

## TABLE OF CONTENTS:

Change LOG .....	6
version 1809131448 .....	6
version 1809251730 .....	6
version 1809271050 .....	6
version 1809281300 .....	6
version 1810041010 .....	6
version 1810081621 .....	6
version 1810091542 .....	6
version 1810161100 .....	7
version 1811151556 .....	7
version 1812121625 .....	7
version 1812141409 .....	7
version 1901071057 .....	7
version 1901211436 .....	7
version 1909031127 .....	7
version 1910241630 .....	7
version 2007081630 .....	7
version 2008111400 .....	7
6 new commands added for reading and setting GPRS .....	7
version 2008181208 .....	7
version 2009241551 .....	7
version 2103081722 .....	7
added commands with zfpdefs: ActivateSIM, UnlockDeviceforUpdate .....	7
<b>1. COMMUNICATION PROTOCOL .....</b>	<b>8</b>
1.1. Message format from the software application to the FPR: .....	8
1.2. Message format from the FPR to the software application: .....	9
1.2.1. ACK response: .....	9
1.2.2. Message response .....	10
1.3. Short messages for testing the status of the FPR .....	10
<b>2. DESCRIPTION OF THE COMMANDS .....</b>	<b>11</b>
2.1. Format and presentation of commands .....	11
2.2. General commands .....	12
2.2.1. Command: 20h / SP - Status .....	12
2.2.2. Command: 21h / ! - Version .....	14
2.2.3. Command: 22h / " - Diagnostics .....	14
2.2.4. Command: 24h / # - Clear display .....	14
2.2.5. Command: 25h / % - Display text line 1 .....	14
2.2.6. Command: 26h / & - Display text line 2 .....	14
2.2.7. Command: 27h / ' - Display text lines 1 and 2 .....	15
2.2.8. Command: 28h / ( - Display date and time .....	15
2.2.9. Command: 29h / ) - Cut paper, FP only .....	15
2.2.11. Command: 2Ah / * - Cash drawer opening .....	15
2.2.11. Command: 2Bh / + - Paper feeding .....	15
2.2.12. Command: 51h / Q - Print barcode 'QP' .....	16
2.3. Fiscal commands .....	17
2.3.1. Command: 3Fh / ? - Software Reset .....	17
2.3.2. Command: 40h / @ - Set Manufacturing number .....	17
2.3.3.1. Command: 41h / A- Set VAT and fiscal number, Option 1 .....	17
2.3.3.2. Command: 41h / A - Confirm fiscalization, Option 2 .....	18
2.3.4. Command: 42h / B - Change VAT rates .....	18
2.3.5. Command: 43h / C - Change decimal point position .....	18
2.3.6. Command: 57h / W - Restore previous header, Option 0 .....	18
2.3.7. Command: 57h / W - Print current header, Option 1 .....	19
2.3.8. Command: 57h / W - Store current header into fiscal memory, Option 2 .....	19
2.3.9. Command: 5Ah / Z - Activation of SIM, Option 5 .....	19
2.3.10. Command: 5Ah / Z - Device unlock for update, Option 4 .....	19
2.4. Programming commands .....	20
2.4.1. Command: 44h / D - Program payment types .....	20
2.4.2. Command: 45h / E - Program parameters .....	21
2.4.3. Command: 46h / F - Program external display .....	21
2.4.4. Command: 47h / G - Program department .....	23
2.4.5. Command: 48h / H - Program date and time .....	23
2.4.6. Command: 49h / I - Program display greeting message .....	24
2.4.7. Command: 49h / I - Program header lines .....	24
2.4.8. Command: 49h / I - Program footer lines .....	24
2.4.9. Command: 49h / I - Program CIF name message .....	25
2.4.10. Command: 49h / I - Program storno name message .....	25
2.4.11. Command: 49h / I - Program VAT number .....	25
2.4.12. Command: 49h / I - Program customer receipt name .....	25

2.4.13. Command: 4Ah / J – Program operator's name and password .....	26
2.4.14. Command: 4Bh / K – Program article general registers, Option 1 .....	26
2.4.15. Command: 4Bh / K – Program article quantity in stock, Option 2 .....	27
2.4.17. Command: 4Bh / K – Program article price, Option 4 .....	27
2.4.18. Command: 4Bh / K – Program PLU Category, Option 5 .....	28
2.4.19. Command: 4Bh / K – Erase all PLU database, Option \$ .....	28
2.4.20. Command: 4Ch / L – Program logo without setting a number (default number 0) .....	28
2.4.21. Command: 4Dh / M- Program logo with setting a number .....	28
2.4.22. Command: 23h / # - Set active logo number .....	29
2.4.23. Command: 52h / R – Program Customer data, Option P .....	29
2.4.24. Command: 53h / S – Program Other Charges (Alte tax) name, Option A .....	29
2.4.25. Command: 53h / S –Program service contract date, Option 1 .....	30
2.4.26. Command: 53h / S –Disable the service contract function, Option 2 .....	30
2.4.27. Command: 53h / S – Program service contract warning messages, Option 3 .....	30
2.4.28. Command: 4Fh / O – Program the duplicates number of the invoice receipt .....	31
2.5. Data reading commands .....	32
2.5.1. Command: 60h / ` – Read Serial number .....	32
2.5.2. Command: 61h / a – Read VAT and FM numbers .....	32
2.5.3. Command: 62h / b – Read VAT rates .....	32
2.5.4. Command: 63h / c – Read decimal point position .....	33
2.5.5. Command: 64h / d – Read payment types .....	33
2.5.6. Command: 65h / e – Read parameters .....	33
2.5.7. Command: 67h / g – Read department registers .....	34
2.5.8. Command: 68h / h – Read date and time .....	35
2.5.9. Command: 69h / i – Read display greeting message .....	35
2.5.10. Command: 69h / i – Read header lines .....	35
2.5.11. Command: 69h / i – Read footer lines .....	36
2.5.12. Command: 69h / i – Read CIF name message .....	36
2.5.13. Command: 69h / i – Read storno name message .....	36
2.5.14. Command: 69h / i – Read VAT number .....	36
2.5.15. Command: 6Ah / j – Read operator's name and password .....	37
2.5.16. Command: 6Bh / k – Read article registers, Option “ (All) .....	37
2.5.17. Command: 6Bh / k – Read article registers, Option 1 (General) .....	38
2.5.18. Command: 6Bh / k – Read article registers, Option 2 (Quantity) .....	39
2.5.19. Command: 6Bh / k – Read article registers, Option 3 (Barcode) .....	39
2.5.20. Command: 6Bh / k – Read article registers, Option 4 (Price) .....	39
2.5.21. Command: 6Bh / k – Read article registers, Option 5 (Category) .....	40
2.5.22. Command: 6Ch / l – Print Logo .....	40
2.5.23. Command: 52h / R – Read customer database .....	40
2.5.24. Command: 53h / S – Read service contract date, Option 1 .....	41
2.5.25. Command: 53h / S – Read service contract warning messages, Option 3 .....	41
2.5.26. Command: 53h / S – Read Other Charges (Alte tax) name, Option A .....	42
2.5.27. Command: 58h / X –Reset Odometer function, Option Z .....	42
2.5.28. Command: 58h / X – Read Odometer function, Option R .....	42
2.5.29. Command: 4Fh / O – Read duplicate number of the invoice .....	43
2.6. Receipt operations commands .....	44
2.6.1. Command: 2Eh / . – Non-fiscal receipt opening .....	44
2.6.2. Command: 2Fh / / – Non-fiscal receipt closure .....	44
2.6.3. Command: 30h / 0 – Open fiscal receipt .....	44
2.6.4. Command: 30h / 0 – Open fiscal special customer document .....	45
2.6.5. Command: 31h / 1 – Sell/Correction of article belonging to VAT class definition .....	45
2.6.6. Command: 31h / 1 – Sell/Correction of article belonging to department .....	46
2.6.7. Command: 31h / 1 – Sell/Correction of article with specified VAT belonging to department .....	47
2.6.8. Command: 32h / 2 – Sell/Correction of article from FD database .....	48
2.6.9. Command: 33h / 3 – Subtotal .....	48
2.6.10. Command: 33h / 3 – Subtotal with value discount from specified VAT definition .....	49
2.6.11. Command: 34h / 4 – Sell/Correction of article with department definition belonging to VAT class .....	50
2.6.12. Command: 34h / 4 – Sell/Correction of article with specified department .....	51
2.6.13. Command: 35h / 5 – Payment .....	52

2.6.14. Command: 35h / 5 – Pay exact sum.....	53
2.6.15. Command: 36h / 6 – Automatic receipt closure .....	53
2.6.16. Command: 37h / 7 – Free text printing .....	53
2.6.17. Command: 38h / 8 – Fiscal receipt closure.....	54
2.6.18. Command: 39h / 9 – Fiscal receipt cancel.....	54
2.6.19. Command: 3Ah / : – Print a copy of the last document.....	54
2.6.20. Command: 3Bh / ; – Non-fiscal RA and PO amounts .....	54
2.6.21. Command: 3Ch / < – Storno function of article with VAT class definition .....	55
2.6.22. Command: 3Ch / < – Storno function of article belonging to departament .....	56
2.6.23. Command: 3Ch / < – Storno function of article with specified VAT belonging to departament .....	57
2.6.24. Command: 3Dh / = – Sell / Correction of article with fractional quantity belonging to VAT class definition .....	58
2.6.25. Command: 3Dh / = – Sell / Correction of article with fractional quantity belonging to departament.....	59
2.6.26. Command: 3Dh / = – Sell / Correction of article with fractional quantity with specified VAT belonging to departament .....	60
2.6.27. Command: 3Eh / > – Discount/ addition .....	61
2.7. Sales operation commands for currency exchange offices .....	62
2.7.1. Command: 30h / 0 – Open receipt for CURRENCY SALE .....	62
2.7.2. Command: 30h / 0 – Open receipt for CURRENCY BUYING.....	62
2.7.3. Command: 31h / 1 – SELL/PURCHASE OF CURRENCY .....	63
2.8. Commands for reading the data in FD's registers .....	64
2.8.1. Command: 6Dh / m – Read amounts by VAT Class.....	64
2.8.2. Command: 6Eh / n – Read registers, Option '0' (on hand) .....	64
2.8.3. Command: 6Eh / n – Read registers, Option '1' (general) .....	65
2.8.4. Command: 6Eh / n – Read registers, Option '2' (RA) .....	65
2.8.5. Command: 6Eh / n – Read registers, Option '3' (PO).....	66
2.8.6. Command: 6Eh / n – Read registers, Option '4' (received).....	66
2.8.7. Command: 6Eh / n – Read registers, Option '5' (counters) .....	67
2.8.8. Command: 6Eh / n – Read daily registeras, Option '7' (Currency Exchange Offices).....	67
2.8.9. Command: 6Fh / o – Read operator's report, Option '1' (general) .....	67
2.7.10. Command: 6Fh / o – Read operator's report, Option '2' (RA) .....	68
2.7.11. Command: 6Fh / o – Read operator's report, Option '3' (PO) .....	68
2.7.15. Command: 6Fh / o – Read operator's report, Option '4' (received).....	69
2.7.16. Command: 6Fh / o – Read operator's report, Option '5' (counters) .....	70
2.8.15. Command: 6Fh / o – Read operators registers, Option '7' (Currency Exchange Offices) .....	71
2.8.16. Command: 71h / q – Read last and total receipt numbers.....	71
2.8.17. Command: 71h / q – Read receipt numbers (Currency Exchange Offices) .....	71
2.8.18. Command: 72h / r – Read information about the current opened receipt .....	72
2.8.19. Command: 73h / s – Read last daily report info .....	73
2.8.20. Command: 74h / t – Read free FM reporting records .....	73
2.8.21. Command: 75h / u – Read FM contents .....	73
2.8. Reports printing commands .....	74
2.9.1. Command: 76h / v – Print department report.....	74
2.9.2. Command: 77h / w – Print special events FM report .....	74
2.9.3. Command: 77h / w – Print special events FM report by number of Z reports .....	74
2.9.4. Command: 77h / w – Print special events FM report by date .....	74
2.9.5. Command: 78h / x – Print detailed FM report by number of Z reports .....	75
2.9.6. Command: 78h / x – Print detailed Payment FM report by number of Z reports .....	75
2.9.7. Command: 78h / x – Store detailed FM report by number of Z reports .....	75
2.9.8. Command: 79h / y – Print brief FM report by number of Z reports.....	75
2.9.9. Command: 79h / y – Print brief payment FM report by number of Z reports.....	76
2.9.10. Command: 79h / y – Store brief FM report by number of Z reports .....	76
2.9.11. Command: 7Ah / z – Print detailed FM report by date .....	76
2.9.12. Command: 7Ah / z – Print detailed Payment FM report by date .....	76
2.9.13. Command: 7Ah / z – Store detailed FM report by date.....	77
2.9.14. Command: 7Bh / { – Print brief FM report by date .....	77
2.9.15. Command: 7Bh / { – Print brief payment FM report by date .....	77
2.9.16. Command: 7Bh / { – Store brief FM report by date.....	77
2.9.17. Command: 7Ch /   – Print daily fiscal report X or Z.....	78
2.9.18. Command: 7Ch /   – Print/Store Electronic Journal report from date do date .....	78

2.9.19. Command: 7Ch /   – Print/Store Electronic Journal report from receipt number to receipt number .....	78
2.9.20. Command: 7Ch /   – Print/Store Electronic Journal report from number Z report to number Z report.....	79
2.9.21. Command: 7Ch /   – Print/Store Electronic Journal report from beginning to end.....	79
2.9.22. Command: 7Ch /   – Export xml format receipts from number Z report to number Z report .....	79
2.9.23. Command: 7Dh / } – Print operator's report .....	80
2.9.24. Command: 7Eh / ~ – Print article report.....	80
2.9.25. Command: 7Fh / █ – Print detailed daily report.....	80
2.9.26. Command: 52h / R – Print Customer X or Z report.....	80
2.9.27. Command: 59h / Y – Print hourly report, Option H .....	81
2.9.28. Command: 5Ah / Z – Read Z report number from date, Option 1 .....	81
2.9.29. Command: 5Ah / Z – Read Z report number from date, Option 2 .....	81
2.10. REPORTS READING COMMANDS .....	82
2.10.1. Command: 78h / x – Read detailed FM report by number of Z reports .....	82
2.10.2. Command: 79h / y – Read brief FM report by number of Z reports .....	82
2.10.3. Command: 7Ah / z – Read detailed FM report by date.....	82
2.10.4. Command: 7Bh / { – Read brief FM report by date .....	82
2.10.5. Command: 7Ch /   – Read Electronic Journal report from date do date .....	83
2.10.6. Command: 7Ch /   – Read/Store Electronic Journal report from date do date with selected documents content .....	83
2.10.7. Command: 7Ch /   – Read Electronic Journal report from receipt number to receipt number.....	84
2.10.8. Command: 7Ch /   – Read/Store Electronic Journal report from receipt number to receipt number with selected documents content .....	84
2.10.9. Command: 7Ch /   – Read Electronic Journal report from number Z report to number Z report .....	85
2.10.10. Command: 7Ch /   – Read/Store Electronic Journal report by number of Z report blocks with selected documents content .....	85
2.10.11. Command: 7Ch /   – Read Electronic Journal report from beginning to end.....	86
2.10.12. Command: 7Ch /   – Read/Store Electronic Journal report from beginning to end with selected documents content.....	86
2.11. SETTINGS LAN/WIFI/BLUETOOTH/GPRS COMMANDS.....	87
2.11.1. Command: 4Eh / N – Read Device modules support by Firmware.....	87
2.11.2. Command: 4Eh / N – Read Device modules support .....	88
2.11.3. Command: 4Eh / N – Read TCP password .....	88
2.11.4. Command: 4Eh / N – Read TCP Auto Start status .....	89
2.11.5. Command: 4Eh / N – Read Device TCP addresses .....	89
2.11.6. Command: 4Eh / N – Read TCP DHCP status .....	90
2.11.7. Command: 4Eh / N – Scan and print available WiFi networks.....	90
2.11.8. Command: 4Eh / N – Read TCP WiFi network name .....	90
2.11.9. Command: 4Eh / N – Read TCP WiFi password .....	91
2.11.10. Command: 4Eh / N – Read TCP module - LAN or WiFi .....	91
2.11.11. Command: 4Eh / N – Read TCP idle timeout .....	91
2.11.12. Command: 4Eh / N – Read Bluetooth password .....	92
2.11.13. Command: 4Eh / N – Read Bluetooth status .....	92
2.11.14. Command: 4Eh / N – Read GPRS APN .....	92
2.11.15. Command: 4Eh / N – Read TCP device MAC Address .....	92
2.11.16. Command: 4Eh / N – Read Server Address .....	94
2.11.17. Command: 4Eh / N – Read Server ECRS Password.....	94
2.11.18. Command: 4Eh / N – Read Server Communication Module.....	94
2.11.19. Command: 4Eh / N – Read GPRS password .....	95
2.11.20. Command: 4Eh / N – Read GPRS username.....	95
2.11.21. Command: 4Eh / N – Read GPRS signal .....	95
2.11.22. Command: 4Eh / N – Read profile type .....	96
2.11.23. Command: 4Eh / N – Read profile active date.....	96
2.11.24. Command: 4Eh / N – Read profile send after Z report.....	96
2.11.25. Command: 4Eh / N – Read profile connection period.....	97
2.11.26. Command: 4Eh / N – Set TCP password .....	97
2.11.27. Command: 4Eh / N – Set TCP Auto Start.....	97
2.11.28. Command: 4Eh / N – Set Device TCP addresses .....	98
2.11.29. Command: 4Eh / N – Set TCP DHCP enabled.....	98
2.11.30. Command: 4Eh / N – Set TCP WiFi network name .....	98

2.11.30. Command: 4Eh / N – Set TCP WiFi password .....	99
2.11.31. Command: 4Eh / N – Set TCP module - LAN or WiFi.....	99
2.11.32. Command: 4Eh / N – Set idle timeout.....	99
2.11.33. Command: 4Eh / N – Set Bluetooth password.....	100
2.11.34. Command: 4Eh / N – Set Bluetooth module enable status.....	100
2.11.35. Command: 4Eh / N – Set TCP device MAC Address .....	100
2.11.36. Command: 4Eh / N – Set Server ECRS Password.....	100
2.11.37. Command: 4Eh / N – Set Server Communication Module.....	101
2.11.38. Command: 4Eh / N – Unpair all connected devices - BT .....	101
2.11.39. Command: 4Eh / N – Save network settings .....	101
2.11.40. Command: 4Eh / N – Start device LAN test.....	101
2.11.41. Command: 4Eh / N – Start device WiFi test .....	102
2.11.42. Command: 4Eh / N – Start device GPRS test.....	102
2.11.43. Command: 4Eh / N – Start device Bluetooth test.....	102
2.11.44. Command: 4Eh / N – Set GPRS APN .....	102
2.11.45. Command: 4Eh / N – Set GPRS password .....	103
2.11.46. Command: 4Eh / N – Set GPRS username.....	103
<b>3. SOFTWARE APPLICATION REQUIREMENTS.....</b>	<b>104</b>
3.1. Rules for using the commands.....	104
3.2. Sample sale transaction of FPR.....	104
<b>4. AUXILARY GS PROTOCOL (COMMANDS 1Dh).....</b>	<b>105</b>

## CHANGE LOG

### VERSION 1809131448

Added Network commands for get and set LAN/WiFi/Bluetooth and GPRS.

Command 72h, [ReadCurrentRecInfo\(\)](#)

- Remove "Param" in the name of the output parameters

In the Communication Protocol: e.g. *ParamOpenRec* => *OpenRec* and etc.

In the generated Libraries: e.g. *OptionParamOpenRec* => *OptionOpenRec* and etc

Commands 31h, 34h, 3Ch and 3Dh changed position of the *DepNo* parameter.  
(SaleOrCorrection and Storno family of commands)

### VERSION 1809251730

No protocol changes

### VERSION 1809271050

No protocol changes

### VERSION 1809281300

New commands for Currency Exchange Offices printers are added:

Command 30h, [OpenFiscReceiptSaleVal](#)

Open fiscal receipt for Currency SALE transaction

Command 30h, [OpenFiscReceiptBuyVal](#)

Open fiscal receipt for Currency Purchase transaction

Command 31h, [CurrencyTransaction](#)

Currency SALE or Purchase transaction depend of receipt type opening

Command 6Eh, [ReadDailyValuta](#)

Reading of Daily registers for Currency Exchange Offices printers

Command 6Fh, [ReadOperValuta](#)

Reading of Operators registers for Currency Exchange Offices printers

Command 71h, [ReadLastReceiptNoVALUTA](#)

Reading of fiscal counters for Currency Exchange Offices printers

### VERSION 1810041010

New commands for make software reset is add:

Command 3Fh, [MakeSoftwareReset\(Password\)](#)

### VERSION 1810081621

Command 31h, [CurrencyTransaction](#): Added OptionPaymentType parameter

### VERSION 1810091542

Command 7Ch, [PrintEJreportByDate](#) and [ReadEJreportByDate](#) : changed parameters names  
from *StartDate* and *EndDate* to *StartRepFromDate* and *EndRepFromDate*

Command 7Ch, [PrintEJreportByRecNo](#) and [ReadEJreportByRecNo](#): changed parameters  
*StartNum* and *EndNum* length from 5 to 6

## **VERSION 1810161100**

Command 20h, [ReadStatus](#): status bit “near paper end” is added

## **VERSION 1811151556**

More of [zfpdef](#): names, parameter namers are changed.

