

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

**Согласовано**

Профессор департамента  
программной инженерии  
факультета компьютерных наук  
канд. техн. наук

\_\_\_\_\_ Гринкруг Е. М.  
"    "    \_\_\_\_\_ 2017 г

**Утверждаю**

Академический руководитель  
образовательной программы  
«Программная инженерия»  
профессор департамента программной  
инженерии канд. техн. наук

\_\_\_\_\_ Шилов В. В.  
"    "    \_\_\_\_\_ 2017 г

**ПРОГРАММАТОР МИКРОКОНТРОЛЛЕРОВ PIC НА ОСНОВЕ ORANGE  
PI LITE**

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.509000 81 01-1

Студент группы БПИ 151 НИУ ВШЭ

\_\_\_\_\_ Абрамов А.М.

"    "    \_\_\_\_\_ 2017 г

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

2017

УТВЕРЖДЕНО  
RU.17701729.509000 81 01-1

## ПРОГРАММАТОР МИКРОКОНТРОЛЛЕРОВ PIC НА ОСНОВЕ ORANGE PI LITE

Пояснительная записка

RU.17701729.509000 81 01-1

Листов 32

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

2017

# Содержание

<b>1 Введение</b>	<b>3</b>
1.1 Наименование . . . . .	3
1.2 Краткая характеристика . . . . .	3
1.3 Документы, на основании которых ведется разработка . . . . .	3
<b>2 Назначение разработки</b>	<b>4</b>
2.1 Функциональное назначение . . . . .	4
2.2 Эксплуатационное назначение . . . . .	4
<b>3 Технические характеристики</b>	<b>5</b>
3.1 Постановка задачи на разработку программы . . . . .	5
3.2 Описание алгоритма и функционирования программы . . . . .	5
3.2.1 Выбор алгоритма . . . . .	5
3.2.2 Основные определения и структуры данных . . . . .	6
3.2.3 Описание алгоритма . . . . .	10
3.3 Метод организации входных и выходных данных . . . . .	16
3.3.1 Описание метода входных и выходных данных . . . . .	16
3.4 Выбор состава технических средств . . . . .	17
3.4.1 Состав технических и программных средств . . . . .	17
<b>4 Техничко-экономические показатели</b>	<b>19</b>
4.1 Ориентировочная экономическая эффективность . . . . .	19
4.2 Экономические преимущества разработки . . . . .	19
<b>5 Источники, используемые при разработке</b>	<b>20</b>
5.1 Список используемой литературы . . . . .	20
<b>6 Приложение 1. Терминология</b>	<b>21</b>
6.1 Терминология . . . . .	21
<b>7 Приложение 2. Формат INTEL HEX8M (.hex)</b>	<b>22</b>
7.1 Формат INTEL HEX8M (.hex) . . . . .	22
<b>8 Приложение 3. Описание классов</b>	<b>23</b>

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

8.1	Класс ActionState . . . . .	23
8.1.1	Подробное описание . . . . .	23
8.1.2	Методы . . . . .	23
8.2	Класс CameraDevice . . . . .	23
8.2.1	Подробное описание . . . . .	24
8.3	Класс DrawConfig . . . . .	24
8.3.1	Подробное описание . . . . .	25
8.4	Класс Entity . . . . .	25
8.4.1	Подробное описание . . . . .	25
8.4.2	Методы . . . . .	25
8.5	Класс Geometry . . . . .	25
8.5.1	Подробное описание . . . . .	26
8.6	Класс MeshDraw . . . . .	26
8.6.1	Подробное описание . . . . .	26
8.6.2	Конструктор(ы) . . . . .	26
8.6.3	Методы . . . . .	26
8.7	Класс MouseState . . . . .	27
8.7.1	Подробное описание . . . . .	27
8.8	Класс Renderer . . . . .	27
8.8.1	Подробное описание . . . . .	28
8.8.2	Методы . . . . .	28
8.9	Интерфейс ITransformState . . . . .	28
8.9.1	Подробное описание . . . . .	28

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

# 1. Введение

## 1.1. Наименование

Наименование: «Программатор микроконтроллеров PIC на основе Orange PI Lite».  
Наименование на английском: «Programmer for PIC Microcontrollers Based on Orange PI Lite».

## 1.2. Краткая характеристика

Цель работы - реализовать программатор для микроконтроллеров PIC серии 16F на тонком клиенте Orange PI Lite. В задачи работы входит расчет и инженерия электронной схемы для программирования, написание программы для управления этой схемой. Электронная схема предоставляет возможность подключить микроконтроллер PIC серии 16F к тонкому клиенту Orange Pi Lite, управлять уровнями вольтажа на 5В и на 3В и возможность проверить процесс программирования на светодиодах. Программа предоставляет пользователю командный и графический интерфейсы, чтение файлов INTEL HEX8M, возможность записать файлы программы в программную и EEPROM память микроконтроллера. В состав работы также входит создание демонстрационных исходных данных (файлов) для данного программатора и микроконтроллеров серии 16F.

Файл программы в формате INTEL HEX8M, удовлетворяющий требованиям входных данных, может быть получен в результате компиляции исходного кода одним из компиляторов для микроконтроллеров серии PIC 16F. Обычно для разработки используются пакеты предоставляющие интегрированную среду разработки. Например пакет MPLAB X (<https://www.microchip.com/>, разработчик: организация Microchip Ltd.)

## 1.3. Документы, на основании которых ведется разработка

Разработка программы ведется на основании приказа №6.18.1-02/1112-19 от 11.12.2016 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук» в соответствии с учебным планом подготовки бакалавров по направлению «Программная инженерия», факультета Компьютерных наук, Национального исследовательского университета «Высшая школа экономики»

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## 2. Назначение разработки

### 2.1. Функциональное назначение

Функциональным назначением программы и электронной схемы является предоставление пользователю возможности загрузить программу из файла INTEL HEX8M (.hex), проинтерпретировать полученную информацию, проверить ее на наличие ошибок, стереть программную память и EEPROM память микроконтроллера, записать прочитанные данные из файла в программную память микроконтроллера, записать новые данные в EEPROM память микроконтроллера без стирания программной памяти.

### 2.2. Эксплуатационное назначение

Программа и электронная схема предназначена для работы на тонком кленте Orange Pi Lite с операционной системой семейства Linux. Программа и схема могут использоваться в учебных целях для демонстрации основных компонентов необходимых для прошивки микроконтроллера. Они предоставляют новое направление использования тонкого клиента Orange Pi Lite. Ими может воспользоваться любой человек, желающий запрограммировать микроконтроллер, не имеющий на руках официального программатора, но у которого есть Orange Pi Lite. Данная программа и электронная схема могут использоваться в качестве дешевой, простой и быстрой альтернативы к покупке официального программатора.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

### 3. Технические характеристики

#### 3.1. Постановка задачи на разработку программы

Цель работы - реализовать программатор для микроконтроллеров PIC серии 16F на тонком клиенте Orange Pi Lite.

Задачи работы:

1. Чтение данных из формата INTEL HEX8M для хранения программы прошивки.
2. Возможность отдельной записи EEPROM памяти, не стирая программную память микроконтроллера.
3. Поддержка 3 линеек микроконтроллеров серии 16F: 627A / 628A / 648A.
4. Проверка входного файла на корректность.
5. Графический интерфейс для оперирования программой.
6. Интерфейс командной строки для оперирования программой.
7. Повышающий переходник с 3.3В на 5В для взаимодействия с микроконтроллером.
8. Схемотехника для платы которая позволяет подключить микроконтроллер к тонкому клиенту Orange Pi Lite.
9. Завершенные, работающие схемы на макетной плате.
10. Схемы разводки макетной платы для подключения микроконтроллера к Orange Pi Lite.

#### 3.2. Описание алгоритма и функционирования программы

##### 3.2.1. Выбор алгоритма

**Различные подходы** для программирования (или прошивки) микроконтроллеров варьируются в зависимости от кампании производителя. Данная курсовой работы нацелена на создание программатора для определенной серии и линейки микроконтроллеров определенного производителя. Микроконтроллер - микросхема, предназначенная для управления электронными устройствами. Микроконтроллер сочетает на одном кристалле функции микропроцессора, а также и функции периферийных устройств, содержит ОЗУ и ПЗУ. Это однокристальный компьютер, способный выполнять относительно простые задачи. Имеет смысл упомянуть две большие компании производящие микроконтроллеры общего назначения. А именно кампанию Microchip, производящую микроконтроллеры PIC и кампанию Atmel чьи микроконтроллеры ATmega легли в основу Arduino.

