**УТВЕРЖДЕНО**

 79753582.62.01.29.000-01 13 01-ЛУ

**СПЕЦИАЛЬНОЕ програмМное обеспечение**

**«PAK\_OSA»**

 **Описание программы**

**79753582.62.01.29.000-01 13 01**

**Листов 19**

**АННОТАЦИЯ**

В данном программном документе приведено описание специального программного обеспечения «PAK\_OSA», входящего в состав программно-аппаратного комплекса «ОСА».

Документ содержит сведения о логической структуре и функционировании данного специального программного обеспечения.

 Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.101-77[[1]](#footnote-1)1), ГОСТ 19.103-77[[2]](#footnote-2)2), ГОСТ 19.104-78[[3]](#footnote-3)3), ГОСТ 19.105-78[[4]](#footnote-4)4), ГОСТ 19.106-78[[5]](#footnote-5)5), ГОСТ 19.402-78[[6]](#footnote-6)6), ГОСТ 19.604-78[[7]](#footnote-7)7)).

**СОДЕРЖАНИЕ**

[1. Общие сведения 4](#_Toc67054709)

[1.1. Обозначение и наименование программы 4](#_Toc67054710)

[1.2. Программное обеспечение, необходимое для функционирования программы 4](#_Toc67054711)

[1.3. Языки программирования, на которых написана программа 4](#_Toc67054712)

[2. Функциональное назначение 5](#_Toc67054713)

[2.1. Классы решаемых задач 5](#_Toc67054714)

[2.2. Назначение программы 5](#_Toc67054715)

[2.3. Сведения о функциональных ограничениях на применение 5](#_Toc67054716)

[3. Описание логической структуры 6](#_Toc67054717)

[3.1. Алгоритм программы 6](#_Toc67054718)

[3.3. Структура программы с описанием функций составных частей и связи между ними 7](#_Toc67054719)

[3.4. Связи программы с другими программами 13](#_Toc67054720)

[4. Используемые технические средства 14](#_Toc67054721)

[5. Вызов и загрузка 15](#_Toc67054722)

[6. Входные данные 16](#_Toc67054723)

[7. Выходные данные 17](#_Toc67054724)

[Перечень терминов и сокращений 18](#_Toc67054725)

[Лист регистрации изменений 19](#_Toc67054726)

# 1. ОБЩИЕ СВЕДЕНИЯ

## **1.1. Обозначение и наименование программы**

Обозначение: 79753582.62.01.29.000-01.

Полное наименование: Специальное программное обеспечение «PAK\_OSA».

Краткое наименование: СПО «PAK\_OSA».

Версия продукта: 1.3.2.

Производитель: Общество с ограниченной ответственностью «Система».

Язык: Русский.

## **1.2. Программное обеспечение, необходимое для функционирования программы**

Системные программные средства, используемые специальным программным обеспечением «PAK\_OSA», представлены операционной системы семейства Linux.

Для функционирования специального программного обеспечения «PAK\_OSA» предустановленное программное обеспечение стороннего разработчика не требуется.

## **1.3. Языки программирования, на которых написана программа**

 Исходные коды программы реализованы на языке С++. В качестве интегрированной среды разработки программы была использована среда Microsoft Visual Studio Code.

# 2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

## **2.1. Классы решаемых задач**

Специальное программное обеспечение «PAK\_OSA» предназначено для решения следующих задач:

1. двухстороннее взаимодействие пользователя с программой;
2. обработка данных видеоустройств и организация видео регистрации (запись видеоданных на носители информации);
3. организация фото регистрации (запись фото данных на носители информации);
4. организация контроля наличия связи с устройствами и протоколирование нарушений связи с устройствами;
5. организация управления поворотным видеоустройством;
6. распознавание государственных регистрационных знаков (далее – ГРЗ);
7. обработка текущих навигационных параметров по сигналам космических навигационных систем GPS/ГЛОНАСС;
8. формирование данных для обработки в специальном программном обеспечении Центра автоматизированной фиксации административных правонарушений ГИБДД (далее – ЦАФАП ГИБДД);

## **2.2. Назначение программы**

Специальное программное обеспечение «PAK\_OSA» входит в состав программно-аппаратного комплекса «ОСА», предназначенного для автоматической регистрации нарушений правил дорожного движения.