## **VERSION 1812121625**

Add new commands with [zfpdef](#): [ReadDetailedFMReportByZNum](#), [ReadBriefFMReportByNum](#), [ReadDetailedFMReportByDate](#), [ReadBriefFMReportByDate](#) for storage FM reports to PC.

## **VERSION 1812141409**

Command with [zfpdef:CurrencyTransaction\(...\)](#) parameter name *OptionPaymentCurrType* is set with default value '0'.

## **VERSION 1901071057**

Fixed bug with formatted parameter values in multiple commands.

## **VERSION 1901211436**

Command 20h, [ReadStatus](#): status bit “Missing external display” is added

## **VERSION 1909031127**

Commands 31h, 34h, 3Ch and 3Dh are which changed *NamePLU* parameter description

## **VERSION 1910241630**

All commands starting with GET have been changed to start with READ

## **VERSION 2007081630**

All SETTINGS LAN/WIFI/BLUETOOTH/GPRS COMMANDS changed and 6 new commands were added

## **VERSION 2008111400**

6 new commands added for reading and setting GPRS

4Eh / N - Read Server Address – option ServerPassword[100] changed to ServerAddress[100]

## **VERSION 2008181208**

Added commands with [zfpdefs](#): [ReadEJByDateCustom](#), [ReadEJByReceiptNumCustom](#), [ReadEJByZBlocksCustom](#), [ReadEJCustom](#), for reading EJ with selected document content

## **VERSION 2009241551**

Added commands with [zfpdefs](#): [ReadECRprofileType](#), [ReadECRprofileActiveDate](#), [ReadECRprofileZreportSending](#), [ReadECRprofileConnectionPeriod](#), for reading ECR Profile data

## **VERSION 2103081722**

added commands with [zfpdefs](#): [ActivateSIM](#), [UnlockDeviceforUpdate](#)

## 1. COMMUNICATION PROTOCOL

The type of the protocol is Master / Slave. The communication session is always initiated by the Application Software. FPR carries out the commands sent by the software application and provides a feedback depending on the result. FPR sends back an „ACK response” or „message response”. All messages of the protocol are either packed or single-byte. FPR 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 FPR)

8 bits word

No parity

1 stop bit

### 1.1. MESSAGE FORMAT FROM THE SOFTWARE APPLICATION TO THE FPR:

All messages except those described in 3.4.3., sent to the FPR 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	<b>Message start</b> – always <b>02h</b>
LEN	1	<b>Message length</b> (number of bytes including LEN, NBL, CMD, DATA) increased by <b>20h</b> i.e. a number in the 20h - FFh range
NBL	1	<b>Message number</b> increased by <b>20h</b> i.e. a number in the 20h - 9Fh range
CMD	1	<b>Command</b> - a number in the 20h - 7Fh range(see the description of commands)
DATA.. DATA	0 - 3902	<b>Additional data</b> – 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	<b>Checksum</b> , 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	<b>End of message</b> – always <b>0Ah</b> (LF)

The text data of the message is sent as ASCII text with code table cp1252 (Windows 1252).



## 1.2. MESSAGE FORMAT FROM THE FPR TO THE SOFTWARE APPLICATION:

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

### 1.2.1. ACK RESPONSE:

**Positive ACK** – 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. of bytes	Value
ACK	1	06h
NBL	1	<b>No. of message</b> = NBL of message related to receipt
STE STE	2	<b>2 error status-bytes</b> . A two-digit ASCII number. (see Table Errors)
CS CS	2	<b>Checksum</b> , 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	FPR errors	Byte value	Command errors
30h	OK	30h	OK
31h	Out of paper, printer failure	31h	Invalid command
32h	Registers overflow	32h	Illegal command
33h	Clock failure or incorrect date&time	33h	Z daily report is not zero
34h	Opened fiscal receipt	34h	Syntax error
35h	Payment residue account	35h	Input registers overflow
36h	Opened non-fiscal receipt	36h	Zero input registers
37h	Registered payment but receipt is not closed	37h	Unavailable transaction for correction
38h	Fiscal memory failure	38h	Insufficient amount on hand
39h	Incorrect password	39h	Not used
3ah	Missing external display	3Ah	No access
3bh	24hours block – missing Z report	3Bh	Certificates are not loaded
3ch	Overheated printer thermal head.	3Ch	Illegal command, server communications are active
3dh	Interrupt power supply in fiscal receipt (one time until status is read)		
3eh	Overflow EJ		
3fh	Insufficient conditions		

A two-digit number is compiled depending on the type of error.

Example: Error 32 – Illegal command due to clock failure

**Negative ACK** – 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 FPR but is returned by the FPR to the SA and contains information – response to the query (see description of commands).

### 1.3. SHORT MESSAGES FOR TESTING THE STATUS OF THE FPR

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

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

## 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:

- ACK 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 9 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 :**

<i>N byte</i>	<i>N bit</i>	<i>status flag</i>
ST0	0	FM Read only
	1	Power down in opened fiscal receipt
	2	Printer not ready or overheated
	3	Incorrect time
	4	Incorrect date
	5	RAM reset
	6	Date and time hardware error
	7	Reserved

ST1	0	Printer not ready or no paper
	1	Reports registers overflow
	2	Blocking after 24 hours
	3	Non-zero daily report
	4	Non-zero article report
	5	Non-zero operator report
	6	Non-printed copy
	7	Reserved

ST2	0	Opened Non-fiscal Receipt
	1	Opened Fiscal Receipt
	2	Standard Cash Receipt
	3	VAT included in the receipt
	4	Reserved
	5	EJ near full
	6	EJ full
	7	Reserved

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	Reserved

ST4	0	Printer: automatic cutting
	1	External Display Management
	2	Reserved
	3	Missing external display
	4	Drawer: automatic opening
	5	Customer logo included in the receipt
	6	Service jumper
	7	Reserved

ST5	0	No Sec.IC
	1	No certificates
	2	Reserved
	3	Reserved
	4	No SD card response
	5	Wrong SD card
	6	Reserved
	7	Reserved

ST6	0	Reserved
	1	Reserved
	2	Reserved
	3	Reserved
	4	Near Paper end
	5	Reserved
	6	Reserved
	7	Reserved

**zfpdef:** [ReadStatus\(\)](#)

### 2.2.2. Command: 21h / ! - Version

input: n. a.

output: <Model[100]>

**FPR operation:** Provides information about the device model and firmware version.

**Input data :** n. a.

**Output data :**

Model                      Model information

**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 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-symbol text in the upper 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-symbol text in the lower 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 40-symbol text in the two display lines.

**Input data :**

Text                      40 symbols text

**Output data: ACK**

**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 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()

### 2.2.11. Command: 2Ah / \* - Cash drawer opening

input: n. a.

output: ACK

FPR operation: Opens 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 1 line of paper.

**Input data : n. a.**

**Output data : n. a.**

**zfpdef:** PaperFeed()

## 2.2.12. 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: <ul style="list-style-type: none"><li>- '0' – UPC A</li><li>- '1' – UPC E</li><li>- '2' – EAN 13</li><li>- '3' – EAN 8</li><li>- '4' – CODE 39</li><li>- '5' – ITF</li><li>- '6' – CODABAR</li><li>- 'H' – CODE 93</li><li>- 'I' – CODE 128</li></ul>
CodeLen	1..2 bytes for number of bytes according to the table
CodeData	From 0 to 255 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)



## 2.3. FISCAL COMMANDS

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

### 2.3.1. Command: 3Fh / ? – Software Reset

**input:** <Password[6]>

**output:** ACK

**FPR operation:** Restore default parameters of the device.

**Input data :**

Password            6-symbols string

**Output data: n. a.**

**zfpdef:** SoftwareReset(Password)

### 2.3.2. Command: 40h / @ – Set Manufacturing number

**input:** <Password[6]> <;> <SerialNum[11]>

**output:** ACK

**FPR operation:** Stores the Manufacturing number into the operative memory.

**Input data :**

Password            6-symbols string

SerialNum           (Serial Number) 11 symbol Manufacturing number

**Output data: n. a.**

**zfpdef:** SetSerialNum(Password, SerialNum)

#### 2.3.3.1. Command: 41h / A– Set VAT and fiscal number, Option 1

**input:** <Password[6]> <;> <'1'> <;> <VATNum[15]><;> <FMNum[10]><;>  
<TypeVATregistration[1]>

**output:** ACK

**FPR operation:** Stores the VAT registration and Fiscal Memory number into the operative memory.

**Input data :**

Password            6-symbols string

'1'                   One symbol is compulsory 1

VATNum              (VAT Number) 15 symbols VAT registration number

FMNum               (FM Number) 10 symbols Fiscal Memory serial number

TypeVATregistration   1 symbol for type of owner's VAT registration:

- '1' – Yes

- '0' – No

**Output data: n. a.**

**zfpdef:** SetFiscalNum(Password, VATNum, FMNum, TypeVATregistration )

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

**input:** <Password[6]> <;> <'2'> <;>

**output:** ACK

**FPR operation:** Confirm VAT number, type of VAT registration and Fiscal Memory number into the operative memory.

**Input data :**

Password            6-symbols string  
'2'                   One symbol for option

**Output data: n. a.**

**zfpdef:** ConfirmFiscalization(Password)

### 2.3.4. Command: 42h / B – Change VAT rates

**input:** <Password[6]> <;> <VATrateA[2..6]> <;> <VATrateB[2..6]> <;>  
<VATrateC[2..6]> <;> <VATrateD[2..6]> <;> <VATrateE['00.00']>

**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  
VATrateA           Value of VAT rate A from 2 to 6 symbols with format ##.##  
VATrateB           Value of VAT rate B from 2 to 6 symbols with format ##.##  
VATrateC           Value of VAT rate C from 2 to 6 symbols with format ##.##  
VATrateD           Value of VAT rate D from 2 to 6 symbols with format ##.##  
VATrateE           5 symbols with value '00.00'

**Output data: n. a.**

**zfpdef:** ProgVATrates(Password, VATRateA, VATRateB, VATRateC, VATRateD)

### 2.3.5. 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-symbol string  
DecimalPointPosition    1 symbol with values:  
                             - '0' - Whole numbers  
                             - '2' - Fractions

**Output data: n. a.**

**zfpdef:** ProgDecimalPointPosition(Password, OptionDecimalPoint)

### 2.3.6. Command: 57h / W – Restore previous header, Option 0

**input:** <'0'>

**output:** ACK

**FPR operation:** Restore previous header if current header is not saved into fiscal memory.

**Input data :**

'0'                   One symbol for option 0

**Output data: n. a.**

**zfpdef:** RestorePreviousHeader()

### 2.3.7. Command: 57h / W – Print current header, Option 1

input: <'1'>

output: ACK

FPR operation: Print current headers and Fiscal Memory operative header

**Input data :**

'1'                      One symbol for option 1

**Output data: n. a.**

**zfpdef:** *PrintCurrentHeader()*

### 2.3.8. Command: 57h / W – Store current header into fiscal memory, Option 2

input: <'2'> <;> <Password[6]>

output: ACK

FPR operation: Store the header into fiscal memory.

**Input data :**

'2'                      One symbol for option 2

Password                6-symbols string

**Output data: n. a.**

**zfpdef:** *StoreCurrentHeaderInFM(Password)*

### 2.3.9. Command: 5Ah / Z – Activation of SIM, Option 5

input: <'5'> <;> <Password[10]>

output: ACK

FPR operation: Activates the SIM card.

**Input data :**

'5'                      One symbol for option 5

Password                10-symbols string

**Output data: n. a.**

**zfpdef:** *ActivateSIM(Password)*

### 2.3.10. Command: 5Ah / Z – Device unlock for update, Option 4

input: <'4'> <;> <Password[10]>

output: ACK

FPR operation: Unlocks the device for update procedure.

**Input data :**

'4'                      One symbol for option 4

Password                10-symbols string

**Output data: n. a.**

**zfpdef:** *UnlockDeviceforUpdate(Password)*

## 2.4. PROGRAMMING COMMANDS

Set of commands, for programming the FPR 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:** Programs the name of the type of payment.

**Input data :**

PaymentNum	(Payment Types) 1 symbol for payment type: <ul style="list-style-type: none"><li>- '1' – Payment 1</li><li>- '2' – Payment 2</li><li>- '3' – Payment 3</li><li>- '4' – Payment 4</li><li>- '5' – Payment 5</li><li>- '6' – Payment 6</li><li>- '7' – Payment 7</li><li>- '8' – Payment 8</li><li>- '9' – Payment 9</li></ul>
Name	(Payment Name) 10 symbols for payment type name
Rate	(Exchange Rate) 10 symbols for exchange rate in format: #####.##### of the 9th payment type, maximal value 0420.00000

**Output data: n. a.**

**zfpdef:** *ProgPayment(PaymentType, Name, Rate)*

## 2.4.2. Command: 45h / E – Program parameters

**input:** <POSNum[4]> <;> <PrintLogo[1]> <;> <AutoOpenDrawer[1]> <;>  
<AutoCut[1]> <;> <ExternalDispManagement[1]> <;> <reserved[0]>  
<;><EnableCurrency[1]> <;><reserved[1]> <;> <USBHost[1]>

**output:** ACK

**FPR operation:** Programs the number of POS, printing of logo, Cash drawer opening, display mode, cutting permission. Changes in USBHost parameter will power off the printer.

### **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	(External Display Management) 1 symbol of value: - '1' - Manual - '0' - Auto
reserved	1 symbol reserved with value '0'
EnableCurrency	(Enable Currency) 1 symbol of value: - '1' – Yes - '0' – No
reserved	1 symbol reserved with value '1'
USBHost	(USB in host mode) 1 symbol, FP Only, with value: - '0' – No - '1' – Yes

**Output data: n. a.**

### **Notes:**

The logo is a graphical file in **BMP** format which is printed at the head of every receipt

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

[\*\*zfpdef:\*\* ProgParameters\(POSNum, PrintLogo, CashDrawer, AutoCut, ExternalDispManagment, ShortEJ, EnableCurrency\)](#)

## 2.4.3. Command: 46h / F – Program external display

**input:** <Password[6]> <NoBytesCom1line[1]> <Com1line[8]>  
<NoBytesCom2line[1]> <Com2Line[8]> <NoBytesClrDis[1]> <ComClrDis[8]>  
<NobytesXtrCom[1]> <ComXtrCom[1]> <FlagPrecod[1]> {<PrecodTabl[64]>}

**output:** ACK

**FPR operation:** Preprograms the external display.

### **Input data :**

Password	A 6-symbol string
NoBytesCom1line	Number of bytes (X = 1..8), for Command: show on line 1 of the display – 1 byte
Com1line	Command string show on line 1 of the display – 8 bytes, the first X bytes are command

<i>NoBytesCom2line</i>	Number of bytes (Y = 1..8), for Command: show on line 2 of the display – 1 byte
<i>Com2line</i>	String for Command show on line 2 of the display – 8 bytes, the first Y bytes are command
<i>NoBytesClrDis</i>	Number of bytes (Z = 1..8) – 1 byte
<i>ComClrDis</i>	String for Command clear display – 8 bytes, the first Z bytes are command
<i>NoBytesXtrCom</i>	Number of bytes (U = 0..8, 0 if there is no such command), for screensaver mode command – 1 byte, for hello message use line 0 of the template
<i>ComXtrCom</i>	String for Command for screensaver mode – 8 bytes, the first U bytes are command
<i>FlagPrecod</i>	Flag for precoding of the codetable for display cyrillization (0 – w/o precoding, 1 – with precoding) length 1 byte
<i>PrecodTabl</i>	Precoding table with the codes of the Cyrillic alphabet, capital and small letters

**Output data: n. a.**

#### **Notes:**

N command symbols should be specified for the number of bytes command. Then specify 8 bytes of control symbols, the first N of which are the command and the rest will be ignored. However, the symbols must be 8 in order to keep the format. If the display supports animation suitable for *screen-saver* - follow the above steps, otherwise set the <NoBytesXtrCom> as a 0. <FlagShift> is either 0 or 1 depending on whether a Cyrillic precoding is to be done or not. If precoding should be done input the code table.

**zfpdef:** ProgExtDisplay(Password, NoBytesCom1line, Com1line, NoBytesCom2line, Com2Line, NobytesClrDis, ComClrDis, NobytesXtrCom, ComXtrCom, FlagPrecod, PrecodTabl)

#### 2.4.4. Command: 47h / G – Program department

**input:** <Number[2]> <;><Name[20]> <;> <OptionVATClass[1]> {<;> <Price[1..10]>  
<;> <OptionDepPrice[1]><;> <AdditionalName[14]> <;> <Category[1..7]>}

**output:** ACK

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

**Input data:**

Number (Department No) 2 symbols department number in format: ##  
Name (Department Name) 20 characters department name  
OptionVATClass 1 symbol for article's VAT class with optional values:"

- 'A' – VAT Class A
- 'B' – VAT Class B
- 'C' – VAT Class C
- 'D' – VAT Class D
- 'E' – VAT Class E
- 'F' – Alte taxe

Price Up 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

AdditionalName 14 characters additional department name

Category From 1 to 7 symbols for category in format: #####.##

**Output data : n. a.**

**Note:**

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

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

#### 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 the current values using the RECEIPT printer. 64h

**Input data :**

DateTime Date Time parameter in format: DD-MM-YY [Space] HH:MM:SS

**Output data : n. a.**

**zfpdef:** *SetDateTime*(DateTime)

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

**input:** <'D'> <;> <DisplayGreetingText[20]>

**output:** ACK

**FPR operation:** Program the contents of a Display Greeting message.

**Input data :**

'D' 1 symbol with value 'D'

DisplayGreetingText 20 symbols for display greeting message

**Output data: n. a.**

**zfpdef:** *ProgDisplayGreetingMessage(DisplayGreetingText)*

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

**input:** <'H'> <;> <OptionHeaderLine[1]> <;> <HeaderText[TextLength]>

**output:** ACK

**FPR operation:** Program the contents of a header lines.

**Input data :**

'H' 1 symbol with value 'H'

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

- '8' – Header 8

HeaderText TextLength symbols for header lines

**Output data: n. a.**

**zfpdef:** *ProgHeader(OptionHeaderLine, Text)*

#### 2.4.8. Command: 49h / I – Program footer lines

**input:** <'F'> <;> <OptionFooterLine[1]> <;> <FooterText[TextLength]>

**output:** ACK

**FPR operation:** Program the contents of a footer lines.

**Input data :**

'F' 1 symbol 'F'

OptionFooterLine (Line Number) 1 symbol for option footer lines:

- '1' – Footer 1

- '2' – Footer 2

- '3' – Footer 3

FooterText TextLength symbols for footer line

**Output data: n. a.**

**zfpdef:** *ProgFooter(OptionFooterLine, FooterText)*



#### 2.4.9. Command: 49h / I – Program CIF name message

input: <'C'> <;> <Password[6]> <;> <CIFName[8]>

output: ACK

FPR operation: Program the contents of a CIF name message.

##### Input data :

'C'	1 symbol with value 'C'
Password	6-symbol string
CIFName	8 symbols for CIF name

Output data: n. a.

zfpdef: [ProgCIFNameMessage>Password, CIFName](#)

#### 2.4.10. Command: 49h / I – Program storno name message

input: <'S'> <;> <StornoName[TextLength]>

output: ACK

FPR operation: Program the contents of storno name message.

##### Input data :

'S'	1 symbol with value 'S'
StornoName	TextLength symbols for storno name

Output data: n. a.

zfpdef: [ProgStornoNameMessage](#)(StornoName)

#### 2.4.11. Command: 49h / I – Program VAT number

input: <'V'> <;> <Password[6]> <;> <OwnerVATNum[15]> <;> <TypeVATRegistration[1]>

output: ACK

FPR operation: Program the owner's VAT number and VAT registration type.

##### Input data :

'V'	1 symbol with value 'V'
Password	6-symbol string
OwnerVATNum	15 symbols for VAT number
TypeVATRegistration	1 symbol for type of owner's VAT registration: - '1' – Yes - '0' – No

Output data: n. a.

zfpdef: [ProgCustomerVATNum](#)(Password, OwnerVATNum, TypeVATRegistration)

#### 2.4.12. Command: 49h / I – Program customer receipt name

input: <'I'> <;> <CustRecieptName[TextLength]>

output: ACK

FPR operation: Program the contents of customer receipt name message.

##### Input data :

'I'	1 symbol with value 'I'
CustRecieptName	TextLength symbols for customer receipt name

Output data: n. a.

zfpdef: [ProgCustomerReceiptNameMessage](#)(CustRecieptName)

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

**input:** <Number[1..2]> <;> <Name[20]> <;> <Password[4]>

**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* 4 symbols for operator's password

**Output data : n. a.**

**zfpdef:** *ProgOperator(Number, Name, Password)*

### 2.4.14. 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]><;>  
<AlteTaxNum[1..11]><;> <AlteTaxValue[1..11]> {<;> <OptionTransaction[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* (PLU No) 5 symbols for article number in format: #####

*Option* '1'

*Name* 34 symbols for article name; Symbol for LF=7Ch - '|'; separator for MU=80h or 60h followed up to 3 symbols for unit.

