод может быть связан только с одним периферийным устройством одновременно.

1.2.12 Физический и канальный уровни интерфейса *I2C*

Протокол связи *I2*C (*Inter-Integrated Circuit*) был разработан компанией *Philps*. В настоящее время мало где используется ввиду низкой пропускной способности, но широко использовался для связи между несколькими интегральными схемами (ИС) в системе. Состоит из двух уровней – физического и канального.

Физический уровень представлен шиной, в которой есть две линии шины: двунаправленная последовательная линия данных (*SDA*) и

последовательная линия синхронизации (*SCL*). Линия данных используется для представления данных, а линия синхронизации используется для синхронизации передачи и приема данных.

Каждое устройство, подключенное к шине, имеет независимый адрес, и хост может использовать этот адрес для доступа между различными устройствами.

На шине *I2C* высокий уровень представляет собой логическую 1, низкий уровень представляет собой логический 0, а другое состояние является состоянием с высоким импедансом.

Когда несколько хостов используют шину одновременно, для предотвращения конфликтов данных используется арбитраж для определения того, какое устройство занимает шину.

I2C имеет три режима передачи: стандартный режим имеет скорость передачи 100 кбит/с, быстрый режим – 400 кбит/с, а высокоскоростной режим может достигать 3,4 Мбит/с, но большинство устройств *I2C* в настоящее время не поддерживают высокоскоростной режим.

Количество микросхем, подключенных к одной шине, ограничено максимальной емкостью шины 400 пФ.

Логический уровень *I2C* определяет сигналы запуска и остановки, достоверность данных, ответ, арбитраж, синхронизацию тактов и адресную широковещательную передачу связи.

После того, как сгенерирован стартовый сигнал, все ведомые устройства начинают ждать адресного сигнала ведомого устройства (*SLAVE_ADDRESS*), которое затем передает ведущее устройство. На шине *I2C* адрес каждого устройства уникален. Когда адрес, передаваемый хостом, совпадает с адресом определенного устройства, устройство выбирается, а невыбранное устройство игнорирует последующий сигнал данных. В соответствии с протоколом *I2C* этот адрес ведомого устройства может быть 7- или 10-битным.

После адресного бита идет бит выбора направления передачи. Когда этот бит равен 0, это означает, что последующее направление передачи данных – от хоста к ведомому, то есть хост записывает данные в ведомый. Когда этот бит равен 1, происходит обратное, то есть ведущий считывает данные с ведомого.

После того, как ведомое устройство получит соответствующий адрес, ведущее или ведомое устройство вернет сигнал подтверждения (*ACK* = 0) или отклонения (*NACK* = 1).

Только после получения сигнала подтверждения мастер может продолжать отправлять или получать данные.

Размер пакета данных составляет 8 бит, и хост будет отправлять его каждый раз. Для байтовых данных необходимо дождаться ответного сигнала (*ACK*) от ведомого устройства. Чтобы повторять неограниченное число байтов данных, нужно просто повторять этот процесс. Когда передача данных

завершена, мастер отправляет подчиненному сигнал остановки передачи, что означает, что данные больше не передаются.

Когда ведущий хочет прекратить прием данных, он возвращает сигнал отклонения (*NACK*) ведомому, и мастер автоматически прекращает передачу данных.

Когда линия *SCL* имеет высокий уровень, линия *SDA* переключается с высокого уровня на низкий. Эта ситуация свидетельствует о начале общения.

Когда *SCL* находится на высоком уровне, линия *SDA* переключается с низкого уровня на высокий уровень, указывая на остановку связи.

I2C использует сигнальную линию *SDA* для передачи данных и сигнальную линию *SCL* для синхронизации данных. Линия данных *SDA* передает один бит данных в каждом такте *SCL*.

1.2.13 Регистровая модель *I2C* микроконтроллера на базе ядра *ARM Cortex–M4*

Регистровая модель *I2C* у *STM32* представлена на Рисунок 7 – Регистровая модель микроконтроллера.

Регистр управления *I2C_CR1* содержит в себе:

– *SWRST(Software reset)* – единица в этом бите сбрасывает значение всех регистров модуля в базовое стояние, может использоваться для сброса при возникновении ошибки.

– *ALERT(SMBus alert)* – установка единицы в этот бит разрешает генерировать сигнал *alert* в режиме *SMBus*;

– *PEC(Packet error checking)* – управление этим битом производится программно, но он может быть сброшен аппаратно когда передается *PEC*, *START*, *STOP* или *PE*=0. Единица в этом бите разрешает передачу *CRC*;

– *POS(Acknowledge/PEC Position (for data reception))* – состояние этого бита определяет положение *ACK/PEC* в двух байтовой конфигурации в режиме *Master*;

– *ACK(Acknowledge enable)* – единица в этом бите разрешает отправлять *ACK/NACK* после приема байта адреса или данных;

– *STOP(Stop generation)* – установка единицы в этот бит генерирует сигнал *STOP* в режиме *Master*;

– *START(Start generation)* – установка единицы в этот бит генерирует состояние *START* в режиме *Master*;

– *NOSTRETCH(Clock stretching disable (Slave mode))* – если на обработку данных требуется время *Slave* может остановить передачу мастера, прижав линию *SCL* к земле, *Master* будет ждать и не будет ни чего слать, пока линия не будет отпущена. Ноль в этом бите прижимает SCL к земле;

– *ENGC(General call enable)* – если в этом бите установлена единица, модуль отвечает *ACK* на широковещательный адрес 0x00.

– *ENPEC(PEC enable)* – установка единицы в этот бит включает аппаратный подсчет *CRC*;

– *ENARP(ARP enable)* – установка единицы в этот бит включает *ARP*;

– *SMBTYPE(SMBus type)* – если в этом бите установлен ноль модуль работает в режиме *Slave*, если единица в режиме *Master*;

– *SMBUS(SMBus mode)* – если в этом бите установлен ноль модуль работает в режиме *I2C*, если единица *SMBus*;

– *PE(Peripheral enable)* – единица в этом бите включает модуль.

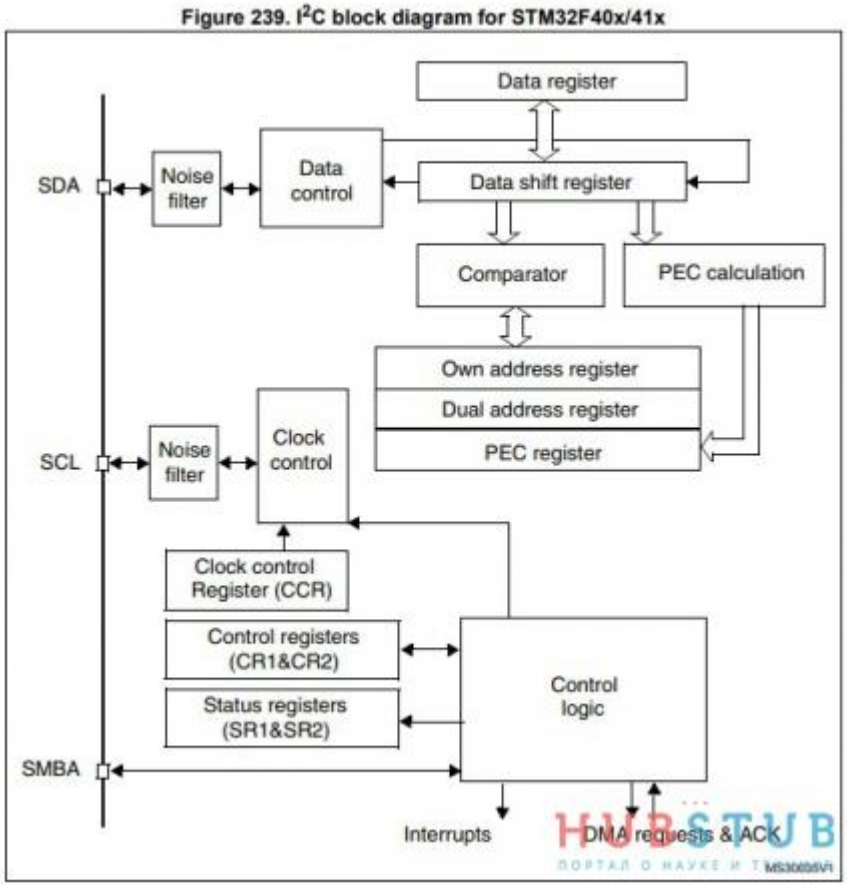

Рисунок 7 – Регистровая модель микроконтроллера

Регистр управления *I2C_CR2* включает в себя:

– *LAST(DMA last transfer)* – единица в этом бите разрешает *DMA* генерировать сигнал окончания передачи *EOT(End of Transfer);*

– *DMAEN(DMA requests enable)* – единица в этом бите разрешает делать запрос к *DMA* при установке флагов *TxE* или *RxNE*;

– *ITBUFEN(Buffer interrupt enable)* – если этот бит сброшен, разрешены все прерывания, кроме прерываний по приему и передаче;

– *ITEVTEN(Event interrupt enable)* – единица в этом бите разрешает прерывания по событию;

– *ITERREN(Error interrupt enable)* – единица в этом бите разрешает прерывания при возникновении ошибок;

– *FREQ[5:0](Peripheral clock frequency)* – в это битовое битовое поле необходимо записать частоту тактирования модуля, она может принимать значение от 2 до 50.

Регистр I2C_OAR1 состоит из:

– *ADDMODE(Addressing mode)* – этот бит определяет размер адреса *Slave*, ноль соответствует размеру адреса 7 бит, единица – 10 бит;

– *ADD[9:8](Interface address)* – старшие биты адреса, в случае если адрес 10-битный;

– *ADD[1:7](Interface address)* – адрес устройства;

– *ADD0(Interface address*) – младший бит адреса, в случае если адрес 10-битный.

Регистр I2C_OAR2:

– *ADD2[7:1]* – альтернативный адрес на который будет отзываться *Slave*;

– *ENDUAL(Dual addressing mode enable)* – единица в этом бите разрешает *Slave* отзываться на альтернативный адрес в 7-битном режиме;

– *I2C_DR* – регистр данных, для отправки данных пишем в регистр *DR*, для приёма читаем его же.

Регистр статуса *I2C_SR1*:

– *SMBALERT(SMBus alert)* – возникает в случае *alert* в шине *SMBus*;

– *TIMEOUT (Timeout or Tlow error)* – возникает если линия *SCL* прижата к земле. Для *master 10mS*, для *slave 25mS*;

– *PECERR (PEC Error in reception)* – возникает при ошибке *PEC* при приеме;

– *OVR (Overrun/Underrun)* – возникает при переполнении данных;

– *AF (Acknowledge failure)* – устанавливается при получении сигнала *NACK*. Для сброса нужно записать 0;

– ARLO *(Arbitration lost (master mode))* – устанавливается при потере арбитража. Для сброса нужно записать 0;

– *BERR (Bus error)* – ошибка шины. Устанавливается в случае возникновения сигнала *START* или *STOP* в неправильный момент;

– *TxE (Data register empty (transmitters))* – устанавливается при опустошении регистра *DR*, а точнее когда данные из него были перемещены в сдвиговый регистр;

– *RxNE (Data register not empty (receivers))* – устанавливается при приеме байта данных, кроме адреса;

– *STOPF(Stop detection (slave mode))* – при работе в режиме *slave* устанавливается при обнаружении сигнала *STOP*, если перед этим был сигнал *ACK*. Для сброса необходимо прочитать *SR1* и произвести запись в *CR1*;

– *ADD10 (10-bit header sent (Master mode))* – устанавливается при отправке первого байта 10-битного адреса;

– *BTF (Byte transfer finished)* – флаг устанавливается по окончании приема/передачи байта, работает только при *NOSTRETCH* равном нулю;

– *ADDR(Address sent (master mode)/matched (slave mode))* – в режиме *master* устанавливается после передачи адреса, в режиме *slave* устанавливается при совпадении адреса. Для сброса нужно прочитать регистр *SR1*, а затем *SR2*;

– *SB(Start bit (Master mode))* – устанавливается при возникновении сигнала *START*. Для сброса флага необходимо прочитать *SR1* и записать данные в регистр *DR* .

Регистр статуса *I2C_SR2*:

– *PEC[7:0](Packet error checking register)* – в это битовое поле записывается контрольная сумма кадра;

– *DUALF(Dual flag (Slave mode))* – ноль в этом бите говорит о том, что адрес который принял *Slave* соответствует *OAR1*, иначе *OAR2*;

– *SMBHOST(SMBus host header (Slave mode))* – устанавливается, когда принят заголовок *SMBus Host*;

– *SMBDEFAULT(SMBus device default address (Slave mode))* – устанавливается, если принят адрес по умолчанию для *SMBus*-устройства;

– *GENCALL(General call address (Slave mode))* – устанавливается, если принят широковещательный адрес в режиме ведомого;

– *TRA(Transmitter/receiver)* – единица в этом бите говорит о том, что модуль работает как передатчик, иначе приемник;

– *BUSY(Bus busy)* – флаг занятости;

– *MSL(Master/slave)* – единица в этом бите говорит о том, что модуль работает в режиме *Master*, иначе *Slave*;

Регистр управления частотой *I2C_CCR*:

– *F/S(I2C master mode selection)* – при установке единицы в этот бит модуль работает в режиме *FAST*, иначе *STANDART*;

– *DUTY (Fm mode duty cycle)* – этот бит задает скважность сигнала *SCL* в режиме *FAST*. Если установлен ноль *tlow/thigh* = 2, иначе *tlow/thigh* = 16/9;

– *CCR[11:0](Clock control register in Fm/Sm mode (Master mode))* – при работе в режиме *Master* задает тактовую частоту линии *SCL*;

– *Sm mode* или *SMBus*;

– *Fm mode*.

Регистр *I2C_TRISE*:

– *TRISE[5:0]* – определяет время нарастания фронта.

– Регистр управления фильтрами I2C_FLTR:

– *ANOFF(Analog noise filter OFF)* – ноль в этом бите включает аналоговый фильтр;

– *DNF[3:0](Digital noise filter)* – битовое поле для настройки цифрового фильтра. За подробностями нужно обратиться к документации.

Таким образом была описана регистровая модель I2C микроконтроллера STM32 с описанием каждого регистра.

1.2.14 Интерфейс *DCMI*

DCMI – это интерфейс цифровой камеры. *DCMI* интерфейс используется для подключения параллельного модуля камеры к *STM32*. Камера генерирует параллельный поток данных вместе с пиксельным тактовым сигналом (*DCMI_PIXCLK*), который позволяет интерфейсу захватывать входящий поток данных. Можно использовать два дополнительных сигнала (*HSYNC* и *VSYNC*) для синхронизации кадра изображения между камерой и микроконтроллером. *DCMI* также поддерживает встроенные коды синхронизации строки/кадра в потоке данных.

DCMI позволяет выполнять непрерывный захват. Этот процесс начинается по запросу приложения и продолжается до тех пор, пока бит *CAPTURE* не очистится. В качестве альтернативы снимку, интерфейс позволяет захватить один кадр по запросу приложения. Благодаря функции обрезки интерфейс камеры может вырезать и сохранить прямоугольную часть полученного изображения.

