Версия 1.13_356

ШТРИХ-М: JPOS драйвер

Руководство пользователя

«ШТРИХ-М», 2017

Версия: 1.2
Дата: 31.01.2017

Добавлено описание DIO-комманд установки email и телефона при работе с ФН.

Версия: 1.1
Дата: 05.09.2016

Вступление

Данная документация описывает особенности реализации драйвера и не является руководством по стандарту UPOS. (Со стандартом UPOS можно ознакомиться здесь).

Методы

```
public interface BaseService
{
    // Properties
    public String getCheckHealthText() throws JposException;
    public boolean getClaimed() throws JposException;
    public boolean getDeviceEnabled() throws JposException;
    public void    setDeviceEnabled(boolean deviceEnabled)
                       throws JposException;
    public String  getDeviceServiceDescription() throws JposException;
    public int     getDeviceServiceVersion() throws JposException;
    public boolean getFreezeEvents() throws JposException;
    public void    setFreezeEvents(boolean freezeEvents) throws JposException;
    public String  getPhysicalDeviceDescription() throws JposException;
    public String  getPhysicalDeviceName() throws JposException;
    public int     getState() throws JposException;

    // Methods
    public void    claim(int timeout) throws JposException;
    public void    close() throws JposException;
    public void    checkHealth(int level) throws JposException;
    public void    directIO(int command, int[] data, Object object)
                       throws JposException;
    public void    open(String logicalName, EventCallbacks cb)
                        throws JposException;
    public void    release() throws JposException;


}
public interface FiscalPrinterService13 extends BaseService
{
    // Capabilities
    public boolean getCapAdditionalLines() throws JposException;
    public boolean getCapAmountAdjustment() throws JposException;
    public boolean getCapAmountNotPaid() throws JposException;
    public boolean getCapCheckTotal() throws JposException;
    public boolean getCapCoverSensor() throws JposException;
    public boolean getCapDoubleWidth() throws JposException;
    public boolean getCapDuplicateReceipt() throws JposException;
    public void    setDuplicateReceipt(boolean duplicateReceipt) throws
JposException;
    public boolean getCapFixedOutput() throws JposException;
    public boolean getCapHasVatTable() throws JposException;
    public boolean getCapIndependentHeader() throws JposException;
    public boolean getCapItemList() throws JposException;
    public boolean getCapJrnEmptySensor() throws JposException;
    public boolean getCapJrnNearEndSensor() throws JposException;
    public boolean getCapJrnPresent() throws JposException;
    public boolean getCapNonFiscalMode() throws JposException;
    public boolean getCapOrderAdjustmentFirst() throws JposException;
    public boolean getCapPercentAdjustment() throws JposException;
    public boolean getCapPositiveAdjustment() throws JposException;
    public boolean getCapPowerLossReport() throws JposException;
    public int     getCapPowerReporting() throws JposException;
```

```
public boolean getCapPredefinedPaymentLines() throws JposException;
public boolean getCapReceiptNotPaid() throws JposException;
public boolean getCapRecEmptySensor() throws JposException;
public boolean getCapRecNearEndSensor() throws JposException;
public boolean getCapRecPresent() throws JposException;
public boolean getCapRemainingFiscalMemory() throws JposException;
public boolean getCapReservedWord() throws JposException;
public boolean getCapSetHeader() throws JposException;
public boolean getCapSetPOSID() throws JposException;
public boolean getCapSetStoreFiscalID() throws JposException;
public boolean getCapSetTrailer() throws JposException;
public boolean getCapSetVatTable() throws JposException;
public boolean getCapSlpEmptySensor() throws JposException;
public boolean getCapSlpFiscalDocument() throws JposException;
public boolean getCapSlpFullSlip() throws JposException;
public boolean getCapSlpNearEndSensor() throws JposException;
public boolean getCapSlpPresent() throws JposException;
public boolean getCapSlpValidation() throws JposException;
public boolean getCapSubAmountAdjustment() throws JposException;
public boolean getCapSubPercentAdjustment() throws JposException;
public boolean getCapSubtotal() throws JposException;
public boolean getCapTrainingMode() throws JposException;
public boolean getCapValidateJournal() throws JposException;
public boolean getCapXReport() throws JposException;

// Properties
public int     getOutputID() throws JposException;
public int     getPowerNotify() throws JposException;
public void    setPowerNotify(int powerNotify) throws JposException;
public int     getPowerState() throws JposException;

public int     getAmountDecimalPlace() throws JposException;
public boolean getAsyncMode() throws JposException;
public void    setAsyncMode(boolean asyncMode) throws JposException;
public boolean getCheckTotal() throws JposException;
public void    setCheckTotal(boolean checkTotal) throws JposException;
public int     getCountryCode() throws JposException;
public boolean getCoverOpen() throws JposException;
public boolean getDayOpened() throws JposException;
public int     getDescriptionLength() throws JposException;
public boolean getDuplicateReceipt() throws JposException;
public int     getErrorLevel() throws JposException;
public int     getErrorOutID() throws JposException;
public int     getErrorState() throws JposException;
public int     getErrorStation() throws JposException;
public String  getErrorString() throws JposException;
public boolean getFlagWhenIdle() throws JposException;
public void    setFlagWhenIdle(boolean flagWhenIdle) throws JposException;
public boolean getJrnEmpty() throws JposException;
public boolean getJrnNearEnd() throws JposException;
public int     getMessageLength() throws JposException;
public int     getNumHeaderLines() throws JposException;
public int     getNumTrailerLines() throws JposException;
public int     getNumVatRates() throws JposException;
public String  getPredefinedPaymentLines() throws JposException;
public int     getPrinterState() throws JposException;
public int     getQuantityDecimalPlaces() throws JposException;
public int     getQuantityLength() throws JposException;
public boolean getRecEmpty() throws JposException;
public boolean getRecNearEnd() throws JposException;
public int     getRemainingFiscalMemory() throws JposException;
public String  getReservedWord() throws JposException;
public boolean getSlpEmpty() throws JposException;
```

```
public boolean getSlpNearEnd() throws JposException;
public int     getSlipSelection() throws JposException;
public void    setSlipSelection(int slipSelection) throws JposException;
public boolean getTrainingModeActive() throws JposException;

// Methods
public void    beginFiscalDocument(int documentAmount)
                   throws JposException;
public void    beginFiscalReceipt(boolean printHeader)
                   throws JposException;
public void    beginFixedOutput(int station, int documentType)
                   throws JposException;
public void    beginInsertion(int timeout) throws JposException;
public void    beginItemList(int vatID) throws JposException;
public void    beginNonFiscal() throws JposException;
public void    beginRemoval(int timeout) throws JposException;
public void    beginTraining() throws JposException;
public void    clearError() throws JposException;
public void    clearOutput() throws JposException;
public void    endFiscalDocument() throws JposException;
public void    endFiscalReceipt(boolean printHeader) throws JposException;
public void    endFixedOutput() throws JposException;
public void    endInsertion() throws JposException;
public void    endItemList() throws JposException;
public void    endNonFiscal() throws JposException;
public void    endRemoval() throws JposException;
public void    endTraining() throws JposException;
public void    getData(int dataItem, int[] optArgs, String[] data)
                   throws JposException;
public void    getDate(String[] Date) throws JposException;
public void    getTotalizer(int vatID, int optArgs, String[] data)
                   throws JposException;
public void    getVatEntry(int vatID, int optArgs, int[] vatRate)
                   throws JposException;
public void    printDuplicateReceipt() throws JposException;
public void    printFiscalDocumentLine(String documentLine)
                   throws JposException;
public void    printFixedOutput(int documentType, int lineNumber,
                   String data) throws JposException;
public void    printNormal(int station, String data) throws JposException;
public void    printPeriodicTotalsReport(String date1, String date2)
                   throws JposException;
public void    printPowerLossReport() throws JposException;
public void    printRecItem(String description, long price, int quantity,
                   int vatInfo, long unitPrice, String unitName)
                   throws JposException;
public void    printRecItemAdjustment(int adjustmentType,
                   String description, long amount, int vatInfo)
                   throws JposException;
public void    printRecMessage(String message) throws JposException;
public void    printRecNotPaid(String description, long amount)
                   throws JposException;
public void    printRecRefund(String description, long amount, int vatInfo)
                   throws JposException;
public void    printRecSubtotal(long amount) throws JposException;
public void    printRecSubtotalAdjustment(int adjustmentType,
                   String description, long amount) throws JposException;
public void    printRecTotal(long total, long payment, String description)
                   throws JposException;
public void    printRecVoid(String description) throws JposException;
public void    printRecVoidItem(String description, long amount,
                   int quantity, int adjustmentType, long adjustment,
                   int vatInfo) throws JposException;
```

```
    public void    printReport(int reportType, String startNum, String endNum)
                       throws JposException;
    public void    printXReport() throws JposException;
    public void    printZReport() throws JposException;
    public void    resetPrinter() throws JposException;
    public void    setDate(String date) throws JposException;
    public void    setHeaderLine(int lineNumber, String text,
                       boolean doubleWidth) throws JposException;
    public void    setPOSID(String POSID, String cashierID)
                       throws JposException;
    public void    setStoreFiscalID(String ID) throws JposException;
    public void    setTrailerLine(int lineNumber, String text,
                       boolean doubleWidth) throws JposException;
    public void    setVatTable() throws JposException;
    public void    setVatValue(int vatID, String vatValue)
                       throws JposException;
    public void    verifyItem(String itemName, int vatID) throws JposException;


}
public interface FiscalPrinterService16
  extends FiscalPrinterService15
{
    // Capabilities
    public boolean getCapAdditionalHeader() throws JposException;
    public boolean getCapAdditionalTrailer() throws JposException;
    public boolean getCapChangeDue() throws JposException;
    public boolean getCapEmptyReceiptIsVoidable() throws JposException;
    public boolean getCapFiscalReceiptStation() throws JposException;
    public boolean getCapFiscalReceiptType() throws JposException;
    public boolean getCapMultiContractor() throws JposException;
    public boolean getCapOnlyVoidLastItem() throws JposException;
    public boolean getCapPackageAdjustment() throws JposException;
    public boolean getCapPostPreLine() throws JposException;
    public boolean getCapSetCurrency() throws JposException;
    public boolean getCapTotalizerType() throws JposException;

    // Properties
    public int     getActualCurrency() throws JposException;
    public String  getAdditionalHeader() throws JposException;
    public void    setAdditionalHeader(String additionalHeader)
                       throws JposException;
    public String  getAdditionalTrailer() throws JposException;
    public void    setAdditionalTrailer(String additionalTrailer)
                       throws JposException;
    public String  getChangeDue() throws JposException;
    public void    setChangeDue(String changeDue) throws JposException;
    public int     getContractorId() throws JposException;
    public void    setContractorId(int contractorId) throws JposException;
    public int     getDateType() throws JposException;
    public void    setDateType(int dateType) throws JposException;
    public int     getFiscalReceiptStation() throws JposException;
    public void    setFiscalReceiptStation(int fiscalReceiptStation)
                       throws JposException;
    public int     getFiscalReceiptType() throws JposException;
    public void    setFiscalReceiptType(int fiscalReceiptType)
                       throws JposException;
    public int     getMessageType() throws JposException;
    public void    setMessageType(int messageType) throws JposException;
    public String  getPostLine() throws JposException;
    public void    setPostLine(String postLine) throws JposException;
    public String  getPreLine() throws JposException;
    public void    setPreLine(String preLine) throws JposException;
```

```
    public int      getTotalizerType() throws JposException;
    public void     setTotalizerType(int totalizerType) throws JposException;

    // Methods
    public void     setCurrency(int newCurrency) throws JposException;
    public void     printRecCash(long amount) throws JposException;
    public void     printRecItemFuel(String description, long price,
                        int quantity, int vatInfo, long unitPrice,
                        String unitName, long specialTax, String specialTaxName)
                        throws JposException;
    public void     printRecItemFuelVoid(String description, long price,
                        int vatInfo, long specialTax) throws JposException;
    public void     printRecPackageAdjustment(int adjustmentType,
                        String description, String vatAdjustment)
                        throws JposException;
    public void     printRecPackageAdjustVoid(int adjustmentType,
                        String vatAdjustment) throws JposException;
    public void     printRecRefundVoid(String description, long amount,
                        int vatInfo) throws JposException;
    public void     printRecSubtotalAdjustVoid(int adjustmentType, long amount)
                        throws JposException;
    public void     printRecTaxID(String taxID) throws JposException;
}

public interface FiscalPrinterService17
  extends FiscalPrinterService16
{
      // The AmountDecimalPlaces property was incorrectly spelled
      // AmountDecimalPlace since version 1.3.  In version 1.7.2 and later,
      // the correct spelling is supported. The old version is left for
      // Application and Device Service compatibility. The implementations
      // of getAmountDecimalPlaces and getAmountDecimalPlace should be
      // identical.

      // Properties
      public int      getAmountDecimalPlaces() throws JposException;


}
public interface FiscalPrinterService18
  extends FiscalPrinterService17
{
  // Capabilities
  public boolean getCapStatisticsReporting() throws JposException;
  public boolean getCapUpdateStatistics() throws JposException;

  // Methods
  public void     resetStatistics(String statisticsBuffer)
                      throws JposException;
  public void     retrieveStatistics(String[] statisticsBuffer)
                      throws JposException;
  public void     updateStatistics(String statisticsBuffer)
                      throws JposException;

}
public interface FiscalPrinterService19
  extends FiscalPrinterService18
{
  // Capabilities
  public boolean getCapCompareFirmwareVersion() throws JposException;
  public boolean getCapUpdateFirmware() throws JposException;
```

```
  // Methods
  public void    compareFirmwareVersion(String firmwareFileName, int[] result)
                     throws JposException;
  public void    updateFirmware(String firmwareFileName)
                     throws JposException;
}
public interface FiscalPrinterService111 extends FiscalPrinterService110
{
  // Capabilities
  public boolean getCapPositiveSubtotalAdjustment() throws JposException;

  // Methods
  public void    printRecItemVoid(String description,
                                  long price,
                                  int quantity,
                                  int vatInfo,
                                  long unitPrice,
                                  String unitName)
                     throws JposException;
  public void    printRecItemAdjustmentVoid(int adjustmentType,
                                            String description,
                                            long amount,
                                            int vatInfo)
                     throws JposException;
}
public interface FiscalPrinterService112 extends FiscalPrinterService111
{
  // Methods
  public void    printRecItemRefund(String description,
                                    long amount,
                                    int quantity,
                                    int vatInfo,
                                    long unitAmount,
                                    String unitName)
                     throws JposException;
  public void    printRecItemRefundVoid(String description,
                                        long amount,
                                        int quantity,
                                        int vatInfo,
                                        long unitAmount,
                                        String unitName)
                     throws JposException;
}
```