*Price* 1 to 10 symbols for article price

*OptionPrice* 1 byte 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 symbol for article's VAT class with optional values:"

- 'A' – VAT Class A

- 'B' – VAT Class B

- 'C' – VAT Class C

- 'D' – VAT Class D

- 'E' – VAT Class E

- 'F' – Alte taxe

*BelongToDepNum* BelongToDepNo + 80h. 1 symbol for article department attachment, formed in the following manner: example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h

*AlteTaxNum* Up to 11 symbols for Alte Tax number

*AlteTaxValue* Up to 11 symbols for Alte tax value

*OptionTransaction* 1 symbol with value:

- '1' - Active Single transaction in receipt

- '0' – Inactive default value

Note: this parameter is not obligatory

**Output data: n. a.**

**zfpdef:** *ProgPLUGeneral(PLUNum, Name, Price, OptionPrice, OptionVATClass, BelongToDepNum, AlteTaxNum, AlteTaxValue, OptionTransaction)*

#### 2.4.15. 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 Quantity type for a certain article in the internal database.

**Input data :**

PLUNum	(PLU Number) 5 symbols for article number in format: #####
Option	'2'
AvailableQuantity	(Available Quantity) Up to 11 symbols for quantity in stock
OptionQuantityType	1 symbol for Quantity flag with next value: <ul style="list-style-type: none"><li>- '0'- Availability of PLU stock is not monitored</li><li>- '1'- Disable negative quantity</li><li>- '2'- Enable negative quantity</li></ul>

**Output data: n. a.**

**zfpdef:** ProgPLUqty(PLUNum, AvailableQuantity, OptionQuantityType)

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

**input:** <PLUNum[5]><;><Option['3']><;><Barcode[13]>

**output:** ACK

**FPR operation:** Program the Barcode number for a certain article (item) from the internal database.

**Input data :**

PLUNum	(PLU Number) 5 symbols for article number in format: #####
Option	'3'
Barcode	13 symbols for barcode

**Output data: n. a.**

**zfpdef:** ProgPLUbarcode(PLUNum, Barcode)

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

**input:** <PLUNum[5]><;><Option['4']><;><Price[1..10]><;><OptionPrice[1]>  
<;><AlteTaxNum[1..11]><;> <AlteTaxValue[1..11]>

**output:** ACK

**FPR operation:** Program the price for a certain article (item) from the internal database.

**Input data :**

PLUNum	(PLU Number) 5 symbols for article number in format: #####
Option	'4'
Price	Up to 10 symbols for article price
OptionPrice	1 byte for Price flag with next value: <ul style="list-style-type: none"><li>- '0'- Free price is disable valid only programmed price</li><li>- '1'- Free price is enable</li><li>- '2'- Limited price</li></ul>
AlteTaxNum	Up to 11 symbols for Alte Tax number
AlteTaxValue	Up to 11 symbols for Alte tax value

**Output data: n. a.**

**zfpdef:** ProgPLUprice(PLUNum, Price, OptionPrice, AlteTaxNum, AlteTaxValue)

#### 2.4.18. Command: 4Bh / K – Program PLU Category, Option 5

**input:** <PLUNum[5]><;><Option['5']><;><Category[1..7]>

**output:** ACK

**FPR operation:** Programs the PLU Category for a certain article (item) from the internal database.

**Input data :**

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

Option '5'

Category Up to 7 symbols for PLU Category code in format #####.##

**Output data: n. a.**

**zfpdef:** [ProgPLUcategory\(PLUNum, Category\)](#)

#### 2.4.19. Command: 4Bh / K – Erase all PLU database, Option \$

**input:** <PLUNum['00000']><;><Option['\$']><;><Password[6]>

**output:** ACK

**FPR operation:** Programs the PLU Category for a certain article (item) from the internal database.

**Input data :**

PLUNum 5 symbols '00000'

Option '\$'

Password 6 symbols for password

**Output data: n. a.**

**zfpdef:** [EraseAllPLUs\(PLUNo, Password\)](#)

#### 2.4.20. 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:**

FP 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\(BMPfile\)](#)

#### 2.4.21. Command: 4Dh / M- Program logo with setting a number

**input:** <LogoNum[1]> <BMPfile[9022]>

**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:**

LogoNum 1 character value from '0' to '9' setting the number where the logo will be saved.

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:** ProgLogoNum(LogoNum, BMPfile)

## 2.4.22. Command: 23h / # - Set active logo number

**Input:** <LogoNumber[1]>

**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' or '?'. The number sets the active logo number, and the '?' invokes only printing of information

**Output data: n. a.**

**zfpdef:** SetActiveLogo(LogoNumber)

## 2.4.23. Command: 52h / R – Program Customer data, Option P

**input:** <Option['P']> <;><CustomerNum[3]> <;> <VATNumber[15]> <;>  
<CustomerCompanyName[30]> <;> <Address[30]> <;><FreeLine1[20]> <;> <FreeLine2[20]>  
<;> <FreeLine3[20]> <;> <FreeLine4[20]>

**output:** ACK

**FPR operation:** Programs the customer DB for special customer receipt issuing.

**Input data :**

Option	'P' – for programming customer data
CustomerNum	3 symbols for customer number in order in format ###
VATNumber	15 symbols for customer VAT registration number
CustomerCompanyName	(Company name) 30 symbols for customer name
Address	30 symbols for address on customer
FreeLine1	20 ASCII symbols for customer data
FreeLine2	20 ASCII symbols for customer data
FreeLine3	20 ASCII symbols for customer data
FreeLine4	20 ASCII symbols for customer data

**Output data: n. a.**

**zfpdef:** ProgCustomerData(CustomerNum, VATNumber, CustomerCompanyName, Address, FreeLine1, FreeLine2, FreeLine3, FreeLine4)

## 2.4.24. Command: 53h / S – Program Other Charges (Alte taxe) name, Option A

**input:** <Option['A']> <;><Option['P']> <;><Number[1]> <;> <Name[12]>

**output:** ACK

**FPR operation:** Program the other charges (Alte taxe) name.

**Input data :**

Option	'A' – for alte taxe data
Option	'P' – for programming
Number	1 symbol for number in order up to 7
Name	12 symbols for Alte taxe name

**Output data: n. a.**

**zfpdef:** ProgAlteTaxe(Number, Name)

#### 2.4.25. Command: 53h / S –Program service contract date, Option 1

**input:** <Option['L']> <;><'P'[1]> <;><Password[6]> <;> <'1'><;>  
<ExpiryDate "DD-MM-YYYY">

**output:** ACK

**FPR operation:** Program the service contract expiry date.

**Input data :**

Option	'L' – for service contract
'P'	'P' – for programming
Password	6 symbols for service password
'1'	'1' – for date
ExpiryDate	10 symbols for expiry date of service contract

**Output data: n. a.**

**zfpdef:** ProgServiceContractDate>Password, ExpiryDate)

#### 2.4.26. Command: 53h / S –Disable the service contract function, Option 2

**input:** <Option['L']> <;><'P'[1]> <;><Password[6]> <;> <'2'[1]>

**output:** ACK

**FPR operation:** Set service contract disable

**Input data :**

Option	'L' – for service contract
'P'	'P' – for programming
Password	6 symbols for service password
'2'	'2' – for disable service contract function

**Output data: n. a.**

**zfpdef:** DisbleServiceContract>Password)

#### 2.4.27. Command: 53h / S – Program service contract warning messages, Option 3

**input:** <Option['L']> <;><'P'[1]> <;><Password[6]> <;> <'3'[1]><;>  
<Line1[TextLength]> <;><Line2[TextLength]> <;> <Line3[TextLength]>

**output:** ACK

**FPR operation:** Program the service contract expired warning message text.

**Input data :**

Option	'L' – for service contract
'P'	'P' – for programming
Password	6 symbols for service password
'3'	'3' – for warning message
Line1	TextLength symbols for warning message of line 1
Line2	TextLength symbols for warning message of line 2
Line3	TextLength symbols for warning message of line 3

**Output data: n. a.**

**zfpdef:** ProgContractWarningMessages>Password, Line1, Line2, Line3)

#### 2.4.28. Command: 4Fh / O – Program the duplicates number of the invoice receipt

**input:** <Option1['D']> <;><Option2['W']><;> <DuplicatesNumber[1]>

**output:** ACK

**FPR operation:** Program the the number of the duplicates which can be printed after invoice receipt.

**Input data :**

Option1	1 symbol with value 'D'
Option2	1 symbol with value 'W'
DuplicatesNumber	1 symbol for number of duplicates which can be print. Possible values from '0' to '5'.

**Output data: n. a.**

**zfpdef:** ProgDuplicatesNumber(DuplicatesNumber)

## 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 Serial number

**input:** n. a.

**output:** <SerialNum[11]>

**FPR operation:** Provides information about the manufacturing number of the fiscal device.

**Input data :** n. a.

**Output data :**

SerialNum      11 symbols for individual number of the fiscal device

**Zfpdef:** ReadSerialNum()

### 2.5.2. Command: 61h / a – Read VAT and FM numbers

**input:** n. a.

**output:** <VATNum[15]><;><FMnum[10]><;><TypeVATregistration[1]>

**FPR operation:** Read the VAT registration and Fiscal Memory numbers.

**Input data :** n. a.

**Output data :**

VATNum              15 symbols for owner's VAT registration number

FMnum              10 symbols for FM serial number

TypeVATregistration      1 symbol for type of owner's VAT registration:

- '1' – Yes

- '0' – No

**Zfpdef:** ReadVATNum()

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

**input:** n. a.

**output:** < VATRateA[1..6]> <;> < VATRateB[1..6]> <;> < VATRateC[1..6]> <;>  
<VATRateD[1..6]> <;> <VATRateE[1..6]> <;> <AlteTaxeF[1..6]>

**FPR operation:** Provides information about the current VAT rates (the last value stored in FM).

**Input data :** n. a.

**Output data :**

VATRateA          6 symbols for VATrates of VAT class A in format ###.##%

VATRateB          6 symbols for VATrates of VAT class B in format ###.##%

VATRateC          6 symbols for VATrates of VAT class C in format ###.##%

VATRateD          6 symbols for VATrates of VAT class D in format ###.##%

VATRateE          6 symbols for VATrates of VAT class E in format ###.##%

AlteTaxeF          6 symbols for VATrates of Alte Taxe F 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:** <NamePaym0[10]> <;> <NamePaym1[10]> <;> <NamePaym2[10]> <;>  
<NamePaym3[10]> <;> <NamePaym4[10]> <;> <NamePaym5[10]> <;>  
<NamePaym6[10]> <;> <NamePaym7[10]> <;> <NamePaym8[10]> <;>  
<NamePaym9[10]> <;> <ExchangeRate[10]>

**FPR operation:** Provides information about all programmed types of payment.

**Input data : n. a.**

**Output data :**

<i>NamePaym0</i>	10 symbols for type 0 of payment name
<i>NamePaym1</i>	10 symbols for type 1 of payment name
<i>NamePaym2</i>	10 symbols for type 2 of payment name
<i>NamePaym3</i>	10 symbols for type 3 of payment name
<i>NamePaym4</i>	10 symbols for type 4 of payment name
<i>NamePaym5</i>	10 symbols for type 5 of payment name
<i>NamePaym6</i>	10 symbols for type 6 of payment name
<i>NamePaym7</i>	10 symbols for type 7 of payment name
<i>NamePaym8</i>	10 symbols for type 8 of payment name
<i>NamePaym9</i>	10 symbols for type 9 of payment name
<i>ExchangeRate</i>	10 symbols for exchange rate of payment type 9 in format: #####.#####

**Zfpdef:** ReadPayments()

## 2.5.6. Command: 65h / e – Read parameters

**input:** n. a.

**output:** <POSNum[4]> <;> <PrintLogo[1]> <;> <AutoOpenDrawer[1]> <;>  
<AutoCut[1]> <;> <ExternalDispManagement[1]> <;> <reserved[0]>  
<;> <EnableCurrency[1]> <;> <reserved[1]> <;> <USBHost[1]>

**FPR operation:** Provides information about the programmed number of POS and the current values of the logo, cutting permission, display mode, enable/disable currency in receipt and enable/disable USB host mode.

**Input data : n. a.**

**Output data :**

<i>POSNum</i>	(POS No) 4 symbols for number of POS in format ####
<i>PrintLogo</i>	(Print Logo) 1 symbol of value: - '1' – Yes - '0' - No
<i>AutoOpenDrawer</i>	(Auto Open Drawer) 1 symbol of value: - '1' - Yes - '0' - No

<i>AutoCut</i>	(Auto Cut) 1 symbol of value: - '1' - Yes - '0' - No
<i>ExternalDispManagement</i>	(External Display Management) 1 symbol of value: - '1' - Manual - '0' - Auto
<i>reserved</i>	1 symbol reserved with value '0'
<i>EnableCurrency</i>	(Enable Currency) 1 symbol of value: - '1' – Yes - '0' – No
<i>reserved</i>	1 byte reserved with value '1'
<i>USBHost</i>	(USB in host mode)1 symbol with value: - '1' – Yes - '0' – No

**zfpdef:** ReadParameters()

### 2.5.7. Command: 67h / g – Read department registers

input: <DepNum[2]>

**output:** <DepNum[2]> <;><DepName[34]> <;> <OptionVATClass[1]> <;>  
<Turnover[1..11]> <;> <SoldQuantity[1..11]> <;><LastZReportNumber[1..5]><;>  
<LastZReportDate“DD-MM-YYYY HH:MM”> <;><Price[1..10]> <;> <OptionDepPrice[1]>  
<;> <Category[1..7]>

**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 :**

<i>DepNum</i>	1..2 symbols for department number in format ##
<i>DepName</i>	34 symbols for department name
<i>OptionVATClass</i>	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
<i>Turnover</i>	1..11 symbols for accumulated turnover of the department
<i>SoldQuantity</i>	1..11 symbols for sold quantity of the department
<i>LastZReportNumber</i>	1..5 symbols for the number of last Z report in format #####
<i>LastZReportDate</i>	16 symbols for the date and hour in last Z report
<i>Price</i>	1 to 10 symbols for department price
<i>OptionDepPrice</i>	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
<i>Category</i>	Up to 7 symbols for PLU Category code in format #####.##

**zfpdef:** ReadDepartment(DepNum)

### 2.5.8. Command: 68h / h – Read date and time

input: n. a.

output: <Date Time “DD-MM-YYYY HH:MM”>

**FPR operation:** Provides information about the current date and time.

**Input data : n. a.**

**Output data :**

*Date Time* Date Time parameter in format: DD-MM-YY [Space] hh:mm:ss

**Zfpdef:** [ReadDateTime\(\)](#)

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

input: <'D'>

output: <'D'> <;> <DisplayGreetingText[20]>

**FPR operation:** Provide information about the display greeting message.

**Input data :**

'D' 1 symbol with value 'D'

**Output data:**

'D' 1 symbol with value 'D'

*DisplayGreetingText* 20 symbols for greeting message

**Zfpdef:** [ReadDisplayGreetingMessage\(\)](#)

### 2.5.10. Command: 69h / i – Read header lines

input: <'H'><;><OptionHeaderLine[1]>

output: <'H'><;><OptionHeaderLine[1]> <;><HeaderText[TextLength]>

**FPR operation:** Provides the content of the header lines.

**Input data :**

*OptionHeaderLine* (Line Number)1 byte with value:

- '1' – Header 1
- '2' – Header 2
- '3' – Header 3
- '4' – Header 4
- '5' – Header 5
- '6' – Header 6
- '7' – Header 7
- '8' – Header 8

**Output data:**

'H' 1 symbol with value 'H'

*OptionHeaderLine* (Line Number)1 byte with value:

- '1' – Header 1
- '2' – Header 2
- '3' – Header 3
- '4' – Header 4
- '5' – Header 5
- '6' – Header 6
- '7' – Header 7
- '8' – Header 8

*HeaderText* LineLength symbols

**Zfpdef:** [ReadHeader\(OptionHeaderLine\)](#)

### 2.5.11. Command: 69h / i – Read footer lines

**input:** <'F'><;><OptionFooterLine[1]>

**output:**<'F'><;><OptionFooterLine[1]><;> <FooterText[TextLength]>

**FPR operation:** Provides the content of the footer lines.

**Input data :**

'F' 1 symbol with value 'F'  
OptionFooterLine (Line Number) 1 symbol with value:  
- '1' – Footer 1  
- '2' – Footer 2  
- '3' – Footer 3

**Output data:**

'F' 1 symbol with value 'F'  
OptionFooterLine (Line Number)1 symbol with value:  
- '1' – Footer 1  
- '2' – Footer 2  
- '3' – Footer 3

FooterText LineLength symbols for footer line

**Zfpdef:** [ReadFooter\(OptionFooterLine\)](#)

### 2.5.12. Command: 69h / i – Read CIF name message

**input:** <'C'>

**output:**<'C'> <;> <CIFName[8]>

**FPR operation:** Provide information about the CIF name message.

**Input data :**

'C' 1 symbol with value 'C'

**Output data:**

'C' 1 symbol with value 'C'  
CIFName 8 symbols for CIF name message

**Zfpdef:** [ReadCIFNameMessage\(\)](#)

### 2.5.13. Command: 69h / i – Read storno name message

**input:** <'S'>

**output:** <'S'> <;> <StornoName[TextLength]>

**FPR operation:** Provide information about the Storno Name Message.

**Input data :**

'S' 1 symbol with value 'S'

**Output data:**

'S' 1 symbol with value 'S'  
StornoName TextLength symbols for storno name

**Zfpdef:** [ReadStornoNameMessage\(\)](#)

### 2.5.14. Command: 69h / i – Read VAT number

**input:** <'V'>

**output:**<'V'> <;> <CustomerVATNum[15]><;> <TypeVATregistration[1]>

**FPR operation:** Provide information about the owner's VAT number and VAT registration type

**Input data :**

'V' 1 symbol with value 'V'

**Output data:**

'V' 1 symbol with value 'V'  
CustomerVATNum 15 symbols for VAT number

TypeVATregistration 1 symbol for type of owner's VAT registration:  
 - '1' – Yes  
 - '0' – No

**Zfpdef:** *ReadCustomerVATNum()*

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

**input:** <Number[1..2]>

**output:** <Number[1..2]> <;> <Name[20]> <;> <Password[4]>

**FPR operation:** Provides information about an operator's name and password.

### **Input data :**

Number (Operator No) 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 4 symbols for operator's password

**zfpdef:** *ReadOperatorNamePassword(Number)*

## 2.5.16. Command: 6Bh / k – Read article registers, Option “ (All)

**input:** <PLUNum[1..5]><;><Option[""]>

**output:** < PLUNum [1..5]><;><Option[""]><;><PLUName[34]><;><Price[1..10]><;>  
 <FlagsPricePLU[1]><;><OptionVATClass[1]><;> <BelongToDepNumber[1]> <;>  
 <AlteTaxNum[1..11]><;> <AlteTaxValue[1..11]><;> < Turnover[1..10]> <;>  
 <QuantitySold[1..11]><;><LastZReportNumber[1..5]><;>  
 <LastZReportDate“DD-MM-YYYY HH:MM”> <;> <AvailableQTY[1..11]><;>  
 <Barcode[13]><;> <AlteTaxAmount[1..11]> <;><Category[1..7]>

**FPR operation:** Provides information about the all 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 ""

### **Output data :**

PLUNum 5 symbols for article number with leading zeroes in format: #####

Option One symbol with value ""

PLUName 34 symbols for article name (LF=7Ch, MU separator = 80h or 60h followed up to 3 symbols for unit)

Price 1..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 symbol for article's VAT class with optional values:"

- 'A' – VAT Class A

- 'B' – VAT Class B

- 'C' – VAT Class C

- 'D' – VAT Class D

- 'E' – VAT Class E

- 'F' – Alte taxe

<i>BelongToDepNumber</i>	<i>BelongToDepNo</i> + 80h, 1 symbol for PLU department = 0x80 ... 0x93
<i>AlteTaxNum</i>	Up to 11 symbols for Alte Tax number
<i>AlteTaxValue</i>	Up to 11 symbols for Alte tax value
<i>Turnover</i>	Up to 11 symbols for PLU accumulated turnover
<i>QuantitySold</i>	Up to 11 symbols for Sales quantity of the article
<i>LastZReportNumber</i>	Up to 5 symbols for the number of the last article report with zeroing
<i>LastZReportDate</i>	16 symbols for the date and time of the last article report with zeroing
<i>AvailableQTY</i>	(Available Quantity) - 1..11 symbols for quantity in stock
<i>Barcode</i>	13 symbols for article barcode
<i>AlteTaxAmount</i>	Up to 11 symbols for Alte tax amount
<i>Category</i>	Up to 7 symbols for PLU Category code in format #####.##
<b>zfpdef:</b> <i>ReadPLUallData(PLUNum)</i>	

## 2.5.17. 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]><;>  
 <AlteTaxNum[1..11]><;> <AlteTaxValue[1..11]> <;><TurnoverAmount[1..10]> <;>  
 <SoldQuantity[1..11]><;> <LastZReportNumber[1..5]><;>  
 <LastZReportDate“DD-MM-YYYY HH:MM”><;><SingleTransaction[1]> <;>  
 <AlteTaxTurnover[1..11]>