**Программирование** PIC16F627A/628A/648A производится с помощью серийного (последовательного) метода. Серийный режим позволяет PIC16F627A/628A/648A быть запрограммированным с использованием лишь 5 ножек микроконтроллера (или 6 ножек при режиме низковольтного программирования) уже будучи встроенным в систему пользователя. Это предоставляет большую гибкость в процессе программирования (позволяет пользователю более свободно выбирать «место» и «время» для программирования).

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

**Вольтаж режима программирования** определяет если будет использован низковольтный или высоковольтный режим. Использование низковольтного режима позволяет программировать PIC имея в доступности только источники питания от 2.0В до 5.0В, но требует дополнительной ножки микроконтроллера. Использование только высоковольтных режимов позволяет переопределить ножку PGM микроконтроллера под пользовательские нужды, но требует наличия источника питания на 12В. В данной работе используется низковольтный режим поскольку тонкий клиент Orange Pi Lite не имеет возможности предоставить 12В питание.

**Режим программирования** для PIC16F627A/628A/648A позволяет программировать ячейки программной памяти, ячейки памяти данных, конфигурационное слово, а также специальные 7 ячеек, которые используются для хранения ID устройства.

**Команды программирования** и их операнды в режиме программирования определяют обмен информацией с PIC. Операнды и команды передаются в микроконтроллер и/или обратно через серийные кабели.

### 3.2.2. Основные определения и структуры данных

**Ячейки памяти** в данном микроконтроллере есть как 14, так и 8 битные ячейки. На PIC16F627A/628A/648A реализованна Гарвардская архитектура с отдельными шинами для инструкций и данных, что позволяет 14-разрядным инструкциям работать с 8-разрядными данными.

**Пространство программной памяти** отведенное пользователю простирается от 0x0000 до 0x1FFF. В режиме программирования, пространство программной памяти простирается от 0x0000 до 0x3FFF, с первой половиной (от 0x0000-0x1FFF), которая отведена программной памяти, и второй половиной (0x2000-0x3FFF), которая отведена конфигурационной памяти. Все другие адреса в конфигурационной памяти PIC зарезервированы и не могут быть запрограммированы пользователем.

**В пространство конфигурационной памяти** (адреса 0x2000 - 0x2007) можно попасть через передачу в PIC специальной команды «Загрузить данные для конфигурационной памяти». Только адреса 0x2000-0x200F конфигурационного пространства памяти физически реализованы. Однако, только ячейки 0x2000 вплоть до 0x2007 доступны для программирования. Остальные ячейки зарезервированы. Переход по адресу за пределами 0x200F будет физически осуществлять доступ к пользовательской памяти.

**Пространство ПЗУ памяти** простирается от 0x00 до 0xFF и находится отдельно от пространства программной памяти и пространства оперативной памяти. Для ПЗУ реализуются только нижние 128 байт для устройств PIC16F627A/628A, в то время как для PIC16F648A реализуются все 256 байт. Программирование ПЗУ памяти данных использует тот же программный счетчик что и для программирования конфигурационной и программной памяти, однако только нижние биты декодируются и используются. Поэтому перед программированием ПЗУ необходимо чтобы программный счетчик указывал на 0x0000 или 0x2000.

**Создание структуры** для представления памяти микроконтроллера. Ключевым элементом в представлении памяти микроконтроллера является массив типа `uint16_t` размером с память PIC. В массиве хранится пользовательская программа непосред-

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата



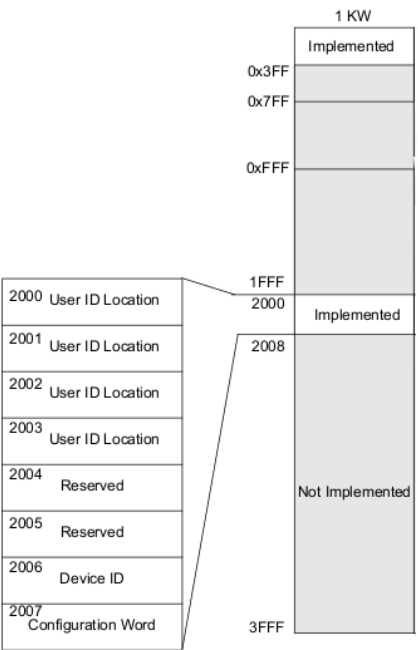


Рис. 1: Карта памяти микроконтроллера. Белым выделены области для пользовательского кода.

ственно перед её побитовой передачей на микроконтроллер. Из 16 битов отведенных под тип `uint16_t` для хранения данных и команд используются только нижние 8 и 14 байт соответственно.

Для того чтобы ускорить процесс прошивки микроконтроллера вводятся переменные `program_memory_used_cells`, и `program_memory_max_used_address`. Они обозначают общее количество задействованных программой ячеек памяти, и самый высокий адрес в программной памяти. Таким образом можно заметить условия при которых можно досрочно закончить программирование программной памяти PIC и перейти к следующей стадии программирования.

```
struct picmemory {
    uint16_t program_memory_used_cells;
    uint16_t program_memory_max_used_address;

    uint8_t has_configuration_data;
    uint8_t has_eeprom_data;

    uint16_t *data; /* 14-bit and 8-bit data */
    uint8_t *filled; /* 1 if this cell is used */
};
```

**Внутренний программный** счетчик может увеличиваться от 0x0000 до конца реализованной программной памяти 0x03FF (или 0x07FF, или 0x0FFF в зависимости от модели PIC) после чего он вновь обернется на адресс 0x0000. Для включения высоких бит программного счетчика и перехода к программированию конфигурационной памяти, необходимо послать специальную команду. После нее программный счетчик будет оперировать в

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

пространстве от 0x2000 до 0x3FFF (по достижении 0x3FFF он будет вновь обворачиваться на 0x2000). Единственным способом сбросить верхние биты программного счетчика вновь на 0x0000 это покинуть и повторно войти в режим программирования.

**Ячейки для ID** пользователя отображаются на адреса [0x2000 : 0x2003] и являются частью конфигуриционной памяти. Пользователь может хранить идентификационные данные (идентификатор пользователя) в четырех локациях для идентификатора пользователя. Этот идентификатор всегда можно считать корректно, даже если будет включена защита кода.

**Питание** в режиме программирования. Для PIC16F627A/628A/648A требуется один источник питания с VDD (2.0V до 5.5V) и VPP от 12В до 14В, или же VPP от 4.5В до 5.5В, при использовании низковольтного программирования. Оба источника должны иметь разрешение как минимум в 0,25В. В данной работе используется режим низковольтного программирования.

**Команды программирования** в режиме программирования, обмен информацией с PIC определяется набором команд и их операндами которые передаются в микроконтроллер и/или обратно через серийные кабели. Общая форма для всех последовательностей команд состоит из 6-битовой команды и условно 16-битного слова данных. И команды и слова данных передаются начиная с наименее значимого бита (LSB first). Для других серий и линеек типа PIC набор команд может отличаться.

