Оглавление

Описание проекта 2
Название проекта 2
Правила разрабатываемой игры 2
Цикл игрового процесса 3
Применяемое аппаратное обеспечение, программные и инструментальные средства 3
Архитектура 4
Организация взаимодействия пользователя с консолью 4
Аппаратная часть 5
Программная часть 5
Программное обеспечение 6
Основная программа 6
Модуль игрового экрана 8
Модуль драйвера ЖКИ 10
Модуль драйвера клавиатуры 11
Модуль драйвера расширителя портов ввода-вывода 12
Модуль драйвера внешней E2PROM 13
Модуль интерфейса I2C 13

Описание проекта

Название проекта

Разработка аркадной игры для портативной игровой консоли, выполненной на базе стенда SDK-1.1.

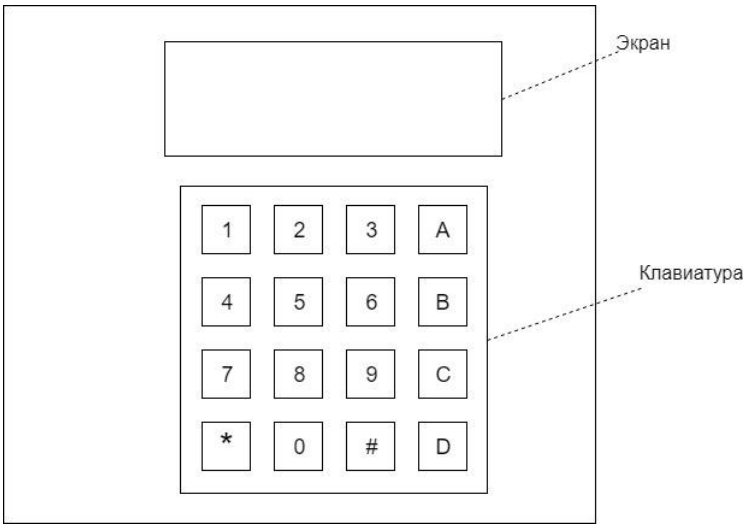


Рисунок 1. Общий вид консоли

Правила разрабатываемой игры

Игрок контролирует небольшую платформу, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под кубик, предотвращая его падение вниз. Платформа перемещается в нижней части игрового поля. В случайных частях игрового поля сверху вниз падают поочередно 10 кубиков.

Главная цель игры – поймать как можно больше кубиков.

К завершению игры приводят 2 случая:

1) Все 10 кубиков упали вниз. Количество кубиков, пойманных игроком, является результатом его игры.
2) Игрок нажал на кнопку выхода из игры. Результатом его игры будет считаться 0 пойманных кубиков.