Стандартным способом использования интерфейса камеры является сохранение полученных данных в кадровом буфере в ОЗУ. Далее ядро контроллера может обрабатывать эти данные или передавать их дальше через другой интерфейс (например, *USB* или *Ethernet*).

1.2.15 Регистровая модель DCMI микроконтроллера с ядром ARM Cortex–M4

Аппаратный модуль *DCMI* содержит регистр данных *(DCMI_DR)*, а также десять регистров управления/статуса:

– регистр управления *(DCMI_CR);*

– регистр состояния *(DCMI_SR)*;

– регистр состояния прерываний *(DCMI_RIS)*;

– регистр разрешения прерываний *(DCMI_IER);*

– регистр маски прерываний *(DCMI_MIS)*;

– регистр сброса флагов прерываний *(DCMI_ICR);*

– регистр кодов внутренней синхронизации *(DCMI_ESCR)*;

– регистр сброса маски кодов внутренней синхронизации *(DCMI_ESUR);*

– регистр стартовых значений при захвате части кадра *(DCMI_CWSTRT);*

– регистр величины фрагмента кадра в режиме *CropWindow (DCMI_CWSIZE).*

Блок схема работы *DCMI* представлена на Рисунок 8 – Блок схема *DCMI*.

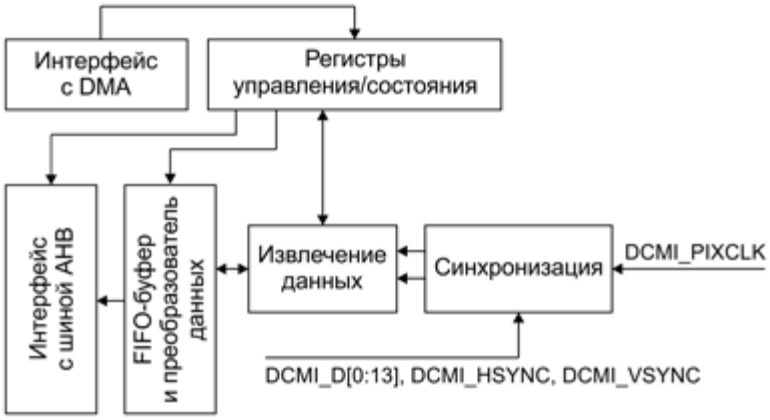

Рисунок 8 – Блок схема *DCMI*

DCMI обеспечивает два режима работы: с однократным захватом кадра *(Snapshot mode)* или с потоковым захватом видео *(Continuous mode).* Режим однократного захвата кадра активируется после установки бита *CM* в регистре *DCMI_CR*. Для инициализации захвата необходимо установить бит CAPTURE в регистре *DCMI_CR*. Далее *DCIM*-контроллер ждет начала первого кадра и выполняет его захват. После чего бит *CAPTURE* автоматически сбрасывается, а прием следующего кадра не выполняется.

1.2.16 Принципы функционирования блока *DMA* прямого доступа к памяти

Прямой доступ к памяти *(DMA)* – это метод, который позволяет устройству ввода-вывода *(I/O)* отправлять или получать данные непосредственно в основную память или из нее, минуя ЦП для ускорения операций с памятью.

Определенная часть памяти используется для отправки данных напрямую с периферийного устройства на материнскую плату без участия микропроцессора, чтобы этот процесс не мешал работе компьютера в целом.

В старых компьютерах четыре канала *DMA* были пронумерованы 0, 1, 2 и 3. Когда была введена 16-битная шина расширения промышленного стандарта *(ISA),* были добавлены каналы 5, 6 и 7.

Канал *DMA* позволяет устройству передавать данные, не подвергая ЦП рабочей перегрузке. Без каналов прямого доступа к памяти ЦП копирует каждый фрагмент данных с устройства ввода-вывода, используя периферийную шину. Использование периферийной шины занимает центральный процессор во время процесса чтения/записи и не позволяет выполнять другую работу, пока операция не будет завершена.

С *DMA* ЦП может выполнять другие задачи, пока выполняется передача данных. Передача данных сначала инициируется ЦП. Блок данных может быть передан в память и из памяти с помощью DMAC тремя способами.

Стандартный прямой доступ к памяти (также называемый сторонним прямым доступом к памяти) использует контроллер прямого доступа к памяти. Контроллер прямого доступа к памяти может создавать адреса памяти и запускать циклы чтения или записи памяти. Он охватывает несколько аппаратных регистров, которые могут быть прочитаны и записаны ЦП.

Эти регистры состоят из регистра адреса памяти, регистра счетчика байтов и одного или нескольких управляющих регистров. В зависимости от функций, предоставляемых контроллером прямого доступа к памяти, эти управляющие регистры могут назначать некоторую комбинацию источника, назначения, направления передачи (чтение с устройства ввода-вывода или запись на него), размер блока передачи и/или количество байт для передачи в одном пакете.

Для выполнения операций ввода, вывода или памяти в память хост-процессор инициализирует *DMA*-контроллер количеством байтов для передачи и используемым адресом памяти. Затем ЦП дает команду периферийному устройству начать передачу данных. Затем контроллер прямого доступа к памяти предлагает адреса и линии управления чтением/записью в системную память. Каждый раз, когда байт данных готовится к передаче между периферийным устройством и памятью, контроллер прямого доступа к памяти увеличивает свой внутренний адресный регистр до тех пор, пока не будет передан полный блок данных.

Прямой доступ к памяти работает по-разному в разных режимах работы:

1 В пакетном режиме полный блок данных передается в непрерывной последовательности. Как только ЦП разрешает контроллеру *DMA* доступ к системной шине, контроллер *DMA* передаст все байты данных в блоке данных, прежде чем передать управление системными шинами обратно ЦП, но это приведет к тому, что ЦП будет неактивен в течение некоторого времени. Этот режим также называется «режимом блочной передачи».

2 Циклический режим используется в системе, в которой невозможно отключить ЦП на время, необходимое для режима пакетной передачи. В режиме захвата цикла контроллер *DMA* получает доступ к системной шине с помощью сигналов *BR* (запрос шины) и *BG* (предоставление шины), которые аналогичны пакетному режиму. Эти два сигнала управляют интерфейсом между ЦП и контроллером прямого доступа к памяти. С одной стороны, в циклическом режиме скорость передачи блоков данных не такая высокая, как в пакетном режиме, а с другой стороны, время простоя процессора не такое продолжительное, как в пакетном режиме.

3 В прозрачном режиме передача блоков данных занимает больше всего времени, но это также и самый эффективный режим с точки зрения общей производительности системы. В прозрачном режиме контроллер прямого доступа к памяти передает данные только тогда, когда ЦП выполняет операции, не использующие системные шины. Основное преимущество прозрачного режима заключается в том, что ЦП никогда не прекращает

выполнение своих программ, а передачи с прямым доступом к памяти бесплатны с точки зрения времени, а недостатком является то, что аппаратному обеспечению необходимо определять, когда ЦП не использует системные шины, что может быть сложным. Это также называется «скрытый режим передачи данных *DMA*».

1.2.17 Методика обработки прерывания *DMA*

Прямой доступ к памяти (*DMA*) работает следующим образом:

– устройство, желающее выполнить *DMA*, устанавливает сигнал запроса шины процессора;

– процессор завершает текущий цикл шины, а затем выдает на устройство сигнал предоставления шины;

– затем устройство подтверждает сигнал подтверждения предоставления шины;

– процессор улавливает изменение состояния сигнала подтверждения предоставления шины и начинает прослушивать данные и адресную шину на предмет активности контроллера-*DMA;*

– контроллер-*DMA* выполняет передачу с адреса источника на адрес назначения.

