UNIVERSITATEA DE STAT DIN MOLDOVA

FACULTATEA DE FIZICĂ ȘI INGINERIE

DEPARTAMENTUL FIZICA TEORETICĂ “IU. PERLIN”

ABDULLAEV RAMIL

DEZVOLTAREA GENERATORULUI DE SEMNAL VIDEO DE TESTARE: CONTROLUL PLACII FOLOSIND MICROCONTROLERUL AtXmega128A1

061 TEHNOLOGII ALE INFORMAȚIEI ȘI COMUNICAȚIILOR

TEHNOLOGII INFORMAȚIONALE ÎN MODELARE

Teză de master

Șef departament **\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DMITROGLO Liliana, dr., conf.univ.**

Conducător ştiinţific **\_\_\_\_\_\_\_\_\_\_\_\_\_\_ NICA Denis, dr.hab., conf.univ.**

Autorul \_\_\_\_\_\_\_\_\_\_\_\_\_\_ **ABDULLAEV Ramil**

CHIȘINĂU-2023

МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ФИЗИКИ И ИНЖИНЕРИИ

ДЕПАРТАМЕНТ ТЕОРЕТИЧЕСКОЙ ФИЗИКИ ИМ. Ю. ПЕРЛИНА

АБДУЛЛАЕВ РАМИЛ

РАЗРАБОТКА ГЕНЕРАТОРА ТЕСТОВЫХ ВИДЕОСИГНАЛОВ:УПРАВЛЕНИЕ ПЛАТОЙ С ПОМОЩЬЮ МИКРОКОНТРОЛЛЕРА AtXmega128A1

061 ТЕХНОЛОГИИ ИНФОРМАЦИИ И КОММУНИКАЦИЙ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В МОДЕЛИРОВАНИИ

Магистерская диссертация

Зав. департаментом: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лилиана ДМИТРОГЛО, др., доцент**

Научный руководитель: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_Денис НИКА, др. хаб., доцент**

Автор \_\_\_\_\_\_\_\_\_\_\_\_\_\_**Рамил АБДУЛЛАЕВ**

**КИШИНЕВ-2023**

СОДЕРЖАНИЕ

[СПИСОК АББРЕВИАТУР 4](#_Toc137038907)

[АННОТАЦИЯ 5](#_Toc137038908)

[ADNOTARE 6](#_Toc137038909)

[ВВЕДЕНИЕ 7](#_Toc137038910)

[Глава I. ОБЗОР ТЕХНИЧЕСКИХ ВОЗМОЖНОСТЕЙ ПОСТРОЕНИЯ СОВРЕМЕННЫХ СИСТЕМ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ И МИКРОСХЕММ ПРОГРАММИРУЕМОЙ ЛОГИКИ 8](#_Toc137038911)

[1.1 Краткий обзор различных архитектур микропроцессоров 8](#_Toc137038912)

[1.2 Особенности работы микроконтроллеров 11](#_Toc137038913)

[1.3 Микроконтроллеры семейства AtXmega 12](#_Toc137038914)

[1.4 Среда разработки IAR 5.3 15](#_Toc137038915)

[1.5 Способы передачи данных между микросхемами. 18](#_Toc137038916)

[1.6 Методы загрузки микросхем FPGA 26](#_Toc137038917)

[1.7 Заключение 27](#_Toc137038918)

[Глава II. ПОСТАНОВКА ЗАДАЧИ И АНАЛИЗ МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННЫХ ЗАДАЧ 28](#_Toc137038919)

[2.1 Постановка задачи 28](#_Toc137038920)

[2.2 Описание выбранного способа загрузки FPGA 28](#_Toc137038921)

[2.3 Краткое описание возможностей микросхемы HDMI выхода AD9889B 30](#_Toc137038922)

[2.4 Конфигурация микросхемы AD9889B 31](#_Toc137038923)

[2.5 Использование роторного переключателя 32](#_Toc137038924)

[2.6 Заключение 34](#_Toc137038925)

[Глава III. РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ГЕНЕРАТОРА ТЕСТОВЫХ ВИДЕОСИГНАЛОВ 36](#_Toc137038926)

[3.1 Принципиальная схема 36](#_Toc137038927)

[3.2 Инициализация портов ввода-вывода 37](#_Toc137038928)

[3.2.1 Макросы настройки портов ввода-вывода. 37](#_Toc137038929)

[3.2.2 Начальная инициализация портов ввода вывода 39](#_Toc137038930)

[3.2.3 Загрузка FPGA 40](#_Toc137038931)

[3.2.4 Инициализация микросхемы HDMI выхода AD9889B 41](#_Toc137038932)

[3.2.5 Отработка прерывания по таймеру 43](#_Toc137038933)

[3.2.6 Работа основной части программы 45](#_Toc137038934)

[3.2.7 Обновление регистров выбранного видеосигнала 46](#_Toc137038935)

[3.3 Заключение 47](#_Toc137038936)

[ВЫВОДЫ И РЕКОМЕНДАЦИИ 48](#_Toc137038937)

[БИБЛИОГРАФИЯ 50](#_Toc137038938)

[ДЕКЛАРАЦИЯ ОБ ОТВЕТСТВЕННОСТИ 51](#_Toc137038939)

СПИСОК АББРЕВИАТУР

ARM – Advanced RISC (Reduced Instruction Set Computer) Machine

AVR – Alf and Vegard's RISC processor

EEPROM – Electrically Erasable Programmable Read-Only Memory

FPGA **-**  Field-Programmable Gate Array

SPI **-**  Serial Peripheral Interface

I2C – Philips two wire interface

ПО – Программное обеспечение

АННОТАЦИЯ

Абдуллаев Рамил, Разработка Генератора тестовых видеосигналов: программа микроконтроллера, Кишинев, 2023. Введение, три главы, выводы и рекомендации, библиография из 6 наименований, 48 страниц, 12 листингов, 12 рисунков.

**Ключевые слова**

AVR, микроконтроллер, FPGA, схемотехника, роторный переключатель.

**Цели и задачи работы**

Целью работы является разработка устройства генерации тестовых видео изображений, основанного микросхеме программируемой логики, микроконтроллере фирмы Atmel, а также микросхеме физического уровня HDMI. Для осуществления поставленной цели в работе необходимо решить следующие задачи: разработка электрической принципиальной схемы, сборка конструктива и реализация программного обеспечения.

**Теоретическая и прикладная значимость**

В процессе выполнения работы были изучены способы загрузки микросхем программируемой логики. Кроме этого были получен опыт в реализации систем управления посредством роторного переключателя.

ADNOTARE

Abdullaev Ramil, Elaborarea hardware și software a unui dispozitiv dispozitiv de iluminat decorativ, Chișinău, 2023. Introducere, trei capitole, concluzii și recomandări, bibliografie din 6 titluri, 48 pagini, 12 listări, 12 figuri.

**Cuvintele-cheie**

AVR, microcontroler, FPGA, schemotehnică, comutator rotativ.

**Scopul si obiectivele cercetarii**

Scopul proiectului este dezvoltarea unui dispozitiv pentru generarea imaginilor video de test, bazat pe un cip logic programabil, un microcontroler de la firma Atmel și un cip HDMI de nivel fizic. Pentru atingerea acestui scop, sunt necesare rezolvarea următoarelor sarcini: dezvoltarea schemei electrice de principiu, asamblarea dispozitivului și implementarea software-ului corespunzător.

**Valoarea teoretică şi aplicativă**

În timpul desfășurării lucrării, au fost studiate metodele de încărcare a cipurilor logice programabile. De asemenea, a fost obținută experiență în implementarea sistemelor de control prin intermediul unui comutator rotativ.

ВВЕДЕНИЕ

В данной работе были рассмотрены компоненты и подходы к созданию устройств генерации цифровых видеоизображений. Представленное устройство функционирует на базе микросхемы семейства Altera(Intel) Cyclone IV. Микросхемы данного семейства, хоть и выпускаются более пяти лет, все ещё производятся и актуальны для решения многих задач. Возможности микросхемы данного семейства позволили решить поставленную задачу.

В ходе работы были проанализированы различные архитектура микросхем программируемой логики и способы их программирования. В данном проекте предпочтение было отдано графическому методу проектирования. Данный метод показался более наглядным и понятным при решении задач генерации видео изображений.

В результате работы был сконструирован генератор видео изображений в формате 1920\*1080@50, управляемый микроконтроллером(AtxMega128) на базе ПЛИС Altera(Intel) Cyclone IV и использующий видео интерфейс HDMI в качестве выходного сигнала. Готовый генератор оправдал ожидания – полученные изображения соответствуют рекомендациям из свободным источников, а реализованные видео изображения позволяют оценить работу тестируемого монитора.

На базе полученных результатов можно c уверенностью сказать что микросхемы ПЛИС подходят для проектирования устройств генерации статических и динамических видео таблиц для тестирования мониторов и телевизоров.

Результаты данной работы могут послужить обогащению знаний в проектировании устройств генерации тестовых видео изображений, программирования микроконтроллеров, а также систем обработки несжатого видео.

*Ключевые слова:* AVR, микроконтроллер, FPGA, схемотехника.

Глава I. ОБЗОР ТЕХНИЧЕСКИХ ВОЗМОЖНОСТЕЙ ПОСТРОЕНИЯ СОВРЕМЕННЫХ СИСТЕМ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ И МИКРОСХЕММ ПРОГРАММИРУЕМОЙ ЛОГИКИ

Понятие устройств генерации тестовых изображений

Устройства генерации тестовых изображений, построенные на базе микросхем ПЛИС, в настоящее время получили широкое распространение в различных сферах. Технологии производства ПЛИС позволяют строить на их базе различные устройства высокоскоростной обработки несжатого видео.

Одной из направлений применения ПЛИС является создание устройств генерации тестовых видео изображений. Такие устройства лишены недостатков связанных с декомпрессией видео так как значения пикселей генерируются в режиме реального времени.

