**Описание функций и алгоритмов работы ПО платы СЦ4**

**1. Функции ПО платы СЦ4**

ПО платы СЦ4 для обеспечения работы пультов ПДО выполняет следующие функции:

- начальное тестирование устройств платы с индикацией состояния светодиодами и выдачей сообщений по интерфейсу RS232;

- опрос кнопок платы СЦ4 (регулировка громкости выводимого звука, чувствительности микрофона, кнопка ТЕСТ) и индикацию состояния (светодиоды МКВКЛ и НОРМА);

- обмен с платой УИ по I2C-шине для опроса кнопок и индикации состояния;

- опрос контактов внешних подключений (плата УИ, аудиоустройства) и выдачу сигналов управления устройствами, подключенными к плате (усилитель, реле).

- обмен данными с микросхемой EEPROM по I2C-шине для обеспечения хранения конфигурационных данных;

- обмен данными с внешней микросхемой ОЗУ для временного хранения массивов данных (конфигурация, прошивка, аудио)

- обмен по интерфейсу RS232 и обработку поступающих команд в режиме командной строки;

- реализацию режима TEST с управлением выполнением тестов и выдачей результатов через COM-терминал в режиме командной строки;

- реализацию режима DEBUG с выдачей отладочных сообщений по интерфейсу RS232;

- реализацию режима SERVICE с выдачей служебных сообщений по интерфейсу RS232;

- реализацию режима конфигурирования через интерфейс RS232 для чтения/записи серийного номера, ключа шифрования по сети, конфигурационных данных при помощи ПО «Конфигуратор»

- настройку микросхемы аудиокодека, ввод и вывод звука через аудиокодек по шине SAI;

- регулировку при помощи аудиокодека громкости выводимого звука и чувствительности микрофона;

- генерацию тональных информационных аудиосигналов;

- обработку аудиосигналов - буферизация, микширование, кодек G711;

- настройку микросхемы PHY-контроллера Ethernet и обмен данными по сети Ethernet;

- обмен данными по сетевым протоколам стека TCP/IP при помощи lwIP;

- реализацию сетевого протокола межприборного обмена для обработки режимов связи и обмена служебной информацией с другими приборами GIT-Comm IPS;

- реализацию сетевого протокола обмена и обмена по интерфейсу RS232 с ПО «Конфигуратор» для защищенного реконфигурирования и перепрограммирования платы;

- реализацию сетевого протокола документирования работы для отправки состояния и аудиопотока по сети на сервер документирования.

**2. Алгоритм работы ПО платы СЦ4**

Программа после запуска выполняет последовательность инициализации аппаратных и программных модулей, алгоритм которой приведен на рисунке 2.1.

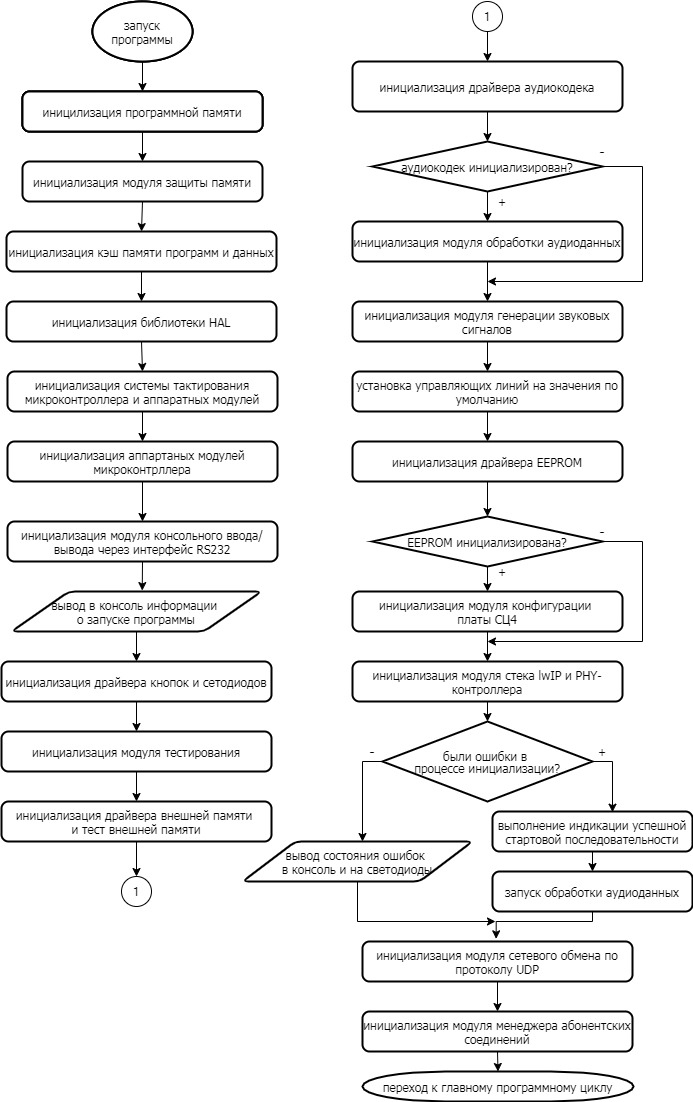


Рисунок 2.1 Блок-схема алгоритма последовательности инициализации аппаратных и программных модулей.

В процессе инициализации аппаратных модулей драйверов внешних устройств производится тестирование и собирается информация об отказах во внешних устройствах. Начальное тестирование проводится при инициализации следующих устройств:

- микросхема аудиокодека;

- микросхема памяти EEPROM;

- микросхемы внешней памяти SRAM;

- микросхема PHY-контроллера Ethernet.

В случае если зарегистрирован отказ, выдается сообщение об ошибке через интерфейс RS232, устанавливаются соответствующие состояния светодиодов и программа переводится в режим работы HWFAIL обеспечивающим управление программой через интерфейс RS232 с COM-терминала или ПО «Конфигуратор».

В случае, если не было отказа микросхемы памяти EEPROM, выполняется попытка загрузки конфигурационных данных.

При неуспешной загрузке данных, если программа уже не находится в режиме HWFAIL, выдается сообщение об ошибке через интерфейс RS232, устанавливаются соответствующие состояния светодиодов и программа переводится в режим работы NOCFG с конфигурационными данными по умолчанию. обеспечивающим управление программой через интерфейс RS232 с COM-терминала или ПО «Конфигуратор», а также при отсутствии отказа PHY-контроллера – по сети с ПО «Конфигуратор».

В случае, если конфигурационные данные успешно считаны и не было отказов, то выполняется процедура индикации успешной инициализация (мигание всеми светодиодами), запуск процесса обработки аудиоданных и программа переводится в режим работы WORK.

После выполнения инициализации запускается главный цикл работы программы, алгоритм которого приведен на рисунке 2.2.

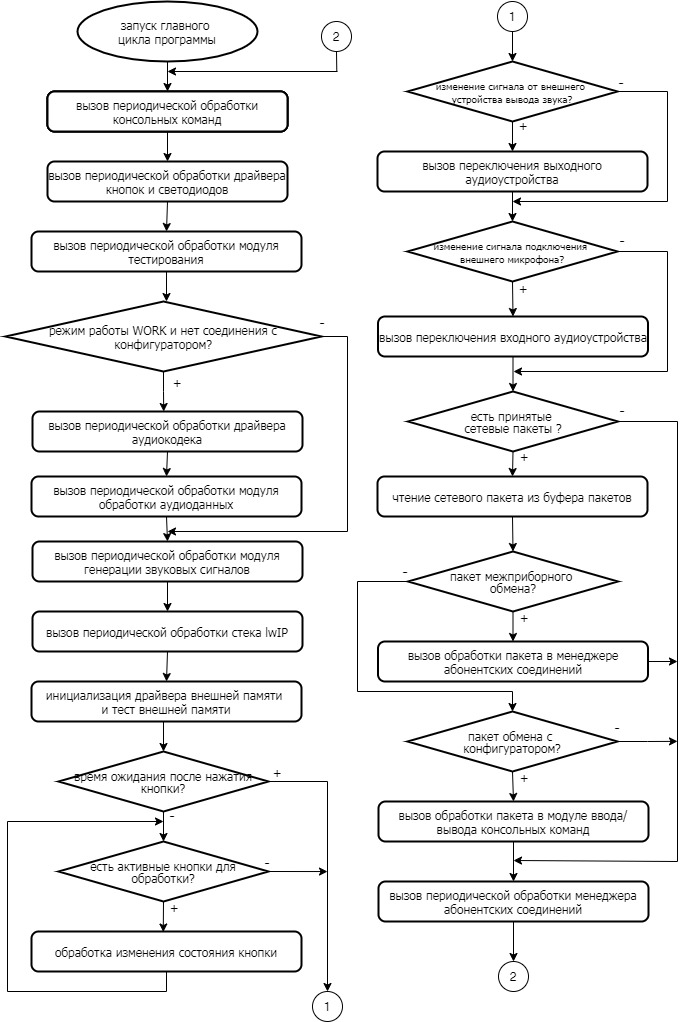


Рисунок 2.2 Блок-схема алгоритма работы главного цикла программы.