Во время этих передач процессор отслеживает адреса на шине и проверяет, кэшируются ли в процессоре какие-либо адреса, измененные во время операций прямого доступа к памяти. Если процессор обнаруживает кэшированный адрес на шине, он может предпринять одно из двух действий: сделать недействительной запись внутреннего кэша для адреса, участвующего в операции записи или обновить внутренний кэш при обнаружении записи.

После завершения операций прямого доступа к памяти устройство освобождает шину, подавая сигнал освобождения шины. Процессор подтверждает освобождение шины и возобновляет свои циклы шины с того места, где он был остановлен.

Непосредственно обработка прерываний может выполняться по двум сценариям – когда аппаратное обеспечение не поддерживает идентификацию устройства, инициировавшего прерывание, и когда поддерживает.

В случаях, когда идентификация устройства не поддерживается на аппаратном уровне, возможные прерывающие устройства должны быть опрошены программно:

– устройство устанавливает сигнал прерывания на аппаратно-запрограммированном уровне прерывания;

– процессор регистрирует прерывание и ожидает завершения выполнения текущей инструкции;

– как только выполнение текущей инструкции завершено, процессор инициирует обработку прерывания, сохраняя содержимое текущего регистра в стеке;

– затем процессор переключается в режим супервизора и инициирует цикл подтверждения прерывания;

– ни одно устройство не отвечает на цикл подтверждения прерывания, поэтому процессор выбирает вектор, соответствующий уровню прерывания;

– адрес, найденный в векторе, является адресом процедуры обслуживания прерываний (*ISR*);

– *ISR* опрашивает все устройства, чтобы найти устройство, вызвавшее прерывание. Это достигается путем проверки регистров состояния прерывания на устройствах, которые могли инициировать прерывание;

– как только устройство обнаружено, управление передается обработчику, специфичному для прерывающего устройства;

– после того, как специфичная для устройства подпрограмма *ISR* выполнила свою работу, *ISR* выполняет инструкцию «возврата из прерывания».

Выполнение команды «возврат из прерывания» приводит к восстановлению состояния процессора. Процессор возвращается в пользовательский режим.

В случаях, когда идентификация устройства поддерживается на аппаратном уровне, возможные прерывающие устройства идентифицируются на аппаратном уровне:

– устройство устанавливает сигнал прерывания на аппаратно-запрограммированном уровне прерывания;

– процессор регистрирует прерывание и ожидает завершения выполнения текущей инструкции;

– как только выполнение текущей инструкции завершено, процессор инициирует обработку прерывания, сохраняя содержимое текущего регистра в стеке;

– затем процессор переключается в режим супервизора и инициирует цикл подтверждения прерывания;

– прерывающее устройство отвечает на цикл подтверждения прерывания номером вектора для прерывания;

– процессор использует номер вектора, полученный выше, и выбирает вектор;

– адрес, найденный в векторе, является адресом процедуры обслуживания прерываний (*ISR*) прерывающего устройства.

После того, как подпрограмма *ISR* выполнила свою работу, *ISR* выполняет команду «возврата из прерывания».

Выполнение команды «возврат из прерывания» приводит к восстановлению состояния процессора. Процессор возвращается в пользовательский режим.

1.2.18 Структура и логика функционирования цифровой видеокамеры на базе процессора *OV9655*

OV9655 CameraChip представляет собой датчик изображения, обеспечивающий полную функциональность одночиповой камеры *SXGA* (1280x1024) и процессора изображений в одном корпусе. OV9655 обеспечивает полнокадровый, субдискретизированный, масштабированный или оконный режим изображения в широком диапазоне форматов, управляемый через интерфейс последовательной шины управления камерой (*SCCB*). Камера способна работать в режиме 15 кадров в секунду (*fps*) в разрешении *SXGA* с полным пользовательским контроль над качеством изображения, форматированием и передачей выходных данных.

Вся необходимая обработка изображений функции, такая как управление экспозицией, гамма, баланс белого, насыщенность цвета, регулировка оттенка, шумоподавление и многое другое, также программируется через интерфейс *SCCB*. Структура *OV9655* приведена на Рисунок 9 – Структура *OV9655*.

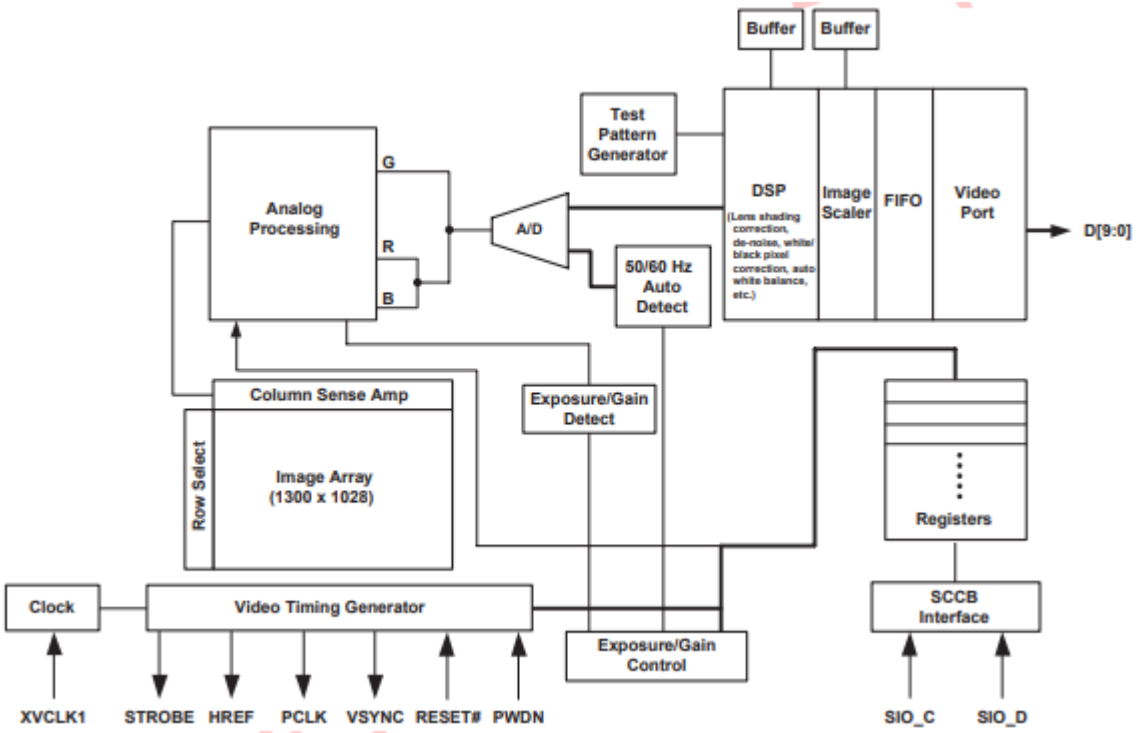

Рисунок 9 – Структура *OV9655*

Как можно увидеть на схеме, *OV9655* содержит в себе следующие компоненты:

– массив пикселей;
– аналоговый обработчик сигналов;
– аналого-цифровые преобразователи;

– цифровой обработчик сигналов;
– блок форматирования выхода;
– синхронизирующий генератор;
– интерфейс *SCCB*;
– цифровой видео интерфейс.

Массив пикселей определяет максимальное разрешение камеры, а именно 1300 столбцов на 1028 строк (всего 1336400 пикселей).

Синхронизирующий генератор управляет генерацией кадров, генерацией внутренних сигналов синхронизации, синхронизацией частоты кадров, контролем экспозиции, а также выходами внешней синхронизации.

Аналоговый обработчик сигналов – блок, который выполняет управление экспозицией и балансом белого у аналогового изображения.