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

### **Input data :**

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

### **Output data :**

<i>PLUNum</i>	5 symbols for article number with leading zeroes in format #####
<i>Option</i>	One symbol with value '1'
<i>PLUName</i>	34 symbols for article name /LF=7Ch, MU separator = 80h or 60h followed up to 3 symbols for unit/

<i>Price</i>	Up to 10 symbols for article price
<i>OptionPrice</i>	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

<i>OptionVATClass</i>	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
-----------------------	---

<i>BelongToDepNumber</i>	<i>BelongToDepNo</i> + 80h, 1 symbol for PLU department = 0x80 ... 0x93
<i>AlteTaxNum</i>	Up to 11 symbols for Alte Tax number
<i>AlteTaxValue</i>	Up to 11 symbols for Alte tax value
<i>TurnoverAmount</i>	Up to 11 symbols for PLU accumulated turnover
<i>SoldQuantity</i>	Up to 11 symbols for Sales quantity of the article
<i>LastZReportNumber</i>	5 symbols for the number of the last in format ##### article report with zeroing
<i>LastZReportDate</i>	16 symbols for the date and time of the last article report with zeroing
<i>SingleTransaction</i>	1 symbol with value: - '0' – Inactive, default value - '1' - Active Single transaction in receipt
<i>AlteTaxTurnover</i>	Up to 11 symbols for Alte Tax Turnover

**zfpdef:** *ReadPLUgeneral(PLUNum)*



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

**input:** <PLUNum[5]><;><Option['2']>

**output:** <PLUNum[5]><;><Option['2']><;><AvailableQuantity[1..11]><;>  
<OptionQuantityType[1]>

**FPR operation:** Provides information about the quantity 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 2

#### **Output data :**

PLUNum 5 symbols for article number with leading zeroes in format #####

Option One symbol with value 2

AvailableQuantity Up to 13 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)*

### 2.5.19. 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 No) 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)*

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

**input:** <PLUNum[5]><;><Option['4']>

**output:** <PLUNum[5]><;><Option['4']><;><Price[1..10]><;><OptionPrice[1]><;>  
<AlteTaxNum[1]><;> <AlteTaxValue[1..11]>

**FPR operation:** Provides information about the price and price type of the specified article.

#### **Input data :**

PLUNum (PLU No) 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 1..10 symbols for article price

OptionPrice 1 byte for Price flag with next value:

- '0'- Free price is disable valid only programmed price
- '1'- Free price is enable
- '2'- Limited price

AlteTaxNum 1 symbol for Alte Tax number

AlteTaxValue Up to 11 symbols for Alte tax value

**zfpdef:** *ReadPLUprice(PLUNum)*

### 2.5.21. Command: 6Bh / k – Read article registers, Option 5 (Category)

**input:** <PLUNum[5]><;><Option['5']>

**output:** <PLUNum[5]><;><Option['5']><;><Category[1..7]>

**FPR operation:** Provides information about the category of the specified article.

#### **Input data :**

*PLUNum* (PLU Number) 5 symbols for article number with leading zeroes in format: #####

*Option* One symbol with value 5

#### **Output data :**

*PLUNum* 5 symbols for article number with leading zeroes in format #####

*Option* One symbol with value 5

*Category* Up to 7 symbols for PLU Category code in format #####.##

**zfpdef:** *ReadPLUcategory(PLUNum)*

### 2.5.22. 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)*

### 2.5.23. Command: 52h / R – Read customer database

**input:** <Option['R']> <;><CustomerNum[3]>

**output:** <Option['R']><;><CustomerNum[3]> <;> <CustomerVatNum[15]> <;>

<CustomerName[30]> <;> <CustomerAddress[30]> <;><FreeLine1[20]> <;>

<FreeLine2[20]> <;> <FreeLine3[20]> <;> <FreeLine4[20]><;> <CustTurnover[1..11]>

**FPR operation:** Provide information for specified customer from FD database.

#### **Input data :**

*Option* 'R' – for reading customer data

*CustomerNum* 3 symbols for customer number in order in format ###

#### **Output data:**

*Option* 'R' – for reading customer data

*CustomerNum* 3 symbols for customer number in order in format ###

*CustomerVatNum* 15 symbols for customer VAT registration number

*CustomerName* 30 symbols for customer name

*CustomerAddress* 30 symbols for customer address

*FreeLine1* 20 ASCII symbols for customer data

*FreeLine2* 20 ASCII symbols for customer data

*FreeLine3* 20 ASCII symbols for customer data

*FreeLine4* 20 ASCII symbols for customer data

*CustTurnover* 1..11 symbols for accumulated turnover of the customer

**zfpdef:** *ReadCustomerData(CustomerNum)*



## 2.5.24. Command: 53h / S – Read service contract date, Option 1

**input:** <Option['L']> <;><'R'[1]><;><Password[6]> <;> <'1'[1]>

**output:** <Option['L']> <;> <'R'[1]> <;> <Password[6]> <;> <'1'[1]> <;>

<ExpiryDate "DD-MM-YYYY">

**FPR operation:** Read the the service contract expiry date.

### **Input data :**

*Option* 'L' – for alte taxe data  
'R' 'R' – for reading  
*Password* 6 symbols for service password  
'1' '1' – for date

### **Output data: n. a.**

*Option* 'L' – for alte taxe data  
'R' 'R' – for reading  
*Password* 6 symbols for service password  
'1' '1' – for date  
*ExpiryDate* 10 symbols for expiry date of service contract

**zfpdef:** [ReadServiceContractDate>Password](#)

## 2.5.25. Command: 53h / S – Read service contract warning messages, Option 3

**input:** <Option['L']> <;><'R'[1]><;><Password[6]> <;> <'3'[1]>

**output:** <Option['L']> <;> <'R'[1]> <;> <Password[6]> <;> <'3'[1]> <;>

<Line1[TextLength]> <;> <Line2[TextLength]> <;><Line3[TextLength]>

**FPR operation:** Read the programmed service contract warning message text.

### **Input data:**

*Option* 'L' – for alte taxe data  
'R' 'R' – for reading  
*Password* 6 symbols for service password  
'3' '3' – for warning message

### **Output data: n. a.**

*Option* 'L' – for alte taxe data  
'R' 'R' – for reading  
*Password* 6 symbols for service password  
'3' '3' – for warning message  
*Line1* TextLength symbols for warning message for line 1  
*Line2* TextLength symbols for warning message for line 2  
*Line3* TextLength symbols for warning message for line 3

**zfpdef:** [ReadServiceWarningMessages>Password](#)

### 2.5.26. Command: 53h / S – Read Other Charges (Alte tax) name, Option A

**input:** <Option['A']> <;><Option['R']> <;><Number[1]>

**output:** <Option['A']> <;><Option['R']> <;><Number[1]> <;> <Name[12]>

**FPR operation:** Read the other charges (Alte tax) name.

#### **Input data :**

Option 'A' – for alte tax data

Option 'R' – for reading

Number 1 symbol for number in order up to 7

#### **Output data: n. a.**

Option 'A' – for alte tax data

Option 'R' – for reading

Number 1 symbol for number in order

Name 12 symbols for Alte tax name

**zfpdef:** [ReadAlteTaxe\(Number\)](#)

### 2.5.27. Command: 58h / X –Reset Odometer function, Option Z

**Input:** <Option['O']><;><Option['Z']><;><Password[6]>

**output:** ACK

**FPR Operation:** Reset odometer function

*\*the command is valid for ADPOS model only.*

#### **Input data:**

Option 1 symbol 'O'

Option 1 symbol 'Z'

Password 6 symbols for access password

#### **Output data: n.a.**

**zfpdef:** [ResetOdometer\(Password\)](#)

### 2.5.28. Command: 58h / X – Read Odometer function, Option R

**Input:** <Option['O']><;><Option['R']>

**output:** <Option['O']><;> <OdomResult[1..19]>

**FPR Operation:** Read odometer result.

*\*the command is valid for ADPOS model only.*

#### **Input data:**

Option 1 symbol 'O'

Option 1 symbol 'R'

#### **Output data:**

Option 1 symbol 'O'

OdomResult 1..19 symbols for odometer result: used paper length in mm

**zfpdef:** [ReadOdometer\(\)](#)

## 2.5.29. Command: 4Fh / O – Read duplicate number of the invoice

**input:** <Option1['D']> <;><Option2['R']>

**output:** <Option1['D']> <;><Option2['R']> <;> <DuplicatesNumber[1]>

**FPR operation:** Read the the number of the duplicates which can be printed after invoice receipt.

### **Input data :**

Option1 1 symbol with value 'D'

Option2 1 symbol with value 'R'

### **Output data:**

Option1 1 symbol with value 'D'

Option2 1 symbol with value 'R'

DuplicatesNumber 1 symbol for number of duplicates which can be print. Possible values from '0' to '5'.

**zfpdef:** ReadDuplicateInvoiceNumber()

## 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 / . – Non-fiscal receipt opening

**input:** <OperNum[1..2]> <;> <OperPass[4]>  
<;><reserved['0']><;><NonFiscalPrintType[1]>

**output:** ACK

**FPR operation:** Opens a non-fiscal receipt assigned to the specified operator and print type depends on NonFiscalPrintType parameter.

**Input data :**

OperNum	(Operator Number) Symbols from '1' to '20' corresponding to operator's number
OperPass	(Operator Password) 4 symbols for operator's password
reserved	1 symbol with value '0'
NonFiscalPrintType	1 symbol with value: <ul style="list-style-type: none"><li>- '0' – Step by step printing</li><li>- '1' – Postponed printing</li></ul>

**Output data: n. a.**

**zfpdef:** [OpenNonFiscalReceipt\(Number, Password, NonFiscalPrintType\)](#)

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

**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 receipt

**input:** <OperNum[1..2]> <;> <OperPass[4]> <;><reserved['0']> <;>  
<reserved['0']><;><FiscalReceiptPrintType[1]>

**output:** ACK

**FPR operation:** Opens a fiscal receipt assigned to the specified operator and print type depends of FiscalReceiptPrintType parameter.

**Input data :**

OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 4 symbols for operator's password
reserved	1 symbol with value '0'
reserved	1 symbol with value '0'
FiscalReceiptPrintType	1 symbol with value: <ul style="list-style-type: none"><li>- '0' – Step by step printing</li><li>- '2' – Postponed printing</li><li>- '4' – Buffered Printing</li></ul>

**Output data: n. a.**

**zfpdef:** [OpenReceipt\(OperNum, OperPass, FiscalReceiptPrintType\)](#)

#### 2.6.4. Command: 30h / 0 – Open fiscal special customer document

**input:** <OperNum[1..2]> <;> <OperPass[4]><;> <reserved['0']> <;> <reserved['0']>  
<;> <CustomerReceiptPrintType[1]> <;> <CustomerVATNum[15]> <;> <InvRecipient[30]>  
<;> <InvFree1[20]> <;> <InvFree2[20]> <;> <InvFree3[20]> <;> <InvFree4[20]> <;>  
<Address[30]>

**output: ACK**

**FPR operation:** Opens a special Customer fiscal receipt document assigned to the specified operator, Customer data and print type depends on CustomerReceiptPrintType parameter.

**Input data :**

OperNum	(Operator No) Symbol from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 4 symbols for operator's password
Reserved	1 symbol with value '0'
Reserved	1 symbol with value '0'
CustomerReceiptPrintType	1 symbol with value: - '1' - Step by step printing - '3' – Postponed printing - '5' – Buffered printing
CustomerVATNum	15 ASCII symbols text for Customer VAT number
InvRecipient	30 ASCII symbols for Recipient name
InvFree1	20 ASCII symbols for free text line
InvFree2	20 ASCII symbols for free text line
InvFree3	20 ASCII symbols for free text line
InvFree4	20 ASCII symbols for free text line
Address	30 ASCII symbols for customer address

**Output data: n. a.**

**zfpdef:** *OpenSpecialDocumentReceipt(OperNum, OperPass, CustomerReceiptPrintType CustomerVATNo, InvRecipient, InvFree1, InvFree2, InvFree3, InvFree4, InvAddr)*

#### 2.6.5. 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]>}  
{<'@> <DiscNamed[1..8]>} {<'+'> <Category[1..7]>} {<'!'> <NamePLUextension[12]>}  
<;> <AdditionalNamePLU[108]> }

**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) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
OptionVATClass	1 symbol for article's VAT class with optional values: - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
Price	1 to 10 symbols for article's price

'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity) 1 to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %.) 1 to 7 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

**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, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)

### 2.6.6. 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]>}

{<'@> <DiscNamed[1..8]>} {<'+'> <Category[1..7]>} {<'&> <DepNum[1..2]>}

{<'!'> <NamePLUextension[12]>} {<';> <AdditionalNamePLU[108]>}

**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) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
Reserved	1 symbol with value "space"
Price	1 to 10 symbols for article's price
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity) 1 to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %.) 1 to 7 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'&'	1 symbol '&' indicating the presence of department
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
'!'	1 symbol with value '!'

*NamePLUextension* (PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only  
*AdditionalNamePLU* (Additional PLU name) 108 symbols for additional PLU name

**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, DepNum, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)

## 2.6.7. 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]>}  
 {<'@> <DiscNamed[1..8]>} {<'+'> <Category[1..7]>} {<'&> <DepNum[1..2]>}  
 {<'!'> <NamePLUextension[12]>} {<';> <AdditionalNamePLU[108]> }

**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 performed in its registers.

**Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
OptionVATClass	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
Price	1 to 10 symbols for article's price
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity) 1 to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) 1 to 7 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'&'	1 symbol '&' indicating the presence of departament
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

**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, DepNo, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)

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

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

**output:** ACK

**FPR operation:** Registers the sale or correction of a specified quantity of an article of the internal database of the FD.

### **Input data :**

<i>OptionSign</i>	(Sale/Correction)1 symbol with optional value: - '+' - Sale - '-' - Correction
<i>PLUNum</i>	(PLU Number) 5 symbols for number of article of FPR's db in format: #####
<i>'*'</i>	1 symbol '*' indicating the presence of quantity field
<i>Quantity</i>	(Quantity)1 to 10 symbols for article's quantity sold
<i>','</i>	1 symbol ',' indicating the presence of discount/addition field
<i>DiscAddP</i>	(Discount/Addition %) 1 to 7 for percentage of discount/addition
<i>':'</i>	1 symbol ':' indicating the presence of value discount/addition field
<i>DiscAddV</i>	(Discount/Addition Value)1 to 8 symbols for value of discount/addition
<i>'@'</i>	1 symbol '@' indicating the presence value of named discount
<i>DiscNamed</i>	(Named Discount) 1 to 8 symbols for value of named discount

### **Output data : n. a.**

**zfpdef:** SellPLUFromFD\_DB(OptionSign, PLUNum, Price, Quantity, DiscAddP, DiscAddV, DiscNamed)

## 2.6.9. Command: 33h / 3 – Subtotal

**input:** <OptionPrinting[1]> <;> <OptionDisplay[1]> {<':'> <DiscAddV[1..10]>} {<','> <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 :**

<i>OptionPrinting</i>	(Print)1 symbol with value: - '1' – Yes - '0' – No
<i>OptionDisplay</i>	(Display)1 symbol with value: - '1' – Yes - '0' – No
<i>':'</i>	1 symbol ':' indicating the presence of value discount/addition field
<i>DiscAddV</i>	(Discount/Addition Value)1 to 10 symbols for the value of the discount/addition
<i>','</i>	1 symbol ',' indicating the presence of percent discount/addition field
<i>DiscAddP</i>	(Discount/Addition %) 1 to 7 symbols for the percentage value of the discount/addition

### **Output data:**

*SubtotalValue* 1..10 symbols for the value of the subtotal amount

### **Notes:**

The discount/addition may be either values or percentages.



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*(*OptionPrint*, *OptionDisplay*, *DiscAddV*, *DiscAddP*)

## 2.6.10. Command: 33h / 3 – Subtotal with value discount from specified VAT definition

**input:** <*OptionPrinting*[1]><;><*OptionDisplay*[1]>{<':'><*DiscAddV*[1..10]>} {<;><*OptionVATClass*[1]>}

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

**FPR operation:** Calculates the subtotal amount for the specified VAT, with printing and display visualization options. Provides 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 :**

<i>OptionPrinting</i>	(Print)1 symbol with value: - '1' – Yes - '0' – No
<i>OptionDisplay</i>	(Display)1 symbol with value: - '1' – Yes - '0' – No
':'	1 symbol ':' indicating the presence of value discount/addition field
<i>DiscAddV</i>	(Discount/Addition Value)1 to 10 symbols for the value of the discount/addition
<i>OptionVATClass</i>	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe

### **Output data:**

*SubtotalValue* 1..10 symbols for the value of the subtotal amount

### **Notes:**

The discount/addition may be either values or percentages.

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 classes.

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

**zfpdef:** *SubtotalWithSpecifiedVAT*(*OptionPrinting*, *OptionDisplay*, *DiscAddV*, *OptionVATClass*)

## 2.6.11. 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]>} {<'&'> <OptionVATClass[1]>} {<'> <DiscAddP[1..7]>}  
{<'> <DiscAddV[1..8]>} {<'> <Category[1..7]>} {<'@> <DiscNamed[1..8]>} {<'!>  
<NamePLUextension[12]>} {<;> <AdditionalNamePLU[108]> }

**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 performed in its registers.

### Input data :

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
Price	1 to 10 symbols for article's price
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	1 to 10 symbols for quantity
'&'	1 symbol '&' indicating the VAT class field
OptionVATClass	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
''	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition Percentage) 1 to 7 symbols for percentage of discount/addition
':'	1 symbol ':' indicating the of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	1..7 symbols for PLU Category code in format #####.##
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 2 to 8 symbols for value of named discount
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

### 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:** SellPLUfromDep\_(NamePLU, DepNum, Price, Quantity, OptionVATClass, DiscAddP, DiscAddV, Category, DiscNamed, NamePLUextension, AdditionalNamePLU)

## 2.6.12. 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]>} {<'+'><Category[1..7]>}  
{<'@><DiscNamed[1..8]>} {<'!'> <NamePLUextension[12]>} {<;>  
<AdditionalNamePLU[108]> }

**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) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
Price	1 to 10 symbols for article's price
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	1 to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition Percentage) 1 to 7 symbols for percentage of discount/addition
':'	1 symbol ':' indicating the of value discount/addition field
DiscAddV	(Discount/Addition Value) 1..8 for value of discount/addition
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	1..7 symbols for PLU Category code in format #####.##
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

### Output data : n. a.

### Notes:

#### 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:** SellPLUwithSpecifiedVATfromDep\_(NamePLU, DepNum, Price, Quantity, DiscAddP, DiscAddV, Category, DiscNamed, NamePLUextension, AdditionalNamePLU)

### 2.6.13. Command: 35h / 5 – Payment

**input:** <PaymentType[1]> <;> <reserved['0']> <;> <Amount[1..10]> <;>  
<reserved['1']>

**output: ACK**

**FPR operation:** Registers the payment in the receipt with specified type of payment and amount received (if the payment type is 1-9 the amount of change due is not obligatory.)

**Input data :**

PaymentType	(Payment Type Options)1 symbol for payment type: <ul style="list-style-type: none"><li>- '0' – Payment 0</li><li>- '1' – Payment 1</li><li>- '2' – Payment 2</li><li>- '3' – Payment 3</li><li>- '4' – Payment 4</li><li>- '5' – Payment 5</li><li>- '6' – Payment 6</li><li>- '7' – Payment 7</li><li>- '8' – Payment 8</li><li>- '9' – Payment 9</li></ul>
reserved	1 symbol with value '0'
Amount	1 to 10 characters for received amount
reserved	1 symbols with value '1'

**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, Amount)*

### 2.6.14. Command: 35h / 5 – Pay exact sum

**input:** <PaymentType[1]> <;> <reserved['0']> <;> <Amount[""]> <;><reserved['1']>

**output:** ACK

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

**Input data :**

PaymentType	(Payment Type)1 symbol for payment type: <ul style="list-style-type: none"><li>- '0' – Payment 0</li><li>- '1' – Payment 1</li><li>- '2' – Payment 2</li><li>- '3' – Payment 3</li><li>- '4' – Payment 4</li><li>- '5' – Payment 5</li><li>- '6' – Payment 6</li><li>- '7' – Payment 7</li><li>- '8' – Payment 8</li><li>- '9' – Payment 9</li></ul>
reserved	1 symbol with value '0'
Amount	1 symbol ' ' - quotation mark, for pay with exact sum
reserved	1 symbol with value '1'

**Output data:** n.a.

**zfpdef:** *PayExactSum(PaymentType)*

### 2.6.15. 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()*

### 2.6.16. Command: 37h / 7 – Free text printing

**input:** <Text[TextLength]>

**output:** ACK

**FPR operation:** Prints a free text.

**Input data :**

Text Free text - TextLength symbols

**Output data:** ACK

**zfpdef:** *PrintText(Text)*

### 2.6.17. Command: 38h / 8 – Fiscal receipt closure

input: n. a.

output: ACK

FPR operation: Closes the opened fiscal receipt.

