Table of Contents

Change LOG

version 1811201459
Changed zfpdef name from OpenCreditNoteFromFDCustomerDB() to OpenCreditNoteWithFreeCustomerData()
Changed zfpdef name from OpenInvoiceFromFDCustomerDB() to OpenInvoiceWithFDCustomerDB()

version 1812051456
Parameter *OptionVATClass*: added new option **'*' – Forbidden**
Command with **zfpdef** ReadCurrentReceiptInfo: change parameter *VATinReceipt* values from – *0 – Yes, 1 – No*   to   **– 1 – Yes, 0 - No**
Command with **zfpdef** *Payment:* change delimiter before parameter *OptionChangeType* from *'*'* to **';'**

version 1812171521
No protocol changes

version 1812201208
Add command with zfpdef:ReadSentRcpInfoStatus for read last sent and first unsent receipts numbers and date time.

version 1901101636
Remove *URN* input field from commands with zfpdefs: OpenStornoReceipt, OpenCreditNoteWithFreeCustomerData, OpenCreditNoteWithFDCustomerDB
Adding a command for reading QR code data in the last issued receipt with zfpdef: ReadLastReceiptQRcodeData

version 1901311135
Adding commands for KL devices V2 with zfpdefs:
ProgPayment_Old, ReadPayments_Old, ReadDailyAvailableAmounts_Old, ReadDailyRA_Old, ReadDailyPO_Old, ReadDailyReceivedSalesAmounts_Old, ReadDailyReturnedChangeAmounts_Old, ReadDailyRAbyOperator_Old, ReadDailyPObyOperator_Old, ReadDailyReceivedSalesAmountsByOperator_Old, ReadDailyReturnedChangeAmountsByOperator_Old
Command with *zfpdef*: *OpenStornoReceipt* from parameter *StornoReason* is removed option *2 – Tax relief*
Commands with *zfpdefs*: *OpenCreditNoteWithFreeCustomerData* and *OpenCreditNoteWithFDCustomerDB,* parameter *StornoReason* is hardcoded with value '*2' – tax relief*
Command with zfpdef: *ReceivedOnAccount_PaidOut* Parameter length *OperPass* is changed from 4 symbols to 6 symbols.
Command with zfpdef: ReadCurrentReceiptInfo() changes positions of paramters *SubtotalAmountVAT4, SubtotalAmountVAT5, SubtotalAmountVAT6, SubtotalAmountVAT7.*

version 1902191535
Adding additional message response format in point 1.3
Adding parameter for printing RA/PO availability in command *ReceivedOnAccount*, parameter name *PrintAvailability*.

version 1903211145

Commands with *zfpdefs: SetDateTime(DateTime),* OpenStornoReceipt(), OpenCreditNoteWithFreeCustomerData(), OpenCreditNoteWithFDCustomerDB() are with added seconds in the related parameters. This will not confuse the workflow with V2 devices!

Command with *zfpdef*: *OpenStornoReceipt* from parameter *StornoReason* is added option *2 – Tax relief*

Commands with *zfpdefs*: *OpenCreditNoteWithFreeCustomerData* and *OpenCreditNoteWithFDCustomerDB,* parameter *StornoReason* is with added options values *- '0' – Operator error, - '1' – Goods Claim or Goods return*

1. COMMUNICATION PROTOCOL

The type of the protocol is Master / Slave. The communication session is always initiated by the Application Software. FD caries out the commands send by the software application and provides a feedback depending on the result. FD sends back an „Acknowledgement response" or „message response". All messages of the protocol are either packed or single-byte. FD supports communication standard RS232, USB, BT, TCP (WiFi, LAN).

Serial port adjustment parameters:

Speed: 115200 bit/s (or 19200, 38400, 57600 and 9600 if such is set for the FD)

8 bits word

No parity

1 stop bit

1.1. MESSAGE FORMAT FROM THE SOFTWARE APPLICATION TO THE FD

All messages except those described in section 4., sent to the FD by the PC have the following structure:

<STX><LEN><NBL><CMD><DATA…DATA><CS><CS><ETX>

The table below contains description of the field enclosed between the symbols **<** and **>**:

Field	No. of bytes	Value
STX	1	*Message start* – always **02h**
LEN	1	*Message length* (number of bytes including LEN, NBL, CMD, DATA) increased by **20h** i.e. a number in the 20h - 9Fh range
NBL	1	*Message number* increased by **20h** i.e. a number in the 20h - 9Fh range
CMD	1	*Command* - a number in the 20h - 7Fh range(see the description of commands)
DATA.. DATA	0 - 3902	*Additional data* – a group of data fields separated with the symbols **';'**, giving additional information needed for execution of the command (see the description of commands)
CS CS	2	*Checksum*, compiled as follows: 1) Operation XOR of all bytes from LEN to DATA inclusive = 0 .. FFh 2) Conversion of 2 bytes by adding 30h, for example: B5h -> 3Bh 35h
ETX	1	*End of message* – always **0Ah** (LF)

The texts data of the message is sent as ASCII text with code table cp1251 (Windows 1251).

1.2. MESSAGE FORMAT FROM THE FD TO THE SOFTWARE APPLICATION

There are several types of response depending on the message received.

1.2.1. Acknowledgement response

Positive acknowledgement – when package format is correct. It is sent when the command is acknowledged as well as when it is rejected (errors in the data sent (field <DATA…DATA>) or the command cannot be executed or the command is illegal depending on the current status of the FD indicated by the two status bytes). It is a package message with the following format:

<ACK><NBL><STE><STE><CS><CS><ETX>

Fields description:

Field	No bytes	Value
ACK	1	06h
NBL	1	*No. of message* = NBL of message related to receipt
STE STE	2	*2 error status-bytes*. A two-digit ASCII number. (see Table Errors)
CS CS	2	*Checksum*, compiled as follows: 1) Operation XOR on NBL STE и STE = 00h .. FFh 2) Conversion of 2 bytes by adding 30h, for example:B5h -> 3Bh 35h
ETX	1	0Ah (LF)

The two status-bytes are a two-digit ASCII number, in which the first digit provides information about the error in the FD, and the second one – about a command error.

Table Errors:

Byte value	FD errors	Byte value	Command errors
30	OK	30	OK
31	Out of paper, printer failure	31	Invalid command
32	Registers overflow	32	Illegal command
33	Clock failure or incorrect date&time	33	Z daily report is not zero
34	Opened fiscal receipt	34	Syntax error
35	Payment residue account	35	Input registers overflow
36	Opened non-fiscal receipt	36	Zero input registers
37	Registered payment but receipt is not closed	37	Unavailable transaction for correction
38	Fiscal memory failure	38	Insufficient amount on hand
39	Incorrect password		
3a	Missing external display		
3b	24hours block – missing Z report		
3c	Overheated printer thermal head.		
3d	Interrupt power supply in fiscal receipt (one time until status is read)		
3e	Overflow EJ		
3f	Insufficient conditions		

A two-digit number is compiled depending on the type of error.
Example: Error 32 – Illegal command due to clock failure

Negative acknowledgement – It is sent when the package format is incorrect. It is 1-byte **NACK = 15h** without checksum.

Repetition request – It is sent when the FD is busy executing the preceding command. It is 1 byte **RETRY = 0Eh** without checksum.

1.2.2. Message response

It has the format of the packed message sent by the SA to the FD (see 3.1.) but is returned by the FD to the SA and contains information – response to the query (see description of commands).

1.3. SHORT MESSAGES FOR TESTING THE STATUS OF THE FD

The exchange protocol includes two unpacked single-byte codes for testing the status of the FD, which can quickly determine the status of the device. The two codes and their meaning are shown in the table below:

Query SA	Response FD		Meaning
04	04		FD is on
09	40	Bit.0	FD is READY
	41		FD is busy
	42	Bit.1	FD out of paper
	43		FD out of paper and busy
	44	Bit.2	FD printer is overheated
	45		FD printer is overheated and busy
	48	Bit.3	FD missing external display
	49		FD missing external display and busy
	50		FD is waiting for password (LAN or WiFi connection only)
	60		FD is already busy with another connection (LAN or WiFi connection only)
	70		Wrong password (LAN or WiFi connection only)

The format of the commands is described in art. 2. **DESCRIPTION OF THE COMMANDS OF FISCAL DEVICE**

2. DESCRIPTION OF THE COMMANDS
2.1. FORMAT AND PRESENTATION OF COMMANDS

All commands are described and presented using the following terms and symbols:

Key terms:

Command – the value of the CMD field of the message sent by the software application and in the message response of the the FD.

input – structure of the fields included in the DATA field of the message sent by the software application.

output – for each command it may be one of the following:

- Acknowledgement response.
- Structure of the fields included in the DATA field of the message response sent by the FD.

Input data – description of the contents of the "input" fields.
Output data – description of the contents of the "output" fields.

Key symbols:

' '	– compulsory symbol
" "	– compulsory DateTime format
< >	– compulsory data field
<;>	– field separator
[ ]	– field length
{ }	– non-compulsory data field
zfpdef:	- function name in the generated source code and file server file name

General rules:

Format of the price/value field – from 1 to 10 symbols, a floating decimal point number, preceded by +, - or SPACE.

Examples: -12.34   +56.7   8

Format of the quantity field – from 1 to 10 symbols, a floating decimal point number, up to three digits after the decimal point.

Examples: 1.234   56.78   9

Format of the rate (percentage) field – from 2 to 7 symbols, a floating decimal point number, up to two digits after the decimal point, preceded by the percent symbol - %.

Examples: -12.34%   +5.67%   8.9%   10%

Payment Number 0 corresponds to the main payment – IN CASH, payment Number 11 corresponds to the special currency payment - VAT account.

2.2. GENERAL COMMANDS

These are commands for the general functions of the FD, related to obtaining diagnostic information and to direct access to some of the functions of the device (paper feeding, paper cutting, and display visualization).

2.2.1. Command: 20h / SP - Status

input: n. a.
output: *<StatusBytes[7]>*
FPR operation: Provides detailed 7-byte information about the current status of the fiscal printer.
Input data : n. a.
Output data :

N byte	N bit	status flag
ST0	0	FM Read only
	1	Power down in opened fiscal receipt
	2	Printer not ready - overheat
	3	DateTime not set
	4	DateTime wrong
	5	RAM reset
	6	Hardware clock error
	7	1

N byte	N bit	status flag
ST1	0	Printer not ready - no paper
	1	Reports registers Overflow
	2	Customer report is not zeroed
	3	Daily report is not zeroed
	4	Article report is not zeroed
	5	Operator report is not zeroed
	6	Duplicate printed
	7	1

N byte	N bit	status flag
ST2	0	Opened Non-fiscal Receipt
	1	Opened Fiscal Receipt
	2	Opened Fiscal Detailed Receipt
	3	Opened Fiscal Receipt with VAT
	4	Opened Invoice Fiscal Receipt
	5	SD card near full
	6	SD card full
	7	1

ST3	0	No FM module
	1	FM error
	2	FM full
	3	FM near full
	4	Decimal point (1=fract, 0=whole)
	5	FM fiscalized
	6	FM produced
	7	1

ST4	0	Printer: automatic cutting
	1	External display: transparent display
	2	Speed is 9600
	3	reserve
	4	Drawer: automatic opening
	5	Customer logo included in the receipt
	6	reserve
	7	1

ST5	0	Wrong SIM card
	1	Blocking 3 days without mobile operator
	2	No task from NRA
	3	reserved
	4	reserved
	5	Wrong SD card
	6	Deregistered
	7	1

ST6	0	No SIM card
	1	No GPRS Modem
	2	No mobile operator
	3	No GPRS service
	4	Near end of paper
	5	Unsent data for 24 hours
	6	reserved
	7	1

zfpdef: ReadStatus()

2.2.2. Command: 21h / ! – Version

input: n. a.
output: *<DeviceType[1..2]> <;> <CertificateNum[6]> <;> <CertificateDateTime "DD-MM-YYYY HH:MM"> <;> <Model[50]> <;> <Version[20]>*
FPR operation: Provides information about the device type, Certificate number, Certificate date and time and Device model.
Input data : n. a.
Output data :

DeviceType	1 or 2 symbols for type of fiscal device:
	- '1' – ECR
	- '11' – ECR for online store
	- '2' – FPr
	- '21' – FPr for online store
	- '3' – Fuel
	- '31' – Fuel system
	- '5' – for FUVAS device
CertificateNum	6 symbols for Certification Number of device model
CertificateDateTime	16 symbols for Certificate Date and time parameter in format: DD-MM-YYYY HH:MM
Model	Up to 50 symbols for Model name
Version	Up to 20 symbols for Version name and Check sum

zfpdef: ReadVersion()

2.2.3. Command: 22h / " – Diagnostics

input: n. a.
output: ACK
FPR operation: Prints out a diagnostic receipt.
Input data : n. a.
Output data : n. a.
zfpdef: PrintDiagnostics()

2.2.4. Command: 24h / $ – Clear display

input: n. a.
output: ACK
FPR operation: Clears the external display.
Input data : n. a.
Output data : n. a.
zfpdef: ClearDisplay()

2.2.5. Command: 25h / % – Display text line 1

input: *<Text[20]>*
output: ACK
FPR operation: Shows a 20-symbols text in the upper external display line.
Input data :

Text	20 symbols text

Output data: n. a.
zfpdef: DisplayTextLine1(Text)

2.2.6. Command: 26h / & – Display text line 2

input: *<Text[20]>*
output: ACK
FPR operation: Shows a 20-symbols text in the lower external display line.
Input data :
 Text   20 symbols text
Output data: n. a.
zfpdef: DisplayTextLine2(Text)

2.2.7. Command: 27h / '– Display text lines 1 and 2

input: *<Text[40]>*
output: ACK
FPR operation: Shows a 20-symbols text in the first line and last 20-symbols text in the second line of the external display lines.
Input data :
 Text   40 symbols text
Output data: n. a.
zfpdef: DisplayTextLines1and2(Text)

2.2.8. Command: 28h / (– Display date and time

input: n. a.
output: ACK
FPR operation: Shows the current date and time on the external display.
Input data : n. a.
Output data : n. a.
zfpdef: DisplayDateTime()

2.2.9. Command: 29h /) – Cut paper, FP only

input: n. a.
output: ACK
FPR operation: Start paper cutter. The command works only in fiscal printer devices**.**
Input data : n. a.
Output data : n. a.
zfpdef: CutPaper()

2.2.10. Command: 2Ah / * – Cash drawer opening

input: n. a.
output: ACK
FPR operation: Opens the cash drawer.
Input data : n. a.
Output data : n. a.
zfpdef: CashDrawerOpen()

2.2.11. Command: 2Bh / + – Paper feeding

input: n. a.
output: ACK
FPR operation: Feeds one line of paper.
Input data : n. a.
Output data : n. a.
zfpdef: PaperFeed()

2.3. FISCAL COMMANDS

These are commands requiring data recording in the fiscal memory of the device. Password access is required.

2.3.1.1. Command: 56h / V – Set type of fiscal device, Option T

input: *<'T'> <;> <FDType[1]> <;> <Password[3]>*
output: *ACK*
FPR operation: Define Fiscal device type. The command is allowed only in non-fiscal mode, before fiscalization and after deregistration before the next fiscalization.
The type of device can be read by Version command 0x21.
Input data :

'T'   1 symbol for option 'T'
FDType   1 symbol for fiscal device type with value:
      - '0' - FPr for Fuel type 3
      - '1' – Main FPr for Fuel system type 31
      - '2' - ECR for online store type 11
      - '3' - FPr for online store type 21
      - '' – reset default type*
Password   3-symbols string
Output data: n. a.
zfpdef: SetFiscalDeviceType(FDType, Password)

2.3.1.2. Command: 41h / A – Set customer UIC information, Option 1

input: *<Password[6]> <;> <'1'> <;> <UIC[13]> <;> <UICType[1]>*
output: ACK
FPR operation: Stores the Unique Identification Code (UIC) and UIC type into the operative memory.
Input data :

Password   6-symbols string
'1'   One symbol with value 1
UIC   13 symbols for UIC
UICType   1 symbol for type of UIC number*:*
      - '0' - Bulstat
      - '1' - EGN
      - '2' – Foreigner Number
      - '3' – NRA Official Number
Output data: n. a.
zfpdef: SetCustomerUIC(Password, UIC, UICType)

2.3.1.3. Command: 41h / A – Confirm fiscalization, Option 2

input: *<Password[6]> <;> <'2'>*
output: ACK
FPR operation: Confirm Unique Identification Code (UIC) and UIC type into the operative memory.
Input data :

Password   6-symbols string
'2'   One symbol with value '2'
Output data: n. a.
zfpdef: ConfirmFiscalization(Password)

2.3.2. Command: 42h / B – Change VAT rates

input: *<Password[6]> <;> <VATrate0[6]> <;> <VATrate1[6]> <;> <VATrate2[6]> <;>*
<VATrate3[6]> <;> <VATrate4[6]><;> <VATrate5[6]><;> <VATrate6[6]> <;> <VATrate7[6]>
output: ACK
FPR operation: Stores a block containing the values of the VAT rates into the fiscal memory. Print the values on the printer.

Input data :

Password	6-symbols string
VATrate0	Value of VAT rate A from 6 symbols in format ##.##
VATrate1	Value of VAT rate Б from 6 symbols in format ##.##
VATrate2	Value of VAT rate В from 6 symbols in format ##.##
VATrate3	Value of VAT rate Г from 6 symbols in format ##.##
VATrate4	Value of VAT rate Д from 6 symbols in format ##.##
VATrate5	Value of VAT rate Е from 6 symbols in format ##.##
VATrate6	Value of VAT rate Ж from 6 symbols in format ##.##
VATrate7	Value of VAT rate З from 6 symbols in format ##.##

Output data: n. a.
zfpdef: ProgVATrates(Password, VATrate0, VATrate1, VATrate2, VATrate3, VATrate4, VATrate5, VATrate6, VATrate7)

2.3.3. Command: 43h / C – Change decimal point position

input: *<Password[6]> <;> <DecimalPointPosition[1]>*
output: ACK
FPR operation: Stores a block containing the number format into the fiscal memory. Print the current status on the printer.

Input data :

Password	6-symbols string
DecimalPointPosition	1 symbol with values:
	- '0'- Whole numbers
	- '2' - Fractions

Output data: n. a.
zfpdef: ProgDecimalPointPosition(Password, DecimalPointPosition)

2.4. PROGRAMMING COMMANDS

Set of commands, for programming the FD configuration according to the POS requirements and the user's needs.

2.4.1. Command: 44h / D – Program payment types

input: *<PaymentNum[1]> <;> <Name[10]> { <;> <Rate[10]> }*
output: ACK
FPR operation: Preprogram the name of the payment type.

Input data :

PaymentNum	1 symbol for payment type
	- '9' – Payment 9
	- '10' – Payment 10
	- '11' – Payment 11
Name	10 symbols for payment type name
Rate	(Exchange Rate) 10 symbols for exchange rate in format: ####.#####
	of the 11th payment type, maximal value 0420.00000

Output data: n. a.
zfpdef: *ProgPayment(PaymentNum, Name, PaymentRate)*

2.4.2. Command: 44h / D – Program payment types, KL

input: *<Number[1]><;> <Name[10]>{<;><Rate[10]><;><CodePayment[1]>}*
output: ACK
FPR operation: Preprogram the name of the type of payment. Command works for KL version 2 devices.

Input data :

Number	1 symbol for payment type
	- '1' – Payment 1
	- '2' – Payment 2
	- '3' – Payment 3
	- '4' – Payment 4
Name	10 symbols for payment type name
Rate	(Exchange Rate) 10 symbols for exchange rate in format: ####.#####
	of the 4th payment type, maximal value 0420.00000
CodePayment	1 symbol for code payment type with name:
	- '1' – Check
	- '2' – Talon
	- '3' – V. Talon
	- '4' – Packaging
	- '5' – Service
	- '6' – Damage
	- '7' – Card
	- '8' – Bank
	- '9' – Programming Name1
	- ':' – Programming Name2

Output data: n. a.
Note: For Payment type 4 code payment type cannot be selected
zfpdef: *ProgPayment_Old(Number, Name, PaymentRate, CodePayment)*

2.4.3. Command: 45h / E – Program parameters

input: *<POSNum[4]> <;> <PrintLogo[1]> <;> <AutoOpenDrawer[1]> <;>*
<AutoCut[1]> <;> <ExternalDispManagement[1]> <;> <ArticleReportType[1]> <;>
<EnableCurrency[1]> <;> <EJFontType[1]> <;> <reserved['0']> <;>
<WorkOperatorCount[1]>

output: ACK

FPR operation: Programs the number of POS, printing of logo, cash drawer opening, cutting permission, external display management mode, article report type, enable or disable currency in receipt, EJ font type and working operators counter.