Цифровой обработчик сигналов – блок, который управляет изменением данных в *RGB* и некоторым контролем качества изображения: устранять цветовые помехи, контролировать оттенок и насыщенность, шумоподавление, удаление белых пикселей.

Блок форматирования выхода управляет форматированием данных перед отправкой изображения (например, уменьшение размера изображения).

Интерфейс последовательной шины управления камерой (*SCCB*) контролирует все происходящие операции.

1.2.19 Физический и канальный уровни интерфейса *MII*

MII (независимый от среды интерфейс) — это стандарт, используемый для соединения блока *MAC* (управление доступом к среде) с физическим уровнем *PHY* для сетевых устройств. Эти две части общего сетевого устройства *Ethernet* выполняют разные функции в рамках модели *OSI* (межсетевого взаимодействия открытых систем). Каждый уровень системы выполнял следующие функции:

1 *PHY* (физический уровень) – это микросхема, которая преобразует цифровые данные из *MAC* и отправляет их по физическому сетевому интерфейсу в виде аналогового сигнала. Эти микросхемы функционируют как приемопередатчики, поэтому они модулируют аналоговые сигналы, передаваемые по физическому уровню. При подключении к физическому каналу по *Ethernet* выходные данные *PHY* отправляются на приемопередатчик для преобразования модулированных аналоговых сигналов в цифровой сигнал.

2 *MAC* работает на логическом уровне и функционирует как интерфейс между *CPU/FPGA/MCU/ASIC* для обработки данных и связи с чипом *PHY*. *MAC* обеспечивает необходимые возможности обработки данных, а также отправляет данные и получает данные от *PHY*.

Стандарт *MII* передает 4-битные блоки данных между *MAC* и *PHY* для передачи данных *TX* и *RX*. *PHY* работает на частоте 2,5 МГц (режим 10 Мбит/с) или 25 МГц (режим 100 Мбит/с). Связь в *MII* не является двунаправленной, поэтому определенные сигналы разделяются на наборы сигналов *TX* и *RX*. Тактовый сигнал, используемый для управления *PHY*, также используется для того, чтобы *MAC* отправлял *TX*-данные на *PHY*; данные отправляются с MAC на *PHY* по переднему фронту этого импульса, что позволяет передавать данные синхронно.

Микросхемы *PHY* также используются в других протоколах связи, таких как *USB, SATA* и беспроводная локальная сеть/*Wi-Fi*. Некоторые функции могут быть интегрированы в уровень *MAC*, в зависимости от соответствующих приложений. В *Ethernet* количество сигналов, необходимых для связи *PHY* с *MAC*, довольно велико в соответствии со стандартом *MII*, поэтому был разработан стандарт *RMII* для уменьшения количества сигналов.

1.2.20 Физический и канальный уровни интерфейса *RMII*

Что касается *RMII*, то разница с *MII* лишь в тактовой частоте и количестве сигналов для связи с *MAC*.

Каждая микросхема *PHY* управляет одним физическим интерфейсом, поэтому печатные платы для таких устройств, как сетевые коммутаторы, содержат множество каналов для обеспечения связи между *PHY* и *MAC*. В *MII* каждому *PHY* требуется 18 сигналов для связи с *MAC*, и только 2 из этих сигналов могут совместно использоваться несколькими устройствами *PHY*. Поэтому *RMII* (сокращенный MII) был разработан как вариант *MII*, чтобы вдвое сократить количество неразделяемых сигналов на *PHY*-интерфейс до 8.

Интерфейс *RMII* также способен поддерживать скорость передачи данных 10 Мбит/с и 100 Мбит/с, а также существуют варианты с поддержкой 1 гигабит. В *RMII* тактовая частота, используемая в PHY, постоянно работает на уровне 50 МГц для скоростей передачи данных как 10 Мбит/с, так и 100 Мбит/с. Это удвоение тактовой частоты при 100 Мбит/с также позволяет вдвое сократить количество сигналов для связи между *PHY* и *MAC*. Всего для связи требуется 9 сигналов, из которых до 3 могут быть разделены между несколькими *PHY*.

1.2.21 Структура и логика функционирования микросхемы *LAN8720*

LAN8720 – это приемопередатчик физического уровня (*PHY*) *10BASE-T/100BASE-TX* с низким энергопотреблением и переменным напряжением ввода-вывода, соответствующий стандартам *IEEE 802.3-2005*.

LAN8720 поддерживает связь с *Ethernet MAC* через стандартный интерфейс *RMII*. Он содержит полнодуплексный приемопередатчик *10-BASE-*

T/100BASE-TX и поддерживает работу на скоростях 10 Мбит/с (*10BASE-T*) и 100 Мбит/с (*100BASE-TX*). В LAN8720 реализовано автоматическое согласование для автоматического определения наилучшей возможной скорости и дуплексного режима работы. Поддержка *HP Auto-MDIX* позволяет использовать прямые или перекрестные кабели *LAN*. *LAN8720* поддерживает как функции регистрации, соответствующие стандарту *IEEE* 802.3-2005, так и функции регистрации, зависящие от поставщика. Однако для работы не требуется доступ к регистру. Выбираемые регистром параметры конфигурации могут использоваться для дальнейшего определения функциональности трансивера. В соответствии со стандартами *IEEE* 802.3-2005 все контакты цифрового интерфейса рассчитаны на напряжение 3,6 В. Устройство может быть настроено для работы от одного источника питания 3,3 В с использованием встроенного линейного регулятора от 3,3 В до 1,2 В. Линейный регулятор может быть опционально отключен, что позволяет использовать высокоэффективный внешний регулятор для снижения рассеиваемой мощности системы.

Структура *LAN8720* приведена на Рисунок 10 - Структурная схема *LAN8720*.