**Input data : n. a.**

**Output data : n. a.**

**zfpdef:** *CloseReceipt()*

### 2.6.18. Command: 39h / 9 – Fiscal receipt cancel

input: n. a.

output: ACK

FPR operation: Cancel the opened fiscal receipt.

**Input data : n. a.**

**Output data : n. a.**

**zfpdef:** *CancelReceipt()*

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

input: n. a.

output: ACK

FPR operation: Print a copy of the last receipt document issued

**Input data : n. a.**

**Output data : n. a.**

**zfpdef:** *PrintLastReceiptDuplicate()*

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

input: <OperNum[1..2]> <;> <OperPass[4]> <;> <reserved['0']> <;>  
<Amount[1..10]>{<'@'> <Text[TextLength-2]>}

output: ACK

FPR operation: Registers cash received on account or paid out.

**Input data :**

OperNum	(Operator No) Symbol from 1 to 20 corresponding to the operator's number
OperPass	(Operator Password) 4 symbols for operator's password
reserved	1 symbol with value 0
Amount	1 to 10 symbols for the amount lodged/withdrawn
'@'	Symbol @
Text	Text - TextLength-2 symbols length

**Output data: n. a.**

**zfpdef:** *ReceivedOnAccount\_PaidOut(OperNum, OperPass, OptionPayment, Amount, Text)*

## 2.6.21. Command: 3Ch / < – Storno function of article with VAT class definition

**input:** <NamePLU[36]> <;> <OptionVATClass[1]> <;> <Price[1..10]>  
 {<'\*> <Quantity[1..10]>}{<','> <DiscAddP[1..7]>} {<':'><DiscAddV[1..8]>}  
 {<'@><DiscNamed[1..8]>} {<'+><Category[1..7]>} {<'!><NamePLUextension[12]>}  
 {<';><AdditionalNamePLU[108]> }

**output: ACK**

**FPR operation:** Registers the correction article with specified name, price, quantity, VAT class and discount/addition on the transaction.

### **Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
OptionVATClass	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
Price '*'	1 to 10 symbols for article's price with minus sign for storno operation 1 symbol '*' indicating the presence of quantity field
Quantity ' '	(Quantity) 1 to 10 symbols for quantity 1 symbol ' ' indicating the presence of discount/addition field
DiscAddP '.'	(Discount/Addition %) 1 to 7 for percentage of discount/addition 1 symbol '.' indicating the presence of value discount/addition field
DiscAddV '@'	(Discount/Addition Value) 1 to 8 for value of discount/addition 1 symbol '@' indicating the presence value of named discount
DiscNamed '+'	(Named Discount) 1 to 8 symbols for value of named discount 1 symbol '+' indicating the presence of value PLU Category code
Category '!'	Up to 7 symbols for PLU Category code in format #####.## 1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

**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 class 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:** StornoPLU(NamePLU, OptionVATClass, Price, Quantity, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)



## 2.6.22. Command: 3Ch / < – Storno function of article belonging to departament

**input:** <NamePLU[36]> <;> <Reserved[' ']> <;> <Price[1..10]>  
{<'\*> <Quantity[1..10]>} {<'> <DiscAddP[1..7]>} {<'> <DiscAddV[1..8]>}  
{<'@> <DiscNamed[1..8]>} {<'> <Category[1..7]>} {<'&> <DepNum[1..2]>}  
{<'!> <NamePLUextension[12]>} {<;> <AdditionalNamePLU[108]> }

**output: ACK**

**FPR operation:** Registers the correction of article with specified name, price, quantity, VAT class and/or discount/addition on the transaction.

### **Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
Reserved	1 symbol with value "space"
Price	1 to 10 symbols for article's price with minus sign for storno operation
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity) 1 to 10 symbols for quantity
' '	1 symbol ' ' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) 1 to 7 for percentage of discount/addition
'.'	1 symbol '.' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'&'	1 symbol '&' indicating the presence of departament
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

**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 class 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:** StornoPLUfromDep(NamePLU, Price, Quantity, DepNum, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)



## 2.6.23. Command: 3Ch / < – Storno function of article with specified VAT belonging to departament

**input:** <NamePLU[36]> <;> <OptionVATClass[1]> <;> <Price[1..10]> {<'\*> <Quantity[1..10]>}{<','> <DiscAddP[1..7]>}{<':'><DiscAddV[1..8]>}{<'@><DiscNamed[1..8]>}{<'+><Category[1..7]>}{<'&><DepNum[1..2]>}{<'!'><NamePLUextension[12]>}{<';><AdditionalNamePLU[108]> }

**output: ACK**

**FPR operation:** Registers the correction of article with specified name, price, quantity, VAT class and/or discount/addition on the transaction.

### **Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
OptionVATClass	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
Price	1 to 10 symbols for article's price with minus sign for storno operation
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity) 1 to 10 symbols for quantity
','	1 symbol ',' indicating the presence of discount/addition field
DiscAddP	(Discount/Addition %) 2 to 7 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 2 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 2 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'&'	1 symbol '&' indicating the presence of departament
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

### **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 class 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:** [StornoPLUwithSpecifiedVATfromDep\(NamePLU, OptionVATClass, Price, Quantity, DepNum, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU\)](#)

## 2.6.24. 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]>} {<'@> <DiscNamed[1..8]>} {<'+'> <Category[1..7]>} {<'!'> <NamePLUextension[12]>} {<';> <AdditionalNamePLU[108]> }

**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 performed in its registers.

### **Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
OptionVATClass	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte taxe
Price	1 to 10 symbols for article's price
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity)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 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

### **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, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)

## 2.6.25. Command: 3Dh / = – Sell / Correction of article with fractional quantity belonging to departament

**input:** <NamePLU[36]> <;> <Reserved[' ']> <;> <Price[1..10]>  
 {<'\*> <Quantity[10]>} {<','> <DiscAddP[1..7]>} {<':'> <DiscAddV[1..8]>}  
 {<'@> <DiscNamed[1..8]>} {<'+'> <Category[1..7]>} {<'&> <DepNum[1..2]>}  
 {<'!'> <NamePLUextension[12]>} {<;> <AdditionalNamePLU[108]> }

**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 performed in its registers.

### **Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
Reserved	1 symbol with value "space"
Price	1 to 10 symbols for article's price
'*'	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity) 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 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'&'	1 symbol '&' indicating the presence of department
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

**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, DepNum, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)

## 2.6.26. 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]>}  
 {<'@> <DiscNamed[1..8]>} {<'> <Category[1..7]>} {<'> <DepNum[1..2]>}  
 {<'> <NamePLUextension[12]>} {<'> <AdditionalNamePLU[108]> }

**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 performed in its registers.

### **Input data :**

NamePLU	(PLU Name) 30 symbols for article's name plus separator for MU=60h followed up to 3 symbols for unit plus 2 symbols spaces
OptionVATClass	1 symbol for article's VAT class with optional values:" - 'A' – VAT Class A - 'B' – VAT Class B - 'C' – VAT Class C - 'D' – VAT Class D - 'E' – VAT Class E - 'F' – Alte tax
Price	1 to 10 symbols for article's price
'*	1 symbol '*' indicating the presence of quantity field
Quantity	(Quantity)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 for percentage of discount/addition
':'	1 symbol ':' indicating the presence of value discount/addition field
DiscAddV	(Discount/Addition Value) 1 to 8 for value of discount/addition
'@'	1 symbol '@' indicating the presence value of named discount
DiscNamed	(Named Discount) 1 to 8 symbols for value of named discount
'+'	1 symbol '+' indicating the presence of value PLU Category code
Category	Up to 7 symbols for PLU Category code in format #####.##
'&'	1 symbol '&' indicating the presence of departament
DepNum	DepNum + 80h (Department No) 1 symbol for article department attachment, formed in the following manner; <i>example: Dep01 = 81h, Dep02 = 82h ... Dep19 = 93h</i>
'!'	1 symbol with value '!'
NamePLUextension	(PLU Name Extension) 12 symbols for extension of the PLU Name: FP Only
AdditionalNamePLU	(Additional PLU name) 108 symbols for additional PLU name

### **Output data : n. a.**

#### **Notes:**

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

The quantity fields are not obligatory. If no value is stated for them the FPR 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, DepNum, DiscAddP, DiscAddV, DiscNamed, Category, NamePLUextension, AdditionalNamePLU)

## 2.6.27. Command: 3Eh / > – Discount/ addition

**input:** <OptionType[1]><;><OptionDisplay[1]>{<'><DiscAddV[1..10]>}  
{<'><DiscAddP[1..7]>}

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

**FPR operation:** Percent or value discount/addition over sum of transaction or over subtotal sum depends on parameter “OptionType”.

### **Input data :**

*OptionType* 1 symbol with value  
- '2' - Defined from the device  
- '1' - Over subtotal  
- '0' - Over transaction sum  
*OptionDisplay* (Display) 1 symbol with value:  
- '1' – Yes  
- '0' – No  
'.' 1 symbol '.' indicating the presence of value discount/addition field  
*DiscAddV* (Discount/Addition Value) 1 to 10 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 %) 1 to 7 symbols for the percentage value of the discount/addition. Use minus sign '-' for discount

### **Output data :**

*DiscAddValue* 1 to 10 symbols for the value of the applied discount/addition.

**zfpdef:** [PrintDiscountOrAddition\(OptionType, OptionDisplay, DiscAddV, DiscAddP\)](#)

## 2.7. SALES OPERATION COMMANDS FOR CURRENCY EXCHANGE OFFICES

These commands are used mainly for FD sale/buy currency.

### 2.7.1. Command: 30h / 0 – Open receipt for CURRENCY SALE

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

<CurrencySaleRcpPrintType[1]> <;> <Text1[26]> <;> <Text2[26]> <;><Text3[26]> <;>  
<Text4[26]> <;> <Text5[26]> <;> <Text6[26]>

**output:** ACK

**FPR operation:** Opens a fiscal receipt assigned to the specified operator for Currency Sale transaction.

#### **Input data :**

OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 4 symbols for operator's password
'1'	1 symbol with value '1'
'1'	1 symbol with value '1'
CurrencySaleRcpPrintType	1 symbol with value - '0' – Step by step printing - '2' – Postponed printing
Text1	26 symbols free text for header line 1 in the receipt
Text2	26 symbols free text for header line 2 in the receipt
Text3	26 symbols free text for header line 3 in the receipt
Text4	26 symbols free text for header line 4 in the receipt
Text5	26 symbols free text for header line 5 in the receipt
Text6	26 symbols free text for header line 6 in the receipt

**Output data: n. a.**

**zfpdef:** *OpenCurrencySaleReceipt(OperNum, OperPass, CurrencySaleRcpPrintType, Text1, Text2, Text3, Text4, Text5, Text5, Text6)*

### 2.7.2. Command: 30h / 0 – Open receipt for CURRENCY BUYING

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

<CurrencyBuyingRcpPrintType[1]> <;> <Text1[26]> <;> <Text2[26]> <;> <Text3[26]> <;>  
<Text4[26]> <;> <Text5[26]> <;> <Text6[26]>

**output:** ACK

**FPR operation:** Opens a fiscal receipt assigned to the specified operator for Currency Purchase transaction.

#### **Input data :**

OperNum	(Operator Number) Symbols from 1 to 20 corresponding to operator's number
OperPass	(Operator Password) 4 symbols for operator's password
'1'	1 symbol with value '1'
'1'	1 symbol with value '1'
CurrencyBuyingRcpPrintType	1 symbol with value - '8' – Step by step printing - '.' – Postponed printing
Text1	26 symbols free text for header line 1 in the receipt
Text2	26 symbols free text for header line 2 in the receipt
Text3	26 symbols free text for header line 3 in the receipt
Text4	26 symbols free text for header line 4 in the receipt
Text5	26 symbols free text for header line 5 in the receipt
Text6	26 symbols free text for header line 6 in the receipt

**Output data: n. a.**

**zfpdef:** *OpenCurrencyBuyingReceipt(OperNum, OperPass, CurrencyBuyingRcpPrintType, Text1, Text2, Text3, Text4, Text5, Text5, Text6)*

### 2.7.3. Command: 31h / 1 – SELL/PURCHASE OF CURRENCY

**input:** <CurrencyName[3]><;> <Rate[1..11]> { <'?'> <PerCurrencyUnit[1..6]> <'\*'>  
<Quantity[1..10]> <','><Commission[1..5]> <'\$'> <OptionPaymentCurrType['0']>}

**output: ACK**

**FPR operation:** Registers the sale/buying of currency with specified name, rate of exchange, quantity and percentage of commission and closes the receipt. The type of transaction – **sale or purchase transaction depends on the receipt opening type**. This command closes the receipt with payment cash or other type, defined in PaymentType field if present.

The Quantity, Commission and PaymentType fields are not obligatory.

**Input data :**

<i>CurrencyName</i>	3 symbols for currency name
<i>Rate</i>	Up to 11 symbols "xxxxx.xxxxx" for rate of exchange
<i>'?'</i>	1 symbol '?' indicating the presence of Per field
<i>PerCurrencyUnit</i>	Up to 6 digits for rate "per" factor
<i>'*'</i>	1 symbol '*' indicating the presence of quantity field
<i>Quantity</i>	1 to 8 symbols "xxxxxxx" for amount of currency
<i>','</i>	1 symbol ',' indicating the presence of commission
<i>Commission</i>	1 to 5 symbols for percentage of commission
<i>'\$'</i>	1 symbol '\$' indicating the presence of Payment type field
<i>OptionPaymentCurrType</i>	1 symbol with value '0'

**Output data : n. a.**

**zfpdef:** CurrencyTransaction(CurrencyName, Rate, PerCurrencyUnit, Quantity, Comission, OptionPaymentCurrType)



## 2.8. 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.

### 2.8.1. Command: 6Dh / m – Read amounts by VAT Class

**input:** n. a.

**output:** <SaleAmountVATGrA[1..11]> <;> <SaleAmountVATGrB[1..11]> <;>  
<SaleAmountVATGrC[1..11]> <;> <SaleAmountVATGrD[1..11]>  
<;> <SaleAmountVATGrE[1..11]> <;> <SaleAmountAlteTaxeF[1..11]>

**FPR operation:** Provides information about the accumulated amount by VAT class.

**Input data : n. a.**

**Output data :**

SaleAmountVATGrA	Up to 11 symbols for the amount accumulated in the VAT group A
SaleAmountVATGrB	Up to 11 symbols for the amount accumulated in the VAT group B
SaleAmountVATGrC	Up to 11 symbols for the amount accumulated in the VAT group C
SaleAmountVATGrD	Up to 11 symbols for the amount accumulated in the VAT group D
SaleAmountVATGrE	Up to 11 symbols for the amount accumulated in the VAT group E
SaleAmountAlteTaxeF	Up to 11 symbols for the amount accumulated in the Alte Taxe F

**zfpdef:** [ReadDailyAmountsByVAT\(\)](#)

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

**input:** <'0'>

**output:** <'0'> <;> <AmountPayment0[1..11]> <;> <AmountPayment1[1..11]> <;>  
<AmountPayment2[1..11]> <;> <AmountPayment3[1..11]> <;> <AmountPayment4[1..11]>  
<;> <AmountPayment5[1..11]> <;> <AmountPayment6[1..11]> <;>  
<AmountPayment7[1..11]> <;> <AmountPayment8[1..11]> <;> <AmountPayment9[1..11]>

**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 11 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 11 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 11 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 11 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 11 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 11 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 11 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 11 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 11 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 11 symbols for the accumulated amount by payment type 9

**zfpdef:** [ReadDailyAvailableAmounts\(\)](#)



### 2.8.3. Command: 6Eh / n – Read registers, Option ‘1’ (general)

**input:** <'1'>

**output:** <'1'> <;> <CustomersNum[1..5]> <;> <DiscountsNum[1..5]> <;>  
<DiscountsAmount[1..11]> <;> <AdditionsNum[1..5]> <;> <AdditionsAmount[1..11]> <;>  
<CorrectionsNum[1..5]> <;> <CorrectionsAmount[1..11]>

**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 11 symbols for accumulated amount of discounts

AdditionsNum Up to 5 symbols for number of additions

AdditionsAmount Up to 11 symbols for accumulated amount of additions

CorrectionsNum Up to 5 symbols for number of corrections

CorrectionsAmount Up to 11 symbols for accumulated amount of corrections

[\*zfpdef: ReadGeneralDailyRegisters\(\)\*](#)

### 2.8.4. Command: 6Eh / n – Read registers, Option ‘2’ (RA)

**input:** <'2'>

**output:** <'2'> <;> <AmountPayment0[1..11]> <;> <AmountPayment1[1..11]> <;>  
<AmountPayment2[1..11]> <;> <AmountPayment3[1..11]> <;> <AmountPayment4[1..11]>  
<;> <AmountPayment5[1..11]> <;> <AmountPayment6[1..11]> <;>  
<AmountPayment7[1..11]> <;> <AmountPayment8[1..11]> <;> <AmountPayment9[1..11]>  
<;> <NumRA[1..5]> <;>

**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 11 symbols for the accumulated amount by payment type 0

AmountPayment1 Up to 11 symbols for the accumulated amount by payment type 1

AmountPayment2 Up to 11 symbols for the accumulated amount by payment type 2

AmountPayment3 Up to 11 symbols for the accumulated amount by payment type 3

AmountPayment4 Up to 11 symbols for the accumulated amount by payment type 4

AmountPayment5 Up to 11 symbols for the accumulated amount by payment type 5

AmountPayment6 Up to 11 symbols for the accumulated amount by payment type 6

AmountPayment7 Up to 11 symbols for the accumulated amount by payment type 7

AmountPayment8 Up to 11 symbols for the accumulated amount by payment type 8

AmountPayment9 Up to 11 symbols for the accumulated amount by payment type 9

NumRA Up to 5 symbols for the total number of operations

[\*zfpdef: ReadDailyRA\(\)\*](#)

## 2.8.5. Command: 6Eh / n – Read registers, Option ‘3’ (PO)

**input:** <'3'>

**output:** <'3'> <;> <AmountPayment0[1..11]> <;> <AmountPayment1[1..11]> <;>  
<AmountPayment2[1..11]> <;> <AmountPayment3[1..11]> <;> <AmountPayment4[1..11]>  
<;> <AmountPayment5[1..11]> <;> <AmountPayment6[1..11]> <;>  
<AmountPayment7[1..11]> <;> <AmountPayment8[1..11]> <;> <AmountPayment9[1..11]>  
<;> <NumPO[1..5]> <;>

**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 11 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 11 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 11 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 11 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 11 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 11 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 11 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 11 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 11 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 11 symbols for the accumulated amount by payment type 9
NumPO	Up to 5 symbols for the total number of operations

**zfpdef:** [ReadDailyPO\(\)](#)

## 2.8.6. Command: 6Eh / n – Read registers, Option ‘4’ (received)

**input:** <'4'>

**output:** <'4'> <;> <AmountPayment0[1..11]> <;> <AmountPayment1[1..11]> <;>  
<AmountPayment2[1..11]> <;> <AmountPayment3[1..11]> <;> <AmountPayment4[1..11]>  
<;> <AmountPayment5[1..11]> <;> <AmountPayment6[1..11]> <;>  
<AmountPayment7[1..11]> <;> <AmountPayment8[1..11]> <;> <AmountPayment9[1..11]>

**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 11 symbols for the accumulated amount by payment type 0
AmountPayment1	Up to 11 symbols for the accumulated amount by payment type 1
AmountPayment2	Up to 11 symbols for the accumulated amount by payment type 2
AmountPayment3	Up to 11 symbols for the accumulated amount by payment type 3
AmountPayment4	Up to 11 symbols for the accumulated amount by payment type 4
AmountPayment5	Up to 11 symbols for the accumulated amount by payment type 5
AmountPayment6	Up to 11 symbols for the accumulated amount by payment type 6
AmountPayment7	Up to 11 symbols for the accumulated amount by payment type 7
AmountPayment8	Up to 11 symbols for the accumulated amount by payment type 8
AmountPayment9	Up to 11 symbols for the accumulated amount by payment type 9

**zfpdef:** [ReadDailyReceivedSalesAmounts\(\)](#)

### 2.8.7. Command: 6Eh / n – Read registers, Option ‘5’ (counters)

**input:** <'5'>

**output:** <'5'> <;> <LastReportNumFromReset[1..5]> <;> <NumLastFMBlock[1..5]> <;> <NumEJ[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

NumLastFMBlock Up to 5 symbols for number of the last FM report

NumEJ 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()

### 2.8.8. Command: 6Eh / n – Read daily registeras, Option '7' (Currency Exchange Offices)

**input:** <'7'>

**output:** <'7'> <;> <Vanzari[1..12]> <;> <Cumparari[1..12]> <;> <Commission\_vanzari[1..12]> <;> <Commission\_cumparari[1..12]>

**FPR operation:** Provides information about the accumulated amounts from sale of foreign currency, currency purchase and commissions.