Input data :

POSNum	(POS Number) 4 symbols for number of POS in format ####
PrintLogo	(Print Logo) 1 symbol of value:
	- '1' - Yes
	- '0' - No
AutoOpenDrawer	(Auto Open Drawer) 1 symbol of value:
	- '1' - Yes
	- '0' - No
AutoCut	(Auto Cut) 1 symbol of value:
	- '1' - Yes
	- '0' - No
ExternalDispManagement	(Ext. Display Management) 1 symbol of value:
	- '1' - Manuel
	- '0' - Auto
ArticleReportType	(Article Report) 1 symbol of value:
	- '1' - Detailed
	- '0' - Brief
EnableCurrency	(Enable Currency) 1 symbol of value:
	- '1' – Yes
	- '0' – No
EJFontType	(EJ Font) 1 symbol of value:
	- '1' - Low Font
	- '0' - Normal Font
reserved	1 symbol reserved '0'
WorkOperatorCount	(Work Operator Count) 1 symbol of value:
	- '1' - One
	- '0' - More

Output data: n. a.

Notes:

The logo is a graphical file in **BMP** format with dimensions 384x80 / 448 X 160 /576x80 points, which is printed at the head of every receipt

"ExternalDispManagement" is a mode, in which the FD does not send information to the display except when executing the 25h, 26h and 27h commands. When this mode is off the FD "uses" the display to show data during sales, at receipt finalization, etc.

zfpdef: ProgParameters(POSNum, *PrintLogo, AutoOpenDrawer, AutoCut, ExternalDispManagement, EJFormat, ArticleRepType, EnableCurrency, EJFontType, WorkOperatorCount*)

2.4.4. Command: 47h / G – Program department

input: *<Number[2]> <;> <Name[20]> <;> <OptionVATClass[1]> { <;> <Price[1..10]> <;> <OptionDepPrice[1]> <;> <AdditionalName[14]> }*

output: ACK

FPR operation: Set data for the state department number from the internal FD database. Parameters *Price*, *OptionDepPrice* and *AdditionalName* are not obligatory and require the previous not obligatory parameter.

***Input data**:*

Number	2 symbols department number in format ##
Name	20 characters department name
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Price	Up to 10 symbols for department price
OptionDepPrice	1 symbol for Department price flags with next value:
	- '0' - Free price disabled
	- '1' - Free price enabled
	- '2' - Limited price
	- '4' - Free price disabled for single transaction
	- '5' - Free price enabled for single transaction
	- '6' - Limited price for single transaction
AdditionalName	14 characters additional department name

Output data : n. a.

Note:

When changing the VAT group attachment of department must actualize the VAT groups of all articles attached to this department. Otherwise they won't be accessible for sale.

zfpdef: *ProgDepartment(Number, Name, OptionVATClass, Price, OptionDepPrice, AdditionalName)*

2.4.5. Command: 48h / H – Program date and time

input: *<DateTime "DD-MM-YY HH:MM:SS">*

output: ACK

FPR operation: Sets the date and time and prints out the current values.

Input data :

DateTime	Date Time parameter in format: *DD-MM-YY HH:MM:SS*

Output data : n. a.

zfpdef: *SetDateTime(DateTime)*

2.4.6. Command: 49h / I – Program display greeting message

input:<*'0'> <;> <DisplayGreetingText[20]*>
output: ACK
 FPR operation: Program the contents of a Display Greeting message.

Input data :
 '0'   1 symbol with value '0'
 DisplayGreetingText   20 symbols for Display greeting message
Output data: n. a.
zfpdef:*ProgDisplayGreetingMessage(DisplayGreetingText)*

2.4.7. Command: 49h / I – Program header lines

input:<*OptionHeaderLine[1]> <;> <HeaderText[TextLength]*>
output: ACK
 FPR operation: Program the contents of a header lines.

Input data :
 OptionHeaderLine   (Line Number) 1 symbol with value:
                          - '1' – Header 1
                          - '2' – Header 2
                          - '3' – Header 3
                          - '4' – Header 4
                          - '5' – Header 5
                          - '6' – Header 6
                          - '7' – Header 7
 HeaderText   TextLength symbols for header lines
Output data: n. a.
zfpdef:*ProgHeader(OptionHeaderLine, HeaderText)*

2.4.8. Command: 49h / I – Program footer line

input:<*'8'> <;> <FooterText[TextLength]*>
output: ACK
 FPR operation: Program the contents of a footer lines.

Input data :
 '8'   1 symbol with value '8'
 FooterText   TextLength symbols for footer line
Output data: n. a.
zfpdef:*ProgFooter(FooterText)*

2.4.9. Command: 49h / I – Program header UIC prefix

input:<*'9'> <;> <HeaderUICprefix[12]*>
output: ACK
 FPR operation: Program the content of the header UIC prefix.

Input data :
 '9'   1 symbol with value '9'
 HeaderUICprefix   12 symbols for header UIC prefix
Output data: n. a.
zfpdef: *ProgHeaderUICprefix(HeaderUICprefix)*

2.4.10. Command: 4Ah / J – Program operator's name and password

input: *<Number[1..2]> <;> <Name[20]> <;> <Password[6]>*
output: ACK
FPR operation: Programs the operator's name and password.

Input data :
Number	Symbols from '1' to '20' corresponding to operator's number
Name	20 symbols for operator's name
Password	6 symbols for operator's password

Output data : n. a.
zfpdef: ProgOperator(Number, Name, Password)

2.4.11. Command: 4Bh / K – Program article

input: *<PLUNum[5]> <;> <Name[20]> <;> <Price[1..10]> <;> <OptionVATClass[1]> <;> <BelongToDepNum[1..2]>*
output: ACK
FPR operation: Programs the data for a certain article (item) in the internal database. The price may have variable length, while the name field is fixed.

Input data :
PLUNum	5 symbols for article number in format: #####
Name	20 symbols for article name
Price	Up to 10 symbols for article price
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
BelongToDepNum	*(Department Number) BelongToDepNum* + 80h, 1 symbol for article department attachment, formed in the following manner: *BelongToDepNum* [HEX] + 80h example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h

Output data: n. a.
Notes:
When programming department attachment, FD checks whether the corresponding department is attached to same VAT group. In case they don't match no changes will be applied. Programming of value 0 (no department attachment) is possible any time.
If no number is entered in the field of department attachment the command will execute with value 0 (no department attachment).
zfpdef: ProgPLU_Old(PLUNum, Name, Price, OptionVATClass, BelongToDepNum)

2.4.12. Command: 4Bh / K – Program article general registers, Option 1

input: *<PLUNum[5]> <;> <Option['#@1+$']> <;> <Name[34]> <;> <Price[1..10]> <;> <OptionPrice[1]> <;> <OptionVATClass[1]> <;> <BelongToDepNum[1..2]> <;> <SingleTransaction[1]>*

output: ACK

FPR operation: Programs the general data for a certain article in the internal database. The price may have variable length, while the name field is fixed.

Input data :

PLUNum	5 symbols for article number in format: #####
Option	5 symbols with value '#@1+$'
Name	34 symbols for article name
Price	Up to 10 symbols for article price
OptionPrice	1 symbol for price flag with next value:
	- '0'- Free price is disable valid only programmed price
	- '1'- Free price is enable
	- '2'- Limited price
OptionVATClass	*1 character for VAT class:*
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '' - Forbidden*
BelongToDepNum	*(Department Number) BelongToDepNum + 80h, 1 symbol for article department attachment, formed in the following manner: BelongToDepNum[HEX] + 80h example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h*
SingleTransaction	1 symbol with value:
	- '0' – Inactive, default value
	- '1' - Active Single transaction in receipt

Output data: n. a.

zfpdef: ProgPLUgeneral(PLUNum, Name, Price, OptionPrice, OptionVATClass, BelongToDepNum, SingleTransaction)

2.4.13. Command: 4Bh / K – Program article quantity in stock, Option 2

input: *<PLUNum[5]> <;> <Option['#@2+$']> <;> <AvailableQuantity[1..11]> <;> <OptionQuantityType[1]>*

output: ACK

FPR operation: Programs available quantity and Quantiy type for a certain article in the internal database.

Input data :

PLUNum	5 symbols for article number in format: #####
Option	5 symbols with value '#@2+$'
AvailableQuantity	(Available Quantity) Up to 11 symbols for available quantity in stock
OptionQuantityType	1 symbol for Quantity flag with next value:
	- '0'- Availability of PLU stock is not monitored
	- '1'- Disable negative quantity
	- '2'- Enable negative quantity

Output data: n. a.

zfpdef: ProgPLUqty(PLUNum, AvailableQty, OptionQuantityType)

2.4.14. Command: 4Bh / K – Program article barcode, Option 3

input: *<PLUNum[5]> <;> <Option['#@3+$']> <;> <Barcode[13]>*
output: ACK
FPR operation: Programs Barcode of article in the internal database.

Input data :

PLUNum	5 symbols for article number in format: #####
Option	5 symbols with value '#@3+$'
Barcode	13 symbols for barcode

Output data: n. a.
zfpdef: ProgPLUbarcode(PLUNum, Barcode)

2.4.15. Command: 4Bh / K – Program article price, Option 4

input: *<PLUNum[5]> <;> <Option['#@4+$']> <;> <Price[1..10]> <;> <OptionPrice[1]>*
output: ACK
FPR operation: Programs price and price type for a certain article in the internal database.

Input data :

PLUNum	5 symbols for article number in format: #####
Option	5 symbols with value '#@4+$'
Price	Up to 10 symbols for article price
OptionPrice	1 symbol for price flag with next value:
	- '0'- Free price is disable valid only programmed price
	- '1'- Free price is enable
	- '2'- Limited price

Output data: n. a.
zfpdef: ProgPLUprice(PLUNum, Price, OptionPrice)

2.4.16. Command: 4Bh / K – Erase all PLU database

input: *<PLUNum['00000']> <;> <Option['#@$+$']> <;> <Password[6]>*
output: ACK
FPR operation: Erase all articles in PLU database.

Input data :

PLUNum	5 symbols '00000'
Option	5 symbols with value '#@$+$'
Password	6 symbols for password

Output data: n. a.
zfpdef: EraseAllPLUs(Password)

2.4.17. Command: 4Ch / L – Program logo without setting a number (default number 0)

input: *<BMPfile[9022]>*
output: ACK
FPR operation: Stores in the memory the graphic file under number 0. Prints information about loaded in the printer graphic files.

Input data:

BMPfile	*BMP file with fixed size 9022 bytes

Output data: n. a.
Notes:
FD has the ability to store up to 10 different BMP files for logo with numbers from 0 to 9, as one of them is „active" and is printed as receipt's logo. If there is no file loaded under the number, stated as „active", FD will work as set for work without logo.
zfpdef: ProgLogo(BMPfile)

2.4.18. Command: 4Dh / M – Program logo with setting a number

> **input:** *<LogoNumber[1]> <BMPfile[3902/9022/5822]>*
> **output: ACK**
> **FPR operation:** Stores in the memory the graphic file under stated number. Prints information about loaded in the printer graphic files.

Input data:

LogoNumber	1 character value from '0' to '9' setting the number where the logo will be saved.
BMPfile	BMP file with fixed size 3902/9022/5822 bytes

Output data: n. a.
zfpdef: ProgLogoNum(LogoNumber, BMPfile)

2.4.19. Command: 23h / # – Program active logo number

> **Input***: <LogoNumber[1]>*
> **output: ACK**
> **FPR operation:** Sets logo number, which is active and will be printed as logo in the receipt header. Print information about active number.

Input data:

LogoNumber	1 character value from '0' to '9' or '?'. The number sets the active file, and the '?' invokes only printing of information

Output data: n. a.
zfpdef: SetActiveLogoNum(LogoNumber)

2.4.20. Command: 51h / Q – Print barcode 'QP'

> **input:** *<'P'> <;> <CodeType[1]> <;> <CodeLen[1..2]> <;> <CodeData[100]>*
> **output: ACK**
> **FPR Operation:** Prints barcode from type stated by CodeType and CodeLen and with data stated in CodeData field.

Input data:

'P'	1 character 'P'
CodeType	1 symbol with possible values:
	- '0' – UPC A
	- '1' – UPC E
	- '2' – EAN 13
	- '3' – EAN 8
	- '4' – CODE 39
	- '5' – ITF
	- '6' – CODABAR
	- 'H' – CODE 93
	- 'I' – CODE 128
CodeLen	Up to 2 bytes for number of bytes according to the table
CodeData	Up to 100 bytes data in range according to the table

Output data: n.a.

Table:

Barcode type	<CodeType>	<CodeLen>	Range of <CodeData>
UPC-A	**'0'** or **'A'**	11 or 12	Digits from **'0'** to **'9'**
UPC-E	**'1'** or **'B'**	11 or 12	Digits from **'0'** to **'9'**
JAN13 (EAN13)	**'2'** or **'C'**	12 or 13	Digits from **'0'** to **'9'**
JAN8 (EAN8)	**'3'** or **'D'**	7 or 8	Digits from **'0'** to **'9'**

CODE 39	'4' or 'E'	from 1 to 10	Characters: 'SP' '$' '%' '+' '-' '.' '/' Digits from '0' to '9' letters from 'A' to 'Z'
ITF	'5' or 'F'	from 2 to 18 (evens only)	Digits from "0' to '9'
CODABAR	'6' or 'G'	From 1 to 15	Characters: '$' '+' '-' '/' digits from '0' to '9' letters from 'A' to 'D'
CODE 93	'H'	From 1 to 14	Bytes from 0 to 7F
CODE 128	'I'	From 1 to 12	Bytes from 0 to 7F

The length restriction for some of the barcode types is because of the the print area not because of the barcode standard. If more data is sent the printed barcode may not be read correctly.

zfpdef: PrintBarcode(CodeType, CodeLen, CodeData)

2.4.21. Command: 50h / P – Program invoice number range

input: *<StartNum[10]> <;> <EndNum[10]>*
output: ACK
FPR Operation: Set invoice start and end number range. To execute the command is necessary to grand following condition: the number range to be spent, not used, or not set after the last RAM reset.

Input data:

StartNum	10 characters for start number in format: ##########
EndNum	10 characters for end number in format: ##########

Output data: n. a.

zfpdef: SetInvoiceRange(StartNumber, EndNumber)

2.4.22. Command: 52h / R – Program customer database

input: *<Option['P']> <;> <CustomerNum[4]> <;> <CustomerCompanyName[26]> <;> <CustomerFullName[16]> <;> <VATNumber[13]> <;> <UIC[13]> <;> <Address[30]> <;> <UICType[1]>*
output: ACK
FPR Operation: Program customer in FD data base.

Input data:

Option	1 symbol with value 'P'
CustomerNum	(Customer Number) 4 symbols for customer number in format ####
CustomerCompanyName	(Company name) 26 symbols for customer name
CustomerFullName	(Buyer Name) 16 symbols for Buyer name
VATNumber	13 symbols for VAT number on customer
UIC	13 symbols for customer Unique Identification Code
Address	30 symbols for address on customer
UICType	1 symbol for type of Unique Identification Code:
	- '0' - Bulstat
	- '1' - EGN
	- '2' – Foreigner Number
	- '3' – NRA Official Number

Output data: n. a.

zfpdef: ProgCustomerData(CustomerNum, CustomerFirmName, CustomerFullName, UIC, FiscNum, Address, UICType)

2.5. DATA READING COMMANDS
Set of commands for receiving information from the FD about programmed values as well as additional information.

2.5.1. Command: 60h / ' – Read FD numbers

input: n. a.
output: *<SerialNumber[8]> <;> <FMNumber[8]>*
FPR operation: Provides information about the manufacturing number of the fiscal device and FM number.
Input data : n. a.
Output data :

SerialNumber	8 symbols for individual number of the fiscal device
FMNumber	8 symbols for individual number of the fiscal memory

zfpdef: ReadSerialAndFiscalNums()

2.5.2. Command: 61h / a – Read registration information

input: n. a.
output: *<UIC[13]> <;> <UICType[1]><;> <NRARegistrationNumber[6]><;>*
<NRARegistrationDate "DD-MM-YYYY HH:MM" >
FPR operation: Provides information about the programmed VAT number, type of VAT number, register number in NRA and Date of registration in NRA.
Input data : n. a.
Output data :

UIC	13 symbols for Unique Identification Code
UICType	1 symbol for type of Unique Identification Code: - '0' - Bulstat - '1' - EGN - '2' – Foreigner Number - '3' – NRA Official Number
NRARegistrationNumber	Register number on the Fiscal device from NRA
NRARegistrationDate	Date of registration in NRA

zfpdef: ReadRegistrationInfo()

2.5.3. Command: 62h / b – Read VAT rates

input: n. a.
output: *<VATrate0[7]> <;> <VATrate1[7]> <;> <VATrate2[7]> <;> <VATrate3[7]>*
<;> <VATrate4[7]> <;> <VATrate5[7]> <;> <VATrate6[7]> <;> <VATrate7[7]>
FPR operation: Provides information about the current VAT rates which are the last values stored into the FM.
Input data : n. a.
Output data :

VATrate0	Value of VAT rate A from 7 symbols in format ##.##%
VATrate1	Value of VAT rate Б from 7 symbols in format ##.##%
VATrate2	Value of VAT rate В from 7 symbols in format ##.##%
VATrate3	Value of VAT rate Г from 7 symbols in format ##.##%
VATrate4	Value of VAT rate Д from 7 symbols in format ##.##%
VATrate5	Value of VAT rate Е from 7 symbols in format ##.##%
VATrate6	Value of VAT rate Ж from 7 symbols in format ##.##%
VATrate7	Value of VAT rate З from 7 symbols in format ##.##%

zfpdef: ReadVATrates()

2.5.4. Command: 63h / c – Read decimal point position

input: n. a.
output: *<DecimalPointPosition[1]>*
FPR operation: Provides information about the current (the last value stored into the FM) decimal point format.
Input data : n. a.
Output data:

DecimalPointPosition	1 symbol with values:
	   - '0'- Whole numbers
	   - '2' - Fractions

zfpdef: ReadDecimalPoint()

2.5.5. Command: 64h / d – Read payment types

input: n. a.
output: *<NamePayment0[10]> <;> <NamePayment1[10]> <;>*
<NamePayment2[10]> <;> <NamePayment3[10]> <;> <NamePayment4[10]> <;>
<NamePayment5[10]> <;> <NamePayment6[10]> <;> <NamePayment7[10]> <;>
<NamePayment8[10]> <;> <NamePayment9[10]> <;> <NamePayment10[10]> <;>
<NamePayment11[10]> <;> <ExchangeRate[10]>
FPR operation: Provides information about all programmed types of payment, currency name and currency exchange rate.
Input data : n. a.
Output data :

NamePayment0	10 symbols for payment name type 0
NamePayment1	10 symbols for payment name type 1
NamePayment2	10 symbols for payment name type 2
NamePayment3	10 symbols for payment name type 3
NamePayment4	10 symbols for payment name type 4
NamePayment5	10 symbols for payment name type 5
NamePayment6	10 symbols for payment name type 6
NamePayment7	10 symbols for payment name type 7
NamePayment8	10 symbols for payment name type 8
NamePayment9	10 symbols for payment name type 9
NamePayment10	10 symbols for payment name type 10
NamePayment11	10 symbols for payment name type 11
ExchangeRate	10 symbols for exchange rate of payment type 11 in format: ####.#####

zfpdef: ReadPayments()

2.5.6. Command: 64h / d – Read payment types, KL

input: n. a.

output: <NamePaym0[10]> <;> < NamePaym1[10]> <;> < NamePaym2[10]> <;> <NamePaym3[10]> <;> <NamePaym4[10]><;><ExRate[10]> <;> <CodePaym0[1]><;> <CodePaym1[1]><;> < CodePaym2[1]><;> <CodePaym3[1]><;> <CodePaym4[1]>

FPR operation: Provides information about all programmed types of payment. Command works for KL version 2 devices.