Command	Mapping (MSb ... LSb)						Data
Load Configuration	X	X	0	0	0	0	0, data (14), 0
Load Data for Program Memory	X	X	0	0	1	0	0, data (14), 0
Load Data for Data Memory	X	X	0	0	1	1	0, data (8), zero (6), 0
Increment Address	X	X	0	1	1	0	
Read Data from Program Memory	X	X	0	1	0	0	0, data (14), 0
Read Data from Data Memory	X	X	0	1	0	1	0, data (8), zero (6), 0
Begin Programming Only Cycle	X	0	1	0	0	0	
Bulk Erase Program Memory	X	X	1	0	0	1	
Bulk Erase Data Memory	X	X	1	0	1	1	

Рис. 2: Последовательности 6-разрядных команд для PIC16F627A/628A/648A

**I/O Выходы** используемые для программирования. Положительный входной сигнал на ножке RB4 называемой PGM вводит микроконтроллер в режим низковольтного программирования (если данная опция была включена в конфигурационном слове микроконтроллера). Ножка RB7 называемая DATA, настраивается как вход и используется для по-битовой передачи данных программ в микроконтроллер. Ножка RB6 называемая CLOCK, также настраивается на прием входного сигнала и используется для синхронизации состояния напряжения на ножке RB7. Во время падения напряжения на ножке RB6, микроконтроллер считывает следующий бит с ножки RB7. Ножка MCLR/Vpp используется для выбора режима программирования. В PIC16F627A/628A/648A, высокое напряжение для работы с ячейками памяти генерируется автоматически. Для активации режима программирования, необходимо применить высокое напряжение ко входу MCLR. Поскольку MCLR используется на уровне источника, это означает, что MCLR не тянет какой-либо значительный ток. Ножка VDD предоставляет 5В необходимые для стабильной работы микроконтроллера в штатном режиме. Ножка VSS определяет напряжение на уровне земли.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

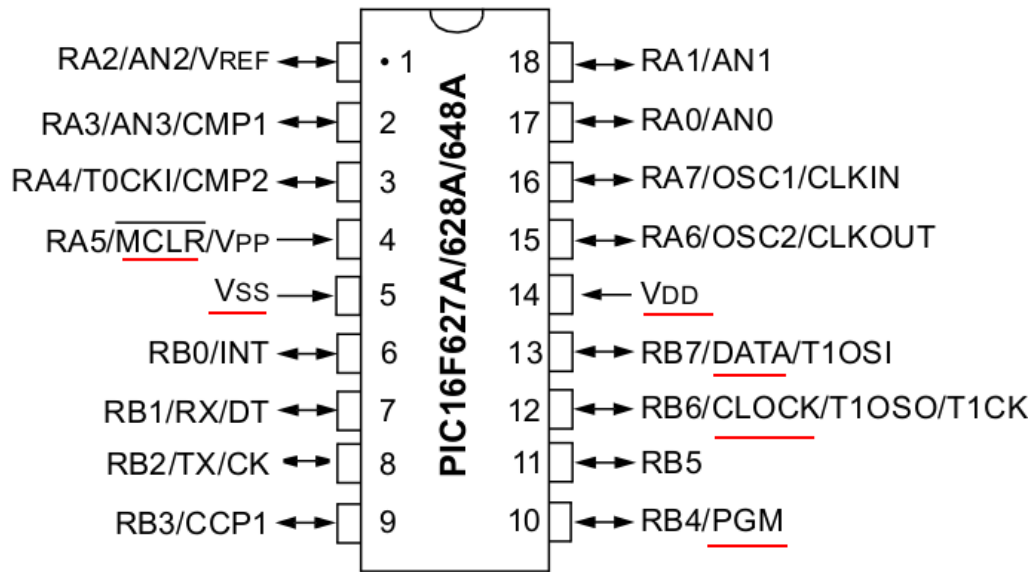
**PDIP, SOIC**

Рис. 3: I/O выходы необходимые для программирования в серийном режиме

**Абстрагирование программатора** от конкретной модели микроконтроллера семейства PIC делается с помощью определения дополнительной структуры в которой для каждой конкретной модели хранится информация о карте памяти, о командах программирования, о задержках необходимых между сигнальными последовательностями.

```
struct picmicro {
    uint16_t device_id;
    char name[16];
    size_t program_memory_size;
    size_t data_memory_size;

    int program_cycle_time; /* in microseconds */ // T_PROG
    int eeprom_program_cycle_time; // T_DPROG
    int bulk_erase_cycle_time; // T_ERA

    uint8_t load_configuration_cmd;
    uint8_t load_data_for_program_memory_cmd;
    uint8_t load_data_for_data_memory_cmd;
    uint8_t read_data_from_program_memory_cmd;
    uint8_t read_data_from_data_memory_cmd;
    uint8_t increment_address_cmd;
    uint8_t begin_erase_programming_cycle_cmd;
    uint8_t begin_programming_only_cycle_cmd;
    uint8_t bulk_erase_program_memory_cmd;
    uint8_t bulk_erase_data_memory_cmd;
};
```

Для микроконтроллеров PIC16F627A/628A/648A используется нижеследующее

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

определение экземпляра данной структуры.

```
const struct picmicro pic16f628a = {
    /* General */
    .device_id =          0x1060,
    .name =              "pic16f628a",
    .program_memory_size = 0x800,
    .data_memory_size =  128,

    /* Time intervals in microseconds */
    .program_cycle_time = 4000,
    .eeprom_program_cycle_time = 6000,
    .bulk_erase_cycle_time = 6000,

    /* Commands */
    .load_configuration_cmd = 0x00,
    .load_data_for_program_memory_cmd = 0x02,
    .load_data_for_data_memory_cmd = 0x03,
    .read_data_from_program_memory_cmd = 0x04,
    .read_data_from_data_memory_cmd = 0x05,
    .increment_address_cmd = 0x06,
    .begin_erase_programming_cycle_cmd = 0xFF,
    .begin_programming_only_cycle_cmd = 0x08,
    .bulk_erase_program_memory_cmd = 0x09,
    .bulk_erase_data_memory_cmd = 0x0B
};
```

### 3.2.3. Описание алгоритма

**Чтение файла** с программой пользователя, это первый шаг программатора. Следующим шагом является проверка входного файла на корректность. Для этого используются особенности формата INTEL HEX8M. В частности приведенные в конце строки проверочные суммы и адреса для данных.

```
1 :02000000FE2FD1
2 :10001000FF3FFF3FFF3FFF3FFF3FFF3FFF3FFF3D0
3 :100FD4000730831203139F008501831603138501D1
4 :100FE400831203138601831603138601FF308312D1
5 :0C0FF4008500FF308600F82F8301EA2FF3
6 :02400E00D0FFE1
7 :00000001FF
```

Рис. 4: Пример входного файла к программатору

По мере чтения входного файла в соответствующий адрес в массиве структуры «picmemory» вставляются значения из входного файла. Поскольку на тонком клиенте Orange Pi Lite установлен little-endian процессор компании Allwinner модели H3, а в формате INTEL HEX8M данные записываются в порядке MSB, то при вставке в массив необходимо поменять местами верхний и нижний байты.

В цикле происходит прочтение входного файла и создание структуры для представления памяти микроконтроллера, а также подсчет проверочных значений.

```
for (i = 0; i < byte_count / 2; i++) {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

```

nread = sscanf(&line[9+4*i], "%4hx", &data);
uint16_t pic_data = swap_uint16(data);
if (nread != 1) {
    fprintf(stderr, "Error: cannot read data.\n");
    free_picmemory(&pm);
    return NULL;
}
if (debug)
    fprintf(stderr, " data      = 0x%04X (file) = 0x%04X (micro)\n", data, pic_data);
checksum_calculated += (data >> 8) & 0xFF;
checksum_calculated += data & 0xFF;

if (address + i < 0x2000) {
    pm->program_memory_used_cells += 1;
    pm->program_memory_max_used_address = address + i;
} else if (0x2000 <= address + i && address + i < 0x2008)
    pm->has_configuration_data = 1;
else if (address + i >= 0x2100)
    pm->has_eeprom_data = 1;