В главном цикле выполняется ряд процедур:

1. Периодическая обработка драйверов устройств;

2. Периодическая обработка программных модулей (обработка команд по интерфейсу RS232, обновление состояния кнопок и светодиодов);

3. Периодическая обработка стека lwIP;

4. Периодическая обработка менеджера абонентских соединений;

5. Если программа находится в режиме WORK и не выполняются процедуры реконфигурирования или перепрограммирования устройства с ПО «Конфигуратор», выполняется процедура периодической обработки аудиоданных;

6. Отслеживание состояние кнопок и при их изменении выполняются соответствующие процедуры:

6.1 для кнопки «ТЕСТ» - выполняется включение/выключение режима тестирования в модуле тестового режима.

6.2- для кнопок регулировки громкости звука и чувствительности микрофона - выполняется изменение уровней усиления АЦП и ЦАП аудиокодека в модуле драйвера аудиокодека.

6.3 для кнопок платы УИ – вызывается менеджер абонентских соединений для выполнения необходимых действий для изменения состояний режимов связи.

7. Проверка контактов подключений внешних аудиоустройств (микрофон, телефоны) и выполнения процедур соответствующих переключений в модуле драйвера аудиокодека и модуле управления линиями ввода/вывода.

8. Проверка поступления пакетов из сети Ethernet и обработка пекетов в соответствии с их типом:

8.1 Если принят пакет межприборного обмена, вызов менеджера абонентских соединений для обработки пакета и выполнения необходимых действий для изменения состояний режимов связи.

8.2 Если принят пакет с командой от ПО «Конфигуратор», вызов процедуры обработки команды в модуле консольного ввода/вывода.

Взаимодействие программных функций относительно главного (основного) цикла программы представлено в отдельном документе «Функциональная схема ПО платы СЦ4» и на рисунке 2.3.

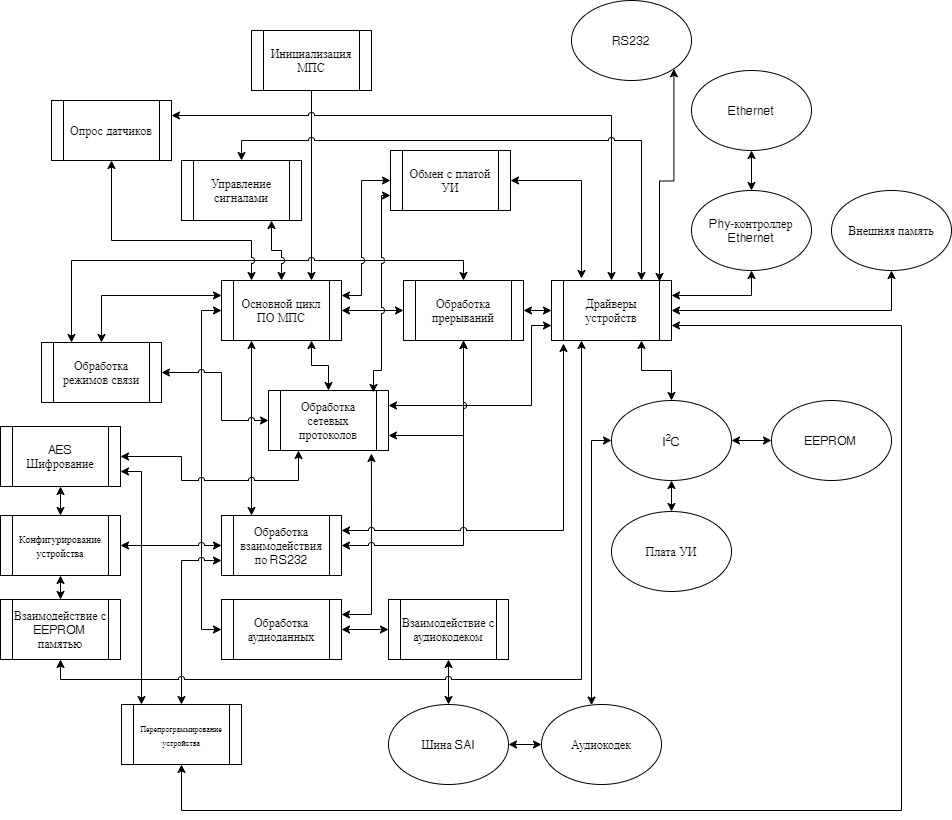


Рисунок 2.3 Функциональная схема ПО платы СЦ4.

Диаграмма основных состояний цикла программы приведена в отдельном документе «Диаграмма состояний ПО платы СЦ4» и на рисунке 2.4.

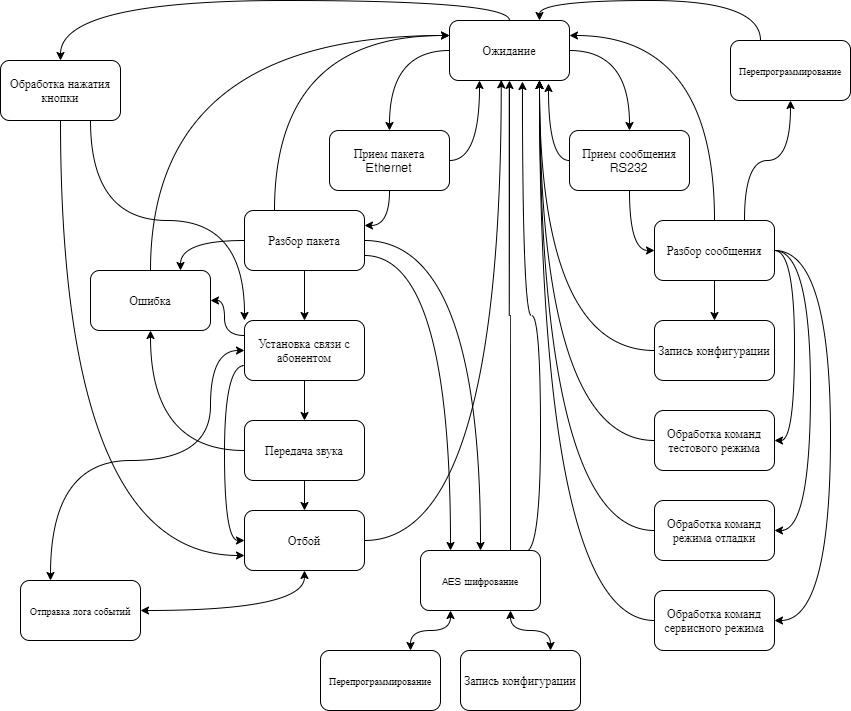


Рисунок 2.4 Диаграмма основных состояний главного цикла ПО платы СЦ4.

Обработка изменений состояний режима в менеджере абонентских соединений и выполнение соответствующих действий происходит согласно диаграмме состояний режимов связи, приведенной на рисунке .2.5

Для реализации всех необходимых режимов связи пульт может находиться в одном из девятнадцати состояний. Переход из одного состояния в другое происходит только по заданным событиям. Имеется три группы состояний для обработки режимов связи, группа состояний обработки сигнала «Отбой», а также группа состояний обработки пропущенных вызовов.

Базовым состоянием является ABONENT\_IDLE. В этом состоянии возможны осуществление вызовов абонентов и прием входящих вызовов со всех запрограммированных кнопок.