Input data : n. a.

Output data :

NamePaym0   10 symbols for payment name type 0
NamePaym1   10 symbols for payment name type 1
NamePaym2   10 symbols for payment name type 2
NamePaym3   10 symbols for payment name type 3
NamePaym4   10 symbols for payment name type 4
ExRate   10 symbols for exchange rate of payment type 4 in format: ####.#####
CodePaym0   1 symbol for code of payment 0 = 0xFF (currency in cash)
CodePaym1   1 symbol for code of payment 1 (default value is '7')
CodePaym2   1 symbol for code of payment 2 (default value is '1')
CodePaym3   1 symbol for code of payment 3 (default value is '2')
CodePaym4   1 symbol for code of payment 4 = 0xFF (currency in cash)

zfpdef: ReadPayments_Old()

2.5.7. Command: 65h / e – Read parameters

input: n. a.

output: <POSNum[4]> <;> <PrintLogo[1]> <;> <AutoOpenDrawer[1]> <;> <AutoCut[1]> <;> <ExternalDispManagement[1]> <;> <ArticleReportType[1]> <;> <EnableCurrency[1]> <;> <EJFontType[1]> <;> <reserved['0']> <;> <WorkOperatorCount[1]>

FPR operation: Provides information about the number of POS, printing of logo, cash drawer opening, cutting permission, display mode, article report type, Enable/Disable currency in receipt, EJ font type and working operators counter.

Input data : n. a.

Output data :

POSNum	(POS Number) 4 symbols for number of POS in format ####
PrintLogo	(Print Logo) 1 symbol of value: - '1' - Yes - '0' - No
AutoOpenDrawer	(Auto Open Drawer) 1 symbol of value: - '1' - Yes - '0' - No
AutoCut	(Auto Cut) 1 symbol of value: - '1' - Yes - '0' - No
ExternalDispManagement	(External Display Management) 1 symbol of value: - '1' - Manuel - '0' - Auto
ArticleReportType	(Article Report) 1 symbol of value: - '1' - Detailed - '0' - Brief
EnableCurrency	(Enable Currency) 1 symbol of value: - '1' – Yes - '0' – No

EJFontType	(EJ Font) 1 symbol of value:
	- '1' - Low Font
	- '0' - Normal Font
reserved	1 symbol reserved '0'
WorkOperatorCount	(Work Operator Count) 1 symbol of value:
	- '1' - One
	- '0' - More

zfpdef: ReadParameters()

2.5.8. Command: 66h / f – Read detailed printer status

   input: n. a.
   output: *<ExternalDisplay[1]> <;> <StatPRN[4]> <;> <FlagServiceJumper[1]>*
   FPR operation: Provides additional status information

Input data : n. a.
Output data :

ExternalDisplay	1 symbol – connection with external display
	- 'Y' - Yes
	- 'N' - No
StatPRN	4 symbols for detailed status of printer (only for printers with **ASB**)

N byte	N bit	status flag
ST0	0	Reserved
	1	Reserved
	2	Signal level for drawer
	3	Printer not ready
	4	Reserved
	5	Open cover
	6	Paper feed status
	7	Reserved

N byte	N bit	status flag
ST1	0	Reserved
	1	Reserved
	2	Reserved
	3	Cutter error
	4	Reserved
	5	Fatal error
	6	Overheat
	7	Reserved

N byte	N bit	status flag
ST2	0	JNP (journal paper near end)
	1	RNP (receipt paper near end)
	2	JPE (journal paper end)
	3	RPE (receipt paper end)
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

N byte	N bit	status flag
ST3	0	Print data buffer data exists

	1	Reserved
	2	Reserved
	3	Reserved
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

FlagServiceJumper   1 symbol with value:
                        - 'J' - Yes
                        - ' ' – No

zfpdef: ReadDetailedPrinterStatus()

2.5.9. Command: 67h / g – Read department registers

     input*: <DepNum[2]>*
     output: *<DepNum[2]> <;> <DepName[20]> <;> <OptionVATClass[1]> <;>*
<Turnover[1..13]> <;> <SoldQuantity[1..13]> <;> <LastZReportNumber[1..5]> <;>
 <LastZReportDate "DD-MM-YYYY HH:MM">

     FPR operation: Provides information for the programmed data, the turnover from the stated department number

Input data :
DepNum   (Department Number) 2 symbols for department number in format: ##
Output data :
DepNum   2 symbols for department number in format: ##
DepName   20 symbols for department name
OptionVATClass   1 character for VAT class:
                   - 'А' – VAT Class 0
                   - 'Б' – VAT Class 1
                   - 'В' – VAT Class 2
                   - 'Г' – VAT Class 3
                   - 'Д' – VAT Class 4
                   - 'Е' – VAT Class 5
                   - 'Ж' – VAT Class 6
                   - '3' – VAT Class 7
                   - '*' - Forbidden
Turnover   Up to 13 symbols for accumulated turnover of the article
SoldQuantity   Up to 13 symbols for sold quantity of the department
LastZReportNumber   Up to 5 symbols for the number of last Z Report
LastZReportDate   16 symbols for date and hour on last Z Report in format
                   "DD-MM-YYYY HH:MM"

zfpdef: ReadDepartment(*DepNum*)

2.5.10. Command: 67h / g – Read department registers, Option "(All)

input*: <DepNum[2]> <;> <reserved[""]>*
output*: <DepNum[2]> <;> <reserved[""]> <;> <DepName[34]> <;>*
<OptionVATClass[1]> <;> <Price[1..10]> <;> <OptionDepPrice[1]> <;>
<TurnoverAmount[1..13]> <;> <SoldQuantity[1..13]> <;> <StornoAmount[1..13]> <;>
<StornoQuantity[1..13]> <;> <LastZReportNumber[1..5]> <;>
<LastZReportDate "DD-MM-YYYY HH:MM">
FPR operation: Provides information for the programmed data, the turnovers from the stated department number

Input data :

DepNum	(Department Number) 2 symbols for department number in format: ##
Reserved	1 symbol with value '"'

Output data :

DepNum	2 symbols for department number in format: ##
reserved	1 symbol with value '"' quotation mark.
DepName	20 symbols for department name
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Price	Up to 10 symbols for department price
OptionDepPrice	1 symbol for Department flags with next value:
	- '0' - Free price disabled
	- '1' - Free price enabled
	- '2' - Limited price
	- '4' - Free price disabled for single transaction
	- '5' - Free price enabled for single transaction
	- '6' - Limited price for single transaction
TurnoverAmount	Up to 13 symbols for accumulated turnover of the article
SoldQuantity	Up to 13 symbols for sold quantity of the department
StornoAmount	Up to 13 symbols for accumulated storno amount
StornoQuantity	Up to 13 symbols for accumulated storno quantiy
LastZReportNumber	Up to 5 symbols for the number of last Z Report
LastZReportDate	16 symbols for date and hour on last Z Report in format "DD-MM-YYYY HH:MM"

zfpdef: ReadDepartmentAll(DepNum)

2.5.11. Command: 68h / h – Read date and time

input: n. a.
output: *<DateTime "DD-MM-YYYY HH:MM">*
FPR operation: Provides information about the current date and time.
Input data : n. a.
Output data :

DateTime	Date Time parameter in format: *DD-MM-YYYY HH:MM*

zfpdef: ReadDateTime()

2.5.12. Command: 69h / i – Read display greeting message

      input: *<'0'>*
      output: *<'0'> <;> <DisplayGreetingText[20]>*
      FPR operation: Provides the content of the Display Greeting message.

Input data :
'0'	1 symbol with value 0

Output data:
'0'	1 symbol with value 0
DisplayGreetingText	20 symbols for display greeting message

zfpdef: ReadDisplayGreetingMessage()

2.5.13. Command: 69h / i – Read header lines

      input: *<OptionHeaderLine[1]>*
      output: *<OptionHeaderLine[1]> <;> <HeaderText[TextLength]>*
      FPR operation: Provides the content of the header lines

Input data :
OptionHeaderLine	(Line Number) 1 symbol with value:
	- '1' – Header 1
	- '2' – Header 2
	- '3' – Header 3
	- '4' – Header 4
	- '5' – Header 5
	- '6' – Header 6
	- '7' – Header 7

Output data:
OptionHeaderLine	(Line Number) 1 symbol with value:
	- '1' – Header 1
	- '2' – Header 2
	- '3' – Header 3
	- '4' – Header 4
	- '5' – Header 5
	- '6' – Header 6
	- '7' – Header 7
HeaderText	TextLength symbols for header lines

zfpdef: ReadHeader(OptionHeaderLine)

2.5.14. Command: 69h / i – Read footer line

     input: *<'8'>*
     output: *<'8'> <;> <FooterText[TextLength]>*
     FPR operation: Provides the content of the footer line.

Input data :
 '8'   1 symbol with value '8'

Output data:
 '8'   1 symbol with value '8'
 FooterText   TextLength symbols for footer line
zfpdef: ReadFooter()

2.5.15. Command: 69h / i – Read Header UIC prefix

     input: *<'9'>*
     output: *<'9'> <;> <HeaderUICprefix[12]>*
     FPR operation: Provides the content of the header UIC prefix.

Input data :
 '9'   1 symbol with value '9'

Output data:
 '9'   1 symbol with value '9'
 HeaderUICprefix   *12* symbols for *Header UIC prefix*
zfpdef:*ReadHeaderUICPrefix()*

2.5.16. Command: 6Ah / j – Read operator's name and password

     input: *<Number[1..2]>*
     output: *<Number[1..2]> <;> <Name[20]> <;> <Password[6]>*
     FPR operation: Provides information about operator's name and password.

Input data :
Number   (Operator Number) Symbol from 1 to 20 corresponding to the number of operators.

Output data:
Number   Symbol from 1 to 20 corresponding to the number of operator
Name   20 symbols for operator's name
Password   6 symbols for operator's password
zfpdef: *ReadOperatorNamePassword(Number)*

2.5.17. Command: 6Bh / k – Read article

input: *<PLUNum[5]>*

output: *<PLUNum[5]> <;> <PLUName[20]> <;> <Price[1..11]> <;>
<OptionVATClass[1]> <;> <Turnover[1..13]> <;> <QuantitySold[1..13]> <;>
<LastZReportNumber[1..5]> <;> <LastZReportDate "DD-MM-YYYY HH:MM"> <;>
<BelongToDepNumber[1..2]>*

FPR operation: Provides information about the registers of the specified article.

Input data :

PLUNum	(PLU Number) 5 symbols for article number in format #####

Output data:

PLUNum	5 symbols for article number format #####
PLUName	20 symbols for article name
Price	Up to 11 symbols for article price
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Turnover	Up to 13 symbols for turnover by this article
QuantitySold	Up to 13 symbols for sold quantity
LastZReportNumber	Up to 5 symbols for the number of last Z Report
LastZReportDate	16 symbols for date and hour on last Z Report in format DD-MM-YYYY HH:MM
BelongToDepNumber	BelongToDepNumber + 80h, 1 symbol for article department attachment, formed in the following manner: BelongToDepNumber[HEX] + 80h example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h

zfpdef: ReadPLU_Old(PLUNum)

2.5.18. Command: 6Bh / k – Read article registers, Option " (All)

input: *<PLUNum[5]> <;> <Option[""]>*
output: *<PLUNum[5]> <;> <Option[""]> <;> <PLUName[34]> <;> <Price[1..10]> <;> <FlagsPricePLU[1]> <;> <OptionVATClass[1]> <;> <BelongToDepNumber[1..2]> <;> <TurnoverAmount[1..13]> <;> <SoldQuantity[1..13]> <;> <StornoAmount[1..13]> <;> <StornoQuantity[1..13]> <;> <LastZReportNumber[1..5]> <;> <LastZReportDate "DD-MM-YYYY HH:MM"> <;> <AvailableQuantity[1..11]> <;> <Barcode[13]>*

FPR operation: Provides information about all the registers of the specified article.

Input data :

PLUNum	(PLU Number) 5 symbols for article number with leading zeroes in format: #####
Option	One symbol with value '"' *quotation mark*

Output data :

PLUNum	5 symbols for article number with leading zeroes in format: #####
Option	One symbol with value '"'
PLUName	34 symbols for article name, *new line=0x7C.*
Price	Up to 10 symbols for article price
FlagsPricePLU	1 symbol for flags = 0x80 + FlagSinglTr + FlagQTY + OptionPrice
	Where
	OptionPrice:
	0x00 - for free price is disable valid only programmed price
	0x01 - for free price is enable
	0x02 - for limited price
	FlagQTY:
	0x00 - for availability of PLU stock is not monitored
	0x04 - for disable negative quantity
	0x08 - for enable negative quantity
	FlagSingleTr:
	0x00 – no single transaction
	0x10 – single transaction is active
OptionVATClass	*1 character for VAT class:*
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '' - Forbidden*
BelongToDepNumber	*BelongToDepNumber* + 80h, 1 symbol for PLU department attachment = 0x80 … 0x93
TurnoverAmount	Up to 13 symbols for PLU accumulated turnover
SoldQuantity	Up to 13 symbols for Sales quantity of the article
StornoAmount	Up to 13 symbols for accumulated storno amount
StornoQuantity	Up to 13 symbols for accumulated storno quantiy
LastZReportNumber	Up to 5 symbols for the number of the last article report with zeroing
LastZReportDate	16 symbols for the date and time of the last article report with zeroing in format *DD-MM-YYYY HH:MM*
AvailableQuantity	(Available Quantity) Up to 11 symbols for quantity in stock
Barcode	13 symbols for article barcode

zfpdef: *ReadPLUallData(PLUNum)*

2.5.19. Command: 6Bh / k – Read article registers, Option 1 (General)

input: *<PLUNum[5]> <;> <Option['1']>*
output: *<PLUNum[5]> <;> <Option['1']> <;> <PLUName[34]> <;> <Price[1..10]> <;> <OptionPrice[1]> <;> <OptionVATClass[1]> <;> <BelongToDepNumber[1..2]> <;> <TurnoverAmount[1..13]> <;> <SoldQuantity[1..13]> <;> <StornoAmount[1..13]> <;> <StornoQuantity[1..13]> <;> <LastZReportNumber[1..5]> <;> <LastZReportDate "DD-MM-YYYY HH:MM"> <;> <SingleTransaction[1]>*

FPR operation: Provides information about the general registers of the specified article.

Input data :

PLUNum	(PLU No) 5 symbols for article number with leading zeroes in format: #####
Option	One symbol with value '1'

Output data :

PLUNum	5 symbols for article number with leading zeroes in format #####
Option	One symbol with value '1'
PLUName	34 symbols for article name, *new line=0x7C.*
Price	Up to 10 symbols for article price
OptionPrice	1 symbol for price flag with next value:
	- '0'- Free price is disable valid only programmed price
	- '1'- Free price is enable
	- '2'- Limited price
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
BelongToDepNumber	*BelongToDepNumber* + 80h, 1 symbol for PLU department attachment= 0x80 … 0x93
TurnoverAmount	Up to 13 symbols for PLU accumulated turnover
SoldQuantity	Up to 13 symbols for Sales quantity of the article
StornoAmount	Up to 13 symbols for accumulated storno amount
StornoQuantity	Up to 13 symbols for accumulated storno quantiy
LastZReportNumber	Up to 5 symbols for the number of the last article report with zeroing
LastZReportDate	16 symbols for the date and time of the last article report with zeroing in format *DD-MM-YYYY HH:MM*
SingleTransaction	1 symbol with value:
	- '0' – Inactive, default value
	- '1' - Active Single transaction in receipt

zfpdef: ReadPLUgeneral(PLUNum)

2.5.20. Command: 6Bh / k – Read article registers, Option 2 (Quantity)

input: *<PLUNum[5]> <;> <Option['2']>*
output: *<PLUNum[5]> <;> <Option['2']> <;> <AvailableQuantity[1..13]> <;> <OptionQuantityType[1]>*
FPR operation: Provides information about the quantity registers of the specified article.

Input data :
PLUNum   (PLU Number) 5 symbols for article number with leading zeroes in format: #####
Option   One symbol with value 2

Output data :
PLUNum   5 symbols for article number with leading zeroes in format #####
Option   One symbol with value 2
AvailableQuantity   Up to13 symbols for quantity in stock
OptionQuantityType   1 symbol for Quantity flag with next value:
                     - '0'- Availability of PLU stock is not monitored
                     - '1'- Disable negative quantity
                     - '2'- Enable negative quantity

zfpdef: ReadPLUqty(PLUNum)

2.5.21. Command: 6Bh / k – Read article registers, Option 3 (Barcode)

input: *<PLUNum[5]> <;> <Option['3']>*
output: *<PLUNum[5]> <;> <Option['3']> <;> <Barcode[13]>*
FPR operation: Provides information about the barcode of the specified article.

Input data :
PLUNum   (PLU Number) 5 symbols for article number with leading zeroes in format: #####
Option   One symbol with value 3

Output data :
PLUNum   5 symbols for article number with leading zeroes in format #####
Option   One symbol with value 3
Barcode   13 symbols for article barcode

zfpdef: ReadPLUbarcode(PLUNum)

2.5.22. Command: 6Bh / k – Read article registers, Option 4 (Price)

input: *<PLUNum[5]> <;> <Option['4']>*
output: *<PLUNum[5]> <;> <Option['4']> <;> <Price[1..10]> <;> <OptionPrice[1]>*
FPR operation: Provides information about the price and price type of the specified article.

Input data :
PLUNum   (PLU Number) 5 symbols for article number with leading zeroes in format: #####
Option   One symbol with value 4

Output data :
PLUNum   5 symbols for article number with leading zeroes in format #####
Option   One symbol with value 4
Price   Up to 10 symbols for article price
OptionPrice   1 symbol for price flag with next value:
                 - '0'- Free price is disable valid only programmed price
                 - '1'- Free price is enable
                 - '2'- Limited price

zfpdef: ReadPLUprice(PLUNum)

2.5.23. Command: 6Ch / l – Print Logo

    input: *<Number[1..2]>*
    output: ACK
    FPR operation: Prints the programmed graphical logo with the stated number.
Input data :
 Number   Number of logo to be printed. If missing, prints logo with number 0
Output data : n. a.
zfpdef:*PrintLogo(Number)*

2.5.24. Command: 52h / R – Read customer database

    input: *<Option['R']> <;> <CustomerNum[4]>*
    output: *<CustomerNum[4]> <;> <CustomerCompanyName[26]> <;>*
<CustomerFullName[16]> <;> <VATNumber[13]> <;> <UIC[13]> <;> <Address[30]> <;>
<UICType[1]>
    FPR Operation: Provide information for specified customer from FD data base.
Input data:

Option	1 symbol 'R'
CustomerNum	(Customer Number) 4 symbols for customer number in format ####

Output data:

CustomerNum	(Customer Number) 4 symbols for customer number in format ####
CustomerCompanyName	(Company name) 26 symbols for customer name
CustomerFullName	(Buyer Name) 16 symbols for Buyer name
VATNumber	13 symbols for VAT number on customer
UIC	13 symbols for customer Unique Identification Code
Address	30 symbols for address on customer
UICType	*1 symbol for type of* Unique Identification Code*:*
	- '0' - Bulstat
	- '1' - EGN
	- '2' – Foreigner Number
	- '3' – NRA Official Number

zfpdef: ReadCustomerData(CustomerNum)

2.6. RECEIPT OPERATIONS COMMANDS

These commands are used mainly for FD sales registration. The group also includes some auxiliary commands providing information for the current receipt as well as commands for RA and PO amounts.