Свойства

```
public String getCheckHealthText() throws JposException;
```

Свойство содержит результат вызова метода `checkHealth()` в текстовом виде.

```
public boolean getClaimed() throws JposException;
```

Свойство имеет значение True после успешного вызова метода Claim, False после вызова Release.

```
public boolean getDeviceEnabled() throws JposException;
```

Свойство имеет значение True после успешного вызова метода setDeviceEnabled.

```
public void setDeviceEnabled(boolean deviceEnabled) throws JposException;
```

Метод выполняет подключение и инициализацию устройства. Если связаться с устройством с заданными параметрами не удается, драйвер может выполнить поиск устройства и запрограммировать в нем заданные

параметры связи (это зависит от параметров searchByPortEnabled и searchByBaudRateEnabled). По умолчанию разрешен поиск на всех скоростях. При инициализации устройства драйвер может записывать данные таб лиц ФР из файла (см. параметры fieldsFileName, fieldsFilesPath).

```
public String getDeviceServiceDescription() throws JposException;
```
Возвращает "Fiscal Printer Service , SHTRIH-M, 2016"

```
public int getDeviceServiceVersion() throws JposException;
```
Возвращает 1013273

```
public boolean getFreezeEvents() throws JposException;
```
Возвращает свойство FreezeEvents, true – доставка событий запрещена, false – доставка событий разрешена.

```
public void setFreezeEvents(boolean freezeEvents) throws JposException;
```
Разрешает или запрещает доставку событий.

```
public String getPhysicalDeviceDescription() throws JposException;
```
Возвращает строку вида "ШТРИХ-МИНИ-ФР-К, 12345678, ПО ФР: А4.12345, 01.01.2012, ПО ФП: 1.35.23456, 02.02.2012"

```
public String  getPhysicalDeviceName() throws JposException;
```
Возвращает строку вида "ШТРИХ-МИНИ-ФР-К, №12345678", то есть название устройства и его серийный номер

```
public int getState() throws JposException;
```
Текущее состояние устройства. Допустимые значения:

JPOS_S_CLOSED	Устройство закрыто (не был вызван метод Open)
JPOS_S_IDLE	Устройство находится в рабочем состоянии и не занято выполнением операций.

```
public boolean getCapAdditionalLines() throws JposException;
```
Возвращает true, метод PrintRecMessage может печатать произвольный текст после печати итога.

```
public boolean getCapAmountAdjustment() throws JposException;
```
Возвращает true, метод PrintRecItemAdjustment поддерживает скидку в денежных единицах.

```
public boolean getCapAmountNotPaid() throws JposException;
```
Возвращает false.

```
public boolean getCapCheckTotal() throws JposException;
```
Возвращает true, проверка итога в методе PrintRecTotal может производиться.

```
public boolean getCapCoverSensor() throws JposException;
```
Возвращает true, если устройство имеет датчик крышки корпуса. Значение свойства зависит подключенной модели ФР.

```
public boolean getCapDoubleWidth() throws JposException;
```
Возвращает true, все модели ФР ШТРИХ-М имеют шрифт двойной ширины (шрифт 2 – шрифт удвоенной высоты и ширины, обычно им печатается слово ИТОГ на чеке).

```
public boolean getCapDuplicateReceipt() throws JposException;
```
Возвращает true, все модели ФР ШТРИХ-М могут распечатать копию последнего чека.

```
public void setDuplicateReceipt(boolean duplicateReceipt) throws JposException;
```

Устанавливает свойство DuplicateReceipt. Ничего не делает в драйвере, так как копия данных чека есть в ФР.

```
public boolean getCapFixedOutput() throws JposException;
```
Возвращает false, драйвер не поддерживает печать методами BeginFixedOutput, PrintFixedOutput и EndFixedOutput.

```
public boolean getCapHasVatTable() throws JposException;
```
Возвращает true, все модели ФР поддерживают таблицу налоговых ставок.

```
public boolean getCapIndependentHeader() throws JposException;
```
Возвращает false, драйвер автоматически печатает клише следующего чека после предыдущего.

```
public boolean getCapItemList() throws JposException;
```
Возвращает false, ФР не печатает список товаров в отчете по налогам.

```
public boolean getCapJrnEmptySensor() throws JposException;
```
Значение зависит от модели ФР.

```
public boolean getCapJrnNearEndSensor() throws JposException;
```
Значение зависит от модели ФР.

```
public boolean getCapJrnPresent() throws JposException;
```
Значение зависит от модели ФР.

```
public boolean getCapNonFiscalMode() throws JposException;
```
Возвращает true, драйвер поддерживает нефискальный режим.

```
public boolean getCapOrderAdjustmentFirst() throws JposException;
```
Возвращает false, метод PrintRecItemAdjustment должен вызываться после PrintRecItem.

```
public boolean getCapPercentAdjustment() throws JposException;
```
Возвращает true, в методе PrintRecItemAdjustment можно задавать значение скидки/надбавки в процентах.

```
public boolean getCapPositiveAdjustment() throws JposException;
```
Возвращает true, драйвер позволяет делать надбавки на позицию с помощью метода PrintRecItemAdjustment.

```
public boolean getCapPowerLossReport() throws JposException;
```
Возвращает false, фискальные регистраторы ШТРИХ-М не поддерживают печать отчета об отключении питания.

```
public int getCapPowerReporting() throws JposException;
```
Возвращает JPOS_PR_STANDARD, драйвер может определять два состояния ФР, OFF_OFFLINE (отключен или выключен) и ONLINE (подключен).

```
public boolean getCapPredefinedPaymentLines() throws JposException;
```
Возвращает true, драйвер поддерживает названия платежей, список которых определен в свойстве PredefinedPaymentLines.

```
public boolean getCapReceiptNotPaid() throws JposException;
```
Возвращает false, метод PrintRecNotPaid нельзя использовать для печати неоплаченной части итога чека.

```
public boolean getCapRecEmptySensor() throws JposException;
```
Значение зависит от модели ФР.

```
public boolean getCapRecNearEndSensor() throws JposException;
```
Значение зависит от модели ФР.

```
public boolean getCapRecPresent() throws JposException;
```
Возвращает true, на всех ФР есть чековая станция.

```
public boolean getCapRemainingFiscalMemory() throws JposException;
```
Возвращает true, ФР может сообщить количество оставшихся записей в ФП (количество оставшихся смен).

```
public boolean getCapReservedWord() throws JposException;
```
Возвращает false, в ФР и драйвере нет зарезервированных слов.

```
public boolean getCapSetHeader() throws JposException;
```
Возвращает true, драйвер позволяет устанавливать заголовок чека.

```
public boolean getCapSetPOSID() throws JposException;
```
Возвращает true, драйвер позволяет устанавливать номер ФР в магазине .

```
public boolean getCapSetStoreFiscalID() throws JposException;
```
Возвращает false, драйвер не позволяет устанавливать ИНН ФР.

```
public boolean getCapSetTrailer() throws JposException;
```
Возвращает true, драйвер позволяет устанавливать подвал чека.

```
public boolean getCapSetVatTable() throws JposException;
```
Возвращает true, драйвер позволяет устанавливать налоговые ставки.

```
public boolean getCapSlpEmptySensor() throws JposException;
```
Поддерживает ли ФР датчик наличия подкладного документа. Возвращаемое значение зависит от модели ФР.

```
public boolean getCapSlpFiscalDocument() throws JposException;
```
Поддеживает ли ФР фискальную печать на подкладном документе.
Возвращает false, так как драйвер не поддерживает фискальную печать на подкладном документе, хотя сам ФР поддерживает.

```
public boolean getCapSlpFullSlip() throws JposException;
```
Возвращает false, драйвер не поддерживает полноразмерные подкладные документы.

```
public boolean getCapSlpNearEndSensor() throws JposException;
```
Поддерживает ли ФР датчик конца подкладного документа. Возвращаемое значение зависит от модели ФР.

```
public boolean getCapSlpPresent() throws JposException;
```
Поддерживает ли ФР печать подкладных документов. Возвращаемое значение зависит от модели ФР.

```
public boolean getCapSlpValidation() throws JposException;
```
Возвращает false, драйвер не поддерживает печать проверочной информации на подкладном документе.

```
public boolean getCapSubAmountAdjustment() throws JposException;
Возвращает true, фискальный регистратор поддерживает скидки суммой.

public boolean getCapSubPercentAdjustment() throws JposException;
Возвращает true, фискальный регистратор поддерживает процентные скидки.

public boolean getCapSubtotal() throws JposException;
Возвращает true, драйвер поддерживает метод printRecSubtotal.

public boolean getCapTrainingMode() throws JposException;
Режим тренировки не поддерживается в драйвере.
```

```
public boolean getCapValidateJournal() throws JposException;
```
Возвращает false.

```
public boolean getCapXReport() throws JposException;
```
Возвращает true, ФР поддерживает печать X отчета.

```
public int getOutputID() throws JposException;
```
Возвращает идентификатор последнего события.

```
public int getPowerNotify() throws JposException;
```
Возвращает тип уведомления о питании принтера. По умолчанию уведомление отключено, powerNotify= PN_DISABLED.

```
public void setPowerNotify(int powerNotify) throws JposException;
```
Устанавливает тип уведомления о питании принтера. Возможные значения: JPOS_PN_ENABLED, JPOS_PN_DISABLED.

```
public int getPowerState() throws JposException;
```
Возвращает текущее состояние питания принтера. Возможные значения:
PS_UNKNOWN Невозможно определить состояние принтера по следующим причинам:
CapPowerReporting = PR_NONE; устройство не поддерживает уведомление о питании.
PowerNotify = PN_DISABLED; уведомления о питании запрещены.
DeviceEnabled = false; отслеживание состояния питания не ведется, пока устройство не будет включено.
PS_ONLINE Устройство включено и готово к работе. Возвращается если
CapPowerReporting = PR_STANDARD или PR_ADVANCED.
PS_OFF Устройство не включено или отключено от POS терминала. Возвращается если
CapPowerReporting = PR_ADVANCED.
PS_OFFLINE Устройство включено, но не готово или не может отвечать на запросы. Возвращается если
CapPowerReporting = PR_ADVANCED.
PS_OFF_OFFLINE Устройство выключено или отключено от POS терминала. Возвращается если
CapPowerReporting = PR_STANDARD.
Это свойство инициализируется значением PS_UNKNOWN после вызова метода **open**. Если **PowerNotify** разрешено и **DeviceEnabled** = true, то это свойство обновляется как только сервис определит изменение состояния.

```
public int getAmountDecimalPlace() throws JposException;
```
Возвращает количество десятичных разрядов, которые ФР использует для вычислений.
Свойство инициализируется, когда устройство включается (DeviceEnabled=true).

```
public boolean getAsyncMode() throws JposException;
```
Если true, то некоторые методы будут выполняться асинхронно, такие методы как **printRecItemAdjustment**, **printRecItem**, **printNormal**, и т.д.
Свойство инициализируется после вызова метода **open**.

```
public void setAsyncMode(boolean asyncMode) throws JposException;
```

```
public boolean getCheckTotal() throws JposException;
```
Если true, то драйвер автоматически сравнивает сумму чека фискального регистратора и сумму чека приложения. Если возвращается false, то автоматическое сравнение итогов не поддерживается и свойство **CheckTotal** доступно только для чтения.

```
public void setCheckTotal(boolean checkTotal) throws JposException;
```

```
public int getCountryCode() throws JposException;
```
Возвращает значение, определяющее какие страны поддерживает ФР. Возможные значения:

FPTR_CC_BRAZIL	ФР поддерживае налоговые правила Бразилии.
FPTR_CC_GREECE	ФР поддерживае налоговые правила Греции.
FPTR_CC_HUNGARY	ФР поддерживае налоговые правила Венгрии.
FPTR_CC_ITALY	ФР поддерживае налоговые правила Италии.

FPTR_CC_POLAND	ФР поддерживае налоговые правила Польши.
FPTR_CC_TURKEY	ФР поддерживае налоговые правила Турции.
FPTR_CC_RUSSIA	ФР поддерживае налоговые правила России.
FPTR_CC_BULGARIA	ФР поддерживае налоговые правила Болгарии.
FPTR_CC_ROMANIA	ФР поддерживае налоговые правила Румынии.
FPTR_CC_CZECH_REPUBLIC	ФР поддерживае налоговые правила Чехии.
FPTR_CC_UKRAINE	ФР поддерживае налоговые правила Украины.
FPTR_CC_SWEDEN	ФР поддерживае налоговые правила Швеции.
FPTR_CC_OTHER	Неизвестная или новая страна. .

Свойство инициализируется после первого включения устройства.

```
public boolean getCoverOpen() throws JposException;
```

Если true, значит крышка ФР открыта.
Если **CapCoverSensor** = false, значит у ФР нет датчика крышки и это свойство всегда false.
Свойство обновляется, если

```
public boolean getDayOpened() throws JposException;
```

```
    public int     getDescriptionLength() throws JposException;
    public boolean getDuplicateReceipt() throws JposException;
    public int     getErrorLevel() throws JposException;
    public int     getErrorOutID() throws JposException;
    public int     getErrorState() throws JposException;
    public int     getErrorStation() throws JposException;
    public String  getErrorString() throws JposException;
    public boolean getFlagWhenIdle() throws JposException;
    public void    setFlagWhenIdle(boolean flagWhenIdle) throws JposException;
    public boolean getJrnEmpty() throws JposException;
    public boolean getJrnNearEnd() throws JposException;
    public int     getMessageLength() throws JposException;
    public int     getNumHeaderLines() throws JposException;
    public int     getNumTrailerLines() throws JposException;
    public int     getNumVatRates() throws JposException;
    public String  getPredefinedPaymentLines() throws JposException;
    public int     getPrinterState() throws JposException;
    public int     getQuantityDecimalPlaces() throws JposException;
    public int     getQuantityLength() throws JposException;
    public boolean getRecEmpty() throws JposException;
    public boolean getRecNearEnd() throws JposException;
    public int     getRemainingFiscalMemory() throws JposException;
    public String  getReservedWord() throws JposException;
    public boolean getSlpEmpty() throws JposException;
    public boolean getSlpNearEnd() throws JposException;
    public int     getSlipSelection() throws JposException;
    public void    setSlipSelection(int slipSelection) throws JposException;
    public boolean getTrainingModeActive() throws JposException;


    public void    beginFiscalDocument(int documentAmount)
                       throws JposException;
    public void    beginFiscalReceipt(boolean printHeader)
                       throws JposException;
    public void    beginFixedOutput(int station, int documentType)
                       throws JposException;
    public void    beginInsertion(int timeout) throws JposException;
    public void    beginItemList(int vatID) throws JposException;
    public void    beginNonFiscal() throws JposException;
    public void    beginRemoval(int timeout) throws JposException;
    public void    beginTraining() throws JposException;
    public void    clearError() throws JposException;
    public void    clearOutput() throws JposException;
    public void    endFiscalDocument() throws JposException;
    public void    endFiscalReceipt(boolean printHeader) throws JposException;
```

```
    public void    endFixedOutput() throws JposException;
    public void    endInsertion() throws JposException;
    public void    endItemList() throws JposException;
    public void    endNonFiscal() throws JposException;
    public void    endRemoval() throws JposException;
    public void    endTraining() throws JposException;
    public void    getData(int dataItem, int[] optArgs, String[] data)
                       throws JposException;
    public void    getDate(String[] Date) throws JposException;
    public void    getTotalizer(int vatID, int optArgs, String[] data)
                       throws JposException;
    public void    getVatEntry(int vatID, int optArgs, int[] vatRate)
                       throws JposException;
    public void    printDuplicateReceipt() throws JposException;
    public void    printFiscalDocumentLine(String documentLine)
                       throws JposException;
    public void    printFixedOutput(int documentType, int lineNumber,
                       String data) throws JposException;
    public void    printNormal(int station, String data) throws JposException;
    public void    printPeriodicTotalsReport(String date1, String date2)
                       throws JposException;
    public void    printPowerLossReport() throws JposException;
    public void    printRecItem(String description, long price, int quantity,
                       int vatInfo, long unitPrice, String unitName)
                       throws JposException;
    public void    printRecItemAdjustment(int adjustmentType,
                       String description, long amount, int vatInfo)
                       throws JposException;
    public void    printRecMessage(String message) throws JposException;
    public void    printRecNotPaid(String description, long amount)
                       throws JposException;
    public void    printRecRefund(String description, long amount, int vatInfo)
                       throws JposException;
    public void    printRecSubtotal(long amount) throws JposException;
    public void    printRecSubtotalAdjustment(int adjustmentType,
                       String description, long amount) throws JposException;
    public void    printRecTotal(long total, long payment, String description)
                       throws JposException;
    public void    printRecVoid(String description) throws JposException;
    public void    printRecVoidItem(String description, long amount,
                       int quantity, int adjustmentType, long adjustment,
                       int vatInfo) throws JposException;
    public void    printReport(int reportType, String startNum, String endNum)
                       throws JposException;
    public void    printXReport() throws JposException;
    public void    printZReport() throws JposException;
    public void    resetPrinter() throws JposException;
    public void    setDate(String date) throws JposException;
    public void    setHeaderLine(int lineNumber, String text,
                       boolean doubleWidth) throws JposException;
    public void    setPOSID(String POSID, String cashierID)
                       throws JposException;
    public void    setStoreFiscalID(String ID) throws JposException;
    public void    setTrailerLine(int lineNumber, String text,
                       boolean doubleWidth) throws JposException;
    public void    setVatTable() throws JposException;
    public void    setVatValue(int vatID, String vatValue)
                       throws JposException;
    public void    verifyItem(String itemName, int vatID) throws JposException;

}
public interface FiscalPrinterService16
```

```
  extends FiscalPrinterService15
{
    // Capabilities
    public boolean getCapAdditionalHeader() throws JposException;
    public boolean getCapAdditionalTrailer() throws JposException;
    public boolean getCapChangeDue() throws JposException;
    public boolean getCapEmptyReceiptIsVoidable() throws JposException;
    public boolean getCapFiscalReceiptStation() throws JposException;
    public boolean getCapFiscalReceiptType() throws JposException;
    public boolean getCapMultiContractor() throws JposException;
    public boolean getCapOnlyVoidLastItem() throws JposException;
    public boolean getCapPackageAdjustment() throws JposException;
    public boolean getCapPostPreLine() throws JposException;
    public boolean getCapSetCurrency() throws JposException;
    public boolean getCapTotalizerType() throws JposException;

    // Properties
    public int     getActualCurrency() throws JposException;
    public String  getAdditionalHeader() throws JposException;
    public void    setAdditionalHeader(String additionalHeader)
                       throws JposException;
    public String  getAdditionalTrailer() throws JposException;
    public void    setAdditionalTrailer(String additionalTrailer)
                       throws JposException;
    public String  getChangeDue() throws JposException;
    public void    setChangeDue(String changeDue) throws JposException;
    public int     getContractorId() throws JposException;
    public void    setContractorId(int contractorId) throws JposException;
    public int     getDateType() throws JposException;
    public void    setDateType(int dateType) throws JposException;
    public int     getFiscalReceiptStation() throws JposException;
    public void    setFiscalReceiptStation(int fiscalReceiptStation)
                       throws JposException;
    public int     getFiscalReceiptType() throws JposException;
    public void    setFiscalReceiptType(int fiscalReceiptType)
                       throws JposException;
    public int     getMessageType() throws JposException;
    public void    setMessageType(int messageType) throws JposException;
    public String  getPostLine() throws JposException;
    public void    setPostLine(String postLine) throws JposException;
    public String  getPreLine() throws JposException;
    public void    setPreLine(String preLine) throws JposException;
    public int     getTotalizerType() throws JposException;
    public void    setTotalizerType(int totalizerType) throws JposException;

    // Methods
    public void    setCurrency(int newCurrency) throws JposException;
    public void    printRecCash(long amount) throws JposException;
    public void    printRecItemFuel(String description, long price,
                       int quantity, int vatInfo, long unitPrice,
                       String unitName, long specialTax, String specialTaxName)
                       throws JposException;
    public void    printRecItemFuelVoid(String description, long price,
                       int vatInfo, long specialTax) throws JposException;
    public void    printRecPackageAdjustment(int adjustmentType,
                       String description, String vatAdjustment)
                       throws JposException;
    public void    printRecPackageAdjustVoid(int adjustmentType,
                       String vatAdjustment) throws JposException;
    public void    printRecRefundVoid(String description, long amount,
                       int vatInfo) throws JposException;
    public void    printRecSubtotalAdjustVoid(int adjustmentType, long amount)
                       throws JposException;
```

```
    public void    printRecTaxID(String taxID) throws JposException;
}

public interface FiscalPrinterService17
  extends FiscalPrinterService16
{
      // The AmountDecimalPlaces property was incorrectly spelled
      // AmountDecimalPlace since version 1.3.  In version 1.7.2 and later,
      // the correct spelling is supported. The old version is left for
      // Application and Device Service compatibility. The implementations
      // of getAmountDecimalPlaces and getAmountDecimalPlace should be
      // identical.

      // Properties
      public int     getAmountDecimalPlaces() throws JposException;


}
public interface FiscalPrinterService18
  extends FiscalPrinterService17
{
  // Capabilities
  public boolean getCapStatisticsReporting() throws JposException;
  public boolean getCapUpdateStatistics() throws JposException;

  // Methods
  public void    resetStatistics(String statisticsBuffer)
                     throws JposException;
  public void    retrieveStatistics(String[] statisticsBuffer)
                     throws JposException;
  public void    updateStatistics(String statisticsBuffer)
                     throws JposException;

}
public interface FiscalPrinterService19
  extends FiscalPrinterService18
{
  // Capabilities
  public boolean getCapCompareFirmwareVersion() throws JposException;
  public boolean getCapUpdateFirmware() throws JposException;

  // Methods
  public void    compareFirmwareVersion(String firmwareFileName, int[] result)
                     throws JposException;
  public void    updateFirmware(String firmwareFileName)
                     throws JposException;
}
public interface FiscalPrinterService111 extends FiscalPrinterService110
{
  // Capabilities
  public boolean getCapPositiveSubtotalAdjustment() throws JposException;

  // Methods
  public void    printRecItemVoid(String description,
                                  long price,
                                  int quantity,
                                  int vatInfo,
                                  long unitPrice,
                                  String unitName)
                     throws JposException;
  public void    printRecItemAdjustmentVoid(int adjustmentType,
                                            String description,
```

```
                                        long amount,
                                        int vatInfo)
                    throws JposException;
}
public interface FiscalPrinterService112 extends FiscalPrinterService111
{
  // Methods
  public void    printRecItemRefund(String description,
                                    long amount,
                                    int quantity,
                                    int vatInfo,
                                    long unitAmount,
                                    String unitName)
                    throws JposException;
  public void    printRecItemRefundVoid(String description,
                                        long amount,
                                        int quantity,
                                        int vatInfo,
                                        long unitAmount,
                                        String unitName)
                    throws JposException;
}
```