Для полноценной работы ПЛИС необходимо осуществить процедуру начальной загрузки и дальнейшее управление. Для этих целей в данной работе используется микроконтроллер фирмы Atmel семейства AtXmega. В этой главе будут рассмотренны возможности микроконтроллер, а также способы реализации поставленных задач.

* 1. Краткий обзор различных архитектур микропроцессоров

Архитектура, то есть логическая организациямикропроцессора, однозначно определяет свойства, особенности и возможности построения вычислительной системы на базе данного микропроцессора.[1]

Большинство современных микропроцессоров при всем разнообразии их типов, моделей и производителей имеют одну из трех типов архитектуры: CISC, RISC и MISC (это относится к микропроцессорам универсального, а не специального применения).

**Архитектура CISC** (Complex Instruction Set Computer) - концепция проектирования процессоров, которая характеризуется следующим набором свойств:

1. большим числом различных по формату и длине команд;
2. использованием различных систем адресации;
3. обладает сложной кодировкой инструкции.

Процессору с архитектурой CISC приходится иметь дело с более сложными инструкциями неодинаковой длины. Выполнение одиночной CISC-инструкции может происходить быстрее, однако обрабатывать несколько таких инструкций параллельно сложнее.

Облегчение отладки программ на ассемблере влечет за собой загромождение узлами микропроцессорного блока. Для повышения быстродействия следует увеличить тактовую частоту и степень интеграции, что вызывает необходимость совершенствования технологии и, как следствие, более дорогого производства.

**Достоинства архитектуры CISC**

1. Компактность наборов инструкций уменьшает размер программ и уменьшает количество обращений к памяти.
2. Наборы инструкций включают поддержку конструкций высокоуровневого программирования.

**Недостатки архитектуры CISC**

1. Нерегулярность потока команд.
2. Высокая стоимость аппаратной части.
3. Сложности с распараллеливанием вычислений.

**Архитектура RISC** (Reduced Instruction Set Computer) - Процессор с сокращенным набором команд. Система команд имеет упрощенный вид. Все команды одинакового формата с простой кодировкой. Обращение к памяти происходит посредством команд загрузки и записи, остальные команды типа регистр-регистр. Команда, поступающая в CPU, уже разделена по полям и не требует дополнительной дешифрации.

Часть кристалла освобождается для включения дополнительных компонентов. Степень интеграции ниже, чем в предыдущем архитектурном варианте, поэтому при высоком быстродействии допускается более низкая тактовая частота. Команда меньше загромождает ОЗУ, CPU дешевле. Программной совместимостью указанные архитектуры не обладают. Отладка программ на RISC более сложна. Данная технология может быть реализована программно-совместимым с технологией CISC (например, суперскалярная технология).

Поскольку RISC-инструкции просты, для их выполнения нужно меньше логических элементов, что в конечном итоге снижает стоимость процессора. Но большая часть программного обеспечения сегодня написана и откомпилирована специально для CISC-процессоров фирмы Intel. Для использования архитектуры RISC нынешние программы должны быть перекомпилированы, а иногда и переписаны заново.

**Достоинства архитектуры RISC**

1. Снижение нерегулярности потока команд
2. Обогащение пространственным параллелизмом

**Недостатки архитектуры RISC**

1. Каждое действие выполняется в 1 такт.

**Архитектура MISC** (Multipurpose Instruction Set Computer) - многоцелевая командная система управления компьютером, сочетает в себе преимущества CISC и RISC. Элементная база состоит из отдельных частей (могут быть объединены в одном корпусе): основная часть (HOST - ведущая), архитектуры RISC CPU, а расширяемая часть - с подключением ПЗУ (ROM) микропрограммного управления. При этом вычислительная система приобретает свойства CISC: - основные команды работают на HOST, а команды расширения образуют адрес микропрограммы для своего выполнения. HOST выполняет команды за один такт, а расширение эквивалентно CPU со сложным набором команд (CISC). Наличие ПЗУ устраняет недостаток RISC, связанный с тем, что при компиляции с языка высокого уровня код операции (микропрограмма) уже дешифрирована и открыта для программиста.

Данный микроконтроллер построен на RISC архитектуре. Более подробно особенности выбранной аппаратной платформы будут рассмотрены ниже.

* 1. Особенности работы микроконтроллеров

Современные микроконтроллеры (МК) относятся к полупроводниковым компонентам, основное назначение которых – управление электронными устройствами. Они встраиваются непосредственно в исполнительный модуль, задавая режимы работы обслуживаемых изделий. На их основе реализована оригинальная идея, позволяющая совместить в одном кристалле сразу несколько функций.

Подобно классическому компьютеру они содержат в своем составе не только центральный процессор. Помимо него в МК входят рабочая память, ПЗУ, а также элементы периферии, объединенные в похожем на микросхему корпусе.

**Что такое микроконтроллеры**

Современный микроконтроллер, можно сравнить с основным элементом любого компьютерного устройства – микропроцессором. Последний представляет собой центральную часть ПК, выполненную на основе интегральных схем. Название «микропроцессор» (МП) указывает на то, что именно в этом модуле происходят все вычислительные операции, определяющие производительность системы в целом.

Чтобы получить на его основе полноценную ЭВМ – необходимо дополнить МП целым рядом внешних устройств. Добиться этого удается добавлением к этому модулю оперативной памяти и периферийных портов ввода-вывода информации. В результате получается законченное автономное устройство или компьютер.

Подобно ПК микроконтроллеры содержат в своем составе следующие компоненты:

* Микропроцессор.
* Оперативную память.
* Комплект периферии, состоящей из портов обмена информацией с обслуживаемым устройством.

В отличие от ПК он не является полностью самостоятельным (автономным) изделием. Микроконтроллеры, как правило, встраиваются в гаджеты, игрушки и в другие исполнительные устройства, функционированием которых они управляют.

Отличия микроконтроллера от микропроцессора

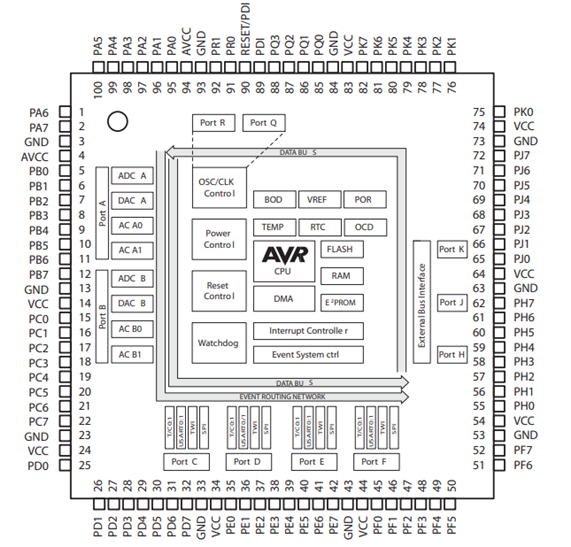
  Разница между микропроцессором и микроконтроллером заключается в наличии оперативной памяти, ПЗУ и других периферийных устройств в микроконтроллере. Микропроцессор содержит только процессор и не имеет других компонентов.

Микропроцессор и микроконтроллер, оба являются основными процессорами, предназначенными для работы компьютеров. Функции обоих процессоров одинаковы. Основное различие между ними состоит в том, что микропроцессоры выполняют различные функции, тогда как микроконтроллеры - это небольшие компьютеры, предназначенные для конкретных задач.

* 1. Микроконтроллеры семейства AtXmega

Микроконтроллеры **AVR XMEGA** выполнены на основе: Технологии picoPower второго поколения Инновационной системы обработки событий "Event System", которая обеспечивает независимую от ЦПУ быстродействующую передачу данных между внутренними периферийными устройствами 4-канального контроллера ПДП, улучшающего характеристики микроконтроллера Быстродействующих 12-битных АЦП и ЦАП Ускорителя криптографических алгоритмов AES и DES  
      Микроконтроллеры AVR XMEGA производят такое же ошеломляющее впечатление, которое производили их предшественники микроконтроллеры AVR после своего появления. Новые микроконтроллеры способны работать при напряжении питания всего лишь 1.6 В и достигать производительности 32 MIPS на тактовой частоте 32 МГц.

Микроконтроллеры содержат флэш-память размером 16…384 кбайт и поставляются в 44…100-выводных корпусах. Микроконтроллеры XMEGA являются микроконтроллерами общего назначения и могут использоваться в широком числе применений, в т.ч. аудиосистемы, ZigBee® системы, медицинская техника, контроллеры автоматизации, коммуникационное оборудование, измерительные приборы, оптические трансиверы, системы управления электроприводами, бытовое электрооборудование, оптические трансиверы и любая другая продукция с батарейным питанием.