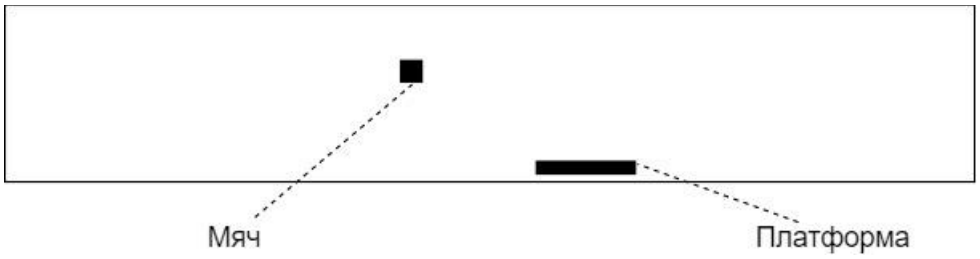
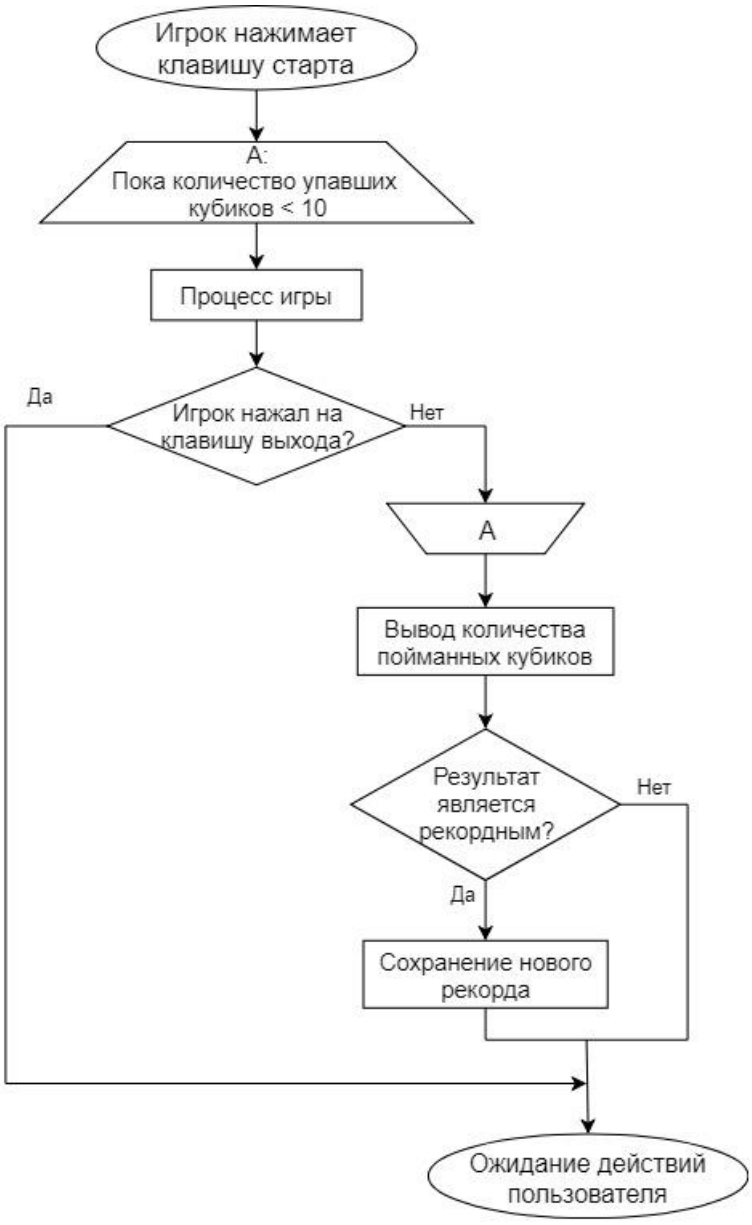


Рисунок 2. Общий вид игрового поля

Цикл игрового процесса



Применяемое аппаратное обеспечение, программные и инструментальные средства

- Аппаратная часть портативной игровой консоли – стенд SDK-1.1.
- Язык программирования – C-51.
- Компилятор – SDCC.

Архитектура

Организация взаимодействия пользователя с консолью

Взаимодействие пользователя с разрабатываемой консолью осуществляется посредством клавиатуры и экрана.

Для подачи консоли команд пользователю понадобится 5 клавиш: запуск игры, движение платформы вправо и влево, остановка игры, сброс рекорда. Расположение указанных клавиш изображено на рисунке 3.