#### **Input data :**

'7' 1 symbol obligatory '7'

#### **Output data:**

'7' 1 symbol obligatory '7'

Vanzari Up to 12 symbols for accumulated amount of sales of currency without commission

Cumparari Up to 12 symbols for accumulated amount of currency purchase without commission

Commission\_vanzari Up to 12 symbols for accumulated amount of commissions in sales

Commission\_cumparari Up to 12 symbols for accumulated amount of commissions in purchase

**zfpdef:** ReadDailyCurrency()

### 2.8.9. 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..11]> <;> <AdditionsNum[1..5]> <;> <AdditionsAmount[1..11]> <;> <CorrectionsNum[1..5]> <;> <CorrectionsAmount[1..11]>

**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 11 symbols for accumulated amount of discounts
AdditionsNum	Up to 5 symbols for number of additions
AdditionsAmount	Up to 11 symbols for accumulated amount of additions
CorrectionsNum	Up to 5 symbols for number of corrections
CorrectionsAmount	Up to 11 symbols for accumulated amount of corrections
<b>zfpdef:</b> <a href="#">ReadDailyGeneralRegistersByOperator(OperNum)</a>	

**2.7.10. Command: 6Fh / o – Read operator's report, Option '2' (RA)****input:** <'2'> <;> <OperNum[1..2]>**output:** <'2'> <;> <OperNum[1..2]> <;> <AmountRA\_Payment0[1..11]> <;>

&lt;AmountRA\_Payment1[1..11]&gt; &lt;;&gt; &lt;AmountRA\_Payment2[1..11]&gt; &lt;;&gt;

&lt;AmountRA\_Payment3[1..11]&gt; &lt;;&gt; &lt;AmountRA\_Payment4[1..11]&gt; &lt;;&gt;

&lt;AmountRA\_Payment5[1..11]&gt; &lt;;&gt; &lt;AmountRA\_Payment6[1..11]&gt; &lt;;&gt;

&lt;AmountRA\_Payment7[1..11]&gt; &lt;;&gt; &lt;AmountRA\_Payment8[1..11]&gt; &lt;;&gt;

&lt;AmountRA\_Payment9[1..11]&gt; &lt;;&gt; &lt;NumRA[5]&gt;

**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 20 corresponding to operator's number

**Output data:**

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

**zfpdef:** [ReadDailyRAbyOperator\(OperNum\)](#)**2.7.11. Command: 6Fh / o – Read operator's report, Option '3' (PO)****input:** <'3'> <;> <OperNum[1..2]>**output:** <'3'> <;> <OperNum[1..2]> <;> <AmountPO\_Payment0[1..11]> <;>

&lt;AmountPO\_Payment1[1..11]&gt; &lt;;&gt; &lt;AmountPO\_Payment2[1..11]&gt; &lt;;&gt;

&lt;AmountPO\_Payment3[1..11]&gt; &lt;;&gt; &lt;AmountPO\_Payment4[1..11]&gt; &lt;;&gt;

&lt;AmountPO\_Payment5[1..11]&gt; &lt;;&gt; &lt;AmountPO\_Payment6[1..11]&gt; &lt;;&gt;

&lt;AmountPO\_Payment7[1..11]&gt; &lt;;&gt; &lt;AmountPO\_Payment8[1..11]&gt; &lt;;&gt;

&lt;AmountPO\_Payment9[1..11]&gt; &lt;;&gt; &lt;NumPO[1..5]&gt;

**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'
Communication Protocol	

*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 20 corresponding to operator's number  
*AmountPO\_Payment0* Up to 11 symbols for the PO by type of payment 0  
*AmountPO\_Payment1* Up to 11 symbols for the PO by type of payment 1  
*AmountPO\_Payment2* Up to 11 symbols for the PO by type of payment 2  
*AmountPO\_Payment3* Up to 11 symbols for the PO by type of payment 3  
*AmountPO\_Payment4* Up to 11 symbols for the PO by type of payment 4  
*AmountPO\_Payment5* Up to 11 symbols for the PO by type of payment 5  
*AmountPO\_Payment6* Up to 11 symbols for the PO by type of payment 6  
*AmountPO\_Payment7* Up to 11 symbols for the PO by type of payment 7  
*AmountPO\_Payment8* Up to 11 symbols for the PO by type of payment 8  
*AmountPO\_Payment9* Up to 11 symbols for the PO by type of payment 9  
*NumPO* 5 symbols for the total number of operations

**zfpdef:** *ReadDailyPObyOperator(OperNum)*

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

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

**output:** <'4'> <;> <OperNum[1..2]> <;> <ReceivedSalesAmountPayment0[1..11]> <;>  
<ReceivedSalesAmountPayment1[1..11]> <;> <ReceivedSalesAmountPayment2[1..11]> <;>  
<ReceivedSalesAmountPayment3[1..11]> <;> <ReceivedSalesAmountPayment4[1..11]> <;>  
<ReceivedSalesAmountPayment5[1..11]> <;> <ReceivedSalesAmountPayment6[1..11]> <;>  
<ReceivedSalesAmountPayment7[1..11]> <;> <ReceivedSalesAmountPayment8[1..11]> <;>  
<ReceivedSalesAmountPayment9[1..11]>

**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 11 symbols for amounts received by sales for payment 0  
*ReceivedSalesAmountPayment1* Up to 11 symbols for amounts received by sales for payment 1  
*ReceivedSalesAmountPayment2* Up to 11 symbols for amounts received by sales for payment 2  
*ReceivedSalesAmountPayment3* Up to 11 symbols for amounts received by sales for payment 3  
*ReceivedSalesAmountPayment4* Up to 11 symbols for amounts received by sales for payment 4  
*ReceivedSalesAmountPayment5* Up to 11 symbols for amounts received by sales for payment 5  
*ReceivedSalesAmountPayment6* Up to 11 symbols for amounts received by sales for payment 6  
*ReceivedSalesAmountPayment7* Up to 11 symbols for amounts received by sales for payment 7  
*ReceivedSalesAmountPayment8* Up to 11 symbols for amounts received by sales for payment 8  
*ReceivedSalesAmountPayment9* Up to 11 symbols for amounts received by sales for payment 9

**zfpdef:** *ReadDailyReceivedSalesAmountsByOperator(OperNum)*



## 2.7.16. 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\(OperNum\)](#)

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

**input:** <'6'> <;> <OperNum[1..2]>

**output:** <'6'> <;> <OperNum[1..2]> <;> <ChangeAmountPayment0[1..11]> <;>

<ChangeAmountPayment1[1..11]> <;> <ChangeAmountPayment2[1..11]> <;>

<ChangeAmountPayment3[1..11]> <;> <ChangeAmountPayment4[1..11]> <;>

<ChangeAmountPayment5[1..11]> <;> <ChangeAmountPayment6[1..11]> <;>

<ChangeAmountPayment7[1..11]> <;> <ChangeAmountPayment8[1..11]> <;>

<ChangeAmountPayment9[1..11]>

**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 11 symbols for amounts received by type of payment 0
ChangeAmountPayment1	Up to 11 symbols for amounts received by type of payment 1
ChangeAmountPayment2	Up to 11 symbols for amounts received by type of payment 2
ChangeAmountPayment3	Up to 11 symbols for amounts received by type of payment 3
ChangeAmountPayment4	Up to 11 symbols for amounts received by type of payment 4
ChangeAmountPayment5	Up to 11 symbols for amounts received by type of payment 5
ChangeAmountPayment6	Up to 11 symbols for amounts received by type of payment 6
ChangeAmountPayment7	Up to 11 symbols for amounts received by type of payment 7
ChangeAmountPayment8	Up to 11 symbols for amounts received by type of payment 8
ChangeAmountPayment9	Up to 11 symbols for amounts received by type of payment 9

**zfpdef:** [ReadDailyReturnedChangeAmountsByOperator\(OperNum\)](#)

## 2.8.15. Command: 6Fh / o – Read operators registers, Option '7' (Currency Exchange Offices)

**input:** <'7'><;> <OperNum[2]>

**output:** <'7'> <;> <OperNum[2]><;> <Vanzari\_op[1..12]> <;> <Cumparari\_op[1..12]> <;> <Commission\_vanzari\_op[1..12]> <;> <Commission\_cumparari\_op[1..12]>

**FPR operation:** Provides information about the accumulated amounts from sale of foreign currency, currency purchase and commissions for operator.

### **Input data :**

'7'	1 symbol obligatory '7'
OperNum	(Operator Number) Symbol from '1' to '20' corresponding to operator's number

### **Output data:**

'7'	1 symbol obligatory '1' to '20'
OperNum	Symbol from 1 to 20 corresponding to operator's number
Vanzari_op	Up to 12 symbols for accumulated amount of sales of currency without commission
Cumparari_op	Up to 12 symbols for accumulated amount of currency purchase without commission
Commission_vanzari_op	Up to 12 symbols for accumulated amount of commissions in sales
Commission_cumparari_op	Up to 12 symbols for accumulated amount of commissions in purchase

**zfpdef:** ReadCurrencyAmountsByOperator(OperNum)

## 2.8.16. Command: 71h / q – Read last and total receipt numbers

**input:** n. a.

**output:** <LastReceiptNum[4]><;><TotalReceiptCounter[7]>

**FPR operation:** Provides information about the number of the last issued receipt.

### **Input data : n. a.**

### **Output data :**

LastReceiptNum	4 symbols for the number of last issued fiscal receipt in format ####
TotalReceiptCounter	7 symbols for the number of totals issued fiscal receipts in format #####

**zfpdef:** ReadLastAndTotalReceiptNum()

## 2.8.17. Command: 71h / q – Read receipt numbers (Currency Exchange Offices)

**input:** n. a.

**output:** <LastReceiptNum[4]><;>< TotalReceiptCounter[7]> <;>

<Daily\_Vanzari\_RecCounter[4]> <;> <Daily\_Cumparari\_RecCounter[4]> <;>  
<Daily\_Anulat\_RecCounter[4]>

**FPR operation:** Provides information about the fiscal receipt counters for Currency Exchange Offices

**Input data : n. a.**

### **Output data :**

LastReceiptNum	4 symbols in format ####.
	For the number of last issued fiscal receipts
TotalReceiptCounter	7 symbols in format #####.

*For the number of totals issued fiscal receipts*  
*Daily\_Vanzari\_RecCounter 4 symbols in format ####.*  
*For the number of sale of currency receipts*  
*Daily\_Cumparari\_RecCounter 4 symbols in format ####.*  
*For the number of currency purchase receipts*  
*Daily\_Anulat\_RecCounter 4 symbols in format ####.*  
*For the number of cancel receipts*

**zfpdef:** *ReadLastCurrencyReceiptNum()*

## 2.8.18. Command: 72h / r – Read information about the current opened receipt

**input: n. a.**

**output:** *<IsReceiptOpened[1]> <;> <SalesNumber[3]> <;>*

*<SubtotalAmountVATGA[1..11]> <;> <SubtotalAmountVATGB[1..11]> <;>*

*<SubtotalAmountVATGC[1..11]> <;> <SubtotalAmountVATGD[1..11]> <;>*

*<SubtotalAmountVATGE[1..11]> <;> <ForbiddenVoid[1]> <;> <VATinReceipt[1]> <;>*

*<ReceiptFormat[1]> <;> <InitiatedPayment[1]> <;> <FinalizedPayment[1]> <;>*

*<PowerDownInReceipt[1]> <;> <ClientReceipt[1]> <;> <ChangeAmount[1..11]> <;>*

*<OptionChangeType[1]> <;> <AlteTaxeValue[1..11]>*

**FPR operation:** Read the current status of the receipt.

**Input data : n. a.**

**Output data :**

<i>IsReceiptOpened</i>	1 symbol with value: - '0' - No - '1' - Yes
<i>SalesNumber</i>	3 symbols for number of sales
<i>SubtotalAmountVATGA</i>	Up to 11 symbols for subtotal by VAT group A
<i>SubtotalAmountVATGB</i>	Up to 11 symbols for subtotal by VAT group B
<i>SubtotalAmountVATGC</i>	Up to 11 symbols for subtotal by VAT group C
<i>SubtotalAmountVATGD</i>	Up to 11 symbols for subtotal by VAT group D
<i>SubtotalAmountVATGE</i>	Up to 11 symbols for subtotal by VAT group E
<i>ForbiddenVoid</i>	1 symbol with value: - '0' – allowed - '1' - forbidden
<i>VATinReceipt</i>	1 symbol with value: - '0' – with printing - '1' - without printing
<i>ReceiptFormat</i>	(Format) 1 symbol with value: - '1' - Detailed - '0' - Brief
<i>InitiatedPayment</i>	1 symbol with value: - '0' – initiated payment - '1' - not initiated payment
<i>FinalizedPayment</i>	1 symbol with value: - '0' – finalized payment - '1' - not finalized payment
<i>ClientReceipt</i>	1 symbol with value: - '0' - standard receipt - '1' - invoice (client) receipt
<i>PowerDownInReceipt</i>	1 symbol with value: - '0' - No - '1' - Yes



<i>ChangeAmount</i>	Up to 11 symbols the amount of the due change in the stated payment type
<i>OptionChangeType</i>	1 symbols with value: - '0' - Change In Cash - '1' - Same As The payment - '2' – Change In Currency
<i>AlteTaxeValue</i>	Up to 11 symbols for alte tax amount
<b>zfpdef:</b> <a href="#">ReadCurrentReceiptInfo()</a>	

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

**input:** n. a.

**output:** <LastZDailyReportDate “DD-MM-YYYY”> <;> <LastZDailyReportNum[1..4]>  
<;> <LastRAMResetNum[1..4]>

**FPR operation:** Read the date and number of the last Z-report and the last RAM reset event.

**Input data : n. a.**

**Output data :**

<i>LastZDailyReportDate</i>	10 symbols for last Z-report date in DD-MM-YYYY format
<i>LastZDailyReportNum</i>	Up to 4 symbols for the number of the last daily report
<i>LastRAMResetNum</i>	Up to 4 symbols for the number of the last RAM reset

**zfpdef:** [ReadLastDailyReportInfo\(\)](#)

### 2.8.20. 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 :**

<i>FreeFMrecords</i>	4 symbols for the number of free records for Z-report in the FM
----------------------	---

**zfpdef:** [ReadFMfreeRecords\(\)](#)

### 2.8.21. 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 :**

<i>Nsegm</i>	4 symbols for physical FM block number
<i>OptionCodeStored</i>	1 symbol stating the type of the stored block with the following values: 0 – factory number of FPR 1 – tax number, decimal point position and tax rate at fiscalization 4 – daily financial report 6 – change of VAT rates 7 – change of decimal point position
<i>DateStored</i>	16 symbols for the date and time of block storing
<i>Status</i>	1 symbol 0 or 1 resp. for correct/incorrect block checksum
<i>ReadData</i>	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.

### 2.9.1. Command: 76h / v – Print department report

**input:** <OptionZeroing[1]>

**output:** ACK

**FPR operation:** Prints departments report

**Input data :**

**OptionZeroing** 1 symbol (Parameter) with following values:  
- 'Z' –Zeroing  
- 'X' – Not zeroing

**Output data : n. a.**

**zfpdef:** PrintDepartmentReport(OptionZeroing)

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

**input:** n. a.

**output:** ACK

**FPR operation:** Print all special FM events report.

**Input data : n. a.**

**Output data : n. a.**

**zfpdef:** PrintSpecialEventsFMReport()

### 2.9.3. Command: 77h / w – Print special events FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]>

**output:** ACK

**FPR operation:** Print the special FM events report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format ####

**EndNum** (End) 4 symbols for the final report number included in report, format ####

**Output data: n. a.**

**zfpdef:** PrintSpecialFMReportByZReportNum(StartNum, EndNum)

### 2.9.4. Command: 77h / w – Print special events FM report by date

**input:** <StartDate “DDMMYY”><;><EndDate “DDMMYY”>

**output:** ACK

**FPR operation:** Prints the special FM events 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:** PrintSpecialFMReportByDate(StartDate, EndDate)

### 2.9.5. Command: 78h / x – Print detailed FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]>

**output:** ACK

**FPR operation:** Print a detailed FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format ####

**EndNum** (End) 4 symbols for the final report number included in report, format ####

**Output data: n. a.**

**zfpdef:** *PrintDetailedFMReportByZNum(StartNum, EndNum)*

### 2.9.6. Command: 78h / x – Print detailed Payment FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]><;><OptionPayment["P"]>

**output:** ACK

**FPR operation:** Print a detailed FM payment report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format ####

**EndNum** (End) 4 symbols for the final report number included in report, format ####

**OptionPayment** 1 symbol 'P'

**Output data: n. a.**

**zfpdef:** *PrintDetailedFMPaymentsReportByZNum(StartNum, EndNum)*

### 2.9.7. Command: 78h / x – Store detailed FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]><;><OptionStorage[1]>

**output:** ACK

**FPR operation:** Storage a detailed FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format ####

**EndNum** (End) 4 symbols for the final report number included in report, format ####

**OptionStorage** (Storage FM Report) 1 symbol for destination:  
- '2' – Storage in External USB Flash memory.  
- '4' – Storage in External SD card memory

**Output data: n. a.**

**zfpdef:** *StoreDetailedFMReportByZNum(StartNum, EndNum, OptionStorage)*

### 2.9.8. Command: 79h / y – Print brief FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]>

**output:** ACK

**FPR operation:** Print a brief FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial FM report number included in report, format ####

**EndNum** (End) 4 symbols for the final FM report number included in report, format ####

**Output data: n. a.**

**zfpdef:** *PrintBriefFMReportByZNum(StartNum, EndNum)*

### 2.9.9. Command: 79h / y – Print brief payment FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]><;><OptionPayment['P']>

**output:** ACK

**FPR operation:** Print a brief payment FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format #####

**EndNum** (End) 4 symbols for the final report number included in report, format #####

**OptionPayment** 1 symbol 'P'

**Output data: n. a.**

**zfpdef:** *PrintBriefFMPaymentsReportByZNum(StartNum, EndNum)*

### 2.9.10. Command: 79h / y – Store brief FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]> {<;><OptionStorage[1]>}

**output:** ACK

**FPR operation:** Store a brief FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format #####

**EndNum** (End) 4 symbols for the final report number included in report, format #####

**OptionStorage** (Storage FM Report) 1 symbol for destination:  
- '2' – Storage in External USB Flash memory.  
- '4' – Storage in External SD card memory

**Output data: n. a.**

**zfpdef:** *StoreBriefFMReportByNum(StartNum, EndNum, OptionStorage)*

### 2.9.11. 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)*

### 2.9.12. Command: 7Ah / z – Print detailed Payment FM report by date

**input:** <StartDate "DDMMYY"> <;> <EndDate "DDMMYY"> <;> <OptionPayment['P']>

**output:** ACK

**FPR operation:** Print a detailed payment 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

**OptionPayment** 1 symbol 'P'

**Output data: n. a.**

**zfpdef:** *PrintDetailedFMPaymentsReportByDate(StartDate, EndDate)*

### 2.9.13. Command: 7Ah / z – Store detailed FM report by date

**input:** <StartDate "DDMMYY"><;><EndDate "DDMMYY"> {<;><OptionStorage[1]>}

**output:** ACK

**FPR operation:** Storage 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

OptionStorage (Storage FM Report) 1 symbol for destination:  
- '2' – Storage in External USB Flash memory.  
- '4' – Storage in External SD card memory

**Output data: n. a.**

**zfpdef:** *StoreDetailedFMReportByDate(StartDate, EndDate, OptionStorage)*

### 2.9.14. 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)*

### 2.9.15. Command: 7Bh / { – Print brief payment FM report by date

**input:** <StartDate "DDMMYY"> <;> <EndDate "DDMMYY"> <;> <OptionPayment["P"]>

**output:** ACK

**FPR operation:** Print a brief payment 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

OptionPayment 1 symbol 'P'

**Output data: n. a.**

**zfpdef:** *PrintBriefFMPaymentReportByDate(StartDate, EndDate)*

### 2.9.16. Command: 7Bh / { – Store brief FM report by date

**input:** <StartDate "DDMMYY"><;><EndDate "DDMMYY"> {<;><OptionStorage[1]>}

**output:** ACK

**FPR operation:** Store 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

OptionStorage (Storage FM Report) 1 symbol for destination:  
- '2' – Storage in External USB Flash memory.  
- '4' – Storage in External SD card memory

**Output data: n. a.**

**zfpdef:** *StoreBriefFMReportByDate(StartDate, EndDate, OptionStorage)*

### 2.9.17. 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 symbol (Parameter) with following values:  
- 'Z' –Zeroing  
- 'X' – Not zeroing

**Output data:** n.a.

**zfpdef:** *PrintDailyReport(OptionZeroing)*

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

**input:** <OptionReportStorage[2]> <;> <'D'> <;> <StartRepFromDate “DDMMYY”> <;>  
<EndRepFromDate “DDMMYY”>

**output:** ACK

**FPR operation:** Store Electronic Journal Report from report from date to date to External USB Flash memory, External SD card or Print.

**Input data:**

*OptionReportStorage* (Printing/Storage EJ Report) 2 symbols for destination:  
- 'J1' – Printing  
- 'J2' – Storage in External USB Flash memory.  
- 'J4' – Storage in External SD card memory  
  
'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(OptionReportStorage, StartDate, EndDate)*

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

**input:** <OptionReportStorage[2]><;><'N'><;><StartNum[6]><;><EndNum[6]>

**output:** ACK

**FPR operation:** Store Electronic Journal Report from receipt number to receipt number to External USB Flash memory, External SD card or Print.