2.6.1. Command: 2Eh / . – Open Non-fiscal receipt

input: <Oper*Num[1..2]*> <;> <*OperPass[6]*> {<;> <*Reserved['0']*> <;> <*NonFiscalPrintType[1]*>}
output: ACK
FPR operation: Opens a non-fiscal receipt assigned to the specified operator number, operator password and print type.
Input data :

Oper*Num*	(Operator Number) Symbols from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
Reserved	1 symbol with value '0'
NonFiscalPrintType	(Printing type) 1 symbol with value:
	- '0' – Step by step printing
	- '1' – Postponed Printing

Output data: n. a.
zfpdef: OpenNonFiscalReceipt(OperNum, OperPass, NonFiscalPrintType)

2.6.2. Command: 2Fh / / – Close Non-fiscal receipt

input: n. a.
output: ACK
FPR operation: Closes the non-fiscal receipt.
Input data : n. a.
Output data : n. a.
zfpdef: CloseNonFiscalReceipt()

2.6.3. Command: 30h / 0 –Open Fiscal sales receipt

input:<Oper*Num[1..2]> <;> <OperPass[6]> <;> <ReceiptFormat[1]> <;> <PrintVAT[1]> <;> <FiscalRcpPrintType[1]> {<'$'> <UniqueReceiptNumber[24]>}*

output: ACK

FPR operation: Opens a fiscal receipt assigned to the specified operator number and operator password, parameters for receipt format, print VAT, printing type and unique receipt number.

Input data :

OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
ReceiptFormat	(Format) 1 symbol with value:
	- '1' - Detailed
	- '0' - Brief
PrintVAT	(VAT included in the receipt) 1 symbol with value:
	- '1' – Yes
	- '0' – No
FiscalRcpPrintType	(Printing type) 1 symbol with value:
	- '0' – Step by step printing
	- '2' – Postponed printing
	- '4' – Buffered printing
UniqueReceiptNumber	*Up to 24 symbols for unique receipt number.*
	NRA format: XXXXXXX-ZZZZ-YYYYYYY where:
	* XXXXXXX – 8 symbols [A-Z, a-z, 0-9] for individual device number,
	* ZZZZ – 4 symbols [A-Z, a-z, 0-9] for code of the operator,
	* YYYYYYY – 7 symbols [0-9] for next number of the receipt

Output data: n. a.

zfpdef: OpenReceipt(OperNum, OperPass, RcpFormat, PrintVAT, FiscalRcpPrintType, UniqueReceiptNumber)

2.6.4. Command: 30h / 0 –Open Fiscal storno receipt

      input:<*OperNum[1..2]*> <*;*> <*OperPass[6]*> <*;*> <*ReceiptFormat[1]*> <*;*> <*PrintVAT[1]*> <*;*> <*StornoRcpPrintType[1]*> <*;*> <*StornoReason[1]*> <*;*> <*RelatedToRcpNum[1..6]*> <*;*> <*RelatedToRcpDateTime "DD-MM-YY HH:MM:SS"*> <*;*> <*FMNum[8]*> {<*;*> <*RelatedToURN[24]*>}

      output: ACK

      FPR operation: Open a fiscal storno receipt assigned to the specified operator number and operator password, parameters for receipt format, print VAT, printing type and parameters for the related storno receipt.

Input data :

OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
ReceiptFormat	(Format) 1 symbol with value:
	- '1' - Detailed
	- '0' - Brief
PrintVAT	(VAT included in the receipt) 1 symbol with value:
	- '1' – Yes
	- '0' – No
StornoRcpPrintType	(Printing type) 1 symbol with value:
	- '@' – Step by step printing
	- 'B' – Postponed Printing
	- 'D' – Buffered Printing
StornoReason	*1 symbol for reason of storno operation with value:*
	- '0' – Operator error
	- '1' – Goods Claim or Goods return
	- '2' – Tax relief
RelatedToRcpNum	(Receipt number) Up to 6 symbols for issued receipt number
RelatedToRcpDateTime	(Receipt Date and Time) 17 symbols for Date and Time of the issued receipt in format DD-MM-YY HH:MM:SS
FMNum	8 symbols for number of the Fiscal Memory
RelatedToURN	*Up to 24 symbols for the issed receipt unique receipt number.*
	NRA format: XXXXXXX-ZZZZ-YYYYYYY where:
	* XXXXXXX – 8 symbols [A-Z, a-z, 0-9] for individual device number,
	* ZZZZ – 4 symbols [A-Z, a-z, 0-9] for code of the operator,
	* YYYYYYY – 7 symbols [0-9] for next number of the receipt

Output data: n. a.

      Note:

      If Postponed printing is enabled after opened fiscal receipt, all the next commands will be executed but won't be printed immediately. The data for whole receipt is stored to be printed up to AS sent information for receipt closure. If up to 5 sec timeout no command ECR closing the receipt

zfpdef: OpenStornoReceipt(OperNum, OperPass, ReceiptFormat, PrintVAT, StornoRcpPrintType, StornoReason, RelatedToRcpNum, RelatedToRcpDateTime, FMNum, RelatedToURN)

2.6.5. Command: 30h / 0 – Open Fiscal Invoice receipt with free customer data

input: <OperNum[1..2]> <;> <OperPass[6]> <;> <reserved['0']> <;> <reserved['0']> <;> <InvoicePrintType[1]> <;> <Recipient[26]> <;> <Buyer[16]> <;> <VATNumber[13]> <;> <UIC[13]> <;> <Address[30]> <;> <UICType[1]> { <'$'> <UniqueReceiptNumber[24]>}

output: ACK

FPR operation: Opens a fiscal invoice receipt assigned to the specified operator number and operator password with free info for customer data. The Invoice receipt can be issued only if the invoice range (start and end numbers) is set.

Input data :

OperNum	(Operator Number) Symbol from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
reserved	1 symbol with value '0'
reserved	1 symbol with value '0'
InvoicePrintType	(Printing type) 1 symbol with value: - '1' – Step by step printing - '3' – Postponed Printing - '5' – Buffered Printing
Recipient	26 symbols for Invoice recipient
Buyer	16 symbols for Invoice buyer
VATNumber	13 symbols for customer Fiscal number
UIC	13 symbols for customer Unique Identification Code
Address	30 symbols for Address
UICType	1 symbol for type of Unique Identification Code: - '0' - Bulstat - '1' - EGN - '2' – Foreigner Number - '3' – NRA Official Number
UniqueReceiptNumber	Up to 24 symbols for unique receipt number. NRA format: XXXXXXXX-ZZZZ-YYYYYYY where: * XXXXXXXX – 8 symbols [A-Z, a-z, 0-9] for individual device number, * ZZZZ – 4 symbols [A-Z, a-z, 0-9] for code of the operator, * YYYYYYY – 7 symbols [0-9] for next number of the receipt

Output data: n. a.

Note:

If Postponed printing is enabled after opened fiscal receipt, all the next commands will be executed but won't be printed immediately. The data for whole receipt is stored to be printed up to AS sent information for receipt closure. If up to 5 sec timeout no command ECR closing the receipt

zfpdef: OpenInvoiceWithFreeCustomerData(OperNum, OperPass, *InvoicePrintType, Recipient, Buyer, UIC, IDNumber, Address, UICType, UniqueReceiptNumber*)

2.6.6. Command: 30h / 0 – Open Fiscal Invoice Credit Note receipt with free customer data

input: *<OperNum[1..2]> <;> <OperPass[6]> <;> <reserved['0']> <;> <reserved['0']> <;>* <InvoiceCreditNotePrintType[1]> *<;> <Recipient[26]> <;> <Buyer[16]> <;> <VATNumber[13]> <;> <UIC[13]> <;> <Address[30]>* <;> <UICType[1]> <;> <StornoReason[1]> <;> <RelatedToInvoiceNum[10]> <;> <RelatedToInvoiceDateTime"DD-MM-YY HH:MM:SS"> <;> <RelatedToRcpNum[1..6]> <;> <FMNum[8]> { <;> <RelatedToURN[24]> }

output: ACK

FPR operation: Opens a fiscal invoice credit note receipt assigned to the specified operator number and operator password with free info for customer data. The Invoice receipt can be issued only if the invoice range (start and end numbers) is set.

Input data :

Oper*Num*	(Operator Number) Symbol from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
reserved	1 symbol with value '0'
reserved	1 symbol with value '0'
InvoiceCreditNotePrintType	(Printing type) 1 symbol with value: - 'A' – Step by step printing - 'C' – Postponed Printing - 'E' – Buffered Printing
Recipient	26 symbols for Invoice recipient
Buyer	16 symbols for Invoice buyer
VATNumber	13 symbols for customer Fiscal number
UIC	13 symbols for customer Unique Identification Code
Address	30 symbols for Address
UICType	*1 symbol for type of Unique Identification Code:* *- '0' - Bulstat* *- '1' - EGN* *- '2' – Foreigner Number* *- '3' – NRA Official Number*
StornoReason	*1 symbol for reason of storno operation with value:* *- '0' – Operator error* *- '1' – Goods Claim or Goods return* *- '2' – Tax relief*
RelatedToInvoiceNum	10 symbols for issued invoice number
RelatedToInvoiceDateTime	17 symbols for issued invoice date and time in format *DD-MM-YY HH:MM:SS*
RelatedToRcpNum	(Receipt number) Up to 6 symbols for issued receipt number
FMNum	8 symbols for number of the Fiscal Memory
RelatedToURN	*Up to 24 symbols for the issed invoice receipt unique receipt number. NRA format: XXXXXXX-ZZZZ-YYYYYYY where:* ** XXXXXXX – 8 symbols [A-Z, a-z, 0-9] for individual device number,* ** ZZZZ – 4 symbols [A-Z, a-z, 0-9] for code of the operator,* ** YYYYYYY – 7 symbols [0-9] for next number of the receipt*

Output data: n. a.

Note:

If Postponed printing is enabled after opened fiscal receipt, all the next commands will be executed but won't be printed immediately. The data for whole receipt is stored to be printed up to AS sent information for receipt closure. If up to 5 sec timeout no command ECR closing the receipt

zfpdef: OpenCreditNoteWithFreeCustomerData(OperNum, OperPass, *InvoiceCreditNotePrintType, Recipient , Buyer, UIC, IDNumber, Address, UICType, RelatedToInvoiceNum, RelatedToInvoiceDateTime, RelatedToRcpNum, FMNum, RelatedToURN*)

2.6.7. Command: 30h / 0 – Open Fiscal Invoice receipt with customer data from FD database

input: *<OperNum[1..2]> <;> <OperPass[6]> <;> <reserved['0']> <;> <reserved['0']> <;> <InvoicePrintType[1]> <;> <CustomerNum[5]> { <'$'> <UniqueReceiptNumber[24]> }*

output: ACK

FPR operation: Opens a fiscal invoice receipt assigned to the specified operator number and operator password with internal DB info for customer data. The Invoice receipt can be issued only if the invoice range (start and end numbers) is set.

Input data :

Oper*Num*	(Operator Number) Symbol from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
reserved	1 symbol with value '0'
reserved	1 symbol with value '0'
InvoicePrintType	(Printing type) 1 symbol with value:
	- '1' – Step by step printing
	- '3' – Postponed Printing
	- '5' – Buffered Printing
CustomerNum	Symbol '#' and following up to 4 symbols for related customer ID number corresponding to the FD database
UniqueReceiptNumber	*Up to 24 symbols for unique receipt number.*
	NRA format: XXXXXXXX-ZZZZ-YYYYYYY where:
	* XXXXXXXX – 8 symbols [A-Z, a-z, 0-9] for individual device number,
	* ZZZZ – 4 symbols [A-Z, a-z, 0-9] for code of the operator,
	* YYYYYYY – 7 symbols [0-9] for next number of the receipt

Output data: n. a.

Note:

If Postponed printing is enabled after opened fiscal receipt, all the next commands will be executed but won't be printed immediately. The data for whole receipt is stored to be printed up to AS sent information for receipt closure. If up to 5 sec timeout no command ECR closing the receipt

zfpdef: OpenInvoiceWithFDCustomerDB(OperNum, OperPass, *InvoicePrintType, CustomerNum, UniqueReceiptNumber)*

2.6.8. Command: 30h / 0 – Open Fiscal Invoice Credit Note receipt with customer data from FD database

input: *<OperNum[1..2]> <;> <OperPass[6]> <;> <reserved['0']> <;> <reserved['0']> <;> <InvoiceCreditNotePrintType[1]> <;> <CustomerNum[5]> <;> <StornoReason[1]> <;> <RelatedToInvoiceNum[10]> <;> <RelatedToInvoiceDateTime "DD-MM-YY HH:MM:SS"> <;> <RelatedToRcpNum[1..6]> <;> <FMNum[8]> { <;> <RelatedToURN[24]> }*

output: ACK

FPR operation: Opens a fiscal invoice credit note receipt assigned to the specified operator number and operator password with internal DB info for customer data. The Invoice receipt can be issued only if the invoice range (start and end numbers) is set.

Input data :

Oper*Num*	(Operator Number) Symbol from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 6 symbols for operator's password
reserved	1 symbol with value '0'
reserved	1 symbol with value '0'
InvoiceCreditNotePrintType	(Printing type) 1 symbol with value: - 'A' – Step by step printing - 'C' – Postponed Printing - 'E' – Buffered Printing
CustomerNum	*Symbol '#' and following up to 4 symbols for related customer ID number corresponding to the FD database*
StornoReason	*1 symbol for reason of storno operation with value:* *- '0' – Operator error* *- '1' – Goods Claim or Goods return* *- '2' – Tax relief*
RelatedToInvoiceNum	10 symbols for issued invoice number
RelatedToInvoiceDateTime	17 symbols for issued invoice date and time in format *DD-MM-YY HH:MM:SS*
RelatedToRcpNum	(Receipt number) Up to 6 symbols for issued receipt number
FMNum	*8 symbols for number of the Fiscal Memory*
RelatedToURN	*Up to 24 symbols for the issed invoice receipt unique receipt number. NRA format: XXXXXXXX-ZZZZ-YYYYYYY where:* ** XXXXXXXX – 8 symbols [A-Z, a-z, 0-9] for individual device number,* ** ZZZZ – 4 symbols [A-Z, a-z, 0-9] for code of the operator,* ** YYYYYYY – 7 symbols [0-9] for next number of the receipt*

Output data: n. a.

Note:

If Postponed printing is enabled after opened fiscal receipt, all the next commands will be executed but won't be printed immediately. The data for whole receipt is stored to be printed up to AS sent information for receipt closure. If up to 5 sec timeout no command ECR closing the receipt

zfpdef: OpenCreditNoteWithFDCustomerDB(OperNum, OperPass, *InvoiceCreditNotePrintType, CustomerNum, RelatedToInvoiceNum, RelatedToInvoiceDateTime, RelatedToRcpNum, FMNum, RelatedToURN)*

2.6.9. Command: 31h / 1 – Sell/Correction of article belonging to VAT class definition

input: *<NamePLU[36]> <;> <OptionVATClass[1]> <;> <Price[1..10]> {<'*'> <Quantity[1..10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>}*

output: ACK

FPR operation: Register the sell (for correction use minus sign in the price field) of article with specified name, price, quantity, VAT class and/or discount/addition on the transaction.

Input data :

NamePLU	(PLU Name) 36 symbols for article's name. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
''*	1 symbol '*' indicating the presence of quantity field
Quantity	Up to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 symbols for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for value of discount/addition. Use minus sign '-' for discount

Output data : n. a.

Notes:

The quantity fields are not obligatory. If no value is stated for them the FD executed the command for a default quantity of 1.000

The discount/addition fields are not obligatory. The discount/addition must be in percents and is determined from the presence or absence of the '-' symbol.

zfpdef: SellPLUwithSpecifiedVAT(NamePLU, OptionVATClass, Price, Quantity, DiscAddP, DiscAddV)

2.6.10. Command: 31h / 1 – Sell/Correction of article belonging to department

input: *<NamePLU[36]> <;> <reserved[' ']> <;> <Price[1..10]> {<'*'> <Quantity[1..10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>} {<'!'> <DepNum[1..2]>}*

output: ACK

FPR operation: Register the sell (for correction use minus sign in the price field) of article belonging to department with specified name, price, quantity and/or discount/addition on the transaction. The VAT of article got from department to which article belongs.

Input data :

NamePLU	(PLU Name) 36 symbols for article's name. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
reserved	1 symbol with value "space"
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	Up to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 symbols for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for value of discount/addition. Use minus sign '-' for discount
'!'	1 symbol '!' indicating the presence of departament
DepNum	DepNum + 80h (Department Number) 1 symbol for article department attachment, formed in the following manner; *example: Dep01=81h, Dep02=82h … Dep19=93h*

Output data : n. a.

Notes:

The quantity fields are not obligatory. If no value is stated for them the FD executed the command for a default quantity of 1.000

The discount/addition fields are not obligatory. The discount/addition must be in percents and is determined from the presence or absence of the '-' symbol.

zfpdef: SellPLUfromDep(NamePLU, Price, Quantity, DiscAddP, DiscAddV, DepNum)

2.6.11. Command: 31h / 1 – Sell/Correction of article with specified VAT belonging to department

input: *<NamePLU[36]> <;> <OptionVATClass[1]> <;> <Price[1..10]> {<'*'> <Quantity[1..10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>} {<'!'> <DepNum[1..2]>}*

output: ACK

FPR operation: Register the sell (for correction use minus sign in the price field) of article with specified VAT. If department is present the relevant accumulations are perfomed in its registers.

Input data :

NamePLU	(PLU Name) 36 symbols for article's name. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	Up to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 symbols for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for value of discount/addition. Use minus sign '-' for discount
'!'	1 symbol '!' indicating the presence of departament
DepNum	*DepNum* + 80h (Department Number) 1 symbol for article department attachment, formed in the following manner; *example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h*

Output data : n. a.

Notes:

The quantity fields are not obligatory. If no value is stated for them the FD executed the command for a default quantity of 1.000

The discount/addition fields are not obligatory. The discount/addition must be in percents and is determined from the presence or absence of the '-' symbol.

zfpdef: SellPLUwithSpecifiedVATfromDep(NamePLU, OptionVATClass, Price, Quantity, DiscAddP, DiscAddV, DepNum)

2.6.12. Command: 32h / 2 – Sell/Correction of article from FD database

input: *<OptionSign[1]> <PLUNum[5]> {<'$'> <Price[1..8]>} {<'*'> <Quantity[1..10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>}*

output: ACK

FPR operation: Register the sell or correction with specified quantity of article from the internal FD database. The FD will perform a correction operation only if the same quantity of the article has already been sold.

Input data :

OptionSign	(Sale/Correction) 1 symbol with optional value:
	- '+' –Sale
	- '-' - Correction
PLUNum	(PLU Number) 5 symbols for PLU number of FD's database in format #####
'$'	1 symbol '$' indicating the presence of price
Price	Up to 10 symbols for sale price
''*	1 symbol '*' indicating the presence of quantity field
Quantity	Up to 10 symbols for article's quantity sold
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbolsfor percentage of discount/addition. Use minus sign '-' for discount

Output data : n. a.

zfpdef: Sell*PLUFromFD_DB(OptionSign, PLUNum, Price, Quantity, DiscAddP, DiscAddV)*

2.6.13. Command: 33h / 3 – Subtotal

input: *<OptionPrinting[1]> <;> <OptionDisplay[1]> {<':'> <DiscAddV[1..8]>} {<','> <DiscAddP[1..7]>}*

output:*<SubtotalValue[1..10]>*

FPR operation: Calculate the subtotal amount with printing and display visualization options. Provide information about values of the calculated amounts. If a percent or value discount/addition has been specified the subtotal and the discount/addition value will be printed regardless the parameter for printing.

Input data :

OptionPrinting	1 symbol with value:
	- '1' – Yes
	- '0' – No
OptionDisplay	1 symbol with value:
	- '1' – Yes
	- '0' – No
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for the value of the discount/addition. Use minus sign '-' for discount
','	1 symbol ',' indicating the presence of percent discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 symbols for the percentage value of the discount/addition. Use minus sign '-' for discount

Output data:

SubtotalValue	Up to 10 symbols for the value of the subtotal amount

Notes:

When the discount/addition is a percentage the amount is distributed proportionally over the turnover items and is automatically transferred to the turnovers of the corresponding VAT groups.

A value discount/addition may be specified only if all sales are of articles (items) belonging to one and the same VAT group.

zfpdef: *Subtotal(OptionPrinting, OptionDisplay, DiscAddP, DiscAddV)*

2.6.14. Command: 34h / 4 – Sell/Correction of article with department definition belonging to VAT class

input: *<NamePLU[36]> <;> <DepNum[1..2]> <;> <Price[1..10]> {<'*'> <Quantity[1..10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>} {<'!'> <OptionVATClass[1]>}*

output: ACK

FPR operation: Register the sell (for correction use minus sign in the price field) of article with specified department. If VAT is present the relevant accumulations are perfomed in its registers.

