***Инструкция по программированию***

**1.Описание протокола.**

***1.1 Формат протокола передачи команд***

Связь контрольно-кассовой техники (далее ККТ) с компьютером осуществляется по интерфейсу RS232 или USB в режиме виртуального COM порта. Параметры COM порта по умолчанию - 57600, no parity, 8 data bits, 1 stop bit.

Для по подключения по COM-порту используйте функцию openPort:

openPort(“COM1”, 57600);

COM1 – номер порта;

57600 – скорость работы COM-порта.

Любое взаимодействие между устройствами инициируется со стороны компьютера: он посылает командное сообщение (команду), а ККТ всегда посылает ответное сообщение (ответ). Взаимодействие компьютера с ККТ может осуществляться в двух режимах: синхронном и асинхронном. В синхронном режиме компьютер посылает команду и ждет ответ на него. После получения ответа компьютер посылает следующую команду. Взаимодействие в синхронном режиме может осуществляться без использования линий аппаратного квитирования (DTR/DSR, CTS/RTS). В асинхронном режиме компьютер посылает командные сообщения и принимает ответные сообщения в произвольной последовательности. Взаимодействие в этом режиме должно осуществляться с использованием линий аппаратного квитирования CTS/RTS.

Пакет команды со стороны компьютера

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STX** | **Пароль связи** | **ID пакета** | **Код команды** | **Данные** | **ETX** | **CRC** |

Пакет ответа со стороны ККТ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STX** | **ID пакета** | **Код команды** | **Код ошибки** | **Данные** | **ETX** | **CRC** |

**STX** - байт начала пакета (0**x**02).

**ETX** -байт окончания пакета (0**x**03).

**CRC** - контрольная сумма пакета. Контрольная сумма подсчитывается по алгоритму: выполнение операции XOR для всех байт блока, включая ETX, но исключая STX. Данные контрольной суммы занимают два байта и являются символьным представлением числа в шестнадцатеричной системе исчисления.

**ID пакета**- идентификатор пакета. Произвольный байт, имеющий код в промежутке между 0**x**20 и 0**x**F0. ID пакета в ответе на команду, всегда совпадает с ID пакета команды. Может использоваться для синхронизации пакета команды и ответа на нее в условиях многозадачности в приложении или в асинхронном режиме.

**Пароль связи** - четырехбайтовый пароль, предназначенный для ограничения возможности несанкционированной работы с ККТ. Пароль по умолчанию - ”PIRI”.

**Код команды** - два байта, представляющие собой код команды в шестнадцатеричном исчислении, т.е. если код команды равен 0x21, необходимо передать два символа в виде – «21».

**Код ошибки** - два байта, с символьным представлением числа в шестнадцатеричном исчислении. Поле содержит число «00» в случае успешного выполнения команды или код ошибки.

**Данные** - параметры команды, или ответа на команду, разделенные между собой символом FS (0x1C ). Количество передаваемых и возвращаемых параметров зависит от кода конкретной команды.

***1.2 Специальные команды.***

Специальные команды протокола состоят из одного байта в двоичном формате, без заголовка, концовки и контрольной суммы.

**1.2.1 Прервать выполнение отчета**

Все отчеты, кроме X или Z, могут быть прерваны с помощью посылки на ККТ специальной однобайтовой команды:

0**x**18 (CAN) – прервать выполнение отчета.

**1.2.2 Проверка связи с ККТ**

Для проверки связи с ККТ существует специальная команда:

0**x**05 (ENQ) – проверка связи.

Ответ состоит тоже из одного байта:

0**x**06 (ACK) – ККТ на связи.

Если в момент проверки связи ККТ передает данные в ответ на другую команду, то ответ может быть получен только после завершения этой передачи.

**1.2.3 Промотка бумаги**

Для промотки бумаги в ККТ существует специальная команда:

промотать бумагу на одну строку

**Функция:** *int scrollPaper ();*

**Входные параметры: нет**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***1.3 Времена ожидания***

В протоколе обмена данными между ККТ и компьютером предусмотрены два времени ожидания:

* Время ожидания приема между байтами пакета информации – 200 мс. По истечении данного промежутка времени возвращается код ошибки передачи.
* Время ожидания ответа от ККТ - зависит от выполняемой ККТ в данный момент операции. Например, при печати длинных отчетов из ККТ может быть занята десятки минут. Для принятия решения о неисправности ККТ (коммуникационного порта или кабеля передачи данных) необходимо использовать команду “Проверка связи с ККТ”.

**2.Основные типы передаваемых параметров**

***2.1 Строка*** ***–*** любая последовательность символов, с кодами от 0x20 до 0xF0.Длина конкретной строки зависит от значения передаваемого параметра. Может иметь нулевую длину (пустая строка). Для печати символов на русском языке необходимо использовать кодировку CP866.

***2.2 Дата –*** строка длиной 6 символов вида «**ДДММГГ**», где***:***

ДД - день месяца;

ММ - номер месяца в году;

ГГ - последние две цифры года.

Все числа передаются с точностью до 2-х цифр, если число меньше 10-ти, то добавляется старший ноль.

struct MPiritDate {

int year; // Год

unsigned char month; // Месяц

unsigned char day; // День

};

***2.3 Время***- строка длиной 6 символов вида «**ЧЧММСС**»***,*** где:

ЧЧ - часы;

ММ - минуты;

СС - секунды.

Все числа передаются с точностью до 2-х цифр, если число меньше 10-ти, то добавляется старший ноль.

struct MPiritTime {

unsigned char hour; // Часы

unsigned char minute; // Минуты

unsigned char second; // Секунды

};

***2.4 Номер команды*** – однобайтовое число, соответствующее номеру команды.

В перечисление добавлены номера команд, с которыми рекомендуется использовать функции из раздела 3.1, но при необходимости в качестве номера команды можно передать байт с соответствующим значением.

enum {

PIRIT\_COMMAND\_\_SET\_EXTRA\_REQUISITES = 0x24,

PIRIT\_COMMAND\_\_OPEN\_DOC = 0x30,

PIRIT\_COMMAND\_\_CLOSE\_DOC = 0x31,

PIRIT\_COMMAND\_\_ADD\_POSITION = 0x42,

} piritCommand\_t;

***2.5 Целое число*** - строка состоящая только из цифр, и представляющей собой целое число в десятеричной системе исчисления. Пустая строка интерпретируется как ноль. Используется для передачи номеров, индексов, битовых масок(полей) и т.д.

***2.6 Дробное число*** -строка состоящая из цифр, десятичной точки и знака «-». Пустая строка интерпретируется как ноль. Используется для передачи суммы, количества и процентных ставок. Рекомендуется передавать суммы с точностью 2 знака после десятичной точки (с точностью до копейки), процентные ставки с точностью до 4-х знаков после десятичной точки и количество до 9-ти знаков после десятичной точки.

***2.7 Имя оператора*** - строка размером 0..23 символа. Если первые два символа являются цифрами, то они воспринимаются как код оператора. По умолчанию код равен 01.

***2.8 Структура ответа MData***

struct MData {

int errCode; // Код ошибки

char data[256]; // Строка с ответом(массив байт)

int dataLength; // Длина строки с ответом

};

**3.Описание команд ККТ**

***3.1 Общий список команд***

|  |  |
| --- | --- |
| **Код команды**  **(HEX)** | **Название команды** |
| *Команды получения данных о ККТ* | |
| **00** | Запрос флагов статуса ККТ |
| **01** | Запрос сменных счетчиков и регистров |
| **02** | Запрос сведений о ККТ |
| **03** | Запрос данных по чеку |
| **04** | Запрос состояния печатающего устройства |
| **05** | Запрос сервисной информации |
| **06** | Запрос дополнительной информации о ошибках |
| *Команды настройки параметров ККТ* | |
| **10** | Начало работы с ККТ |
| **11** | Чтение таблицы настроек |
| **12** | Запись таблицы настроек |
| **13** | Чтение даты/времени ККТ |
| **14** | Запись даты/времени ККТ |
| **15** | Программировать логотип |
| **16** | Удалить логотип |
| **17** | Загрузить дизайн чека |
| **18** | Загрузить графическое изображение |
| **19** | Поиск графического изображения |
| *Команды основных операций ККТ* | |
| **20** | Распечатать отчет без гашения (X-отчет) |
| **21** | Сформировать отчет о закрытии смены |
| **23** | Открыть смену |
| **24** | Установить дополнительные реквизиты позиции |
| **30** | Открыть документ |
| **31** | Завершить документ |
| **32** | Аннулировать документ |
| **33** | Отложить документ |
| **34** | Отрезать документ |
| **40** | Печать текста |
| **41** | Печатать штрих-код |
| **42** | Добавить товарную позицию |
| **43** | Сторнировать товарную позицию |
| **44** | Подытог |
| **45** | Скидка на товар/чек |
| **46** | Наценка на товар/чек |
| **47** | Оплата |
| **48** | Внесение / изъятие суммы |
| **49** | Печать реквизита |
| **50** | Зарегистрировать сумму по отделу |
| **51** | Зарегистрировать сумму налога |
| **52** | Сравнить сумму по чеку |
| **53** | Открыть копию чека |
| **54** | Обнулить наличные в денежном ящике |
| **55** | Печать графики в документе |
| **56** | Печать загруженной картинки |
| **57** | Печать реквизита для ОФД |
| **58** | Печать чека коррекции |
| **59** | Печать отчета о текущем состоянии расчетов |
| *Команды регистрации* | |
| **60** | Регистрация / перерегистрация |
| **63** | Технологическое обнуление |
| *Команды для работы с ККТ* | |
| **70** | Активизация ККТ |
| **71** | Закрытие архива ККТ |
| **72** | Распечатать контрольную ленту из ККТ |
| **73** | Распечатать документ из ККТ |
| **74** | Распечатать отчет по сменам из ККТ |
| **75** | Распечатать отчет по датам из ККТ |
| **76** | Распечатать отчет по активизации ККТ |
| **77** | Распечатать отчет по смене из ККТ |
| **78** | Запрос информации из ККТ |
| *Команды для работы с ФН* | |
| **71** | Закрытие ФН |
| **73** | Распечатать документ из ФН |
| **76** | Распечатать отчет о регистрации ККТ из архива ФН |
| **78** | Запрос информации из ФН |
| *Команды управления внешними устройствами* | |
| **80** | Открыть денежный ящик |
| **81** | Получить статус денежного ящика |
| **82** | Подача звукового сигнала |
| *Системные команды* | |
| **92** | Чтение ПО ККТ |
| **93** | Установить скорость обмена |
| **94** | Распечатать сервисные данные |
| **A0** | Аварийное закрытие смены |
| **A1** | Печать копии отчета о закрытии смены |

3

**3.1.1 Отправка команды с чтением ответа.**

**Функция:** *MData libSendCommand(piritCommand\_t cmd, int num, ...);*

**Входные параметры: (Номер команды)** номер команды, **(*Целое число)*** Количество параметров типа **Строка** (может быть 0), [**(Строка)** параметр 1, …]

Функция принимает переменное число параметров. Пример использования:

MData response = libSendCommand(PIRIT\_COMMAND\_\_CLOSE\_DOC, 3, “0”, “+79555555555”, “0”);

Каждый из переданных параметров после параметра num должен иметь тип const char\* либо приводиться к нему. Значение параметра num должно соответствовать количеству переданных параметров типа const char\*. Кодировка строк – OEM 866.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР.

**3.1.2 Отправка команды без чтения ответа.**

**Функция:** *int libSendCommandAsync(piritCommand\_t cmd, int num, ...);*

**Входные параметры: (Номер команды)** номер команды, **(*Целое число)*** Количество параметров типа **Строка** (может быть 0), [**(Строка)** параметр 1, …]

Функция работает аналогично *libSendCommand*, но чтение ответа не производится.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.1.2 Инструкции по программированию устройств.**

Ссылки на актуальные инструкции по программированию: [FM HYPERLINK "https://htmlpreview.github.io/?https://github.com/dreamkas/pirit\_documentation/blob/master/output/documentation\_fm16\_1\_05.html"1 HYPERLINK "https://htmlpreview.github.io/?https://github.com/dreamkas/pirit\_documentation/blob/master/output/documentation\_fm16\_1\_05.html"6](https://htmlpreview.github.io/?https://github.com/dreamkas/pirit_documentation/blob/master/output/documentation_fm16_1_05.html) [Пирит 2Ф](https://htmlpreview.github.io/?https://github.com/dreamkas/pirit_documentation/blob/master/output/documentation_pirit2f_1_05.html).

***3.2 Команды получения данных о ККТ***

**3.2.0 Открытие порта**

**Функция:** *int openPort(char \*fileName, long speed);*

**Входные параметры: (Строка)** Имя порта, например “COM1” или “dev/ttyS1”, **(*Целое число)*** Скорость, например 57600.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.2.1 Запрос флагов статуса ККТ (0x00)**

**Функция:** *MData libGetStatusFlags();*

**Входные параметры: нет**

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит **(*Целое число)*** Статус фатального состояния ККТ, **(*Целое число)*** Статус текущих флагов ККТ, **(*Целое число)*** Статус документа

**Функция:** *int getStatusFlags (int \*fatalStatus, int \*currentFlagsStatus, int \*documentStatus);*

**Входные параметры: нет**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

*fatalStatus* **(*Указатель на целое число)*** Статус фатального состояния ККТ, *currentFlagsStatus* **(*Указатель на целое число)*** Статус текущих флагов ККТ, *documentStatus* **(*Указатель на целое число)*** Статус документа

**Статус фатального состояния ККТ**

|  |  |
| --- | --- |
| **Номер бита** | **Пояснения** |
| 0 | Неверная контрольная сумма NVR |
| 1 | Неверная контрольная сумма в конфигурации |
| 2 | Зарезервировано |
| 3 | Зарезервировано |
| 4 | Зарезервировано |
| 5 | ККТ не авторизовано |
| 6 | Фатальная ошибка ФН |
| 7 | Зарезервировано |

При установке любого бита статуса фатального состояния блокируются все функции ККТ, за исключением битов 6 и 7, при установке которых доступны команды закрытия смены и закрытия архива ККТ.

**Статус текущих флагов ККТ**

|  |  |
| --- | --- |
| **Номер бита** | **Пояснения** |
| 0 | Не была вызвана функция “Начало работы” |
| 1 | Нефискальный режим |
| 2 | Смена открыта |
| 3 | Смена больше 24 часов |
| 4 | Архив ККТ закрыт |
| 5 | ККТ не активирована |
| 6 | Нет памяти для закрытия смены в ФП |
| 7 | Был введен неверный пароль доступа к ФП |
| 8 | Не было завершено закрытие смены, необходимо повторить операцию |

**Статус документа**

|  |  |
| --- | --- |
| **Номера бит** | **Пояснения** |
| Тип текущего открытого документа | |
| 0,1,2,3 | 0 Документ закрыт  1 Сервисный документ  2 Чек на продажу (приход)  3 Чек на возврат (возврат прихода)  4 Внесение в кассу  5 Инкассация  6 Чек на покупку (расход)  7 Чек на возврат покупки (возврат расхода) |
| Состояние документа | |
| 4,5,6,7 | Для всех типов документов:  0 Документ закрыт  1 Устанавливается после команды «открыть документ». (Для типов документа 2, 3 - можно вводить товарные позиции и скидки/наценки на них.)  Для документов типа 2, 3:  2 Устанавливается после первой команды «Подытог». Можно делать скидки/наценки на чек.  3 Устанавливается после второй команды «Подытог» или после начала команды «Оплата». Можно только производить оплату различными типами платежных средств.  4 Расчет завершен – требуется закрыть документ.  8 Команда закрытия документа была дана в ФН, но документ не был завершен. Аннулирование документа невозможно. |

**3.2.2 Запрос сменных счетчиков и регистров (0x01)**

**Функция:** *MData libGetCountersAndRegisters (unsigned char numRequest);*

Эта команда позволяет получать различные значения сменных счетчиков и регистров ККТ.

**Входные параметры:** *numRequest* (***Целое число*** 1..14) Номер запроса.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (***Целое число***) Номер запроса, Возвращаемые данные.

Тип и количество возвращаемых данных зависит от значения номера запроса.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер**  **запроса**  **(DEC)** | **Наименование**  **Запроса** | **Формат возвращаемых данных** | **Комментарии** |
| **1** | Вернуть номер текущей смены | **Целое число** |  |
| **2** | Вернуть номер следующего чека | **Целое число** | Возвращается правильный номера чека только при автоматической нумерации чеков средствами самой ККТ. |
| **3** | Вернуть суммы продаж по типам платежа | **Дробное число\*16** | Возвращается 16 значений – по максимально возможному количеству типов платежей. |
| **5** | Вернуть суммы возвратов по типам платежа | **Дробное число\*16** |
| **7** | Вернуть количество оформленных чеков по типам операций | **Целое число\*6** | Возвращается 6 значений – количество чеков продажи, возврата, аннулированных, отложенных, чеков внесения и изъятия. |
| **8** | Вернуть суммы по оформленным чекам | **Дробное число\*4** | Возвращается 4 значения – суммы по аннулированным и отложенным чекам, чекам внесения и изъятия. |
| **9** | Вернуть суммы по скидкам/ наценкам | **Дробное число\*4** | Возвращается 4 значения – суммы скидок по продажам, наценок по продажам, скидок по возвратам, наценок по возвратам. |
| **10** | Вернуть суммы налогов по продажам | **Дробное число\*6** |  |
| **11** | Вернуть суммы налогов по возвратам | **Дробное число\*6** |  |
| **12** | Вернуть данные по последнему X отчету или отчету о закрытии смены | **Строка,**  **Целое число,**  **Дробное число,**  **Целое число,**  **Дробное число,**  **Целое число,**  **Дробное число,**  **Целое число,**  **Дробное число,**  **Целое число,**  **Дробное число,**  **Целое число,**  **Дробное число,** | Возвращается 13 значений –  текущий операционный счетчик,  номер документа,  сумма в кассе,  кол-во продаж (кол-во приходов),  сумма продаж (сумма приходов),  кол-во возвратов (возвратов прихода),  сумма возвратов (возвратов прихода),  количество аннулированных,  сумма аннулированных,  количество внесений,  сумма внесений,  количество изъятий,  сумма изъятий. |
| **13** | Вернуть суммы по секциям/отделам по продажам | **Дробное число\*16** | Возвращается 16 значений – по максимально возможному количеству секций/отделов |
| **14** | Вернуть суммы по секциям/отделам по возвратам | **Дробное число\*16** | Возвращается 16 значений – по максимально возможному количеству секций/отделов |
| **15** | Вернуть количество оформленных чеков по типам операций | **Целое число,**  **Целое число** | кол-во чеков покупок (расходов),  кол-во чеков возвратов покупок (возвратов расхода), |
| **16** | Вернуть суммы покупок по типам платежа | **Дробное число\*16** | Возвращается 16 значений – по максимально возможному количеству типов платежей. |
| **17** | Вернуть суммы возвратов покупок по типам платежа | **Дробное число\*16** | Возвращается 16 значений – по максимально возможному количеству типов платежей. |
| **18** | Вернуть данные по коррекциям | **Целое число,**  **Дробное число\*2** | Количество коррекций  Суммы коррекций наличными и безналичными |

**3.2.3 Запрос сведений о ККТ (0x02)**

**Функция:** *MData libGetKKTInfo(unsigned char numRequest);*

Эта команда позволяет получать разнообразную информацию о ККТ.

**Входные параметры:** *numRequest* (***Целое число*** 1..16)номер запроса.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (***Целое число***) Номер запроса, Возвращаемые данные.

Тип и количество возвращаемых данных зависит от значения номера запроса.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер**  **запроса**  **(DEC)** | **Наименование**  **запроса** | **Формат возвращаемых данных** | **Комментарии** |
| **1** | Вернуть заводской номер ККТ | **Строка** |  |
| **2** | Вернуть идентификатор прошивки | **Целое число**  **Целое число**  **Целое число** | Начиная с версии 665/565  Начиная с версии 665/565 |
| **3** | Вернуть ИНН | **Строка** |  |
| **4** | Вернуть регистрационный номер ККТ | **Строка** |  |
| **5** | Вернуть дату и время последней фискальной операции | **Дата, Время** |  |
| **6** | Вернуть дату регистрации / перерегистрации | **Дата** |  |
| **7** | Вернуть сумму наличных в денежном ящике | **Дробное число** |  |
| **8** | Вернуть номер следующего документа | **Целое число** |  |
| **9** | Вернуть номер смены регистрации | **Целое число** |  |
| **10** | Вернуть номер следующего X отчета | **Целое число** |  |
| **11** | Вернуть текущий операционный счетчик | **Строка** |  |
| **12** | Вернуть нарастающий итог | **Дробное число,**  **Дробное число,**  **Дробное число,**  **Дробное число** | Продажа (приход),  Возврат (возврат прихода),  Покупка (расход),  Возврат покупки (возврат расхода) |
| **14** | Вернуть дату окончания временного ресурса ФН | **Дата** |  |
| **15** | Вернуть тип прошивки | **Целое число** | 0- стандартная прошивка  1- отладочный комплект |
| **16** | Вернуть размер бумаги текущего дизайна | **Целое число** | 0- 80мм  1- 57мм |
| **17** | Вернуть дату и время открытия смены | **Дата, Время** |  |
| **21** | Вернуть модель устройства | ***Целое число*** |  |
| **23** | Вернуть режим работы и систему налогообложения ФН | **Целое число**  **Целое число** | Система налогообложения  Режим работы |

**3.2.4 Запрос данных по чеку (0x03)**

**Функция:** *MData libGetReceiptData(unsigned char numRequest);*

Эта команда позволяет получать данные по чеку. Тип и количество возвращаемых данных зависит от значения передаваемого параметра – номер запроса.

**Входные параметры:** *numRequest* (***Целое число*** 1..2) Номер запроса.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (***Целое число***) Номер запроса, Возвращаемые данные.

Тип и количество возвращаемых данных зависит от значения номера запроса.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер**  **запроса**  **(DEC)** | **Наименование**  **запроса** | **Формат возвращаемых данных** | **Комментарии** |
| **1** | Вернуть счетчики текущего документа | **Дробное число\*3** | Возвращается 3 значения - сумма чека, 0,0 |
| **2** | Вернуть данные по последнему закрытому чеку | **Целое число,**  **Строка,**  **Целое число,**  **Целое число,**  **Дробное число,**  **Дробное число,**  **Дробное число,**  **Строка** | Возвращается 8 значений –  тип чека (для аннулиров. = 0),  текущий операц. счетчик,  номер чека,  номер документа,  сумма чека,  0,  0,  строка ФП (фиск. признак) |

**3.2.5 Запрос состояния печатающего устройства (ПУ) (0x04)**

**Функция:** *MData libGetPrinterStatus();*

Эта команда позволяет получить текущее состояние печатающего устройства.

**Входные параметры: нет**

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: **(*Целое число)*** Статус ПУ.

**Статус ПУ**.

|  |  |
| --- | --- |
| **Номер бита** | **Пояснения** |
| 0 | Принтер не готов |
| 1 | В принтере нет бумаги |
| 2 | Открыта крышка принтера |
| 3 | Ошибка резчика принтера |
| 7 | Нет связи с принтером |

**3.2.6 Запрос сервисной информации (0x05)**

**Функция:** *MData libGetServiceInfo(unsigned char numRequest);*

Эта команда позволяет получать разнообразную сервисную информацию о ККТ.

**Входные параметры:** *numRequest* (***Целое число*** 1..9)номер запроса.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (***Целое число***) Номер запроса, Возвращаемые данные.

Тип и количество возвращаемых данных зависит от значения номера запроса.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер**  **запроса**  **(DEC)** | **Наименование**  **Запроса** | **Формат возвращаемых данных** | **Комментарии** |
| **1** | Вернуть напряжение питания (мВ) | **Целое число** | Возвращается значение в милливольтах |
| **2** | Вернуть температуру термоголовки | **Целое число** | Возвращается значение в градусах |
| **3** | Вернуть количество отрезов резчика | **Целое число** |  |
| **4** | Вернуть ресурс термоголовки | **Целое число** | Возвращается значение в мм |
| **5** | Обнулить количество отрезов резчика | **-** |  |
| **6** | Обнулить ресурс термоголовки | **-** |  |
| **7** | Вернуть напряжение на батарейке (мВ) | **Целое число** | Возвращается значение в милливольтах |
| **8** | Вернуть количество отрезов резчика (необнуляемое) |  |  |
| **9** | Вернуть ресурс термоголовки (необнуляемый) | **Целое число** | Возвращается значение в мм |

**3.2.7 Запрос дополнительной информации о ошибках (0x06)**

**Функция:** *MData libGetExErrorInfo(unsigned char numRequest);*

Эта команда позволяет получать данные по чеку. Тип и количество возвращаемых данных зависит от значения передаваемого параметра – номер запроса.

**Входные параметры:** (***Целое число*** 1..2) Номер запроса.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (***Целое число***) Номер запроса, Возвращаемые данные.

Тип и количество возвращаемых данных зависит от значения номера запроса.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер**  **запроса**  **(DEC)** | **Наименование**  **запроса** | **Формат возвращаемых данных** | **Комментарии** |
| **1** | Вернуть расширенный код ошибки (01h) | **Целое число** | Возвращается код, указывающий на причину возникновения ошибки (01h) “Функция невыполнима при данном статусе ККТ” |
| **2** | Вернуть статус блокировок по ФН | **Целое число** | Возвращается битовая маска, значения бит указаны в соответствующей таблице |

**Расширенный код ошибки (01h)**

|  |  |
| --- | --- |
| **Номер**  **кода** | **Пояснения** |
| 0 | Ошибок не было. |
| 1 | Не была вызвана функция “Начало работы” |
| 2 | Нефискальный режим |
| 3 | Архив ФН закрыт |
| 4 | ФН не зарегистрирован |
| 5 | ФН уже зарегистрирован |
| 7 | Нет изменений для перерегистрации ФН |
| 8 | Документ не был открыт |
| 9 | Предыдущий документ не закрыт |
| 10 | Сторнирование на копии документа |
| 11 | Состояние документа не равно 1 (см. **Состояние документа**) |
| 12 | Состояние документа не равно 1 или 2 (см. **Состояние документа**) |
| 13 | Состояние документа не равно 1 или 2 или 3 (см. **Состояние документа**) |
| 14 | Состояние документа не равно 4 (см. **Состояние документа**) |
| 15 | Документ закрыт в ФН |
| 16 | Документ не является продажей (приходом) или возвратом (возвратом прихода) |
| 17 | Документ не является внесением или изъятием |
| 18 | Документ не является сервисным |
| 19 | Документ является сервисным |
| 20 | Смена не открыта |
| 21 | Фатальная ошибка ФН |
| 22 | ФН не в режиме получения документа для ОФД |

**Статус блокировок по ФН**

|  |  |
| --- | --- |
| **Номер**  **бита** | **Пояснения** |
| 0 | Зарезервировано |
| 1 | ФН не найден |
| 2 | Не был закрыт архив ФН |
| 3 | Ошибка теста архива ФН |
| 4 | Ошибка связи с ФН |
| 5 | Не завершена операция закрытия смены |
| 6 | Зарезервировано |

***3.3 Команды настройки параметров ККТ***

***3.3.1* Начало работы с ККТ (0x10)**

Эта функция вызывается **всегда перед началом работы с ККТ**.

**Функция:** *int libCommandStart(MPiritDate mpDate, MPiritTime mpTime);*

**Входные параметры:** *mpDate* **(Дата)**Текущая дата, *mpTime* (**Время**)Текущее время

**Функция:** *int commandStart();*

**Входные параметры:** *нет*

Дата и время необходимы для проверки и согласования даты и времени работы компьютера и ККТ. Если разница между передаваемым в команде ”Начало работы” временем и текущим временем ККТ меньше 8 минут, то синхронизация ККТ происходит автоматически, если разница больше – возвращается один из кодов ошибки 0Bh или 0Сh.

Ошибка 0Bh возникает, в случае если передаваемые в команде ”Начало работы” дата и время отличаются от текущих даты и времени ККТ более чем на 8 минут. В случае возврата от ККТ ошибки 0Bh команда выполняется и ККТ может продолжить работу по своему внутреннему времени. При получении такого кода ошибки рекомендуется проверить время на управляющем компьютере, и, если оно верно, то выполнить следующие действия:

* закрыть смену (в случае если она открыта)
* установить время в ККТ соответствующей командой.

Ошибка 0Сh возникает, в случае если передаваемая в команде ”Начало работы” дата меньше даты последней фискальной операции зарегистрированной в ККТ. В случае возврата ошибки 0Сh команда инициализации не выполняется, т.е. блокируются все операции ККТ. При получении такого кода ошибки рекомендуется проверить дату и время на управляющем компьютере и сравнить его с датой и временем последней фискальной операции. В случае если установлена неправильная дата на управляющем компьютере, ее необходимо исправить и затем повторно выполнить команду ”Начало работы”.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.2* Чтение таблицы настроек (0x11)**

**Функция:** *MData libReadSettingsTable(unsigned char number, int index);*

Команда содержит два параметра для адресации элемента таблицы, значение которого необходимо получить. Возможные значения параметров указаны в п.4.

**Входные параметры:** *number* **(*Целое число)*** Номер, *index* **(*Целое число)*** Индекс(для массивов).

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: **(тип элемента таблицы)** Значение элемента таблицы.

***3.3.3* Запись таблицы настроек (0x12)**

**Функция:** *int libWriteSettingsTable(unsigned char number, int index, const char\* data);*

Команда содержит три параметра: два параметра для адресации элемента таблицы, и новое значение элемента таблицы. Возможные значения параметров указаны в п.4.

**Входные параметры:** *number* **(*Целое число)*** Номер, *index* **(*Целое число)*** Индекс(для массивов),*data* **(тип элемента таблицы)** Новое значение элемента таблицы.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.4* Чтение даты/времени ККТ (0x13)**

**Функция:** *MData libGetPiritDateTime();*

Эта команда позволяет получить время и дату ККТ.

**Входные параметры: нет**.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (**Дата**)Текущая дата, (**Время**)Текущее время.

***3.3.5* Запись даты/времени ККТ (0x14)**

**Функция:** *int libSetPiritDateTime(MPiritDate mpDate, MPiritTime mpTime);*

Эта команда позволяет устанавливать новые время и дату ККТ, при условии закрытой смены. Вводимая дата не может быть более ранней, чем дата последней фискальной операции. При попытке ввода такой даты, ККТ блокируется, и устанавливает флаг статуса “Не была вызвана функция Начало работы”. Для снятия блокировки, необходимо установить корректную дату и вызвать функцию “Начало работы”.

Если новые значения времени/даты больше текущих, более чем на 24 часа, в ответе на данную команду возвращается ошибка 0Ah. Для установки таких значений времени и даты, после возврата ошибки 0Ah , команда должна подаваться повторно.

**Входные параметры:** *mpDate* (**Дата**)Текущая дата, *mpTime* (**Время**)Текущее время.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.6* Программировать логотип (0x15)**

**Функция:** *int libLoadLogo(int size, unsigned char\* data);*

Функция используется для загрузки графического логотипа торгового предприятия для его печати перед каждым документом.

Образ логотипа представляет собой монохромное изображение в формате несжатого BMP. Ширина изображения – 576 точек, высота – от 8 до 220 точек. Для бумаги шириной 57мм зона печати ограничена 400 точками справа, 176 точек слева должны быть без изображения(см. рис. 1).

Рис 1. Расположение логотипа на бумаге 80мм и 57мм.

**Входные параметры:** *size* **(*Целое число)*** Количество байт, *data* **(*Массив байт)*** массив байт файла BMP

После загрузки логотипа возвращается стандартный ответ с кодом ошибки.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.7* Удалить логотип (0x16)**

**Функция:** *int libDeleteLogo();*

Функция используется для удаления ранее загруженного логотипа.

**Входные параметры: нет**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.8* Загрузить дизайн чека (0x17)**

**Функция:** *libLoadReceiptDesign(int size, unsigned char\* data)*

Функция используется для загрузки файла дизайнов чека.

**Входные параметры:** *size* **(*Целое число)*** Количество байт, *data* **(*Массив байт)*** массив байт файла дизайна

После загрузки возвращается стандартный ответ с кодом ошибки.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.9* Загрузить графическое изображение (0x18)**

**Функция:** *int libLoadPicture(int width, int height, int sm, const char\* name, int number, unsigned char\* data);*

Функция используется для загрузки графического изображения в память устройства.

Максимальная ширина изображения при использовании бумаги шириной 80мм – 512 точек(64 мм). Максимальный размер – 25088 байт( 448x448 точек).

Максимальная ширина изображения при использовании бумаги шириной 57мм – 336 точек точек(42 мм). 176 точек слева должны быть без изображения(как показано на рис. 1).

Графика печатается прижатой к левому краю, вне зоны печати фискального логотипа.

**Входные параметры:** *width* **(*Целое число)*** Ширина картинки, *height* **(*Целое число)*** Высота картинки, *sm* **(*Целое число)*** Смещение по умолчанию,*name* **(*Строка)***[1..15] – идентификатор картинки, *number* **(*Целое число)*** Номер, *data* **(*Массив байт)*** массив байт массив байт аналогичный монохромному изображению BMP файла.

Если параметр смещение равен 0, картинка прижимается влево. Если параметр смещение равен 1, картинка выравнивается по центру. Если параметр смещение равен 2, картинка прижимается к правой границе зоны печати графики(512 точек).

После печати изображения возвращается стандартный ответ с кодом ошибки.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.3.*10 Поиск графического изображения (0x19)**

Функция используется поиска и проверки загруженного графического изображения в памяти устройства.

**Входные параметры: (*Строка)***[1..15] – идентификатор картинки, передаваемый в функции(0x18).

**Ответные параметры: (*Целое число)*** Номер.

Если идентификатор не найден, возвращается “0”, если найден “1”

***3.4 Команды основных операций ККТ***

**3.4.1 Распечатать отчет без гашения (X-отчет) (0x20)**

**Функция:** *int libPrintXReport(const char\* nameCashier);*

**Входные параметры:** *nameCashier* **(Имя оператора)** Код и/или имя оператора

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.2 Распечатать отчет с гашением (Z-отчет) (0x21)**

**Функция:** *int libPrintZReport(const char\* nameCashier, int options);*

**Входные параметры:** *nameCashier* **(Имя оператора)** Код и/или имя оператора, *options* (***Целое число***) Опции отчета.

Если параметр “Опции отчета” равен 11, печатается краткий Z-отчет, иначе параметр игнорируется.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.2 Открыть смену (0x23)**

**Функция:** *libOpenShiftEx(const char\* nameCashier, const char\* placeSettle, int timeZoneNumber)*

**Входные параметры:**

**(Имя оператора)** *nameCashier*

(***Целое число***) Место расчетов (тег 1187).

(***Целое число***) номер часовой зоны (1..11) (тег 1011)

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**(\*) - поддерживается, начиная с прошивки 665.4.32**

**3.4.3 Открыть документ (0x30)**

**Функция:** *int libOpenDocumentEx(unsigned char type, unsigned char numDepart, const char\* nameCashier*, *long docNumber, unsigned char taxN);*

Эта команда открывает новый документ и переводит ККТ в режим ввода документа.

После успешного выполнения этой команды во внутренней переменной «статуса документа» устанавливается соответствующий тип текущего документа. Этот статус можно получить по команде “Запрос флагов статуса ККТ”.

**Входные параметры:** *type* (***Целое число***)Режим и тип документа,*numDepart* (***Целое число1..99***)Номер отдела, *nameCashier* (**Имя оператора**)Код и/или имя оператора, (***Целое число***)Номер документа, (***Целое число***) Код системы налогообложения

Параметр режим и тип документа представляет собой битовую маску, определяющую тип открываемого документа и режим его формирования.

|  |  |  |
| --- | --- | --- |
| **Режим и тип документа** | | |
| **№ бита** | **Значение** | **Комментарий** |
| 0 | 1-Сервисный документ  2-Чек на продажу (приход)  3-Чек на возврат (возврат прихода)  4-Внесение в кассу  5-Изъятие  6-Покупка (расход)  7-Возврат покупки (возврат расхода) | Младший полубайт задает тип открываемого документа |
| 1 |
| 2 |
| 3 |
| 4 | 0 | Обычный режим формирования документа |
| 1 | Пакетный режим формирования документа |
| 5 | 0 | Обычный режим печати реквизитов |
| 1 | Режим отложенной печати реквизитов |

В пакетном режиме формирования документа, при успешном выполнении команд формирования чека (с кодами 40..49,52) , ответ на команду не посылается. Если к.л. команда выполняется с ошибкой, то на нее возвращается стандартный ответ с кодом ошибки и последующие команды формирования чека (с кодами 40..49,52) игнорируются до команд “Завершить документ” или “Аннулировать документ”. Такой документ должен быть аннулирован или повторен, начиная с команды вернувшей ошибку в обычном режиме формирования документа, после подачи команды ”Завершить документ”.

Номер отдела- число от 1 до 99, которое сохраняется в ККТ

Номер документа- При установке бита в таблице настроек ККТ «**Нумерация чеков со стороны внешней программы**» в значение «1», передается номер чека, в других ситуациях параметр игнорируется.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Если нужно указать адрес покупателя, то перед завершением документа вызовите следующую функцию:**

**unsigned long libSetBuyerAddress(const char \*buyerAddress) – в случае успешной установки адреса покупателя возвращает количество записанных символов в буфер, иначе – 0.**

**Дополнительно:**

**unsigned long libGetBuyerAddress(char \*buyerAddress, unsigned long baLength) –** считывает установленный адрес покупателя в указанный буфер и возвращает количество записанных символов;

**void libCleanBuyerAddress() –** очищает адрес покупателя.

**3.4.4 Завершить документ (0x31)**

**Функция:** *MData libCloseDocument(unsigned char cutPaper);*

**Входные параметры:** *cutPaper* **(*Целое число)*** Флаг отрезки**.**

Если параметр “Флаг отрезки” = 1, отрезка сервисных документов по завершению не выполняется.

Если параметр “Флаг отрезки” = 5, отрезка чеков продажи и возврата по завершению не выполняется.

**Методы для установки дополнительные параметров команды закрытия смены:**

Адрес покупателя (Тег 1008) передается в buyerAddress:

* *unsigned long libSetBuyerAddress(const char \*buyerAddress);*

Для указания признак расчета в "Интернет" (тег 1125), необходим пере методом закрытия передать “1” в Разные флаги documentFlags:

* *unsigned long libSetDocumentFlags(const char \*documentFlags);*

Для использования признак расчета в "Интернет" должны быть заполнены поля "Место расчётов" и "Адрес покупателя"

Место расчётов (Тег 1187) передается в placeSettle:

* *unsigned long libSetPlaceSettle(const char \*placeSettle);*
* *unsigned long libSetBuyerName(const char \*buyerName);*
* *unsigned long libSetBuyerInn(const char \*buyerInn);*
* *unsigned long libSetBuyerBirthDate(const char \*birthDate);*
* *unsigned long libSetBuyerCitizenship(const char \*citizenship);*
* *unsigned long libSetBuyerPassportNum(const char \*passportNum);*
* *unsigned long libSetBuyerPassportData(const char \*passportData);*
* *unsigned long libSetBuyerGeographicAddress(const char \*geographicAddress);*

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит:(***Целое число***) Сквозной номер документа, **(Строка*)*** Операционный счетчик, **(*Строка***) Строка КПК**.**

Строка КПК возвращается только при завершении чеков на продажу и возврат.

В пакетном режиме формирования документа, команда “Завершить документ” выключает пакетный режим, Если в пакетном режиме ошибка возникла ранее команды “Завершить документ”, то ответ команду “Завершить документ” не возвращается. Можно продолжить формирование документа, начиная с команды вернувшей ошибку, обычном режиме.

**3.4.5 Аннулировать документ (0x32)**

**Функция:** *int libCancelDocument();*

Эта команда прерывает формирование текущего документа, данные удаляются из оперативной памяти ККТ и печатается сообщение об аннулировании.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.6 Отложить документ (0x33)**

**Функция:** *int libPostponeDocument(const char\* info);*

Эта команда работает аналогично команде “Аннулировать документ”, но подается в случаях, когда документ отменяется не на всегда, а будет обязательно повторно введен, в течении данной смены. Данные документа удаляются из оперативной памяти ККТ и печатается причина отказа от чека.

Используя параметры печати Z-отчета в “Таблица настроек ККТ”, можно настроить в Z-отчете печать информации по отложенным за смену чекам. При этом, если команда «Отложить чек» выполняется без параметра(пустая строка), то такие чеки учитываются в Z-отчете, если с параметром – не учитываются.

**Входные параметры:** *info* (**Строка**[40]) Причина отказа от чека.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.6 Отрезать документ (0x34)**

**Функция:** *int libCutDocument();*

Эта команда выполняет принудительную отрезку документа с предпечатью.

**Входные параметры: нет**.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.8 Печать текста (0x40)**

**Функция:** *int libPrintString(char\* textStr, unsigned char attribute);*

C помощью данной команды печатается текст внутри открытого сервисного документа.

**Входные параметры:** *textStr* (**Строка**[1..72]) Текст, *attribute* (***Целое число)*Атрибуты текста**

Атрибуты текста – опциональный параметр, представляющий собой битовую маску, значения которой приведены в таблице:

|  |  |
| --- | --- |
| **№ бита** | Значения атрибутов текста |
| **0..3**  **(N:шрифта)** | 0 –Шрифт 13х24, 44 символа в строке |
| 1 –Шрифт 10х20 |
| 2–Шрифт 13х24 жирный |
| 3 –Шрифт 10х20 жирный |
| 4 –Шрифт 8х14, 56 символов в строке |
| 5 –Шрифт 24х45 |
| 6 –Шрифт 24х45 жирный |
| **4** | Печать двойной высоты текста |
| **5** | Печать двойной ширины текста |
| **6** | Не используется |
| **7** | Не используется |

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.9 Печатать штрих-код (0x41)**

**Функция:** *int libPrintBarCode(unsigned char posText, unsigned char widthBarCode, unsigned char heightBarCode, unsigned char typeBarCode, const char\* barCode);*

С помощью данной команды можно распечатать штрих-код товара.

**Входные параметры:** *posText* (***Целое число***)Вывод текста,*widthBarCode* (***Целое число***)Ширина штрих-кода, *heightBarCode* (***Целое число***)Высота штрих-кода, *typeBarCode* (***Целое число***)Тип штрих-кода, *barCode* (**Строка**)Штрих-код.

Вывод текста (только для линейных кодов):

0-не выводить;

1-вывести наверху штрих-кода;

2-вывести внизу;

3-вывести наверху и внизу;

Ширина штрих-кода - значение задается в точках и может быть от 2 до 8.

Для UPC-A и EAN-13 максимальная ширина штрих-кода = 6, для остальных кодов зависит от длины параметра “Штрих-код”.

Высота штрих-кода - значение задается в точках и может принимать значения от 1 до 255 (только для линейных кодов).

Тип штрих-кода - Определяет, какой штрих-код будет напечатан. Длина и набор символов определяется типом штрих-кода:

0 - UPC-A;

1 - UPC-E;

2 - EAN-13;

3 - EAN-8;

4 - Code 39;

5 - Interleaved 2 of 5;

6 - Codabar.

7 – PDF417.

8 – QR CODE.

9 - Code 128

Штрих-код- строка содержащая штрих-код, причем контрольная сумма может и не указываться.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.10 Добавить товарную позицию (0x42)**

**Функция:** *int* ***libAddPosition****(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart, unsigned char coefType, const char \*coefName, double coefValue); [устаревшая]*

*int* ***libAddPositionEx****(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart, long signMethodCalculation, long signCalculationObject); [для прошивоки 560]*

*int* ***libAddPositionLarge****(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart, unsigned char coefType, double coefValue, long signMethodCalculation, long signCalculationObject); [для прошивок \*70 и выше]*

Эта команда может быть вызвана сразу после открытия чека и может повторяться любое количество раз внутри текущего документа для отражения всего списка товаров. Если позиция не может быть добавлена в ККТ, на чеке после данных о позиции печатается строка ”ОПЕРАЦИЯ ОТМЕНЕНА” и возвращается соответствующий код ошибки ККТ.

**Входные параметры:**

*goodsName:* (**Строка**[0...224]) Название товара,

*barcode:* (**Строка**[0..18]) Артикул или штриховой код товара/номер ТРК,

*quantity* (**Дробное число**)Количество товара в товарной позиции,

*price* (**Дробное число**) Цена товара по данному артикулу,

*taxNumber* (**Целое число**)Номер ставки налога,

*numGoodsPos* (**Строка**[0..4]) Номер товарной позиции,

*numDepart* (**Целое число 1..16**)Номер секции,

*coefType –* не используется, должно быть 0,

*coefValue –* не используется, должно быть 0,

*signMethodCalculation* (**Целое число**) Признак способа расчета,

*signCalculationObject* (**Целое число**) Признак предмета расчета.

Тип скидки или наценки:

0 – нет скидки или наценки (CoefficientTypes::NONE);

1 – процентная скидка (CoefficientTypes::DISCOUNT\_PERCENT);

2 – скидка на сумму (CoefficientTypes::DISCOUNT\_SUM);

3 – процентная наценка (CoefficientTypes::MARGIN\_PERCENT);

4 – наценка на сумму (CoefficientTypes::MARGIN\_SUM).

Для повышения точности вычислений, можно передавать количество с точностью до 9 знаков после запятой, при этом в умножении на цену будет участвовать 9 знаков после запятой, а печататься только первые 3.

Сумма позиции, получаемая в результате умножения цены на количество, округляется к ближайшему целому, т.е. часть менее 0.5 коп отбрасывается, 0.5 коп и более округляется до 1 коп.

Параметр taxNumber номера ставки налога может принимать следующие значения:

0 - Ставка "НДС 20%"

1 - Ставка "НДС 10%"

2 - Ставка "НДС 0%"

3 - Ставка "без НДС"

4 - Ставка "НДС 20/120"

5 - Ставка "НДС 10/110"

6 - Ставка "НДС 5%" (\*)

7 - Ставка "НДС 7%" (\*)

8 - Ставка "НДС 5/105" (\*)

9 - Ставка "НДС 7/107" (\*)

(\*) - Реализованы, начиная с версии прошивки 565.1.32

Параметры ”Номер товарной позиции” и “Номер секции” не являются обязательными и могут отсутствовать. Если номер секции отсутствует (или равен нулю), учет ведется на номер отдела, указанный при открытии документа.

Номер позиции - символьное поле, состоящее из цифр и символа разделителя, которым может быть пробел, двоеточие, тире и другие символы.

Признак способа расчета:

1- Предоплата 100%;

2- Предоплата;

3- Аванс;

4- Полный расчет

5- Частичный расчет и кредит

6- Передача в кредит

7- Оплата кредита

Признак предмета расчета:

|  |  |
| --- | --- |
| **Значение реквизита** | Название товара содержит сведения |
| 1 | о реализуемом товаре, за исключением подакцизного товара (наименование и иные сведения, описывающие товар) |
| 2 | о реализуемом подакцизном товаре (наименование и иные сведения, описывающие товар) |
| 3 | о выполняемой работе (наименование и иные сведения, описывающие работу) |
| 4 | об оказываемой услуге (наименование и иные сведения, описывающие услугу) |
| 5 | о приеме ставок при осуществлении деятельности по проведению азартных игр |
| 6 | о выплате денежных средств в виде выигрыша при осуществлении деятельности по проведению азартных игр |
| 7 | о приеме денежных средств при реализации лотерейных билетов, электронных лотерейных билетов, приеме лотерейных ставок при осуществлении деятельности по проведению лотерей |
| 8 | о выплате денежных средств в виде выигрыша при осуществлении деятельности по проведению лотерей |
| 9 | о предоставлении прав на использование результатов интеллектуальной деятельности или средств индивидуализации |
| 10 | об авансе, задатке, предоплате, кредите, взносе в счет оплаты, пени, штрафе, вознаграждении, бонусе и ином аналогичном предмете расчета |
| 11 | о вознаграждении пользователя, являющегося платежным агентом (субагентом), банковским платежным агентом (субагентом), комиссионером, поверенным или иным агентом |
| 12 | о предмете расчета, состоящем из предметов, каждому из которых может быть присвоено значение от «0» до «11» |
| 13 | о предмете расчета, не относящемуся к предметам расчета, которым может быть присвоено значение от «0» до «12» |

Ответные параметры: (**Дробное число**) Сумма налога в рублях, с точностью до 4 знаков после запятой (лишние знаки передаются КП для корректного округления до копеек).

**3.4.10.2 Установить дополнительные реквизиты позиции (0x24)**

Описание

Команда вызывается перед командой добавления товарной позиции (0x42) и устанавливает дополнительные реквизиты(тэги) для этой товарной позиции. Действие команды распространяется на одну товарную позицию в открытом документе.

**Функция:** *int libSetExtraRequisiteEx(const char\* nomenclatureCode,*

*const char\* extReq, const char\* measureName,*

*const char\* agentSign, const char\* supplierINN,*

*const char\* supplierPhone, const char\* supplierName,*

*const char\* operatorAddress, const char\* operatorINN,*

*const char\* operatorName, const char\* operatorPhone,*

*const char\* payAgentOperation, const char\* payAgentPhone,*

*const char\* recOperatorPhone, const char\* idFOIV,*

*const char\* establishmentDocDate, const char\* establishmentDocNum,*

*const char\* sectoralRequisite)*

**Входные параметры:**

(Строка) Код товара или код маркировки (Тег 1162 для ФФД 1.05 или тег 1163 для ФФД 1.2)

(Строка[1..64]) Дополнительный реквизит предмета расчёта (Тег 1191)

(Строка) Зарезервировано (для передачи тега 1197 используется команда 0x42)

(Целое число) Признак агента по предмету расчёта (Тег 1222)

(Строка)[0..12] ИНН поставщика (Тег 1226)

(Строка)[0..40] Телефон(ы) поставщика (Тег 1171)

(Строка)[0..256] Наименование поставщика (Тег 1225)

(Строка)[0..256] Адрес оператора перевода (для банк.пл.агента/субагента, иначе пустой) (Тег 1005)

(Строка)[0..12] ИНН оператора перевода (для банк.пл.агента/субагента, иначе пустой) (Тег 1016)

(Строка)[0..64] Наименование оператора перевода (для банк.пл.агента/субагента, иначе пустой) (Тег 1026)

(Строка)[0..40] Телефон(ы) оператора перевода (для банк.пл.агента/субагента, иначе пустой) (Тег 1075)

(Строка)[0..24] Операция платежного агента (для банк.пл.агента/субагента, иначе пустой) (Тег 1044)

(Строка)[0..60] Телефон(ы) платежного агента (для пл.агента/субагента, иначе пустой) (Тег 1073)

(Строка)[0..60] Телефон(ы) оператора по приему платежей (для пл.агента/субагента, иначе пустой) (Тег 1074)

(Число)[0..3] Идентификатор ФОИВ (тег 1262). Значение определяется ФНС РФ. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.

(Дата8) Дата документа основания (тег 1263) в формате ddmmyyyy. Должен содержать сведения об НПА отраслевого регулирования. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.

(Строка)[0..32] Номер документа основания (тег 1264). Должен содержать сведения об НПА отраслевого регулирования. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.

(Строка)[0..256] Значение отраслевого реквизита (тег 1265). Значение определяется отраслевым НПА. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.

**Ответные параметры**:

(Строка)[0..8] Результат проверки поля "Код товара" (Реализован, начиная с версии 665.3.0, при некорректном поле будет возвращено значение "Unrecogn")

**3.4.11 Сторнировать товарную позицию (0x43)**

**Функция:** *int libDelPosition(const char\* goodsName, const char\* barcode, long quantity, long price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart);*

Эта команда может сторнировать любую товарную позицию в чеке. Если позиция не может быть сторнирована в ККТ, на чеке после данных о позиции печатается строка ”ОПЕРАЦИЯ ОТМЕНЕНА” и возвращается соответствующий код ошибки ККТ.

**Входные параметры:** *goodsName* (**Строка**[0..224]) Название товара,*barcode*(**Строка**[0..18]) Артикул или штриховой код товара/ номер ТРК, *quantity* (***Целое число***) Количество товара в товарной позиции в 1/1000, *price* (***Целое число***) Цена товара по данному артикулу в копейках, *taxNumber* (***Целое число 0..5***) Номер ставки налога, *numGoodsPos* (***Целое число*** 0..4) Номер товарной позиции, *numDepart* (***Целое число 1..16***) Номер секции.

Параметры ”Номер ставки налога”, ”Номер товарной позиции” и “Номер секции” не являются обязательными и могут отсутствовать. Если номер секции отсутствует (или равен нулю), учет ведется на номер отдела, указанный при открытии документа.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.12 Подытог (0x44)**

**Функция:** *int libSubTotal();*

Эта команда заканчивает ввод товаров в чеке. Команда «Подитог» может даваться дважды.

После первой команды «Подитог» становится невозможно добавлять новые позиции в чек. Далее можно установить скидки/наценки на весь чек, зарегистрировать суммы налогов, распечатать дополнительные реквизиты, прервать оформление чека командами «Отложить чек» и «Аннулировать чек» или продолжить оформление с использованием повторной команды «Подитог» или команды «Оплата».

После повторной команды «Подитог», документ переходит в состояние оплаты. Далее мы можем распечатать дополнительные реквизиты, прервать оформление чека командами «Отложить чек» и «Аннулировать чек», либо продолжить оформление документа, выполнив команду «Оплата» и команду «Завершить документ».

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.13 Скидка на товар/чек (0x45)**

**Функция:** *int libAddDiscount(unsigned char typeDiscount, const char\* nameDiscount, long sum);*

Данная команда позволяет устанавливать процентные или абсолютные скидки на отдельный товар или непосредственно на весь чек. Команда может быть вызвана несколько раз, при этом необходимо помнить, что итоговая сумма после применения скидки не должна быть отрицательной.

Для установки скидки на товар команда должна быть вызвана непосредственно после команды “Добавить товарную позицию” и для отмены скидки, после команды “ Сторнировать товарную позицию”.

Для установки скидки на весь чек команда вызывается после первой команды “Подитог ”

**Входные параметры:** *typeDiscount* (**Целое число**) Тип скидки, *nameDiscount* (**Строка**[0..38]) Название скидки, *sum* (**Дробное число**) Процент (в тысячных долях процента) или сумма скидки(в копейках).

Тип скидки:

0- процентная скидка;

1- скидка на сумму;

Сумма скидки, вычисляемая при процентной скидке, округляется к ближайшему целому.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.14 Наценка на товар/чек (0x46)**

**Функция:** *int libAddMargin(unsigned char typeDiscount, const char\* nameDiscount, long sum);*

Данная команда позволяет устанавливать процентные или абсолютные наценки на отдельный товар или непосредственно на весь чек. Команда может быть вызвана несколько раз, при этом необходимо помнить, что итоговая сумма после отмены наценки не должна быть отрицательной.

Для установки наценки на товар команда должна быть вызвана непосредственно после команды “Добавить товарную позицию” и для отмены наценки, после команды “ Сторнировать товарную позицию”.

Для установки наценки на весь чек команда вызывается после первой команды “Подитог ”

**Входные параметры:** *typeDiscount* (**Целое число**) Тип скидки, *nameDiscount* (**Строка**[0..38]) Название скидки, *sum* (**Целое число**) Процент (в сотых долях процента) или сумма скидки(в копейках).

Тип наценки:

0- процентная наценка;

1- наценка на сумму;

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.15 Оплата (0x47)**

**Функция:** *int libAddPayment(unsigned char typePayment, long sum, const char\* infoStr);*

С помощью этой команды производится фиксирование всех взаиморасчетов с клиентом с указанием сумм и типов оплаты. При первом использовании этой команды в чеке печатается «Итоговая сумма», что является окончательной суммой чека. После этого производить скидки на чек нельзя.

При передаче суммы наличными, больше требуемой, ККТ самостоятельно рассчитывает сдачу. Сумма по безналичным типам платежа (с кодами от 1 до 15), не должна превышать итоговой суммы по чеку.

**Функция:** *int libAddPaymentEx(unsigned char typePayment, long long sum, const char\* infoStr, int cashlessSign, const char\* cashlessId, const char\* cashlessInfo)*

*Добавлены параметры безналичного расчета:*

**(Целое число)** *cashlessSign -* Признак способа оплаты безналичными (тег 1236)

**(Строка[0..256])** *cashlessId* - Идентификатор безналичной оплаты (тег 1237)

**(Строка[0..256])** *cashlessInfo* - Дополнительные сведения о безналичной оплате (тег 1238)

**Входные параметры:** *typePayment*(**Целое число 0..15**) Код типа платежа, *sum* (**Целое число**) Сумма, принятая от покупателя по данному платежу в копейках, *infoStr* (**Строка**[0..40]) Дополнительный текст.

Код типа платежа должен соответствовать одному из запрограммированных средств оплаты в “Таблице настроек ККТ”.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.16 Внесение / изъятие суммы (0x48)**

**Функция:** *int libCashInOut(const char\* infoStr, long sum);*

Команда регистрирует внесение или изъятие суммы денег из денежного ящика.

Команда выполняется после команды “Открыть документ**”**. Если производится внесение денег, тип документа должен быть 4, если изъятие - 5.

При этом может быть напечатана информация о купюрах, которые были использованы при выполнении данной операции.

**Входные параметры:** *infoStr* (**Строка**[0..40]) Название или тип купюры, *sum* (**Целое число**) Сумма в копейках.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.17 Печать реквизита (0x49)**

**Функция:** *int libPrintRequsit(unsigned char codeReq, unsigned char attributeText, const char\* str1,const char\* str2,const char\* str3,const char\* str4);*

C помощью данной команды могут печататься дополнительные реквизиты чека, при открытом чеке на продажу, возврат, внесения или изъятия.

**Входные параметры:** *codeReq* (***Целое число)* Код реквизита,***attributeText* (***Целое число)*Атрибуты текста,** *str1*(**Строка**[1..72]) Значение реквизита 1-я строка, *str2* (**Строка**[0..72]) Значение реквизита 2-я строка, *str3* (**Строка**[0..72]) Значение реквизита 3-я строка, *str4* (**Строка**[0..72]) Значение реквизита 4-я строка.

Код реквизита должен соответствовать одному из запрограммированных наименований реквизита в “Таблице настроек ККТ”. Если значение реквизита обеспечивает однозначное его толкование, его можно распечатать без предварительно запрограммированного наименования, с кодом реквизита равным нулю. Для печати всех запрограммированных реквизитов поставщика услуг, код реквизита должен быть равен 10.

Атрибуты текста – параметр, представляющий собой битовую маску, значения которой приведены в таблице:

|  |  |
| --- | --- |
| **№ бита** | Значения атрибутов текста |
| **0..3**  **(N:шрифта)** | 0 –Шрифт 13х24, 44 символа в строке |
| 1 –Шрифт 10х20 |
| 2–Шрифт 13х24 жирный |
| 3 –Шрифт 10х20 жирный |
| 4 –Шрифт 8х14б 56 символов в строке |
| 5 –Шрифт 24х45 |
| 6 –Шрифт 24х45 жирный |
| **4** | Печать двойной высоты текста |
| **5** | Печать двойной ширины текста |
| **6** | Не используется |
| **7** | Не используется |

Значение реквизита может состоять из 4-х строк, при этом 1-я строка значения реквизита печатается непосредственно за наименованием реквизита в одну строку. Общая длина наименования и значения реквизита составляет не более 4-х строк.

Если значение реквизита состоит из одной строки и первым символом является символ '&', то перевод строки после печати данного реквизита не выполняется, можно на этой же строке начать печать следующего реквизита.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.18 Зарегистрировать сумму по отделу (0x50)**

**Функция:** *int libRegisterSumToDepart(unsigned char typeOperation, unsigned char numberDepart, long sum);*

Команда регистрирует сумму по указанному отделу для одного из типов операции продажа или возврат. Эта команда может быть вызвана после любой команды.

Для печати итоговых сумм по отделам на Z-отчете необходимо запрограммировать заголовок группы отделов в “Таблице настроек ККТ”.

**Входные параметры:** *typeOperation* (**Целое число**)Тип операции, *numberDepart* (**Целое число**)Номер отдела в “Таблице настроек ККТ”, *sum* (**Целое число**) Сумма в копейках.

Тип операции указывает, для какого типа операций регистрируется сумма по товарной группе:

0 – продажи;

1 – возвраты;

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.19 Зарегистрировать сумму налога (0x51)**

**Функция:** *int libRegisterTaxSum(unsigned char numberTax, long sum);*

Команда регистрирует суммы налогов по запрограммированным налоговым ставкам. Эта команда может быть вызвана после любой команды, если документ открыт.

Данную команду целесообразно использовать, если нет необходимости вычислять налоги на каждую позицию, а достаточно регистрировать суммы налогов уже после вычисления суммы чека, непосредственно после команды «Подитог».

При этом, печать налогов на чеке возможна только при использовании команды «Установить сумму налога» после первой команды «Подитог» и при включении соответствующего параметра в “Таблице настроек ККТ”

**Входные параметры:** *numberTax* (**Целое число**) Номер ставки налога в “Таблице настроек ККТ”, *sum* (**Целое число**) Сумма в копейках.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.19 Сравнить сумму по чеку(0x52)**

**Функция:** *int libCompareSum(long sum);*

Команда сравнивает текущую сумму по чеку с переданной в параметре и возвращает ошибку “Некорректный формат или параметр команды”, если суммы не совпадают. Эта команда может быть вызвана после любой команды, если документ открыт.

**Входные параметры:** (**Целое число**) Сумма в копейках для сравнения.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.20 Открыть копию чека (0x53)**

**Функция:** *int libOpenCopyReceipt(unsigned char type, unsigned char numDepart, const char\* nameCashier, int numCheck, int numCash, MPiritDate mpDate, MPiritTime mpTime);*

Эта команда открывает копию чека, как сервисный документ, с печатью необходимых реквизитов оригинала чека. Реквизиты оригинала чека передаются во входных параметрах команды. Для формирования тела копии доступны команды: “Печать текста”, “Добавить товарную позицию”, “Подитог”, “Скидка на товар/чек”, “Наценка на товар/чек”, “Оплата”.

**Входные параметры:** *type* (***Целое число***)Тип чека, *numDepart*(***Целое число1..99***)Номер отдела, *nameCashier* (**Имя оператора**)Код и/или имя оператора, *numCheck* (***Целое число***)Номер чека, *numCash* (***Целое число1..9999***)Логический номер кассы, *mpDate* (**Дата**)Дата чека, *mpTime* (**Время**)Время чека.

Тип чека: 2- продажа, 3- возврат.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.21 Обнулить наличные в денежном ящике (0x54)**

**Функция:** *int libSetToZeroCashInCashDrawer ();*

Команда обнуляет счетчик наличных в денежном ящике.

**Входные параметры: нет**.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.22 Печать графики в документе (0x55)**

**Функция:** *int libPrintPictureInDocument(int width, int height, int sm, unsigned char\* data);*

Функция используется для печати графики в открытом документе.

Максимальная ширина изображения при использовании бумаги шириной 80мм – 512 точек(64 мм). Максимальная ширина изображения при использовании бумаги шириной 57мм – 336 точек точек(42 мм). 176 точек слева должны быть без изображения(как показано на рис. 1).

Графика печатается прижатой к левому краю, вне зоны печати фискального логотипа.

Команда выполняется в два этапа.

На первом передается параметры графики для печати.

**Входные параметры:** *width* **(*Целое число)*** Ширина картинки, *height* **(*Целое число)*** Высота картинки, *sm* **(*Целое число)*** Смещение от левого края, *data* **(*Массив байт)*** изображение аналогичное монохромному BMP.

Если параметр смещение равен 0, картинка прижимается влево. Если параметр смещение равен 1, картинка выравнивается по центру. Если параметр смещение равен 2, картинка прижимается к правой границе зоны печати графики(512 точек).

После печати изображения возвращается стандартный ответ с кодом ошибки.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.23 Печать загруженной картинки (0x56)**

**Функция:** *int libPrintPreloadedPicture(int sm, int number);*

Функция используется для печати графики загруженной ранее в память устройства командой “Загрузить графическое изображение (0x18)”.

**Входные параметры:** *sm* **(*Целое число)*** Смещение, *number* **(*Целое число)*** Номер(всегда =1).

Если параметр смещение равен 0, картинка прижимается влево. Если параметр смещение равен 1, картинка выравнивается по центру. Если параметр смещение равен 2, картинка прижимается к правой границе зоны печати графики(512 точек). Если параметр смещение равен 3 – используется значение смещения, заданное при загрузке.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.4.24 Печать реквизита для ОФД (0x57)**

**Функция:** *int libPrintRequsitOFD(int codeReq, unsigned char attributeText, const char\* reqName, const char\* reqStr)*

C помощью данной команды могут печататься дополнительные реквизиты чека для передачи в ОФД , при открытом чеке на продажу (приход), возврат (возврат прихода), внесения или изъятия. В ОФД будет отправлено только значение реквизита.

**Входные параметры:** (***Целое число)* Код реквизита*,*** (***Целое число)* Атрибуты текста*, (Строка[1..72])*** Название реквизита, ***(Строка[0..255])*** Значение реквизита.

Код реквизита должен соответствовать перечню реквизитов из документа "Форматы ОФД".

Атрибуты текста – параметр, представляющий собой битовую маску, значения которой приведены в таблице:

|  |  |
| --- | --- |
| **№ бита** | **Значения атрибутов** **текста** |
| 0..3  (N:шрифта) | 0 –Шрифт 12х24 |
| 1 –Шрифт 9х17 |
| 4 | Печать двойной высоты текста |
| 5 | Печать двойной ширины текста |
| 6 | Целочисленный или суммовой атрибут (не строка) |
| 7 | Реквизит не печатается, только отправляется в ОФД |

Если первым символом названия реквизита является символ '&', то перевод строки после печати данного реквизита не выполняется, можно на этой же строке начать печать следующего реквизита.

Если при установленном бите 6 первым символом значения реквизита является '#', то атрибут рассматривается, как целое число, иначе - как денежная сумма.

**Ответные параметры:** Код ошибки.

**3.4.24** **Сформировать чек коррекции (0x58)**

**Функция:** *libDoCheckCorrection(char\* nameCashier, double cash, double cashless, unsigned char correctionFlags); [устаревшая]*

*int libDoCheckCorrectionEx(const char \*nameCashier, double cash, double cashless, double sum1, double sum2, double sum3, unsigned char correctionFlags, const PLDate \*docDate, const char \*docNumber, const char \*correctionName, double sum\_18, double sum\_10, double sum\_0, double sum\_WT, double sum\_18\_118, double sum\_10\_110); [для прошивок 560 и \*70]*

*Для ФФД 1.2:*

*libOpenCorrection(const char \*nameCashier, int correctionType, const PLDate \*docDate, const char \*docNumber);*

*Входные параметры*

*(Имя оператора)* ***nameCashier*** *Имя оператора*

*(Число)* ***correctionType*** *Тип коррекции*

*(Дата)* ***docDate*** *Дата документа основания коррекции*

*(Строка)[1..32]* ***docNumber*** *Номер документа основания коррекции*

Входные параметры:

**(Имя оператора)** Имя оператора,

**(Дробное число) Сумма наличного платежа,**

**(Дробное число) Сумма электронного платежа,**

**(Дробное число) Сумма предоплатой,**

**(Дробное число) Сумма постоплатой,**

**(Дробное число) Сумма встречным представлением,**

**(Число)** Тип коррекции,

*(***Дата)Д**ата документа основания коррекции**,**

**(Строка[1..32]) Номер** документа основания коррекции**,**

**(Строка[1..64]) Наименование**основания коррекции**,**

**(Дробное число) Сумма налога по ставке 18%,**

**(Дробное число) Сумма налога по ставке 10%,**

**(Дробное число) Сумма расчета по ставке 0%,**

**(Дробное число**) **Сумма расчета без налога*,***

**(Дробное число) Сумма расчета по расч. ставке 18/118,**

**(Дробное число) Сумма расчета по расч. ставке 10/110.**

Тип коррекции:

|  |  |
| --- | --- |
| № бита | **Значения атрибутов****текста** |
| 0 | 0 – самостоятельная коррекция  1 – по предписанию |
| 1 | 0 – приход  1 – расход |
| 2-4 | Система налогообложения (0..5) |
| 5 | Если бит = 0, то вместо суммы налога в поле "Сумма налога по ставке 18%" передается номер ставки налога, по которой исчисляется сумма налога, остальные суммы налогов не воспринимаются |

CORRECTION\_INCOMING 0x00 – приход;

CORRECTION\_OUTGOING 0x02 – расход.

***3.5 Команды регистрации***

Пароль налогового инспектора по умолчанию “32323”.

**3.5.1 Фискализация/Перерегистрация (0x60)**

**Функция:** *int libFiscalization(const char \*oldPassword, const char \*regNumber, const char \*INN, const char \*newPassword); [устаревшая]*

*int PIRITLIB\_CALL libRegistrationEx(unsigned char type, const char \*regNumber, const char \*INN, int systemTax, int rej, const char \*cashierName, PLDateTime regDateTime, const char \*userName1, const char \*userName2, const char \*addressSettle1, const char \*addressSettle2, const char \*placeSettle, const char \*autoNumber, const char \*OFDName, const char \*OFDINN, const char \*senderEmail, const char \*FNSEmail, bool automatic); [для прошивок \*70]*

При вводе в эксплуатацию ККТ данная операция называется «фискализацией». Функция переводит ККТ из «нефискального» режима в «фискальный» и выполняет активацию ККТ, что влечет за собой запоминание всех операций в памяти ККТ и их учет для проверки налоговыми органами.

Далее производится **перерегистрация** в случае смены юридического владельца ККТ.

**Входные параметры:**

**(Число[0..1])** Замена ФН

**(Строка[16])** Регистрационный номер

**(Строка[12])** ИНН пользователя

**(Число)** Система налогообложения

**(Число)** Режим работы

**(Строка)** Имя кассира

**(ДатаВремя)** Текущая дата и время

**(Строка[0..44])** Наименование пользователя (строка 1)

**(Строка[0..44])** Наименование пользователя (строка 2)

**(Строка[0..44])** Адрес пользователя (строка 1)

**(Строка[0..44])** Адрес пользователя (строка 2)

**(Строка[0..112])** Место расчетов.

**(Строка[0..12])** Номер автомата.

**(Строка[0..64])** Наименование ОФД.

**(Строка[0..12])** ИНН ОФД.

**(Строка[0..44])** Адрес электронной почты отправителя чека.

**(Строка[0..64])** Адрес сайта ФНС.

**(Boolean)** Признак установки принтера в автомате.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.5.2 Распечатать фискальный отчет по сменам (0x61)**

**Функция:** *int libPrintFiscalReportByShifts(unsigned char typeReport, int startShiftNumber, int endShiftNumber, const char \*password);*

**Входные параметры:** *typeReport* (**Целое число**)Тип отчета, *startShiftNumber* (**Целое число**)Номер начальной смены, *endShiftNumber* (**Целое число**) Номер конечной смены, *password* (**Строка**[1..10])Пароль налогового инспектора.

Параметр “Тип отчета” задает формат отчета:

0 - краткий; 1 - полный; 2- вернуть итоговую сумму.

**Ответные параметры: (Дробное число)** итоговая сумма (толькодля типа 2)**.**

**3.5.3 Распечатать фискальный отчет по датам (0x62)**

**Функция:** *int libPrintFiscalReportByDate(unsigned char typeReport, MPiritDate startDate, MPiritDate endDate, const char \*password);*

**Входные параметры:** *typeReport* (**Целое число**)Тип отчета, *startDate* (**Дата**) Начальная дата, *endDate* (**Дата**) Конечная дата, *password* (**Строка**[1..10])Пароль налогового инспектора.

Параметр “Тип отчета” задает формат отчета:

0 - краткий; 1-полный; 2- вернуть итоговую сумму.

**Ответные параметры: (Дробное число)** итоговая сумма, (толькодля типа 2).

**3.5.4 Технологическое обнуление (0x63)**

**Функция:** *int libTechnologicalReset(const DateTime \*dateTime);*

**Входные параметры:** *dateTime* (**Дата & время**) Текущая дата.

**Ответные параметры:** нет.

***3.6 Команды для работы с ККТ и ФН***

**3.6.1 Активизация ККТ (0x70)**

**Функция:** *int libActivizationECT();*

С помощью этой команды производится активизация новой ККТ, в случае замены предыдущей.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.2 Закрытие архива ККТ или ФН (0x71)**

**Функция:** *int libCloseArchiveECT();*

При помощи данной команды закрывается архив ККТ и становится невозможно добавлять новые операции в память ККТ.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Функция:** *int libCloseFN(const char \*cashierName)*

При помощи данной команды закрывается архив ФН и становится невозможно добавлять новые операции в память ФН.

**Входные параметры:** *cashierName* (**Строка**) Имя уполномоченного кассира**.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.3 Распечатать контрольную ленту из ККТ (0x72)**

**Функция:** *int libPrintControlTapeFromECT(int shiftNumber);*

**Входные параметры:** *shiftNumber* (**Целое число**)Номер смены.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.4 Распечатать документ из ККТ (0x73)**

**Функция:** *int libPrintDocumentFromECT (int KPKNumber);*

**Входные параметры:** *KPKNumber* (**Целое число**) Номер КПК.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.5 Распечатать отчет по сменам из ККТ (0x74)**

**Функция:** *int libPrintReportFromECTByShifts(unsigned char typeReport, int startShiftNumber, int endShiftNumber);*

**Входные параметры:** *typeReport* (**Целое число**)Тип отчета, *startShiftNumber* (**Целое число**)Номер начальной смены, *endShiftNumber* (**Целое число**) Номер конечной смены.

Параметр “Тип отчета” задает формат отчета:

0 - краткий; 1 - полный.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.6 Распечатать отчет по датам из ККТ (0x75)**

**Функция:** *int libPrintReportFromECTByDate(unsigned char typeReport, MPiritDate startDate, MPiritDate endDate);*

**Входные параметры:** *typeReport* (**Целое число**)Тип отчета, *startDate* (**Дата**) Начальная дата, *endDate* (**Дата**) Конечная дата.

Параметр “Тип отчета” задает формат отчета:

0 - краткий; 1 - полный.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.7 Распечатать отчет по активизации ККТ (0x76)**

**Функция:** *int libPrintReportActivizationECT();*

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.8 Распечатать отчет по смене из ККТ (0x77)**

**Функция:** *int libPrintReportFromECTByShift(int shiftNumber);*

**Входные параметры:** *shiftNumber* (**Целое число**)Номер смены.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.6.9 Запрос информации из ККТ или ФН (0x78)**

**Функция:** *MData libGetInfoFromECT(unsigned char number, long dataL1, long dataL2);*

Эта команда позволяет получать данные от ККТ или ФН. Тип и количество возвращаемых данных зависит от значения передаваемого параметра – номер запроса.

**Входные параметры:** *number* (***Целое число*** 1..4,10,11,12,20) Номер запроса, *dataL1* (***Целое число***) Номер КПК или смены(\*), *dataL2* (***Целое число***) Номер конечной смены(\*\*).

\* Параметр ”Номер КПК или смены” необходим только для запросов №10, 11 или 12.

\*\* Параметр ”Номер конечной смены” необходим только для запроса №13.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит:

(***Целое число***) Номер запроса, Возвращаемые данные.

Тип и количество возвращаемых данных зависит от значения номера запроса.

**Для ККТ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер**  **запроса** | **Наименование**  **запроса** | **Формат возвращаемых данных** | **Коментарии** |
| **1** | Вернуть регистрационный номер ККТ | **Строка** |  |
| **2** | Вернуть статус ККТ | **Целое число** |  |
| **3** | Вернуть номер последнего КПК | **Строка** |  |
| **4** | Вернуть дату и время активизации | **Дата**, **Время** |  |
| **5** | Вернуть номер закрытой смены, перед активизации | **Целое число** |  |
| **6** | Вернуть номер текущей смены | **Целое число** |  |
| **10** | Запрос данных ККТ по № КПК | ----------------- | **Все данные по запросу 10 обязательно должны быть зачитаны(с помощью запроса 20)** |
| **11** | Запрос данных ККТ по № смены | ----------------- | **Все данные по запросу 11 обязательно должны быть зачитаны(с помощью запроса 20)** |
| **12** | Запрос контрольной ленты из ККТ по № смены | ----------------- | **Все данные по запросу 12 обязательно должны быть зачитаны(с помощью запроса 20)** |
| **13** | Запрос краткого отчета по сменам из ККТ | ----------------- | **Все данные по запросу 13 обязательно должны быть зачитаны(с помощью запроса 20)** |
| **20** | Читать строку данных из ККТ | **Строка** | Возвращает строки по запросам №10..№12, сформированные ККТ, аналогично выводу их на печать командами: “Распечатать документ из ККТ”, “Распечатать отчет по смене из ККТ” и “Распечатать контрольную ленту из ККТ”.  **Чтение обязательно производится до получения строки нулевой длины (пустая строка).** |

**Для ФН**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер  запроса | Наименование  запроса | Формат возвращаемых данных | Комментарии |
| 1 | Вернуть регистрационный номер ФН | **Строка** |  |
| 2 | Вернуть статус ФН | **Целое число,**  **Целое число,**  **Целое число** | Состояние ФН  Состояние текущего документа  Флаги предупреждения |
| 3 | Вернуть номер последнего фискального документа | **Строка** |  |
| 4 | Вернуть дату и время регистрации | Дата, Время |  |
| 5 | Вернуть номер закрытой смены перед регистрацией | **Целое число** |  |
| 6 | Вернуть состояние текущей смены | **Целое число**  **Целое число**  **Целое число** | Номер смены  1, если смена открыта, 0 – если нет  Номер чека в смене |
| 7 | Вернуть состояние обмена с ОФД | **Целое число**  **Целое число**  **Целое число**  **Дата, Время** | Статус обмена  Количество документов для передачи в ОФД  Номер первого документа для передачи в ОФД  Дата/время первого док-та для передачи в ОФД |
| 11 | Запрос документа из архива | **Целое число**  **Массив HEX** | 1 - Если получена квитанция  Возвращается блок данных документа в шестнадцатеричном виде |
| 12 | Запрос квитанции о получении документа из архива | **Массив HEX** | Возвращается блок данных квитанции в шестнадцатеричном виде |
| 13 | Запрос последних ошибок ФН | **Массив HEX** | Возвращается блок данных в шестнадцатеричном виде |
| 14 | Запрос версии ФН | **Строка** | Возвращается версия в виде строки |

***3.7 Команды управления внешними устройствами***

**3.7.1 Открыть денежный ящик (0x80)**

**Функция:** *int libOpenCashDrawer(int pulseDuration);*

С помощью этой команды можно открыть денежный ящик, подключенный к ККТ.

**Входные параметры:** *pulseDuration* (**Целое число** 50...500) Длительность импульса в мс(150 по умолчанию).

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**3.7.2 Получить статус денежного ящика (0x81)**

**Функция:** *MData libGetCashDrawerStatus();*

**Входные параметры: нет.**

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: (**Целое число**) Статус: 0-закрыт;1-открыт**.**

**3.7.3 Подать звуковой сигнал (0x82)**

**Функция:** *int libBeep(int duration);*

**Входные параметры:** *duration* (**Целое число** 10...2000) Длительность в мс**.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

***3.8 Системные команды***

**Авторизация ККТ (0x90)**

**Функция:** *int libAuthorization(MPiritDate mpDate, MPiritTime mpTime, const char \*numKKT);*

С помощью этой команды предприятием-изготовителем вводится заводской номер ККТ.

**Входные параметры:** *mpDate* (**Дата**)Текущая дата, *mpTime* (**Время**)Текущее время, *numKKT* (**Строка**[12]) Заводской номер ККТ.

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Чтение блока памяти ККТ (0x91)**

**Функция:** *MData libReadMemoryBlock(unsigned char type, long startAdress, long numBytes);*

Функция используется для чтения блока данных из памяти ККТ. Данные передаются из ККТ в виде последовательности пар символов, каждая из которых является шестнадцатеричным представлением одного байта данных.

**Входные параметры:** *type* (**Целое число**) Тип данных, *startAdress* (**Целое число**)Начальный адрес, *numBytes* (**Целое число** 1..64) Количество байт.

Тип данных определяет, из какой области нужно передать данные. Возможные значения - от 1 до 6.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: Блок данных в шестнадцатеричном виде.

**Чтение ПО ФП (0x92)**

**Функция:**

Для включения режима чтения ПО ФП необходимо предварительно дать команду 0x92 с параметрами: 0 (включение режима чтения), “пароль” (пароль на чтение).

Для выключения режима чтения ПО ФП необходимо после выполнения операции чтения ПО ФП еще раз дать команду 0x92 с параметрами: 0, 0

**Входные параметры:** (**Целое число**)Тип операции, (**Целое число**)Начальный адрес, (**Целое число** 1..64) Количество байт.

**Ответные параметры: (*Структура* MData*)*** Ответ от ФР, который содержит: Блок данных в шестнадцатеричном виде.

**Установить скорость обмена (0x93)**

**Функция:** *int libSetSpeed(unsigned char numSpeed);*

**Входные параметры:** *numSpeed* (**Целое число** 0..5) Скорость обмена.

|  |  |
| --- | --- |
| Значение параметра ”Скорость обмена” | Значение скорости |
| 0 | 4800 |
| 1 | 9600 |
| 2 | 19200 |
| 3 | 38400 |
| 4 | 57600 |
| 5 | 115200 |

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Распечатать сервисные данные (0x94)**

**Функция:** *int libPrintServiceData();*

Эта команда печатает на чеке данные по текущим настройкам, состоянию и дополнительной сервисной информации.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Аварийное закрытие смены (0xA0)**

**Функция:** *int libEmergencyCloseShift ();*

Эта команда позволяет закрыть смену в случае неисправности ККТ. Результатом является отчет по закрытой смене, далее для продолжения работы необходимо заменить ККТ.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Печать копии последнего Z-отчета (0xA1)**

**Функция:** *int libPrintCopyLastZReport();*

Эта команда позволяет распечатать копию последнего Z-отчета, для проверки со стороны контролирующих органов.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Включение сервисного канала к ККТ (0xA2)**

**Функция:** *int libEnableServiceChannelToECT();*

Эта команда позволяет включить канал связи с ККТ, для выполнения сервисных операций стандартными сервисными утилитами ККТ.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Печать копии чека фискализации/перерегистрации (0xA3)**

**Функция:** *int libPrintCopyReportFiscalization();*

Эта команда позволяет распечатать копию чека фискализации/перерегистрации, для проверки со стороны контролирующих органов.

**Входные параметры: нет.**

**Ответные параметры: (*Целое число)*** *Код ошибки***.**

**Получение информации об ошибке**

**Функция:** *unsigned long libFormatMessage(int errorCode, char \*msgBuffer, unsigned long cbBuffer);*

Эта функция позволяет получить текстовое описание об ошибке.

**Входные параметры:**

errorCode – код ошибки;

msgBuffer – буфер для вывода сообщения;

cbBuffer – размер буфера в байтах.

**Ответные параметры: (*Целое число)*** *Количество записанных символов***.**

**Полный перечень функций PiritLib:**

int PIRITLIB\_CALL openPort(const char \*fileName, long speed);

int PIRITLIB\_CALL closePort();

int PIRITLIB\_CALL checkPortAsync(const char \*fileName, long speed);

int PIRITLIB\_CALL commandStart();

MData PIRITLIB\_CALL libGetStatusFlags();

int PIRITLIB\_CALL getStatusFlags(int \*fatalStatus, int \*currentFlagsStatus, int \*documentStatus);

MData PIRITLIB\_CALL libGetCountersAndRegisters(unsigned char numRequest);

MData PIRITLIB\_CALL libGetKKTInfo(unsigned char numRequest);

int PIRITLIB\_CALL getKKTInfo(unsigned char numRequest, char \*data);

MData PIRITLIB\_CALL libGetReceiptData(unsigned char numRequest);

MData PIRITLIB\_CALL libGetPrinterStatus();

MData PIRITLIB\_CALL libGetServiceInfo(unsigned char numRequest);

MData PIRITLIB\_CALL libGetExErrorInfo(unsigned char numRequest);

int PIRITLIB\_CALL scrollPaper();

int PIRITLIB\_CALL libCommandStart(PLDate mpDate, PLTime mpTime);

MData PIRITLIB\_CALL libReadSettingsTable(unsigned char number, int index);

int PIRITLIB\_CALL libWriteSettingsTable(unsigned char number, int index, const char\* data);

MData PIRITLIB\_CALL libGetPiritDateTime();

int PIRITLIB\_CALL libSetPiritDateTime(PLDate mpDate, PLTime mpTime);

int PIRITLIB\_CALL libLoadLogo(int size, unsigned char\* data);

int PIRITLIB\_CALL libDeleteLogo();

int PIRITLIB\_CALL libLoadReceiptDesign(int size, unsigned char\* data);

int PIRITLIB\_CALL libLoadPicture(int width, int height, int sm, const char\* name, int number, unsigned char\* data);

int PIRITLIB\_CALL libPrintXReport(const char\* nameCashier);

int PIRITLIB\_CALL libPrintZReport(const char\* nameCashier, int options);

int PIRITLIB\_CALL libOpenDocument(unsigned char type, unsigned char numDepart, char\* nameCashier, long docNumber);

int PIRITLIB\_CALL libOpenDocumentEx(unsigned char type, unsigned char numDepart, char\* nameCashier, long docNumber, unsigned char taxN);

unsigned long PIRITLIB\_CALL libSetBuyerAddress(const char \*buyerAddress);

unsigned long PIRITLIB\_CALL libGetBuyerAddress(char \*buyerAddress, unsigned long baLength);

void PIRITLIB\_CALL libCleanBuyerAddress();

MData PIRITLIB\_CALL libCloseDocument(unsigned char cutPaper);

int PIRITLIB\_CALL libCancelDocument();

int PIRITLIB\_CALL libPostponeDocument(const char\* info);

int PIRITLIB\_CALL libCutDocument();

int PIRITLIB\_CALL libPrintString(char\* textStr, unsigned char attribute);

int PIRITLIB\_CALL libPrintBarCode(unsigned char posText, unsigned char widthBarCode, unsigned char heightBarCode, unsigned char typeBarCode, const char\* barCode);

int PIRITLIB\_CALL libAddPosition(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart, unsigned char coefType, const char \*coefName, double coefValue);

int PIRITLIB\_CALL libAddPositionEx(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart, long signMethodCalculation, long signCalculationObject);

int PIRITLIB\_CALL libAddPositionLarge(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart, unsigned char coefType, double coefValue, long signMethodCalculation, long signCalculationObject);

int PIRITLIB\_CALL libDelPosition(const char\* goodsName, const char\* barcode, double quantity, double price, unsigned char taxNumber, int numGoodsPos, unsigned char numDepart);

int PIRITLIB\_CALL libSubTotal();

int PIRITLIB\_CALL libAddDiscount(unsigned char typeDiscount, const char\* nameDiscount, long sum);

int PIRITLIB\_CALL libAddMargin(unsigned char typeMargin, const char\* nameMargin, long sum);

int PIRITLIB\_CALL libAddPayment(unsigned char typePayment, long long sum, const char\* infoStr);

int PIRITLIB\_CALL libCashInOut(const char\* infoStr, long long sum);

int PIRITLIB\_CALL libPrintRequsit(unsigned char codeReq, unsigned char attributeText, const char\* str1, const char\* str2, const char\* str3, const char\* str4);

int PIRITLIB\_CALL libPrintRequsitOFD(int codeReq, unsigned char attributeText, const char\* reqName, const char\* reqStr);

int PIRITLIB\_CALL libRegisterSumToDepart(unsigned char typeOperation, unsigned char numberDepart, long sum);

int PIRITLIB\_CALL libRegisterTaxSum(unsigned char numberTax, long sum);

int PIRITLIB\_CALL libCompareSum(long sum);

int PIRITLIB\_CALL libOpenCopyReceipt(unsigned char type, unsigned char numDepart, const char\* nameCashier, int numCheck, int numCash, PLDate mpDate, PLTime mpTime);

int PIRITLIB\_CALL libSetToZeroCashInCashDrawer();

int PIRITLIB\_CALL libPrintPictureInDocument(int width, int height, int sm, unsigned char\* data);

int PIRITLIB\_CALL libPrintPreloadedPicture(int sm, int number);

int PIRITLIB\_CALL libTechnologicalReset(const PLDateTime \*dateTime);

int PIRITLIB\_CALL libFiscalization(const char \*oldPassword, const char \*regNumber, const char \*INN, const char \*newPassword);

int PIRITLIB\_CALL libPrintFiscalReportByShifts(unsigned char typeReport, int startShiftNumber, int endShiftNumber, const char \*password);

int PIRITLIB\_CALL libPrintFiscalReportByDate(unsigned char typeReport, PLDate startDate, PLDate endDate, const char \*password);

int PIRITLIB\_CALL libActivizationECT();

int PIRITLIB\_CALL libCloseArchiveECT();

int PIRITLIB\_CALL libCloseFN(const char \*cashierName);

int PIRITLIB\_CALL libPrintControlTapeFromECT(int shiftNumber);

int PIRITLIB\_CALL libPrintDocumentFromECT(int KPKNumber);

int PIRITLIB\_CALL libPrintReportFromECTByShifts(unsigned char typeReport, int startShiftNumber, int endShiftNumber);

int PIRITLIB\_CALL libPrintReportFromECTByDate(unsigned char typeReport, PLDate startDate, PLDate endDate);

int PIRITLIB\_CALL libPrintReportActivizationECT();

int PIRITLIB\_CALL libPrintReportFromECTByShift(int shiftNumber);

MData PIRITLIB\_CALL libGetInfoFromECT(unsigned char number, long dataL1, long dataL2);

int PIRITLIB\_CALL libOpenCashDrawer(int pulseDuration);

MData PIRITLIB\_CALL libGetCashDrawerStatus();

int PIRITLIB\_CALL getCashDrawerStatus(int \*drawerStatus);

int PIRITLIB\_CALL libBeep(int duration);

int PIRITLIB\_CALL libAuthorization(PLDate mpDate, PLTime mpTime, const char \*numKKT);

MData PIRITLIB\_CALL libReadMemoryBlock(unsigned char type, long startAdress, long numBytes);

int PIRITLIB\_CALL libSetSpeed(unsigned char numSpeed);

int PIRITLIB\_CALL libPrintServiceData();

int PIRITLIB\_CALL libEmergencyCloseShift();

int PIRITLIB\_CALL libPrintCopyLastZReport();

int PIRITLIB\_CALL libEnableServiceChannelToECT();

int PIRITLIB\_CALL libPrintCopyReportFiscalization();

int PIRITLIB\_CALL getCurMPTime(PLDate \*mpDate, PLTime \*mpTime);

int PIRITLIB\_CALL saveLogoToFile(wchar\_t \*fileName);

void PIRITLIB\_CALL setDebugLevel(int level);

long long PIRITLIB\_CALL getDriverVersion();

int PIRITLIB\_CALL libPrintDocsFromECTSDByNumberDoc(int startNumber, int endNumber);

int PIRITLIB\_CALL libPrintDocsFromECTSDByNumberShift(int startNumber, int endNumber);

int PIRITLIB\_CALL libPrintDocsFromECTSDByDate(PLDate mpDateStart, PLDate mpDateEnd);

int PIRITLIB\_CALL libGetInfoFromECT\_NumberDoc(int \*numDoc);

int PIRITLIB\_CALL libBLRPrintControlTapeFromECT();

int PIRITLIB\_CALL getCountersAndRegisters(int requestNumber, int \*data);

void PIRITLIB\_CALL setAmountDecimalPlaces(int decimalPlaces);

void PIRITLIB\_CALL setQuantityDecimalPlaces(int decimalPlaces);

void PIRITLIB\_CALL setPercentageDecimalPlaces(int decimalPlaces);

int PIRITLIB\_CALL getAmountDecimalPlaces();

int PIRITLIB\_CALL getQuantityDecimalPlaces();

int PIRITLIB\_CALL getPercentageDecimalPlaces();

int PIRITLIB\_CALL libGetCountersAndRegisters\_12(int data[], int maxElement, char \*str);

int PIRITLIB\_CALL getPrinterStatus(int\* result);

int PIRITLIB\_CALL getPiritDateTime(int\* cDate, int\* cTime);

int PIRITLIB\_CALL libOpenShift(const char\* nameCashier);

int PIRITLIB\_CALL libDoCheckCorrection(const char \*nameCashier, double cash, double cashless, unsigned char correctionFlags);

int PIRITLIB\_CALL libDoCheckCorrectionEx(const char \*nameCashier, double cash, double cashless, double sum1, double sum2, double sum3, unsigned char correctionFlags, const PLDate \*docDate, const char \*docNumber, const char \*correctionName, double sum\_18, double sum\_10, double sum\_0, double sum\_WT, double sum\_18\_118, double sum\_10\_110);

int PIRITLIB\_CALL libCurrentStatusReport(char\* nameCashier);

int PIRITLIB\_CALL libAddPaymentD(unsigned char typePayment, double sum, const char\* infoStr);

int PIRITLIB\_CALL libRegistration(unsigned char type, const char \*regNumber, const char \*INN, int systemTax, int rej, const char \*cashierName);

int PIRITLIB\_CALL libRegistrationEx(unsigned char type, const char \*regNumber, const char \*INN, int systemTax, int rej, const char \*cashierName, PLDateTime regDateTime, const char \*userName1, const char \*userName2, const char \*addressSettle1, const char \*addressSettle2, const char \*placeSettle, const char \*autoNumber, const char \*OFDName, const char \*OFDINN, const char \*senderEmail, const char \*FNSEmail, bool automatic);

int PIRITLIB\_CALL libGetInfoFromECT\_NumberFP(char \*data);

int PIRITLIB\_CALL libGetInfoFromECT\_FP(int numDoc, char \*data);

unsigned long PIRITLIB\_CALL libFormatMessage(int errorCode, char \*msgBuffer, unsigned long cbBuffer);

Все функции вызываются с соглашением о вызоые **\_\_stdcall**.

**4. Таблица настроек ККТ**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Индексы  (только для  массивов) | Наименование  (тип) | Значения |
| 1 | 0 | Параметры ПУ  (битовая маска) | |  |  |  | | --- | --- | --- | | № бита | Значение | Комментарий | | 0 | 0 | Нормальный режим печати | | 1 | Печать с уменьшенным межстрочным интервалом, для экономии бумаги(\*) | | 1 | 0 | Частичная отрезка чека(\*) | | 1 | Полная отрезка чека | | 2 | 0 | Не печатать графический логотип(\*) | | 1 | Печатать графический логотип  (логотип должен быть предварительно загружен в ПУ) | | 5 | 0 | |  | | --- | | Не печатать QR-код на чеке | | | 1 | |  | | --- | | Печатать QR-код на чеке(\*) | | |
| 2 | 0 | Параметры чека  (битовая маска) | |  |  |  | | --- | --- | --- | | 0 | Номер дизайна чека:  0- обычный (\*)  1- расширенный  2- экономный  3- нано  4 - расширенный для бумаги 57мм  5 - экономный для бумаги 57мм  16 и выше - дополнительные загружаемые дизайны | | | 1 | | 2 | | 3 | | 4 | | 5 | 0 | Зарезервирован | | 6 | 0 | Печатать наличные в ДЯ на чеках внесения/инкассации (\*) | | 1 | Не печатать наличные в ДЯ на чеках внесения/инкассации | | 7 | 0 | Нумерация чеков ККТ (\*) | | 1 | Нумерация чеков внешней программой | |
| 3 | 0 | Параметры отчета о закрытии смены  (битовая маска) | |  |  |  | | --- | --- | --- | | 0 | 0 | Не печатать сумму нарастающего итога на начало смены (\*) | | 1 | Печатать сумму нарастающего итога на начало смены | | 1 | 0 | Не печатать суммы нарастающего итога | | 1 | Печатать суммы нарастающего итога(\*) | | 2 | 0 | Не печатать информацию об отложенных чеках(\*) | | 1 | Печатать информацию об отложенных чеках | | 3 | 0 | Зарезервировано | | 4 | 0 | Не печатать информацию об операциях с денежным ящиком | | 1 | Печатать информацию об операциях с денежным ящиком (строки: «НА НАЧАЛО В КАССЕ», «СУММА В КАССЕ», «ВНЕСЕНИЕ», «ИЗЪЯТИЕ») (\*) | | 5 | 0 | Печатать информацию по неиспользованным за смену платежным средствам(\*) | | 1 | Не печатать информацию по неиспользованным за смену платежным средствам | | 6 | 0 | Не печатать дату и время начала смены(\*) | | 1 | Печатать дату и время начала смены | | 7 | 0 | Не печатать секции на отчете(\*) | | 1 | Печатать секции на отчете | |
| 4 | 0 | Управление внешними устройствами  (битовая маска) | |  |  |  | | --- | --- | --- | | 0 | 0 | Денежный ящик открывает внешняя программа (\*) | | 1 | Денежный ящик открывает ККТ при работе с наличными | |
| 5 | 0 | Управление расчетами  (битовая маска) | |  |  |  | | --- | --- | --- | | 0 | 0 | Включен контроль наличных в денежном ящике. (\*) | | 1 | Контроль наличных в денежном ящике отключен. При этом сумма в денежном ящике может быть отрицательной | | 1 | 0 | Учитывать чеки, аннулированные при включении питания(\*) | | 1 | Не учитывать чеки, аннулированные при включении питания | | 2 | 0 | Автоматическая инкассация выключена(\*) | | 1 | Автоматическая инкассация включена.  При этом перед печатью отчета о закрытии смены печатается чек инкассации на всю сумму наличных в денежном ящике | | 3 | 0 | Счетчики покупок(расходов) выключены(\*) | | 1 | Счетчики покупок(расходов) включены. | | 4 | 0 | Автоматическая печать СКЛ выключена(\*) | | 1 | Автоматическая печать СКЛ включена.  При этом перед печатью отчета о закрытии смены печатается СКЛ за текущую смену | | 5 | 0 | СКЛ отключена(\*) | | 1 | СКЛ включена. Внимание, после включения СКЛ, требуется перезагрузка и в случае использования не новой SD карты ее очистка. | | 6 | 0 | Печать суммы нарастающего итога продаж(приходов)/покупок(расходов) на X-отчетах и отчетах о закрытии смены отключена(\*) | | 1 | Печать суммы нарастающего итога продаж(приходов)/покупок(расходов) на X-отчетах и отчетах о закрытии смены включена | | 7 | 0 | Печать суммы нарастающего итога возвратов (прихода и расхода) на X-отчетах и отчетах о закрытии смены отключена(\*) | | 1 | Печать суммы нарастающего итога возвратов (прихода и расхода) на X-отчетах и отчетах о закрытии смены х включена | |
| 6 | 0 | Управление расчетами и печатью налогов  (битовая маска) | |  |  |  | | --- | --- | --- | | 0 | 0 | Не печатать налоги на отчетах(\*) | | 1 | Печатать налоги на отчетах | | 1 | 0 | Не печатать налоги на чеках(\*) | | 1 | Печатать налоги на чеках | | 2 | 0 | Не печатать нулевые налоговые суммы  на отчетах(\*) | | 1 | Печатать нулевые налоговые суммы  на отчетах | | 3 | 0 | Округлять сумму налога после каждой позиции (\*) | | 1 | Округлять сумму налога только после ввода всех позиций и скидок | |
| 10 | 0 | Логический номер ККТ  (число 1..9999) | Логический номер кассы, присвоенный ей системой данного торгового предприятия |
| 11 | 0 | Доп. ячейка  (число 0..65535) | Ячейка таблицы, которую внешнее ПО может использовать для своих целей |
| 12 | 1..3 | Настройки ПУ | 1 - коррекция яркости печати (20..100%)  2 - коррекция частичной отрезки для бумаги 80мм (0…100)  3 - коррекция частичной отрезки для бумаги 57мм (0…100) |
| 20 | 0 | Пароль для связи  (строка 4 символа) |  |
| 30 | 0..1 | Наименование организации (массив из 2-х строк) | Строки, длиной 0..44 символов, печатаемые в начале каждого чека. Пустые строки не печатаются. |
| 2..3 | Адрес организации (массив из 2-х строк) | Строки, длиной 0..44 символов, печатаемые после наименования организации. Пустые строки не печатаются. |
| 31 | 0..4 | Строки окончания чеков  (массив из 5-х строк) | Строки, длиной 0..44 символов, печатаемые в конце каждого чека. Пустые строки не печатаются. |
| 32 | 0..15 | Названия типов платежей  (массив из 16-х строк) | Строка длиной 0..18 символов. Если строка пустая – платеж не может быть использован для расчетов.  Платеж с индексом 0 -зарезервирован типом “Наличные” (только чтение).  Платеж с индексом 13 - зарезервирован типом  "Предоплата" (только чтение)  Платеж с индексом 14 - зарезервирован типом  "Постоплата" (только чтение)  Платеж с индексом 15 - зарезервирован типом  "Иная оплата" (только чтение) |
| 40 | 0..9 | Название ставки налога  (массив из 10 строк) | Строка длиной 0..18 символов. Если строка пустая – налог не может быть использован для расчетов. |
| 41 | 0..9 | Процент ставки налога  (массив из 10 дробных чисел) | Установленные ставки:  0 - Ставка "НДС 20%"  1 - Ставка "НДС 10%"  2 - Ставка "НДС 0%"  3 - Ставка "без НДС"  4 - Ставка "НДС 20/120"  5 - Ставка "НДС 10/110"  6 - Ставка "НДС 5%"  7 - Ставка "НДС 7%"  8 - Ставка "НДС 5/105"  9 - Ставка "НДС 7/107" |
| 42 | 0 | Строка названия налоговой группы в чеке  (строка) | Строка длиной 0..44 символов. |
| 50 | 1..16 | Наименование отдела/секции (массив из 16-х строк) | Строка длиной 0..44 символов. Если строка пустая – отдел/секция не может быть использована для расчетов. |
| 51 | 0 | Строка названия группы отделов/секции на отчете о закрытии  (строка) | Строка длиной 0..44 символов. Если строка пустая – название раздела не печатается. |
| 52 | 1..5 | Наименование реквизита (массив из 5-х строк) | Строка длиной 0..30 символов. Если строка менее 3-х символов – реквизит не может быть использована для печати. |
| 54 | 0..1 | Реквизиты ЦТО (массив из 2-х строк) | Строка длиной 0..44 символов. |
| 70 | 0 | Номер автомата | Строка длиной 0..12 символов. |
| 71 | 0 | ИНН ОФД | Строка длиной 10..12 символов. |
| 72 | 0 | Содержание QR-кода | Строка длиной 20..160 символов. |
| 73 | 0 | IP-адрес ККТ | Строка из 4-х октетов через точку(\*\*) |
| 74 | 0 | Маска подсети | Строка из 4-х октетов через точку(\*\*) |
| 75 | 0 | IP- адрес шлюза | Строка из 4-х октетов через точку(\*\*) |
| 76 | 0 | IP- адрес DNS | Строка из 4-х октетов через точку(\*\*) |
| 77 | 0 | Адрес сервера ОФД для отправки документов | Строка 0..64 символа |
| 78 | 0 | Порт сервера ОФД | Число 1..65534 |
| 79 | 0 | Таймер ФН | Число 0..60 сек (30\*) |
| 80 | 0 | Таймер С | Число 0..3600 сек (60\*) |
| 81 | 0 | Наименование ОФД | Строка 0..64 символа |
| 82 | 0 | Электронная почта отправителя чека | Строка 0..64 символа |
| 83 | 0 | URL сайта ФНС | Строка 0..64 символа |

(\*) – Значения параметров маска по умолчанию. Для строковых параметров по умолчанию используется пустая строка, кроме параметров: 32- Названия типов платежей и 20- Пароль для связи.

(\*\*) - При изменении этих параметров необходима перезагрузка ККТ.

**Внимание !** Запись параметров конфигурации 41 и 52, выполняется только при закрытой смене.

После изменения параметров конфигурации 1 и/или 30, рекомендуется выполнять функцию ”Начало работы”, для перенастройки печатающего устройства и выполнения правильной предпечати следующего документа.

**5. Список кодов ошибок**

|  |  |  |
| --- | --- | --- |
| Код, возвращаемый в ответе ККТ | | Описание |
| Decimal | Hex |
| 0 | 00h | Команда выполнена без ошибок |
| **Ошибки выполнения команд** | | |
| 1 | 01h | Функция невыполнима при данном статусе ККТ |
| 2 | 02h | В команде указан неверный номер функции |
| 3 | 03h | Некорректный формат или параметр команды |
| **Ошибки протокола передачи данных** | | |
| 4 | 04h | Переполнение буфера коммуникационного порта |
| 5 | 05h | Таймаут при передаче байта информации |
| 6 | 06h | В протоколе указан неверный пароль |
| 7 | 07h | Ошибка контрольной суммы в команде |
| **Ошибки печатающего устройства** | | |
| 8 | 08h | Конец бумаги |
| 9 | 09h | Принтер не готов |
| **Ошибки даты/времени** | | |
| 10 | 0Ah | Текущая смена больше 24 часов.  Установка даты времени больше чем на 24 часа. |
| 11 | 0Bh | Разница во времени, ККТ и указанной в команде начала работы, больше 8 минут |
| 12 | 0Ch | Вводимая дата более ранняя, чем дата последней фискальной операции |
| **Прочие ошибки** | | |
| 14 | 0Eh | Отрицательный результат |
| 15 | 0Fh | Для выполнения команды необходимо закрыть смену |
| 16 | 10h | Нет данных в журнале |
| 17 | 11h | Ошибка при работе с SD-картой |
| 18 | 12h | Ошибка посылки данных в ОФД |
| **Фатальные ошибки** | | |
| 32 | 20h | Фатальная ошибка ККТ.  Причины возникновения данной ошибки можно уточнить в ”Статусе фатальных ошибок ККТ” |
| **Ошибки ФН** | | |
| 65 | 41h | Некорректный формат или параметр команды ФН |
| 66 | 42h | Некорректное состояние ФН |
| 67 | 43h | Ошибка ФН |
| 68 | 44h | Ошибка КС (Криптографического сопроцессора) в составе ФН |
| 69 | 45h | Исчерпан временной ресурс использования ФН |
| 70 | 46h | ФН переполнен |
| 71 | 47h | Неверные дата или время |
| 72 | 48h | Нет запрошенных данных |
| 73 | 49h | Некорректные параметры команды |
| 80 | 50h | Превышен размер данных TLV |
| 81 | 51h | Нет транспортного соединения |
| 82 | 52h | Исчерпан ресурс КС |
| 84 | 54h | Исчерпана память хранения документов для ОФД |
| 85 | 55h | Время нахождения в очереди самого старого сообщения на выдачу более 30 календарных дней. |
| 86 | 56h | Продолжительность смены ФН более 24 часов |
| 87 | 57h | Разница более чем на 5 минут отличается от разницы, определенной по внутреннему таймеру ФН. |
| 96 | 60h | Неверное сообщение от ОФД |
| 97 | 61h | Нет связи с ФН |
| 98 | 62h | Ошибка обмена с ФН |
| 99 | 63h | Слишком длинная команда для посылки в ФН |

Дополнение для ошибок ФН:

Ошибки 41, 49h, 50h, 51h, 56h, 57h и 63h при корректной работе возникать не должны.

При возникновении ошибок 43h, 44h, 45h, 46h, 52h ФН подлежит замене.

Ошибка 47h говорит о том, что установленные в Пирит дата/время не

соответствуют дате/времени, записанным в ФН, следует установить корректные дату и время

в Пирит.

Ошибка 48h ошибкой не является, это сообщение о том, что запрашиваемых данных в ФН

не содержится (например, запрашиваемого документа с заданным ФД в ФН не содержится).

Следует запросить команду с другими параметрами.

В случае ошибок 54h и 55h следует освободить память документов для ОФД, обеспечив передачу

неподтвержденных документов в ОФД. Обычно она возникает при длительном отсутствии связи с ОФД.

В случае ошибки 61h и 62h следует проверить подключение ФН к Пирит.

Ошибка 42h может возникать при рассогласовании состояний ФН и Пирит, например, при незакрытом

документе в Пирит и ФН. Следует сравнить состояние Пирит с использованием команд запроса 0x00, 0x01, 0x02 и 0x03

с состоянием ФН запросами команды 0x78. В случае незакрытого документа или смены в Пирит следует еще раз исполнить

соотв. команды.

При возникновении ошибок 42h, 43h и 44h может помочь перезагрузка Пирита по питанию.

***5.1 Список команд, которые могут вернуть код ошибки “Функция невыполнима при данном статусе ККТ ” (0x01)***

|  |  |  |
| --- | --- | --- |
| **Код команды**  **(HEX)** | **Название команды** | **Список условий, при которых возвращается ошибка 0x01** |
| **20** | Сформировать отчет без гашения (X-отчет) | -Открыт документ;  -Не было команды “начало работы”;  - В фискальном режиме установлен зарегистрированный ФН;  - ФН закрыт; |
| **21** | Сформировать отчет о закрытии смены |
| **30** | Открыть документ |
| **31** | Завершить документ | -Состояние документа не равно 4 (Расчет завершен – требуется закрыть документ);  -Документ не открыт в ФН. |
| **32** | Аннулировать документ | -Состояние документа равно 8 (Команда закрытия документа была дана в ФН). |
| **33** | Отложить документ |
| **40** | Печать текста | -Тип текущего открытого документа не равен 1 (Открыт сервисный документ). |
| **41** | Печатать штрих-код | -Документ не открыт. |
| **42** | Добавить товарную позицию | -Тип текущего открытого документа не равен 2 или 3 ( продажа или возврат);  -Состояние документа не равно 1 (ввод товарных позиций). |
| **43** | Сторнировать товарную позицию |
| **44** | Подытог | -Тип текущего открытого документа не равен 2 или 3 (продажа или возврат);  -Состояние документа не равно 1 или 2. |
| **45** | Скидка на товар/чек |
| **46** | Наценка на товар/чек |
| **47** | Оплата | -Тип текущего открытого документа не равен 2 или 3 (продажа или возврат);  -Состояние документа не равно 1, 2 или 3. |
| **48** | Внесение / изъятие суммы | -Тип текущего открытого документа не равен 4 или 5 (внесение или изъятие). |
| **49** | Печать реквизита | -Документ не открыт;  -Тип текущего открытого документа равен 1 (Открыт сервисный документ). |
| **51** | Зарегистрировать сумму налога | -Документ не открыт. |
| **52** | Сравнить сумму по чеку | -Документ не открыт. |
| **53** | Открыть копию чека | -Открыт документ;  -Не было команды “начало работы”;  -Незарегистрированный ФН;  -ФН закрыт; |
| **60** | Регистрация/Перерегистрация | -Открыт документ;  -Не было команды “начало работы”;  -ФН уже зарегистрирован; |
| **71** | Закрытие ФН | -Не было команды “начало работы”;  -Нефискальный режим; |
| **73** | Распечатать документ из ФН |
| **76** | Распечатать отчет о регистрации |  |
| **78** | Запрос информации из ФН | -Нефискальный режим. |
| **90** | Авторизация ККТ | -ККТ уже авторизована. |
| **93** | Установить скорость обмена | -Фатальная ошибка ФН. |

**6. Таблица флагов состояния ККТ, необходимых для успешного выполнения операции.**

|  |
| --- |
| **Тип операции** |
|  | **Неверная контрольная сумма NVR** | **Неверная контрольная сумма в конфигурации** | **Фатальная ошибка ФН** | **Расхождение данных ККТ и ФН** | **Не была вызвана функция “Начало работы”** | **Нефискальный режим** | **Смена открыта** | **Смена больше 24 часов** | **ФН закрыт** | **ФН не активирована** | **Не было завершено закрытие смены, необходимо повторить операцию** |  |
| Регистрация | **-** | - | - | - | - | + | - | - | - | + | - | 0 |
| Перерегистрация | **-** | - | - | - | - | - | - | - | - | + | - | 0 |
| Закрытие ФН | X | X | X | X | - | X | X | X | X | X | - | 0 |
| Печать КЛ | X | X | - | X | - | - | X | X | X | - | - | 0 |
| Печать д-та из ФН | X | X | - | X | - | - | X | X | X | - | - | 0 |
| Запись настроек в ККТ (не реквизиты) | - | X | X | X | X | X | X | X | X | X | X | X |
| Запись настроек в ККТ (реквизиты) | - | X | X | X | X | X | - | - | X | X | X | X |
| Авторизация (ввод заводского номера) ККТ | X | X | X | X | X | X | X | X | X | X | X | X |
| Установка даты/времени | X | X | X | X | X | X | - | X | X | X | X | X |
| Проверка исправности с пробитием чеков и закрытием смены  Выполнение тестовой задачи (пробитие чеков, возврат, закрытие смены) | - | - | - | - | - | X | - | - | - | - | - | 0 |

'+’ – флаг должен быть установлен

'-’ – флаг должен быть сброшен

'X’ – не влияет

'0' – документ должен быть закрыт

**7. Примеры**

* **Регистрация чека продажи(возврата)**

int errorCode; // Код возвращаемой ошибки

int fatalStatus; // Статус фатального состояния ККТ

int currentFlagsStatus; // Статус текущих флагов ККТ

int documentStatus; // Статус документа

errorCode = *getStatusFlags*(&fatalStatus, &currentFlagsStatus, &documentStatus); // Запрос флагов статуса ККТ

if(errorCode > 0) || (fatalStatus > 0) || (currentFlagsStatus > 0) ||(documentStatus > 0){

провести анализ состояния и выполнить необходимые действия

}

Для печати изображения внутри чека необходимо сначала загрузить данное изображение в ККТ:

int widthPicture = 512; // Ширина изображения в пикселях (максимально 512)

int heightPicture = 100; // Высота изображения в пикселях

int alignment; // Выравнивание картинки (0 – левый край, 1 – по центру, 2 – правый край)

unsigned char \*data; // массив точек 1 бит – одна точка (Максимальный размер – 25088 байт или 200704 точки)

errorCode = *libLoadPicture*(widthPicture, heightPicture, alignment, "Picture Name", 1, data); //Загрузить графическое изображение

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

int docType = 2; // Режим и тип документа (2-продажа, 3-возврат)

int numDepart = 1; // Номер отдела (1..99)

int docNumber = 333; // Номер документа

errorCode = *libOpenDocument*(docType, numDepart, "OPEXOB A.A.", docNumber); // Открыть документ

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Далее необходимо добавить товарную позицию:

long quantity = 2000; // Количество в граммах или в 1/1000 штук (2000 – это 2 штуки или 2 кг)

long price = 15782; // Цена в копейках (15782 – это 157 рублей 82 копейки)

unsigned char taxNumber = 1; // Номер ставки налога (0..5)

unsigned char numDepart = 1; // Номер секции (1..16)

errorCode = *libAddPosition*("TOBAP N:1 KPEM 'ABCDEFGH'", "9785845913784", quantity, price, taxNumber, 0, numDepart);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Также можно сторнировать позицию:

errorCode = *libDelPosition*("TOBAP N:1 KPEM 'ABCDEFGH'", "9785845913784", quantity, price, taxNumber, 0, numDepart);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Можно добавить скидки или наценки на конкретную позицию:

long sumOrPercent = 700; // Сумма скидки в копейках или сотых процента (700 – это 7 рублей или 7%)

unsigned char typeDiscount = 0; // Тип скидки (0-процент, 1-сумма)

errorCode = *libAddDiscount*(typeDiscount, "Discount 7 %", sumOrPercent); // Скидка

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

errorCode = *libAddMargin*(typeDiscount, "Margin 7 %", sumOrPercent); // Наценка

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Далее необходимо можно вызвать команду подытога

errorCode = *libSubTotal*(); // Подытог

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

После первой команды подытог можно задать скидку/наценку на весь чек той же командой.

В любом месте чека мы можем напечатать загруженное изображение:

int alignment = 0;// Выравнивание картинки (0 – левый край, 1 – по центру, 2 – правый край)

errorCode = *libPrintPreloadedPicture*(alignment, 1);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

После окончания добавления позиций, скидок и наценок в чек необходимо добавить оплаты:

int alignment = 0;// Выравнивание картинки (0 – левый край, 1 – по центру, 2 – правый край)

unsigned char typePayment = 0;// тип платежа (0-наличные, 1..15 запрограммированные типы оплат)

long sum = 100000;// Сумма в копейках (100000 – это 1000 рублей)

errorCode = *libAddPayment*(type, sum, “Наличные”);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

В чеке так же можно распечатать штрих-код:

unsigned char posText = 3; //Вывод текста (0-не выводить; 1-вывести наверху штрих-кода; 2-вывести внизу; 3-вывести наверху и внизу;)

unsigned char widthBarCode = 2;// Ширина линии штрих-кода в точках (2..8)

unsigned char heightBarCode = 40; //Высота штрих-кода (1..255)

unsigned char typeBarCode = 2; //Тип штрих-кода (0..8)

errorCode = *libPrintBarCode*(posText, widthBarCode, heightBarCode, typeBarCode, “4712759211502”);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Далее можно проверить сумму чека

long sum = 57012; // Сумма чека в копейках (57012 – это 570 рублей 12 копеек)

errorCode = *libCompareSum*(sum);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Для фискализации документа его необходимо закрыть

unsigned char cutPaper = 0; // флаг отреза(0 – отрезать, 1 – не отрезать)

errorCode = *libCloseDocument*(sum);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

* **Внесение/Изъятие**

int docType = 4; // Режим и тип документа (4-внесение, 5-изъятие)

int numDepart = 2; // Номер отдела (1..99)

int docNumber = 334; // Номер документа

errorCode = *libOpenDocument*(docType, numDepart, "OPEXOB A.A.", docNumber); // Открыть документ

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

В документ можно добавить текст:

unsigned char attributeText = 1; // Тип и атрибуты шрифта

errorCode = *libPrintRequsit*(0, attributeText, "Текст для добавления", “”, “”, “”);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

long sum = 70045; // Сумма внесения/изъятия в копейках (70045 – это 700 рублей 45 копеек)

errorCode = *libCashInOut*("Дополнительная информация", sum);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Для завершения документа его необходимо закрыть

unsigned char cutPaper = 0; // флаг отреза(0 – отрезать, 1 – не отрезать)

errorCode = *libCloseDocument*(sum);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

* **X и Z отчёты**

Для снятия X-отчёта необходимо вызвать команду

errorCode = *libPrintXReport*(“Иванова А.С”);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}

Для снятия Z-отчёта необходимо вызвать команду

Unsigned char options = 0;// Тип отчёта (0 – полный, 11 - краткий)

errorCode = *libPrintZReport*(“Петрова Т.В”, options);

if(errorCode > 0){

провести анализ состояния и выполнить необходимые действия

}