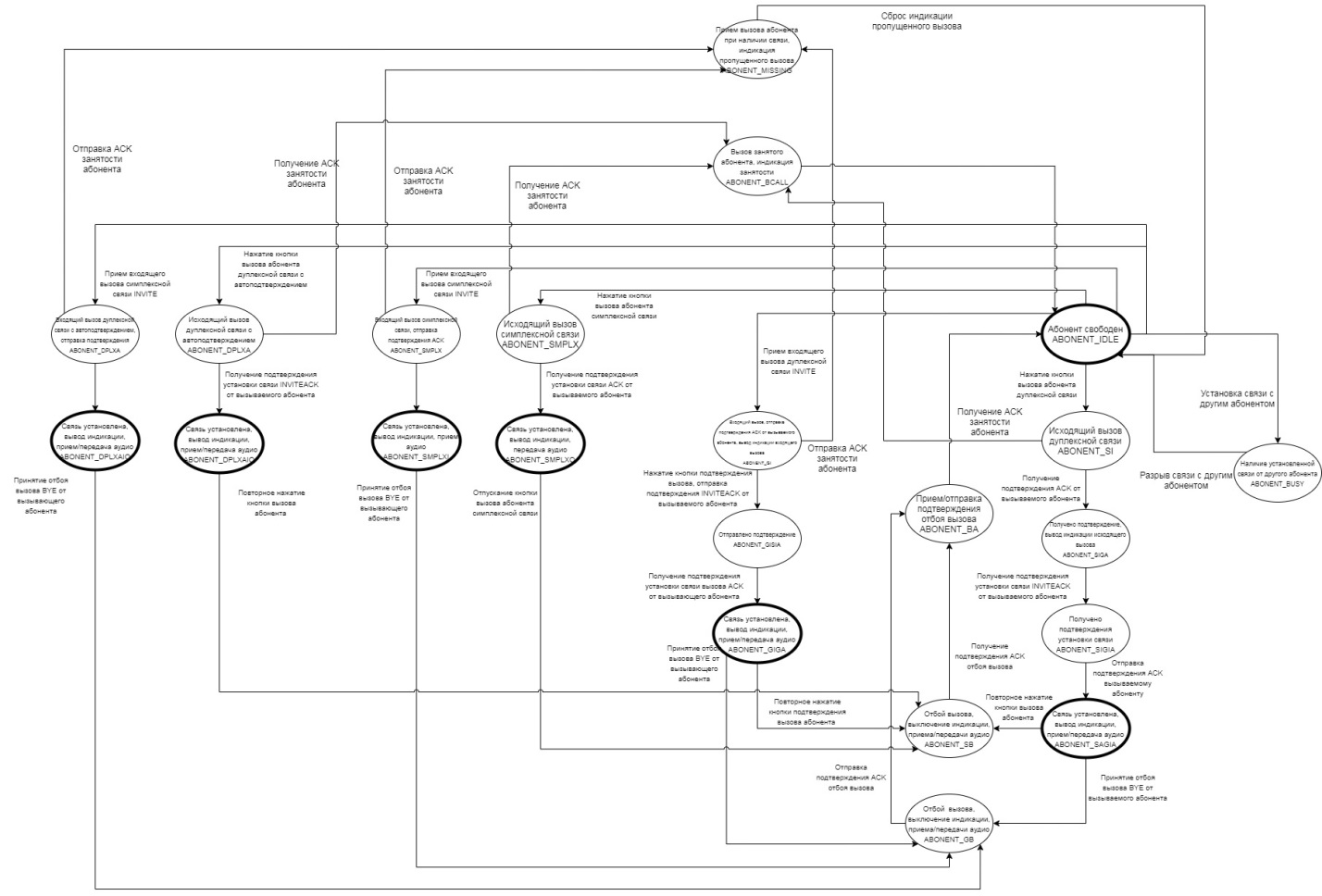


Рисунок 2.5 Диаграмма состояний режимов связи ПО платы СЦ4.

Переход в группу состояний обработки дуплексной связи происходит при нажатии кнопки вызова абонента в режиме дуплексной связи или при принятии запроса установки связи данного типа. Состояния реализации режима дуплексной связи:

1. ABONENT\_SI – нажата кнопка вызова абонента в режиме дуплексной связи.
2. ABONENT\_SIGA – от пульта вызванного абонента получено подтверждение о принятии входящего вызова
3. ABONENT\_SIGIA – вызываемый пользователь подтвердил установку связи нажатием кнопки ответа.
4. ABONENT\_SAGIA – связь установлена, идет передача аудиопотока в обе стороны.
5. ABONENT\_GI – получен запрос на установку соединения в режиме дуплексной связи, отправлено подтверждение.
6. ABONENT\_GISIA – пользователь подтвердил установку связи, отправлено сообщение с подтверждением.
7. ABONENT\_GIGA – связь установлена, идет прием/передача аудиопотока в обе стороны.

Переход в группу состояний обработки симплексной связи происходит при удержании кнопки вызова абонента в режиме симплексной связи или при принятии запроса установки связи данного типа. Состояния реализации режима симлексной связи:

1. ABONENT\_SMPLX – удерживается кнопка вызова абонента в режиме симплексной связи или принят запрос на установку соединения в данном режиме.
2. ABONENT\_SMPLXI – установлена связь в режиме симплексной связи в качестве принимающей стороны, идет прием аудиопотока.
3. ABONENT\_SMPLXO – установлена связь в режиме симплексной связи в качестве отправляющей стороны, идет передача аудиопотока.

Переход в группу состояний обработки дуплексной связи с автоподтверждением происходит при нажатии кнопки вызова абонента в режиме дуплексной связи с автоподтверждением или при принятии запроса установки связи данного типа. Связь устанавливается автоматически и не требует подтверждения на вызываемой стороне. Состояния реализации режима дуплексной связи:

1. ABONENT\_DPLXA – нажата кнопка вызова абонента в режиме дуплексной связи с автоподтверждением или принят запрос на установку соединения в данном режиме.
2. ABONENT\_DPLXAIO – установлена связь в режиме дуплексной связи с автоподтверждением, идет прием/передача аудиопотока в обе стороны.

Переход в группу состояний обработки сигнала «Отбой» происходит, если вызываемый/вызывающий абонент штатно разрывает сеанс связи. Состояния обработки сигнала «Отбой»:

1. ABONENT\_SB – была нажата кнопка разрыва сеанса связи, отправляется команда «Отбой», идет разрыв соединения.
2. ABONENT\_GB – была принята команда «Отбой», идет разрыв соединения.
3. ABONENT\_BA – отправлено подтверждение приема команды «Отбой».

Переход в группу состояний обработки пропущенных вызовов происходит, если вызываемый абонент уже находится в каком-либо режиме связи либо отсутствует связь с этим абонентом. Состояния реализации обработки пропущенных вызовов:

1. ABONENT\_BUSY – абонент занят. При переходе вызываемого/вызывающего абонента в любой режим связи все остальные абоненты переходят в текущее состояние.
2. ABONENT\_BCALL – связь с абонентом не установлена по причине занятости абонента. Производится световая и звуковая индикация согласно ТЗ.
3. ABONENT\_MISSING – пропущенный вызов. Переход осуществляется при принятии запроса на установку связи с абонентом при наличии любого установленного режима связи с другим абонентом.

**3. Описание модулей ПО платы СЦ4**

ПО платы СЦ4 ips\_sc4\_pdo разработано на языке С++ и имеет модульную структуру. Взаимосвязи между модулями приведены в отдельном документе «Структурная схема ПО платы СЦ4» и на рисунке 3.1.

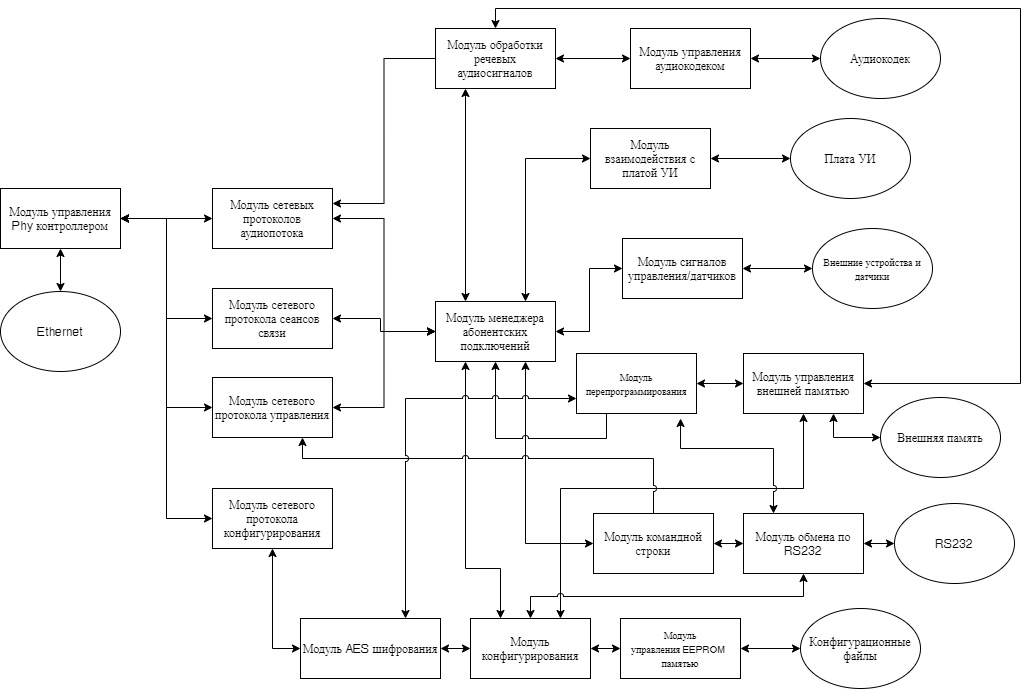


Рисунок 3.1 . Структурная схема ПО платы СЦ4.

В ПО платы СЦ4 используются следующие стандартные библиотеки из состава пакета программного обеспечения поддержки разработки ПО для микроконтроллеров STM32 ARM Сortex-M7:

1. CMSIS – библиотека, реализующая функции управления ядром микроконтроллеров STM32 ARM Сortex-M7.

2. STM32H7xx\_HAL\_Driver – библиотека, реализующая функции управления внутренними аппаратными модулями микроконтроллеров STM32 ARM Сortex-M7.

3. LwIP – библиотека, реализующая сетевой стек протоколов TCP/IP, адаптированная для микроконтроллеров STM32.

Ниже приводится описание модулей ПО платы СЦ4 и содержащихся в них основных программных элементов.

3.1 **startup\_stm32h753iitx.s** – стандартный ассемблерный модуль начальной инициализации программы для микроконтроллеров STM Cortex-M7.

Содержит следующие основные программные элементы:

g\_pfnVectors – таблица векторов прерываний;

Reset\_Handler – обработчик прерывания сброса/включения микроконтроллера, выполняет инициализацию тактирования ядра, базовых регистров, внутренней памяти микроконтроллера и переход на основную подпрограмму main.

3.2 **system\_stm32h7xx.c** – стандартный модуль начальной аппаратной инициализации микроконтроллера STM Cortex-M7.

Содержит следующие основные программные элементы:

SystemInit – подпрограмма инициализации базовых аппаратных модулей ядра микроконтроллера.

SystemCoreClockUpdate – подпрограмма инициализации тактирования ядра микроконтроллера.

3.3 **stm32h7xx\_hal\_msp.c** – модуль инициализации внутренних аппаратных модулей микроконтроллера STM Cortex-M7. Генерируется автоматически в среде STM32CubeIDE согласно описания используемых модулей микроконтроллера.