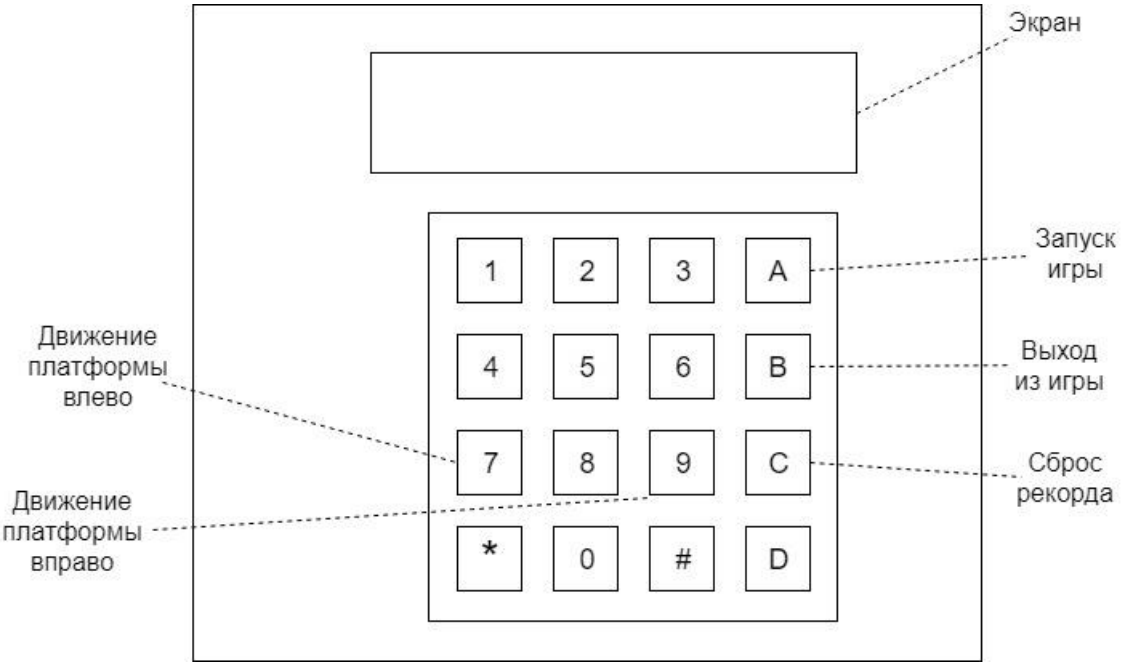


Рисунок 3. Общий вид консоли с указанием задействованных клавиш

Включенная консоль может находиться в двух состояниях:

1) *Ожидание.* Игровой процесс не начат. На экране отображается надпись «Hello! Press A to start». Доступны клавиши запуска игры и сброса рекорда.
    - В случае нажатия клавиши запуска игры отобразиться игровое поле, в центре которого расположена платформа. Далее начнётся игровой процесс.
    - В случае нажатия клавиши сброса рекорда на экран будет выведена либо надпись «Record reset», что означает успешный сброс рекорда, либо надпись «Reset record failed», что означает сбой во время операции сброса рекорда. Через 3 секунды на экране снова будет отображаться приглашение к началу игры.
2) *Игра.* На экране отображается игровое поле и текущие положения платформы и кубика. Доступны клавиши передвижения платформы и выхода из игры.
    - В случае нажатия клавиш движения вправо/влево платформа передвигается на расстояние, равное её длине.
    - В случае нажатия клавиши выхода из игры процесс игры останавливается, результат игрока обнуляется, и консоль переходит в состояние ожидания.

Если игрок не нажимал клавишу выхода, то после того, как все 10 кубиков упадут, на экране отобразиться результат в виде «x / 10», где x – количество пойманных игроком кубиков. Если это количество является рекордом, то на следующей строке экрана отобразиться надпись «New record!», и результат игрока будет сохранён.

Аппаратная часть

В разрабатываемом проекте используются следующие компоненты стенда SDK-1.1:

- Жидкокристаллический индикатор (ЖКИ) – вывод изображений игрового процесса и сообщений. Во время игры периодический вывод падающих кубиков осуществляется с помощью 16-ти битного таймера/счётчика T1.
- Клавиатура – ввод пользовательских команд для консоли. Периодичный опрос используемых клавиш осуществляется с помощью 16-ти битного таймера/счётчика T0.
- Внешняя E^2PROM – сохранение рекордов игры. Микросхема E^2PROM взаимодействует с процессором посредством интерфейса I^2C.

Программная часть

Разрабатываемый проект состоит из набора программных модулей, содержащих функции по работе с одним физическим устройством или программным объектом. Схема взаимодействия модулей показана на рисунке 4.