Параметры драйвера

Параметры драйвера хранятся в файле jpos.xml. Загрузкой параметров занимается ControlObject, то есть библиотека jpos. Драйвер получает JposEntry.

1. Тип порта: 0 - последовательный порт, 1 - bluetooth, 2 - socket, 3 - создание класса по названию

```
<!-- Port type: 0 - serial, 1 - bluetooth, 2 - socket, 3 - from parameter protClass -->
<prop name="portType" type="String" value="0"/>
```

2. Название класса порта

```
<!-- portClass  -->
<prop name="portClass" type="String"
value="com.shtrih.fiscalprinter.port.SerialPrinterPort"/>
```

3. Тип протокола: 0 - протокол 1.0, 1 - протокол версии 2.0

```
<!-- ProtocolType, 0 - protocol 1, 1 - protocol 2 -->
<prop name="protocolType" type="String" value="0"/>
```

4. Имя порта.

```
<!--Port name-->
<prop name="portName" type="String" value="COM1"/>
```

5. Скорость связи

```
<prop name="baudRate" type="String" value="115200"/>
```

6. Отдел по умолчанию. Можно изменить также через directIO

```
<!--Default department-->
<prop name="department" type="String" value="1"/>
```

7. Номер шрифта по умолчанию. Можно изменить также через directIO

```
<!-- Default font number -->
<prop name="fontNumber" type="String" value="1"/>
```

8. Текст для закрытия чека

```
<!-- Close receipt text -->
<prop name="closeReceiptText" type="String" value=""/>
```

9. Текст подитога, печатается в методе printRecSubtotal

```
<!-- Subtotal text -->
<prop name="subtotalText" type="String" value="ПОДИТОГ:"/>
```

10. Таймаут приема байта драйвера в миллисекундах.

```
<!-- Driver byte receive timeout -->
<prop name="byteTimeout" type="String" value="3000"/>
```

11. Таймаут приема байта устройства в миллисекундах, записывается в ФР при инициализации

```
<!-- Device byte receive timeout -->
<prop name="deviceByteTimeout" type="String" value="3000"/>
```

12. Разрешение поиска устройства на всех портах системы. Драйвер начинает поиск устройства, если не удалось подключиться к устройству с заданными параметрами. По умолчанию выключен.

```
<!-- Device search enabled for all serial ports -->
<prop name="searchByPortEnabled" type="String" value="0"/>
```

13. Разрешение поиска устройства на всех скоростях. По умолчанию включен.

```
<!-- Device search enabled for all baud rates -->
<prop name="searchByBaudRateEnabled" type="String" value="1"/>
```

14. Пароль налогового инспектора

```
<!-- Tax officer password -->
<prop name="taxPassword" type="String" value="0"/>
```

15. Пароль оператора

```
<!-- Operator password -->
<prop name="operatorPassword" type="String" value="1"/>
```

16. Пароль системного администратора

```
<!-- System administrator password -->
<prop name="sysAdminPassword" type="String" value="30"/>
```

17. Разрешение опроса устройства. Опрос устройства нужен для оповещения приложения о

```
<!-- device state polling enabled -->
<prop name="pollEnabled" type="String" value="0"/>
```

18. Интервал опроса в миллисекундах

```
<!-- device state polling interval in milliseconds -->
<prop name="pollInterval" type="String" value="100"/>
```

19. Коэффициент для сумм

```
<!-- Amount coefficient -->
<prop name="amountFactor" type="String" value="1"/>
```

20. Коэффициент для количества

```
<!-- Quantity coefficient -->
<prop name="quantityFactor" type="String" value="1"/>
```

21. Кодировка текстовых строк

```
<!-- Strings encoding -->
<prop name="stringEncoding" type="String" value="Cp866"/>
```

22. Имя файла статистики

```
<!-- Statistics file name -->
<prop name="statisticFileName" type="String" value="ShtrihFiscalPrinter.xml"/>
```

23. Задержка после печати штрихкода командой "Печать графической линии"

```
<!-- Barcode print time -->
<prop name="graphicsLineDelay" type="String" value="1000"/>
```

24. Имя файла для записи таблиц ФР. Драйвер запишет эти данные в ФР при инициализации

```
<!-- fieldsFileName to initialize printer tables -->
<prop name="fieldsFileName" type="String" value="tables.csv"/>
```

25. Имя папки для поиска файлов таблиц. Драйвер выбирает файл по имени устройства.

```
<!-- fieldsFilesPath to initialize printer tables -->
<prop name="fieldsFilesPath" type="String" value="I:\Projects\JavaPOS\Bin\tables\"/>
```

26. Количество строк заголовка чека

```
<!-- Number of header lines -->
<prop name="numHeaderLines" type="String" value="4"/>
```

27. Количество строк рекламного текста

```
<!-- Number of trailer lines -->
<prop name="numTrailerLines" type="String" value="3"/>
```

28. Тип отчета, который снимается при вызове printReport: 0 - отчет по ФП, 1 - отчет по ЭКЛЗ

```
<!-- Device to print report, 0 - fiscal memory (FM), 1 - electronic journal (EJ)  -->
<prop name="reportDevice" type="String" value="0"/>
```

29. Тип отчета, 0 - краткий, 1 - полный

```
<!-- Report type, 0 - short, 1 - full -->
<prop name="reportType" type="String" value="1"/>
```

30. Команда для чтения состояния ФР: 0 - краткий запрос состояния 10h, 1 - полный запрос, 2 - автоматический выбор (10h, если поддерживается, иначе 11h)

```
<!-- Status command:   -->
<!-- 0 - command 10h, short status request  -->
<!-- 1 - command 11h, long status request  -->
<!-- 2 - status command selected by driver -->
<prop name="statusCommand" type="String" value="0"/>
```

31. Файл для всех сообщений драйвера

```
<!-- Localization file name -->
<prop name="messagesFileName" type="String" value="shtrihjavapos_en.properties"/>
```

32. Разрешение переноса длинных строк

```
<!-- Wrap text enabled -->
<prop name="wrapText" type="String" value="1"/>
```

33. Задержка после закрытия чека. Может потребоваться для некоторых моделей ФР.

```
<!-- Sleep time after receipt close -->
<prop name="recCloseSleepTime" type="String" value="0"/>
```

34. Тип отрезки чека, 0 - полная, 1 - неполная

```
<!-- Cut type, 0 - full cut, 1 - partial cut -->
<prop name="cutType" type="String" value="1"/>
```

35. Режим отрезки, 0 - автоматически, 1 - отрезка запрещена

```
<!-- Cut mode, 0 - auto, 1 - disabled -->
<prop name="cutMode" type="String" value="0"/>
```

36. Параметр протокола ФР 1.0, максимальное количество запросов ENQ при передаче одной команды

```
<!-- maxEnqNumber -->
<prop name="maxEnqNumber" type="String" value="3"/>
```

37. Параметр протокола ФР 1.0, максимальное количество ответов NAK при передаче одной команды, то есть максимальное количество ошибок при передаче команды.

```
<!-- maxNakCommandNumber -->
<prop name="maxNakCommandNumber" type="String" value="3"/>
```

38. Параметр протокола ФР 1.0, максимальное количество ответов NAK при приеме ответа.

```
<!-- maxNakAnswerNumber -->
<prop name="maxNakAnswerNumber" type="String" value="3"/>
```

39. Максимальное количество повторов команды

```
<!-- maxRepeatCount -->
<prop name="maxRepeatCount" type="String" value="1"/>
```

40. Названия типов оплаты приложения

```
<!-- Payment types -->
<prop name="payType0" type="String" value="0"/>
<prop name="payType39" type="String" value="3"/>
```

41. Названия типов оплаты ФР

```
<!-- Payment names -->
<prop name="paymentName1" type="String" value="CASH"/>
<prop name="paymentName2" type="String" value="CREDIT"/>
<prop name="paymentName3" type="String" value="КАРТА"/>
<prop name="paymentName4" type="String" value="СКИДКА"/>
```

42. Названия налоговых групп

```
<!-- Tax names -->
<prop name="taxName0" type="String" value="НДС 10%"/>
<prop name="taxName1" type="String" value="НДС 18%"/>
<prop name="taxName2" type="String" value="С"/>
<prop name="taxName3" type="String" value="D"/>
```