Содержит следующие используемые программные элементы:

HAL\_MspInit – подпрограмма базовой инициализации внутренних аппаратных модулей (тактирование).

HAL\_I2C\_MspInit – подпрограмма инициализации внутренних аппаратных модулей шины I2C.

HAL\_TIM\_Base\_MspInit – подпрограмма инициализации внутренних аппаратных модулей таймеров.

HAL\_UART\_MspInit – подпрограмма инициализации внутренних аппаратных модулей приемопередатчиков UART.

HAL\_FMC\_MspInit – инициализация внутреннего аппаратного модуля шины памяти FMC.

HAL\_SRAM\_MspInit – инициализация внутреннего аппаратного модуля контроллера внешней памяти.

HAL\_SAI\_MspInit – инициализация внутренних аппаратных модулей шины SAI.

3.4 **stm32h7xx\_it.c** – модуль обработчиков прерываний микроконтроллера STM Cortex-M7. Генерируется автоматически в среде STM32CubeIDE согласно описания используемых модулей микроконтроллера.

Содержит следующие используемые программные элементы:

HardFault\_Handler – обработчик прерывания аппаратного сбоя ядра микроконтроллера. При сбое выводит сообщение с содержимым регистров в момент сбоя через UART (RS232) и останавливает выполнение программы.

fault\_GetRegistersFromStack – выполняет извлечение значений регистров микроконтроллера из аппаратного стека в момент сбоя и их печать в UART.

SysTick\_Handler – обработчик прерывания аппаратного модуля таймера ядра микроконтроллера. Обеспечивает счет системного времени.

DMA1\_Stream0\_IRQHandler – обработчик прерывания аппаратного модуля DMA1\_Stream0. Обеспечивает передачу аудиоданных микросхеме аудиокодека по шине SAI.

DMA1\_Stream1\_IRQHandler – обработчик прерывания аппаратного модуля DMA1\_Stream1. Обеспечивает прием аудиоданных с микросхемы аудиокодека по шине SAI.

TIM3\_IRQHandler – обработчик прерывания аппаратного модуля таймера Timer3. Обеспечивает работу аудиомикшера.

ETH\_IRQHandler – обработчик прерывания аппаратного модуля MAC-контроллера Ethernet. Обеспечивает работу интерфейса Ethernet.

UART7\_IRQHandler – обработчик прерывания аппаратного модуля приемопередатчика UART7. Обеспечивает работу интерфейса RS232.

SAI1\_IRQHandler – обработчик прерывания аппаратного модуля шины SAI1. Обеспечивает работу с микросхемой аудиокодека по шине SAI.

3.5 **syscalls.c** – стандартный модуль адаптации базовых системных функций С++ библиотеки stdlib для микроконтроллера CortexM.

3.6 **sysmem.c** – стандартный модуль адаптации функции работы с памятью С++ библиотеки stdlib для микроконтроллера CortexM. Содержит функцию \_sbrk – обеспечивающую динамическое выделение памяти в области heap с учетом распределения внутренней памяти микроконтроллера.

3.7 **main.c** – главный программный модуль.

Содержит следующие основные программные элементы:

main – главная подпрограмма. Содержит последовательность операторов реализующих процедуру инициализации всего функционала программы и основной программный цикл, обеспечивающий выполнение всех функций программы.

get\_pdo\_work\_mode – функция, возвращающая флаги текущего режима работы программы («нормальный режим», «сбой при тестировании аппаратуры», «нет конфигурации»).

pdo\_led\_hwerror – подпрограмма обеспечивающая индикацию аппаратного сбоя после начального самотестирования.

pdo\_led\_start\_task – подпрограмма обеспечивающая индикацию загрузочной последовательности после начального самотестирования.

Так же модуль содержит следующие программные элементы, генерируемые автоматически в среде STM32CubeIDE согласно описания используемых модулей микроконтроллера:

SystemClock\_Config – подпрограмма настройки тактирования ядра микроконтроллера.

PeriphCommonClock\_Config – подпрограмма настройки тактирования внутренних аппаратных модулей микроконтроллера.

MPU\_Config – подпрограмма настройки модуля защиты памяти контроллера памяти.

MX\_GPIO\_Init – подпрограмма настройки портов ввода/вывода микроконтроллера.

MX\_DMA\_Init – подпрограмма настройки модуля DMA микроконтроллера.

MX\_FMC\_Init – подпрограмма настройки модуля контроллера внешней памяти.

MX\_I2C1\_Init – подпрограмма настройки модуля шины I2C1.

MX\_I2C2\_Init – подпрограмма настройки модуля шины I2C2.

MX\_I2C3\_Init – подпрограмма настройки модуля шины I2C3.

MX\_SAI1\_Init – подпрограмма настройки модуля шины SAI1.

MX\_UART7\_Init – подпрограмма настройки модуля приемопередатчика UART7.

MX\_TIM3\_Init – подпрограмма настройки модуля таймера TIM3.

Error\_Handler – подпрограмма обработки аппаратных ошибок при настройке аппаратных модулей микроконтроллера.

3.8 **driver\_UI.c** – модуль драйвера кнопок и клавиатуры, обеспечивающий обмен с платой УИ по I2C-шине для опроса кнопок и индикации состояния, а также опрос кнопок и управление светодиодами платы СЦ4.

Содержит следующие основные программные элементы:

ui\_getbtnval – функция, возвращающая текущее состояние указанной кнопки (нажата/отжата);

ui\_getbtnstate – функция, возвращающая признак изменения состояния указанной кнопки (не изменилось/было изменение);

ui\_getactivebtn – функция, возвращающая номер первой кнопки с изменившимся состоянием и сбрасывающая флаг изменения состояния данной кнопки.

ui\_resetbtnstate – функция, сбрасывающая флаг изменения состояния указанной кнопки.

ui\_setledstate – подпрограмма, устанавливающая состояние указанного светодиода.

ui\_getledstate – функция, возвращающая состояние указанного светодиода.

ui\_init – подпрограмма инициализации модуля driver\_UI. Обеспечивает инициализацию программных структур и аппаратных средств необходимых для работы модуля.

ui\_task – подпрограмма периодической обработки состояния драйвера. Периодический вызов данной подпрограммы обеспечивает обновление состояния кнопок и реализацию режимов мигания светодиодов.

ui\_get\_uimode – функция, возвращающая тип подключенной платы УИ.

ui\_set\_block\_kbd – подпрограмма включения/выключения блокировки состояния всех кнопок. Используется для реализации защиты от дребезга кнопок или быстрых нажатий.

SC4BTN\_EXTI\_IRQHandler\_Config – подпрограмма настройки аппаратных прерываний по изменению состояния сигнальных линий от платы УИ и от кнопки ТЕСТ.

EXTI9\_5\_IRQHandler – обработчик прерывания от сигнальной линии EXTI9\_5 (кнопка ТЕСТ платы СЦ4)

EXTI15\_10\_IRQHandler – обработчик прерывания от сигнальной линии EXTI15\_10 (изменение состояния кнопок на плате УИ).

TLC59116F\_Init – подпрограмма инициализации микросхем драйверов светодиодов на плате УИ по шине I2C.

MCP23017\_Init – подпрограмма инициализации микросхем регистров кнопок на плате УИ по шине I2C.

TLC59116F\_makeledval – функция, формирующая и возвращающая значение регистра микросхемы драйвера светодиода для состояния указанного светодиода.

TLC59116F\_writeled – подпрограмма записи состояния указанного светодиода в микросхему драйвера светодиода на плате УИ по шине I2C.

MCP23017\_readbuttons – подпрограмма чтения текущего состояния кнопок из микросхем регистров кнопок на плате УИ по шине I2C.

MCP23017\_readbuttonsCapture – подпрограмма чтения состояния кнопок в момент прерывания из микросхем регистров кнопок на плате УИ по шине I2C.

ui\_updateSC4btn – подпрограмма обновления состояния кнопок платы СЦ4.

ui\_updateLED – подпрограмма обновления состояния светодиодов платы СЦ4.

3.9 **driver\_pinIO.c** – модуль драйвера линий ввода/вывода. Обеспечивает опрос контактов внешних подключений (плата УИ, аудиоустройства) и выдачу сигналов управления устройствами, подключенными к плате (усилитель, реле).

Содержит следующие основные программные элементы:

pinio\_init – подпрограмма инициализации модуля driver\_pinIO. Обеспечивает инициализацию программных структур и аппаратных средств необходимых для работы модуля.

pinio\_task – подпрограмма периодической обработки состояния драйвера. Периодический вызов данной подпрограммы обеспечивает обновление состояния сигнальных линий.

pinio\_set\_UPR\_SP – подпрограмма устанавливающая значение линии UPR\_SP (усилитель внутреннего громкоговорителя).

pinio\_set\_UPR\_FAN – подпрограмма устанавливающая значение линии UPR\_FAN (управление фантомным сигналом).

pinio\_set\_UPR\_RELE – подпрограмма устанавливающая значение линии UPR\_RELE (управление контактами реле).

pinio\_get\_CFGUI – функция, читающая с линий ввода и возвращающая код типа подключенной платы УИ.

pinio\_getstate\_DET\_48V – функция, возвращающая признак изменения состояния линии DET\_48V (питание PoE).

pinio\_getval\_DET\_48V – функция, читающая с линий ввода и возвращающая состояние линии DET\_48V (питание PoE).

pinio\_getstate\_DET\_PHONE – функция, возвращающая признак изменения состояния линии DET\_PHONE (подключение внешних телефонов). pinio\_getval\_DET\_PHONE – функция, читающая с линий ввода и возвращающая состояние линии DET\_PHONE (подключение внешних телефонов).

pinio\_getstate\_DET\_MIC – функция, возвращающая признак изменения состояния линии DET\_MIC (подключение внешнего микрофона).

pinio\_getval\_DET\_MIC – функция, читающая с линий ввода и возвращающая состояние линии DET\_MIC (подключение внешнего микрофона).