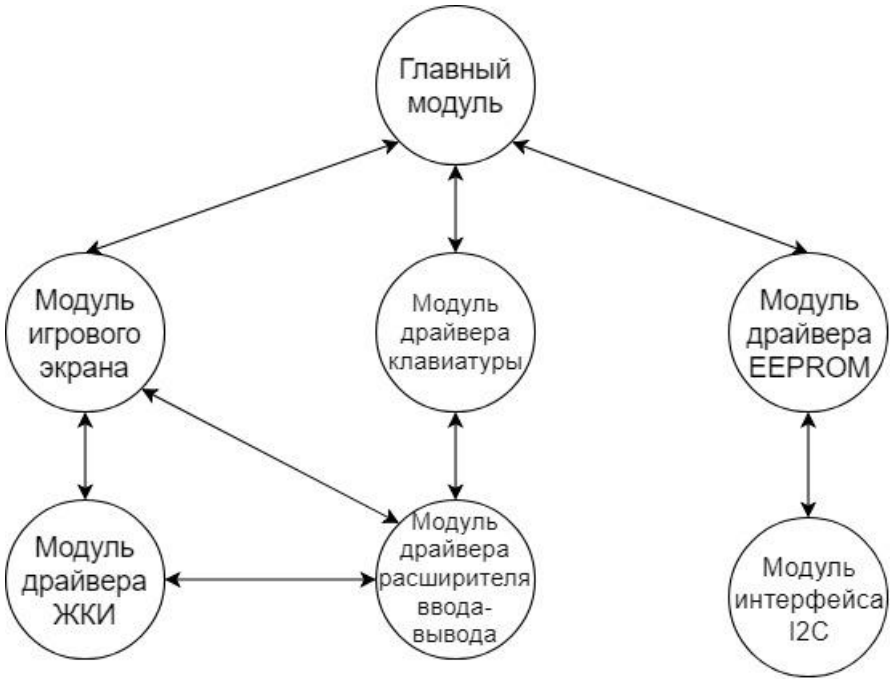


Рисунок 4. Схема взаимодействия программных модулей

В состав проекта входят следующие модули:

- Главный модуль – основная программа, управляющая остальными модулями.
- Модуль игрового экрана – набор функций отображения на ЖКИ различных сообщений и сцен игры.
- Модуль драйвера ЖКИ – набор функций для взаимодействия с ЖКИ.

- Модуль драйвера клавиатуры – набор функций для считывания состояния клавиш, задействованных в проекте.
- Модуль драйвера расширителя портов ввода-вывода – функции считывания и записи в регистр ПЛИС.
- Модуль драйвера внешней E^2PROM – функции чтения и записи байта данных в E^2PROM.
- Модуль интерфейса I^2C – функции передачи и приёма байт данных по протоколу I^2C.

Программное обеспечение

Ниже представлены описания логики работы каждого модуля разрабатываемого приложения.

Основная программа состоит из двух частей: инициализации, включающей ряд подготовительных действий перед началом приёма пользовательских команд, и обработка действий пользователя. Структурная схема основной программы приведена на рисунках 5 и 6.