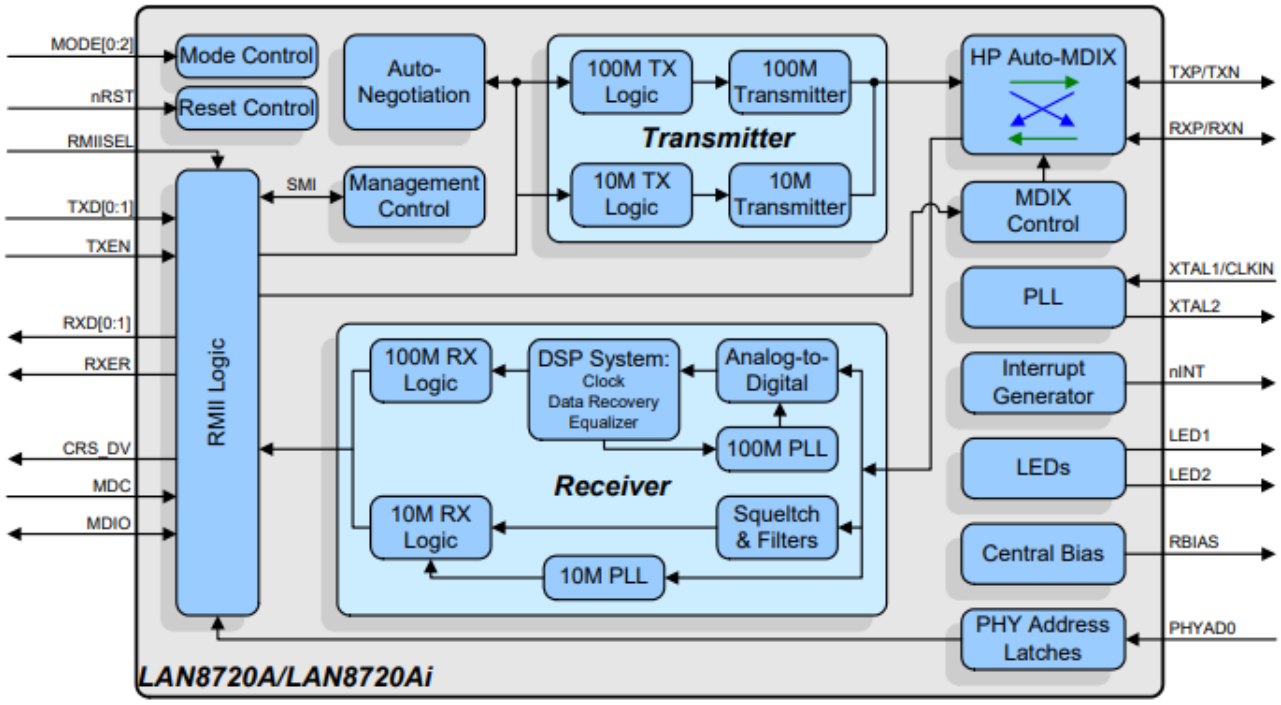

Рисунок 10 - Структурная схема *LAN8720*

Функции, которые может выполнять *LAN8720*, можно классифицировать следующим образом:

– трансивер;

– автосогласование;

– Поддержка HP *Auto-MDIX*;
– *MAC*-интерфейс;
– последовательный интерфейс управления (*SMI*);
– управление прерываниями;
– прочие функции.

Данное устройство можно использовать в качестве *MAC*-интерфейса.

Устройство поддерживает независимый от среды интерфейс (*RMII*) с малым количеством контактов. Интерфейс *RMII* имеет следующие характеристики:

– способен поддерживать скорость передачи данных 10 Мбит/с и 100 Мбит/с;
– один опорный тактовый сигнал используется как для передачи, так и для приема;
– обеспечивает независимые 2-битные пути передачи и приема данных;
– использует уровни сигналов *LVCMOS*, совместимые с обычными цифровыми процессами *CMOS ASIC*.

RMII включает следующие интерфейсные сигналы (1 опционально):
– передавать данные – *TXD*[1:0];
– передать строб – *TXEN*;
– получить данные – *RXD*[1:0];
– получить ошибку – *RXER* (необязательно);
– определение несущей – *CRS_DV*;
– опорный тактовый сигнал *REF_CLK*.

CRS_DV утверждается устройством, когда принимающая среда не простаивает. *CRS_DV* устанавливается асинхронно во время обнаружения несущей по критериям, относящимся к режиму работы. В режиме *10BASE-T*, когда шумоподавление пройдено, или в режиме *100BASE-X*, когда обнаруживаются 2 несмежных нуля в 10 битах, считается, что несущая обнаружена.

Потеря несущей должна привести к отмене подтверждения *CRS_DV* синхронно с циклом *REF_CLK*, который представляет первую пару битов полубайта на *RXD*[1:0].

1.2.22 Организация *LwIP*–стека

LwIP – это небольшая независимая реализация набора протоколов *TCP/IP*, первоначально разработанная Адамом Дункельсом. Целью реализации *lwIP* является сокращение использования ресурсов при сохранении полномасштабного *TCP*. Это делает *lwIP* пригодным для использования во встроенных системах с десятками килобайт свободной оперативной памяти и местом для примерно 40 килобайт кода.

LwIP поставляется со следующими протоколами:

– *IPv4* и *IPv6*, включая пересылку пакетов через несколько сетевых интерфейсов;

– *ICMP* для обслуживания и отладки сети;

– *IGMP* для управления многоадресным трафиком;

– *MLD* (обнаружение прослушивателя многоадресной рассылки для *IPv6*);

– *ND* (обнаружение соседей и автоматическая настройка адресов без сохранения состояния для *IPv6*);

– *DHCP* и *DHCPv6*;

– *UDP* (протокол пользовательских дейтаграмм);

– *TCP* (протокол управления передачей);

– *API* для повышения производительности;

– *TLS*: дополнительный многоуровневый *TCP* для почти прозрачного *TLS* для любого протокола на основе *TCP*;

– *PPPoS* и *PPPoE*;

– *DNS* (преобразователь доменных имен, включая *mDNS*);

– *6LoWPAN* – стандарт взаимодействия по протоколу *IPv6* поверх маломощных беспроводных персональных сетей.

Организация *LwIP* стека представлена ниже (Рисунок 11 - Организация *LwIP*).

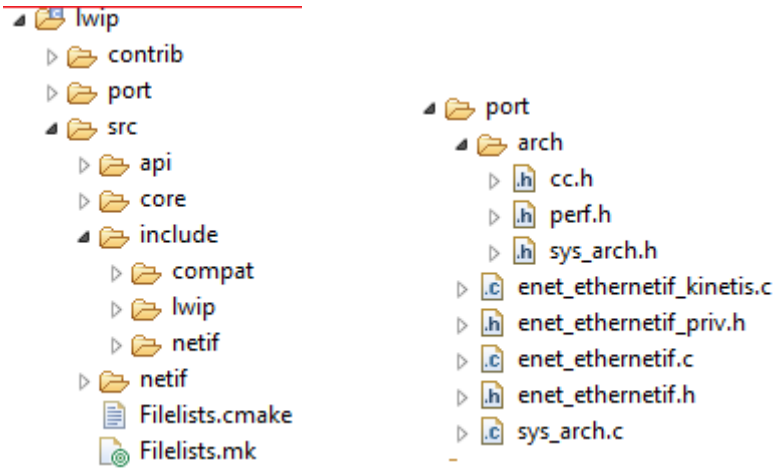

Рисунок 11 - Организация *LwIP*

Enet_ethernetif.c/h нужны для адаптации стека *LwIP* к базовому драйверу *Ethernet MCUXpresso SDK*, предоставляя интерфейсы *Ethernet phy*, *init* и *ethernetif_input.*

Cc.c/h представляет собой подсказки компилятора для упаковки и специфичные для платформы диагностические данные.

Perf.h нужен для измерения производительности для конкретной архитектуры.

Папка *src* содержит последний стабильный исходный код *LwIP*:

– *api* – файлы *netconn* и *API* сокетов;
– *core* – основные файлы *LwIP;*
– *include* – включаемые файлы *LwIP;*
– netif – файлы сетевого интерфейса.

LwIP предоставляет программистам три *API*, которые они могут использовать для связи с кодом TCP/IP

1 *Raw API*. Это управляемый событиями *API*, предназначенный для использования без операционной системы, реализующей отправку и получение без копирования. *Raw API* – это родной интерфейс *LwIP*. Он предусматривает использование обратных вызовов функций (*callbacks*) внутри стека. Это означает, что вам перед началом работы со стеком необходимо присвоить указатели на функции-обработчики событий, которые в процессе работы будут вызываться внутри *LwIP*. *Raw API* имеет наибольшую производительность и наименьший результирующий размер кода.

2 *Netconn API* – высокоуровневый последовательный интерфейс, построенный поверх *Raw API*. Этот интерфейс требует наличия *RTOS* и поддерживает мнопоточные операции.

3 *BSD Socket API* – высокоуровневый интерфейс сокетов, разработанный поверх *Netconn API*. Этот интерфейс обеспечивает высокую переносимость ваших приложений, поскольку является стандартизированным *API*.

LwIP получил очень широкое распространение во встраиваемых системах на базе микроконтроллеров благодаря низкому потреблению оперативной памяти. Именно этот *TCP/IP* стек используется в фреймворке *ARM mbed* и генераторе кода инициализации *STM32CubeMX*.