3.10 **driver\_extRAM.c** – модуль драйвера внешней памяти. Обеспечивает начальное тестирование внешней памяти при запуске программы и распределение данных во внешней микросхеме ОЗУ для временного хранения массивов данных (конфигурация, прошивка).

Содержит следующие основные программные элементы:

extram\_init – подпрограмма инициализации модуля driver\_extRAM. Обеспечивает инициализацию программных структур для работы с внешней памятью и начальное тестирование внешней памяти.

extram\_get\_freeRAM – функция, возвращающая оставшееся количество свободной нераспределенной внешней памяти.

extram\_allocate\_static – функция выделения статического блока во внешней памяти. Возвращает указатель на выделенный блок или 0 – если не осталось свободной памяти.

3.11 **driver\_EEPROM.c** – модуль драйвера энергонезависимой памяти EEPROM. Обеспечивает начальное тестирование микросхемы EEPROM, обмен данными с микросхемой EEPROM по I2C-шине для хранения блоков конфигурационных данных.

Содержит следующие основные программные элементы:

BlockEntry - тип структура, описывающая блок конфигурационных данных;

TelBlock – тип структура, описывающая элемент абонентского справочника;

SystemBlockHdr – тип структура, описывающая главный системный блок;

eeprom\_hdr – структура для хранения содержимого главного системного блока;

eeprom\_init – функция инициализации модуля driver\_EEPROM. Обеспечивает инициализацию программных структур для работы с EEPROM, начальное тестирование EEPROM, загрузку главного системного блока и конфигурационных данных при их наличии в EEPROM. Возвращает код результата выполнения инициализации.

eeprom\_get\_system\_SB – функция, возвращает ссылку структуру главного системного блока.

eeprom\_read\_SB – функция чтения главного системного блока из EEPROM, возвращает код результата операции.

eeprom\_write\_SB – функция записи главного системного блока в EEPROM, возвращает код результата операции.

eeprom\_read\_block – функция чтения указанного конфигурационного блока из EEPROM, возвращает код результата операции.

eeprom\_write\_block – функция записи указанного конфигурационного блока из EEPROM, возвращает код результата операции.

eeprom\_default\_SB – подпрограмма инициализации структуры главного системного блока значениями по умолчанию.

eeprom\_update\_block\_CRC – подпрограмма вычисления контрольной суммы указанного конфигурационного блока и записи её в главный системный блок.

eeprom\_find\_block – функция поиска структуры описателя конфигурационного блока указанного типа в главном системном блоке, возвращает индекс найденной структуры или -1, если структура не найдена.

eeprom\_print\_SB – подпрограмма печати данных главного системного блока через интерфейс RS232.

eeprom\_get\_totalsize – функция, вычисляющая и возвращающая общий объем конфигурационных данных по информации в главном системном блоке.

3.12 **driver\_AIC.c** – модуль драйвера аудиокодека TLV320AIC3254. Обеспечивает инициализацию и начальное тестирование микросхемы аудиокодека, ввод и вывод звука через аудиокодек по шине SAI через каналы DMA, обмен управляющими данными с аудиокодеком по шине I2C для регулировки громкости выводимого звука и чувствительности микрофона, переключения входов аудиоустройств.

Содержит следующие основные программные элементы:

TLV320\_Read – функция чтения по шине I2C указанного регистра аудиокодека на текущей регистровой странице. Возвращает HAL-код результата операции.

TLV320\_ReadPage – функция чтения по шине I2C указанного регистра аудиокодека с переключением на указанную регистровую странице. Возвращает HAL-код результата операции.

TLV320\_Write – функция записи по шине I2C указанного регистра аудиокодека на текущей регистровой странице. Возвращает HAL-код результата операции.

TLV320\_WritePage – функция записи по шине I2C указанного регистра аудиокодека с переключением на указанную регистровую странице. Возвращает HAL-код результата операции.

TLV320\_WriteSq – функция последовательной записи по шине I2C значений регистров из указанного массива пар регистр/значение. Возвращает HAL-код результата операции.

TLV320\_Reset – функция, выполняющая последовательность сброса микросхемы аудиокодека по шине I2C. Возвращает HAL-код результата операции.

HAL\_SAI\_TxHalfCpltCallback – обработчик прерывания окончания передачи первого звукового буфера по каналу DMA.

HAL\_SAI\_TxCpltCallback – обработчик прерывания окончания передачи второго звукового буфера по каналу DMA.

HAL\_SAI\_RxHalfCpltCallback – обработчик прерывания окончания приема первого звукового буфера по каналу DMA.

HAL\_SAI\_RxCpltCallback – обработчик прерывания окончания приема второго звукового буфера по каналу DMA.

aic\_setDACOutVolume – функция установки уровня усиления ЦАП аудиокодека. Возвращает HAL-код результата операции.

aic\_setADCInVolume – функция установки уровня усиления АЦП аудиокодека. Возвращает HAL-код результата операции.

aic\_setDACMute – функция включения/выключения усилителя ЦАП аудиокодека. Возвращает HAL-код результата операции.

aic\_setADCMute – функция включения/выключения усилителя АЦП аудиокодека. Возвращает HAL-код результата операции.

aic\_getDACFlags – функция чтения флагов состояния ЦАП аудиокодека в указанную переменную. Возвращает HAL-код результата операции.

aic\_getADCFlags – функция чтения флагов состояния АЦП аудиокодека в указанную переменную. Возвращает HAL-код результата операции.

aic\_setSAIloop – подпрограмма включения/выключения программной аудиопетли (AIC\_ADC -> SAI\_DMA\_RX -> SAI\_DMA\_TX -> AIC\_DAC) для проверки аудиотракта.

aic\_init – функция инициализации модуля driver\_AIC. Обеспечивает инициализацию программных структур для работы с аудиокодеком, инициализацию и начальное тестирование аудиокодека, запуск обмена по каналам DMA. Возвращает код результата выполнения инициализации.

aic\_deinit – функция деинициализации обмена с аудиокодеком по каналам DMA.

aic\_task – подпрограмма периодической обработки состояния драйвера. Периодический вызов данной подпрограммы обеспечивает обновление состояний каналов DMA, ввод и вывод звуковых буферов из модуля обработки звука, генерацию звуковых информационных сигналов.

aic\_setOutDev – подпрограмма переключения на указанное аудиоустройство вывода звука.

aic\_setInDev – подпрограмма переключения на указанное аудиоустройство ввода звука.

aic\_setInVolUp – подпрограмма увеличения уровня усиления звука с микрофона на один шаг.

aic\_setInVolDown – подпрограмма уменьшения уровня усиления звука с микрофона на один шаг.

aic\_setOutVolUp – подпрограмма увеличения уровня усиления воспроизводимого звука на один шаг.

aic\_setOutVolDown – подпрограмма уменьшения уровня усиления воспроизводимого звука на один шаг.

3.13 **dp83848.c** – модуль драйвера PHY-контроллера DP83848.

Обеспечивает инициализацию, начальное тестирование и управление PHY-контроллером Ethernet по шине MDIO.

Содержит следующие основные программные элементы:

DP83848\_IOCtx\_t – тип структура для хранения информации об интерфейсе драйвера для модуля реализации Ethernet интерфейса ethernetif.

DP83848\_Object\_t – тип структура для хранения информации о PHY-контроллере для модуля реализации Ethernet интерфейса ethernetif.

DP83848\_RegisterBusIO – подпрограмма обеспечивает инициализацию программной структуры интерфейса драйвера для модуля реализации Ethernet интерфейса ethernetif.

DP83848\_Init – функция инициализации PHY-контроллера. Обеспечивает инициализацию программных структур для работы с PHY-контроллером, инициализацию и начальное тестирование PHY-контроллера. Возвращает код результата выполнения инициализации.

DP83848\_StartAutoNego – функция запуска процедуры автосогласования параметров линии Ethernet в PHY-контроллере. Возвращает код результата выполнения операции.

DP83848\_GetLinkState – функция, выполняющее чтение и возвращающая код состояния линии Ethernet PHY-контроллера.

3.14 **ethernetif.c** – стандартный модуль реализации Ethernet интерфейса для микроконтроллера STM Cortex-M7. Обеспечивает обмен данными с сетью Ethernet через интерфейс MII по каналам DMA через PHY-контроллер.

Содержит следующие основные программные элементы:

ethernetif\_init – функция инициализации Ethernet интерфейса. Обеспечивает инициализацию программных структур для работы с PHY-контроллером, инициализацию встроенного аппаратного модуля MAC-контроллера Ethernet и каналов DMA. Возвращает код результата выполнения инициализации.

low\_level\_output – подпрограмма отправки Ethernet-пакетов PHY-контроллеру из стека lwIP.

ethernetif\_input – подпрограмма обработки принятых Ethernet-пакетов от PHY-контроллера и передачу их стеку lwIP.

ethernet\_link\_check\_state – подпрограмма периодического опроса состояния линии Ethernet и включение/выключение обмена пакетами по каналам DMA с PHY-контроллером в зависимости от состояния линии. Вызывается периодически из модуля управления библиотекой lwIP.