**Рис. 1.1.** Структурная схема и система ввода вывода микроконтроллера AtXmega128.

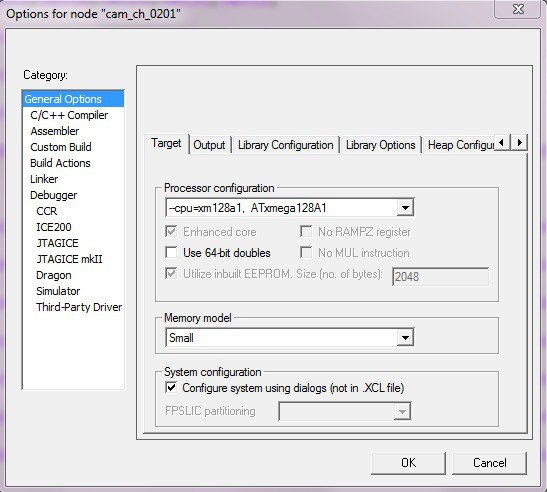
Список основных возможностей микроконтроллеров семейства AVR xMega:

* Энергонезависимая память программ и данных
* 64–128 КБ внутрисистемной самопрограммируемой флэш-памяти
* 4K - 8Кб загрузочный раздел(область Boot Loader)
* 2 КБ EEPROM
* 4 КБ - 8 КБ внутренней SRAM
* Интерфейс внешней шины для SRAM до 16 Мбайт
* Интерфейс внешней шины для SDRAM до 128 Мбит
* Периферийные функции
  + Четырехканальный контроллер прямого доступа к памяти
  + Восьмиканальная система событий
  + Восемь 16-битных таймеров/счетчиков
    - Четыре таймера/счетчика с 4 выходными каналами сравнения или входными каналами захвата
    - Четыре таймера/счетчика с 2 выходными каналами сравнения или входными каналами захвата
    - Расширение высокого разрешения для всех таймеров/счетчиков
    - Усовершенствованное расширение сигнала (AWeX) на двух таймерах/счетчиках
  + Восемь USART с поддержкой IrDA для одного USART
  + Четыре двухпроводных интерфейса с двойным согласованием адресов (I2Совместимость с C и SMBus)
  + Четыре последовательных периферийных интерфейса (SPI)
  + Криптовалютный движок AES и DES
  + 16-битный счетчик реального времени (RTC) с отдельным генератором
  + Два шестнадцатиканальных 12-битных аналого-цифровых преобразователя со скоростью 2 мс
  + Два двухканальных 12-разрядных цифро-аналоговых преобразователя со скоростью 1 мс/с
  + Четыре аналоговых компаратора (AC) с функцией сравнения окна и источниками тока
  + Внешние прерывания на всех контактах ввода/вывода общего назначения
  + Программируемый сторожевой таймер с отдельным встроенным сверхмаломощным генератором
  + Поддержка библиотеки QTouch®
* Емкостные сенсорные кнопки, ползунки и колеса
* Специальные функции микроконтроллера
  + Сброс при включении питания и программируемое обнаружение отключения питания
  + Внутренние и внешние часы с PLL и предварительным делителем
  + Программируемый многоуровневый контроллер прерываний
  + Пять режимов сна
  + Интерфейсы программирования и отладки
    - Интерфейс JTAG (совместимый с IEEE 1149.1), включая сканирование границ
    - PDI (программный и отладочный интерфейс)
* Рабочее напряжение: 1,6 – 3,6 В
* Рабочая частота:
  + 0–12 МГц от 1,6 В
  + 0–32 МГц от 2,7 В
  1. Среда разработки IAR 5.3

Среда разработки IAR 5.3 - устаревшая версия среды разработки от компании IAR Systems, выпущенная более десяти лет назад. Тем не менее, она всё ещё может использоваться для разработки микроконтроллерных приложений на платформах, которые поддерживают эту версию.

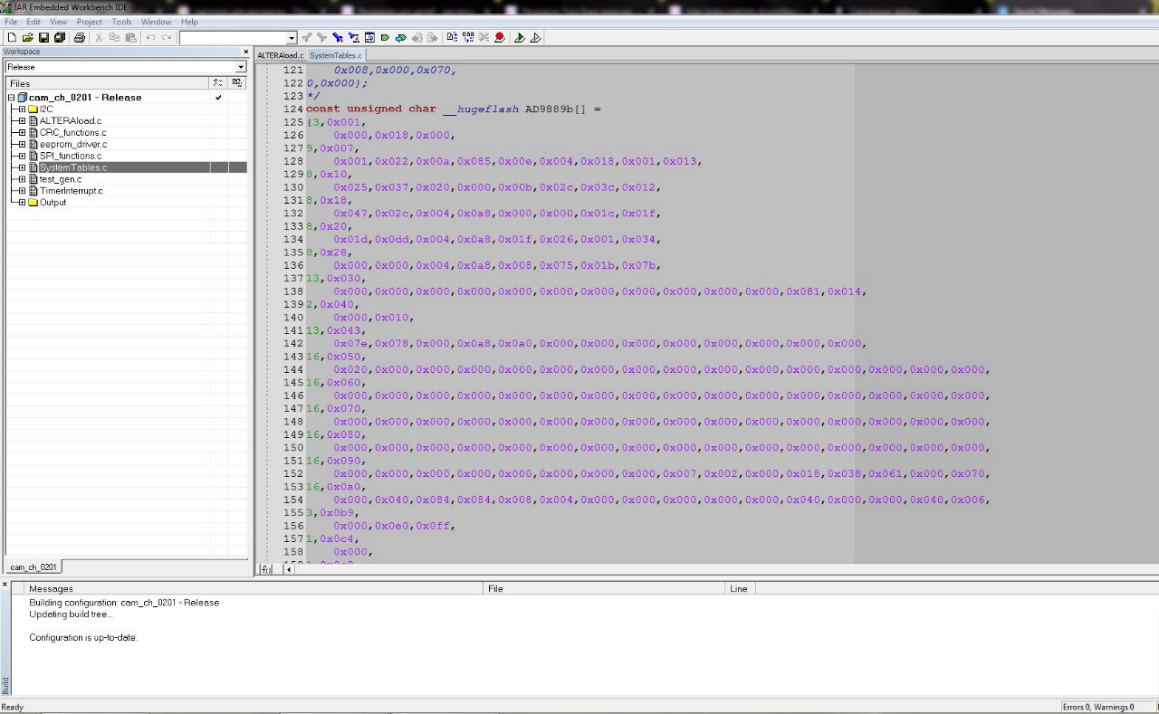
Некоторые из возможностей, которые могут быть доступны в среде разработки IAR 5.3, включают в себя:

Поддержка различных микроконтроллеров: IAR 5.3 может поддерживать различные микроконтроллеры от разных производителей, таких как ARM, Renesas, Atmel, TI, и другие.



**Рис. 1.2.** Окно настройки проекта среды разработки IAR 5.3.

Интерфейс разработки: IAR 5.3 имеет интуитивный интерфейс разработки, который позволяет программистам создавать, отлаживать и тестировать свои проекты в одном месте.



**Рис. 1.3.** Интерфейс среды разработки IAR 5.3.

Редактор кода: IAR 5.3 содержит редактор кода, который может автоматически завершать код и подсвечивать синтаксис для удобства работы с кодом.

Отладчик: IAR 5.3 содержит мощный отладчик, который позволяет программистам отлаживать свои проекты, искать ошибки и просматривать значения переменных и регистров в реальном времени.

Инструменты анализа: IAR 5.3 содержит инструменты анализа, которые позволяют оптимизировать код, анализировать использование памяти и тестировать производительность приложений.

Поддержка RTOS: IAR 5.3 может поддерживать различные операционные системы реального времени, такие как FreeRTOS, Micrium и другие.

Поддержка стандартов: IAR 5.3 поддерживает различные стандарты, такие как C, C++ и Assembler.

* 1. Способы передачи данных между микросхемами.

**SPI** (англ. Serial Peripheral Interface, SPI bus — последовательный периферийный интерфейс, шина SPI) — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии. SPI также иногда называют четырёхпроводным (англ. four-wire) интерфейсом.

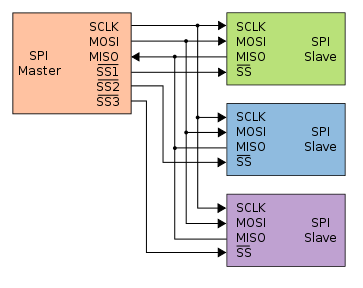


**Рис. 1.4.** Структурная схема сигналов шины SPI.

1. **MOSI** — выход ведущего, вход ведомого (англ. Master Out Slave In). Служит для передачи данных от ведущего устройства ведомому.
2. **MISO** — вход ведущего, выход ведомого (англ. Master In Slave Out). Служит для передачи данных от ведомого устройства ведущему.
3. **SCLK** — последовательный тактовый сигнал (англ. Serial Clock). Служит для передачи тактового сигнала для ведомых устройств.
4. **CS** или **SS** — выбор микросхемы, выбор ведомого (англ. Chip Select, Slave Select).

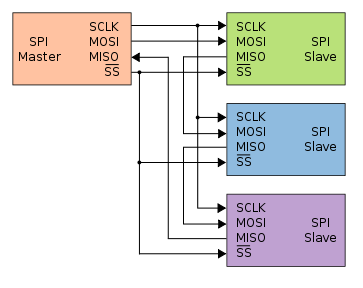
**Принцип работы**  
В простейшем случае к ведущему устройству подключено единственное ведомое устройство и необходим двусторонний обмен данными. В таком случае используется трехпроводная схема подключения. Интерфейс SPI позволяет подключать к одному ведущему устройству несколько ведомых устройств, причем подключение может быть осуществлено несколькими способами.  
Радиальная структура связи с несколькими ведомыми устройствами через SPI.

**Первый способ** позволяет реализовать радиальную структуру связи (топология типа «звезда»), его принято считать основным способом подключения нескольких ведомых устройств. В данном случае для обмена более чем с одним ведомым устройством ведущее устройство должно формировать соответствующее количество сигналов выбора ведомого устройства (SS). При обмене данными с ведомым устройством, соответствующий ему сигнал SS переводится в активное (низкое) состояние, при этом все остальные сигналы SS находятся в неактивном (высоком) состоянии. Выводы данных MISO ведомых устройств соединены параллельно, при этом они находятся в неактивном состоянии, а перед началом обмена один из выходов (выбранного ведомого устройства) переходит в активный режим.  
Кольцевая структура связи с несколькими ведомыми устройствами через SPI



**Рис. 1.5.** Топология типа «звезда» при подключении сигналов шины SPI.

**Второй способ** позволяет реализовать структуру связи типа «кольцо». В данном случае для активации одновременно нескольких ведомых устройств используется один сигнал SS, а выводы данных всех устройств соединены последовательно и образуют замкнутую цепь. При передаче пакета от ведущего устройства этот пакет получает первое ведомое устройство, которое, в свою очередь, транслирует свой пакет следующему ведомому устройству и так далее. Для того, чтобы пакет от ведущего устройства достиг определенного ведомого устройства, ведущее устройство должно отправить еще несколько пакетов.