43. Разрешение получения Z- отчета в виде XML файла. Драйвер сохраняет данные всех денежных и операционных регистров XML файле.

```
<!-- create Z-report in XML format -->
<prop name="XmlZReportEnabled" type="String" value="1"/>
```

44. Добавлять номер смены к имени файла отчета.

```
<!-- Add day number to Z report filename - ZReport_0001.xml -->
<prop name="ZReportDayNumber" type="String" value="1"/>
```

45. Имя XML файла Z отчета

```
<!-- XML Z-report file name -->
<prop name="XmlZReportFileName" type="String" value="ZReport.xml"/>
```

46. Разрешение получения Z- отчета в виде CSV файла. Драйвер сохраняет данные всех денежных и операционных регистров CSV файле.

```
<!-- create Z-report in CSV format -->
<prop name="CsvZReportEnabled" type="String" value="0"/>
```

47. Имя CSV файла Z отчета

```
<!-- CSV Z-report file name -->
<prop name="CsvZReportFileName" type="String" value="ZReport.csv"/>
```

48. Разрешение обработки ESC команд в тексте. Это нужно для поддержки некоторых устаревших приложений

```
<!-- ESC commands enabled -->
<prop name="escCommandsEnabled" type="String" value="1"/>
```

49. Режим записи таблиц, 0 - автоматически, 1 - запрещена запись в таблицы

```
<!-- Table mode, 0 - auto, 1 - disabled -->
<prop name="tableMode" type="String" value="0"/>
```

50. Режим логотипа перед заголовком чека, 0 - промотка чека на величину клише, 1 - печатать логотип в 2 этапа

```
<!-- Logo mode, 0 - feed paper, 1 - split image -->
<prop name="logoMode" type="String" value="1"/>
```

51. Режим поиска ФР. 0 - нет поиска, 1 - поиск ФР при ошибках

```
<!-- SearchMode, 0 - none, 1 - search on error -->
<prop name="searchMode" type="String" value="1"/>
```

52. Задержка после отрезки чека в миллисекундах

```
<!-- Paper cut delay in milliseconds -->
<prop name="cutPaperDelay" type="String" value="0"/>
```

53. Тип чека продажи, 0 - обычный, 1 - текстовый чек

```
<!-- Sales receipt type, 0 - normal, 1 - GLOBUS -->
```

```
<prop name="salesReceiptType" type="String" value="1"/>
```

54. Длина поля "цена" для формата чека

```
<!-- Amount field length -->
<prop name="RFAmountLength" type="String" value="8"/>
```

55. Длина поля "количество" для формата чека

```
<!-- Quantity field length -->
<prop name="RFQuantityLength" type="String" value="10"/>
```

56. Порт для системы мониторинга. Система мониторинга нужна для удаленного запроса состояния ФР и ЭКЛЗ

```
<!-- Monitoring server port -->
<prop name="MonitoringPort" type="String" value="50000"/>
```

57. Разрешение работы системы мониторинга

```
<!-- Monitoring enabled -->
<prop name="MonitoringEnabled" type="String" value="0"/>
```

58. Разрешение сохранения чека в текстовом виде

```
<!-- Receipt report enabled -->
<prop name="receiptReportEnabled" type="String" value="1"/>
```

59. Название файла чека

```
<!-- Receipt report file name -->
<prop name="receiptReportFileName" type="String" value="ZCheckReport.xml"/>
```

60. Тип открытия чека, 0 - открыть чек при печати позиции, 1 - открыть чека в методе beginFiscalReceipt

```
<!-- openReceiptType, 0 - open receipt on item print, 1 - open receipt in
beginFiscalReceipt -->
<prop name="openReceiptType" type="String" value="1"/>
```

61. Режим заголовка, 0 - заголовок в драйвере, 1 - в принтере

```
<!-- headerMode, 0 - header in driver, 1 - header in printer -->
<prop name="headerMode" type="String" value="0"/>
```

62. Позиция логотипа в заголовке чека

```
<!-- headerImagePosition,  -->
<!-- SMFPTR_LOGO_AFTER_HEADER     = 0  -->
<!-- SMFPTR_LOGO_BEFORE_TRAILER   = 1  -->
<!-- SMFPTR_LOGO_AFTER_TRAILER    = 2  -->
<!-- SMFPTR_LOGO_AFTER_ADDTRAILER = 3  -->
<!-- SMFPTR_LOGO_BEFORE_HEADER    = 4  -->
<prop name="headerImagePosition" type="String" value="0"/>
```

63. Позиция логотипа в рекламном тексте

```
<!-- trailerImagePosition,  -->
<!-- SMFPTR_LOGO_AFTER_HEADER     = 0  -->
<!-- SMFPTR_LOGO_BEFORE_TRAILER   = 1  -->
<!-- SMFPTR_LOGO_AFTER_TRAILER    = 2  -->
<!-- SMFPTR_LOGO_AFTER_ADDTRAILER = 3  -->
<!-- SMFPTR_LOGO_BEFORE_HEADER    = 4  -->
<prop name="trailerImagePosition" type="String" value="1"/>
```

64. Центрировать строки заголовка чека

```
<!-- Center header and trailer text automatically  -->
<prop name="centerHeader" type="String" value="1"/>
```

65. Разрешение лога

```
<!-- Log file enabled  -->
<prop name="logEnabled" type="String" value="1"/>
```

66. Передавать ENQ перед каждой командой

```
<!-- Send ENQ before every command or not  -->
<prop name="sendENQ" type="String" value="0"/>
```

67. Печатать буквы налоговых групп на текстовом чеке

```
<!-- Enable tax letters for GLOBUS receipt  -->
<prop name="taxLettersEnabled" type="String" value="0"/>
```

68. Префикс штрихкода для печати штрихкода как текста

```
<!-- Barcode prefix -->
<prop name="barcodePrefix" type="String" value="#*~*#"/>
```

69. Тип штрихкода:

UPCA=0, UPCE=1, EAN13=2, EAN8=3, CODE39=4, ITF=5, CODABAR=6, CODE93=7, CODE128=8, PDF417=10, GS1_OMNI=11, GS1_TRUNC=12, GS1_LIMIT=13, GS1_EXP=14, GS1_STK=15,

GS1_STK_OMNI=16, GS1_EXP_STK=17, AZTEC=18, DATA_MATRIX=19, MAXICODE=20, QR_CODE=21, RSS_14=22, RSS_EXPANDED=23, UPC_EAN_EXTENSION=24

```
<!-- Barcode type -->
<!-- UPCA=0, UPCE=1, EAN13=2, EAN8=3, CODE39=4, ITF=5, CODABAR=6, CODE93=7, CODE128=8,
PDF417=10, GS1_OMNI=11,  -->
<!-- GS1_TRUNC=12, GS1_LIMIT=13, GS1_EXP=14, GS1_STK=15, GS1_STK_OMNI=16, GS1_EXP_STK=17,
AZTEC=18, DATA_MATRIX=19,  -->
<!-- MAXICODE=20, QR_CODE=21, RSS_14=22, RSS_EXPANDED=23, UPC_EAN_EXTENSION=24  -->
<prop name="barcodeType" type="String" value="21"/>
```

70. Ширина штриха в точках. Обычно 2-3 для одномерного штрихкода, 3-4 для двумерного

```
<!-- Barcode bar/module width -->
<prop name="barcodeBarWidth" type="String" value="4"/>
```

71. Высота штрихкода. Имеет значение для одномерных штрихкодов.

```
<!-- Barcode height -->
<prop name="barcodeHeight" type="String" value="100"/>
```

72. Позиция текста относительно штрихкода: 0 - не печатать, 1 - сверху, 2 - снизу, 3 - сверху и снизу

```
<!-- Barcode text position NOTPRINTED=0, ABOVE=1, BELOW=2, BOTH=3 -->
<prop name="barcodeTextPosition" type="String" value="2"/>
```

73. Штрифт текста штрихкода, 1..7

```
<!-- Barcode text font, 1..7, default 1 -->
<prop name="barcodeTextFont" type="String" value="1"/>
```

74. Соотношение ширины и высоты для штрихкода

```
<!-- Barcode aspect ratio -->
<prop name="barcodeAspectRatio" type="String" value="3"/>
```

75. Совместимость со старыми версиями драйвера: 0 - нет, 1 - полная

```
<!-- Compatibility level, 0 - NONE, 1 - FULL -->
<prop name="compatibilityLevel" type="String" value="0"/>
```

Метод DirectIO

1. Выполнить произвольную команду.

```
/** Execute command object **/
public static final int SMFPTR_DIO_COMMAND = 0x00;
```

Драйвер выполняет отдельную команду ФР. В качестве команды передается объект PrinterCommand. Например:

```
public byte[] executeCommand(byte[] tx, int timeout) throws JposException {
    int[] data = new int[1];
    Object[] object = new Object[2];
    data[0] = timeout;
    object[0] = tx;
    directIO(SmFptrConst.SMFPTR_DIO_COMMAND, data, object);
    byte[] rx = (byte[]) object[1];
    return rx;
}
```

2. Печать штрихкода. Параметры передаются через объект PrinterBarcode.

```
/** Print barcode object **/
public static final int SMFPTR_DIO_PRINT_BARCODE_OBJECT = 0x01;

public void printBarcode(PrinterBarcode barcode) throws JposException {
       printer.directIO(SmFptrConst.SMFPTR_DIO_PRINT_BARCODE_OBJECT,
                    null, barcode);
}
```

3. Установка номера отдела.

```
/** Set department **/
public static final int SMFPTR_DIO_SET_DEPARTMENT = 0x02;
public void setDepartment(int department) throws JposException {
    int[] data = new int[1];
    int[] value = new int[1];
    value[0] = department;
    directIO(SmFptrConst.SMFPTR_DIO_SET_DEPARTMENT, data, value);
}
```

4. Чтение номера отдела.

```
/** Get department **/
public static final int SMFPTR_DIO_GET_DEPARTMENT = 0x03;
public int getDepartment() throws JposException {
    int[] data = new int[1];
    int[] value = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_GET_DEPARTMENT, data, value);
    return value[0];
}
```

5. Выполнить команду ФР.

```
/** Execute string command **/
public static final int SMFPTR_DIO_STRCOMMAND = 0x04;
public String executeCommand(int code, int timeout, String inParams)
        throws JposException {
    int[] data = new int[1];
    data[0] = code;
    String[] lines = new String[3];
    lines[0] = String.valueOf(timeout);
    lines[1] = inParams;
    directIO(SmFptrConst.SMFPTR_DIO_STRCOMMAND, data, lines);
    String outParams = lines[2];
    return outParams;
}
```

6. Чтение таблицы ФР

```
/** Read table command **/
public static final int SMFPTR_DIO_READTABLE = 0x05;
public String readTable(int tableNumber, int rowNumber, int fieldNumber)
        throws JposException {
    String[] params = new String[4];
    params[0] = String.valueOf(tableNumber);
    params[1] = String.valueOf(rowNumber);
    params[2] = String.valueOf(fieldNumber);
    directIO(SmFptrConst.SMFPTR_DIO_READTABLE, null, params);
    return params[3];
}
```

Параметр data - не используется. Параметр object - массив строк длины 4.

Строка с индексом 0 - номер таблицы

Строка с индексом 1 - номер ряда

Строка с индексом 2 - номер поля

Строка с индексом 3 - возвращаемое значение

7. Запись таблицы ФР

```
/** Write table command **/
public static final int SMFPTR_DIO_WRITETABLE = 0x06;
public void writeTable(int tableNumber, int rowNumber, int fieldNumber,
        String fieldValue) throws JposException {
    String[] params = new String[4];
    params[0] = String.valueOf(tableNumber);
    params[1] = String.valueOf(rowNumber);
    params[2] = String.valueOf(fieldNumber);
    params[3] = fieldValue;
    directIO(SmFptrConst.SMFPTR_DIO_WRITETABLE, null, params);
}
```

Параметр data - не используется. Параметр object - массив строк длины 4.

Строка с индексом 0 - номер таблицы

Строка с индексом 1 - номер ряда

Строка с индексом 2 - номер поля

Строка с индексом 3 - возвращаемое значение

8. Чтение наименования типа оплаты

```
/** Read payment type name **/
public static final int SMFPTR_DIO_READ_PAYMENT_NAME = 0x07;
public String readPaymentName(int number) throws JposException {
    int[] data = new int[1];
    data[0] = number;
    String[] fieldValue = new String[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_PAYMENT_NAME, data, fieldValue);
    return fieldValue[0];
}
```

```
В параметре data[0] передается номер типа оплаты от 1 до 4.
Параметр object является массивом строк длины 1. В первой строке массива возвращается
название типа оплаты.
```

9. Запись наименования типа оплаты

```
В параметре data[0] передается номер типа оплаты от 1 до 4.
Параметр object является массивом строк длины 1. В первой строке массива передается
название типа оплаты.
/** write payment type name **/
public static final int SMFPTR_DIO_WRITE_PAYMENT_NAME = 0x08;
public void writePaymentName(int number, String value) throws JposException {
    int[] data = new int[1];
    data[0] = number;
    String[] fieldValue = new String[1];
    fieldValue[0] = value;
    directIO(SmFptrConst.SMFPTR_DIO_WRITE_PAYMENT_NAME, data, fieldValue);
}
```

10. Чтение флага окончания смены (когда 24 часа смены истекли)

```
/** Read end of day flag **/
public static final int SMFPTR_DIO_READ_DAY_END = 0x09;
public boolean readDayEnd() throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_DAY_END, data, null);
    return data[0] != 0;
}
```

```
В параметре data передается массив int длиной 1. В элементе с индексом 0 передается флаг
окончания смены
```

11. Печать штрихкода

```
/** Print barcode command **/
public static final int SMFPTR_DIO_PRINT_BARCODE = 0x0A;
public void printBarcode(String barcode, String label, int barcodeType,
        int barcodeHeight, int printType, int barWidth, int textPosition,
        int textFont, int aspectRatio) throws JposException {
```

```
    Object[] params = new Object[9];
    params[0] = barcode; // barcode data
    params[1] = label; // barcode label
    params[2] = new Integer(barcodeType); // barcode type
    params[3] = new Integer(barcodeHeight); // barcode height in pixels
    params[4] = new Integer(printType); // print type
    params[5] = new Integer(barWidth); // barcode bar width in pixels
    params[6] = new Integer(textPosition); // text position
    params[7] = new Integer(textFont); // text font
    params[8] = new Integer(aspectRatio); // narrow to width ratio, 3 by default
    directIO(SmFptrConst.SMFPTR_DIO_PRINT_BARCODE, null, params);
}
```

Параметр data не используется.
Параметр object является массивом объектов длиной 9.
params[0] - данные штрихкода, строка.
params[1] - текст штрихкода, строка.
params[2] - тип штрихкода, Integer
params[3] - высота штрихкода в точках, Integer
params[4] - метод печати штрихкода, Integer
params[5] - ширина элемента в точках, Integer
params[6] - положение текста, Integer
params[7] - шрифт текста, Integer
params[8] - соотношение ширины и высоты штрихкода, Integer

12. Загрузка изображения из файла

```
/** Load image from file **/
public static final int SMFPTR_DIO_LOAD_IMAGE = 0x0B;
public int loadImage(String fileName) throws JposException {
    int[] data = new int[1];
    String[] command = new String[1];
    command[0] = fileName;
    directIO(SmFptrConst.SMFPTR_DIO_LOAD_IMAGE, data, command);
    return data[0];
}
```

в параметре data[0] возвращается индекс загруженного изображения.
Параметр object является массивом строк длины 1. В первом элементе передается имя файла изображения.

13. Печать изображения

```
/** Print image **/
public static final int SMFPTR_DIO_PRINT_IMAGE = 0x0C;
public void printImage(int imageIndex) throws JposException {
    int[] data = new int[1];
    data[0] = imageIndex;
    directIO(SmFptrConst.SMFPTR_DIO_PRINT_IMAGE, data, null);
}
```

В параметре data[0] передается индекс загруженного изображения.

14. Удаление всех изображений

```
/** Clear all images **/
public static final int SMFPTR_DIO_CLEAR_IMAGES = 0x0D;
public void clearImages() throws JposException {
    directIO(SmFptrConst.SMFPTR_DIO_CLEAR_IMAGES, null, null);
}
```

Параметры не используются

15. Добавление логотипа

```
/** Set logo **/
public static final int SMFPTR_DIO_ADD_LOGO = 0x0E;
public void addLogo(int imageIndex, int logoPosition) throws JposException {
    int[] data = new int[2];
    data[0] = imageIndex;
    data[1] = logoPosition;
    directIO(SmFptrConst.SMFPTR_DIO_ADD_LOGO, null, data);
}
```

Параметр data не используется, параметр object представляет собой массив int длиной 2.
Индекс 0 массива - индекс изображения
Индекс 1 массива - позиция логотипа

16. Удаление логотипов

```
/** Clear logo **/
public static final int SMFPTR_DIO_CLEAR_LOGO = 0x0F;
```

```
public void clearLogo() throws JposException {
    directIO(SmFptrConst.SMFPTR_DIO_CLEAR_LOGO, null, null);
}
```

Параметры не используются

17. Печать линии

```
/** Print black line **/
public static final int SMFPTR_DIO_PRINT_LINE = 0x10;
public void printLine(int lineType, int lineHeight) throws JposException {
    int[] data = new int[2];
    data[0] = lineHeight;
    data[1] = lineType;
    directIO(SmFptrConst.SMFPTR_DIO_PRINT_LINE, null, data);
}
```

Параметр data не используется, параметр object представляет собой массив int длиной 2. Первый элемент массива - высота линии в точках, второй элемент - тип линии (0 - черная линия, 1 - белая).

```
public static final int SMFPTR_LINE_TYPE_BLACK = 0;
public static final int SMFPTR_LINE_TYPE_WHITE = 1;
```

18. Чтение параметра драйвера

```
/** Get driver parameter **/
public static final int SMFPTR_DIO_GET_DRIVER_PARAMETER = 0x11;
public String getParameter(int paramType) throws JposException {
    int data[] = new int[1];
    String object[] = new String[1];
    data[0] = paramType;
    directIO(SmFptrConst.SMFPTR_DIO_GET_DRIVER_PARAMETER, data, object);
    return object[0];
}
```

Номер параметра передается в параметре data. Параметр object представляет собой массив строк длиной 1. В первом элементе возвращается значение параметра в виде строки.

Номера параметров:

```
/** Report device for printReport, printPeriodicTotalsReport **/
public static final int SMFPTR_DIO_PARAM_REPORT_DEVICE = 0;
/** Report type for printReport, printPeriodicTotalsReport **/
public static final int SMFPTR_DIO_PARAM_REPORT_TYPE = 1;
/** Number of header lines **/
public static final int SMFPTR_DIO_PARAM_NUMHEADERLINES = 2;
/** Number of trailer lines **/
public static final int SMFPTR_DIO_PARAM_NUMTRAILERLINES = 3;
/** Polling enabled **/
public static final int SMFPTR_DIO_PARAM_POLL_ENABLED = 4;
/** Cut mode **/
public static final int SMFPTR_DIO_PARAM_CUT_MODE = 5;
/** Font number **/
public static final int SMFPTR_DIO_PARAM_FONT_NUMBER = 6;
```

19. Запись параметра драйвера

```
/** Set driver parameter **/
public static final int SMFPTR_DIO_SET_DRIVER_PARAMETER = 0x12;
public void setParameter(int paramType, int paramValue)
        throws JposException {
    int data[] = new int[1];
    String object[] = new String[1];
    data[0] = paramType;
    object[0] = String.valueOf(paramValue);
    directIO(SmFptrConst.SMFPTR_DIO_SET_DRIVER_PARAMETER, data, object);
}
```

Номер параметра передается в параметре data. Параметр object представляет собой массив строк длиной 1. В первом элементе передается значение параметра в виде строки.

Номера параметров приведены выше.

20. Печать текста

```
/** Print text **/
public static final int SMFPTR_DIO_PRINT_TEXT = 0x13;
public void printText(String text) throws JposException {
    int data[] = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_PRINT_TEXT, data, text);
}
```

Параметр data не используется, параметр object представляет текст для печати.

21. Запись таблиц ФР

```
/** Write table values from file **/
public static final int SMFPTR_DIO_WRITE_TABLES = 0x14;
public void writeTables(String fileName) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_WRITE_TABLES, data, fileName);
}
```

Параметр data не используется, параметр object представляет имя файла таблиц ФР.

22. Чтение таблиц ФР

```
/** Read table values to file **/
public static final int SMFPTR_DIO_READ_TABLES = 0x15;
public void readTables(String fileName) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_TABLES, data, fileName);
}
```

Параметр data не используется, параметр object представляет имя файла таблиц ФР.

23. Чтение серийного номера ФР

```
/** Read device serial number **/
public static final int SMFPTR_DIO_READ_SERIAL = 0x16;
public String readSerial() throws JposException {
    String[] serial = new String[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_SERIAL, null, serial);
    return serial[0];
}
```

Параметр data не используется, параметр object представляет массив строк длины 1.

24. Чтение серийного номера ЭКЛЗ

```
/** Read EJ serial number **/
public static final int SMFPTR_DIO_READ_EJ_SERIAL = 0x17;
public String readEJSerial() throws JposException {
    String[] serial = new String[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_EJ_SERIAL, null, serial);
    return serial[0];
}
```

Параметр data не используется, параметр object представляет массив строк длины 1.

25. Открыть денежный ящик

```
/** Open cash drawer **/
public static final int SMFPTR_DIO_OPEN_DRAWER = 0x18;
public void openCashDrawer(int drawerNumber) throws JposException {
    int[] data = new int[1];
    data[0] = drawerNumber;
    directIO(SmFptrConst.SMFPTR_DIO_OPEN_DRAWER, data, null);
}
```

Параметр object не используется, в параметре data передается номер денежного ящика. Обычно денежный ящик один с номером 0

26. Прочитать состояние денежного ящика

```
/** Read cash drawer state **/
public static final int SMFPTR_DIO_READ_DRAWER_STATE = 0x19;
public boolean readDrawerState() throws JposException {
    int[] data = new int[1];
    data[0] = 0;
    directIO(SmFptrConst.SMFPTR_DIO_READ_DRAWER_STATE, data, null);
    return data[0] != 0;
}
```

Параметр object не используется, в параметре data возвращается состояние ящика. 0 - ящик закрыт, 1 - открыт.

27. Прочитать состояние принтера

```
/** Read printer status **/
public static final int SMFPTR_DIO_READ_PRINTER_STATUS = 0x1A;
public PrinterStatus readPrinterStatus() throws JposException {
    int[] data = new int[1];
    Object[] object = new Object[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_PRINTER_STATUS, data, object);
    return (PrinterStatus)object[0];
}
```

Параметр data не используется, в параметре object возвращается объект состояния, PrinterStatus.