3.15 **lwip.c** – стандартный модуль управления библиотекой lwIP. Обеспечивает инициализацию и периодическую обработку стека lwIP.

Содержит следующие основные программные элементы:

MX\_LWIP\_Init – подпрограмма инициализации библиотеки lwIP для микроконтроллера STM. Обеспечивает инициализацию программных структур для библиотеки lwIP и связь с Ethernet интерфейсом (ethernetif).

MX\_LWIP\_Process – подпрограмма периодической обработки стека lwIP. Периодически вызывается из главного цикла программы для обеспечения работы стека lwIP.

ethernet\_link\_status\_updated – подпрограмма обработки изменения состояния линии Ethernet. Обеспечивает индикацию светодиодом НОРМА состояния линии Ethernet и вывод сообщения через RS232 при изменении состояния (в режимах SERVICE/DEBUG).

3.16 **ping.c** – модуль реализации команды PING протокола ICMP. Обеспечивает отправку ICMP-запроса PING на заданный IP-адрес и обработку ответа на запрос. Используется для реализации теста обмена данными по сети Ethernet.

Содержит следующие основные программные элементы:

ping\_init – инициализация структур стека lwIP для ICMP-запроса PING с указанным IP-адресом.

ping\_free – освобождает структуры стека lwIP используемые для ICMP-запроса PING.

ping\_send – выполняет отправку ICMP-запроса PING.

ping\_get\_result – возвращает статус ответа на ICMP-запрос PING.

ping\_recv – обработчик стека lwIP, декодирующий и проверяющий ICMP-ответ.

3.17 **rtp.c** – модуль реализации сетевого протокола RTP для передачи аудиоданных. Обеспечивает формирование RTP-пакетов для отправки по сети и декодирование принимаемых из сети RTP-пакетов.

Содержит следующие основные программные элементы:

rtp\_hdr – структура, описывающая заголовок RTP-пакета.

rtp\_hdr\_init – подпрограмма инициализации структуры заголовка RTP-пакета с указанным идентификатором аудиоисточника.

rtp\_hdr\_encode – подпрограмма формирования и записи RTP-заголовка в указанный UDP-пакет.

rtp\_hdr\_decode – функция декодирования принятого RTP-пакета. Возвращает код идентификации аудиоисточника или 0 – если пакет имеет неверный формат.

3.18 **cbuffer.c** – модуль реализации кольцевого байтового FIFO-буфера. Используется программой для хранения обрабатываемых аудиоданных.

Содержит следующие основные программные элементы:

CBuffer – структура, описывающая кольцевой байтовый буфер.

CBuffer\_Init – подпрограмма инициализации кольцевого байтового буфера.

CBuffer\_Reset – подпрограмма сброса (очистки) кольцевого байтового буфера.

CBuffer\_ReadByte – функция чтения байта из кольцевого байтового буфера. Возвращает значение байта из буфера или 0, если буфер пуст.

CBuffer\_WriteData – функция записи указанного количества из массива байт в кольцевой байтовый буфер. Возвращает реальное количество записанных байт из буфера или 0, если буфер полностью заполнен.

CBuffer\_ReadData – функция чтения указанного количества байт в массив байт из кольцевого байтового буфера. Возвращает реальное количество считанных байт из буфера или 0, если буфер пуст.

3.19 **pckt\_buffer.c** – модуль реализации кольцевого пакетного FIFO-буфера. Используется программой для хранения сетевых пакетов.

Содержит следующие основные программные элементы:

PktBuffer – структура, описывающая кольцевой пакетный буфер.

PktBuffer\_Init – подпрограмма инициализации кольцевого пакетного буфера.

PktBuffer\_Reset – подпрограмма сброса (очистки) кольцевого пакетного буфера.

PktBuffer\_Write – функция записи указанного пакета в кольцевой пакетный буфер. Возвращает 1, если пакет записан или 0, если буфер полностью заполнен.

PktBuffer\_Read – функция чтения пакета в указанный буфер из кольцевого пакетного буфера. Возвращает 1, если пакет прочитан или 0, если буфер пуст.

PktBuffer\_GetNextPktSize – функция возвращающая размер следующего пакета в кольцевом пакетном буфере или 0, если буфер пуст.

3.20 **CRC32.c** – модуль реализации вычисления контрольных сумм CRC32.

Содержит следующие основные программные элементы:

crc32NT – функция, вычисляющая методом прямого расчета и возвращающая контрольных сумму CRC32 данных из указанного массива.

crc32T – функция, вычисляющая табличным методом и возвращающая контрольных сумму CRC32 данных из указанного массива. В программе не используется.

crc32\_calc – макрозамена, используемая программой для обращения к функции crc32NT.

3.21 **AES\_codec.c** – модуль реализации шифрования и дешифрования данных блочным алгоритмом AES-128.

Содержит следующие основные программные элементы:

AES\_State – структура настройки алгоритма AES.

AES\_key\_init – подпрограмма инициализации структуры AES\_State указанным ключом шифрования.

AES\_code\_data – подпрограмма шифрования указанного массива байт алгоритмом AES.

AES\_decode\_data – подпрограмма дешифрования указанного массива байт алгоритмом AES.

AES\_get\_static\_state – функция, возвращающая ссылку на статическую структуру AES\_State.

3.22 **g711.c** – модуль кодирования и декодирования аудиоданных кодеком G.711 A-law.

Содержит следующие основные программные элементы:

g711\_pcm2alaw – подпрограмма кодирования указанного массива 16-битных аудиоданных в указанный массив байт кодеком G.711 A-law.

g711\_alaw2pcm – подпрограмма декодирования указанного массива байт в указанный массив 16-битных аудиоданных кодеком G.711 A-law.

3.23 **system\_settings.c** – модуль работы с конфигурационными данными платы СЦ4. Обеспечивает загрузку, хранение в ОЗУ и обработку конфигурационных данных.

Содержит следующие основные программные элементы:

sPDOSettings – структура описывающая конфигурационные данные.

sButtonSettings – структура описывающая конфигурацию кнопки.

sGroupsSettings – структура описывающая конфигурацию группы связи.

sStationSettings – структура описывающая конфигурацию станции.

sNetworkSettings – структура описывающая сетевые настройки.

sTelBlockEntry – структура описывающая элемент абонентского справочника.

json\_static\_block – массив конфигурационных данных формата JSON, хранимый во внешней памяти.

phonebook\_block – массив абонентского справочника хранимый во внешней памяти.

default\_net\_settings – константная структура описывающая сетевые настройки по умолчанию.

sysset\_init – функция загрузки из EEPROM массива конфигурационных данных формата JSON во внешнюю память и инициализации ими конфигурационной структуры. Возвращает код результата инициализации.

sysset\_load – функция, выполняющая разбор массива конфигурационных данных формата JSON из внешней памяти в конфигурационную структуру. Возвращает результат операции разбора.

sysset\_get\_settings – функция, возвращающая ссылку на статическую структуру с конфигурационными данными.

sysset\_get\_mainconf\_adr – функция, возвращающая ссылку на массив конфигурационных данных формата JSON.

sysset\_get\_telconf\_adr – функция, возвращающая ссылку на массив абонентского справочника.

sysset\_print\_settings – подпрограмма печати данных конфигурационной структуры через интерфейс RS232.

3.24 **json\_settings\_parser.c** – модуль разбора конфигурационных данных платы СЦ4 формата JSON.

Содержит следующие основные программные элементы:

json\_tokens – статический массив JSON-токенов для модуля jsonmn.

Parse\_JSON – функция, выполняющая разбор конфигурационных данных формата JSON и заполнение конфигурационной структуры. Возвращает код результата разбора.

3.25 **jsonmn.c** – модуль синтаксического разбора (парсера) данных формата JSON на основе адаптированной для микроконтроллера STM32 библиотеки jsonmn.

Содержит следующие основные программные элементы:

jsmntok\_t – структура, описывающая токен - синтаксическую единицу JSON.

jsmn\_parser – структура, описывающая состояние парсера.

jsmn\_init – подпрограмма инициализации парсера JSON.

jsmn\_parse – функция разбора данных формата JSON на синтаксические единицы (токены). Возвращает код результата разбора (успех / синтаксическая ошибка / недостаточно памяти для токенов).

3.26 **audio\_process.c** – модуль обработки аудиоданных. Обеспечивает буферизацию, кодирование/декодирование аудиоданных при помощи кодека G711, микширование принимаемых аудиоданных.

Содержит следующие основные программные элементы:

audio\_mic\_input – структура кольцевого буфера входного аудиоканала с микрофона.

audio\_mix\_output – структура кольцевого буфера выходного аудиоканала.

audio\_mix\_channels – массив структур кольцевых буферов каналов микширования.

audio\_mix\_channels\_state – массив состояний каналов микширования аудиоданных.

audio\_init – подпрограмма инициализации модуля обработки аудиоданных.

audio\_start – подпрограмма запуска процесса обработки аудиоданных.

audio\_stop – подпрограмма останова процесса обработки аудиоданных.

audio\_process – подпрограмма периодического процесса обработки аудиоданных. Вызывается из основного цикла программы и обеспечивает формирование и отправку RTP-пакетов входных аудиоданных с микрофона сетевым абонентам в зависимости от режимов связи.