pm->data[address + i] = pic_data;
pm->filled[address + i] = 1;
}

```

Программатор работает через посылание последовательности команд и данных, введенных в серийном режиме в котором бит на линии данных загоняется в микроконтроллер на падающем фронте напряжения на линии часов. Команда + данные, вводятся последовательно, через линию часы и линию данных, которые с аппаратной точки зрения являются входными линиями использующими триггеры Шмитта для различения напряжения 0 или 1.

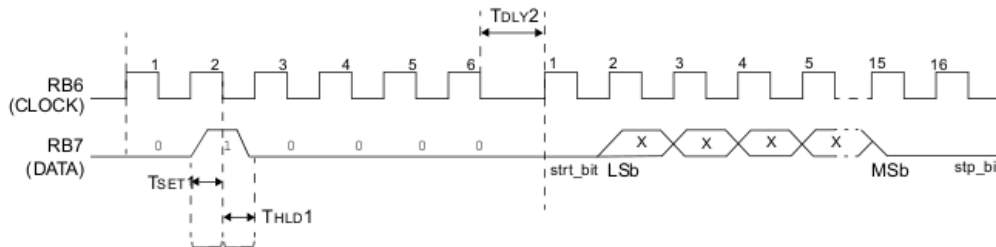


Рис. 5: Временные интервалы для побитовой передачи команды «Загрузка данных в программную память»

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

**Минимальные время** установки и удержания приписываются каждому сигнал на ножке данных (описанные в таблице ниже, вместе с ограничением по напряжению) по отношению к падающему фронту напряжения на линии часов. Командам на чтение и запись, которые требуют передачи данных связанных с ними, требуется минимальная задержка между передачей команды и передачей данных.

AC/DC Characteristics		Standard Operating Conditions (unless otherwise stated) Operating Temperature: $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ Operating Voltage: $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$				
Characteristics	Sym	Min	Typ	Max	Units	Conditions/ Comments
<b>General</b>						
VDD level for word operations, program memory	VDD	2.0	—	5.5	V	
VDD level for word operations, data memory	VDD	2.0	—	5.5	V	
VDD level for Bulk Erase operations, program and data memory	VDD	4.5	—	5.5	V	
High voltage on $\overline{\text{MCLR}}$	$V_{IH}$	10.0	—	13.5	V	
$\overline{\text{MCLR}}$ rise time ( $V_{SS}$ to $V_{IH}$ ) for Programming mode entry	$T_{VHR}$	—	—	1.0	$\mu\text{s}$	
Hold time after $\overline{\text{MCLR}}\uparrow$	$T_{PPDP}$	5	—	—	$\mu\text{s}$	
Hold time after $\text{LVP}\uparrow$	$T_{LVPP}$	5	—	—	$\mu\text{s}$	
(CLOCK, DATA) input high level	$V_{IH1}$	0.8 VDD	—	—	V	Schmitt Trigger input
(CLOCK, DATA) input low level	$V_{IL1}$	—	—	0.2 VDD	V	Schmitt Trigger input
CLOCK, DATA setup time before $\overline{\text{MCLR}}\uparrow$	$T_{SET0}$	100	—	—	ns	
Hold time after $V_{DD}\uparrow$	$T_{HLD0}$	5	—	—	$\mu\text{s}$	
<b>Serial Program/Verify</b>						
Data in setup time before clock $\downarrow$	$T_{SET1}$	100	—	—	ns	
Data in hold time after clock $\downarrow$	$T_{HLD1}$	100	—	—	ns	
Data input not driven to next clock input (delay required between command/data or command/command)	$T_{DLY1}$	1.0	—	—	$\mu\text{s}$	
Delay between clock $\downarrow$ to clock $\uparrow$ of next command or data	$T_{DLY2}$	1.0	—	—	$\mu\text{s}$	
Clock $\uparrow$ to data out valid (during read data)	$T_{DLY3}$	—	—	80	ns	
Programming cycle time	$T_{PROG}$	—	—	4	ms	
Data EEPROM Programming cycle time	$T_{DPROG}$	—	—	6	ms	
Bulk Erase cycle time	$T_{ERA}$	—	—	6	ms	

Рис. 6: Спецификация AC/DC для поднятия/опускания и удержания линий данных и часов

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

**Алгоритм** записи данных в память программы приведен в нижеследующем графе.

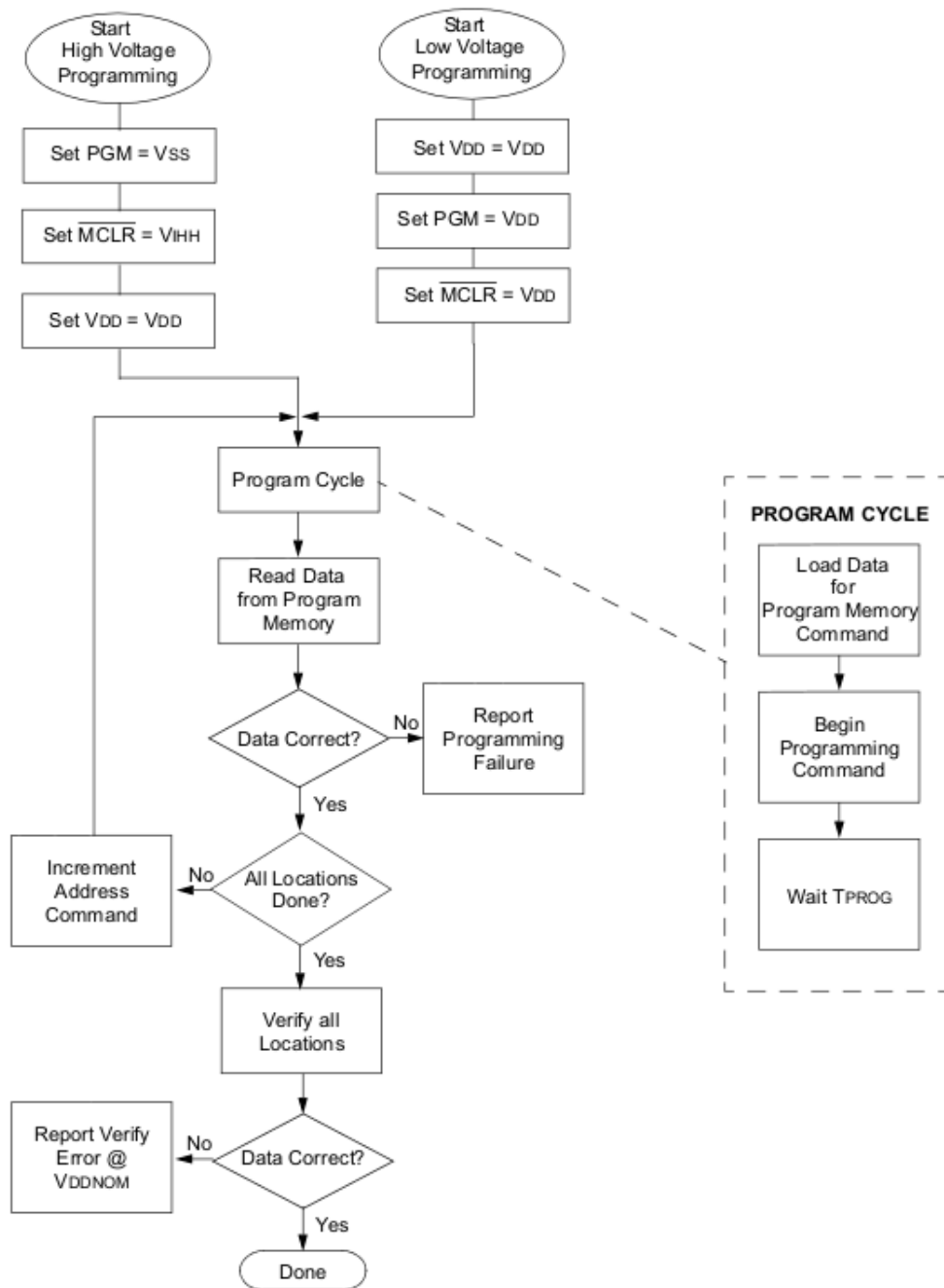


Рис. 7: Граф алгоритма записи данных в программную память микроконтроллера

**После создания** карты памяти которая должна быть загружена в микроконтроллер, следующим шагом является непосредственная передача этих данных в PIC. Для этого необходимо получить доступ к GPIO выходам тонкого клиента Orange Pi Lite. Со стороны Orange Pi Lite надо было посмотреть на поставляемую с ним схемотехнику платы чтобы определить какие выходы GPIO наиболее подходят для работы программатора. Сигналы

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

передаваемые на ножки микроконтроллера имеют строгие временные рамки, и для того что бы в них вписаться необходимо использовать механизм «memory mapped file» для включения и отключения сигналов на выходах GPIO на уровне процессора.

Посмотрев на схемотехнику для выходов GPIO, решено было выбрать 5 последовательных выходов 31, 33, 35, 37, 39 которые подсоединены к процессору H3 соответственно на ножках 7, 8, 9, 10, 20 порта A.

**Получение доступа** к выходам GPIO, осуществляется через манипуляцию регистрами отвечающими за порт A напрямую. Для этого делается маппинг регистров настройки и состояния порта A в адресное пространство программы. Значения для адресов регистров и их размера берутся из руководства инженера к процессору H3.

В итоге удалось найти адреса всех регистров влияющих на состояние порта A. Далее приведена краткая сводка констант в которые были записаны необходимые адреса.