Специальное программное обеспечение «PAK\_OSA» используется для мониторинга и управления оборудованием, входящим в данный программно-аппаратный комплекс, устанавливает логические схемы ее работы, обеспечивает двухстороннее взаимодействие оператора с системой, осуществляет автоматическое формирование материала для постановления о нарушении ПДД.

## **2.3. Сведения о функциональных ограничениях на применение**

Специальное программное обеспечение «PAK\_OSA» входящее в состав программно-аппаратного комплекса «ОСА, является автономной системой и не требует постоянного контроля и управления со стороны оператора.

Специальное программное обеспечение «PAK\_OSA» имеет аппаратный (программный) ключ защиты.

# 3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

## **3.1. Алгоритм программы**

 Алгоритм работы специального программного обеспечения «PAK\_OSA» представлен на рис. 1.

Рисунок 1

**3.2. Используемые методы**

Специальное программное обеспечение «PAK\_OSA» использует следующие методы:

1. асинхронное сетевое взаимодействие;
2. микросервисная архитектура.

3.3. Структура программы с описанием функций составных частей и связи между ними

1. Старт программы происходит с функции main(int argc, char \*argv[]) - точки входа. В этой функции создается объект класса QApplication, стороннего набора библиотек QT, на котором строятся графический интерфейс и некоторые другие функциональные возможности. Далее функция создает объект класса SNSystem и посылает ему команду на запуск. После этого объект класса QApplication выполняет функции exec(), которая является основной.
2. Класс SNSystem, наследуется от класса SNModule, является элементом ядра системы, который запускает и контролирует все модули системы и обеспечивает взаимодействие между ними.
3. ParseIni(). Вызов функции происходит в конструкторе объекта SNSystem. Функция считывает параметры из командной строки исполняемого файла, и запоминает их для дальнейшего использования.
4. CommandModuleStart переопределение функции SNModule::CommandModuleStart, вызов которой инициируется посылкой команды из main(), предназначена для инициализации системы и запуска всех дочерних модулей. Так же в ней происходить проверка присутствия ключа и сертификата функцией SNProtection::GetState().
5. InitializeDatabase(), вызов которой происходит в функции SNSystem::CommandModuleStart. Функция создает и инициализирует объект класса SNMemoryDB для работы с базой данных в памяти и объект класса SNSQLite3Manager для работы с внешней базой данных SQLite. Также функция синхронизирует структуру базы данных с новой версией, описанной в специализированных файлах с расширением \*.xsn.
6. RegisterObject. Инициализация вызова происходит при необходимости зарегистрировать новый объект, например в функции SNModule::CommandModuleStart, посылкой команды объекту SNSystem. Функция регистрирует объект, запоминая его по уникальному идентификатору. При необходимости объект будет передан любому другому объекту. Функция может удалять из своего списка объект, при необходимости.
7. SNModule наследник класса SNObject. Модуль системы, имеющий функционал запуска, проверки прав пользователя и работы с объектом конфигурации. Базовый класс для всех устройств системы.
8. CommandModuleStart виртуальная функция, вызываемая посылкой команды объекту. Функция запуска модуля. В ней инициализируются списки пользователей и их прав на этот модуль. Соединяется с объектом данных SNDataObject данного модуля функцией SNObject:: Connect с событием изменения параметров конфигурации. Производится запись в протокол событий о запуске модуля с помощью функции SNLog::SNLogInfo. Регистрируется новый объект в системе посылкой команды PostCommand объекту SNSystem, что вызывает функцию SNSystem::RegisterObject. CommandModuleStart получает из объекта данных список дочерних модулей, передает его функции OnAddChildModule и получает список запускаемых алгоритмов, настроенных для этого модуля и передает их функции OnAddAlgorithm.
9. InitCreateModule из функции OnAddChildModule. Функция создает объект класса нового модуля. Устанавливает новому модулю объект данных SNDataObject - конфигурацию модуля. Происходит проверка возможности запуска модуля, путем проверки ключа сертификата функцией CheckCertificate. Если проверка сертификата пройдена, выполняется запуск модуля посылкой команды старта функцией PostCommand, что вызовет у модуля функция CommandModuleStart. Возвращаемое значение указатель на объект нового модуля.