HAL\_TIM\_PeriodElapsedCallback – обработчик прерывания таймера микширования аудиоданных. Обеспечивает синхронное декодирование и микширование данных, принятых по сети, в выходной кольцевой буфер.

audio\_write\_input – функция записи аудиоданных во входной кольцевой буфер. Возвращает код результата записи. Используется драйвером аудиокодека для записи аудиоданных с микрофона для дальнейшей обработки.

audio\_get\_output – функция чтения аудиоданных из выходного кольцевого буфера. Возвращает код результата чтения. Используется драйвером аудиокодека для получения выводимых аудиоданных.

audio\_write\_mix\_channel – функция записи аудиоданных в указанный буфер канала микширования. Возвращает код результата записи. Используется модулем протокола UDP для передачи аудиоданных, принимаемых из сети, модулю обработки аудиоданных.

audio\_get\_input – функция чтения аудиоданных из входного кольцевого буфера. Возвращает код результата чтения. Используется для получения аудиоданных, которые отправляются сетевым абонентам.

audio\_get\_mix\_channel – функция выделения нового канала микширования. Возвращает индекс свободного канала микширования или   
-1, если все каналы заняты.

audio\_free\_mix\_channel – подпрограмма освобождения указанного канала микширования.

audio\_reset\_mix\_channel – функция сброса (очистки) буфера указанного канала микширования. Возвращает индекс канала, или -1, если канал с указанным индексом не выделен.

audio\_play\_mix\_channel – функция, запускающая микширование указанного канала. Возвращает индекс канала, или -1, если канал с указанным индексом не выделен.

audio\_stop\_mix\_channel – функция, останавливающая микширование указанного канала. Возвращает индекс канала, или -1, если канал с указанным индексом не выделен.

3.27 **tone\_gen\_state\_processor.c** – модуль генерации звуковых информационных сигналов.

Содержит следующие основные программные элементы:

sTONE\_gen\_state – структура, описывающая состояние генерации звукового информационного сигнала.

tone\_genInit – инициализация модуля генерации звуковых информационных сигналов.

tone\_genPlayTone – инициализация генерации и запуск проигрывания звукового информационного сигнала с указанным кодом.

tone\_genIsTone – функция, возвращающая признак текущего проигрывания звукового информационного сигнала.

tone\_genStop – остановка проигрывания текущего звукового информационного сигнала.

tone\_genProcess – периодическая обработка состояния проигрываемого звукового информационного сигнала. Вызывается в основном программном цикле.

tone\_genGetData – функция получения указанного количества аудиоданных текущего проигрываемого звукового информационного сигнала в указанный массив. Возвращает количество сгенерированных аудиоданных (0 – если проигрывание завершено).

3.28 **tone\_generator.c** – модуль генерации тональных звуковых сигналов.

Содержит следующие основные программные элементы:

gen\_freq\_coef – таблица параметров тональных сигналов.

Freq\_GenInit – процедура инициализация генерации тонального звукового сигнала с указанным индексом по таблице gen\_freq\_coef.

Freq\_GenSample – функция, вычисляющая и возвращающая очередной 16-битный отсчет тонального звукового сигнала.

3.29 **CLI\_CmdTable.c** – модуль таблицы консольных команд. Используется для обработки и выполнения команд, получаемых по интерфейсу RS232 или по сети.

Содержит следующие основные программные элементы:

sCLI\_cmd\_desc – структура описания параметров команды.

cmd\_tab – таблица команд.

CLI\_searchCMD – функция поиска команды из указанной строки в таблице команд. Возвращает ссылку на элемент таблицы или 0 – если команда не найдена.

3.30 **CLI\_CmdProcess.c** – модуль обработки консольных команд. Используется для обработки и выполнения команд, получаемых по интерфейсу RS232 или по сети.

Содержит следующие основные программные элементы:

arrPSTR – массив указателей на строки – параметры команды.

CLI\_executeCMD – подпрограмма определяет и вызывает функцию обработки указанной строки с консольной командой согласно таблице команд.

CLI\_getParNumMsg – функция подсчета и возврата количества параметров в указанной строке с командой.

CLI\_parse\_msg – подпрограмма разбора указанной строки с командой на отдельные параметры.

3.31 **CLI\_io.c** – модуль ввода/вывода для обработки команд. Обеспечивает прием и обработку команд по интерфейсу RS232, обработку команд, принятых из сети, формирование и выдачу ответа на команды по интерфейсу RS232 или по сети.

Содержит следующие основные программные элементы:

CLI\_init – подпрограмма инициализации модуля ввода/вывода.

CLI\_uart\_input – подпрограмма обработки и выполнения команд, принятых по интерфейсу RS232.

CLI\_ethernet\_input – подпрограмма обработки, дешифрования и выполнения команд, принятых из сети.

CLI\_print – подпрограмма формирования сообщения и печати его по интерфейсу RS232, или в режиме ответа на команду - формирования ответа и отправки его по интерфейсу RS232 или с шифрованием по сети, в зависимости откуда пришел запрос.

CLI\_print\_lev – подпрограмма печати сообщения по интерфейсу RS232 с учетом уровня печати текущего режима работы платы СЦ4 (DEBUG/SERVICE/TEST/WORK).

CLI\_setPrintLevel – подпрограмма установки уровня печати.

CLI\_uart\_task – подпрограмма периодической обработки данных, поступающих по интерфейсу RS232, формирование из них строк и их обработку. Вызывается в основном цикле программы.

HAL\_UARTEx\_RxEventCallback – обработчик прерывания модуля UART для приема данных из интерфейса RS232.

Hex2Int – функция преобразования строки со значением в HEX-формате в возвращаемое числовое значение.

Int2Hex – подпрограмма преобразования числового значения в строку в HEX-формате.

Hex2Array – функция преобразования строки со значениями в HEX-формате в массив байт. Возвращает: 1 – преобразование успешно; 0 – синтаксическая ошибка.

Array2Hex – подпрограмма преобразования массива байт в строку значений в HEX-формате.

3.32 **CLI\_CmdDeviceFn.c** – модуль обработки команд общего назначения.

Содержит следующие основные программные элементы:

fnResetDevice – подпрограмма обработки команды программного сброса устройства.

fnGetType – подпрограмма обработки команды получения типа устройства.

fnGetFirmwareInfo – подпрограмма обработки команды получения информации о прошивку устройства.

printDeviceID – подпрограмма печати типа устройства.

checkFirmware – подпрограмма печати информации о прошивке устройства.

3.33 **CLI\_CmdServiceFn.c** – модуль обработки команд ПО «Конфигуратор». Обеспечивает реализацию сетевого протокола обмена и обмена по интерфейсу RS232 с ПО «Конфигуратор» для защищенного реконфигурирования и перепрограммировия платы СЦ4.

Содержит следующие основные программные элементы:

service\_init – инициализация модуля обработки команд ПО «Конфигуратор».

service\_getmode – функция возвращающая текущий режим работы с ПО «Конфигуратор» (нет обмена, режим конфигурации, режим прошивки)

fnConfigOn – подпрограмма обработки команды включения режима конфигурирования.

fnConfigOFF – подпрограмма обработки команды выключения режима конфигурирования.

fnFmwareOn – подпрограмма обработки команды включения режима прошивки.

fnFmwareOFF – подпрограмма обработки команды выключения режима прошивки.

fnSetBlock – подпрограмма обработки команды передачи блока данных от ПО «Конфигуратор».

fnGetBlock – подпрограмма обработки команды запроса блока данных от ПО «Конфигуратор».

fnConfigStartWrite – подпрограмма обработки команды старта записи конфигурационных данных в микросхему EEPROM.

fnFmwareStartWrite – подпрограмма обработки команды старта записи принятого образа программы во внутреннюю программную флеш-память.

3.34 **CLI\_CmdTestFn.c** – модуль обработки команд тестирования, сервисного и отладочного режимов.

Содержит следующие основные программные элементы:

test\_init – подпрограмма инициализация модуля.

test\_task – подпрограмма периодической обработки в режиме теста кнопок и клавиатуры. Вызывается из основного цикла программы.

test\_get\_mode – функция возвращает флаги текущего режим работы модуля.

fnStartTestMode – подпрограмма обработки команды включения тестового режима TEST.

fnStopTestMode – подпрограмма обработки команды выключения тестового режима TEST.

fnStartDebugMode – подпрограмма обработки команды включения отладочного режима DEBUG.

fnStopDebugMode – подпрограмма обработки команды выключения отладочного режима DEBUG.

fnStartServiceMode – подпрограмма обработки команды включения сервисного режима SERVICE.

fnStopServiceMode – подпрограмма обработки команды выключения сервисного режима SERVICE.

fnTestSRAM – подпрограмма обработки команды тестирования внешней памяти.

fnTestEEPROM – подпрограмма обработки команды тестирования микросхемы EEPROM.

fnTestCODEC – подпрограмма обработки команды тестирования микросхемы аудиокодека.

fnTestPHY – подпрограмма обработки команды тестирования микросхемы PHY-контроллера.

fnTestLEDOn – подпрограмма обработки команды включения режима тестирования светодиодов.

fnTestLEDOff – подпрограмма обработки команды выключения режима тестирования светодиодов.

fnTestBTNOn – подпрограмма обработки команды включения режима тестирования кнопок.

fnTestBTNOff – подпрограмма обработки команды выключения режима тестирования кнопок.

fnTestAudioLoopOn – подпрограмма обработки команды включения режима тестирования аналоговых трактов (аудиопетля).

fnTestAudioLoopOff – подпрограмма обработки команды выключения режима тестирования аналоговых трактов.

fnTestTone – подпрограмма обработки команды тестирования воспроизведения звукового информационного сигнала.

fnTestPrn – подпрограмма обработки команды тестовой печати информации.

fnSetPIN – подпрограмма обработки команды установки значения линии вывода.

fnGetPIN – подпрограмма обработки команды получения значения сигнальной линии ввода.

fnSetLED – подпрограмма обработки команды установки состояния светодиода.

fnGetBTN – подпрограмма обработки команды получения состояния кнопки.

fnSetAIC – подпрограмма обработки команды установки значения регистра аудиокодека.

fnGetAIC – подпрограмма обработки команды чтения значения регистра аудиокодека.

fnSetETH – подпрограмма обработки команды установки значения регистра PHY-контроллера.

fnGetETH – подпрограмма обработки команды чтения значения регистра PHY-контроллера.