**Рис. 1.6** Топология типа «кольцо» при подключении сигналов шины SPI.

**Особенности:**

Большим преимуществом шины SPI является наличие сигнала тактовой частоты. Многие SPI чипы и микроконтроллеры поддерживают работу на частоте более 10 МГц, обеспечивая намного более высокую скорость передачи данных, чем интерфейс I2C.

**Использование:**

Часто этот интерфейс используется для работы с SPI-Flash в которой хранится программное обеспечение. Кроме этого, SPI используется для сопряжения различных микросхем устройства для таких целей как конфигурация или считывание статусной информации.

**Протокол I2C.**

**I2C (Inter-Integrated Circuit)** — последовательная шина данных для связи интегральных схем, использующая две двунаправленные линии связи (SDA и SCL).

Используется для соединения низкоскоростных периферийных компонентов с центральным микроконтроллером во встраиваемых системах. Иногда эту шину называют "квадратичной" или "квадратной" или "Ай-Ту-Си"

**Принцип работы:**



**Рис. 1.7.** Топология подключения устройств по шине I2C**.**

Микроконтроллер на рисунке — это ведущий элемент **(Master1)** им может быть процессор. На рисунке представлено 3 ведомых перефириных элемента **Slave** В качестве **Slave** могут быть память, ЦАП, АЦП и пр. К шине может быть подключено до 127 устройств.  
Процессор с памятью соединен в данном случае по двум шинам:

**SDA** (Serial DATA)- шина последовательной передачи данных. Данные по этой шине могут передаваться в двух направлениях.

**SCL** (Serial Clock) - шина, по которой идет тактирование шины данных. Шина синхронизации данных. Она также определяет в какой момент куда пойдут данные. В схеме Master-Master первым битом определяется, кто займет главную роль.  
**Скорость передачи данных.** Так как передаются по 1 биту за 1 такт, то скорость передачи данных составляет 1/8 от тактовой частоты.



**Рис.1.6.** Осциллограмма состояний переходов сигналов при передаче данных по шине I2C.

**Процесс передачи посылки. Состояние СТАРТ и СТОП**

Процедура обмена начинается с того, что ведущий формирует **состояние СТАРТ**: при ВЫСОКОМ уровне на линии SCL он генерирует переход сигнала линии SDA из ВЫСОКОГО состояния в НИЗКОЕ. Этот переход воспринимается всеми устройствами, подключенными к шине, как признак начала процедуры обмена. Генерация синхросигнала — это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине.

При передаче посылок по шине I²C каждый ведущий генерирует свой синхросигнал на линии SCL. После формирования состояния СТАРТ ведущий опускает состояние линии SCL в НИЗКОЕ состояние и выставляет на линию SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено. Спецификация шины I²C разрешает изменения на линии SDA только при НИЗКОМ уровне сигнала на линии SCL. Данные действительны и должны оставаться стабильными только во время ВЫСОКОГО состояния синхроимпульса. Для подтверждения приёма байта от ведущего-передатчика ведомым-приёмником в спецификации протокола обмена по шине I²C вводится специальный бит подтверждения, выставляемый на шину SDA после приёма 8 бит данных.

Процедура обмена завершается тем, что ведущий формирует **состояние СТОП** — переход состояния линии SDA из НИЗКОГО состояния в ВЫСОКОЕ при ВЫСОКОМ состоянии линии SCL. Состояния СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния СТАРТ. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.

**Подтверждение**

Таким образом, передача 8 бит данных от передатчика к приёмнику завершаются дополнительным циклом (формированием 9-го тактового импульса линии SCL), при котором приёмник выставляет низкий уровень сигнала на линии SDA, как признак успешного приёма байта.

Подтверждение при передаче данных обязательно, кроме случаев окончания передачи ведомой стороной. Соответствующий импульс синхронизации генерируется ведущим. Передатчик отпускает (переводит в НИЗКОЕ состояние) линию SDA на время синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение ВЫСОКОГО состояния синхроимпульса подтверждения в стабильном НИЗКОМ состоянии.

В том случае, когда ведомый-приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), линия данных должна быть оставлена в ВЫСОКОМ состоянии. После этого ведущий может выдать состояние СТОП для прерывания пересылки данных. Если в пересылке участвует ведущий-приёмник, то он должен сообщить об окончании передачи ведомому-передатчику путём неподтверждения последнего байта. Ведомый-передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать состояние СТОП или повторить состояние СТАРТ.

**Синхронизация**

Синхронизация выполняется с использованием подключения к линии SCL по правилу монтажного И. Это означает, что ведущий не имеет монопольного права на управление переходом линии SCL из НИЗКОГО состояния в ВЫСОКОЕ. В том случае, когда ведомому необходимо дополнительное время на обработку принятого бита, он имеет возможность удерживать линию SCL в низком состоянии до момента готовности к приёму следующего бита. Таким образом, линия SCL будет находиться в НИЗКОМ состоянии на протяжении самого длинного НИЗКОГО периода синхросигналов.

Устройства с более коротким НИЗКИМ периодом будут входить в состояние ожидания на время, пока не кончится длинный период. Когда у всех задействованных устройств кончится НИЗКИЙ период синхросигнала, линия SCL перейдет в ВЫСОКОЕ состояние. Все устройства начнут проходить ВЫСОКИЙ период своих синхросигналов. Первое устройство, у которого кончится этот период, снова установит линию SCL в НИЗКОЕ состояние. Таким образом, НИЗКИЙ период синхролинии SCL определяется наидлиннейшим периодом синхронизации из всех задействованных устройств, а ВЫСОКИЙ период определяется самым коротким периодом синхронизации устройств.

Механизм синхронизации может быть использован приёмниками как средство управления пересылкой данных на байтовом и битовом уровнях.

На уровне байта, если устройство может принимать байты данных с большой скоростью, но требует определенное время для сохранения принятого байта или подготовки к приёму следующего, то оно может удерживать линию SCL в НИЗКОМ состоянии после приёма и подтверждения байта, переводя таким образом передатчик в состояние ожидания.

На уровне битов устройство, такое, как микроконтроллер без встроенных аппаратных цепей I²C или с ограниченными цепями, может замедлить частоту синхроимпульсов путём продления их НИЗКОГО периода. Таким образом скорость передачи любого ведущего адаптируется к скорости медленного устройства.

**Адресация в шине I²C**

Каждое устройство, подключённое к шине, может быть программно адресовано по уникальному адресу. Для выбора приёмника сообщения ведущий использует уникальную адресную компоненту в формате посылки. При использовании однотипных устройств ИС часто имеют дополнительный селектор адреса, который может быть реализован как в виде дополнительных цифровых входов селектора адреса, так и в виде аналогового входа. При этом адреса таких однотипных устройств оказываются разнесены в адресном пространстве устройств, подключенных к шине.

В обычном режиме используется 7-битная адресация.

Процедура адресации на шине I²C заключается в том, что первый байт после сигнала СТАРТ определяет, какой ведомый адресуется ведущим для проведения цикла обмена. Исключение составляет адрес «Общего вызова», который адресует все устройства на шине. Когда используется этот адрес, все устройства в теории должны послать сигнал подтверждения. Однако устройства, которые могут обрабатывать «общий вызов», на практике встречаются редко.

Первые семь битов первых двух байтов образуют адрес ведомого. Восьмой, младший бит, определяет направление пересылки данных. «Ноль» означает, что ведущий будет передавать информацию выбранному ведомому. «Единица» означает, что ведущий будет получать информацию от ведомого.

После того, как адрес послан, каждое устройство в системе сравнивает первые семь бит после сигнала СТАРТ со своим адресом. При совпадении устройство полагает себя выбранным как ведомый-приёмник или как ведомый-передатчик, в зависимости от бита направления.

Адрес ведомого может состоять из фиксированной и программируемой части. Часто случается, что в системе имеется несколько однотипных устройств (к примеру, ИМС памяти, или [драйверов](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0)) светодиодных [индикаторов](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D0%B4%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80)), поэтому при помощи программируемой части адреса становится возможным подключить к шине максимально возможное количество таких устройств. Количество программируемых битов в адресе зависит от количества свободных выводов микросхемы. Иногда используется один вывод с аналоговой установкой программируемого диапазона адресов. При этом в зависимости от потенциала на этом адресном выводе ИМС, возможно смещение адресного пространства драйвера так, чтобы однотипные ИМС не конфликтовали между собой на общей шине.

Все специализированные ИМС, поддерживающие работу в стандарте шины I²C, имеют набор фиксированных адресов, перечень которых указан производителем в описаниях контроллеров.

Комбинация бит 11110ХХ адреса зарезервирована для 10-битной адресации.

Как следует из спецификации шины, допускаются как простые форматы обмена, так и комбинированные, когда в промежутке от состояния СТАРТ до состояния СТОП ведущий и ведомый могут выступать и как приёмник, и как передатчик данных. Комбинированные форматы могут быть использованы, например, для управления последовательной памятью.

Во время первого байта данных можно передавать адрес в памяти, который записывается во внутренний [регистр-защёлку](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_(%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)). После повторения сигнала СТАРТа и адреса ведомого выдаются данные из памяти. Все решения об авто-инкременте или декременте адреса, к которому произошёл предыдущий доступ, принимаются конструктором конкретного устройства. Поэтому в любом случае лучший способ избежать неконтролируемой ситуации на шине перед использованием новой (или ранее не используемой) ИМС — следует тщательно изучить паспорт изделия или справочное руководство.

В любом случае по спецификации шины все разрабатываемые устройства должны сбрасывать логику шины при получении сигнала СТАРТ или повторный СТАРТ и подготавливаться к приёму адреса.

Тем не менее, основные проблемы с использованием I²C шины возникают именно из-за того, что разработчики, «начинающие» работать с I²C шиной, не учитывают того факта, что ведущий (часто — микропроцессор) не имеет монопольного права ни на одну из линий шины.

* 1. Методы загрузки микросхем FPGA

**FPGA** (Field-Programmable Gate Array) — это интегральная схема, предназначенная для программирования после изготовления. FPGA представляет собой программируемую логическую матрицу, состоящую из логических элементов, макроклеток и программируемых межсоединений.

После подачи напряжения питания внутренние соединения микросхемы FPGA не настроены. Обычно настройка внутренних меж соединений хранится в отдельной микросхеме памяти (FLASH, EEPROM). Задача микроконтроллера в этом случае осуществить передачу данных из энергонезависимой памяти в микросхему программируемой логики.

Существует несколько методов загрузки микросхем FPGA, вот некоторые из них:

**Active Serial** - в этом методе конфигурационные данные передаются последовательно через один контакт микросхемы. Данный метод подходит для загрузки FPGA из внешней памяти, такой как EEPROM, Flash или FRAM. Микросхема FPGA при этом сама осуществляет синхронизацию и передачу данных.

**Passive Serial** - данный метод загрузки использует несколько контактов микросхемы FPGA для передачи конфигурационных данных. В этом методе используется контроллер, который передает данные на определенные контакты микросхемы FPGA.

**JTAG** - это метод, который использует интерфейс JTAG (Joint Test Action Group), который обычно используется для тестирования и отладки микросхем. Однако он также может быть использован для загрузки конфигурационных данных на FPGA.

**HPS** (Hard Processor System) - встроенный процессор загружает конфигурационные данные на FPGA. Данный метод может использоваться в FPGA с встроенным процессором.

**Remote Update** - в этом методе FPGA загружается через интерфейс Ethernet или другие беспроводные сети. Этот метод может использоваться для обновления FPGA на удаленном оборудовании.

Каждый из этих методов имеет свои преимущества и недостатки и выбор метода зависит от требований к конкретному приложению.

* 1. Заключение

В данной главе были рассмотрены аппаратные и программные средства для реализации генератора тестовых видеосигналов.

Были рассмотрены существующие архитектуры процессоров и в том числе архитектура AVR, используемая во встроенных системах. Были описаны способы обмена процессором информацией с периферийными устройствами посредством шин: I2C(two wire interface) и SPI(serial peripheral interface). Был рассмотрен микропроцессор AtXmega, его аппаратные компоненты и процесс написания программ под него.

Из периферийных устройств была представлена микросхема программируемой логики. Распараллеливание процессов а также высокая производительность позволяют решить задачу формирования видео изображения. Задача микроконтроллера в этом случае сводится к загрузке FPGA и управлению переключением выбора типа видео изображения.

Понимание принципа работы аппаратной и программной части является необходимым требованием и будет использоваться в дальнейшем при построении и анализе оптимальности построенного устройства.

Глава II. ПОСТАНОВКА ЗАДАЧИ И АНАЛИЗ МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННЫХ ЗАДАЧ

1. Постановка задачи

Целью данной главы является изучение выбранного способа загрузки FPGA. Также будет рассмотрена процедура передачи данных при загрузке и способы решения задачи.

1. Описание выбранного способа загрузки FPGA

Для загрузки микросхемы FPGA будет использован метод Passive Serial.

Вот краткое описание этого способа:

1. Подготовка конфигурационных данных: сначала необходимо подготовить конфигурационные данные для FPGA. Эти данные определяют логику и соединения блоков внутри FPGA. Проект внутренней логики разрабатывается в специализированной IDE и не является частью этой мастерской работы. Результатом компиляции проекта является бинарный файл, который при помощи программатора переносится во Флеш память устройства. Задача микроконтроллера при этом обеспечить надежную передачу данных для загрузки FPGA из Флэш память в соответствии с выбранной процедурой загрузки.

2. Подключение микросхемы FPGA: Микросхема FPGA должна быть правильно подключена к программатору или контроллеру, который будет передавать конфигурационные данные. При правильном подключении микроконтроллер инициализирует начало процедуры конфигурации посредством сигнала nCONF (запрос на начало конфигурации). В ответ на запрос FPGA отвечает сигналом статуса nSTATUS. В случае если все условия удовлетворяют требуемым – ответ положительный (низкий уровень сигнала).

3. Последовательная передача данных: В методе passive serial данные передаются по нескольким контактам микросхемы FPGA последовательно. Контроллер или программатор отправляет биты данных последовательно через контакты FPGA. При этом микроконтроллер осуществляет побитную передачу загрузочных данных в FPGA.

4. Управляющий сигнал: кроме передачи данных, также используется управляющий сигнал, обычно называемый "контрольным сигналом загрузки" (load control signal). Этот сигнал указывает микросхеме FPGA начало передачи новых данных.

5. Загрузка данных: Контроллер или программатор последовательно отправляет биты данных через контакты микросхемы FPGA в соответствии с протоколом передачи данных. Данные сохраняются во внутренней конфигурационной памяти FPGA.

6. Завершение загрузки: после передачи всех конфигурационных данных контроллер отправляет специальный сигнал, который указывает микросхеме FPGA завершить процесс загрузки и начать работу с новой конфигурацией.

7. Проверка и запуск: после успешной загрузки FPGA проверяет корректность конфигурации и переходит к выполнению задач, определенных внутри программы.

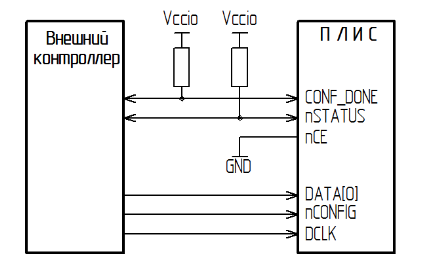
Метод passive serial прост в реализации и требует минимального количества контактов для передачи данных, представляет собой способ последовательной передачи информации между устройствами. В этом методе одно устройство является активным и генерирует тактовый сигнал, а другое устройство пассивно принимает данные.

При использовании passive serial данные передаются последовательно по одному биту за раз. Активное устройство генерирует тактовый сигнал, который управляет синхронизацией передачи данных. Каждый бит данных передается в момент активного состояния тактового сигнала.

Преимуществом passive serial является его простота и низкая стоимость реализации. Он может использоваться в различных приложениях, включая коммуникацию с периферийными устройствами, такими как дисплеи, сенсоры или память.

Однако, следует отметить, что передача данных в режиме passive serial обычно происходит медленнее, чем в других методах передачи данных, таких как параллельная передача. Это связано с ограничениями на скорость передачи данных через один канал.

Таким образом, метод передачи данных passive serial является простым и экономически выгодным способом последовательной передачи информации, который может быть использован в различных приложениях с небольшими требованиями к скорости передачи данных.



**Рис.2.1.** Подключения ПЛИС к внешнему контроллеру для загрузки в режиме PS.

1. Краткое описание возможностей микросхемы HDMI выхода AD9889B

Микросхема AD9889B является высококачественным HDMI-трансмиттером, который обеспечивает широкий набор возможностей для передачи аудио и видео сигналов по интерфейсу HDMI. Вот краткое описание основных возможностей микросхемы AD9889B:

1. Поддержка HDMI-стандартов: Микросхема AD9889B полностью соответствует стандартам HDMI 1.3a и DVI 1.0. Она обеспечивает высокое качество передачи видео и аудио сигналов с разрешением до 1080p.

2. Входные интерфейсы: AD9889B имеет различные входные интерфейсы для подключения к источнику видеосигнала. Она поддерживает аналоговый компонентный видеоинтерфейс (YPbPr) и аналоговый RGB-интерфейс.

3. Кодирование видеосигнала: Микросхема AD9889B предоставляет возможность кодирования аналоговых видеосигналов в цифровой формат HDMI для передачи по интерфейсу HDMI.

4. Поддержка аудио: AD9889B поддерживает передачу аудиосигналов с различными форматами, включая двухканальный PCM (Pulse Code Modulation), многоканальный линейный PCM и сжатые аудиоформаты, такие как Dolby Digital и DTS.

5. Гибкая настройка: Микросхема AD9889B обеспечивает различные настройки и конфигурации, позволяющие пользователю оптимизировать передачу видео и аудио сигналов в соответствии с требованиями конкретного приложения.

6. Защита контента: AD9889B поддерживает технологию HDCP (High-bandwidth Digital Content Protection), которая обеспечивает защиту цифрового контента от несанкционированного копирования или перехвата.

7. Низкое энергопотребление: Микросхема AD9889B имеет низкое энергопотребление, что делает ее подходящей для применения в системах с ограниченным энергоснабжением.

Микросхема AD9889B предоставляет широкий набор функций и возможностей для передачи видео и аудио сигналов по интерфейсу HDMI. Она является надежным и высококачественным решением для передачи HDMI-сигналов в различных приложениях.

1. Конфигурация микросхемы AD9889B

Микросхема AD9889B конфигурируется путем программирования регистров внутри самой микросхемы. Конфигурация определяет параметры работы микросхемы, такие как разрешение видео, режимы цветности, настройки аудио и другие параметры.

Для конфигурации микросхемы AD9889B используется последовательный интерфейс контроля (I2C). Через этот интерфейс можно записывать значения в регистры микросхемы для установки нужных параметров.

Процесс конфигурации обычно включает следующие шаги:

1. Подключение: Микросхема AD9889B должна быть правильно подключена к контроллеру или микроконтроллеру, который будет осуществлять программирование регистров.

2. Инициализация: перед началом конфигурации микросхемы AD9889B необходимо выполнить инициализацию, включающую установку начальных значений регистров.