```
// AllWinner H3 datasheet, page 86, page 316
uint32_t CCU_BASE = 0x01C20000ul;    // Port controller
uint32_t PIO_OFS = 0x00000800ul;    // GPIO offset

// AllWinner H3 datasheet, page 86, block size is 1024 = 1K = 0x400 bytes
uint32_t PIO_MAP_LEN = 0x2000;      // Port controller end (0x01C20BFF) - Port controller start (0x01C20800)

// AllWinner H3 datasheet, page 316
uint32_t PA_CFG0_OFS = 0x00000000ul; // Port A Configure Register 0
uint32_t PA_CFG1_OFS = 0x00000004ul; // Port A Configure Register 1
uint32_t PA_CFG2_OFS = 0x00000008ul; // Port A Configure Register 2
uint32_t PA_DAT_OFS = 0x00000010ul;  // Port A Data Register
```

Следом были написаны функции для манипулирования этими регистрами. Ниже приведены фрагменты функций которые используются чтобы выставить состояние ножки GPIO на выход (output) и чтобы менять напряжение на данной ножке с 0В до 3.3В. Для регистров бит 0 - это наименее значимый бит.

Ниже приведены фрагменты функций которые используются чтобы выставить состояние ножки GPIO на выход (output) и чтобы менять напряжение на данной ножке с 0В до 3.3В. Для регистров бит 0 - это наименее значимый бит.

```
// получение указателя на адрес по которому хранится регистр
volatile uint32_t* get_data_reg(int pin)
{
    volatile uint32_t* data_reg = (volatile uint32_t*)(gpio + PA_DAT_OFS);
    return data_reg;
}

// создает новое измененное значение для записи в регистр
uint32_t modify_data(uint32_t data, int pin, int value)
{
    uint32_t new_data;
    if (value) {
        new_data = data | (1U << pin);
    }
    else {
        new_data = data & ~(1U << pin);
    }
    return new_data;
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата



```

}

// выставляет на ножке номер "pin" значение напряжения 0 или 1.
void gpio_wr(int pin, int value) {
    volatile uint32_t* data_reg = get_data_reg(pin);
    uint32_t data = *data_reg;
    uint32_t new_data = modify_data(data, pin, value);
    *(data_reg) = new_data;
}

// модифицирует копию текущих настроек чтобы ножка номер "pin" оказалась настроена как ВЫХОД
// возвращает новое значение которое должно быть записано в регистр
uint32_t modify_cfg_output(uint32_t cfg, int pin)
{
    /* Write value 001 to position of pin, each pin is z001 ('z' is dont care) */
    uint32_t new_cfg = cfg & ~(6U << 4 * (pin % 8));
    uint32_t new_new_cfg = new_cfg | (1U << 4 * (pin % 8));
    return new_new_cfg;
}

// модифицирует регистр настроек порта А чтобы ножка номер "pin" оказалась настроена как ВЫХОД
// Pin is one of PA7, PA8, PA9, PA10, PA20
void gpio_output(int pin)
{
    volatile uint32_t* cfg_reg = get_cfg_reg(pin);
    uint32_t cfg = *cfg_reg;
    uint32_t new_cfg = modify_cfg_output(cfg, pin);
    *(cfg_reg) = new_cfg;
}

// модифицирует копию текущих настроек чтобы ножка номер "pin" оказалась настроена как ВХОД
// возвращает новое значение которое должно быть записано в регистр
uint32_t modify_cfg_input(uint32_t cfg, int pin)
{
    /* Write value 000 to position of pin, each pin is z000 ('z' is dont care) */
    uint32_t new_cfg = cfg & ~(7U << 4 * (pin % 8));
    return new_cfg;
}

// модифицирует регистр настроек порта А чтобы ножка номер "pin" оказалась настроена как ВХОД
// Pin is one of PA7, PA8, PA9, PA10, PA20
void gpio_input(int pin)
{
    volatile uint32_t* cfg_reg = get_cfg_reg(pin);
    uint32_t cfg = *cfg_reg;
    uint32_t new_cfg = modify_cfg_input(cfg, pin);
    *(cfg_reg) = new_cfg;
}

```

**Последним шагом** происходящим на уровне программы необходимо пройти по созданной карте памяти микроконтроллера и посылая в PIC команд и их операнды через GPIO передать в микроконтроллер необходимую информацию.

```

// Bulk erase the chip, and then write contents of the .hex file to the PIC
void pic_write(const struct picmicro *pic, char *infile, int debug)
{

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

```
uint16_t addr;
struct picmemory *pm;

pm = read_inhx16(infile, debug);

/* Turn pic on. Give supply power. */
pic_enter_lvp();

/* Bulk erase the chip first */
pic_bulk_erase(pic, debug);

/* Write Program Memory */
for (addr = 0; addr <= pm->program_memory_max_used_address; addr++) {
    if (pm->filled[addr]) {
        pic_send_cmd(pic->load_data_for_program_memory_cmd);
        pic_load_data(pm->data[addr]);
        // trigger
        pic_send_cmd(pic->begin_programming_only_cycle_cmd);
        usleep(pic->program_cycle_time);
    }
    pic_send_cmd(pic->increment_address_cmd);
}

/* Write Configuration Memory */
if (pm->has_configuration_data) {
    pic_send_cmd(pic->load_configuration_cmd);
    pic_load_data(pm->data[0x2000]);
    for (addr = 0x2000; addr < 0x2008; addr++) {
        if ((addr <= 0x2003 || addr == 0x2007) && pm->filled[addr]) {
            pic_send_cmd(pic->load_data_for_program_memory_cmd);
            pic_load_data(pm->data[addr]);
            // trigger
            pic_send_cmd(pic->begin_programming_only_cycle_cmd);
            usleep(pic->program_cycle_time);
        }
        pic_send_cmd(pic->increment_address_cmd);
    }
}