3.35 **MemTest.c** – модуль тестирования памяти.

Содержит следующие основные программные элементы:

SRam\_Status – структура для хранения информации о результате тестировании памяти.

Test\_RAM – функция полного тестирования памяти. Возвращает флаги результатов тестов памяти. Применяется для тестирования внешней памяти SRAM в режиме TEST.

Test\_RAM\_fast – функция быстрого тестирования памяти. Возвращает 1- при успешном тестировании, 0 – при обнаруженном сбое памяти. Применяется для начального тестирования внешней памяти SRAM при запуске программы.

Test\_RAM\_printResult – подпрограмма печати результатов тестирования памяти через интерфейс RS232.

TestRAM\_byteio – подпрограмма теста байтовой записи памяти.

TestRAM\_runzeros – подпрограмма теста шины данных памяти по алгоритму «бегущий ноль».

TestRAM\_runones – подпрограмма теста шины данных памяти по алгоритму «бегущая единица».

TestRAM\_address\_ptr – подпрограмма теста шины адреса памяти по алгоритму «запись адреса».

TestRAM\_address\_cnt – подпрограмма теста шины адреса памяти по алгоритму «запись счетчика».

TestRAM\_hashaddress – подпрограмма теста шины адреса памяти по алгоритму «запись hash-последовательности».

3.36 **fmware\_flasher.c** – модуль прошивки внутренней программной flash-памяти микроконтроллера. При запуске программы код модуля перемещается во внутреннее ОЗУ микроконтроллера и обеспечивает свою работу независимо от основной программы.

Основная подпрограмма модуля – WriteFirmware – выполняет процедуру прошивки программной flash-памяти микроконтроллера образом программы, расположенной во внешней памяти. При старте процедуры прошивки блокирует основную программу и отключает обработку прерываний, после окончания прошивки выполняет перезагрузку микроконтроллера и запуск нового образа программы.

3.37 **udp\_exchange.c** – модуль обмена по протоколу UDP. Обеспечивает обмен данными по протоколу UDP при помощи стека lwIP для реализации межприборного обмена и документирования работы.

Содержит следующие основные программные элементы:

udp\_ips\_init – подпрограмма инициализации модуля обмена по протоколу UDP.

udp\_ips\_send – подпрограмма отправки UDP-пакетов через стек lwIP.

udp\_ips\_send\_rtp – подпрограмма отправки RTP-пакетов c аудиоданными указанным абонентам.

udp\_ips\_send\_log – подпрограмма отправки пакетов с состоянием на UDP-порт сервера документирования.

udp\_ips\_send\_rtp\_log – подпрограмма отправки RTP-пакетов c аудиоданными на UDP-порт сервера документирования.

udp\_rec\_callback – обработчик для стека lwIP UDP-порта приема RTP-пакетов c аудиоданными.

udp\_cfg\_callback – обработчик для стека lwIP UDP-порта приема командных пакетов.

CmdCfgBuffer\_GetNextPktSize – функция возвращающая размер следующего принятого UDP-пакета (0 – если пакетов нет).

cmdcfg\_pckt\_read – подпрограмма чтения очередного принятого UDP-пакета из кольцевого буфера пакетов.

3.38 **connect\_manager.c** – модуль менеджера абонентских соединений и обработки команд протоколов межприборного обмена и документирования событий.

Содержит следующие основные программные элементы:

cmanager\_struct – структура описания абонентского соединения.

cm\_array – массив структур описания абонентских соединений.

fnReceivedIPSPING – подпрограмма обработки команды опроса устройства.

fnReceivedIPSPONG – подпрограмма обработки ответа на опрос устройства.

fnReceivedINVITE – подпрограмма обработки команды запроса на установление связи.

fnReceivedACK – подпрограмма обработки ответа на запрос.

fnReceivedINVITEACK – подпрограмма обработки команды подтверждения дуплексного вызова абонентом.

fnReceivedBYE – подпрограмма обработки команды запроса завершения связи.

fnReceivedLOG – подпрограмма обработки команды запроса от сервера документирования.

cmanager\_init – подпрограмма инициализации модуля менеджера абонентских соединений.

cmanager\_get\_audiobuf – функция выделяет канал микширования для приема звука абонентского соединения. Возвращает индекс выделенного канала (-1, если нет свободных каналов).

cmanager\_get\_bufnum – функция возвращает индекс канала микширования для указанного абонентского соединения (-1, если канал не выделен).

cmanager\_free\_audiobuf – подпрограмма освобождает указанный канал микширования.

cmanager\_get\_ip – функция, выполняющая поиск и возвращающая индекс в массиве абонентских соединений по указанному IP-адресу абонента (-1, если абонентское соединение для абонента не найдено).

cmanager\_process\_cmd – подпрограмма декодирования и выполнения принятой по сети команды.

cmanager\_process\_button – подпрограмма обработки изменения состояния указанной абонентской кнопки.

cmanager\_task – подпрограмма периодической обработки состояния абонентских соединений. Вызывается из главного цикла программы.

fnSendIPSPING – подпрограмма формирования и отправки команды опроса устройства.

fnSendIPSPONG – подпрограмма формирования и отправки ответа на опрос устройства.

fnSendINVITE – подпрограмма формирования и отправки команды запроса на установление связи.

fnSendACK – подпрограмма формирования и отправки ответа на запрос.

fnSendINVITEACK – подпрограмма формирования и отправки команды подтверждения дуплексной связи абонентом

fnSendBYE – подпрограмма формирования и отправки команды запроса на завершение связи.

fnSendLOG – подпрограмма формирования и отправки информации об изменении состояния на сервер документирования.

fnSendAllLOG – подпрограмма формирования и отправки состояния устройства на запрос от сервера документирования.

**4. Тестирование ПО платы СЦ4.**

Для тестирования модулей платы ПО платы СЦ4 обеспечивающих работу с аппаратными ресурсами, а также тестирования алгоритмов управления аппаратными ресурсами была разработана отдельная программа sc4\_test, которая представляет собой отладочную версию основной программы без реализации функций сетевого обмена через стек lwIP и работающую по умолчанию в отладочном и тестовом режимах платы (TEST / DEBUG).

Модули программы sc4\_test аналогичные модулям основной программы:

1. Модули работы с аппаратными ресурсами startup\_stm32h753iitx.s, system\_stm32h7xx.c, stm32h7xx\_it.c, stm32h7xx\_hal\_msp.c,

2. Стандартные модули поддержки системной библиотеки syscalls.c, sysmem.c.

3. Модели драйверов driver\_UI.c, driver\_pinIO.c, driver\_extRAM.c, driver\_EEPROM.c, driver\_AIC.c, dp83848.c.

4. Модули для работы с конфигурационными данными system\_settings.c, json\_settings\_parser.c, jsonmn.c.

5. Модули для генерации звуковых информационных сигналов tone\_gen\_state\_processor.c, tone\_generator.c.

6. Модули обработки команд CLI\_CmdTable.c, CLI\_io.c, CLI\_CmdProcess.c, CLI\_CmdDeviceFn.c, CLI\_CmdServiceFn.c, CLI\_CmdTestFn.c.

7. Вспомогательные модули cbuffer.c, CRC32.c, AES\_codec.c, g711.c.

8. Модуль тестирования памяти MemTest.c,

9. Модуль перепрошивки программной flash-памяти fmware\_flasher.c.

Модули программы sc4\_test дополнительно адаптированные для выполнения программно-аппаратной отладки и тестирования:

ethernetif\_test.c – обеспечивает инициализацию и работу Ethernet интерфейса без использования стека lwIP.

audio\_process.c – дополнительно обеспечивает реализацию аудиокольца через буфера аудиомикшера с использованием кодека G711, для тестирования и отладки алгоритмов обработки аудиоданных.

main.c – обеспечивает в главном программном цикле дополнительный периодический вывод отладочный сообщений, а также запуск при нажатии на кнопки функций тестирования алгоритмов передачи по Ethernet и отображение состояния регистров Phy-контроллера. Остальные тесты запускаются командами из COM-терминала, аналогично, как в основной программе, но с дополнительным выводом отладочных сообщений.