Input data :

NamePLU	(PLU Name) 36 symbols for name of sale. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
DepNum	*DepNum* + 80h, (Department Number) 1 symbol for article department attachment, formed in the following manner: DepNum[HEX] + 80h example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
''*	1 symbol '*' indicating the presence of quantity field
Quantity	Up to 10 symbols for article's quantity sold
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for percentage of discount/addition. Use minus sign '-' for discount
'!'	1 symbol with value '!'
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden

Output data: n. a.

Notes:

Sale/correction will be collected in VAT group on department, value and qty will be gain (removed) in this department.

For correction use '-' before value price.

Field for quantity is optional. If there no set qty default value is 1.000

Fields for Disc/add are optional. Disc/add can be only in percents, for discount use '-' before value. Cannot make correction with Disc/add.

zfpdef: SellPLUwithSpecifiedVATfromDep_*(NameSale, DepNum, Price, Quantity, DiscAddP, DiscAddV, OptionVATClass)*

2.6.15. Command: 34h / 4 – Sell/Correction of article with specified department

input: *<NamePLU[36]> <;> <DepNum[1..2]> <;> <Price[1..10]> {<'*'> <Quantity[1..10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>}*

output: ACK

FPR operation: Registers the sell (for correction use minus sign in the price field) of article with specified department, name, price, quantity and/or discount/addition on the transaction.

Input data :

NamePLU	(PLU Name) 36 symbols for name of sale. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
DepNum	*DepNum* + 80h, (Department Number) 1 symbol for article department attachment, formed in the following manner: DepNum[HEX] + 80h example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
''*	1 symbol '*' indicating the presence of quantity field
Quantity	Up to 10symbols for article's quantity sold
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for percentage of discount/addition. Use minus sign '-' for discount
'!'	1 symbol with value '!'

Output data: n. a.

Notes:

Sale/correction will be collected in VAT group on department, value and qty will be gain (removed) in this department.

For correction use '-' before value price.

Fields for qty are optional. If there no set qty default value is 1.000

Fields for Disc/add are optional. Disc/add can be only in percents, for discount use '-' before value. Cannot make correction with Disc/add.

zfpdef: SellPLUfromDep_*(NameSale, DepNum, Price, Quantity, DiscAddP, DiscAddV)*

2.6.16. Command: 35h / 5 – Payment

input: *<PaymentType[1..2]> <;> <OptionChange[1]> <;> <Amount[1..10]> { <;> <OptionChangeType[1]> }*

output: ACK

FPR operation: Register the payment in the receipt with specified type of payment with amount received.

Input data :

PaymentType	(Payment Types)1 symbol for payment type:
	- '0' – Payment 0
	- '1' – Payment 1
	- '2' – Payment 2
	- '3' – Payment 3
	- '4' – Payment 4
	- '5' – Payment 5
	- '6' – Payment 6
	- '7' – Payment 7
	- '8' – Payment 8
	- '9' – Payment 9
	- '10' – Payment 10
	- '11' – Payment 11
OptionChange	(Change) Default value is 0, 1 symbol with value:
	- '0 – With Change
	- '1' – Without Change
Amount	Up to 10 characters for received amount
'*'	1 symbol with value '*'
OptionChangeType	1 symbols with value:
	- '0' – Change In Cash
	- '1' – Same As The payment
	- '2' – Change In Currency

Output data: n. a.

Notes:

By executing this command the FD enters the payment mode. No further sales and/or corrections are allowed.

If the amount received is equal to or greater than the grand total amount (the amount due) the FD quits the procedure and calculates the change in the specified type of payment except in the cases when OptionChange is not 1 – in such cases the operator is liable for the stated amount.

The receipt can be finalized only when the last payment transfer is sufficient to cover the whole amount due (the grand total amount), i.e. the payment procedure has been finalized.

zfpdef: Payment(PaymentType, OptionChange, Amount, OptionChangeType)

2.6.17. Command: 35h / 5 – Pay exact sum

input: *<PaymentType[1..2]> <;> <Option['0']> <;> <Amount[""]>*
output: ACK
FPR operation: Register the payment in the receipt with specified type of payment and exact amount received.
Input data :

PaymentType	(Payment Types)1 symbol for payment type:
	- '0' – Payment 0
	- '1' – Payment 1
	- '2' – Payment 2
	- '3' – Payment 3
	- '4' – Payment 4
	- '5' – Payment 5
	- '6' – Payment 6
	- '7' – Payment 7
	- '8' – Payment 8
	- '9' – Payment 9
	- '10' – Payment 10
	- '11' – Payment 11

Option   1 symbol with value 0
Amount   1 symbol ' ” ', quotation mark, for pay with exact sum
Output data: **n.a.**
zfpdef: PayExactSum(PaymentType)

2.6.18. Command: 36h / 6 – Automatic receipt closure

input: n. a.
output: ACK
FPR operation: Paying the exact amount in cash and close the fiscal receipt.
Input data : n. a.
Output data : n. a.
zfpdef: CashPayCloseReceipt()

2.6.19. Command: 37h / 7 – Free text printing

input: *<Text[TextLength-2]>*
output: ACK
FPR operation: Print a free text. The command can be executed only if receipt is opened (Fiscal receipt, Invoice receipt, Storno receipt, Credit Note or Non-fical receipt). In the beginning and in the end of line symbol '#' is printed.
Input data :

Text   *TextLength-2* symbols
Output data: **n. a.**
zfpdef: PrintText(Text)

2.6.20. Command: 38h / 8 – Close Fiscal receipt

input: n. a.
output: ACK
FPR operation: Close the fiscal receipt (Fiscal receipt, Invoice receipt, Storno receipt, Credit Note or Non-fical receipt). When the payment is finished.
Input data : n. a.
Output data : n. a.
zfpdef: CloseReceipt()

2.6.21. Command: 39h / 9 – Cancel fiscal receipt

input: n. a.
output: ACK
FPR operation: Available only if receipt is not closed. Void all sales in the receipt and close the fiscal receipt (Fiscal receipt, Invoice receipt, Storno receipt or Credit Note). If payment is started, then finish payment and close the receipt.
Input data : n. a.
Output data : n. a.
zfpdef: CancelReceipt()

2.6.22. Command: 3Ah / : – Print a copy of the last document

input: *n. a.*
output: ACK
FPR operation: Print a copy of the last receipt issued. When FD parameter for duplicates is enabled.
Input data : n. a.
Output data : n. a.
zfpdef: PrintLastReceiptDuplicate()

2.6.23. Command: 3Bh / ; – Non-fiscal RA and PO amounts

input:*<OperNum[1..2]> <;> <OperPass[6]> <;> <reserved['0']> <;> <Amount[1..10]> {<'$'> <PrintAvailability[1]> } {<;> <Text[TextLength-2]>}*
output: ACK
FPR operation: Registers cash received on account or paid out.
Input data :

OperNum	(Operator Number) Symbols from 1 to 20 corresponding to the operator's number
OperPass	(Operator Password) 6 symbols for operator's password
reserved	1 symbol reserved with value '0'
Amount	Up to 10 symbols for the amount lodged. Use minus sign for withdrawn
'$'	1 symbol with value '$'
PrintAvailability	1 symbol with value: - '0' – No - '1' - Yes
Text	TextLength-2 symbols. In the beginning and in the end of line symbol '#' is printed.

Output data: n. a.
zfpdef: ReceivedOnAccount_PaidOut(OperNum, OperPass, Amount, PrintAvailability, Text)

2.6.24. Command: 3Eh / > – Discount/ Addition

input: *<Type[1]> <;> <OptionSubtotal[1]> {<':'> <DiscAddV[1..8]>} {<','> <DiscAddP[1..7]>}*
output: **ACK**
FPR operation: Percent or value discount/addition over sum of transaction or over subtotal sum specified by field "*Type*".

Input data :

Type	1 symbol with value
	- '2' - Defined from the device
	- '1' - Over subtotal
	- '0' - Over transaction sum
OptionSubtotal	(Display Subtotal) 1 symbol with value
	- '1' - Yes
	- '0' - No
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for the value of the discount/addition. Use minus sign '-' for discount
','	1 symbol ',' indicating the presence of percent discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 symbols for the percentage value of the discount/addition. Use minus sign '-' for discount

Output data: n.a.
zfpdef: PrintDiscountOrAddition(*Type*, *OptionSubtotal*, *DiscAddV*, *DiscAddP*)

2.6.25. Command: 3Dh / = – Sell/Correction of article with fractional quantity belonging to VAT class definition

input: *<NamePLU[36]> <;> <OptionVATClass[1]> <;> <Price[1..10]> {<'*'> <Quantity[10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>}*
output: **ACK**
FPR operation: Register the sell (for correction use minus sign in the price field) of article with specified name, price, fractional quantity, VAT class and/or discount/addition on the transaction.

Input data :

NamePLU	(PLU Name) 36 symbols for article's name. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	From 3 to 10 symbols for quantity in format fractional format, e.g. 1/3
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) 1 to 7 symbols for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 symbols for value of discount/addition. Use minus sign '-' for discount

Output data : n. a.

Notes:

If the price field is preceded by a '-' the command is executed by the FD as a correction/void (only if the amount of the corresponding VAT group of the receipt is sufficient).

The quantity fields are not obligatory. If no value is stated for them the FD executed the command for a default quantity of 1.000

The discount/addition fields are not obligatory. The discount/addition must be in percents and is determined from the presence or absence of the '-' symbol.

zfpdef: SellFractQtyPLUwithSpecifiedVAT(NamePLU, OptionVATClass, Price, Quantity, DiscAddP, DiscAddV)

2.6.26. Command: 3Dh / = – Sell/Correction of article with fractional quantity belonging to department

input: *<NamePLU[36]> <;> <reserved[' ']> <;> <Price[1..10]> {<'*'> <Quantity[10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>} {<'!'> <DepNum[1..2]>}*

output: ACK

FPR operation: Register the sell (for correction use minus sign in the price field) of article belonging to department with specified name, price, fractional quantity and/or discount/addition on the transaction. The VAT of article got from department to which article belongs.

Input data :

NamePLU	(PLU Name) 36 symbols for article's name. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
reserved	1 symbol with value "space"
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	From 3 to 10 symbols for quantity in format fractional format, e.g. 1/3
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) 1 to 7 symbols for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 symbols for value of discount/addition. Use minus sign '-' for discount
'!'	1 symbol '!' indicating the presence of departament
DepNum	DepNum + 80h (Department Number) 1 symbol for article department attachment, formed in the following manner; *example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h*

Output data : n. a.

Notes:

If the price field is preceded by a '-' the command is executed by the FD as a correction/void (only if the amount of the corresponding VAT group of the receipt is sufficient).

The quantity fields are not obligatory. If no value is stated for them the FD executed the command for a default quantity of 1.000

The discount/addition fields are not obligatory. The discount/addition must be in percents and is determined from the presence or absence of the '-' symbol.

zfpdef: SellFractQtyPLUfromDep(NamePLU, Price, Quantity, DiscAddP, DiscAddV, DepNum)

2.6.27. Command: 3Dh / = – Sell/Correction of article with fractional quantity with specified VAT belonging to department

input: *<NamePLU[36]> <;> <OptionVATClass[1]> <;> <Price[1..10]> {<'*'> <Quantity[10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>} {<'!'> <DepNum[1..2]>}*

output: ACK

FPR operation: Register the sell (for correction use minus sign in the price field) of article with specified VAT. If department is present the relevant accumulations are perfomed in its registers.

Input data :

NamePLU	(PLU Name) 36 symbols for article's name. 34 symbols are printed on paper. Symbo 0x7C '|' is new line separator.
OptionVATClass	1 character for VAT class:
	- 'А' – VAT Class 0
	- 'Б' – VAT Class 1
	- 'В' – VAT Class 2
	- 'Г' – VAT Class 3
	- 'Д' – VAT Class 4
	- 'Е' – VAT Class 5
	- 'Ж' – VAT Class 6
	- 'З' – VAT Class 7
	- '*' - Forbidden
Price	Up to 10 symbols for article's price. Use minus sign '-' for correction
''*	1 symbol '*' indicating the presence of quantity field
Quantity	From 3 to 10 symbols for quantity in format fractional format, e.g. 1/3
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) Up to 7 symbols for percentage of discount/addition. Use minus sign '-' for discount
':'	1 symbol ',' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) Up to 8 symbols for value of discount/addition. Use minus sign '-' for discount
'!'	1 symbol '!' indicating the presence of departament
DepNum	DepNum + 80h (Department Number) 1 symbol for article department attachment, formed in the following manner; *example: Dep01 = 81h, Dep02 = 82h … Dep19 = 93h*

Output data : n. a.

Notes:

If the price field is preceded by a '-' the command is executed by the FD as a correction/void (only if the amount of the corresponding VAT group of the receipt is sufficient).

The quantity fields are not obligatory. If no value is stated for them the FD executed the command for a default quantity of 1.000

The discount/addition fields are not obligatory. The discount/addition must be in percents and is determined from the presence or absence of the '-' symbol.

zfpdef: SellFractQtyPLUwithSpecifiedVATfromDep(NamePLU, OptionVATClass, Price, Quantity, DiscAddP, DiscAddV, DepNum)

2.7. COMMANDS FOR READING THE DATA IN FD'S REGISTERS

This set of commands provides information about the status of FD's registers without causing a device activity, i.e. the information is obtained through the communication interface without printing or display visualization.

2.7.1. Command: 6Dh / m – Read daily sale and storno amounts by VAT groups

input: n. a.

output: *<SaleAmountVATGr0[1..13]> <;> <SaleAmountVATGr1[1..13]> <;> <SaleAmountVATGr2[1..13]> <;> <SaleAmountVATGr3[1..13]> <;> <SaleAmountVATGr4[1..13]> <;> <SaleAmountVATGr5[1..13]> <;> <SaleAmountVATGr6[1..13]> <;> <SaleAmountVATGr7[1..13]> <;><SumAllVATGr[1..13]><;>*

<StornoAmountVATGr0[1..13]> <;><StornoAmountVATGr1[1..13]> <;> <StornoAmountVATGr2[1..13]> <;><StornoAmountVATGr3[1..13]> <;> <StornoAmountVATGr4[1..13]> <;>< StornoAmountVATGr5[1..13]> <;> <StornoAmountVATGr6[1..13]> <;> <StornoAmountVATGr7[1..13]> <;> <StornoAllVATGr[1..13]>

FPR operation: Provides information about the accumulated sale and storno amounts by VAT group.

Input data : n. a.

Output data :

SaleAmountVATGr0	Up to 13 symbols for the amount accumulated from sales by VAT group А
SaleAmountVATGr1	Up to 13 symbols for the amount accumulated from sales by VAT group Б
SaleAmountVATGr2	Up to 13 symbols for the amount accumulated from sales by VAT group В
SaleAmountVATGr3	Up to 13 symbols for the amount accumulated from sales by VAT group Г
SaleAmountVATGr4	Up to 13 symbols for the amount accumulated from sales by VAT group Д
SaleAmountVATGr5	Up to 13 symbols for the amount accumulated from sales by VAT group Е
SaleAmountVATGr6	Up to 13 symbols for the amount accumulated from sales by VAT group Ж
SaleAmountVATGr7	Up to 13 symbols for the amount accumulated from sales by VAT group З
SumAllVATGr	Up to 13 symbols for sum of all VAT groups
StornoAmountVATGr0	Up to 13 symbols for the amount accumulated from Storno by VAT group А
StornoAmountVATGr1	Up to 13 symbols for the amount accumulated from Storno by VAT group Б
StornoAmountVATGr2	Up to 13 symbols for the amount accumulated from Storno by VAT group В
StornoAmountVATGr3	Up to 13 symbols for the amount accumulated from Storno by VAT group Г
StornoAmountVATGr4	Up to 13 symbols for the amount accumulated from Storno by VAT group Д
StornoAmountVATGr5	Up to 13 symbols for the amount accumulated from Storno by VAT group Е
StornoAmountVATGr6	Up to 13 symbols for the amount accumulated from Storno by VAT group Ж
StornoAmountVATGr7	Up to 13 symbols for the amount accumulated from Storno by VAT group З
StornoAllVATGr	Up to 13 symbols for the amount accumulated from Storno by all groups

zfpdef: ReadDailySaleAndStornoAmountsByVAT()

2.7.2. Command: 6Eh / n – Read registers, Option '0' (on hand)

  input: *<'0'>*
  output: *<'0'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]> <;> <AmountPayment5[1..13]> <;> <AmountPayment6[1..13]> <;> <AmountPayment7[1..13]> <;> <AmountPayment8[1..13]> <;> <AmountPayment9[1..13]> <;> <AmountPayment10[1..13]> <;> <AmountPayment11[1..13]>*

  FPR operation: Provides information about the amounts on hand by type of payment.

Input data :

'0'	1 symbol obligatory '0'

Output data:

'0'	1 symbol obligatory '0'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 13 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 13 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 13 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 13 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 13 symbols for the accumulated amount by payment type 9
AmountPayment10	Up to 13 symbols for the accumulated amount by payment type 10
AmountPayment11	Up to 13 symbols for the accumulated amount by payment type 11

zfpdef: *ReadDailyAvailableAmounts()*

2.7.3. Command: 6Eh / n – Read registers, Option '0', KL (on hand)

  input: *<'0'>*
  output: *<'0'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]>*

  FPR operation: Provides information about the amounts on hand by type of payment. Command works for KL version 2 devices.

Input data :

'0'	1 symbol obligatory '0'

Output data:

'0'	1 symbol obligatory '0'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4

zfpdef: *ReadDailyAvailableAmounts_Old()*

2.7.4. Command: 6Eh / n – Read registers, Option '1' (general)

input: *<'1'>*
output: *<'1'> <;> <CustomersNum[1..5]> <;> <DiscountsNum[1..5]> <;> <DiscountsAmount[1..13]> <;> <AdditionsNum[1..5]> <;> <AdditionsAmount[1..13]> <;> <CorrectionsNum[1..5]> <;> <CorrectionsAmount[1..13]>*

FPR operation: Provides information about the number of customers (number of fiscal receipt issued), number of discounts, additions and corrections made and the accumulated amounts.

Input data :
'1'	1 symbol with value '1'

Output data:
'1'	1 symbol with value '1'
CustomersNum	Up to 5 symbols for number of customers
DiscountsNum	Up to 5 symbols for number of discounts
DiscountsAmount	Up to 13 symbols for accumulated amount of discounts
AdditionsNum	Up to 5 symbols for number of additions
AdditionsAmount	Up to 13 symbols for accumulated amount of additions
CorrectionsNum	Up to 5 symbols for number of corrections
CorrectionsAmount	Up to 13 symbols for accumulated amount of corrections

zfpdef:ReadGeneralDailyRegisters()

2.7.5. Command: 6Eh / n – Read registers, Option '2' (RA)

input: *<'2'>*
output: *<'2'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]> <;> <AmountPayment5[1..13]> <;> <AmountPayment6[1..13]> <;> <AmountPayment7[1..13]> <;> <AmountPayment8[1..13]> <;> <AmountPayment9[1..13]> <;> <AmountPayment10[1..13]> <;> <AmountPayment11[1..13]> <;> <RANum[1..5]> <;> <SumAllPayment[1..13]>*

FPR operation: Provides information about the RA amounts by type of payment and the total number of operations.

Input data :
'2'	1 symbol obligatory '2'

Output data:
'2'	1 symbol obligatory '2'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 13 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 13 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 13 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 13 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 13 symbols for the accumulated amount by payment type 9
AmountPayment10	Up to 13 symbols for the accumulated amount by payment type 10
AmountPayment11	Up to 13 symbols for the accumulated amount by payment type 11
RANum	Up to 5 symbols for the total number of operations
SumAllPayment	Up to 13 symbols to sum all payments

zfpdef: ReadDailyRA()

2.7.6. Command: 6Eh / n – Read registers, Option '2', KL (RA)

input: <*'2'*>

output: <*'2'*> <;> <*AmountPayment0[1..13]*> <;> <*AmountPayment1[1..13]*> <;> <*AmountPayment2[1..13]*> <;> <*AmountPayment3[1..13]*> <;> <*AmountPayment4[1..13]*> <;> <*RANum[1..5]*> <;> <*SumAllPayment[1..13]*>

FPR operation: Provides information about the RA amounts by type of payment and the total number of operations. Command works for KL version 2 devices.