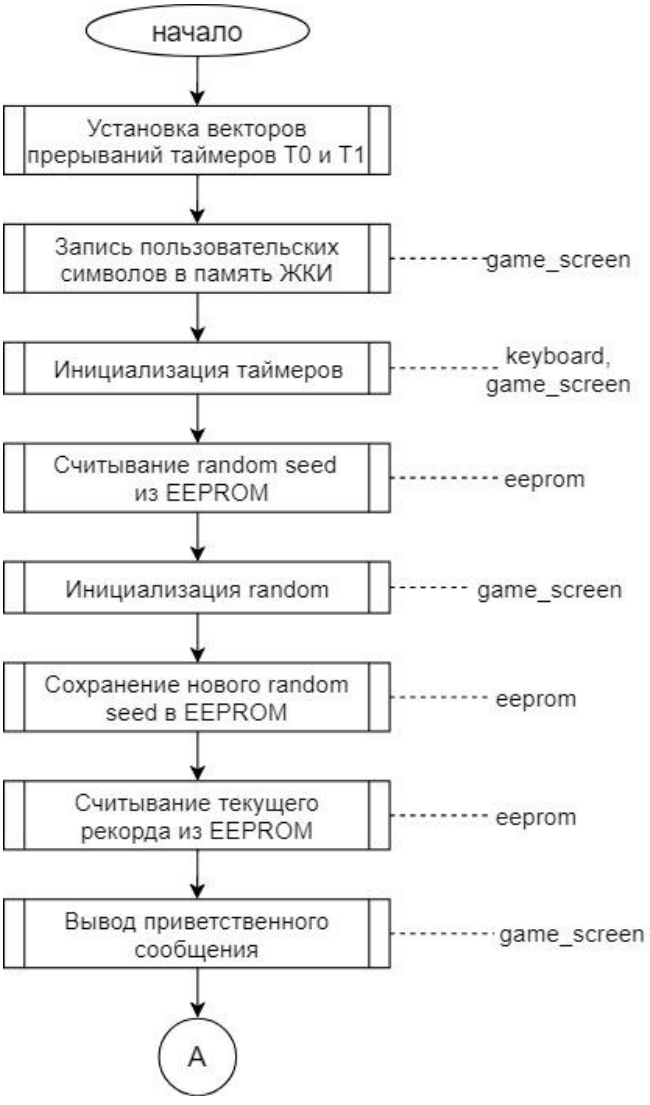


Рисунок 5. Структурная схема инициализационной части основной программы

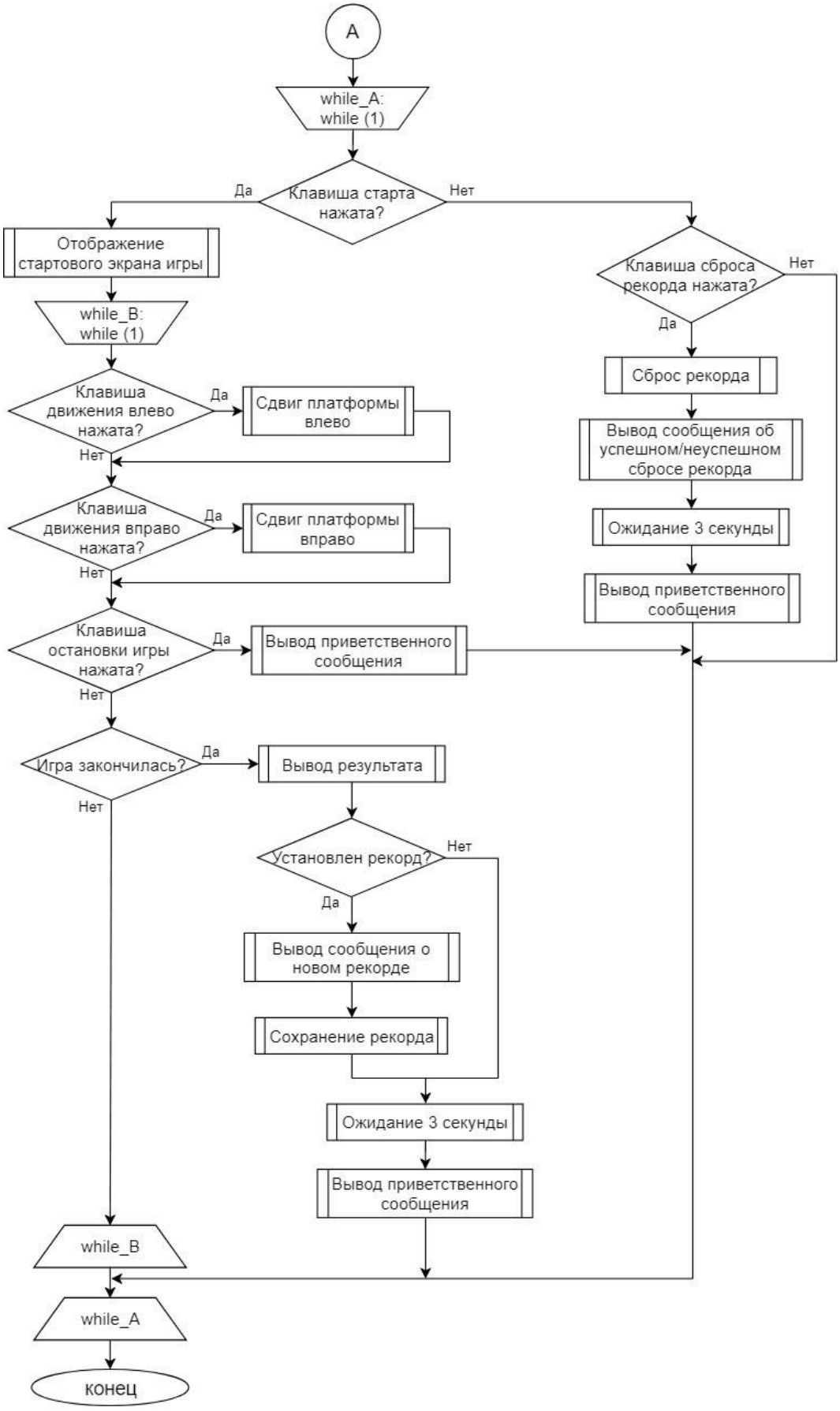


Рисунок 6. Структурная схема части основной программы по обработке команд пользователя

На схеме используются следующие условные обозначения:

- game_screen – модуль игрового экрана;
- keyboard – модуль драйвера клавиатуры;
- eeprom – модуль драйвера внешней E^2PROM;
- lcd – модуль драйвера ЖКИ;
- max – модуль драйвера расширителя портов ввода-вывода;
- i2c – модуль интерфейса I^2C.

Модуль игрового экрана содержит следующие функции:

- `unsigned char initRand(unsigned char randSeed)` – функция для инициализации генератора случайных чисел.

Вход: `unsigned char randSeed` – зерно, на основе которого в дальнейшем будет происходить генерация.

Результат: новое значение зерна, которое будет использоваться в следующей инициализации генератора.

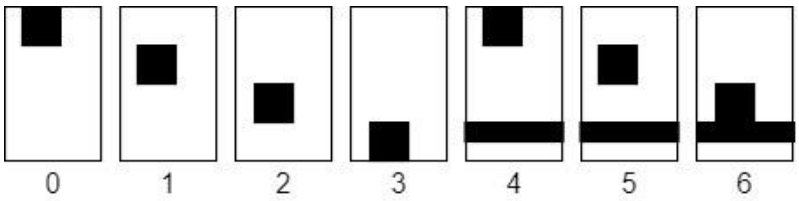
Описание: выполняет инициализацию генератора случайных чисел с использованием аргумента функции и возвращает число, которое будет использоваться в качестве зерна при следующем запуске консоли.

- `void defineUserChars(void)` – функция программной генерации символов, которых нет в ПЗУ, для отображения на ЖКИ падающего кубика.

Вход: нет.

Результат: нет.

Описание: Функция заносит в CGRAM следующие семь символов, которые изображают стадии падения кубика:



- `void initGameTimer(void)` – функция инициализации таймера, задействованного для отображения игрового процесса на ЖКИ.

Вход: нет.

Результат: нет.

Описание: выполняет инициализацию регистров таймера T1, а также задаёт режим работы таймера.

- `void T1Handling(void) __interrupt (3)` – обработчик прерывания таймера T1.

Описание: падение кубиков организуется с помощью прерываний от таймера T1, работающего в 16-ти битном режиме. На каждом срабатывании либо выполняется перерисовка сцены падения кубика, либо ожидания следующего срабатывания прерывания для создания временного интервала между падениями кубиков.

- `void drawMainScreen(void)` – функция вывода приветственного сообщения на экран.

  Вход: нет.

  Результат: нет.

  Описание: функция выводит на ЖКИ надпись «Hello! Press A to start».

- `void drawStartScreen(void)` – функция вывода на экран начальной сцены игры.

  Вход: нет.

  Результат: нет.

Описание: на ЖКИ отображается пустое игровое поле, в центре которого находится платформа.

- `void drawFinishScreen(unsigned char isRecord)` – функция вывода на экран результата игры пользователя.