**Input data:**

*OptionReportStorage* (Printing/Storage EJ Report) 2 symbols for destination:  
- 'J1' – Printing  
- 'J2' – Storage in External USB Flash memory.  
- 'J4' – Storage in External SD card memory  
  
'N' 1 symbol 'N'  
*StartNum* (Start rec. Number) 6 symbols for initial receipt number in format ##### included in report.  
*EndNum* (End rec. Number) 6 symbolsfor final receipt number included in format ##### in report.

**Output data:** n. a.

**zfpdef:** *PrintOrStoreEJByReceiptNum(OptionReportStorage, StartNum, EndNum)*



## 2.9.20. Command: 7Ch / | – Print/Store Electronic Journal report from number Z report to number Z report

**input:** <OptionReportStorage[2]><;><'Z'><;><StartNum[4]><;><EndNum[4]>

**output:** ACK

**FPR operation:** Store Electronic Journal Report from report number to report number to External USB Flash memory, External SD card or Print.

### **Input data:**

*OptionReportStorage* (Printing/Storage EJ Report) 2 symbols for destination:

- 'J1' – Printing
- 'J2' – Storage in External USB Flash memory.
- 'J4' – Storage in External SD card memory

'Z' 1 symbol 'Z'

*StartNum* (Start Z Number)4 symbols for initial number report in format ####

*EndNum* (End Z Number)4 symbols for final number report in format ####

### **Output data: n. a.**

**zfpdef:** PrintOrStoreEJByZReportNum(*OptionRepStorage*, *StartNum*, *EndNum*)

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

**input:** <OptionReportStorage[2]><;><'\*'>

**output:** ACK

**FPR operation:** Store whole Electronic Journal report to External USB Flash memory, External SD card or Print.

### **Input data:**

*OptionReportStorage* (Printing/Storage EJ Report) 2 symbols for destination:

- 'J1' – Printing
- 'J2' – Storage in External USB Flash memory.
- 'J4' – Storage in External SD card memory

'\*' 1 symbol '\*'

### **Output data: n. a.**

**zfpdef:** PrintOrStoreEJ()

## 2.9.22. Command: 7Ch / | – Export xml format receipts from number Z report to number Z report

**input:** <OptionRecieptXmlStorage[2]><;><'Z'><;><StartNum[4]><;><EndNum[4]>

**output:** ACK

**FPR operation:** Storage of receipts xml files by Z-report number to USB Flash memory or SD card memory.

### **Input data:**

*OptionRecieptXmlStorage* (Storage receipts xml) 2 symbols for destination:

- 'Jx' – Storage in External USB Flash memory.
- 'JX' – Storage in External SD card memory

'Z' 1 symbol 'Z'

*StartNum* (Start Z Number)4 symbols for initial number report in format ####

*EndNum* (End Z Number)4 symbols for final number report in format ####

### **Output data: n. a.**

**zfpdef:** StoreEJByZReportNum(*OptionRecieptXmlStorage*,*StartNum*, *EndNum*)

### 2.9.23. 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 symbol (Parameter) with following values:

- 'Z' –Zeroing
- 'X' – Not zeroing

*Number* (Operator Number) Symbols from 0 to 20 corresponding to operator's number,  
0 = all operators

**Output data: n. a.**

**zfpdef:** PrintOperatorReport(*OptionZeroing*, *Number*)

### 2.9.24. 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 symbol (Parameter) with following values:

- 'Z' –Zeroing
- 'X' – Not zeroing

**Output data: n. a.**

**zfpdef:** PrintArticleReport(*OptionZeroing*)

### 2.9.25. 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 symbol (Parameter) with following values:

- 'Z' –Zeroing
- 'X' – Not zeroing

**Output data: n. a.**

**zfpdef:** PrintDetailedDailyReport(*OptionZeroing*)

### 2.9.26. Command: 52h / R – Print Customer X or Z report

**input:** <OptionZeroing[1]>

**output:** ACK

**FPR Operation:** Print Customer X or Z report

**Input data:**

*OptionZeroing* 1 symbol (Parameter) with following values:

- 'Z' –Zeroing
- 'X' – Not zeroing

**Output data: n.a.**

**zfpdef:** PrintCustomerReport(*OptionZeroing*)



### 2.9.27. Command: 59h / Y – Print hourly report, Option H

**Input:** <'H'>

**output:** ACK

**FPR Operation:** Print hourly report

*\*the command is valid for ADPOS model only.*

**Input data:**

'H' 1 symbol 'H'

**Output data:** n.a.

**zfpdef:** PrintHourlyReport()

### 2.9.28. Command: 5Ah / Z – Read Z report number from date, Option 1

**input:** <Option['1']> <;> <DateFrom "DDMMYY">

**output:** <Option['1']> <;> <ZreportNum[4]>

**FPR operation:** Provide information about first located Z report, having the same date or the date after the input date

**Input data :**

Option 1 symbol with value '1'

DateFrom 6 symbols for specified date in format "DDMMYY"

**Output data:**

Option 1 symbol with value '1'

ZreportNum 4 symbols for the first found Z report number in format: ####

**zfpdef:** ReadFirstZReportNumFromDate(Date)

### 2.9.29. Command: 5Ah / Z – Read Z report number from date, Option 2

**input:** <Option['2']> <;> <DateFrom "DDMMYY">

**output:** <Option['2']> <;> <ZreportNum[4]>

**FPR operation:** Provide information about last found Z report, having the same date or date before than the input date

**Input data :**

Option 1 symbol with value '2'

DateFrom 6 symbols for specified date in format "DDMMYY"

**Output data:**

Option 1 symbol with value '2'

ZreportNum 4 symbols for the last found Z report number in format: ####

**zfpdef:** ReadLastZReportNumFromDate(Date)

## 2.10. REPORTS READING COMMANDS

Set of commands for reading of reports generated by FD.

### 2.10.1. Command: 78h / x – Read detailed FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]><;><PCStorage[‘8’]>

**output:** ACK+

**FPR operation:** Storage a detailed FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format #####

**EndNum** (End) 4 symbols for the final report number included in report, format #####

**PCStorage** 1 symbol with value ‘8’

**Output data: n. a.**

**zfpdef:** *ReadDetailedFMReportByZNum(StartNum, EndNum)*

### 2.10.2. Command: 79h / y – Read brief FM report by number of Z reports

**input:** <StartNum[4]><;><EndNum[4]> <;><PCStorage[‘8’]>

**output:** ACK+

**FPR operation:** Store a brief FM report by initial and end FM report number.

**Input data :**

**StartNum** (Start) 4 symbols for the initial report number included in report, format #####

**EndNum** (End) 4 symbols for the final report number included in report, format #####

**PCStorage** 1 symbol with value ‘8’

**Output data: n. a.**

**zfpdef:** *ReadBriefFMReportByNum(StartNum, EndNum)*

### 2.10.3. Command: 7Ah / z – Read detailed FM report by date

**input:** <StartDate “DDMMYY”><;><EndDate “DDMMYY”> <;><PCStorage[‘8’]>

**output:** ACK+

**FPR operation:** Storage 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

**PCStorage** 1 symbol with value ‘8’

**Output data: n. a.**

**zfpdef:** *ReadDetailedFMReportByDate(StartDate, EndDate)*

### 2.10.4. Command: 7Bh / { – Read brief FM report by date

**input:** <StartDate “DDMMYY”><;><EndDate “DDMMYY”> <;><PCStorage[‘8’]>

**output:** ACK+

**FPR operation:** Store 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

**PCStorage** 1 symbol with value ‘8’

**Output data: n. a.**

**zfpdef:** *ReadBriefFMReportByDate(StartDate, EndDate)*

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

**input:** <'J0'><;><'D'><;><StartRepFromDate "DDMMYY"><;>  
<EndRepFromDate "DDMMYY">

**output:** ACK+

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

#### **Input data:**

'J0' (Reading EJ Report) 1 character with value 'J0'  
'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(StartDate, EndDate)

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

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

**output:** ACK+

**FPR operation:** Read or Store Electronic Journal Report by initial to end date and document content. 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) 2 characters with value:  
- 'j0' – To PC  
- 'j2' – To USB Flash Drive  
- 'j4' – To SD card  
reserved 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=0  
Flags.4=0  
Flags.3=0  
Flags.2=0  
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=0  
Flags.3=1 Yes, Flags.3=0 No (Include Non-Fiscal Rcp)  
Flags.2=0  
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, FlagsReceipts, FlagsReports, StartRepFromDate, EndRepFromDate)

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

**input:** <'J0'><;><'N'><;><StartNum[6]><;><EndNum[6]>

**output:** ACK+

**FPR operation:** Read Electronic Journal Report from receipt number to receipt number.

#### **Input data:**

'J0' (Reading EJ Report) 1 character with value 'J0'

'N' 1 symbol 'N'

StartNum (Start rec.No) 6 symbols for initial receipt number included in report in format #####.

EndNum (End rec.No) 6 symbols for final receipt number included in report in format #####.

#### **Output data: n. a.**

**zfpdef:** ReadEJByReceiptNum(StartNum, EndNum)

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

**input:** <StorageReport[2]> <;> <reserved['X']> <;> <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 and document content. 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

reserved 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=0

Flags.4=0

Flags.3=0

Flags.2=0

Flags.1=1 Yes, Flags.1=0 No (Include Invoice)

Flags.0=1 Yes, Flags.0=0 No (Include Fiscal Rcp)

FlagsReports (Reports included in EJ) 1 symbol for Reports included in EJ:

Flags.7=0

Flags.6=1

Flags.5=0

Flags.4=0

Flags.3=1 Yes, Flags.3=0 No (Include Non-Fiscal Rcp)

Flags.2=0

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, FlagsReceipts, FlagsReports, StartNo, EndNo)

### 2.10.9. Command: 7Ch / | – Read Electronic Journal report from number Z report to number Z report

**input:** <'J0'><;><'Z'><;><StartNum[4]><;><EndNum[4]>

**output:** ACK+

**FPR operation:** Read Electronic Journal Report from report number to report number.

**Input data:**

'J0' (Reading EJ Report) 1 character with value 'J0'  
'Z' 1 symbol 'Z'  
StartNum (Start Z No)4 symbols for initial number report in format ####  
EndNum (End Z No)4 symbols for final number report in format ####

**Output data:** n. a.

**zfpdef:** ReadEJByZReportNum(StartNum, EndNum)

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

**input:** <StorageReport[2]> <;> <reserved['X']> <;> <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. 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  
reserved 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=0  
Flags.4=0  
Flags.3=0  
Flags.2=0  
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=0  
Flags.3=1 Yes, Flags.3=0 No (Include Non-Fiscal Rcp)  
Flags.2=0  
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, FlagsReceipts, FlagsReports, StartNo, EndNo)

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

**input:** <'J0'><;><'\*'>

**output:** ACK+

**FPR operation:** Read whole Electronic Journal report from beginning to the end.

#### **Input data:**

'J0' (Reading EJ Report) 1 character with value 'J0'  
'\*' 1 symbol '\*'

#### **Output data: n. a.**

**zfpdef:** ReadEJ()

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

**input:** <StorageReport[2]> <;> <reserved['X']> <;> <FlagsReceipts[1]> <;>  
<FlagsReports[1]> <;> <'\*'>

**output:** ACK+

**FPR operation:** Read or Store Electronic Journal report and document content selecting. 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

**reserved** 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=0
- Flags.4=0
- Flags.3=0
- Flags.2=0
- 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=0
- Flags.3=1 Yes, Flags.3=0 No (Include Non-Fiscal Rcp)
- Flags.2=0
- 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, FlagsReceipts, FlagsReports)

## 2.11. SETTINGS LAN/WIFI/BLUETOOTH/GPRS COMMANDS

Set/Get commands for programming/reading of FD settings for LAN, WiFi, Bluetooth, GRPS.

### 2.11.1. Command: 4Eh / N – Read Device modules support by Firmware

**input:** <'R'> <;> <'D'> <;> <'S'>

**output:** <'R'> <;> <'D'> <;> <'S'> <;> <LAN[1]> <;> <WiFi[1]> <;> <GPRS[1]> <;> <BT[1]>

**FPR operation:** Provide an information about modules supported by device's firmware.

#### **Input data:**

'R' 1 symbol with value 'R'  
'D' 1 symbol with value 'D'  
'S' 1 symbol with value 'S'

#### **Output data:**

'R' 1 symbol with value 'R'  
'D' 1 symbol with value 'D'  
'S' 1 symbol with value 'S'  
*LAN* 1 symbol for LAN support  
- '0' – No  
- '1' – Yes  
*WiFi* 1 symbol for WiFi support  
- '0' – No  
- '1' – Yes  
*GPRS* 1 symbol for GPRS support  
- '0' – No  
- '1' – Yes  
*BT* (Bluetooth) 1 symbol for Bluetooth support  
- '0' – No  
- '1' – Yes

**zfpdef:** [ReadDeviceModuleSupportByFirmware\(\)](#)

### 2.11.2. Command: 4Eh / N – Read Device modules support

**input:** <'R'> <;> <'D'> <;> <'D'>

**output:** <'R'> <;> <'D'> <;> <'D'> <;> <LAN[1]> <;> <WiFi[1]> <;>  
<GPRS[1]> <;> <BT[1]>

**FPR operation:** Provide an information about modules supported by the device.

**Input data:**

'R' 1 symbol with value 'R'  
'D' 1 symbol with value 'D'  
'D' 1 symbol with value 'D'

**Output data:**

'R' 1 symbol with value 'R'  
'D' 1 symbol with value 'D'  
'D' 1 symbol with value 'D'  
LAN 1 symbol for LAN support  
- '0' – No  
- '1' – Yes  
WiFi 1 symbol for WiFi support  
- '0' – No  
- '1' – Yes  
GPRS 1 symbol for GPRS support  
- '0' – No  
- '1' – Yes  
BT (Bluetooth) 1 symbol for Bluetooth support  
- '0' – No  
- '1' – Yes

**zfpdef:** [ReadDeviceModuleSupport\(\)](#)

### 2.11.3. Command: 4Eh / N – Read TCP password

**input:** <'R'> <;> <'Z'> <;> <'1'>

**output:** <'R'> <;> <'Z'> <;> <'1'> <;> <PassLength[1..3]> <;> <Password[100]>

**FPR operation:** Provides information about device's TCP password.

**Input data:**

'R' 1 symbol with value 'R'  
'Z' 1 symbol with value 'Z'  
'1' 1 symbol with value '1'

**Output data:**

'R' 1 symbol with value 'R'  
'Z' 1 symbol with value 'Z'  
'1' 1 symbol with value '1'  
PassLength (Length) Up to 3 symbols for the password length  
Password Up to 100 symbols for the TCP password

**zfpdef:** [ReadTCP\\_Password\(\)](#)



#### 2.11.4. Command: 4Eh / N – Read TCP Auto Start status

**input:** <'R'> <;> <'Z'> <;> <'2'>

**output:** <'R'> <;> <'Z'> <;> <'2'> <;> <TCPAutoStart[1]>

**FPR operation:** Read device TCP Auto Start status

**Input data:**

'R' 1 symbol with value 'R'  
'Z' 1 symbol with value 'Z'  
'2' 1 symbol with value '2'

**Output data:**

'R' 1 symbol with value 'R'  
'Z' 1 symbol with value 'Z'  
'2' 1 symbol with value '2'  
TCPAutoStart 1 symbol for TCP auto start status  
- '0' – No  
- '1' – Yes

**zfpdef:** ReadTCP\_AutoStartStatus()

#### 2.11.5. Command: 4Eh / N – Read Device TCP addresses

**input:** <'R'> <;> <'T'> <;> <AddressType[1]>

**output:** <'R'> <;> <'T'> <;> <AddressType[1]> <;> <DeviceAddress[15]>

**FPR operation:** Provides information about device's IP address, subnet mask, gateway address, DNS address.

**Input data:**

'R' 1 symbol with value 'R'  
'T' 1 symbol with value 'T'  
AddressType (Address) 1 symbol with value:  
- '2' – IP address  
- '3' – Subnet Mask  
- '4' – Gateway address  
- '5' – DNS address

**Output data:**

'R' 1 symbol with value 'R'  
'T' 1 symbol with value 'T'  
AddressType (Address) 1 symbol with value:  
- '2' – IP address  
- '3' – Subnet Mask  
- '4' – Gateway address  
- '5' – DNS address  
DeviceAddress 15 symbols for the device's addresses

**zfpdef:** ReadTCP\_Addresses(AddressType)

### 2.11.6. Command: 4Eh / N – Read TCP DHCP status

**input:** <'R'> <;> <'T'> <;> <'1'>

**output:** <'R'> <;> <'T'> <;> <'1'> <;> <DhcpStatus[1]>

**FPR operation:** Provides information about device's DHCP status

**Input data:**

'R' 1 symbol with value 'R'  
'T' 1 symbol with value 'T'  
'1' 1 symbol with value '1'

**Output data:**

'R' 1 symbol with value 'R'  
'T' 1 symbol with value 'T'  
'1' 1 symbol with value '1'  
DhcpStatus (DHCP Status) 1 symbol with value:  
- '0' – Disabled  
- '1' – Enabled

**zfpdef:** ReadDHCP\_Status()

### 2.11.7. Command: 4Eh / N – Scan and print available WiFi networks

**input:** <'R'> <;> <'W'> <;> <'S'>

**output:** ACK

**FPR operation:** Scan and print all available WiFi networks

**Input data:**

'R' 1 symbol with value 'R'  
'W' 1 symbol with value 'W'  
'S' 1 symbol with value 'S'

**Output data:** n. a.

**zfpdef:** ScanAndPrintWiFiNetworks()

### 2.11.8. Command: 4Eh / N – Read TCP WiFi network name

**input:** <'R'> <;> <'W'> <;> <'N'>

**output:** <'R'> <;> <'W'> <;> <'N'> <;> <WiFiNameLength[1..3]>

<;> <WiFiNetworkName[100]>

**FPR operation:** Read device's connected WiFi network name

**Input data:**

'R' 1 symbol with value 'R'  
'W' 1 symbol with value 'W'  
'N' 1 symbol with value 'N'

**Output data:**

'R' 1 symbol with value 'R'  
'W' 1 symbol with value 'W'  
'N' 1 symbol with value 'N'  
WiFiNameLength (Length) Up to 3 symbols for the WiFi name length  
WiFiNetworkName (Name) Up to 100 symbols for the device's WiFi network name

**zfpdef:** ReadWiFi\_NetworkName()

### 2.11.9. Command: 4Eh / N – Read TCP WiFi password

**input:** <'R'> <;> <'W'> <;> <'P'>

**output:** <'R'> <;> <'W'> <;> <'P'> <;> <PassLength[1..3]> <;> <Password[100]>

**FPR operation:** Read device's connected WiFi network password

#### **Input data:**

'R'	1 symbol with value 'R'
'W'	1 symbol with value 'W'
'P'	1 symbol with value 'P'

#### **Output data:**

'R'	1 symbol with value 'R'
'W'	1 symbol with value 'W'
'P'	1 symbol with value 'P'
PassLength	(Length) Up to 3 symbols for the WiFi password length
Password	Up to 100 symbols for the device's WiFi password

**zfpdef:** ReadWiFi\_Password()

### 2.11.10. Command: 4Eh / N – Read TCP module - LAN or WiFi

**input:** <'R'> <;> <'Z'> <;> <'U'>

**output:** <'R'> <;> <'Z'> <;> <'U'> <;> <UsedModule[1]>

**FPR operation:** Read the used TCP module for communication – Lan or WiFi.

Command is available if the device support both modules only.

#### **Input data:**

'R'	1 symbol with value 'R'
'Z'	1 symbol with value 'Z'
'U'	1 symbol with value 'U'

#### **Output data:**

'R'	1 symbol with value 'R'
'Z'	1 symbol with value 'Z'
'U'	1 symbol with value 'U'
UsedModule	(Module) 1 symbol with value: - '1' – LAN - '2' – WiFi

**zfpdef:** ReadTCP\_UsedModule()

### 2.11.11. Command: 4Eh / N – Read TCP idle timeout

**input:** <'R'> <;> <'Z'> <;> <'I'>

**output:** <'R'> <;> <'Z'> <;> <'I'> <;> <IdleTimeout[4]>

**FPR operation:** Provides information about device's idle timeout. This timeout is for closing the connection if there is an inactivity. Maximal value – 7200, minimal value 1. 0 is for never close the connection. This option can be used only if the device has LAN or WiFi.

#### **Input data:**

'R'	1 symbol with value 'R'
'B'	1 symbol with value 'B'
'I'	1 symbol with value 'I'

#### **Output data:**

'R'	1 symbol with value 'R'
'B'	1 symbol with value 'B'
'I'	1 symbol with value 'I'
IdleTimeout	4 symbols for password in format ####

**zfpdef:** Read\_IdleTimeout()

### 2.11.12. Command: 4Eh / N – Read Bluetooth password

**input:** <'R'> <;> <'B'> <;> <'P'>

**output:** <'R'> <;> <'B'> <;> <'P'> <;> <PassLength[1..3]> <;> <Password[100]>

**FPR operation:** Provides information about device's Bluetooth password.

#### **Input data:**

'R' 1 symbol with value 'R'  
'B' 1 symbol with value 'B'  
'P' 1 symbol with value 'P'

#### **Output data:**

'R' 1 symbol with value 'R'  
'B' 1 symbol with value 'B'  
'P' 1 symbol with value 'P'  
PassLength (Length) Up to 3 symbols for the BT password length  
Password Up to 100 symbols for the BT password

**zfpdef:** [ReadBluetooth\\_Password\(\)](#)

### 2.11.13. Command: 4Eh / N – Read Bluetooth status

**input:** <'R'> <;> <'B'> <;> <'S'>

**output:** <'R'> <;> <'B'> <;> <'S'> <;> <BTstatus[1]>

**FPR operation:** Providing information about if the device's Bluetooth module is enabled or disabled.

#### **Input data:**

'R' 1 symbol with value 'R'  
'B' 1 symbol with value 'B'  
'S' 1 symbol with value 'S'

#### **Output data:**

'R' 1 symbol with value 'R'  
'B' 1 symbol with value 'B'  
'S' 1 symbol with value 'S'  
BTstatus (Status) 1 symbol with value:  
- '0' – Disabled  
- '1' - Enabled

**zfpdef:** [ReadBluetooth\\_Status\(\)](#)

### 2.9.13. Command: 4Eh / N – Read GPRS APN

**input:** <'R'><;><'G'><;><'A'>

**output:** <'R'><;><'G'><;><'A'><;><GPRS\_APN\_Len[1..3]><;><APN[100]>

**FPR operation:** Provides information about device's GRPS APN.

#### **Input data:**

'R' 1 symbol with value 'R'  
'G' 1 symbol with value 'G'  
'A' 1 symbol with value 'A'

#### **Output data:**

'R' 1 symbol with value 'R'  
'G' 1 symbol with value 'G'  
'A' 1 symbol with value 'A'  
GPRS\_APN\_Len (Length) Up to 3 symbols for the APN length  
APN Up to 100 symbols for the device's GPRS APN

**zfpdef:** [ReadGPRS\\_APN\(\)](#)

### 2.11.14. Command: 4Eh / N – Read TCP device MAC Address

**input:** <'R'> <;> <'T'> <;> <'6'>

**output:** <'R'> <;> <'T'> <;> <'6'> <;> <DeviceMAC[12]>

**FPR operation:** Provides information about TCP device MAC address

**Input data:**

'R'	1 symbol with value 'R'
'T'	1 symbol with value 'T'
'6'	1 symbol with value '6'

**Output data:**

'R'	1 symbol with value 'R'
'T'	1 symbol with value 'T'
'6'	1 symbol with value '6'
DeviceMAC	12 symbols for device MAC

**zfpdef:** [ReadDeviceMAC\\_Address\(\)](#)

### 2.11.15. Command: 4Eh / N – Read Server Address

**input:** <'R'> <;> <'S'> <;> <'S'>

**output:** <'R'> <;> <'S'> <;> <'S'> <;> <ParamLength[1..3]> <;>  
<ServerAddress [100]>

**FPR operation:** Provides information about the ECR's password

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'S'	1 symbol with value 'S'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'S'	1 symbol with value 'S'
ParamLength	Up to 3 symbols for parameter length
ServerAddress	Up to 100 symbols for server password

**zfpdef:** [ReadServerAddress\(\)](#)

### 2.11.16. Command: 4Eh / N – Read Server ECRS Password

**input:** <'R'> <;> <'S'> <;> <'Q'>

**output:** <'R'> <;> <'S'> <;> <'Q'> <;> <ParamLength[1..2]> <;>  
<ServerPassword[64]>

**FPR operation:** Provides information about the ECR's password

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'Q'	1 symbol with value 'Q'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'Q'	1 symbol with value 'Q'
ParamLength	Up to 2 symbols for parameter length
ServerPassword	Up to 64 symbols for server password

**zfpdef:** [ReadServerPasswordECRS\(\)](#)

### 2.11.17. Command: 4Eh / N – Read Server Communication Module

**input:** <'R'> <;> <'S'> <;> <'E'>

**output:** <'R'> <;> <'S'> <;> <'E'> <;> <CommunicationModule[1]>

**FPR operation:** Provides information about the communication module, used for talking with the server

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'E'	1 symbol with value 'E'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'E'	1 symbol with value 'E'
CommunicationModule	1 symbol with value: - '0' – GSM - '1' – LAN

**zfpdef:** [ReadUsedComModule\(\)](#)

### 2.11.18. Command: 4Eh / N – Read GPRS password

**input:** <'R'><;><'G'><;><'P'>

**output:** <'R'><;><'G'><;><'P'><;><PassLength[1..3]><;><Password[100]>

**FPR operation:** Provides information about device's GPRS password.

#### **Input data:**

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'P'	1 symbol with value 'P'

#### **Output data:**

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'P'	1 symbol with value 'P'
PassLength	(Length) Up to 3 symbols for the GPRS password length
Password	Up to 100 symbols for the device's GPRS password

**zfpdef:** ReadGPRS\_Password()

### 2.11.19. Command: 4Eh / N – Read GPRS username

**input:** <'R'><;><'G'><;><'U'>

**output:** <'R'><;><'G'><;><'U'><;><GPRS\_User\_Len[1..3]><;><Username[100]>

**FPR operation:** Provides information about device's GPRS username.

#### **Input data:**

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'U'	1 symbol with value 'U'

#### **Output data:**

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'U'	1 symbol with value 'U'
GPRS_User_Len	(Length) Up to 3 symbols for the GPRS username length
Username	Up to 100 symbols for the device's GPRS username

**zfpdef:** ReadGPRS\_Username()

### 2.11.20. Command: 4Eh / N – Read GPRS signal

**input:** <'R'><;><'G'><;><'S'>

**output:** <'R'><;><'G'><;><'S'><;><GPRS\_Signal[3]>

**FPR operation:** Provides information about device's GPRS signal.

#### **Input data:**

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'S'	1 symbol with value 'S'

#### **Output data:**

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'S'	1 symbol with value 'S'
GPRS_Signal	(Signal) 3 symbols for the GPRS signal

**zfpdef:** ReadGPRS\_Signal()

### 2.11.21. Command: 4Eh / N – Read profile type

**input:** <'R'><;><'S'><;><'Y'>

**output:** <'R'><;><'S'><;><'Y'><;><ProfileType[1]>

**FPR operation:** Provides information about device's profile type.

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'Y'	1 symbol with value 'Y'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'Y'	1 symbol with value 'Y'
ProfileType	(Profile type) 1 symbol with value: - '0' - Profile 0 - '1' – Profile 1

**zfpdef:** ReadECRprofileType()

### 2.11.22. Command: 4Eh / N – Read profile active date

**input:** <'R'><;><'S'><;><'D'>

**output:** <'R'><;><'S'><;><'D'><;><ProfileActiveDate "DDMMYYYY">

**FPR operation:** Provides information about active profile date - date from which the account is valid or date from which we return to account 1 in case of mReset.

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'D'	1 symbol with value 'D'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'D'	1 symbol with value 'D'
ProfileActiveDate	(Profile active date) 8 symbols in format DDMMYYYY

**zfpdef:** ReadECRprofileActiveDate()

### 2.11.23. Command: 4Eh / N – Read profile send after Z report

**input:** <'R'><;><'S'><;><'Z'>

**output:** <'R'><;><'S'><;><'Z'><;><SendAfterZ[1]>

**FPR operation:** Provides information about sending of Z report to server automatically after Z report or not.

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'Z'	1 symbol with value 'Z'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'Z'	1 symbol with value 'Z'
SendAfterZ	(Send after Z report) 1 symbol with value: - '0' - No - '1' – Yes

**zfpdef:** ReadECRprofileZreportSending()



#### 2.11.24. Command: 4Eh / N – Read profile connection period

**input:** <'R'><;><'S'><;><'P'>

**output:** <'R'><;><'S'><;><'P'><;><ConnectionPeriod[4]>

**FPR operation:** Provides information about period in which the sending attempt is made.

**Input data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'P'	1 symbol with value 'P'

**Output data:**

'R'	1 symbol with value 'R'
'S'	1 symbol with value 'S'
'P'	1 symbol with value 'P'
ConnectionPeriod	4 symbols about connection in format ####

**zfpdef:** [ReadECRprofileConnectionPeriod\(\)](#)

#### 2.11.25. Command: 4Eh / N – Set TCP password

**input:** <'P'> <;> <'Z'> <;> <'1'> <;> <PassLength[1..3]> <;> <Password[100]>

**output:** ACK

**FPR operation:** Program device's TCP password. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'Z'	1 symbol with value 'Z'
'1'	1 symbol with value '1'
PassLength	(Password length) Up to 3 symbols for the password len
Password	Up to 100 symbols for the TCP password

**Output data: n. a.**

**zfpdef:** [SetTCPpassword\(PassLength, Password\)](#)

#### 2.11.26. Command: 4Eh / N – Set TCP Auto Start

**input:** <'P'> <;> <'Z'> <;> <'2'> <;> <TCPAutoStart[1]>

**output:** ACK

**FPR operation:** Set device's TCP autostart . To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'Z'	1 symbol with value 'Z'
'2'	1 symbol with value '2'
TCPAutoStart	(Auto Start) 1 symbol with value: - '0' – No - '1' - Yes

**Output data: n. a.**

**zfpdef:** [SetTCP\\_AutoStart\(TCPAutoStart\)](#)

### 2.11.27. Command: 4Eh / N – Set Device TCP addresses

**input:** <'P'> <;> <'T'> <;> <AddressType[1]> <;> <DeviceAddress[15]>

**output:** ACK

**FPR operation:** Program device's network IP address, subnet mask, gateway address, DNS address. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'T'	1 symbol with value 'T'
AddressType	(Type) 1 symbol with value: <ul style="list-style-type: none"><li>- '2' – IP address</li><li>- '3' – Subnet Mask</li><li>- '4' – Gateway address</li><li>- '5' – DNS address</li></ul>
DeviceAddress	(Device address)15 symbols for the selected address

**Output data: n. a.**

**zfpdef:** [SetDeviceTCP\\_Addresses\(AddressType, DeviceAddress\)](#)

### 2.11.28. Command: 4Eh / N – Set TCP DHCP enabled

**input:** <'P'> <;> <'T'> <;> <'1'> <;> <DhcpStatus[1]>

**output:** ACK

**FPR operation:** Program device's TCP network DHCP enabled or disabled. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'S'
'T'	1 symbol with value 'T'
'1'	1 symbol with value '1'
DhcpStatus	(DHCP Status)1 symbol with value: <ul style="list-style-type: none"><li>- '0' – Disabled</li><li>- '1' – Enabled</li></ul>

**Output data: n. a.**

**zfpdef:** [SetDHCP\\_Enabled\(DhcpStatus\)](#)

### 2.11.29. Command: 4Eh / N – Set TCP WiFi network name

**input:** <'P'> <;> <'W'> <;> <'N'> <;> <WiFiNameLength[1..3]> <;>  
<WiFiNetworkName[100]>

**output:** ACK

**FPR operation:** Program device's WiFi network name where it will connect. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'W'	1 symbol with value 'W'
'N'	1 symbol with value 'N'
WiFiNameLength	(Length) Up to 3 symbols for the WiFi network name len
WiFiNetworkName	(Name) Up to 100 symbols for the device's WiFi ssid network name

**Output data:**

**zfpdef:** [SetWiFi\\_NetworkName\(NetworkName\\_Length, NetworkName\)](#)

### 2.11.30. Command: 4Eh / N – Set TCP WiFi password

**input:** <'P'> <;> <'W'> <;> <'P'> <;> <PassLength[1..3]> <;> <Password[100]>

**output:** ACK

**FPR operation:** Program device's WiFi network password where it will connect. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'W'	1 symbol with value 'W'
'P'	1 symbol with value 'P'
PassLength	(Length) Up to 3 symbols for the WiFi password len
Password	Up to 100 symbols for the device's WiFi password

**Output data: n. a.**

**zfpdef:** SetWiFi\_Password(Password\_Length, Password)

### 2.11.31. Command: 4Eh / N – Set TCP module - LAN or WiFi

**input:** <'P'> <;> <'Z'> <;> <'U'> <;> <UsedModule[1]>

**output:** ACK

**FPR operation:** Sets the used TCP module for communication – Lan or WiFi. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'Z'	1 symbol with value 'Z'
'U'	1 symbol with value 'U'
UsedModule	(Module) 1 symbol with value: - '1' – LAN - '2' – WiFi

**Output data: n. a.**

**zfpdef:** SetTCP\_ActiveModule(ActiveModule)

### 2.11.32. Command: 4Eh / N – Set idle timeout

**input:** <'P'> <;> <'Z'> <;> <'I'> <;> <IdleTimeout[4]>

**output:** ACK

**FPR operation:** Sets device's idle timeout setting. Set timeout for closing the connection if there is an inactivity. Maximal value – 7200, minimal value 1. 0 is for never close the connection. This option can be used only if the device has LAN or WiFi. To apply use - 4Eh / N – Save network settings

**Input data:**

'P'	1 symbol with value 'P'
'Z'	1 symbol with value 'Z'
'I'	1 symbol with value 'I'
IdleTimeout	(Timeout) 4 symbols for Idle timeout in format ####

**Output data: n. a.**

**zfpdef:** SetIdle\_Timeout(IdleTimeout)

### 2.11.33. Command: 4Eh / N – Set Bluetooth password

input: <'P'> <;> <'B'> <;> <'P'> <;> <PassLength[1..3]> <;> <Password[100]>

output: ACK

FPR operation: Program device's Bluetooth password.

#### Input data:

'P'	1 symbol with value 'P'
'B'	1 symbol with value 'B'
'P'	1 symbol with value 'P'
PassLength	(Length) Up to 3 symbols for the BT password len
Password	Up to 100 symbols for the BT password

#### Output data: n. a.

zfpdef: SetBluetooth\_Password(Password\_Length, Password)

### 2.11.34. Command: 4Eh / N – Set Bluetooth module enable status

input: <'P'> <;> <'B'> <;> <'S'> <;> <BTstatus[1]>

output: ACK

FPR operation: Program device's Bluetooth module to be enabled or disabled.

#### Input data:

'P'	1 symbol with value 'P'
'B'	1 symbol with value 'B'
'S'	1 symbol with value 'S'
BTstatus	(Status) 1 symbol with value: - '0' – Disabled - '1' - Enabled

#### Output data: n. a.

zfpdef: SetBluetooth\_Status(BTstatus)

### 2.11.35. Command: 4Eh / N – Set TCP device MAC Address

input: <'P'> <;> <'T'> <;> <'6'> <;> <DeviceMAC[12]>

output: ACK

FPR operation: Provides information about TCP device MAC address

#### Input data:

'P'	1 symbol with value 'P'
'T'	1 symbol with value 'T'
'6'	1 symbol with value '6'
DeviceMAC	12 symbols for device MAC

#### Output data: n. a.

zfpdef: SetDeviceMAC\_Address ()

### 2.11.36. Command: 4Eh / N – Set Server ECRS Password

input: <'P'> <;> <'S'> <;> <'Q'> <;> <ParamLength[1..3]> <;>  
<ServerPassword[64]>

output: ACK

FPR operation: Program ECRS password

#### Input data:

'P'	1 symbol with value 'P'
'S'	1 symbol with value 'S'
'Q'	1 symbol with value 'Q'
ParamLength	Up to 2 symbols for parameter length
ServerPassword	Up to 64 symbols for server password

#### Output data: n. a.

zfpdef: SetServerPasswordECRS()

### 2.11.37. Command: 4Eh / N – Set Server Communication Module

input: <'P'> <;> <'S'> <;> <'E'> <;> <CommunicationModule[1]>

output: ACK

FPR operation: Program the communication module, which used to talk with the server

#### Input data:

'P'	1 symbol with value 'P'
'S'	1 symbol with value 'S'
'E'	1 symbol with value 'E'
CommunicationModule	1 symbol with value: - '0' – GSM - '1' – LAN

Output data: n. a.

zfpdef: SetServerCommunicationModule()

### 2.11.38. Command: 4Eh / N – Unpair all connected devices - BT

input: <'P'> <;> <'B'> <;> <'D'>

output: ACK

FPR operation: Removes all paired devices.

#### Input data:

'P'	1 symbol with value 'P'
'B'	1 symbol with value 'B'
'D'	1 symbol with value 'D'

Output data: n. a.

zfpdef: UnpairAllDevices()

### 2.11.39. Command: 4Eh / N – Save network settings

input: <'P'> <;> <'A'>

output: ACK

FPR operation: After every change on Idle timeout, LAN/WiFi/GPRS usage, LAN/WiFi/TCP/GPRS password or TCP auto start networks settings this Save command needs to be execute.

#### Input data:

'P'	1 symbol with value 'P'
'A'	1 symbol with value 'A'

Output data:n. a.

zfpdef: SaveNetworkSettings()

### 2.11.40. Command: 4Eh / N – Start device LAN test

input: <'R'> <;> <'T'> <;> <'T'>

output: ACK

FPR operation: Start LAN test on the device and print out the result

#### Input data:

'R'	1 symbol with value 'R'
'T'	1 symbol with value 'T'
'T'	1 symbol with value 'T'

Output data:n. a.

zfpdef: StartTest\_Lan()

#### 2.11.41. Command: 4Eh / N – Start device WiFi test

input: <'R'> <;> <'W'> <;> <'T'>

output: ACK

FPR operation: Start WiFi test on the device and print out the result

##### Input data:

'R'	1 symbol with value 'R'
'W'	1 symbol with value 'W'
'T'	1 symbol with value 'T'

Output data:n. a.

zfpdef: StartTest\_WiFi()

#### 2.11.42. Command: 4Eh / N – Start device GPRS test

input: <'R'> <;> <'G'> <;> <'T'>

output: ACK

FPR operation: Start GPRS test on the device and print out the result

##### Input data:

'R'	1 symbol with value 'R'
'G'	1 symbol with value 'G'
'T'	1 symbol with value 'T'

Output data:n. a.

zfpdef: StartTest\_GPRS()

#### 2.11.43. Command: 4Eh / N – Start device Bluetooth test

input: <'R'> <;> <'B'> <;> <'T'>

output: ACK

FPR operation: Start Bluetooth test on the device and print out the result

##### Input data:

'R'	1 symbol with value 'R'
'B'	1 symbol with value 'B'
'T'	1 symbol with value 'T'

Output data:n. a.

zfpdef: StartTest\_Bluetooth()

#### 2.11.44. Command: 4Eh / N – Set GPRS APN

input: <'P'><;><'G'><;><'A'><;><GPRS\_APN\_Len[1..3]><;><APN[100]>

output: ACK

FPR operation: Program device's GPRS APN. To apply use -  
SaveNetworkSettings()

##### Input data:

'P'	1 symbol with value 'P'
'G'	1 symbol with value 'G'
'A'	1 symbol with value 'A'
GPRS_APN_Len	(Length) Up to 3 symbols for the APN len
APN	Up to 100 symbols for the device's GPRS APN

Output data: n. a.

zfpdef: SetGPRS\_APN(APN\_Len, APN)

### 2.11.45. Command: 4Eh / N – Set GPRS password

**input:** <'P'><;><'G'><;><'P'><;><PassLength[1..3]><;><Password[100]>

**output:** ACK

**FPR operation:** Program device's GPRS password. To apply use -  
SaveNetworkSettings()

**Input data:**

'P'	1 symbol with value 'P'
'G'	1 symbol with value 'G'
'P'	1 symbol with value 'P'
PassLength	(Length) Up to 3 symbols for the GPRS password len
Password	Up to 100 symbols for the device's GPRS password

**Output data: n. a.**

**zfpdef:** SetGPRS\_Password(Password\_Len, Password)

### 2.11.46. Command: 4Eh / N – Set GPRS username

**input:**

<'P'><;><'G'><;><'U'><;><GPRS\_Username\_Len[1..3]><;><Username[100]>

**output:** ACK

**FPR operation:** Program device's GPRS user name. To apply use -  
SaveNetworkSettings()

**Input data:**

'P'	1 symbol with value 'P'
'G'	1 symbol with value 'G'
'U'	1 symbol with value 'U'
GPRS_Username_Len	(Length) Up to 3 symbols for the username len
Username	Up to 100 symbols for the device's GPRS username

**Output data: n. a.**

**zfpdef:** SetGPRS\_Username(Username\_Len, Username)



### **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 FPR**

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 class as well as non-compulsory information about the quantity sold and value/percent discount/addition;
- sale registration from the article database of FD (32h) – contains the number of the article 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 FPR, which contains the current parameters of the receipt, the number of sales, the accumulated amounts by VAT classes, 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: <b>Info</b> – character 49h - 'I'	<'I'><Number of printable characters per line[2]> <;><PLU number[4]> <;><Departments number[1..2]> <;><Operators number[1..2]> <;><VAT classs number[1]> <;><header lines number[1..2]> <;><payments number[1..2]> <;><logo number[1..2]> <;><reserve> <;><receipt transaction number[3]> <;><clients base info> <;><reserve> <0Ah>
Model	<1Dh><Model> where: <b>Model</b> – character 4Dh - 'M'	<'M'><country[2]> <;><encoding[1..20]> <;><device type [2]> <;><build No[2..4]> <;><build 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..2]> <;><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]> <;><article last number[5]> <;><total number of headers> <;><number of fiscal headers only> <;><number of footers> <0Ah>