pic_exit_lvp();
free_picmemory(&pm);
}
```

### 3.3. Метод организации входных и выходных данных

#### 3.3.1. Описание метода входных и выходных данных

Входными данными для работы программатора являются скомпилированный файл программы, микроконтроллер подключенный к плате, а также (для обеспечения взаимодействия с пользователем) клавиатура и/или мышь. Входной файл данных может быть создан в любой среде разработки и любым компилятором поддерживающим формат INTEL HEX8M. Примером такой среды разработки является MPLAB X (<https://www.microchip.com/>, разработчик: коммерческая организация Microchip Ltd.).

1. Из-за огромного количества серий микроконтроллеров поддерживать их все не пред-

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

ставляется возможным. Поэтому программатор работает только с микроконтроллерами PIC серии 16F, конкретно с линейками 627A / 628A / 648A.

2. Входной файл программы должен соответствовать формату INTEL HEX8M. По сравнению с двумя другими часто встречающимися форматами INTEL HEX8S, INTEL HEX32, данный формат наиболее оптимально подходит под серию 16F. В силу того что память 14-битных микроконтроллеров не превышает 64 килобайт (здесь подходит формат HEX32) и программное слово не нуждается в разбиении на высокий и низкий байт как в 16-битных микроконтроллерах (здесь подходит формат HEX8S).
3. Пользователь имеет возможность модифицировать следующие входные данные в процессе работы программы в условиях графического интерфейса и перед запуском программы в командной строке:
  - (а) Указать что требуется запись EEPROM памяти без модификации программной памяти микроконтроллера.
  - (б) Указать что требуется проверить входной файл на ошибки.
  - (с) Указать что требуется записать входной файл в программную память и в EEPROM памяти микроконтроллера.
  - (д) Поменять уровень количества сообщений выводимых программой пользователю.
  - (е) Отменить процесс программирования.

Выходными данными для программатора является запрограммированный микроконтроллер, данные на экране и индикатор программирования на плате программатора.

### 3.4. Выбор состава технических средств

#### 3.4.1. Состав технических и программных средств

Для оптимальной работы приложения необходимы следующие системные требования:

1. Тонкий клиент Orange Pi Lite, оснащенный:
  - (а) Обязательно процессор Allwinner H3 с тактовой частотой 1 гигагерц (ГГц) или выше;
  - (б) 0.5 ГБ оперативной памяти (ОЗУ);
  - (с) 0.5 ГБ свободного места на жестком диске;
  - (д) Периферия: выход GPIO типа Raspberry Pi B+
2. Опционально: Компьютер для удаленного доступа к Orange Pi Lite, оснащенный:
  - (а) Обязательно 64-разрядный (x64) процессор с тактовой частотой 1 гигагерц (ГГц) или выше;
  - (б) 1 ГБ оперативной памяти (ОЗУ);
  - (с) 1.5 ГБ свободного места на жестком диске;
  - (д) Wifi модулем (если Orange Pi Lite подключен к сети, то можно воспользоваться и стандартным Ethernet портом) или TTL переходником для подключения к тонкому клиенту Orange Pi Lite.
3. Монитор

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

- 4. Мышь
- 5. Клавиатура

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## 4. Техничко-экономические показатели

### 4.1. Оринтировочная экономическая эффективность

Оринтировочная экономическая эффективность не рассчитывается.

### 4.2. Экономические преимущества разработки

Существующими коммерческими аналогами данного приложения являются серийные программаторы компании Microchip Ltd. В силу того что схематехника и код данного программатора распространяется бесплатно, экономически выгодным аналогом к нему являются известный программатор K150 (подключающийся к компьютеру через USB), а также разнообразные схемы основанные на портах LDP (для принтера) и COM портах. Однако схематехника K150 в несколько раз сложнее, к тому же для его работы требуются два уже заранее запрограммированных микроконтроллера, поэтому собрать его в домашних условиях невозможно (требуется приобрести 2 уже запрограммированных микроконтроллера). Варианты основанные на портах LDP и COM также не подходят поскольку эти порты морально устарели и все реже и реже присутствуют на современных компьютерах. Данная разработка позволяет использовать современный и актуальный тонкий клиент Orange Pi Lite в качестве программатора микроконтроллеров, к тому же все части для сборки программатора стоят гораздо дешевле чем покупка официального прибора от Microchip, и наконец они предельно просты и позволяют сборку программатора в домашних условиях.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## 5. Источники, используемые при разработке

### 5.1. Список используемой литературы

1. DS40044G PIC16F627A/628A/648A Data Sheet  
Microchip Ltd.  
[www.microchip.com](http://www.microchip.com)
2. DS33014L MPASM Assembler, MPLINK Object Linker, MPLIB Object Librarian User's Guide  
Microchip Ltd.  
[www.microchip.com](http://www.microchip.com)
3. DS33023A PICmicro Mid-Range MCU Family Reference Manual  
Microchip Ltd.  
[www.microchip.com](http://www.microchip.com)
4. DS41196G PIC16F627A/628A/648A EEPROM Memory Programming Specification  
Microchip Ltd.  
[www.microchip.com](http://www.microchip.com)
5. Schematics Index ORANGE PI Lite Ver 1.1  
Orange Pi  
[www.orangepi.com](http://www.orangepi.com)
6. Version 1.2 Allwinner H3 Datasheet Quad-Core OTT Box Processor  
Allwinner Ltd.  
[www.allwinnertech.com](http://www.allwinnertech.com)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## 6. Приложение 1. Терминология

### 6.1. Терминология

**EEPROM** электрически стираемое перепрограммируемое ПЗУ (ЭСППЗУ), один из видов энергонезависимой памяти (таких, как PROM и EPROM). Память такого типа может стираться и заполняться данными до миллиона раз.

**Архитектура набора команд (англ. instruction set architecture, ISA)** часть архитектуры компьютера, определяющая программируемую часть ядра микропроцессора. На этом уровне определяются реализованные в микропроцессоре конкретного типа

**Язык ассемблера (англ. assembly language)** машинно-ориентированный язык низкого уровня с командами, не всегда соответствующими командам машины, который может обеспечить дополнительные возможности вроде макрокоманд.

**MPLAB** интегрированная среда разработки, представляющая собой набор программных продуктов, предназначенная для облегчения процесса создания, редактирования и отладки программ для микроконтроллеров семейства PIC, производимых компанией Microchip Technology. Среда разработки состоит из отдельных приложений, связанных друг с другом и включает в себя компилятор с языка ассемблер, текстовый редактор, программный симулятор и средства работы над проектами, также среда позволяет использовать компилятор с языка C.

**контрольный таймер, англ. Watchdog timer** аппаратно реализованная схема контроля над зависанием системы. Представляет собой таймер, который периодически сбрасывается контролируемой системой. Если сброса не произошло в течение некоторого интервала времени, происходит принудительная перезагрузка системы. В некоторых случаях сторожевой таймер может посылать системе сигнал на перезагрузку («мягкая» перезагрузка), в других же — перезагрузка происходит аппаратно (замыканием сигнального провода RST или подобного ему).

**Внутрисхемное программирование (англ. In-System Programming, сокр. ISP)** технология программирования электронных компонентов (ПЛИС, микроконтроллеры и т. п.), позволяющая программировать компонент, уже установленный в устройство. До появления этой технологии компоненты программировались перед установкой в устройство, для их перепрограммирования требовалось их извлечение из устройства.

**Универсальный асинхронный приёмопередатчик (англ. UART)** узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству. Метод преобразования хорошо стандартизован и широко применяется в компьютерной технике (особенно во встраиваемых устройствах и системах на кристалле (SoC)).

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## 7. Приложение 2. Формат INTEL HEX8M (.hex)

### 7.1. Формат INTEL HEX8M (.hex)

INTEL HEX8M — это формат, разработанный для обмена скомпилированным кодом но не в бинарном а в текстовом формате. В формате указывается связь между адресом и байтовым значением, которое должно находиться по этому адресу.

Формат включает в себя подсчет проверочной суммы, котоая записывается последним байтом в конце строки.

Ниже приведен пример описания простой прогаммы для мигания светодиодом на микроконтроллере PIC16F628A в данном формате:

```
:1000000000000000000000000000000000F0  
:040010000000000000EC  
:1000320000000280040006800A800E800C80028016D  
:100042006801A9018901EA01280208026A02BF02C5  
:10005200E002E80228036803BF03E803C8030804B8  
:1000620008040804030443050306E807E807FF0839  
:06007200FF08FF08190A57  
:00000001FF
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата



## 8. Приложение 3. Описание классов

### 8.1. Класс ActionState

This class knows what arguments to pass to NodeInterpolator.

Базовые классы: BaseForEventDriven.

Открытые члены

- void **SetCurrentAction** (Animation action)  
*Change the animation track. If there is more than one. We don't support this yet.*
- void **SetTime** (double time\_seconds)

Открытые атрибуты

- bool **\_loop**  
*Automatically play the animation again after it has timed out.*

Свойства

- double **TimeCursorInTicks** [get]  
*position of the time cursor in ticks of animation.*
- double **TotalDurationSeconds** [get]  
*Duration of animation.*

#### 8.1.1. Подробное описание

This class knows what arguments to pass to NodeInterpolator.

#### 8.1.2. Методы

**8.1.2.1 void SetTime ( double time\_seconds )** Set the current time for the animation. Note: all the calculations here are done in ticks.

### 8.2. Класс CameraDevice

Maintains camera abstraction. Allows support for orbiting, free fly and even 2D camera.

Открытые члены

- **CameraDevice** (Matrix4 opengl\_init\_mat)  
*Constructor.*
- Vector3 **ConvertScreen2WorldCoordinates** (Point screen\_coords)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

*Get the mouse position and calculate the world coordinates based on the screen coordinates.*

- **Matrix4 MatrixToOpenGL ()**

*Get the camera matrix to be uploaded to drawing 2D.*

- **void OnMouseMove (int x, int y)**

*Respond to mouse events.*

- **void Scroll (float scroll)**

*Zoom in/out of the scene.*

## Свойства

- **CamMode \_cam\_mode** [get]

*Return the currently active camera mode.*

- **Vector3 GetTranslation** [get]

*Get the translation part of the camera matrix.*

### 8.2.1. Подробное описание

Maintains camera abstraction. Allows support for orbiting, free fly and even 2D camera.