1.2.23 Стандарты сжатия данных в *IP*-видеонаблюдении

Ранее были введены такие понятия, как пиксель и массив пикселей (кадр). К слову, на данный момент в системах видеонаблюдения самые распространённые размеры кадров: 960x576 (*WD1*), 1280x720 (*HD*), 19 20x1080 (*FullHD*), 2688x1520 (*4Mpix*) и 2560x1920 (5 *Mpix*).

Когда кадры чередуются с некоторой частотой, получается видео. Частота, при которой человеческий глаз воспринимает такой видеопоток плавным – 24 кадра в секунду.

Также видеопоток характеризуется битрейтом – количеством бит информации, которое используется для хранения или для передачи аудио или видео. Битрейт может быть как постоянным, так и переменным. Постоянный битрейт соответствует заданным параметрам и остаётся неизменным на протяжении всего файла. Его главное достоинство в том, что можно предсказать размер конечного файла. При переменном битрейте кодек выбирает его значение, исходя из параметров желаемого качества. В течение всего кодируемого видеофрагмента битрейт может изменяться.

Итак, зачем сжимать видео? Если, например, взять видеопоток с разреением 1920x1080 пикселей со скоростью 2 кадра в секунду, а также каждый пиксель кодировать в RGB24 (то есть 24 бита на пиксель), то на один кадр уйдёт 47.5 Мбит/с пропускной способности канала, а на 24 таких кадра – 1140 Мбит/с. В свою очередь такая же последовательность кадров, но сжатая в соответствии со стандартом *H*.264 будет занимать в среднем в 150 раз меньше пропускной способности канала, а также места в хранилище.

В контексте видеонаблюдения имеет смысл рассматривать поколение кодеков семейства *H*, так как они нацелены на уменьшение потока цифрового видео по сети.

На данный момент в системах видеонаблюдения долгое время доминирует алгоритм сжатия *H*.264., который заключается в исключении избыточных данных и сокращении их объема по различным алгоритмам.

При настройке кодирования в системах видеонаблюдения встречаются три основных профиля кодека *H*.264:

– *Baseline* профиль, который подразумевает минимальную нагрузку на процессор декодирующего устройства при несильном сжатии. Предназначен для просмотра видео с видеокамеры на внешнем сервере.

– *Main* профиль, который создаёт среднюю нагрузку на процессор при сильном сжатии.

– *High* профиль обеспечивает максимальное сжатие с сильной нагрузкой на устройство декодирования. Битрейт при работе с таким профилем будет в 2-3 раза ниже, чем при использовании *baseline* профиля.

Формат сжатия *H.265 High Efficiency Video Coding (HEVC)* стал значительным шагом вперед в области кодирования цифрового видеосигнала, главным преимуществом которого является почти в 2 раза увеличенная эффективность по сравнению с предшествующим стандартом H.264. То есть благодаря новому алгоритму для передачи сигнала требуется вдвое меньшая пропускная способность сети, а для хранения вдвое меньшая ёмкость накопителей.

Параллельное кодирование, предусмотренное стандартом H.265, даёт возможность одновременной обработки разных частей кадра, что существенно ускоряет воспроизведение и даёт возможность в полной мере использовать свойство новых процессоров – многоядерность.

Кроме этого, новый стандарт получил технологию произвольного доступа к изображению (Clean Random Access), которая позволяет произвести декодирование случайно выбранного кадра без необходимости обработки предыдущих в потоке изображений. Это особенно желательно, когда при мониторинге требуется оперативно переключиться на определённый канал.

Несмотря на все преимущества, H.265 ещё далёк от повсеместного использования. Во-первых, из-за того, что для его использования необходима обновлённая аппаратная часть, во-вторых, чтобы использовать кодек необходима покупка патента.

Кроме двух вышеприведённых кодеков, существует *H*.264+, который был разработан специально под нужны видеонаблюдения.

На видео, полученном с охранных видеокамер, сцена всегда постоянна и практически не изменяется, представляющие интерес подвижные объекты могут отсутствовать на протяжении длительного времени, а шумы, возникающие в плохих условиях освещения, ощутимо влияют на качество изображения. В обновлённом формате все эти особенности были учтены и обрабатываются следующими технологиями, повышающими степень сжатия:

– кодирование с предсказанием на основе модели фона;

– шумоподавление;

– долгосрочное управление видеопотоком.

Кодирование с предсказанием используется тогда, когда фон в видеонаблюдении стабилен, то этот фон лучше всего использовать в качестве опорного кадра, тем самым повысить эффективность сжатия неподвижных объектов и снизить поток данных, приходящийся на опорные кадры. Интеллектуальный алгоритм предсказания выбирает опорные кадры среди тех, в которых меньше всего движущихся объектов.

Шумоподавление. В формате H.264+ с помощью специальных алгоритмов фон отделяется от движущегося объекта и кодируется с более высокой степенью сжатия. Такая технология позволяет частично подавлять шумы и уменьшать битрейт.

Долгосрочное управление видеопотоком. Формат H.264+ имеет алгоритмы отслеживания интенсивности видеопотоков и в зависимости от времени суток автоматически изменяет степень сжатия. Такая технология управления видеопотоком позволяет не только уменьшить объём видеоархива, но и сохранить качество изображения движущихся объектов.

1.2.24 Технология PoE

//TODO

1.2.25 Принципиальные основы и схемы зарядки литий–ионных аккумуляторных батарей

//TODO

1.2.26 Структура и логика функционирования микросхем LTC4058 и BQ24295 зарядки литий-ионных аккумуляторных батарей

//TODO

2 РАЗРАБОТКА СТРУКТУРНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ IP–ВИДЕОКАМЕРЫ

2.1 Обоснование базовых блоков структурной схемы IP–видеокамеры

//TODO

2.2 Обоснование связей структурной схемы IP–видеокамеры

//TODO

3 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ IP–ВИДЕОКАМЕРЫ

3.1 Обоснование выбора САПР для разработки принципиальной электрической схемы

//TODO

3.2 Описание используемых библиотечных элементов и процесса их создания

//TODO

3.3 Обоснование выбора базовых компонентов принципиальной схемы IP–видеокамеры

//TODO

3.4 Обоснование связей принципиальной электрической схемы IP–видеокамеры

//TODO

3.5 Анализ и обоснование принципиальной электрической схемы зарядки аккумуляторной батареи

//TODO

4 РАЗРАБОТКА МОДЕЛИ И АЛГОРИТМА ФУНКЦИОНИРОВАНИЯ IP–ВИДЕОКАМЕРЫ

4.1 Обоснование модели IP–видеокамеры в среде Proteus

//TODO

4.2 Разработка диаграммы состояний IP–видеокамеры

//TODO

4.3 Разработка схемы алгоритма функционирования IP–видеокамеры

//TODO

4.4 Моделирование потока видео образов в среде Proteus

//TODO

4.5 Реализация интерфейса Ethernet в среде Proteus

//TODO

4.6 Алгоритм реализации стандарта сжатия H.264 в контексте LWIP–стека

//TODO

4.7 Программная реализация и отладка в среде Proteus алгоритмов функционирования IP–видеокамеры

//TODO

5 РАЗРАБОТКА КОНСТРУКЦИИ ПРОЕКТИРУЕМОГО ПРИБОРА

5.1 Выбор и обоснование элементной базы

//TODO

5.2 Выбор и обоснование конструктивных элементов и установочных изделий

//TODO

6 РАСЧЁТ КОНСТРУКТИВНО–ТЕХНОЛОГИЧЕСКИХ ПАРАМЕТРОВ ПРОЕКТИРУЕМОГО ПРИБОРА

6.1 Проектирование печатного модуля

6.1.1 Выбор типа конструкции печатной платы, класса точности и шага координатной сетки

//TODO

6.1.2 Выбор и обоснование метода изготовления электронного модуля

//TODO

6.1.1 Расчёт конструктивно–технологических параметров электронного модуля (определение габаритных размеров, выбор толщины печатной платы, определение элементов проводящего рисунка)

//TODO

6.2 Выбор и обоснование материалов конструкции и защитных покрытий, маркировки деталей и сборочных единиц

//TODO

7 ПРИМЕНЕНИЕ СРЕДСТВ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ПРИ РАЗРАБОТКЕ ПРИБОРЫ

//TODO

ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта была спроектирована база данных, которая содержит в себе 13 сущностей. Каждая сущность имеет первичный ключ, данные нормализованы, транзитивные отношения отсутствуют, таким образом база данных приведена к третьей нормальной форме. В качестве СУБД был выбран *MySQL*, настройка базы данных проводилась в *MySQL Workbench* 8.0. В процессе разработки базы данных были учтены особенности, которые перечислены в задании: база данных должна отвечать требованиям надёжности, минимальной избыточности, целостности данных, содержать не менее десяти сущностей, а ее схема должна быть приведена к третьей нормальной форме. База данных должна поддерживать основные современные средства для работы и администрирования.