Input data :

'2'	1 symbol obligatory '2'

Output data:

'2'	1 symbol obligatory '2'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
RANum	Up to 5 symbols for the total number of operations
SumAllPayment	Up to 13 symbols to sum all payments

zfpdef: *ReadDailyRA_Old()*

2.7.7. Command: 6Eh / n – Read registers, Option '3' (PO)

input: <*'3'*>

output: <*'3'*> <;> <*AmountPayment0[1..13]*> <;> <*AmountPayment1[1..13]*> <;> <*AmountPayment2[1..13]*> <;> <*AmountPayment3[1..13]*> <;> <*AmountPayment4[1..13]*> <;> <*AmountPayment5[1..13]*> <;> <*AmountPayment6[1..13]*> <;> <*AmountPayment7[1..13]*> <;> <*AmountPayment8[1..13]*> <;> <*AmountPayment9[1..13]*> <;> <*AmountPayment10[1..13]*> <;> <*AmountPayment11[1..13]*> <;> <*PONum[1..5]*> <;> <*SumAllPayment[1..13]*>

FPR operation: Provides information about the PO amounts by type of payment and the total number of operations.

Input data :

'3'	1 symbol obligatory '3'

Output data:

'3'	1 symbol obligatory '3'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 13 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 13 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 13 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 13 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 13 symbols for the accumulated amount by payment type 9
AmountPayment10	Up to 13 symbols for the accumulated amount by payment type 10
AmountPayment11	Up to 13 symbols for the accumulated amount by payment type 11
PONum	Up to 5 symbols for the total number of operations
SumAllPayment	Up to 13 symbols to sum all payments

zfpdef: *ReadDailyPO()*

2.7.8. Command: 6Eh / n – Read registers, Option '3', KL (PO)

> **input:** <'3'>
>
> **output:** <'3'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]> <;> <PONum[1..5]> <;> <SumAllPayment[1..13]>
>
> **FPR operation:** Provides information about the PO amounts by type of payment and the total number of operations. Command works for KL version 2 devices.

Input data :

'3'	1 symbol obligatory '3'

Output data:

'3'	1 symbol obligatory '3'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
PONum	Up to 5 symbols for the total number of operations
SumAllPayment	Up to 13 symbols to sum all payments

zfpdef: *ReadDailyPO_Old()*

2.7.9. Command: 6Eh / n – Read registers, Option '4' (received)

> **input:** <'4'>
>
> **output:** <'4'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]> <;> <AmountPayment5[1..13]> <;> <AmountPayment6[1..13]> <;> <AmountPayment7[1..13]> <;> <AmountPayment8[1..13]> <;> <AmountPayment9[1..13]> <;> <AmountPayment10[1..13]> <;> <AmountPayment11[1..13]>
>
> **FPR operation:** Provides information about the amounts received from sales by type of payment.

Input data :

'4'	1 symbol obligatory '4'

Output data:

'4'	1 symbol obligatory '4'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 13 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 13 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 13 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 13 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 13 symbols for the accumulated amount by payment type 9
AmountPayment10	Up to 13 symbols for the accumulated amount by payment type 10
AmountPayment11	Up to 13 symbols for the accumulated amount by payment type 11

zfpdef: ReadDailyReceivedSalesAmounts()

2.7.10. Command: 6Eh / n – Read registers, Option '4', KL (received)

> **input:** *<'4'>*
> **output:** *<'4'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]>*
> **FPR operation:** Provides information about the amounts received from sales by type of payment. Command works for KL version 2 devices.

Input data :

'4'	1 symbol obligatory '4'

Output data:

'4'	1 symbol obligatory '4'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4

zfpdef: ReadDailyReceivedSalesAmounts_Old()

2.7.11. Command: 6Eh / n – Read registers, Option '5' (counters)

> **input:** *<'5'>*
> **output:** *<'5'> <;> <LastReportNumFromReset[1..5]> <;> <LastFMBlockNum[1..5]> <;> <EJNum[1..5]> <;> <DateTime "DD-MM-YYYY HH:MM">*
> **FPR operation:** Provides information about the current reading of the daily-report-with-zeroing counter, the number of the last block stored in FM, the number of EJ and the date and time of the last block storage in the FM.

Input data :

'5'	1 symbol obligatory '5'

Output data:

'5'	1 symbol obligatory '5'
LastReportNumFromReset	Up to 5 symbols for number of the last report from reset
LastFMBlockNum	Up to 5 symbols for number of the last FM report
EJNum	Up to 5 symbols for number of EJ
DateTime	16 symbols for date and time of the last block storage in FM in format *"DD-MM-YYYY HH:MM"*

zfpdef: ReadDailyCounters()

2.7.12. Command: 6Eh / n – Read registers, Option '6' (change)

input: *<'6'>*

output: *<'6'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]> <;> <AmountPayment5[1..13]> <;> <AmountPayment6[1..13]> <;> <AmountPayment7[1..13]> <;> <AmountPayment8[1..13]> <;> <AmountPayment9[1..13]> <;> <AmountPayment10[1..13]> <;> <AmountPayment11[1..13]>*

FPR operation: Provides information about the amounts returned as change by type of payment.

Input data :

'6'   1 symbol obligatory '6'

Output data:

'6'	1 symbol obligatory '6'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 13 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 13 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 13 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 13 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 13 symbols for the accumulated amount by payment type 9
AmountPayment10	Up to 13 symbols for the accumulated amount by payment type 10
AmountPayment11	Up to 13 symbols for the accumulated amount by payment type 11

zfpdef: ReadDailyReturnedChangeAmounts()

2.7.13. Command: 6Eh / n – Read registers, Option '6', KL (change)

input: *<'6'>*

output: *<'6'> <;> <AmountPayment0[1..13]> <;> <AmountPayment1[1..13]> <;> <AmountPayment2[1..13]> <;> <AmountPayment3[1..13]> <;> <AmountPayment4[1..13]>*

FPR operation: Provides information about the amounts returned as change by type of payment. Command works for KL version 2 devices.

Input data :

'6'   1 symbol obligatory '6'

Output data:

'6'	1 symbol obligatory '6'
AmountPayment0	Up to 13 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 13 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 13 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 13 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 13 symbols for the accumulated amount by payment type 4

zfpdef: ReadDailyReturnedChangeAmounts_Old()

2.7.14. Command: 6Eh / n – Read registers, Option '7' (Sums in FD)

     input: *<'7'>*
     output:*<'7'> <;> <GrandFiscalTurnover[1..14]> <;> <GrandFiscalVAT[1..14]> <;> <GrandFiscalStornoTurnover[1..14]> <;> <GrandFiscalStornoVAT[1..14]>*
     FPR operation: Read the Grand fiscal turnover sum and Grand fiscal VAT sum.

Input data :

'7'	1 symbol obligatory '7'

Output data:

'7'	1 symbol obligatory '7'
GrandFiscalTurnover	Up to 14 symbols for sum of turnover in FD
GrandFiscalVAT	Up to 14 symbols for sum of VAT value in FD
GrandFiscalStornoTurnover	Up to 14 symbols for sum of STORNO turnover in FD
GrandFiscalStornoVAT	Up to 14 symbols for sum of STORNO VAT value in FD

zfpdef: ReadGrandFiscalSalesAndStornoAmounts()

2.7.15. Command: 6Eh / n – Read registers, Option '9', electronic signature of last daily report

     input: *<'9'>*
     output: *<'9'> <;> <LastDailyReportSignature[40]>*
     FPR operation: Provides information about electronic signature of last daily report.

Input data :

'9'	1 symbol obligatory '9'

Output data:

'9'	1 symbol obligatory '9'
LastDailyReportSignature	40 symbols electronic signature

zfpdef: ReadLastDailySignature()

2.7.16. Command: 6Eh / n – Display daily turnover registers, Option ':'

     input: *<':'>*
     output: ACK
     FPR operation: Provides information about daily turnover on the FD client display

Input data :

':'	1 symbol obligatory ':'

Output data: n. a.

zfpdef: DisplayDailyTurnover()

2.7.17. Command: 6Fh / o – Read operator's report, Option '1' (general)

     input: *<'1'> <;> <OperNum[1..2]>*
     output:*<'1'> <;> <OperNum[1..2]> <;> <CustomersNum[1..5]> <;> <DiscountsNum[1..5]> <;> <DiscountsAmount[1..13]> <;> <AdditionsNum[1..5]> <;> <AdditionsAmount[1..13]> <;> <CorrectionsNum[1..5]> <;> <CorrectionsAmount[1..13]>*
     FPR operation: Read the total number of customers, discounts, additions, corrections and accumulated amounts by specified operator.

Input data :

'1'	1 symbol obligatory '1'
OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number

Output data:

'1'	1 symbol obligatory '1'
OperNum	Symbols from 1 to 20 corresponding to operator's number
CustomersNum	Up to 5 symbols for number of customers
DiscountsNum	Up to 5 symbols for number of discounts
DiscountsAmount	Up to 13 symbols for accumulated amount of discounts

AdditionsNum	Up to 5 symbols for number ofadditions
AdditionsAmount	Up to 13 symbols for accumulated amount of additions
CorrectionsNum	Up to 5 symbols for number of corrections
CorrectionsAmount	Up to 13 symbols for accumulated amount of corrections

zfpdef: ReadDailyGeneralRegistersByOperator(OperNum)

2.7.18. Command: 6Fh / o – Read operator's report, Option '2' (RA)

input: *<'2'> <;> <OperNum[1..2]>*

output:*<'2'> <;> <OperNum[1..2]> <;> <AmountRA_Payment0[1..13]> <;>*
<AmountRA_Payment1[1..13]> <;> <AmountRA_Payment2[1..13]> <;>
<AmountRA_Payment3[1..13]> <;> <AmountRA_Payment4[1..13]> <;>
<AmountRA_Payment5[1..13]> <;><AmountRA_Payment6[1..13]> <;>
<AmountRA_Payment7[1..13]> <;><AmountRA_Payment8[1..13]> <;>
<AmountRA_Payment9[1..13]> <;><AmountRA_Payment10[1..13]> <;>
<AmountRA_Payment11[1..13]> <;> <NoRA[5]>

FPR operation: Read the RA by type of payment and the total number of operations by specified operator.

Input data :

'2'	1 symbol obligatory '2'
OperNum	(Operator Number) Symbols from 1 to 20corresponding to operator's number

Output data:

'2'	1 symbol obligatory '2'
OperNum	Symbols from 1 to 20corresponding to operator's number
AmountRA_Payment0	Up to 13 symbols for the RA by type of payment 0
AmountRA_Payment1	Up to 13 symbols for the RA by type of payment 1
AmountRA_Payment2	Up to 13 symbols for the RA by type of payment 2
AmountRA_Payment3	Up to 13 symbols for the RA by type of payment 3
AmountRA_Payment4	Up to 13 symbols for the RA by type of payment 4
AmountRA_Payment5	Up to 13 symbols for the RA by type of payment 5
AmountRA_Payment6	Up to 13 symbols for the RA by type of payment 6
AmountRA_Payment7	Up to 13 symbols for the RA by type of payment 7
AmountRA_Payment8	Up to 13 symbols for the RA by type of payment 8
AmountRA_Payment9	Up to 13 symbols for the RA by type of payment 9
AmountRA_Payment10	Up to 13 symbols for the RA by type of payment 10
AmountRA_Payment11	Up to 13 symbols for the RA by type of payment 11
NoRA	5 symbols for the total number of operations

zfpdef: ReadDailyRAbyOperator(OperNum)

2.7.19. Command: 6Fh / o – Read operator's report, Option '2', KL (RA)

input: *<'2'> <;> <OperNum[1..2]>*
output: *<'2'> <;> <OperNum[1..2]> <;> <AmountRA_Payment0[1..13]> <;>*
<AmountRA_Payment1[1..13]> <;> <AmountRA_Payment2[1..13]> <;>
<AmountRA_Payment3[1..13]> <;> <AmountRA_Payment4[1..13]> <;> <;> <NoRA[5]>
FPR operation: Read the RA by type of payment and the total number of
operations by specified operator. Command works for KL version 2 devices.
Input data :
'2'   1 symbol obligatory '2'
OperNum   (Operator Number) Symbols from 1 to 20corresponding to operator's number
Output data:
'2'   1 symbol obligatory '2'
OperNum   Symbols from 1 to 20corresponding to operator's number
AmountRA_Payment0   Up to 13 symbols for the RA by type of payment 0
AmountRA_Payment1   Up to 13 symbols for the RA by type of payment 1
AmountRA_Payment2   Up to 13 symbols for the RA by type of payment 2
AmountRA_Payment3   Up to 13 symbols for the RA by type of payment 3
AmountRA_Payment4   Up to 13 symbols for the RA by type of payment 4
NoRA   5 symbols for the total number of operations
zfpdef: ReadDailyRAbyOperator_Old(OperNum)

2.7.20. Command: 6Fh / o – Read operator's report, Option '3' (PO)

input: *<'3'> <;> <OperNum[1..2]>*
output: *<'3'> <;> <OperNum[1..2]> <;> <AmountPO_Payment0[1..13]> <;>*
<AmountPO_Payment1[1..13]> <;> <AmountPO_Payment2[1..13]> <;>
<AmountPO_Payment3[1..13]> <;> <AmountPO_Payment4[1..13]> <;>
<AmountPO_Payment5[1..13]> <;> <AmountPO_Payment6[1..13]> <;>
<AmountPO_Payment7[1..13]> <;> <AmountPO_Payment8[1..13]> <;>
<AmountPO_Payment9[1..13]> <;> <AmountPO_Payment10[1..13]> <;>
<AmountPO_Payment11[1..13]> <;><NoPO[1..5]>
FPR operation: Read the PO by type of payment and the total number of
operations by specified operator
Input data :
'3'   1 symbol obligatory '3'
OperNum   (Operator Number) Symbols from 1 to 20 corresponding to operator's number
Output data:
'3'   1 symbol obligatory '3'
OperNum   Symbols from 1 to 20corresponding to operator's number
AmountPO_Payment0   Up to 13 symbols for the PO by type of payment 0
AmountPO_Payment1   Up to 13 symbols for the PO by type of payment 1
AmountPO_Payment2   Up to 13 symbols for the PO by type of payment 2
AmountPO_Payment3   Up to 13 symbols for the PO by type of payment 3
AmountPO_Payment4   Up to 13 symbols for the PO by type of payment 4
AmountPO_Payment5   Up to 13 symbols for the PO by type of payment 5
AmountPO_Payment6   Up to 13 symbols for the PO by type of payment 6
AmountPO_Payment7   Up to 13 symbols for the PO by type of payment 7
AmountPO_Payment8   Up to 13 symbols for the PO by type of payment 8
AmountPO_Payment9   Up to 13 symbols for the PO by type of payment 9
AmountPO_Payment10   Up to 13 symbols for the PO by type of payment 10
AmountPO_Payment11   Up to 13 symbols for the PO by type of payment 11
NoPO   5 symbols for the total number of operations
zfpdef: *ReadDailyPObyOperator(OperNum)*

2.7.21. Command: 6Fh / o – Read operator's report, Option '3', KL (PO)

input: *<'3'> <;> <OperNum[1..2]>*
output: *<'3'> <;> <OperNum[1..2]> <;> <AmountPO_Payment0[1..13]> <;> <AmountPO_Payment1[1..13]> <;> <AmountPO_Payment2[1..13]> <;> <AmountPO_Payment3[1..13]> <;> <AmountPO_Payment4[1..13]> <;> <;><NoPO[1..5]>*

FPR operation: Read the PO by type of payment and the total number of operations by specified operator. Command works for KL version 2 devices.

Input data :

'3'	1 symbol obligatory '3'
OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number

Output data:

'3'	1 symbol obligatory '3'
OperNum	Symbols from 1 to 20corresponding to operator's number
AmountPO_Payment0	Up to 13 symbols for the PO by type of payment 0
AmountPO_Payment1	Up to 13 symbols for the PO by type of payment 1
AmountPO_Payment2	Up to 13 symbols for the PO by type of payment 2
AmountPO_Payment3	Up to 13 symbols for the PO by type of payment 3
AmountPO_Payment4	Up to 13 symbols for the PO by type of payment 4
NoPO	5 symbols for the total number of operations

zfpdef: *ReadDailyPObyOperator_Old(OperNum)*

2.7.22. Command: 6Fh / o – Read operator's report, Option '4' (received)

input: *<'4'> <;> <OperNum[1..2]>*
output: *<'4'> <;> <OperNum[1..2]> <;> <ReceivedSalesAmountPayment0[1..13]> <;> <ReceivedSalesAmountPayment1[1..13]> <;> <ReceivedSalesAmountPayment2[1..13]> <;> <ReceivedSalesAmountPayment3[1..13]> <;> <ReceivedSalesAmountPayment4[1..13]> <;> <ReceivedSalesAmountPayment5[1..13]> <;> <ReceivedSalesAmountPayment6[1..13]> <;> <ReceivedSalesAmountPayment7[1..13]> <;> <ReceivedSalesAmountPayment8[1..13]> <;> <ReceivedSalesAmountPayment9[1..13]> <;> <ReceivedSalesAmountPayment10[1..13]> <;> <ReceivedSalesAmountPayment11[1..13]>*

FPR operation: Read the amounts received from sales by type of payment and specified operator.

Input data :

'4'	1 symbol obligatory '4'
OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number

Output data:

'4'	1 symbol obligatory '4'
OperNum	Symbols from 1 to 20 corresponding to operator's number
ReceivedSalesAmountPayment0	Up to 13 symbols for amounts received by sales for payment 0
ReceivedSalesAmountPayment1	Up to 13 symbols for amounts received by sales for payment 1
ReceivedSalesAmountPayment2	Up to 13 symbols for amounts received by sales for payment 2
ReceivedSalesAmountPayment3	Up to 13 symbols for amounts received by sales for payment 3
ReceivedSalesAmountPayment4	Up to 13 symbols for amounts received by sales for payment 4
ReceivedSalesAmountPayment5	Up to 13 symbols for amounts received by sales for payment 5
ReceivedSalesAmountPayment6	Up to 13 symbols for amounts received by sales for payment 6
ReceivedSalesAmountPayment7	Up to 13 symbols for amounts received by sales for payment 7
ReceivedSalesAmountPayment8	Up to 13 symbols for amounts received by sales for payment 8
ReceivedSalesAmountPayment9	Up to 13 symbols for amounts received by sales for payment 9
ReceivedSalesAmountPayment10	Up to 13 symbols for amounts received by sales for payment 10
ReceivedSalesAmountPayment11	Up to 13 symbols for amounts received by sales for payment 11

zfpdef: *ReadDailyReceivedSalesAmountsByOperator(OperNum)*

2.7.23. Command: 6Fh / o – Read operator's report, Option '4', KL (received)

input: *<'4'> <;> <OperNum[1..2]>*

output:<'4'> <;> <OperNum[1..2]> <;> <ReceivedSalesAmountPayment0[1..13]> <;> <ReceivedSalesAmountPayment1[1..13]> <;> <ReceivedSalesAmountPayment2[1..13]> <;> <ReceivedSalesAmountPayment3[1..13]> <;> <ReceivedSalesAmountPayment4[1..13]>*

FPR operation: Read the amounts received from sales by type of payment and specified operator. Command works for KL version 2 devices.

Input data :

'4'	1 symbol obligatory '4'
OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number

Output data:

'4'	1 symbol obligatory '4'
OperNum	Symbols from 1 to 20 corresponding to operator's number
ReceivedSalesAmountPayment0	Up to 13 symbols for amounts received by sales for payment 0
ReceivedSalesAmountPayment1	Up to 13 symbols for amounts received by sales for payment 1
ReceivedSalesAmountPayment2	Up to 13 symbols for amounts received by sales for payment 2
ReceivedSalesAmountPayment3	Up to 13 symbols for amounts received by sales for payment 3
ReceivedSalesAmountPayment4	Up to 13 symbols for amounts received by sales for payment 4

zfpdef: ReadDailyReceivedSalesAmountsByOperator_Old(OperNum)

2.7.24. Command: 6Fh / o – Read operator's report, Option '5' (counters)

input: *<'5'> <;> <OperNum[1..2]>*

output:<'5'> <;> <OperNum[1..2]> <;> <WorkOperatorsCounter[1..5]> <;> <LastOperatorReportDateTime "DD-MM-YYYY HH:MM">*

FPR operation: Read the last operator's report number and date and time.