## 8.3. Класс DrawConfig

This class will be passed into the **Entity** (стр. 25) GetSettings() function to make the scene look best.

## Открытые атрибуты

- **bool EnableDepthTest = false**

*Enable and disable OpenGL functionality.*

- **bool EnableDisplayList = false**

*Enable and disable OpenGL functionality.*

- **bool EnableFaceCounterClockwise = false**

*Enable and disable OpenGL functionality.*

- **bool EnableLight = false**

*Enable and disable OpenGL functionality.*

- **bool EnablePerspectiveCorrectionHint = false**

*Enable and disable OpenGL functionality.*

- **bool EnablePolygonModeFill = false**

*Enable and disable OpenGL functionality.*

- **bool EnablePolygonModeLine = false**

*Enable and disable OpenGL functionality.*

- **bool EnableTexture2D = false**

*Enable and disable OpenGL functionality.*

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

### 8.3.1. Подробное описание

This class will be passed into the **Entity** (стр. 25) `GetSettings()` function to make the scene look best.

## 8.4. Класс Entity

Represents the currently loaded object. One day we will have lots of these.

### Открытые члены

- void **RecursiveCalculateVertexTransform** (Node nd, Matrix4x4 current)  
*Find the appropriate matrix to apply to the given vertex.*
- void **RenderModel** (**DrawConfig** settings)  
*Render the model stored in EntityScene useing the **DrawConfig** (стр. 24) settings object.*
- void **UpdateModel** (double dt\_ms)  
*Deform the model vertices to align with the skeleton.*

### Открытые статические члены

- static void **TransformPositionAssimp** (ref Vector3D pos, ref Matrix4x4 mat, out Vector3D result)  
*Transform a Position by the given Matrix. Based on openTK compatiability vector 3 class.*

### 8.4.1. Подробное описание

Represents the currently loaded object. One day we will have lots of these.

### 8.4.2. Методы

**8.4.2.1 static void TransformPositionAssimp ( ref Vector3D pos, ref Matrix4x4 mat, out Vector3D result )** [static] Transform a Position by the given Matrix. Based on openTK compatiability vector 3 class.

### Аргументы

<i>pos</i>	The position to transform
<i>mat</i>	The desired transformation
<i>result</i>	The transformed position

## 8.5. Класс Geometry

Stores info on extra geometry of the entity, bones that is.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

#### Открытые члены

- **Geometry** (IList< Mesh > scene\_meshes, Node nd, BoneNode armature)  
*Build geometry data for node (usually use only for one of the children of scene.RootNode)*
- double **FindAverageBoneLength** (BoneNode nd)  
*For the length of final children bones. Just use average length.*
- void **UpdateBonePositions** (BoneNode nd)  
*Snap the render positions of bones, to deformations in the skeleton.*

#### Открытые атрибуты

- Dictionary< string, BoneBounds > **\_bone\_id2triangle** = new Dictionary<string,Bone↵  
Bounds>()  
*Bone name matched up with the triangle to render.*

#### 8.5.1. Подробное описание

Stores info on extra geometry of the entity, bones that is.

### 8.6. Класс MeshDraw

Mesh rendering using VBOs. Based on [http://www.opentk.com/files/T08\\_VBO.cs](http://www.opentk.com/files/T08_VBO.cs)

#### Открытые члены

- **MeshDraw** (Mesh mesh, IList< Material > materials)  
*Uploads the data to the GPU.*
- void **BeginModifyVertexData** (out IntPtr data, out int qty\_vertices)  
*Call this to get a pointer to OpenGL private memory buffer.*
- void **EndModifyVertexData** ()  
*Call this when done working with OpenGL memory. This uploads it back into OpenGL.*
- void **RenderVBO** ()  
*Render mesh from GPU memory.*

#### 8.6.1. Подробное описание

Mesh rendering using VBOs. Based on [http://www.opentk.com/files/T08\\_VBO.cs](http://www.opentk.com/files/T08_VBO.cs)

#### 8.6.2. Конструктор(ы)

**8.6.2.1 MeshDraw ( Mesh mesh, IList< Material > materials )** Uploads the data to the GPU.

#### 8.6.3. Методы

**8.6.3.1 void RenderVBO ( )** Render mesh from GPU memory.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## 8.7. Класс MouseState

Simple class to store mouse status data. Monitor mouse status (delta, position, click\_position, etc.)

Открытые члены

- void **RecordMouseClicked** (MouseEventArgs e)  
*Updates mouse click position.*
- void **RecordMouseMove** (MouseEventArgs e)  
*Updates current mouse position. Then we can calculate delta better.*

Открытые атрибуты

- Point **ClickPos**  
*Captured mouse position when it was clicked.*
- Point **CurrentPos**  
*Current mouse position, this is updated by you.*
- readonly int **HorizHysteresis** = 4  
*Minimum motion delta for mouse to be recognised.*
- OpenTK.Vector3 **InnerWorldClickPos**  
*Position of click inside the game world.*
- OpenTK.Vector3 **InnerWorldPos**  
*Position of where the user is pointing inside the game world.*

### 8.7.1. Подробное описание

Simple class to store mouse status data. Monitor mouse status (delta, position, click\_position, etc.)

## 8.8. Класс Renderer

Class to control openGL settings and do the actual drawing. All openGL calls will be here.

Открытые члены

- void **ClearOpenGLFrameForRender** (Matrix4 camera\_matrix)  
*Prepare to render next OpenGL frame. Clear depth/color buffers.*
- void **DrawAxis3D** ()  
*Important points to remember: Set normals. Must be clock wise vertex draw order The x-axis is accross the screen, so the Z-axis triangle must have component along X: +-1 since look at looks towards the center, we need to offset it a bit to see the Z axis.*
- void **InitOpenGL** ()  
*Enable default OpenGL settings. Set lights, material, etc. Call this once in the beginning.*
- void **ResizeOpenGL** (int width, int height)  
*Called when window size changes.*

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

### 8.8.1. Подробное описание

Class to control openGL settings and do the actual drawing. All openGL calls will be here.

### 8.8.2. Методы

**8.8.2.1 void DrawAxis3D ( )** Important points to remember: Set normals. Must be clock wise vertex draw order The x-axis is accross the screen, so the Z-axis triangle must have component along X: +-1 since look at looks towards the center, we need to offset it a bit to see the Z axis.

## 8.9. Интерфейс ITransformState

Implement this when class allows local matrix transforms. (**Entity** (стр. 25), Camera)

Производные классы: CameraDrawing2D и CameraFreeFly3D.

### Открытые члены

- void **MoveBy** (Vector3 direction)  
*x,y,z should be direction parameters, one of {-1, 0, 1}. Called on keyboard events.*
- void **RotateBy** (double angle\_degrees)  
*Rotate by angle around default axis. Called on mouse events.*