10. CheckCertificate вызывается из InitCreateModule. Функция проверяет наличие ключа, определённого системой и возможность запуска модуля заданного типа с помощью функции CheckResourceAccess.
11. OnAddAlgorithm вызывается при добавлении нового алгоритма модулю или при выборе уже добавленных алгоритмов. Создает объект класса SNGraph и инициализирует его.
12. OnDelAlgorithm вызывается при удалении алгоритма из модуля, делая его неактивным. Удаляет из внутреннего списка указанный алгоритм, деинициализирует его и посылает команду уничтожения посредством функции PostCommand, которая вызывает у алгоритма функцию CommandObjectDestroy.
13. HandleDestroy переопределённая функция. Вызывает функцию базового класса, так же посылает команду снятия с регистрации с помощью функции PostCommand объекту SNSystem, что вызовет у него функцию RegisterObject, с параметрами удаления зарегистрированного объекта.
14. CheckEventRight переопределённая функция. Фильтрует системные события, которые всегда проходят проверку, и вызывает функцию CheckRightType, для проверки остальных событий.
15. CheckCommandRight переопределенная функция. Фильтрует системные команды, которые всегда проходят проверку, и вызывает функцию CheckRightType, для проверки остальных команд.
16. CheckRightType функция тогда, когда требуется проверить права пользователя. Производит поиск по спискам пользователей, типов прав и возвращает true, если разрешено и false, если запрещено.
17. SNObject базовый класс для объектов, которые будут поддерживать механизм соединения по сообщениям и механизм посылки команд.
18. Connect вызывается в случае необходимости соединения двух объектов по определённому типу сообщения. У получающего сообщение объекта берется указатель на класс SNCommandDescriptor с помощью функции GetCommandDescriptor и посылается сообщение на соединение объекту источнику с помощью функции PostCommand, что вызовет у источника функцию CommandCommonConnect.
19. CommandCommonConnect. Вызов функции инициируется, как правило, посылкой команды на соединение с помощью функции PostCommand. Функция проверяет наличие соединения данного объекта с указанным типом сообщения. Если проверка пройдена, то проверяется право пользователя на подсоединение к данному объекту функцией CheckEventRight. Если проверка на права прошла успешно, то проверяется тип сообщения. Если указанный тип не поддерживается объектом, то происходит поиск конвертера сообщений функцией GetSourseConverter. Если конвертер найден, то возвращается объект конвертера, и далее коннект получателя происходит с конвертером. Если указанный тип поддерживается объектом, то объект запоминается в списках по данному типу сообщения. Далее вызывается функция HandleConnected.
20. HandleConnected виртуальная функция. Вызывается из функции CommandCommonConnect. Функция предназначена для дополнительной обработки соединения некоторых типов событий. Если с объектом произведено соединение на событие изменения состояния, то получателю отправляется текущее состояние.
21. PostCommand. Дружественная функция классу SNObject, не являясь его членом. Вызывается тогда, когда требуется послать любое сообщение объекту. Она проверяет у получателя права пользователя на отправку команды с помощью функции CheckCommandRight. Если проверка прошла успешно, то получает у объекта указатель на класс SNCommandDescriptor с помощью функции GetCommandDescriptor и если указатель существует. то вызывается функция ProcessCommand.
22. PostQuery. Дружественная функция классу SNObject, не являясь его членом. Вызывается тогда, когда требуется послать любой запрос объекту.
23. SetSecurityContext. Функция вызывается тогда когда требуется установить контекст пользователя, объект класса SNSecurityContext, для объекта. Устанавливает новое значение контекста пользователя.
24. CommandObjectDestroy виртуальная функция. Вызов функции инициируется посылкой команды уничтожения функцией PostCommand. Вызывает функцию HandleDestroy.
25. HandleDestroy виртуальная функция. Вызывается для удаления объекта. Функция помечает объект как уничтожающийся функцией SetState. Посылается событие, что объект уничтожается функцией GenerateEvent. Так же вызывается HandleDestroy у всех дочерних объектов. Очищаются списки дочерних объектов. Происходит удаление объекта из списков родителя.