Input data :

'5'	1 symbol obligatory '5'
OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number

Output data:

'5"	1 symbol obligatory '5'
OperNum	Symbols from 1 to 20 corresponding to operator's number
WorkOperatorsCounter	Up to 5 symbols for number of the work operators
LastOperatorReportDateTime	16 symbols for date and time of the last operator's report in format *DD-MM-YYYY HH:MM*

zfpdef: ReadDailyCountersByOperator(OperNum)

2.7.25. Command: 6Fh / o – Read operator's report, Option '6' (change)

input: *<'6'> <;> <OperNum[1..2]>*
output: *<'6'> <;> <OperNum[1..2]> <;> <ChangeAmountPayment0[1..13]> <;>*
<ChangeAmountPayment1[1..13]> <;> <ChangeAmountPayment2[1..13]> <;>
<ChangeAmountPayment3[1..13]> <;> <ChangeAmountPayment4[1..13]> <;>
<ChangeAmountPayment5[1..13]> <;> <ChangeAmountPayment6[1..13]> <;>
<ChangeAmountPayment7[1..13]> <;> <ChangeAmountPayment8[1..13]> <;>
<ChangeAmountPayment9[1..13]> <;> <ChangeAmountPayment10[1..13]> <;>
<ChangeAmountPayment11[1..13]>

FPR operation: Read the amounts returned as change by different payment types for the specified operator.

Input data :

6'	1 symbol obligatory '6'
OperNum	(Operator Number) Symbol from 1 to 20 corresponding to operator's number

Output data:

'6'	1 symbol obligatory '6'
OperNum	Symbols from 1 to 20 corresponding to operator's number
ChangeAmountPayment0	Up to 13 symbols for amounts received by type of payment 0
ChangeAmountPayment1	Up to 13 symbols for amounts received by type of payment 1
ChangeAmountPayment2	Up to 13 symbols for amounts received by type of payment 2
ChangeAmountPayment3	Up to 13 symbols for amounts received by type of payment 3
ChangeAmountPayment4	Up to 13 symbols for amounts received by type of payment 4
ChangeAmountPayment5	Up to 13 symbols for amounts received by type of payment 5
ChangeAmountPayment6	Up to 13 symbols for amounts received by type of payment 6
ChangeAmountPayment7	Up to 13 symbols for amounts received by type of payment 7
ChangeAmountPayment8	Up to 13 symbols for amounts received by type of payment 8
ChangeAmountPayment9	Up to 13 symbols for amounts received by type of payment 9
ChangeAmountPayment10	Up to 13 symbols for amounts received by type of payment 10
ChangeAmountPayment11	Up to 13 symbols for amounts received by type of payment 11

zfpdef: ReadDailyReturnedChangeAmounts*ByOperator(OperNum)*

2.7.26. Command: 6Fh / o – Read operator's report, Option '6', KL (change)

input: *<'6'> <;> <OperNum[1..2]>*
output: *<'6'> <;> <OperNum[1..2]> <;> <ChangeAmountPayment0[1..13]> <;>*
<ChangeAmountPayment1[1..13]> <;> <ChangeAmountPayment2[1..13]> <;>
<ChangeAmountPayment3[1..13]> <;> <ChangeAmountPayment4[1..13]> <;>

FPR operation: Read the amounts returned as change by different payment types for the specified operator. Command works for KL version 2 devices.

Input data :

6'	1 symbol obligatory '6'
OperNum	(Operator Number) Symbol from 1 to 20 corresponding to operator's number

Output data:

'6'	1 symbol obligatory '6'
OperNum	Symbols from 1 to 20 corresponding to operator's number
ChangeAmountPayment0	Up to 13 symbols for amounts received by type of payment 0
ChangeAmountPayment1	Up to 13 symbols for amounts received by type of payment 1
ChangeAmountPayment2	Up to 13 symbols for amounts received by type of payment 2
ChangeAmountPayment3	Up to 13 symbols for amounts received by type of payment 3
ChangeAmountPayment4	Up to 13 symbols for amounts received by type of payment 4

zfpdef: ReadDailyReturnedChangeAmounts*ByOperator_Old(OperNum)*

2.7.27. Command: 70h / p – Read invoice number range

> **input:** *n. a.*
> **output:** *<StartNum[10]> <;> <EndNum[10]>*
> **FPR operation:** Provide information about invoice start and end numbers range.

Input data : n. a.
Output data :

StartNum	10 symbols for start No with leading zeroes in format ##########
EndNum	10 symbols for end No with leading zeroes in format ##########

zfpdef: ReadInvoiceRange*()*

2.7.28. Command: 71h / q – Read total receipt number

> **input:** n. a.
> **output:** *<TotalReceiptCounter[6]>*
> **FPR operation:** Read the total counter of last issued receipt.

Input data : n. a.
Output data :

TotalReceiptCounter	6 symbols for the total receipt counter in format ###### up to current last issued receipt by FD

zfpdef: *ReadLastReceiptNum()*

2.7.29. Command: 72h / r – Read information about current opened receipt

> **input:** n. a.
> **output:** *<IsReceiptOpened[1]> <;> <SalesNumber[3]> <;> <SubtotalAmountVAT0[1..13]> <;> <SubtotalAmountVAT1[1..13]> <;> <SubtotalAmountVAT2[1..13]> <;> <ForbiddenVoid[1]> <;> <VATinReceipt[1]> <;> <ReceiptFormat[1]> <;> <InitiatedPayment[1]> <;> <FinalizedPayment[1]> <;> <PowerDownInReceipt[1]> <;> <TypeReceipt[1]> <;> <ChangeAmount[1..13]> <;> <OptionChangeType[1]> <;> <SubtotalAmountVAT3[1..13]> <;> <SubtotalAmountVAT4[1..13]> <;> <SubtotalAmountVAT5[1..13]> <;> <SubtotalAmountVAT6[1..13]> <;> <SubtotalAmountVAT7[1..13]> <;> <CurrentReceiptNumber[6]>*
> **FPR operation:** Read the current status of the receipt.

Input data : n. a.
Output data :

IsReceiptOpened	1 symbol with value: - '0' - No - '1' - Yes
SalesNumber	3 symbols for number of sales in format ###
SubtotalAmountVAT0	Up to 13 symbols for subtotal by VAT group A
SubtotalAmountVAT1	Up to 13 symbols for subtotal by VAT group Б
SubtotalAmountVAT2	Up to 13 symbols for subtotal by VAT group В
ForbiddenVoid	1 symbol with value: - '0' – allowed - '1' - forbidden
VATinReceipt	1 symbol with value: - '0' – No - '1' - Yes
ReceiptFormat	(Format) 1 symbol with value: - '1' - Detailed - '0' - Brief
InitiatedPayment	1 symbol with value: - '0' – No - '1' – Yes

FinalizedPayment	1 symbol with value:
	- '0' - No
	- '1' - Yes
PowerDownInReceipt	1 symbol with value:
	- '0' - No
	- '1' - Yes
TypeReceipt	*(Receipt and Printing type) 1 symbol with value:*
	- '0' - Sales receipt printing step by step
	- '2' - Sales receipt Postponed Printing
	- '4' - Storno receipt printing step by step
	- '6' - Storno receipt Postponed Printing
	- '1' – Invoice sales receipt printing step by step
	- '3' – Invoice sales receipt Postponed Printing
	- '5' – Invoice Credit note receipt printing step by step
	- '7' – Invoice Credit note receipt Postponed Printing
ChangeAmount	Up to 13 symbols the amount of the due change in the stated payment type
OptionChangeType	1 symbol with value:
	- '0' - Change In Cash
	- '1' - Same As The payment
	- '2' – Change In Currency
SubtotalAmountVAT3	Up to 13 symbols for subtotal by VAT group Г
SubtotalAmountVAT4	Up to 13 symbols for subtotal by VAT group Д
SubtotalAmountVAT5	Up to 13 symbols for subtotal by VAT group Е
SubtotalAmountVAT6	Up to 13 symbols for subtotal by VAT group Ж
SubtotalAmountVAT7	Up to 13 symbols for subtotal by VAT group З
CurrentReceiptNumber	6 symbols for fiscal receipt number in format ######

zfpdef: ReadCurrentReceiptInfo()

2.7.30. Command: 72h / r – Read information about current/last receipt payments amounts, Option P

input: *<'P'>*

output: *<'P'> <;> <IsReceiptOpened[1]> <;> <Payment0Amount[1..13]> <;> <Payment1Amount[1..13]> <;> <Payment2Amount[1..13]> <;> <Payment3Amount[1..13]> <;> <Payment4Amount[1..13]> <;> <Payment5Amount[1..13]> <;> <Payment6Amount[1..13]> <;> <Payment7Amount[1..13]> <;> <Payment8Amount[1..13]> <;> <Payment9Amount[1..13]> <;> <Payment10Amount[1..13]> <;> <Payment11Amount[1..13]>*

FPR operation: Provides information about the payments in current receipt. This command is valid after receipt closing also.

Input data :

'P'	1 symbol 'P' for option

Output data :

'P'	1 symbol 'P' for option
IsReceiptOpened	1 symbol with value:
	- '0' - No
	- '1' - Yes
Payment0Amount	Up to 13 symbols for type 0 payment amount
Payment1Amount	Up to 13 symbols for type 1 payment amount
Payment2Amount	Up to 13 symbols for type 2 payment amount
Payment3Amount	Up to 13 symbols for type 3 payment amount
Payment4Amount	Up to 13 symbols for type 4 payment amount
Payment5Amount	Up to 13 symbols for type 5 payment amount
Payment6Amount	Up to 13 symbols for type 6 payment amount
Payment7Amount	Up to 13 symbols for type 7 payment amount

Payment8Amount	Up to 13 symbols for type 8 payment amount
Payment9Amount	Up to 13 symbols for type 9 payment amount
Payment10Amount	Up to 13 symbols for type 10 payment amount
Payment11Amount	Up to 13 symbols for type 11 payment amount

zfpdef: ReadCurrentOrLastReceiptPaymentAmounts()

2.7.31. Command: 72h / r – Read information about last receipt QR barcode data, Option B

>**input:** <'B'>
>**output:** <QRcodeData[60]>
>**FPR operation:** Provides information about the QR code data in last issued receipt.

Input data :

| 'B' | 1 symbol 'B' for option |

Output data :

| QRcodeData | Up to 60 symbols for last issued receipt QR code data separated by symbol '*' in format: FM Number*Receipt Number*Receipt Date*Receipt Hour*Receipt Amount |

zfpdef: ReadLastReceiptQRcodeData()

2.7.32. Command: 73h / s– Read last daily report info

>**input: n. a.**
>**output:**<LastZDailyReportDate "DD-MM-YYYY"> <;> <LastZDailyReportNum[1..4]> <;> <LastRAMResetNum[1..4]> <;> <TotalReceiptCounter[6]> <;> <DateTimeLastFiscRec "DD-MM-YYYY HH:MM"> <;> <EJNum[2]> <;> <TypeReceipt[1]>
>**FPR operation:** Read date and number of last Z-report and last RAM reset event.

Input data : n. a.

Output data :

LastZDailyReportDate	10 symbols for last Z-report date in DD-MM-YYYY format
LastZDailyReportNum	Up to 4 symbols for the number of the last daily report
LastRAMResetNum	Up to 4 symbols for the number of the last RAM reset
TotalReceiptCounter	6 symbols for the total number of receipts in format ######
DateTimeLastFiscRec	Date Time parameter in format: DD-MM-YYYY HH:MM
EJNum	Up to 2 symbols for number of EJ
TypeReceipt	(Receipt and Printing type) 1 symbol with value:
	- '0' - Sales receipt printing step by step
	- '2' - Sales receipt Postponed Printing
	- '4' - Storno receipt printing step by step
	- '6' - Storno receipt Postponed Printing
	- '1' – Invoice sales receipt printing step by step
	- '3' – Invoice sales receipt Postponed Printing
	- '5' – Invoice Credit note receipt printing step by step
	- '7' – Invoice Credit note receipt Postponed Printing

zfpdef: ReadLastDailyReportInfo()

2.7.33. Command: 74h / t – Read free FM reporting records

>**input: n. a.**
>**output:**<FreeFMrecords[4]>
>**FPR operation:** Read the number of the remaining free records for Z-report in the Fiscal Memory.

Input data : n. a.

Output data :

| FreeFMrecords | 4 symbols for the number of free records for Z-report in the FM |

zfpdef: ReadFMfreeRecords()

2.7.34. Command: 75h / u – Read FM contents

input: n. a.
output: ACK +
end number of packed messages for every block stored in FM:

<Nsegm[4]><OptionCodeStored[1]><DateStored[16]><Status[1]><ReadData [~]> +
a message for end of string: <Nsegm[4]><'@'>

FPR operation: Provides consequently information about every single block stored in the FM starting with Acknowledgements and ending with end message.

Input data : n. a.
Output data :

Nsegm	4 symbols for physical FM block number
OptionCodeStored	1 symbol stating the type of the stored block with the following values:
	0 – factory number of FD
	1 – VAT number, decimal point position and VAT rate at fiscalization
	4 – daily financial report
	5 – RAM reset
	6 – change of VAT rates
	7 – change of decimal point position
	: – deleted EJ
DateStored	16 symbols for the date and time of block storing
Status	1 symbol 0 or 1 resp. for correct/incorrect block checksum
ReadData	Total fields of data read
<'@'>	1 symbol obligatory '@' for end of string

zfpdef: ReadFMcontent()

2.8. REPORTS PRINTING COMMANDS

Set of commands for printing of reports generated by FD.

2.8.1. Command: 76h / v – Print department report

input: <*OptionZeroing*[1]>
output: ACK
FPR operation: Print a department report with or without zeroing ('**Z**' or **'X'**).

Input data :
OptionZeroing   (Parameter Zero) 1 symbol with value:
                         - 'Z' – Zeroing
                         - 'X' – Without zeroing

Output data : n. a.
zfpdef: PrintDepartmentReport(OptionZeroing)

2.8.2. Command: 77h / w – Print special events FM report

input: n. a.
output: ACK
FPR operation: Print whole special FM events report.

Input data : n. a.
Output data : n. a.
zfpdef: PrintSpecialEventsFMreport()

2.8.3. Command: 77h / w – Print brief FM payments report

input: <Option['P']>
output: ACK
FPR operation: Prints a brief payments from the FM.

Input data :
Option   1 symbol with value 'P'
Output data : n. a.
zfpdef: PrintBriefFMPaymentsReport()

2.8.4. Command: 78h / x – Print detailed FM report by number of Z report blocks

input: <*StartZNum*[4]> <;> <*EndZNum*[4]>
output: ACK
FPR operation: Print a detailed FM report by initial and end FM report number.

Input data :
StartZNum   4 symbols for the initial report number included in report, format ####
EndZNum   4 symbols for the final report number included in report, format ####
Output data: n. a.
zfpdef: PrintDetailedFMReportByZBlocks(StartNo, EndNo)

2.8.5. Command: 78h / x – Print detailed FM payments report by number of Z report blocks

      input: *<StartZNum[4]> <;> <EndZNum[4]> <;> <Option['P']>*
      output: ACK
      FPR operation: Print a detailed FM payments report by initial and end FM report number.

Input data :
StartZNum	4 symbols for initial FM report number included in report, format ####
EndZNum	4 symbols for final FM report number included in report, format ####
Option	1 symbol 'P'

Output data: n. a.
zfpdef: *PrintDetailedFMPaymentsReportByZBlocks(StartNo, EndNo)*

2.8.6. Command: 79h / y – Print brief FM report by number of Z report blocks

      input: *<StartZNum[4]> <;> <EndZNum[4]>*
      output: ACK
      FPR operation: Print a brief FM report by initial and end FM report number.

Input data :
StartZNum	4 symbols for the initial FM report number included in report, format ####
EndZNum	4 symbols for the final FM report number included in report, format ####

Output data: n. a.
zfpdef: *PrintBriefFMReportByZBlocks(StartNo, EndNo)*

2.8.7. Command: 79h / y – Print brief FM payments report by number of Z report blocks

      input: *<StartZNum[4]> <;> <EndZNum[4]> <;> <Option['P']>*
      output: ACK
      FPR operation: Print a brief FM payments report by initial and end FM report number.

Input data :
StartZNum	4 symbols for the initial FM report number included in report, format ####
EndZNum	4 symbols for the final FM report number included in report, format ####
Option	1 symbol 'P'

Output data: n. a.
zfpdef: *PrintBriefFMPaymentsReportByZBlocks(StartNo, EndNo)*

2.8.8. Command: 7Ah / z – Print detailed FM report by date

      input: *<StartDate "DDMMYY"> <;> <EndDate "DDMMYY">*
      output: ACK
      FPR operation: Prints a detailed FM report by initial and end date.

Input data :
StartDate	6 symbols for initial date in the DDMMYY format
EndDate	6 symbols for final date in the DDMMYY format

Output data: n. a.
zfpdef: *PrintDetailedFMReportByDate(StartDate, EndDate)*

2.8.9. Command: 7Ah / z – Print detailed FM payments report by date

input: *<StartDate "DDMMYY"> <;> <EndDate "DDMMYY"> <;> <Option['P']>*
output: ACK
FPR operation: Print a detailed FM payments report by initial and end date.

Input data :
StartDate   6 symbols for initial date in the DDMMYY format
EndDate   6 symbols for final date in the DDMMYY format
Option   1 symbol 'P'
Output data: n. a.
zfpdef: PrintDetailedFMPaymentsReportByDate(StartDate, EndDate)

2.8.10. Command: 7Bh / { – Print brief FM report by date

input: *<StartDate "DDMMYY"> <;> <EndDate "DDMMYY">*
output: ACK
FPR operation: Print a brief FM report by initial and end date.

Input data :
StartDate   6 symbols for initial date in the DDMMYY format
EndDate   6 symbols for final date in the DDMMYY format
Output data: n. a.
zfpdef: PrintBriefFMReportByDate(StartDate, EndDate)

2.8.11. Command: 7Bh / { – Print brief FM payments report by date

input: *<StartDate "DDMMYY"> <;> <EndDate "DDMMYY"> <;> <Option['P']>*
output: ACK
FPR operation: Print a brief FM payments report by initial and end date.

Input data :
StartDate   6 symbols for initial date in the DDMMYY format
EndDate   6 symbols for final date in the DDMMYY format
Option   1 symbol 'P'
Output data: n. a.
zfpdef: PrintBriefFMPaymentsReportByDate(StartDate, EndDate)

2.8.12. Command: 7Ch / | – Print daily fiscal report X or Z

input: *<OptionZeroing[1]>*
output: ACK
FPR operation: Depending on the parameter prints:
   – daily fiscal report with zeroing and fiscal memory record, preceded by Electronic Journal report print ('Z');
   – daily fiscal report without zeroing ('X');

Input data:
OptionZeroing   1 character with following values:
                     - 'Z' – Zeroing
                     - 'X' – Without zeroing
Output data: n.a.
zfpdef: PrintDailyReport(OptionZeroing)

2.8.13. Command: 7Ch / | – Print/Store Electronic Journal report from date do date

> **input:** *<ReportStorage[2]> <;> <'D'> <;> <StartRepFromDate "DDMMYY"> <;> <EndRepFromDate "DDMMYY">*

> **output: ACK**

> **FPR operation:** Print or store Electronic Journal Report by initial and end date.