28. Прочитать денежный регистр

```
/** Read cash register **/
public static final int SMFPTR_DIO_READ_CASH_REG = 0x1B;
public CashRegister readCashRegister(int number) throws JposException {
    int[] data = new int[1];
    Object[] object = new Object[1];
    data[0] = number;
    directIO(SmFptrConst.SMFPTR_DIO_READ_CASH_REG, data, object);
    return (CashRegister)object[0];
}
```

В параметре data передается номер регистра, параметр object - массив объектов длины 1. В нем возвращается класс денежного регистра.

29. Прочитать операционный регистр

```
/** Read operation register **/
public static final int SMFPTR_DIO_READ_OPER_REG = 0x1C;
public OperationRegister readOperRegister(int number) throws JposException {
    int[] data = new int[1];
    Object[] object = new Object[1];
    data[0] = number;
    directIO(SmFptrConst.SMFPTR_DIO_READ_OPER_REG, data, object);
    return (OperationRegister)object[0];
}
```

В параметре data передается номер регистра, параметр object - массив объектов длины 1. В нем возвращается класс операционного регистра.

30. Выполнить команду

```
/* Execute command */
public static final int SMFPTR_DIO_COMMAND_OBJECT = 0x1D;
public void executeCommand(PrinterCommand command) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_COMMAND_OBJECT, data, command);
}
```

Параметр data не используется, параметр object - массив объектов длины 1. В нем передается объект команды (класс PrinterCommand).

31. Сохранить данные Z отчета в XML файле

```
 /* Save XML Z report */
public static final int SMFPTR_DIO_XML_ZREPORT = 0x1E;
public void saveXmlZReport(String fileName) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_XML_ZREPORT, data, fileName);
}
```

Параметр data не используется, параметр object - имя XML файла.

32. Сохранить данные Z отчета в CSV файле

```
/* Save CSV Z report */
public static final int SMFPTR_DIO_CSV_ZREPORT = 0x1F;
public void saveCsvZReport(String fileName) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_CSV_ZREPORT, data, fileName);
}
```

Параметр data не используется, параметр object - имя XML файла.

33. Запись параметра устройства

```
/* Write parameter */
public static final int SMFPTR_DIO_WRITE_DEVICE_PARAMETER = 0x20;
```

34. Чтение параметра устройства

```
/* Read parameter */
public static final int SMFPTR_DIO_READ_DEVICE_PARAMETER = 0x21;
```

35. Загрузка логотипа

```
/** Load logo **/
public static final int SMFPTR_DIO_LOAD_LOGO = 0x22;
public int loadLogo(int logoPosition, String fileName) throws JposException {
    int[] data = new int[1];
    data[0] = logoPosition;
    directIO(SmFptrConst.SMFPTR_DIO_ADD_LOGO, data, fileName);
    return data[0];
}
```

В параметре data передается позиция логотипа, параметр object - имя файла логотипа. В параметре data возвращается индекс изображения.

36. Чтение состояния смены (открыта, закрыта, истекли 24 часа).

```
/** Read fiscal day status **/
public static final int SMFPTR_DIO_READ_DAY_STATUS = 0x23;
public int readDayStatus() throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_DAY_STATUS, data, null);
    return data[0];
}
```

В параметре data возвращается состояние смены в ФР. Возможные значения:

```
public static final int SMFPTR_DAY_STATUS_CLOSED = 1;
public static final int SMFPTR_DAY_STATUS_OPENED = 2;
public static final int SMFPTR_DAY_STATUS_EXPIRED = 3;
public static final int SMFPTR_DAY_STATUS_UNKNOWN = 4;
```

37. Чтение номера лицензии.

```
/** Read device license number **/
public static final int SMFPTR_DIO_READ_LICENSE = 0x24;
public String readLicense() throws JposException {
    int[] data = new int[1];
    String[] object = new String[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_LICENSE, data, object);
    return object[0];
}
```

Параметр data не используется, параметр object представляет собой массив строк длиной 1. В первом элементе возвращается номер лицензии в виде строки.

38. Чтение готовности к печати фискальных документов.

```
/** Is printer ready for fiscal documents **/
public static final int SMFPTR_DIO_IS_READY_FISCAL = 0x25;
public boolean isReadyFiscal(String[] text) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_IS_READY_FISCAL, data, text);
    return data[0] != 0;
}
```

В параметре data возвращается состояние, 0 или 1. 1 возвращается, если в ФР смена закрыта или открыта. Если 24 часа истекли, или ФР в другом состоянии, то возвращается 0.

39. Чтение готовности к печати фискальных документов.

```
/** Is printer ready for non fiscal documents **/
public static final int SMFPTR_DIO_IS_READY_NONFISCAL = 0x26;
public boolean isReadyNonfiscal(String[] text) throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_IS_READY_NONFISCAL, data, text);
    return data[0] != 0;
}
```

В параметре data возвращается состояние, 0 или 1. 1 возвращается, если в ФР смена закрыта или открыта или 24 часа истекли. Если ФР в другом состоянии, то возвращается 0.

40. Определение максимального размера графики

```
/** Read maximum graphics size **/
public static final int SMFPTR_DIO_READ_MAX_GRAPHICS = 0x27;
public int readMaxGraphics() throws JposException {
    int[] data = new int[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_MAX_GRAPHICS, data, null);
    return data[0];
}
```

Драйвер выполняет определение при помощи команды 0xC3, Расширенная печать графики. Драйвер действует методом половинного деления. При определении принтер будет подавать сигналы об ошибке.

41. Чтение строки заголовка чека

```
/** Read header line **/
public static final int public String getHeaderLine(int lineNumber) throws JposException
{
    int[] data = new int[1];
    String[] lines = new String[1];
    data[0] = lineNumber;
    directIO(SmFptrConst.SMFPTR_DIO_GET_HEADER_LINE, data, lines);
    return lines[0];
}
```

В параметре data передается номер строки. Параметр object представляет собой массив строк длины 1. В первом элементе передается строка заголовка.

42. Чтение строки рекламного текста

```
/** Read trailer line **/
public static final int SMFPTR_DIO_GET_TRAILER_LINE = 0x29;
public String getTrailerLine(int lineNumber) throws JposException {
    int[] data = new int[1];
    String[] lines = new String[1];
    data[0] = lineNumber;
    directIO(SmFptrConst.SMFPTR_DIO_GET_TRAILER_LINE, data, lines);
    return lines[0];
}
```

В параметре data передается номер строки. Параметр object представляет собой массив строк длины 1. В первом элементе возвращается строка заголовка.

43. Получение длины строки

```
/** Read text length **/
public static final int SMFPTR_DIO_GET_TEXT_LENGTH = 0x2A;
public int getTextLength(int fontNumber) throws JposException {
    int[] data = new int[1];
    data[0] = fontNumber;
    directIO(SmFptrConst.SMFPTR_DIO_GET_TEXT_LENGTH, data, null);
    return data[0];
}
```

В параметре data передается номер шрифта. Параметр object не используется. В параметре data возвращается длина строки.

44. Чтение имени кассира

```
/** Read cashier name **/
public static final int SMFPTR_DIO_READ_CASHIER_NAME = 0x2B;
public String readCashierName() throws JposException {
    String[] lines = new String[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_CASHIER_NAME, null, lines);
    return lines[0];
}
```

Параметр data не используется. Параметр object представляет собой массив строк длины 1. В первом элементе возвращается имя кассира.

45. Запись имени кассира

```
/** Write cashier name **/
public static final int SMFPTR_DIO_WRITE_CASHIER_NAME = 0x2C;
public void writeCashierName(String value) throws JposException {
    String[] lines = new String[1];
    lines[0] = value;
    directIO(SmFptrConst.SMFPTR_DIO_WRITE_CASHIER_NAME, null, lines);
}
```

Параметр data не используется. Параметр object представляет собой массив строк длины 1. В первом элементе передается имя кассира.

46. Отрезка чека

```
/** Cut paper **/
public static final int SMFPTR_DIO_CUT_PAPER = 0x2D;
// cutMode: 0 - full cut, 1 - partial cut
public void cutPaper(int cutMode) throws JposException {
    int[] data = new int[1];
    data[0] = cutMode;
    directIO(SmFptrConst.SMFPTR_DIO_CUT_PAPER, data, null);
}
```

Параметр object не используется. В параметре data передается тип отрезки: 0 – полная, 1 – неполная.

47. Ожидание завершения печати

```
/** Wait for printing **/
public static final int SMFPTR_DIO_WAIT_PRINT = 0x2E;
public void waitForPrinting() throws JposException {
    directIO(SmFptrConst.SMFPTR_DIO_WAIT_PRINT, null, null);
}
```

Параметры data и object не используются. Метод опрашивает состояние принтера и ждет завершения печати.

48. Краткий запрос состояния

```
/** Short status **/
```

```
public static final int SMFPTR_DIO_READ_SHORT_STATUS = 0x30;
public ShortPrinterStatus readShortPrinterStatus() throws JposException {
    Object[] object = new Object[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_SHORT_STATUS, null, object);
    return (ShortPrinterStatus)object[0];
}
```

Параметр data не используется. Параметр object представляет собой массив объектов. В первом элементе возвращается краткий запрос состояния, класс ShortPrinterStatus.

49. Полный запрос состояния

```
/** Long status **/
public static final int SMFPTR_DIO_READ_LONG_STATUS = 0x31;
public LongPrinterStatus readLongPrinterStatus() throws JposException {
    Object[] object = new Object[1];
    directIO(SmFptrConst.SMFPTR_DIO_READ_LONG_STATUS, null, object);
    return (LongPrinterStatus)object[0];
}
```

Параметр data не используется. Параметр object представляет собой массив объектов. . В первом элементе возвращается полный запрос состояния, класс LongPrinterStatus.

50. Запись строки тега

Для записи в ФР строкового тега можно использовать метод fsWriteTag:

```
public void fsWriteTag(int tagId, String tagValue) throws Exception
{
    int[] data = new int[1];
    data[0] = tagId;
    directIO(SmFptrConst.SMFPTR_DIO_FS_WRITE_TAG, data, tagValue);
}
```

tagId - идентификатор тега, идентификаторы приведены в документе " Форматы фискальных документов". tagId = 1008 - адрес покупателя. tagValue - строка значения тега.

51. Запись в чек email покупателя при работе с ФН.

```
public static final int SMFPTR_DIO_FS_WRITE_CUSTOMER_EMAIL = 0x39;
public void fsWriteCustomerEmail() throws JposException {
    String email = "foo@example.com";
    directIO(SMFPTR_DIO_FS_WRITE_CUSTOMER_EMAIL, null, email);
}
```

В параметре object передается строка содержащая email покупателя.

52. Запись в чек телефона покупателя при работе с ФН.

```
public static final int SMFPTR_DIO_FS_WRITE_CUSTOMER_PHONE = 0x3A;
public void fsWriteCustomerPhone() throws JposException {
    String phoneNumber = "89261112233";
    directIO(SMFPTR_DIO_FS_WRITE_CUSTOMER_PHONE, null, phoneNumber);
}
```

В параметре object передается строка содержащая телефон покупателя.

Работа с фискальным накопителем

Для работы с фискальным накопителем можно использовать следующие методы:

1. Запрос статуса ФН FF01H

```
public void fsReadStatus(FSReadStatus command) throws JposException {
    executeCommand(command);
}
```

2. Запрос номера ФН FF02H

```
public void fsReadSerial(FSReadSerial command) throws JposException {
    executeCommand(command);
}
```

3. Чтение срока действия ФН:

```
public void fsReadExpDate(FSReadExpDate command) throws JposException {
    executeCommand(command);
}
```

4. Запрос версии ФН FF04H

```
public void fsReadVersion(FSReadVersion command) throws JposException {
```

```
    executeCommand(command);
}
```

5. Сформировать отчёт о регистрации ККТ FF06H

```
public void fsFiscalization(FSFiscalization command) throws JposException {
    executeCommand(command);
}
```

6. Сброс состояния ФН FF07H

```
public void fsResetState(FSResetState command) throws JposException {
    executeCommand(command);
}
```

7. Отменить документ в ФН FF08H

```
public void fsCancelDoc(FSCancelDoc command) throws JposException {
    executeCommand(command);
}
```

8. Запрос итогов фискализации FF09H:

```
public void fsReadFiscalization(FSReadFiscalization command) throws
JposException {
    executeCommand(command);
}
```

9. Найти фискальный документ по номеру FF0AH

```
public void fsFindDocument(FSFindDocument command) throws JposException {
    executeCommand(command);
}
```

10. Открыть смену в ФН FF0BH

```
public void fsOpenDay(FSOpenDay command) throws JposException {
    executeCommand(command);
}
```

11. Передать произвольную TLV структуру FF0CH

```
public void fsWriteTLV(FSWriteTLV command) throws JposException {
    executeCommand(command);
}
```

12. Операция со скидками и надбавками FF0DH

```
public void fsSale(FSSale command) throws JposException {
    executeCommand(command);
}
```

13. Сформировать отчёт о перерегистрации ККТ FF34H

```
public void fsRegistrationReport(FSRegistrationReport command) throws
JposException {
    executeCommand(command);
}
```

14. Начать формирование чека коррекции FF35H

```
public void fsStartCorrectionReceipt(FSStartCorrectionReceipt command) throws
JposException {
    executeCommand(command);
}
```

15. Сформировать чек коррекции FF36H

```
public void fsPrintCorrectionReceipt(FSPrintCorrectionReceipt command) throws
JposException {
    executeCommand(command);
}
```

16. Начать формирование отчёта о состоянии расчётов FF37H

```
public void fsStartCalcReport(FSStartCalcReport command) throws JposException {
    executeCommand(command);
}
```

17. Сформировать отчёт о состоянии расчётов FF38H

```
public void fsPrintCalcReport(FSPrintCalcReport command) throws JposException {
    executeCommand(command);
}
```

18. Получить статус информационного обмена FF39H

```
public void fsReadCommStatus(FSReadCommStatus command) throws JposException {
    executeCommand(command);
}
```

19. Запрос квитанции о получении данных в ОФД по номеру документа FF3CH

```
public void fsReadDocTicket(FSReadDocTicket command) throws JposException {
    executeCommand(command);
}
```

20. Начать закрытие фискального режима FF3DH

```
public void fsStartFiscalClose(FSStartFiscalClose command) throws JposException
{
    executeCommand(command);
}
```

21. Закрыть фискальный режим FF3EH

```
public void fsPrintFiscalClose(FSPrintFiscalClose command) throws JposException
{
    executeCommand(command);
}
```

22. Запрос количества ФД на которые нет квитанции FF3FH

```
public void fsReadDocCount(FSReadDocCount command) throws JposException {
    executeCommand(command);
}
```

23. Запрос параметров текущей смены FF40H

```
public void fsReadDayParameters(FSReadDayParameters command) throws
JposException {
    executeCommand(command);
}
```

24. Начать открытие смены FF41H

```
public void fsStartDayOpen(FSStartDayOpen command) throws JposException {
    executeCommand(command);
}
```

25. Начать закрытие смены FF42H

```
public void fsStartDayClose(FSStartDayClose command) throws JposException {
    executeCommand(command);
}
```

26. Закрыть смену в ФН FF43H

```
public void fsDayClose(FSDayClose command) throws JposException {
    executeCommand(command);
}
```

27. Операция со скидками, надбавками и налогом FF44H

```
public void fsSale2(FSSale2 command) throws JposException {
    executeCommand(command);
}
```

28. Закрытие чека расширенное вариант №2 FF45H

```
public void fsCloseReceipt(FSCloseReceipt command) throws JposException {
    executeCommand(command);
}
```

XML файл Z отчета

Перед снятием Z отчета драйвер может сохранять значения денежных и операционных регистров ФР в файле с названием по умолчанию ZReport.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<ZReport>
    <Parameters DayNumber="4" FSSerial="9999078900006689"/>
    <CashRegisters>
        <CashRegister Number="0" Value="0"/>
        <CashRegister Number="1" Value="0"/>
        ...
        <CashRegister Number="255" Value="0"/>
    </CashRegisters>
    <OperationRegisters>
        <OperationRegister Number="0" Value="0"/>
        <OperationRegister Number="1" Value="0"/>
        ...
        <OperationRegister Number="252" Value="0"/>
    </OperationRegisters>
    <FMTotals>
        <AllFiscalizations Buy="0" RetBuy="0" RetSale="0" Sales="0"/>
        <LastFiscalization Buy="0" RetBuy="0" RetSale="0" Sales="0"/>
    </FMTotals>
    <FSCalcReport DocumentDate="08.12.2017" DocumentNumber="1"
        DocumentTime="16:48:00" QueueSize="31"/>
</ZReport>
```

Описание тегов:

<CashRegisters> - денежные регистры ФР, значения регистров выводятся в копейках.

<OperationRegisters> - операционные регистры ФР, целые числа.

<FMTotals> - сумма записей ФП

<AllFiscalizations> - по всем фискализациям

<LastFiscalization> - по последней фискализации

<FSCalcReport> - статус информационного обмена с ОФД. QueueSize – количество документов в очереди, DocumentNumber – номер первого документа в очереди, DocumentDate – дата первого документа в очереди, DocumentTime - время первого документа в очереди

Описание регистров приведено в инструкции по эксплуатации ФР.

Денежные регистры

Денежные регистры – регистры в энергонезависимой памяти ККТ. Содержимое их можно запросить командой протокола, в которой указывается номер регистра. Состав денежных регистров:

Накопления в отделы по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке:

0…3 – 1;

4…7 – 2;

8…11 – 3;

12…15 – 4;

16…19 – 5;

20…23 – 6;

24…27 – 7;

28…31 – 8;

32…35 – 9;

36…39 – 10;

40…43 – 11;

44…47 – 12;

48…51 – 13;

52…55 – 14;

56…59 – 15;

60…63 – 16.

64…67 – скидки по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке;

68…71 – надбавки по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке;
Накопления по видам оплаты по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке:
72…75 – наличными;
76…79 – видом оплаты 2;
80…83 – видом оплаты 3;
84…87 – видом оплаты 4;
Обороты по налогам по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке:
88…91 – А;
92…95 – Б;
96…99 – В;
100…103 – Г;
Налоги по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке:
104…107 – А;
108…111 – Б;
112…115 – В;
116…119 – Г;
120 – наличность в кассе на момент закрытия чека;
Накопления в отделы по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену:
121…124 – 1;
125…128 – 2;
129…132 – 3;
133…136 – 4;
137…140 – 5;
141…144 – 6;
145…148 – 7;
149…152 – 8;
153…156 – 9;
157…160 – 10;
161…164 – 11;
165…168 – 12;
169…172 – 13;
173…176 – 14;
177…180 – 15;
181…184 – 16.
185…188 – скидки по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену;
189…192 – надбавки по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену;
Накопления по видам оплаты по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену:
193…196 – наличными;
197…200 – видом оплаты 2;
201…204 – видом оплаты 3;
205…208 – видом оплаты 4;
Обороты по налогам по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену:
209…212 – А;
213…216 – Б;
217…220 – В;

221…224 – Г;
Налоги по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в смене:
225…228 – А;
229…232 – Б;
233…236 – В;
237…240 – Г;
241 – накопление наличности в кассе;
242 – накопление внесений за смену;
243 – накопление выплат за смену;
244 – не используется;
245 – не используется;
246 – не используется;
247 – не используется;
248 – не используется;
249 – сумма аннулированных продаж в смене;
250 – сумма аннулированных покупок в смене;
251 - сумма аннулированных возвратов продаж в смене;
252 - сумма аннулированных возвратов покупок в смене;
253 – не используется;
254 – не используется;
255 – не используется;
***Примечание**: ККТ осуществляет округление всех денежных сумм с точностью до двух знаков после запятой (до копеек) согласно математическим правилам (например, рассчитанное число «10,234» будет округлено до «10,23», а число «10,345» - до «10,35»).*

Операционные регистры

Операционные регистры – регистры в энергонезависимой памяти ККТ, служащие для подсчета количества различных операций в ККТ. Содержимое их можно запросить командой протокола, в которой указывается номер регистра. Состав операционных регистров:
Количество торговых операций в отделы по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке:
0…3 – 1;
4…7 – 2;
8…11 – 3;
12…15 – 4;
16…19 – 5;
20…23 – 6;
24…27 – 7;
28…31 – 8;
32…35 – 9;
36…39 – 10;
40…43 – 11;
44…47 – 12;
48…51 – 13;
52…55 – 14;
56…59 – 15;
60…63 – 16.
64…67 – количество скидок по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке;
68…71 – количество надбавок по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) в чеке;
Количество торговых операций в отделы по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену:
72…75 – 1;

76…79 – 2;
80…83 – 3;
84…87 – 4;
88…91 – 5;
92…95 – 6;
96…99 – 7;
100…103 – 8;
104…107 – 9;
108…111 – 10;
112…115 – 11;
116…119 – 12;
120…123 – 13;
124…127 – 14;
128…131 – 15;
132…135 – 16.
136…139 – количество скидок по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену;
140…143 – количество надбавок по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену;
144…147 – количество чеков по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода) за смену;
148…151 – номер чека по 4 типам торговых операций (приход, расход, возврат прихода, возврат расхода);
152 – сквозной номер документа;
153 – количество внесений денежных сумм за смену;
154 – количество выплат денежных сумм за смену;
155 – номер внесения денежных сумм;
156 – номер выплаты денежных сумм;
157 – количество отмененных документов;
158 – номер сменного отчета без гашения;
159 – номер сменного отчета с гашением до фискализации;
160 – номер общего гашения;
161 – номер полного фискального отчета;
162 – номер сокращенного фискального отчета;
163 – номер тестового прогона;
164 – номер снятия показаний операционных регистров;
165 – номер отчетов по секциям;
166 – количество аннулирований;
167 – количество запусков теста самодиагностики;
168 – не используется;
169 – не используется;
170 – не используется;
171 – не используется;
172 – не используется;
173 – не используется;
174 – не используется;
175 – не используется;
176 – не используется;
177 – не используется;
178 – номер отчетов по налогам;
179 – количество аннулированных чеков продаж;
180 - количество аннулированных чеков покупок;
181 - количество аннулированных чеков возвратов продаж;

182 - количество аннулированных чеков возвратов покупок;
183 – количество нефискальных документов в день;
184 – количество отчетов в буфере отчетов;
185 – сквозной номер документа (младшее слово);
186 - сквозной номер документа (старшее слово);
187 – количество стационарных проверок ПО ФП;
188 – не используется;
189 - не используется;
190 – не используется;
191 - не используется;
192 - не используется;
193 - не используется;
194 - не используется;
195 - не используется;
196 - количество шагов мотора (младшее слово);
197 - количество шагов мотора (старшее слово);
198 - количество отрезов (младшее слово);
199 - количество отрезов (старшее слово);
200 - общее количество чеков коррекции прихода;
201 - общее количество чеков коррекции расхода;
202 - количество чеков коррекции прихода за смену;
203 - количество чеков коррекции расхода за смену;
204…255 – не используется.