26. CheckEventRight виртуальная функция. Вызывается тогда, когда требуется проверить право пользователя на подключение к указанному типу события. Например, в функции CommandCommonConnect. Возвращает true, если проверка прошла успешно. Без переопределения всегда возвращает true.
27. CheckCommandRight виртуальная функция. Вызывается тогда, когда требуется проверить право пользователя выполнить команду указанного типа. Например, в функции PostCommand. Возвращает true, если проверка прошла успешно. Без переопределения всегда возвращает true.
28. GetSourseConverter. Вызов функции происходит в CommandCommonConnect. Функция производит поиск подходящего конвертера по списку, сравнивая входящий тип сообщения конвертора с тем, который может генерировать источник. Если подходящий конвертер найден, то возвращает объект конвертера SNDataConverter.
29. SetState. Функция вызывается тогда, когда требуется поменять состояние объекта. Функция меняет состояние и сообщает об этом все присоединенным на это событие объектам с помощью функции GenerateEvent, вызывая у получателей функцию CommandObjectState.
30. GenerateEvent вызывается тогда, когда объекту нужно сгенерировать команду для всех объектов, подключенных к данному объекту по указанному типу события. Функция находит список объектов, которые зарегистрированы за этим типом события и вызывает функцию ProcessCommand.
31. CheckResourceAccess. Функция вызывается при необходимости определить наличие возможности, заданной сертификатом, запуска данного модуля. Для проверки в сертификате вызывается функция SNProtection::PossibleAccess и в зависимости от результата вносится событие в протокол событий.
32. ReleaseResource. Функция вызывается при необходимости освободить место занимаемое модулем в сертификате при его удалении, вызывая функцию SNProtection::ReleaseAccess и в зависимости от результата вносится событие в протокол событий.
33. ResetStateBit. Функция вызывается при необходимости сброса флага состояния. Вызывает функцию SetState с новым состоянием.
34. SNDataObject класс объект данных, предназначен для работы с данными конфигурации.
35. SNLog класс используется для записи в протокол событий и в отладочный журнал.
36. Initialize. Вызов функции происходит в конструкторе объекта SNSystem, после считывания параметров запуска. Функция проверяет параметр SystemParamNames::LogFileName, и если он установлен, то все дальнейшие сообщения, записываемые через функцию LogString, будут записываться в файл указанный в параметре.
37. LogString. Вызов функции происходит тогда, когда требуется вывести какую-либо строку в файл или стандартный поток выводы текстовых сообщений операционной системы.
38. SNLogInfo. Вызов происходит при необходимости добавить в протокол событий информативное сообщение.
39. SNLogError Вызов происходит при необходимости добавить в протокол событий сообщение об ошибке.
40. SNProtection класс, предназначенный для работы с ключом. Он производит проверку наличия подключенного ключа к системе и соответствие его сертификату. Составляет список доступных для запуска модулей и их количество.
41. GetState вызывается из SNSystem::CommandModuleStart. Функция возвращает состояние, по которому можно определить в порядке ли ключ и сертификат.
42. ReleaseAccess функция вызывается из функции SNObject::ReleaseResource. Она освобождает занимаемую модулем ячейку сертификата.
43. SNBufferPool класс менеджер объектов класса SNBuffer.
44. Allocate. Функция вызывается при необходимости получения массива байт. Функция берет массив из списка свободных массивов. Если список пуст, то создается новый массив. Функция возвращает указатель объект класса SNBuffer.
45. Release. Функция вызывается при окончании работы с массивом. Если количество свободных массивов превышает заданный предел, то освободившийся массив удаляется, если предела не достигнуто, то добавляется в список свободных массивов.
46. SNBuffer. Класс-обертка для массива байт.
47. SNMemoryDB наследник класса SNDataBase.
48. SNSQLite3Manager класс менеджер для работы с базой SQLite.
49. SNDataBase. Класс, реализующий базу данных в памяти.
50. SelectByForeignKey. Функция вызывается при необходимости получения списка данных удовлетворяющих заданному условию.
51. AddObject. Функция вызывается для добавления объекта в базу. Функция добавления объекта в базу и генерирует событие, добавления объекта в базу функцией GenerateEvent.
52. AddObjectWithRight. Функция вызывается для добавления объекта в базу с правами пользователя, наследуемыми от родительского объекта.
53. SNDynamicObjectProxy. Класс является промежуточным объектом, который по идентификатору ищет в системе нужный объект, передает ему все соединения и сигнализирует о том, что искомый объект пропал или появился в системе.
54. Initialize. Функция вызывается при необходимости инициализации или переинициализации объекта. Функция сбрасывает состояние активности объекта с помощью функции ResetStateBit и происходит отправка запроса объекта, с помощью функции PostQuery, объекту провайдеру (обычно это объект SNSystem).
55. SetServerObject. Функция вызывается в момент поступления ответа на запрос объекта или при необходимости установить или обнулить контролируемый объект. Функция производит перенаправление всех своих соединений устанавливаемому объекту и отсоединяет старый объект от всех сообщений. Устанавливается состояние равное состоянию объекта функцией SetState.
56. CommandObjectState. Вызов функции инициируется посылкой события о смене состояния объектом. Если событие пришло от объекта слежения, то устанавливается новое состояние функцией SetState. Если событие пришло от объекта и поменялось состояние активности, то происходит переинициализация функцией Initialize.
57. SNGraph. Базовый класс для всех алгоритмов.
58. SetExternalParameters. Функция вызывается, когда требуется установить алгоритму конфигурацию. В ней производится подключение к изменению параметров алгоритма. И заполняется список с параметрами.
59. CommandGraphStart. Вызывается при запуске алгоритма. При вызове функции происходит создание и инициализация дочерних объектов, генерируется событие о старте алгоритма.
60. SNCommandDescriptor. Класс предназначен для описания механизма обработки команды.
61. ProcessCommand. Функция вызывается при посылке команды и проверяет поток, в котором работает получатель команды. Если поток отличный от текущего, то команда помещается в очередь получателя на выполнение. Если получатель работает в одном потоке с отправителем, то функция, обрабатывающая эту команду, вызывается в синхронном режиме.
62. SNDataConverter. Базовый класс конверторов данных.
63. HandleConnected. Вызывается при подключении какого-либо объекта к объекту конвертора. Если тип подключаемого события равен типу, который генерирует конвертор, то конвертор соединяется с источником на тип, который он умеет принимать.
64. SNSocket. Класс предназначен для взаимодействия с устройствами по сети.
65. SNVideoChannelWindow. Класс окна, отображающего видео поток.
66. SNGPSModule – модуль, обеспечивающий синхронизацию системного времени посредством спутниковых систем. Также модуль обеспечивает определение географических координат установки комплексов для включения данных сведений в квитанции о нарушениях.
67. SNAutoPatrol2Violation. Базовый класс контроля нарушений. Обеспечивает проверку ГРЗ предполагаемых нарушителей на содержание в базе спецтранспорта для исключения генерации квитанций для данной категории транспорта. Обеспечивает обновление списка по заданному расписанию из локальной папки, а также с FTP сервера.
68. SNAutoNumberRecognizerChannel - класс, обеспечивающий распознавание ГРЗ транспортных средств, а также определение направления их движения.
69. SNAutopatrol2UpdateSettings - визуальная форма, обеспечивающая задание расписания и источник обновления списка специальных ТС.
70. SNDisallowedParking – класс, обеспечивающий позиционирование поворотного устройства с целью получения обзорных снимков контролируемых зон, а также детализированных снимков с целью определения наличия ТС в зоне контроля и распознавания их регистрационных знаков. Модуль обеспечивает накопление снимков в течение требуемого времени и сопоставление номеров ТС для выявления факта нарушения.
71. SNAutoPatrol2DisallowedParkingSettings – визуальная форма, содержащая закладки с настройками комплекса.
72. SNDisallowedParkingBasicSettings – визуальная форма, позволяющая задавать основные параметры комплекса (поворотное устройство, разрешенное время стоянки и т. п.)
73. SNDisallowedParkingZoneSettings – визуальная форма, позволяющая задавать список и границы контролируемых зон, а также интервалы обхода зон поворотным устройством.
74. SNDisallowedParkingSchedule — визуальная форма, позволяющая задать дневные и недельные интервалы функционирования комплекса.
75. SNArenaTicketSender – класс, обеспечивающий формирование электронной формы квитанции нарушения в формате TAR 1.3.

## **3.4. Связи программы с другими программами**

Связь специального программного обеспечения «PAK\_OSA» с другим программным обеспечением отсутствует.

# 4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

В состав используемых технических средств входит:

1. Сервер фотофиксации, включающий в себя:
* процессорный модуль с тактовой частотой 2.5 ГГц или выше;
* оперативная память объемом 2 Гб или выше;
* жесткий диск объемом 128 Гб и выше;
* сетевым интерфейсом Ethernet 10/100/100 Base-TX;
* Wi-Fi роутер.
1. Поворотная видеокамера.

# 5. ВЫЗОВ И ЗАГРУЗКА

 Загрузка и запуск специального программного обеспечения «PAK\_OSA» осуществляется автоматически встроенными средствами операционной системы.

# 6. ВХОДНЫЕ ДАННЫЕ

Входными данными специального программного обеспечения «PAK\_OSA» являются:

1. команды, вводимые оператором посредством графического интерфейса;
2. Ethernet-кадры, полученные специальным программным обеспечением с сетевых интерфейсов программно-аппаратного комплекса;
3. состояние аппаратных модулей программно-аппаратного комплекса.

Команды, вводимые оператором посредством графического интерфейса, интуитивно понятны, соответствуют по смыслу действиям, выполняемым командой.

Организация Ethernet-кадров, полученных специальным программным обеспечением с сетевых интерфейсов программно-аппаратного комплекса, определяется стандартами на Ethernet-кадры соответствующих протоколов. В программе Ethernet-кадры представлены в виде сырых двоичных данных в буферах в оперативной памяти. Предварительную подготовку Ethernet-кадров из сигналов физического уровня выполняет техническое средство на аппаратном уровне.

Организация и кодирование данных состояния аппаратных модулей программно-аппаратного комплекса определяется их характером (регистр, таблица, одиночное значение, структура) и способом их получения (посредством чтения памяти, вызова функций библиотеки SDK, обмена данными по аппаратной шине). Предварительная подготовка состояния аппаратных модулей программно-аппаратного комплекса выполняется самими модулями на аппаратном уровне, дальнейшие преобразования производятся в коде программы.

# 7. ВЫХОДНЫЕ ДАННЫЕ

Выходными данными специального программного обеспечения «PAK\_OSA» являются:

1. файлы протокола событий программы;
2. сообщения, передаваемые оператору посредством графического интерфейса;
3. Ethernet-кадры, отправляемые программой с сетевых интерфейсов технических средства;
4. файлы данных для обработки в специальном программном обеспечении ЦАФАП ГИБДД.

Характер, организация, формат, описание и способ отображения данных файла протокола событий определяется пользователем на этапе его формирования.

Сообщения, передаваемые оператору посредством графического интерфейса, представляют собой графические окна, содержавшие символы на русском/английском языке в кодировке ASCII.

Ethernet-кадры, отправляемые программой с сетевых интерфейсов технического средства, представлены в виде двоичных данных, соответствующих сетевым протоколам.

Файлы для обработки в специальном программном обеспечении ЦАФАП ГИБДД представляют собой данные, в формате JPEG и TAG – ASCII файл, который содержит список полей, определенных в таблице базы данных.

# Перечень терминов и сокращений

**ГРЗ –** государственный регистрационный знак.

**ПДД** – Правила дорожного движения.

**СПО –** специальное программное обеспечение.

**ТС** – транспортное средство.

**ЦАФАП ГИБДД -** Центра автоматизированной фиксации административных правонарушений ГИБДД.

|  |
| --- |
| ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ |
| Номера листов (страниц) | Всеголистов(страниц)в документе | №документа | Входящий номерсопроводительногодокумента и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. 1) ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. 2) ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. 3) ГОСТ 19.104-78 ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. 4) ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. 5) ГОСТ 19.106-78 ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. 6) ГОСТ 19.402-78 ЕСПД. Описание программы [↑](#footnote-ref-6)
7. 7) ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)