Input data:

ReportStorage	(EJ Report storage) 1 character with value:
	- 'J1' – Printing
	- 'J2' – USB storage
	- 'J4' – SD card storage
'D'	1 symbol 'D'
StartRepFromDate	6 symbols for initial date in the DDMMYY format
EndRepFromDate	6 symbols for final date in the DDMMYY format

Output data: n. a.

zfpdef: PrintOrStoreEJByDate(StartDate, EndDate)

2.8.14. Command: 7Ch / | – Print Electronic Journal report from date do date with selected documents content

> **input:** *<'j1'> <;> <'X'> <;> <FlagsReceipts [1]> <;> <FlagsReports [1]> <;> <'D'> <;> <StartRepFromDate "DDMMYY"> <;> <EndRepFromDate "DDMMYY">*

> **output: ACK**

> **FPR operation:** Print Electronic Journal Report by initial and end date, and selected documents content. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

'j1'	2 symbols with value 'j1'
'X'	1 symbol with value 'X'
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'
FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'D'	1 symbol 'D'
StartRepFromDate	6 symbols for initial date in the DDMMYY format
EndRepFromDate	6 symbols for final date in the DDMMYY format

Output data: n. a.

zfpdef: PrintEJByDateCustom(FlagsReceipts, FlagsReports, StartDate, EndDate)

2.8.15. Command: 7Ch / | – Print/Store Electronic Journal report from receipt number to receipt number

>**input:** *<ReportStorage[2]> <;> <'N'> <;> <StartRcpNum[6]> <;> <EndRcpNum[6]>*
>**output: ACK**
>**FPR operation:** Print or store Electronic Journal Report from receipt number to receipt number.

Input data:

ReportStorage	(EJ Report storage) 1 character with value:
	- 'J1' – Printing
	- 'J2' – USB storage
	- 'J4' – SD card storage
'N'	1 symbol 'N'
StartRcpNum	6 symbols for initial receipt number included in report, in format ######.
EndRcpNum	6 symbols for final receipt number included in report in format ######.

Output data: n. a.

zfpdef: PrintOrStoreEJByRcpNum(ReportStorage, StartNo, EndNo)

2.8.16. Command: 7Ch / | – Print Electronic Journal report from receipt number to receipt number with selected documents content

>**input:** *<'j1'> <;> <'X'> <;> <FlagsReceipts [1]> <;> <FlagsReports [1]> <;> <'N'> <;> <StartRcpNum[6]> <;> <EndRcpNum[6]>*
>**output: ACK**
>**FPR operation:** Print Electronic Journal Report from receipt number to receipt number and selected documents content. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

'j1'	2 symbols with value 'j1'
'X'	1 symbol with value 'X'
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'
FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'N'	1 symbol 'N'
StartRcpNum	6 symbols for initial receipt number included in report in format ######.
EndRcpNum	6 symbols for final receipt number included in report in format ######.

Output data: n. a.

zfpdef: PrintEJByRcpNumCustom(FlagsReceipts, FlagsReports, StartNo, EndNo)

2.8.17. Command: 7Ch / | – Print/Store Electronic Journal report by number of Z report blocks

> **input:** *<ReportStorage[2]> <;> <'Z'> <;> <StartZNum[4]> <;> <EndZNum[4]>*
> **output: ACK**
> **FPR operation:** Print or store Electronic Journal Report from by number of Z report

blocks.

Input data:

ReportStorage	(EJ Report storage) 1 character with value:
	- 'J1' – Printing
	- 'J2' – USB storage
	- 'J4' – SD card storage
'Z'	1 symbol 'Z'
StartZNum	4 symbols for initial number report in format ####
EndZNum	4 symbols for final number report in format ####

Output data: n. a.

zfpdef: PrintOrStoreEJByZBlocks(StartNo, EndNo)

2.8.18. Command: 7Ch / | – Print Electronic Journal report by number of Z report blocks with selected documents content

> **input:** *<'j1'> <;> <'X'> <;> <FlagsReceipts [1]> <;> <FlagsReports [1]> <;> <'Z'> <;> <StartZNum[4]> <;> <EndZNum[4]>*
> **output: ACK**
> **FPR operation:** Print Electronic Journal Report by number of Z report blocks and

selected documents content. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

'j1'	2 symbols with value 'j1'
'X'	1 symbol with value 'X'
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'
FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'Z'	1 symbol 'Z'
StartZNum	4 symbols for initial number report in format ####
EndZNum	4 symbols for final number report in format ####

Output data: n. a.

zfpdef: PrintEJByZBlocksCustom(FlagsReceipts, FlagsReports, StartNo, EndNo)

2.8.19. Command: 7Ch / | – Print/Store Electronic Journal report from beginning to end

>**input:** *<ReportStorage[2]> <;> <'*'>*
>**output: ACK**
>**FPR operation:** Print or store Electronic Journal report with all documents.

Input data:

ReportStorage (EJ Report storage) 1 character with value:
>- 'J1' – Printing
>- 'J2' – USB storage
>- 'J4' – SD card storage

 ''* 1 symbol '*'

Output data: n. a.
zfpdef: PrintOrStoreEJ()

2.8.20. Command: 7Ch / | – Print Electronic Journal report from beginning to end with selected documents content

>**input:** *<'j1'> <;> <'X'> <;> <FlagsReceipts [1]> <;> <FlagsReports [1]> <;> <'*'>*
>**output: ACK**
>**FPR operation:** Print Electronic Journal report with selected documents content.

FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

'j1' 2 symbols with value 'j1'
'X' 1 symbol with value 'X'
FlagsReceipts (Receipts included in EJ) 1 symbol for Receipts included in EJ:
>Flags.7=0
>Flags.6=1
>Flags.5=1 Yes, Flags.5=0 No (Include PO)
>Flags.4=1 Yes, Flags.4=0 No (Include RA)
>Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
>Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
>Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
>Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)

 ';' 1 symbol with value ';'
FlagsReports (Reports included in EJ) 1 symbol for Reports included in EJ:
>Flags.7=0
>Flags.6=1
>Flags.5=0
>Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
>Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
>Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
>Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
>Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)

 ''* 1 symbol '*'

Output data: n. a.
zfpdef: PrintEJCustom(FlagsReceipts, FlagsReports)

2.8.21. Command: 7Dh / } – Print operator's report

> **input:** *<OptionZeroing[1]> <;> <Number[1..2]>*
> **output: ACK**
> **FPR operation:** Prints an operator's report for a specified operator (0 = all

operators) with or without zeroing ('**Z**' or **'X'**). When a '**Z**' value is specified the report should include all operators.

Input data :

OptionZeroing	1 character (Parameter) with following values:
	- 'Z' – Zeroing
	- 'X' – Without zeroing
Number	(Operator Number) Symbols from 0 to 20corresponding to operator's number ,0 for all operators

Output data: n. a.

zfpdef: PrintOperatorReport(OptionZeroing, Number)

2.8.22. Command: 7Eh / ~ – Print article report

> **input:** *<OptionZeroing[1]>*
> **output: ACK**
> **FPR operation:** Prints an article report with or without zeroing ('**Z**' or **'X'**).

Input data :

OptionZeroing	1 character (Parameter) with following values:
	- 'Z' – Zeroing
	- 'X' – Without zeroing

Output data: n. a.

zfpdef: PrintArticleReport(OptionZeroing)

2.8.23. Command: 7Fh / ▓ – Print detailed daily report

> **input:** *<OptionZeroing[1]>*
> **output: ACK**
> **FPR operation:** Prints an extended daily financial report (an article report followed

by a daily financial report) with or without zeroing ('**Z**' or **'X'**).

Input data :

OptionZeroing	1 character (Parameter) with following values:
	- 'Z' –Zeroing
	- 'X' – Without zeroing

Output data: n. a.

zfpdef: PrintDetailedDailyReport(OptionZeroing)

2.8.24. Command: 52h / R – option X, Print Customer X or Z report

> **input:** *<OptionZeroing[1]>*
> **output:** *ACK*
> **FPR Operation:** Print Customer X or Z report

Input data:

OptionZeroing	1 character (Parameter) with following values:
	- 'Z' –Zeroing
	- 'X' – Without zeroing

Output data: n.a.

zfpdef: PrintCustomerReport(OptionZeroing)

2.9. REPORTS READING COMMANDS
Set of commands for reading of reports generated by FD.

2.9.1. Command: 7Ch / | – Read Electronic Journal report from date do date

input: *<ReportFormat[2]> <;> <'D'> <;> <StartRepFromDate "DDMMYY"> <;> <EndRepFromDate "DDMMYY">*

output: ACK+

FPR operation: Read Electronic Journal Report by initial to end date.

Input data:

ReportFormat	(EJ Report format) 1 character with value
	- 'J0' – Detailed EJ
	- 'J8' – Brief EJ
'D'	1 symbol 'D'
StartRepFromDate	6 symbols for initial date in the DDMMYY format
EndRepFromDate	6 symbols for final date in the DDMMYY format

Output data: n. a.

zfpdef: ReadEJByDate(ReportFormat, StartRepFromDate, EndRepFromDate)

2.9.2. Command: 7Ch / | – Read/Store Electronic Journal report from date do date with selected documents content and format

input: *<StorageReport[2]> <;> <CSVformat[1]> <;> <FlagsReceipts[1]> <;> <FlagsReports[1]> <;> <'D'> <;> <StartRepFromDate "DDMMYY"> <;> <EndRepFromDate "DDMMYY">*

output: ACK+

FPR operation: Read or Store Electronic Journal Report by initial to end date, CSV format option and document content. If CSV format is set the content can includes only fiscal receipts. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

StorageReport	(EJ Report storage) 1 character with value
	- 'j0' – To PC
	- 'j2' – To USB Flash Drive
	- 'j4' – To SD card
CSVformat	1 symbol with value:
	- 'C' – Yes
	- 'X' – No
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'

FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'D'	1 symbol 'D'
StartRepFromDate	6 symbols for initial date in the DDMMYY format
EndRepFromDate	6 symbols for final date in the DDMMYY format

Output data: n. a.

zfpdef: ReadEJByDateCustom(StorageReport, CSVformat, FlagsReceipts, FlagsReports, StartRepFromDate, EndRepFromDate)

2.9.3. Command: 7Ch / | – Read Electronic Journal report from receipt number to receipt number

> **input:** *<ReportFormat[2]> <;> <'N'> <;> <StartRcpNum[6]> <;> <EndRcpNum[6]>*
> **output: ACK+**
> **FPR operation:** Read Electronic Journal Report from receipt number to receipt

number.

Input data:

ReportFormat	(EJ Report format) 1 character with value
	- 'J0' – Detailed EJ
	- 'J8' – Brief EJ
'N'	1 symbol 'N'
StartRcpNum	6 symbols for initial receipt number included in report in format ######
EndRcpNum	6 symbols for final receipt number included in report in format ######

Output data: n. a.

zfpdef: ReadEJByReceiptNum(ReportFormat, StartNo, EndNo)

2.9.4. Command: 7Ch / | – Read/Store Electronic Journal report from receipt number to receipt number with selected documents content and format

> **input:** *<StorageReport[2]> <;> <CSVformat[1]> <;> <FlagsReceipts[1]> <;> <FlagsReports[1]> <;> <'N'> <;> <StartRcpNum[6]> <;> <EndRcpNum[6]>*
> **output: ACK+**
> **FPR operation:** Read or Store Electronic Journal Report from receipt number to receipt number, CSV format option and document content. If CSV format is set the content can includes only fiscal receipts. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

StorageReport	(EJ Report storage) 1 character with value
	- 'j0' – To PC
	- 'j2' – To USB Flash Drive
	- 'j4' – To SD card
CSVformat	1 symbol with value:
	- 'C' – Yes
	- 'X' - No
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:

	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'
FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'N'	1 symbol 'N'
StartRcpNum	6 symbols for initial receipt number included in report in format ######.
EndRcpNum	6 symbols for final receipt number included in report in format ######.

Output data: n. a.

zfpdef: ReadEJByReceiptNumCustom(StorageReport, CSVformat,FlagsReceipts, FlagsReports, StartNo, EndNo)

2.9.5. Command: 7Ch / | – Reading Electronic Journal report from number Z report to number Z report

input: *<ReportFormat[2]> <;> <'Z'> <;> <StartNo[4]> <;> <EndNo[4]>*

output: ACK+

FPR operation: Reading Electronic Journal Report by number of Z report blocks.

Input data:

ReportFormat	(EJ Report format) 1 character with value
	- 'J0' – Detailed EJ
	- 'J8' – Brief EJ
'Z'	1 symbol 'Z'
StartNo	4 symbols for initial number report in format ####
EndNo	4 symbols for final number report in format ####

Output data: n. a.

zfpdef: ReadEJByZBlocks(ReportFormat, StartNo, EndNo)

2.9.6. Command: 7Ch / | – Read/Store Electronic Journal report by number of Z report blocks with selected documents content and format

input: *<StorageReport[2]> <;> <CSVformat[1]> <;> <FlagsReceipts[1]> <;> <FlagsReports[1]> <;> <'Z'> <;> <StartZNum[4]> <;> <EndZNum[4]>*

output: ACK+

FPR operation: Read or Store Electronic Journal Report by number of Z report blocks, CSV format option and document content. If CSV format is set the content can includes only fiscal receipts. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

StorageReport	(EJ Report storage) 1 character with value
	- 'j0' – To PC
	- 'j2' – To USB Flash Drive
	- 'j4' – To SD card
CSVformat	1 symbol with value:
	- 'C' – Yes

	- 'X' - No
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'
FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'Z'	1 symbol 'Z'
StartZNum	4 symbols for initial number report in format ####
EndZNum	4 symbols for final number report in format ####

Output data: n. a.

zfpdef: ReadEJByZBlocksCustom(StorageReport, CSVformat, FlagsReceipts, FlagsReports, StartNo, EndNo)

2.9.7. Command: 7Ch / | – Read Electronic Journal report from beginning to end

> **input:** *<ReportFormat[2]> <;> <'*'>*
> **output: ACK+**
> **FPR operation:** Read Electronic Journal report with all documents.

Input data:

ReportFormat	(EJ Report format) 1 character with value
	- 'J0' – Detailed EJ
	- 'J8' – Brief EJ
''*	1 symbol '*'

Output data: n. a.

zfpdef: ReadEJ(ReportFormat)

2.9.8. Command: 7Ch / | – Read/Store Electronic Journal report from beginning to end with selected documents content and format

input: *<StorageReport[2]> <;> <CSVformat[1]> <;> <FlagsReceipts[1]> <;> <FlagsReports[1]> <;> <'*'>*

output: ACK+

FPR operation: Read or Store Electronic Journal report by CSV format option and document content selecting. If CSV format is set the content can includes only fiscal receipts. FlagsReceipts is a char with bits representing the receipt types. FlagsReports is a char with bits representing the report type.

Input data:

StorageReport	(EJ Report storage) 1 character with value
	- 'j0' – To PC
	- 'j2' – To USB Flash Drive
	- 'j4' – To SD card
CSVformat	1 symbol with value:
	- 'C' – Yes
	- 'X' - No
FlagsReceipts	(Receipts included in EJ) 1 symbol for Receipts included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=1 Yes, Flags.5=0 No (Include PO)
	Flags.4=1 Yes, Flags.4=0 No (Include RA)
	Flags.3=1 Yes, Flags.3=0 No (Include Credit Note)
	Flags.2=1 Yes, Flags.2=0 No (Include Storno Rcp)
	Flags.1=1 Yes, Flags.1=0 No (Include Invoice)
	Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)
';'	1 symbol with value ';'
FlagsReports	(Reports included in EJ) 1 symbol for Reports included in EJ:
	Flags.7=0
	Flags.6=1
	Flags.5=0
	Flags.4=1 Yes, Flags.4=0 No (Include FM reports)
	Flags.3=1 Yes, Flags.3=0 No (Include Other reports)
	Flags.2=1 Yes, Flags.2=0 No (Include Daily X)
	Flags.1=1 Yes, Flags.1=0 No (Include Daily Z)
	Flags.0=1 Yes, Flags.0=0 No (Include Duplicates)
'*'	1 symbol '*'

Output data: n. a.

zfpdef: ReadEJCustom(StorageReport, CSVformat, FlagsReceipts, FlagsReports)

2.10. NETWORK SETTINGS COMMANDS

Set of commands for network settings.

2.10.1. Command: 4Eh / N – Read last sent and first unsent receipts info

input: *<Option['R']> <;> <'R'><;><'R'>*
output: <LastSentRcpNum[6]> <;> <LastSentRcpDateTime "DD-MM-YYYY HH:MM"> <;> <FirstUnsentRcpNum[6]> <;> <FirstUnsentRcpDateTime "DD-MM-YYYY HH:MM"> <;> <NRA_ErrorMessage[100]>

FPR operation: Provides information about last sent receipt number and date time to NRA server and first unsent receipt number and date time to NRA. If there is no unsent receipt the number will be 0 and date time will be 00-00-0000 00:00 Parameter NRA_ErrorMessage provide error message from NRA server if exist. Command is not allowed if device is deregistered, not fiscalized or in opened receipt.

Input data :

Option	1 symbol with value 'R'
'R'	1 symbol with value 'R'
'R'	1 symbol with value 'R'

Output data:

LastSentRcpNum	Up to 6 symbols for the last sent receipt number to NRA server
LastSentRcpDateTime	16 symbols for the date and time of the last sent receipt to NRA server in format DD-MM-YYYY HH:MM
FirstUnsentRcpNum	Up to 6 symbols for the first unsent receipt number to NRA server
FirstUnsentRcpDateTime	16 symbols for the date and time of the first unsent receipt to NRA server in format DD-MM-YYYY HH:MM
NRA_ErrorMessage	Up to 100 symbols for error message from NRA server, if exist

zfpdef: ReadSentRcpInfoStatus()

3. SOFTWARE APPLICATION REQUIREMENTS

3.1. RULES FOR USING THE COMMANDS

The commands should be used observing the following rules:

- Do not send a subsequent command prior to receiving a response of the preceding one.
- Observe the sequence of sent and received messages.
- The number of the message in every subsequent command should differ from that in the preceding one.
- Observe the two status bites of the Acknowledgment response.
- When the information received is insufficient request detailed status information – Command 20h.
- Use unpacked messages to check the standby status of the FD.

3.2. SAMPLE SALE TRANSACTION OF FD

The sale transaction controlled by a software application (SA) is a procedure, which consists of several commands, of which obligatory are: initiation of customer fiscal receipt (command 30h), sale registration (command 31h or 32h), payment (command 35h) and finalization of the fiscal receipt (command 38h).

Sample command sequence for issuing a customer fiscal receipt:

- fiscal receipt opening (command 30h) – contains information about the operator's number and password, the type of receipt – detailed/brief, with/without VAT printing.
- sale registration (command 31h) – contains information about the article's name, price and VAT group as well as non-compulsory information about the quantity sold and value/percent discount/addition;
- subtotal amount (command 33h) – contains non-compulsory parameters for printing, external display visualization and value/percent discount/addition of the amount accumulated;
- current receipt information (command 72h) – requires a response from the FD, which contains the current parameters of the receipt, the number of sales, the accumulated amounts by VAT groups, information about initiated or finalized payments;
- calculation and payment of VAT on VAT account (command 36h) – performs automatic calculation of VAT in the receipt and its payment on VAT account;
- payment (command 35h) – contains information about the amount due and type of payment, which may cover partially or in full the grand total amount due as well as a parameter for calculation of change due;
- fiscal receipt closure (command 38h).

4. AUXILARY GS PROTOCOL (COMMANDS 1Dh)

GS command	Message from the AS	FP Answer
Information	<1Dh><Info> where: **Info** – character 49h - 'I'	<'I'><Number of printable characters per line[2]> <;> <PLU number[5]> <;> <Departments number[2]> <;> <Operators number[2]> <;> <VAT classs number[1]> <;> <header lines number[2]> <;> <payments number[2]> <;> <logo number[2]> <;> <reserve='00'[2]> <;> <receipt transaction number[3]> <;> <customers base info [4]> <0Ah>
Model	<1Dh><Model> where: Model – character 77h - 'M'	<'M'> <country[2]> <;> <encoding[1..20]> <;> <device type [2]> <;> <license No[2..4]> <;> <license date [10]> <;> <interfaces[1..40]> <;> <current interface[1..4]> <;> <battery support[0..12]> <;> <current battery suplay ('E' or 'B') [1]> <;> <sleep mode [1] > <;> < Sd card journal [1]> <;> <SD card format [1..4]> <;> <SD card additional DB [1]> <;> <server exchange[1]> <;> < Number of printable characters per line[2]> <;> <number of printable free text[2]> <;> <article name symbols in command[2]> <;> <printable article name symbols[2]> <;> <department name symbols in command[2]> <;> <printable department name symbols[2]> <;> <department first number[1]> <;> <PLU last number[5]> <;> <total number of headers> <;> <number of fiscal headers only> <;> <number of footers> <0Ah>