3. Запись значений в регистры: Контроллер или микроконтроллер записывает значения параметров в соответствующие регистры микросхемы AD9889B. Эти значения определяют требуемые настройки для разрешения видео, цветности, аудио и других функций.

4. Проверка: после записи значений в регистры микросхемы AD9889B можно выполнить проверку, чтобы убедиться, что конфигурация прошла успешно. Можно считать значения регистров и сравнить их с ожидаемыми значениями.

5. Рабочий режим: после завершения конфигурации микросхема AD9889B работает в соответствии с заданными параметрами, передавая видео- и аудиосигналы по интерфейсу HDMI.

Важно отметить, что конкретные регистры и их значения для конфигурации микросхемы AD9889B зависят от требований и настроек конкретного приложения. Руководство пользователя или документация микросхемы предоставят подробную информацию о доступных регистрах и их настройках.

1. Использование роторного переключателя

Роторный переключатель (Rotary Switch) — это механическое устройство, используемое для выбора одного из нескольких предопределенных вариантов. Он обычно состоит из вала и поворотной крышки с контактами, которые соответствуют различным положениям переключателя.

Использование роторного переключателя сводится к вращению его поворотной крышки для выбора нужного положения. При вращении контакты внутри переключателя переключаются между различными положениями и устанавливаются в соответствующем положении.

Роторные переключатели широко используются в различных приложениях, где требуется выбор между несколькими опциями или настройками. Например, они могут быть использованы в аудиоустройствах для выбора источника звука или регулировки громкости, в электронных приборах для выбора режимов работы или настройки параметров, а также в промышленных системах для выбора режимов работы или управления.

Для использования роторного переключателя необходимо принять во внимание его конструкцию и особенности. Некоторые роторные переключатели могут иметь дополнительные функции, такие как щелчок при переключении или индикацию текущего положения. Также важно правильно подключить контакты переключателя к соответствующим компонентам или цепям в системе.

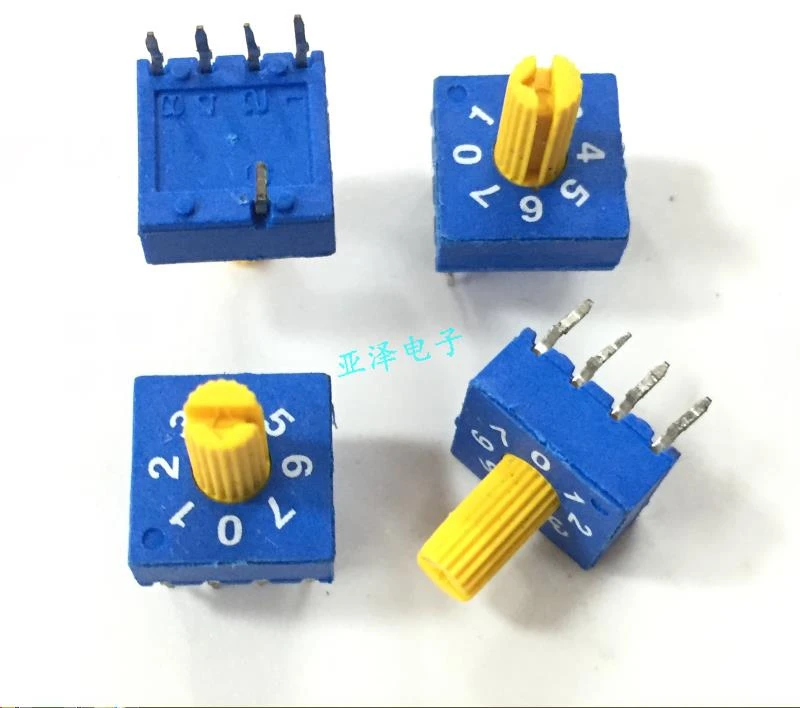
Роторные переключатели предоставляют удобный и надежный способ выбора опций или настроек в различных устройствах и системах.

A picture containing circuit component, passive circuit component, electronic component

Description automatically generated

**Рис.2.2.** Различные варианты исполнения роторных переключателей.

В данной работе используется восьмипозиционный роторный переключатель. Не вдаваясь в детали его внутреннего устройства следует отметить что такой переключатель позволяет пользователю выбрать до восьми разных видео изображений.



**Рис.2.3**.Роторный переключатель на 8 положений.

1. Заключение

Вторая глава была посвящена изучению способов решения поставленной задачи. В качестве основных компонентов будущего устройства были выбраны компоненты, соответствующие минимальным требованиям системы. В качестве метода загрузки FPGA был выбран способ Passive Serial. Такой способ загрузки микросхемы программируемой логики позволяет осуществить начальную перегрузку данных за приемлемое время. Также такой способ минимизирует количество используемых портов микроконтроллера. Кроме этого, микроконтроллер имеет полный доступ как для чтения, так и для записи к микросхеме Флэш памяти, в которой хранится загрузочный файл. Такое решение позволит обновлять загрузочный файл через какой-либо из доступных микроконтроллеру интерфейсов.

В качестве микросхемы вывода видео потока HDMI была выбрана микросхема фирмы Analog Devices серии AD9889B. Такое решение требует от микроконтроллера только первоначальной загрузки массива инициализации регистров. Массив инициализации регистров был сформирован исходя из файла описания микросхемы от производителя.

В качестве способа выбора выходного сигнала был выбран восьмипозиционный роторный переключатель. Такое решение позволяет пользователю изменять параметры устройства без применения других внешних устройств, например компьютера. Роторный переключатель подключен к портам ввода-вывода микроконтроллера. Состояние портов отслеживается в прерывании от таймера. Такое решение позволяет программным способом решить проблему дребезга контактов при переключении, а переключение картинки происходит для пользователя практически мгновенно.

Глава III. РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ГЕНЕРАТОРА ТЕСТОВЫХ ВИДЕОСИГНАЛОВ

В результате практической части данной работы был разработана программа управления генератором тестовых видеоизображений. Были реализованы программные модули загрузки микросхемы ПЛИС, инициализации микросхемы физического уровня HDMI, а также модуль прерывания по таймеру для обработки сигналов от роторного переключателя.

1. Принципиальная схема

Принципиальная схема разрабатываемого устройства состоит из трех основных компонентов - микроконтроллер, ПЛИС, микросхема физического уровня HDMI.

A picture containing text, diagram, line, number

Description automatically generated

**Рис. 3.1** Принципиальная схема генератора.

1. Инициализация портов ввода-вывода
   1. Макросы настройки портов ввода-вывода.

Настройка функциональности портов ввода-вывода производится изменением значений регистров настройки портов. Вариантов настройки порта очень много – можно комбинировать значения регистров одного порта чтоб получить например:

* Порт настроенный как вход микроконтроллера с подтяжкой к питанию
* Порт настроенный как вход микроконтроллера с подтяжкой к земле
* Порт настроенный как выход микроконтроллера
* Порт микроконтроллера, настроенный как ЛОГИЧЕСКОЕ И
* А также много других вариантов.

Для удобства настройки в данной работе используются макросы, которые предоставляют доступ сразу ко всем необходимым регистрам портов ввода-вывода. Эти макросы позволяют настроить порты для случаев, необходимых в процессе работы изделия.

**#define OutputPinWith1(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_TOTEM\_gc; PORT##F.OUTSET = (1 << F); PORT##F.DIRSET = (1 << F)**

**#define OutputPinWith0(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_TOTEM\_gc; PORT##F.OUTCLR = (1 << F); PORT##F.DIRSET = (1 << F)**

**#define InputPinWithZState(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_TOTEM\_gc; PORT##F.DIRCLR = (1 << F)**

**#define InputPinWithPullUp(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_PULLUP\_gc; PORT##F.DIRCLR = (1 << F)**

**#define InputPinWithPullDown(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_PULLDOWN\_gc; PORT##F.DIRCLR = (1 << F)**

**#define OutputPinWireAndPullUpWith1(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_WIREDANDPULL\_gc; PORT##F.OUTSET = (1 << F); PORT##F.DIRSET = (1 << F)**

**#define OutputPinWireAndPullUpWith0(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_WIREDANDPULL\_gc; PORT##F.OUTCLR = (1 << F); PORT##F.DIRSET = (1 << F)**

**#define OutputPinWireAndWith1(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_WIREDAND\_gc; PORT##F.OUTSET = (1 << F); PORT##F.DIRSET = (1 << F)**

**#define OutputPinWireAndWith0(F) \*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) =(\*((register8\_t \*)&(((PORT\_t \*)&PORT##F)->PIN0CTRL) + F) & ~(0x07 << PORT\_OPC\_gp))| PORT\_OPC\_WIREDAND\_gc; PORT##F.OUTCLR = (1 << F); PORT##F.DIRSET = (1 << F)**

**Листинг 3.1.** Макросы настройки портов ввода-вывода микроконтроллера.

Как видно из описания, каждый макрос принимает как параметр название порта. Названия портов определены заранее и определяются тем, каким образом определены соединения внешних сигналов на печатной плате.

В следующем листинге представлен пример использования директивы #define для определения портов, необходимых для внешней шины I2C.

#define PORTSDAa PORTF

#define SDAa 0

#define PORTSCLa PORTF

#define SCLa 1

**Листинг 3.2.** Пример определения портов ввода-вывода.

Как видно из листинга сигнал SDA подключен к порту PORTF. Номер сигнала 0 задает битовое смещение в регистрах настройки.

* 1. Начальная инициализация портов ввода вывода

Правильная начальная инициализация портов ввода-вывода — это залог хорошей работы всей программы. В этой части программа необходимо определить какие порты ввода-вывода потребуются для нормальной работы программы и реализации всех поставленных задач. Конечным результатом работы программы будет загрузка всех узлов генератора и отработка запросов пользователя на изменение вида тестовой картинки.

/\* Порты для передачи данных между микроконтроллером и ПЛИС - шина SPI\*/

OutputPinWith1**(**FSS**);**//Altera Load SPI

OutputPinWith1**(**FMOSI**);**

InputPinWithPullUp**(**FMISO**);**

OutputPinWith1**(**FSCK**);**

/\* Порты для шины I2C - инициализация AD9889b \*/

OutputPinWireAndWith1**(**SDAa**);**

OutputPinWireAndWith1**(**SCLa**);**

/\* Порт сброса внутренней логики ПЛИС \*/

OutputPinWith0**(**NRES**);**

/\* Порты для передачи данных между микроконтроллером и флэш памятью\*/

OutputPinWith1**(**SS1**);**

OutputPinWith1**(**MOSI**);**

InputPinWithPullUp**(**MISO**);**

OutputPinWith1**(**SCK**);**

/\* Порты для загрузки ПЛИС по методу Passive Serial \*/

OutputPinWith0**(**NCONF**);**

InputPinWithZState**(**NSTAT**);**

OutputPinWith0**(**DCLK**);**

InputPinWithZState**(**CONFD**);**

/\* Порт сканирования роторного переключателя \*/

InputPortWithPullUp**(**PORTB**);**//rotary switches port

**Листинг 3.3**. Начальная инициализация портов ввода-вывода

* 1. Загрузка FPGA

Как было описано ранее для загрузки FPGA был выбран метод Passive Serial. Для реализации данного метода необходимо осуществить электрическое соединение трех компонентов: Микроконтроллер, ПЛИС а также микросхема память Флэш. Соединение микроконтроллера и ПЛИС осуществляется при помощи следующих сигналов:

#define PORTDCLK PORTH

#define DCLK 3

#define PORTNCONF PORTH

#define NCONF 4

#define PORTCONFD PORTH

#define CONFD 5

#define PORTNSTAT PORTH

#define NSTAT 6

**Листинг 3.4**. Назначение портов для загрузки ПЛИС.

Процедура передачи данных между микросхемами флэш и ПЛИС осуществляется согласно процедуре, описанной в документации ПЛИС для метода Passive Serial. Листинг программы описан ниже.

#define DMATransmit \

PORTFSCK.OUTCLR = 1 << FSCK; PORTFSCK.OUTSET = 1 << FSCK;\

PORTDCLK.OUTSET = 1 << DCLK; PORTDCLK.OUTCLR = 1 << DCLK;

u8b ALTERAload**(**void**)**

**{**

\_\_watchdog\_reset**();**

/\* Сброс предыдущего состояния ПЛИС \*/

OutputPinWith0 **(**NCONF**);**

/\* Ожидание завершения сброса \*/

**do** \_\_delay\_cycles**(**uS**(**10.0**));**

**while** **(**PORTNSTAT**.**IN **&** **(**1**<<**NSTAT**));**

/\* Ожидание готовности ПЛИС к новой загрузке \*/

**do** \_\_delay\_cycles**(**uS**(**10.0**));**

**while** **(**PORTCONFD**.**IN **&** **(**1**<<**CONFD**));**

\_\_delay\_cycles**(**mS**(**1.0**));**

/\* Запрос на начало процесса загрузки прошивки \*/

OutputPinWith1 **(**NCONF**);**

\_\_delay\_cycles**(**mS**(**1.0**));**

/\* Ожидание ответа \*/

**do** \_\_delay\_cycles**(**uS**(**10.0**));**

**while** **((**PORTNSTAT**.**IN **&** **(**1**<<**NSTAT**))** **==** 0**);**

\_\_watchdog\_reset**();**

\_\_delay\_cycles**(**uS**(**1.0**));**

/\* Передача данных из флеш \*/

/\* Активация запроса к Флеш \*/

OutputPinWith0 **(**FSS**);**

/\* \*/

ByteFlashTransmit**(**ReadDataS25Instruction**);**

/\* Передача начального адреса в памяти флеш 0x00000000 для данного проекта \*/

ByteFlashTransmit**(**0**);**

ByteFlashTransmit**(**0**);**

ByteFlashTransmit**(**0**);**

**do{**

/\* сброс сторожевого таймера \*/

\_\_watchdog\_reset**();**

/\* Перекачка данных из флеш в ПЛИС \*/

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

DMATransmit

/\* В процессе передачи произошла ошибка – выходим из функции

и возвращаем 0 \*/

**if** **(!(**PORTNSTAT**.**IN **&** **(**1**<<**NSTAT**)))**

**return** 0**;**

**}while** **(!(**PORTCONFD**.**IN **&** **(**1**<<**CONFD**)));**

/\* Закрываем транзакцию передачи данных из флеш \*/

OutputPinWith1 **(**FSS**);**

/\* Выход из функции – возвращаем 1 \*/

**return** 1**;**

**}**

**Листинг 3.5**. Процедур загрузки ПЛИС по методу Passive Serial.

Для ускорения загрузки процесс битовой передачи описан в виде макроса DMATransmit.

* 1. Инициализация микросхемы HDMI выхода AD9889B

Инициализация микросхемы HDMI выхода AD9889B нужна для правильного отображения видео картинки на выходе HDMI. Для инициализация используется шина I2C. Порты шины определены в следующем листинге:

#define PORTSDAb PORTE//EDID I2C

#define SDAb 0

#define PORTSCLb PORTE

#define SCLb 1

**Листинг 3.6**. Назначение портов шины I2C.

В следующем листинге представлена таблица инициализации регистров микросхемы AD9889B. Все значения регистров выбраны в соответствии с выбранным режимом работы микросхемы исходя из описания производителя.

const unsigned char \_\_hugeflash AD9889b**[]** **=**

**{**3**,**0x001**,**

0x000**,**0x018**,**0x000**,**

9**,**0x007**,**

0x001**,**0x022**,**0x00a**,**0x085**,**0x00e**,**0x004**,**0x018**,**0x001**,**0x013**,**

8**,**0x10**,**

0x025**,**0x037**,**0x020**,**0x000**,**0x00b**,**0x02c**,**0x03c**,**0x012**,**

8**,**0x18**,**

0x047**,**0x02c**,**0x004**,**0x0a8**,**0x000**,**0x000**,**0x01c**,**0x01f**,**

8**,**0x20**,**

0x01d**,**0x0dd**,**0x004**,**0x0a8**,**0x01f**,**0x026**,**0x001**,**0x034**,**

8**,**0x28**,**

0x000**,**0x000**,**0x004**,**0x0a8**,**0x008**,**0x075**,**0x01b**,**0x07b**,**

13**,**0x030**,**

0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x081**,**0x014**,**

2**,**0x040**,**

0x000**,**0x010**,**

13**,**0x043**,**

0x07e**,**0x078**,**0x000**,**0x0a8**,**0x0a0**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**

16**,**0x050**,**

0x020**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**

16**,**0x060**,**

0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**

16**,**0x070**,**

0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**

16**,**0x080**,**

0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**

16**,**0x090**,**

0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x007**,**0x002**,**0x000**,**0x018**,**0x038**,**0x061**,**0x000**,**0x070**,**

16**,**0x0a0**,**

0x000**,**0x040**,**0x084**,**0x084**,**0x008**,**0x004**,**0x000**,**0x000**,**0x000**,**0x000**,**0x000**,**0x040**,**0x000**,**0x000**,**0x040**,**0x006**,**

3**,**0x0b9**,**

0x000**,**0x0e0**,**0x0ff**,**

1**,**0x0c4**,**

0x000**,**

1**,**0x0c9**,**

0x003**,**

3**,**0x0cd**,**

0x008**,**0x000**,**0x070**,**

0**,**0x000**};**

**Листинг 3.7**. Таблица инициализации регистров AD9889B.

Процесс инициализации микросхемы AD9889B представляет из себя вызов функции передачи массива по шине I2C. В качестве параметров в функцию передаются: адрес микросхемы на шине I2C, байтовый указатель на массив, содержащий таблицу инициализации, байтовый указатель на буфер временного хранения данных при работе с микросхемой.

I2CArrayWra**(**AdAD9889b**,** **&**AD9889b**[**0**],**Buffer**);**

**Листинг 3.8**. Вызов функции передачи данных по шине I2C.

* 1. Отработка прерывания по таймеру

Для сканирования состояния роторного переключателя в программе используется прерывание по таймеру. Таймер микроконтроллера настраивается на генерацию прерываний каждые 20 миллисекунд. В прерывании от таймера программа каждый раз считывает текущее состояние входного порта, к которому подключен роторный переключатель. При изменении текущего состояния входного порта взводится флаг изменения состояния. Для стабильной работы устройства в программе реализован механизм защиты от дребезга контактов. Дребезг контактов возникает при замыкании или размыкании переключателей. В случае неиспользования механизма защиты при переключениях могут происходить многократные детектирования разных промежуточных состояний порта. Также может возникнуть плавное нарастание или спадание сигнала от переключателя. Такое может происходить вследствие паразитной емкости контактов. Плавное изменение фронта сигнала также может быть детектировано двояко на границах значений логических напряжений. Такой механизм предполагает выдерживание паузы при каждом изменении сигнала. Лишь после того, как сигнал изменился и в течении защитного интервала был стабилен новое значение поступит на исполнительный блок.

На следующем листинге представлен код вызова функции инициализации таймера.

/\* Инициализация таймера с прерыванием каждые 20 миллисекунд \*/

InitCounterNorm**(&**TCC0**,** 20000.0**,** 0x02**);**

**Листинг 3.9**. Вызов функции инициализации таймера.

Код программы, описывающий прерывание по таймеру описан в следующем листинге.

#pragma vector = TCC0\_OVF\_vect

\_\_interrupt void SysTimer**(**void**)**

**{** \_\_watchdog\_reset**();**

/\* Определение количества циклов стабильного

состояния для борьбы с дребезгом контактов

\*/

#define MaxScanDelay 5

/\* Внутренние переменные функции сканирования состояния порта \*/

static u8b Rot\_Value**,**Rot\_oldValue**,**ScanDelay2**,**PrevButton2**;**

/\* Считывание текущего состояния регистра порта \*/

Rot\_Value **=** PORTB**.**IN**;**

/\* Текущее значение не равно предыдущему\*/

**if(**Rot\_Value **!=** Rot\_oldValue**)**

**{**

ClrFlag**(**FlagsScanner**,**FlRotEqual**);**

**}**

/\* Состояние не изменилось в текущем цикле \*/

**if** **(**CheckFlag**(**FlagsScanner**,** FlRotEqual**))**

**{**

/\* Состояние кнопки не менялось в течении установленного

интервала - защита от дребезга контактов

\*/

**if** **(**ScanDelay2 **>=** MaxScanDelay**)**

**{**

/\* Текущее стабильное состояние

кнопки отличается от предыдущего

\*/

**if(**PrevButton2 **!=** Rot\_Value**)**

**{**

/\* Установка внутреннего флага сканирования \*/

SetFlag**(**FlagsScanner**,** FlRotEqual**);**

/\* Установка флага готовности нового состояния

переключателя для основной программы

\*/

SetFlag**(**FlagsScanner**,** FlRotReady**);**

/\* Запись нового значения в глобальную переменную \*/

RotState **=** Rot\_Value**;**

PrevButton2 **=** Rot\_Value**;**

**}**

**}**

**else**

**{**

/\* Выдержка интервала для защиты от дребезга контактов \*/

ScanDelay2**++;**

**}**

**}**

**else**

**{**

/\* Сброс счетчика защиты от дребезга контактов \*/

ScanDelay2 **=** 0**;**

SetFlag**(**FlagsScanner**,** FlRotEqual**);**

**}**

/\* Замена предыдущего состояния текущим \*/

Rot\_oldValue **=** Rot\_Value**;**

**}**

**Листинг 3.10**. Вызов функции инициализации таймера.

* 1. Работа основной части программы

После инициализации всех внешних узлов и настройки прерывания по таймеру задача сводится к тому, чтоб отслеживать состояние флага готовности нового значения переключателя. При условии, что изменилось состояние переключателя новое значение переписывается в ПЛИС. Флаг готовности сбрасывается для избежания повторногой перезаписи.

**do{**

/\*Проверка флага изменения состояния роторного переключателя\*/

**if(**CheckFlag**(**FlagsScanner**,** FlRotReady**))**

**{** /\* Сброс флага \*/

ClrFlag**(**FlagsScanner**,** FlRotReady**);**

/\* Запись нового значения в ПЛИС \*/

InitHDMI\_Select**();**

**}**

**}while(**1**);**

**Листинг 3.11**. Работа главного цикла программы.

* 1. Обновление регистров выбранного видеосигнала

После того как механизм сканирования порта обнаружил изменения на шине переключателя новое состояние переключателя необходимо передать в ПЛИС для коммутации выбранного видео сигнала. Связь микроконтроллера и ПЛИС осуществляется посредством шины SPI. Для передачи данных используется аппаратный блок SPI на порту F. Транзакция на шине SPI представляет из себя передачу адреса регистра, а также непосредственно данных. В следующем листинге представлен код передачи байта коммутации по адресу 1 по шине SPI.

void InitHDMI\_Select**(**void**)**

**{**

/\* Временный буффер для хранения

текущего состояния переключателя

\*/

u8b buffer**[**4**];**

/\* Внутренний адрес регистра на шине SPI \*/

u8b SPIAddr **=** 1**;**

/\* Длина блока передачи \*/

u8b TransfLen **=** 1**;**

/\* Копирование данных в буфер \*/

buffer**[**0**]** **=** RotState**;**

/\* Передача данных по шине SPI \*/

OutputPinWith0**(**SS1**);**

WrSPI**(&**SPIF**,** **&**buffer**[**0**]);**

OutputPinWith1**(**SS1**);**

**}**

**Листинг 3.12**.Передача текущего состояния переключателя.

1. Заключение

В третьей главе был описан весь процесс разработки программной части генератора видео изображений. Были реализованы процедуры инициализации портов ввода-вывода микроконтроллера. Для улучшения качества и скорости выполнения кода программы инициализация портов ввода-вывода производилась при помощи макросов. Также был реализован программным способом выбранный метод загрузки FPGA. Программный метод реализации показал приемлемую скорость передачи данных между Flash и FPGA. Время загрузки FPGA получилось не более трех секунд.

Важной частью работы является пользовательский интерфейс. В этой работе реализован один из самых простых видов взаимодействия с пользователем – при помощи роторного переключателя. Такое решение является достаточным для выполнения целей данной работы.

ВЫВОДЫ И РЕКОМЕНДАЦИИ

В данной работе были рассмотрены подходы к написанию программ управления устройств генерации тестовых видео изображений. В работе представлены наиболее распространенные на рынке компоненты для построения устройства, такие как микроконтроллер фирмы Atmel, роторный переключатель на 8 положений, микросхемы FPGA семейства Cyclone IV, микросхема фирмы Analog Devices серии AD9889B. для вывода потока видео изображения .

В процессе разработки были проанализированы различные архитектуры микропроцессоров и способы организации аппаратного обеспечения для достижения лучших результатов с точки зрения экономичности и простоты разработки.

Результатом работы стала разработка программы управления генератором тестовых видео изображений, исполняемой микроконтроллером (AtXmega). Данный микроконтроллер справился со всеми поставленными задачами, в том числе управление периферийными микросхемами, а также первоначальная загрузка FPGA методом Passive Serial. Данный метод загрузки (Passive Serial) позволил ускорить процесс разработки сократив количество соединений между микросхемами.

Использование протоколов I2C (Inter-Integrated Circuit) и SPI (Serial Peripheral Interface) позволило решить все поставленные задачи. Использование этих протоколов может быть рекомендовано по нескольким причинам:

1. Простота реализации: Протоколы I2C и SPI относительно просты в реализации и требуют минимального количества пинов для подключения. Это делает их удобными для применения в системах с ограниченным количеством доступных пинов или ограниченным пространством на плате.
2. Широкая поддержка: Протоколы I2C и SPI широко распространены и поддерживаются множеством микроконтроллеров, микропроцессоров и других устройств. Это обеспечивает совместимость и возможность использования уже существующих компонентов и библиотек, что упрощает разработку и интеграцию системы.
3. Высокая скорость передачи данных: Протокол SPI обеспечивает высокую скорость передачи данных, поскольку он использует отдельные линии для данных и синхронизации. Это может быть важным фактором при загрузке больших объемов данных в FPGA.
4. Гибкость: Оба протокола, I2C и SPI, предлагают гибкость в настройке и расширении системы. Они поддерживают множество устройств на одной шине, позволяя коммуницировать с несколькими устройствами, включая FPGA, по одной линии связи.

Однако при выборе протокола для загрузки FPGA следует учитывать требования проекта, такие как скорость передачи данных, доступность поддерживаемых устройств и сложность реализации. Кроме того, следует также учесть возможность конфликтов с другими устройствами на шине, потребление энергии и надежность связи.

На базе полученных результатов можно рекомендовать использование в качестве метода загрузки FPGA способ Passive Serial, а также протоколы I2C и SPI для связи с периферийными микросхемами.

БИБЛИОГРАФИЯ

БИБЛИОГРАФИЯ

[1] Таненбаум Э., Остин Т., Архитектура компьютера. 6-е изд. - СПб.:Питер, 2017, стр. 69

[2] Архитектура процессоров https://ru.wikiversity.org/wiki/Архитектура\_процессоров\_(CISC \_RISC\_MISC)

[3]AD9889B, https://www.analog.com/en/products/ad9889b.html

[4]I2C, https://ru.wikipedia.org/wiki/I%C2%B2C

[5] Протоколы передачи данных. Протокол SPI,

https://varyag-nord.livejournal.com/64722.html

[6] Интерфейс передачи данных – SPI,

https://3d-diy.ru/wiki/arduino-moduli/interfeys-peredachi-dannykh-spi/

[7]Cyclone® IV Device Handbook, Volume 1, Chapter 1: FPGA Device Family Overview,

<https://www.intel.com/content/www/us/en/content-details/654630/cyclone-iv-device-handbook-volume-1-chapter-1-fpga-device-family-overview.html>

[8] Cyclone® IV Device Handbook, Volume 1, Chapter 8: Configuration and Remote System Upgrades,

<https://www.intel.com/content/www/us/en/content-details/654215/cyclone-iv-device-handbook-volume-1-chapter-8-configuration-and-remote-system-upgrades.html>

[9] Описание микросхемы atXmega,

<http://atxmega.narod.ru/>

[10] <http://ww1.microchip.com/downloads/en/DeviceDoc/doc7925.pdf>

[11] Мистер "Х" или ATXmega (на примере ATXmega32A4) быстрый старт

<https://we.easyelectronics.ru/AVR/mister-h-ili-atxmega-na-primere-atxmega32a4-bystryy-start.html>

[12] <https://ww1.microchip.com/downloads/en/DeviceDoc/doc8077.pdf>

Загрузка кофигурации в ПЛИС

[13] <https://habr.com/ru/articles/426131/>

ПЛИС

[14] <https://ru.wikipedia.org/wiki/%D0%9F%D0%9B%D0%98%D0%A1>

Описание шины I2C

[15] https://itt-ltd.com/reference/ref\_i2c.html

ДЕКЛАРАЦИЯ ОБ ОТВЕТСТВЕННОСТИ

Нижеподписавшаяся, заявляю под личную ответственность, что материалы, представленные в магистерской диссертации, являются результатом личных научных исследований и разработок. Осознаю, что в противном случае, буду нести ответственность в соответствии с действующим законодательством.

Абдуллаев Рамил

12.06.2023