Вход: `unsigned char isRecord` – определяет, нужно ли выводить на экран надпись о новом рекорде. 1 – результат пользователя является рекордным, 0 – нет.

Результат: нет.

Описание: отображает результат игры в виде «x / 10», где x – количество пойманных игроком кубиков. Если это количество является рекордом, то на следующей строке экрана отобразиться надпись «New record!»

- `void drawRecResetScreen(unsigned char isReset)` – функция вывода сообщения об успешном/неуспешном сбросе рекорда.

Вход: `unsigned char isReset` – определяет, какую надпись нужно вывести на экран. 1 – «Record reset», 0 – «Reset record failed».

Результат: нет.

Описание: в зависимости от аргумента выводит либо надпись об успешном сбросе рекорда, либо о том, что сброс не удалось выполнить.

- `void movePlatformToLeft(void)` – функция передвижения платформы влево.

  Вход: нет.

  Результат: нет.

  Описание: функция перемещает платформу на одну символьную клетку влево. Если достигнут край экрана, то платформа остается на месте.

- `void movePlatformToRight(void)` – функция передвижения платформы вправо.

  Вход: нет.

  Результат: нет.

  Описание: функция перемещает платформу на одну символьную клетку вправо. Если достигнут край экрана, то платформа остается на месте.

- `unsigned char isGameFinished(void)` – функция определения окончания игры.

  Вход: нет.

  Результат: 1 – если все 10 кубиков упали, 0 – в противном случае.

  Описание: возвращает флаг окончания игры.

- `unsigned char getGameRes(void)` – функция получения результата игры пользователя.

  Вход: нет.

  Результат: количество пойманных игроком кубиков.

  Описание: возвращает результат игры.

Модуль драйвера ЖКИ содержит следующие функции:

- `unsigned char getBF(void)` – функция опроса флага занятости BF.

  Вход: нет.

  Результат: 1 – если БИС занята выполнением внутренних операций, 0 – если может быть принята следующая команда.

  Описание: возвращает состояние флага занятости BF.

- `void clear(void)` – функция очистки экрана.

  Вход: нет.

  Результат: нет.

Описание: очистка всей DDRAM и возврат курсора в начальную позицию.

- `void setDDRAMaddr(unsigned char addr)` – функция установки адреса DDRAM.

Вход: `unsigned char addr` – адрес в памяти DDRAM.

Результат: нет.

Описание: устанавливает курсор в DDRAM по адресу, указанному в аргументе.

- `void printChar(unsigned char c)` – функция вывода символа на дисплей.

Вход: `unsigned char c` – код выводимого на экран символа.

Результат: нет.

Описание: выводит на экран символ, код которого передаётся в качестве аргумента.

- `void printString(unsigned char *str)` – функция вывода строки символов на дисплей.

Вход: `unsigned char *str` – строка символов, которые необходимо вывести на экран.

Результат: нет.

Описание: выводит на экран строку символов, коды которых передаются в качестве аргумента.

Модуль драйвера клавиатуры содержит следующие функции:

- `void initKeyboardTimer(void)` – функция инициализации таймера, задействованного для периодического опроса клавиш.

Вход: нет.

Результат: нет.

Описание: выполняет инициализацию регистров таймера T0, а также задаёт режим работы таймера.

- `void T0Handling(void) __interrupt (1)` – обработчик прерывания таймера T0.

Описание: опрос клавиш клавиатуры происходит по прерыванию от таймера T0, который работает в 16-ти битном режиме. На каждом срабатывании обновляются переменные, сохраняющее состояние всех задействованных в работе консоли клавиш.

- `unsigned char isStartPressed(void)` – функция получения состояния клавиши старта игры.

Вход: нет.

Результат: 1 – если клавиша нажата, 0 – если нет.

Описание: возвращает текущее состояние клавиши старта игры.

- `unsigned char isStopPressed(void)` – функция получения состояния клавиши остановки игры.

  Вход: нет.

  Результат: 1 – если клавиша нажата, 0 – если нет.

  Описание: возвращает текущее состояние клавиши остановки игры.

- `unsigned char isRecResetPressed(void)` – функция получения состояния клавиши сброса рекорда.

  Вход: нет.

  Результат: 1 – если клавиша нажата, 0 – если нет.

  Описание: возвращает текущее состояние клавиши сброса рекорда.

- `unsigned char isLeftPressed(void)` – функция получения состояния клавиши движения влево.

  Вход: нет.

  Результат: 1 – если клавиша нажата, 0 – если нет.

  Описание: возвращает текущее состояние клавиши движения влево.

- `unsigned char isRightPressed(void)` – функция получения состояния клавиши движения вправо.

  Вход: нет.

  Результат: 1 – если клавиша нажата, 0 – если нет.

  Описание: возвращает текущее состояние клавиши движения вправо.

Модуль драйвера расширителя портов ввода-вывода содержит следующие функции:

- `void writeMax(unsigned char xdata *regnum, unsigned char val)` – функция записи в нужный регистр ПЛИС.

Вход: `unsigned char xdata *regnum` – адрес (номер) регистра, `unsigned char val` – записываемое значение.

Результат: нет.

Описание: производится запись в регистр (порт) ПЛИС путем переключения адресуемой страницы памяти на страницу, где расположены (куда отображаются) порты ввода-вывода ПЛИС.

- `unsigned char readMax(unsigned char xdata *regnum)` – функция чтения из нужного регистра ПЛИС.

Вход: `unsigned char xdata *regnum` – адрес (номер) регистра.

Результат: прочитанное из регистра значение.

Описание: чтение из порта ПЛИС путем переключения адресуемой страницы памяти на страницу, где расположены (куда отображаются) порты ввода-вывода ПЛИС.

Модуль драйвера внешней E2PROM содержит следующие функции:

- `unsigned char writeByteEeprom(unsigned short address, unsigned char dataByte)` – функция записи байта данных в E^2PROM.

Вход: `unsigned short address` – адрес (номер ячейки) E^2PROM, куда нужно записать байт, `unsigned char dataByte` – данные, которые нужно записать.

Результат: 0 – операция выполнена успешно, 1 – E^2PROM не отвечает.

Описание: выполняется запись байта данных в ячейку E^2PROM.

- `unsigned char readByteEeprom(unsigned short address, unsigned char *dataByte)` – функция чтения байта данных из E^2PROM.

Вход: `unsigned short address` – адрес (номер ячейки) E^2PROM, откуда нужно считать байт, `unsigned char *dataByte` – адрес переменной, куда нужно записать считанное значение.

Результат: 0 – операция выполнена успешно, 1 – E^2PROM не отвечает.

Описание: выполняется чтение байта данных из ячейку E^2PROM.

Модуль интерфейса I2C содержит следующие функции:

- `unsigned char getAck(unsigned char periphAddr)` – функция определения готовности slave-устройства.

Вход: нет.

Результат: 0 – устройство готово к обмену, 1 – в противном случае.

Описание: проверка на готовность slave-устройства к обмену (начало получение отклика завершение сессии).

- `unsigned char receiveByte(unsigned char periphAddr, unsigned char addr, unsigned char *dataByte)` – функция приёма байта от slave-устройства.

Вход: `unsigned char periphAddr` – адрес ведомого устройства, `unsigned char addr` – адрес во внутреннем адресном пространстве устройства, `unsigned char *dataByte` – адрес переменной, куда нужно записать принятый байт.

Результат: 0 – операция выполнена успешно, 1 – устройство не откликается.

Описание: получение 8 бит с шины данных I^2C.

- `unsigned char transmitByte(unsigned char periphAddr, unsigned char addr, unsigned char dataByte)` – функция передачи байта slave-устройству.

Вход: `unsigned char periphAddr` – адрес ведомого устройства, `unsigned char addr` – адрес во внутреннем адресном пространстве устройства, `unsigned char dataByte` – передаваемый байт.

Результат: 0 – операция выполнена успешно, 1 – устройство не откликается.

Описание: передача 8 бит по шины данных I^2C.