Были реализованы следующие объекты базы данных в количестве не менее трех каждый: индекс, триггер, хранимая процедура. Также была реализована система разграничения прав доступа к данным минимум для двух ролей пользователей.

Что касается веб–приложения, то приложение разработано на языке *JavaScript* с использованием фреймворка *Express* для создания *API*, а также с использованием *React* и *Axios* для создания пользовательского интерфейса. В приложении реализована система условного рендеринга, которая основывается на разграничении прав доступа к базе данных. Интерфейс приложения разработан удобным и интуитивно понятным.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 96,48%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении А.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Понятие изображения
https://whatis.techtarget.com/definition/image

Представление изображений
https://www.csunplugged.org/en/topics/image–representation/

цветовые модели
https://www.pantone.com/articles/color–fundamentals/color–models–explained

виды ргб
http://photoschool01.ru/8–or–16–bit/

потоковое видео
https://www.cdnetworks.com/media–delivery–blog/video–streaming–protocols/

камеры
https://eltechbook.ru/kamery_videonabljudenie.html

ethernet
http://www.highteck.net/EN/Ethernet/Ethernet.html

tcp/ip
https://isaaccomputerscience.org/concepts/net_internet_tcp_ip_stack?examBoard=all&stage=all
https://selectel.ru/blog/tcp-ip-for-beginners/

SoC
https://anysilicon.com/what-is-a-system-on-chip-soc/
SoC in ip cameras
https://www.unifore.net/ip-video-surveillance/ip-camera-dvr-nvrs-soc-whats-it-what-it-does.html
Popular SoC manufacturers
https://www.unifore.net/ip-video-surveillance/list-of-popular-soc-processors-for-ip-security-cameras.html Architectures
https://www.mathworks.com/discovery/soc-architecture.html

структура ARM Cortex-M4
https://microcontrollerslab.com/arm-cortex-m4-architecture/

cortex-m4 registers
https://www.ics.com/blog/introduction-gpio-programming
https://www.linkedin.com/pulse/all-you-need-know-gpio-akib-islam

уровни i2c
https://blog.actorsfit.com/a?ID=01750-fa7b01f9-36e1-4b59-b44f-0c32b947d709

Регистровая модель i2c
https://controllerstech.com/stm32-i2c-configuration-using-registers/

dcmi
https://www.st.com/resource/en/application_note/an5020-digital-camera-interface-dcmi-on-stm32-mcus-stmicroelectronics.pdf

регистровая модель dcmi
https://habr.com/ru/post/186980/
https://www.compel.ru/lib/125145

how dma works
https://www.minitool.com/lib/direct-memory-access.html

dma interrupts
https://www.eventhelix.com/fault-handling/dma-interrupt-handling/
https://www.brainkart.com/article/DMA-and-Interrupts_8636/

ov9655
https://www.arducam.com/downloads/modules/OV9655/ov9655_full.pdf

физический и логический уровни mii
физические и логический уровни rmii
https://resources.pcb.cadence.com/blog/2019-mii-and-rmii-routing-guidelines-for-ethernet

lan8720
http://ww1.microchip.com/downloads/en/devicedoc/00002165b.pdf

lwip
https://community.nxp.com/t5/Kinetis-Software-Development-Kit/Brief-Introduction-to-LwIP-stack-in-MCUXpresso-SDK/ta-p/1108707
https://cxemotexnika.org/2018/09/vvedenie-v-lightweight-iplwip-stek-protokolov-tcp-ip/

[1] Проектирование БД [Электронный ресурс]. – Режим доступа: https://www.lucidchart.com/pages/database–diagram/database–design. – Дата доступа: 15.11.2021.

[2] Типы данных *MySQL* [Электронный ресурс]. – Режим доступа: https://www.w3schools.com/sql/sql_datatypes.asp. – Дата доступа: 15.11.2021.

[3] Нормализация отношений [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/254773/. – Дата доступа: 21.11.2021.

[4] Первичный ключ [Электронный ресурс]. – Режим доступа: https://www.techopedia.com/definition/5547/primary–key. – Дата доступа: 17.11.2021.

[5] Запросы *MySQL* [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/mysql–common–mysql–queries/. – Дата доступа: 20.11.2021.

[6] *JavaScript* [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en–US/docs/Web/JavaScript/Guide/Introduction. – Дата доступа: 23.11.2021.

[7] *VS Code* [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com/docs/editor/whyvscode. – Дата доступа: 25.11.2021.

[8] *DB documentation* [Электронный ресурс]. – Режим доступа: https://www.holistics.io/blog/top–5–database–documentation–tools–for–any–teams–in–2020/. – Дата доступа: 28.11.2021.

[9] Руководство пользователя [Электронный ресурс]. – Режим доступа: https://zenkit.com/en/blog/how–to–write–a–user–manual/. – Дата доступа: 01.12.2021.

[10] Администрирование [Электронный ресурс]. – Режим доступа: https://www.techopedia.com/definition/24080/database–administration. – Дата доступа: 03.12.2021.

[11] Пользователи и роли [Электронный ресурс]. – Режим доступа: https://satoricyber.com/sql–server–security/sql–server–roles/. – Дата доступа: 07.12.2021.

[12] *MySQL Safety* [Электронный ресурс]. – Режим доступа: https://www.mysql.com/why–mysql/presentations/mysql–security–best–practices/. – Дата доступа: 08.12.2021.

[13] Безопасность БД [Электронный ресурс]. – Режим доступа: https://www.tripwire.com/state–of–security/featured/database–security–best–practices–you–should–know/. – Дата доступа: 09.12.2021.

[14] *Bcrypt* [Электронный ресурс]. – Режим доступа: https://auth0.com/blog/hashing–in–action–understanding–bcrypt/. – Дата доступа: 09.12.2021.

[15] *Cookie* [Электронный ресурс]. – Режим доступа: https://www.kaspersky.com/resource–center/definitions/cookies. – Дата доступа: 10.12.2021.

[16] *HTTP Cookie* [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/HTTP/Cookies. – Дата доступа: 11.12.2021.

ПРИЛОЖЕНИЕ А
(обязательное)
Проверка на оригинальность

Рисунок А.1 – Проверка на оригинальность

ПРИЛОЖЕНИЕ Б
(обязательное)
Перечень элементов

ПРИЛОЖЕНИЕ В
(обязательное)
Спецификация

ПРИЛОЖЕНИЕ Г
(обязательное)
Визуализированная трёхмерная модель

ПРИЛОЖЕНИЕ Д
(обязательное)
Листинг программы

ПРИЛОЖЕНИЕ Е
(обязательное)
Ведомость документов