### Свойства

- Vector3 **GetTranslation** [get]  
*Get the translation part of the matrix.*

### 8.9.1. Подробное описание

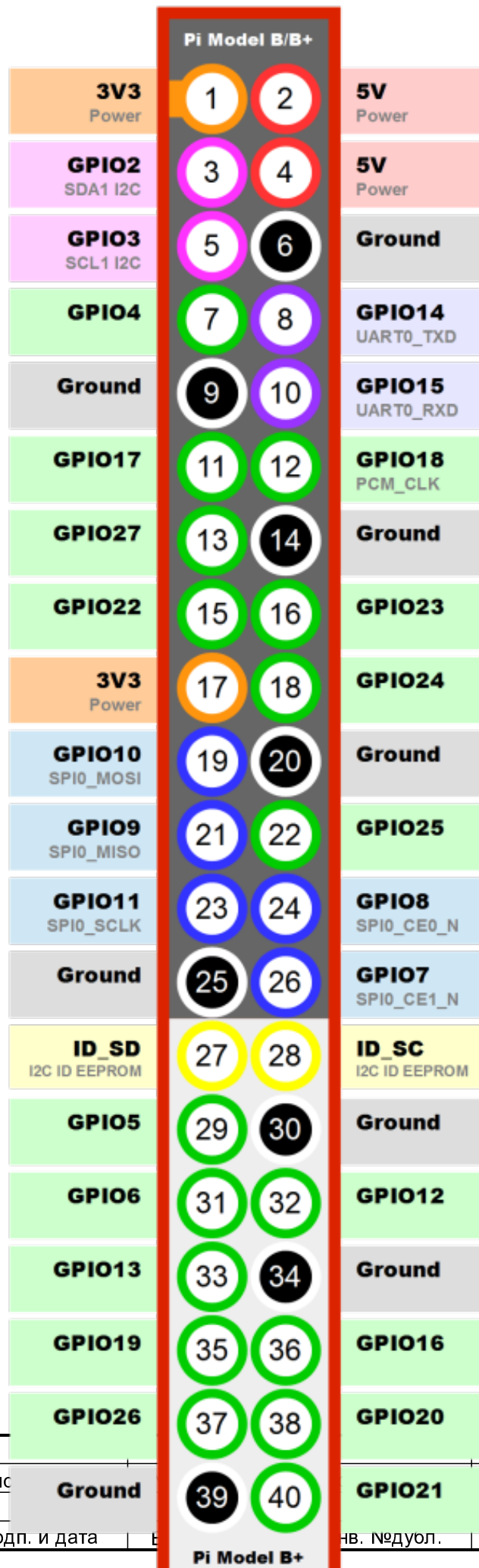
Implement this when class allows local matrix transforms. (**Entity** (стр. 25), Camera)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

**Лист регистрации изменений**

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводительного докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата



Изм.	Лис	Дата
RU.17701729.509000 81 01-1		
Инв. №подл.	Подп. и дата	Подп. и дата



PDIP, SOIC

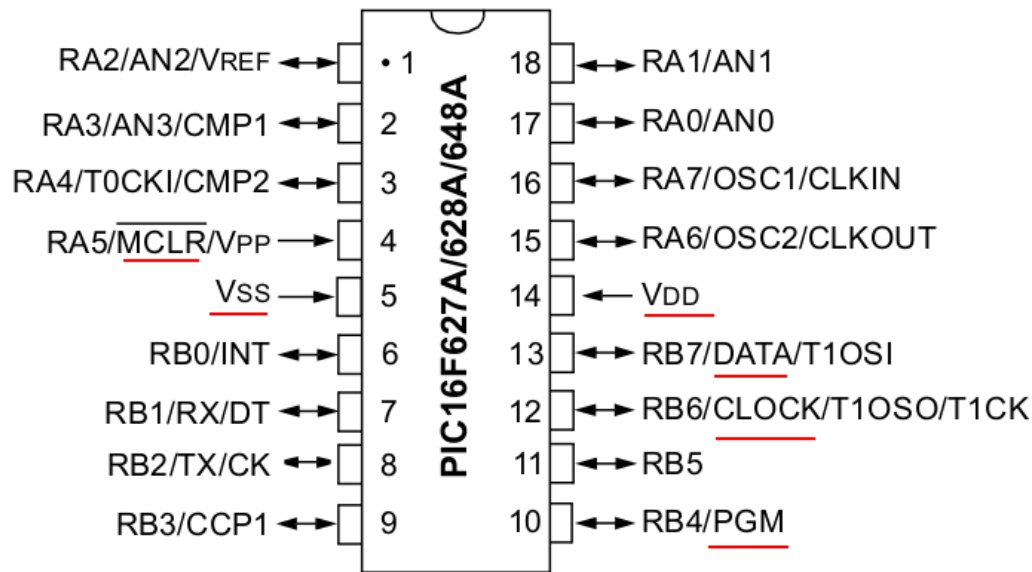


Рис. 9: Схемотехника для выходов GPIO Orange Pi Lite с процессором H3

PDIP, SOIC

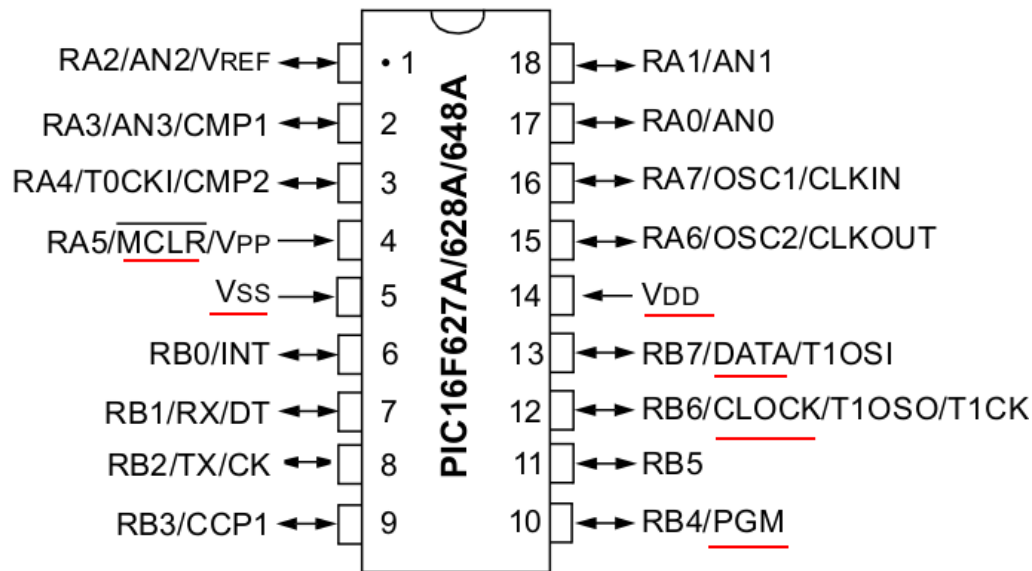


Рис. 10: Схемотехника для процессора H3 Orange Pi Lite

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

PDIP, SOIC

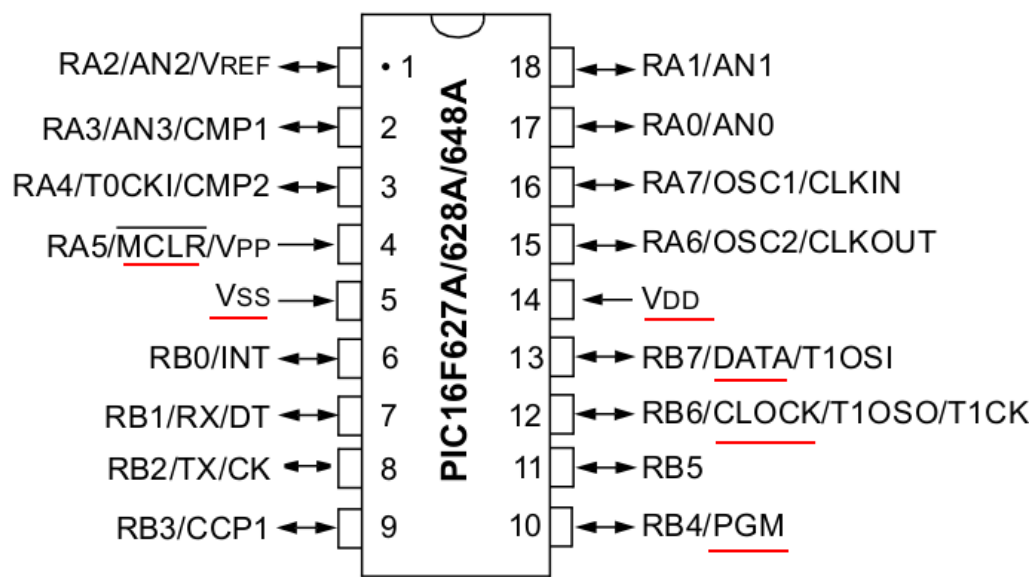


Рис. 11: Выдержка из руководства инженера к процессору НЗ с адресами ключевых регистров

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.509000 81 01-1				
Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата