

## DTDevices

Generated by Doxygen 1.8.3.1

Wed Apr 15 2015 17:57:51



# Contents

<b>1</b>	<b>Class Documentation</b>	<b>1</b>
1.1	FMP10_ROU Class Reference	1
1.1.1	Detailed Description	11
1.1.2	Method Documentation	12
1.1.2.1	addDelegate:	12
1.1.2.2	command101Variant0Version0OperatorCode:oldOperatorPassword:newOperatorPassword:error:	12
1.1.2.3	command102Variant0Version0OperatorCode:operatorPassword:operatorName:error:	12
1.1.2.4	command103Variant0Version0AndReturnError:	13
1.1.2.5	command105Variant0Version0AndReturnError:	14
1.1.2.6	command107Variant0Version0AndReturnError:	14
1.1.2.7	command107Variant10Version0TargetItemPlu:error:	15
1.1.2.8	command107Variant11Version0AndReturnError:	16
1.1.2.9	command107Variant12Version0TargetItemPlu:error:	17
1.1.2.10	command107Variant12Version1TargetItemPlu:error:	18
1.1.2.11	command107Variant1Version0TaxGroup:itemPlu:itemGroup:singleItemPrice:replaceQuantity:itemQuantity:itemName:error:	19
1.1.2.12	command107Variant2Version0ItemPlu:itemQuantity:error:	20
1.1.2.13	command107Variant3Version0ItemPlu:singleItemPrice:error:	21
1.1.2.14	command107Variant4Version0AndReturnError:	21
1.1.2.15	command107Variant4Version1ItemPlu:error:	22
1.1.2.16	command107Variant4Version2StartItemPlu:lastItemPlu:error:	23
1.1.2.17	command107Variant5Version0TargetItemPlu:error:	23
1.1.2.18	command107Variant6Version0TargetItemPlu:error:	25
1.1.2.19	command107Variant7Version0TargetItemPlu:error:	26
1.1.2.20	command107Variant8Version0AndReturnError:	27
1.1.2.21	command107Variant9Version0TargetItemPlu:error:	29
1.1.2.22	command108Variant0Version0ReportTypeOption:error:	30
1.1.2.23	command109Variant0Version0ReceiptCount:error:	31
1.1.2.24	command110Variant0Version0AndReturnError:	32

1.1.2.25	command111Variant0Version0PrintOption:startItemNumber:endItemNumber- :error: . . . . .	33
1.1.2.26	command111Variant1Version0PrintOption:startItemNumber:endItemNumber- :itemGroup:error: . . . . .	34
1.1.2.27	command111Variant2Version0PrintOption:error: . . . . .	35
1.1.2.28	command112Variant0Version0OperatorCode:error: . . . . .	36
1.1.2.29	command113Variant0Version0AndReturnError: . . . . .	37
1.1.2.30	command114Variant0Version0FiscalRecordNumber:error: . . . . .	37
1.1.2.31	command114Variant1Version0FiscalRecordNumber:error: . . . . .	39
1.1.2.32	command114Variant1Version1FiscalRecordNumber1:fiscalRecordNumber2- :error: . . . . .	40
1.1.2.33	command114Variant2Version0FiscalRecordNumber:error: . . . . .	41
1.1.2.34	command114Variant2Version1FiscalRecordNumber1:fiscalRecordNumber2- :error: . . . . .	42
1.1.2.35	command114Variant3Version0FiscalRecordNumber:error: . . . . .	43
1.1.2.36	command114Variant3Version1FiscalRecordNumber1:fiscalRecordNumber2- :error: . . . . .	44
1.1.2.37	command114Variant4Version0FiscalRecordNumber:error: . . . . .	46
1.1.2.38	command114Variant5Version0FiscalRecordNumber:error: . . . . .	47
1.1.2.39	command114Variant6Version0FiscalRecordNumber:error: . . . . .	48
1.1.2.40	command115Variant0Version0RowNumber:rowData:error: . . . . .	49
1.1.2.41	command115Variant1Version0RowNumber:error: . . . . .	49
1.1.2.42	command120Variant0Version0AndReturnError: . . . . .	50
1.1.2.43	command122Variant0Version0AndReturnError: . . . . .	50
1.1.2.44	command122Variant1Version0AndReturnError: . . . . .	51
1.1.2.45	command122Variant1Version1AndReturnError: . . . . .	51
1.1.2.46	command122Variant2Version0AndReturnError: . . . . .	52
1.1.2.47	command122Variant2Version1AndReturnError: . . . . .	52
1.1.2.48	command122Variant3Version0AndReturnError: . . . . .	53
1.1.2.49	command122Variant3Version1AndReturnError: . . . . .	53
1.1.2.50	command122Variant4Version0AndReturnError: . . . . .	53
1.1.2.51	command38Variant0Version0AndReturnError: . . . . .	54
1.1.2.52	command39Variant0Version0AndReturnError: . . . . .	55
1.1.2.53	command41Variant0Version0Switches:error: . . . . .	55
1.1.2.54	command41Variant0Version1AndReturnError: . . . . .	56
1.1.2.55	command42Variant0Version0InputText:error: . . . . .	56
1.1.2.56	command43Variant0Version0ItemIndex:dataValue:error: . . . . .	56
1.1.2.57	command44Variant0Version0TargetLines:error: . . . . .	58
1.1.2.58	command45Variant0Version0AndReturnError: . . . . .	59
1.1.2.59	command46Variant0Version0RotatedTextRow:error: . . . . .	59
1.1.2.60	command47Variant0Version0AndReturnError: . . . . .	60

1.1.2.61	command48Variant0Version0OperatorCode:operatorPassword:tillNumber:error: .	60
1.1.2.62	command49Variant0Version0TaxGroup:itemPrice:error: . . . . .	62
1.1.2.63	command49Variant0Version10TextRow1:taxGroup:itemPrice:itemQuantity:error:	63
1.1.2.64	command49Variant0Version11TextRow1:taxGroup:itemPrice:itemQuantity- :specialTax:error: . . . . .	64
1.1.2.65	command49Variant0Version12TextRow1:textRow2:taxGroup:itemPrice:error: . .	65
1.1.2.66	command49Variant0Version13TextRow1:textRow2:taxGroup:itemPrice:special- Tax:error: . . . . .	66
1.1.2.67	command49Variant0Version14TextRow1:textRow2:taxGroup:itemPrice:item- Quantity:error: . . . . .	67
1.1.2.68	command49Variant0Version15TextRow1:textRow2:taxGroup:itemPrice:item- Quantity:specialTax:error: . . . . .	68
1.1.2.69	command49Variant0Version1TaxGroup:itemPrice:specialTax:error: . . . . .	70
1.1.2.70	command49Variant0Version2TaxGroup:itemPrice:itemQuantity:error: . . . . .	71
1.1.2.71	command49Variant0Version3TaxGroup:itemPrice:itemQuantity:specialTax:error:	72
1.1.2.72	command49Variant0Version4TextRow2:taxGroup:itemPrice:error: . . . . .	73
1.1.2.73	command49Variant0Version5TextRow2:taxGroup:itemPrice:specialTax:error: . .	74
1.1.2.74	command49Variant0Version6TextRow2:taxGroup:itemPrice:itemQuantity:error: .	75
1.1.2.75	command49Variant0Version7TextRow2:taxGroup:itemPrice:itemQuantity- :specialTax:error: . . . . .	76
1.1.2.76	command49Variant0Version8TextRow1:taxGroup:itemPrice:error: . . . . .	78
1.1.2.77	command49Variant0Version9TextRow1:taxGroup:itemPrice:specialTax:error: . .	79
1.1.2.78	command49Variant1Version0TaxGroup:itemPrice:sellWithAbsoluteSumDiscount- :error: . . . . .	80
1.1.2.79	command49Variant1Version10TextRow1:taxGroup:itemPrice:itemQuantity:sell- WithAbsoluteSumDiscount:error: . . . . .	81
1.1.2.80	command49Variant1Version11TextRow1:taxGroup:itemPrice:itemQuantity- :specialTax:sellWithAbsoluteSumDiscount:error: . . . . .	82
1.1.2.81	command49Variant1Version12TextRow1:textRow2:taxGroup:itemPrice:sellWith- AbsoluteSumDiscount:error: . . . . .	83
1.1.2.82	command49Variant1Version13TextRow1:textRow2:taxGroup:itemPrice:special- Tax:sellWithAbsoluteSumDiscount:error: . . . . .	85
1.1.2.83	command49Variant1Version14TextRow1:textRow2:taxGroup:itemPrice:item- Quantity:sellWithAbsoluteSumDiscount:error: . . . . .	86
1.1.2.84	command49Variant1Version15TextRow1:textRow2:taxGroup:itemPrice:item- Quantity:specialTax:sellWithAbsoluteSumDiscount:error: . . . . .	87
1.1.2.85	command49Variant1Version1TaxGroup:itemPrice:specialTax:sellWithAbsolute- SumDiscount:error: . . . . .	89
1.1.2.86	command49Variant1Version2TaxGroup:itemPrice:itemQuantity:sellWithAbsolute- SumDiscount:error: . . . . .	90
1.1.2.87	command49Variant1Version3TaxGroup:itemPrice:itemQuantity:specialTax:sell- WithAbsoluteSumDiscount:error: . . . . .	91
1.1.2.88	command49Variant1Version4TextRow2:taxGroup:itemPrice:sellWithAbsolute- SumDiscount:error: . . . . .	92

1.1.2.89	command49Variant1Version5TextRow2:taxGroup:itemPrice:specialTax:sellWith-AbsoluteSumDiscount:error:	94
1.1.2.90	command49Variant1Version6TextRow2:taxGroup:itemPrice:itemQuantity:sell-WithAbsoluteSumDiscount:error:	95
1.1.2.91	command49Variant1Version7TextRow2:taxGroup:itemPrice:itemQuantity-:specialTax:sellWithAbsoluteSumDiscount:error:	96
1.1.2.92	command49Variant1Version8TextRow1:taxGroup:itemPrice:sellWithAbsolute-SumDiscount:error:	98
1.1.2.93	command49Variant1Version9TextRow1:taxGroup:itemPrice:specialTax:sellWith-AbsoluteSumDiscount:error:	99
1.1.2.94	command49Variant2Version0TaxGroup:itemPrice:sellWithPercentDiscount:error:	100
1.1.2.95	command49Variant2Version10TextRow1:taxGroup:itemPrice:itemQuantity:sell-WithPercentDiscount:error:	101
1.1.2.96	command49Variant2Version11TextRow1:taxGroup:itemPrice:itemQuantity-:specialTax:sellWithPercentDiscount:error:	102
1.1.2.97	command49Variant2Version12TextRow1:textRow2:taxGroup:itemPrice:sellWith-PercentDiscount:error:	104
1.1.2.98	command49Variant2Version13TextRow1:textRow2:taxGroup:itemPrice:special-Tax:sellWithPercentDiscount:error:	105
1.1.2.99	command49Variant2Version14TextRow1:textRow2:taxGroup:itemPrice:item-Quantity:sellWithPercentDiscount:error:	106
1.1.2.100	command49Variant2Version15TextRow1:textRow2:taxGroup:itemPrice:item-Quantity:specialTax:sellWithPercentDiscount:error:	108
1.1.2.101	command49Variant2Version1TaxGroup:itemPrice:specialTax:sellWithPercent-Discount:error:	109
1.1.2.102	command49Variant2Version2TaxGroup:itemPrice:itemQuantity:sellWithPercent-Discount:error:	110
1.1.2.103	command49Variant2Version3TaxGroup:itemPrice:itemQuantity:specialTax:sell-WithPercentDiscount:error:	112
1.1.2.104	command49Variant2Version4TextRow2:taxGroup:itemPrice:sellWithPercent-Discount:error:	113
1.1.2.105	command49Variant2Version5TextRow2:taxGroup:itemPrice:specialTax:sellWith-PercentDiscount:error:	114
1.1.2.106	command49Variant2Version6TextRow2:taxGroup:itemPrice:itemQuantity:sell-WithPercentDiscount:error:	115
1.1.2.107	command49Variant2Version7TextRow2:taxGroup:itemPrice:itemQuantity-:specialTax:sellWithPercentDiscount:error:	117
1.1.2.108	command49Variant2Version8TextRow1:taxGroup:itemPrice:sellWithPercent-Discount:error:	118
1.1.2.109	command49Variant2Version9TextRow1:taxGroup:itemPrice:specialTax:sellWith-PercentDiscount:error:	119
1.1.2.110	command50Variant0Version0StartDate:endDate:error:	120
1.1.2.111	command51Variant0Version0ToPrintOption:toDisplayOption:error:	121
1.1.2.112	command51Variant0Version1ToPrintOption:toDisplayOption:subtotalWith-PercentDiscount:error:	122
1.1.2.113	command51Variant0Version2ToPrintOption:toDisplayOption:subtotalWith-AbsoluteSumDiscount:error:	123

1.1.2.114	command53Variant0Version0PaidMode:amountIn:error:	124
1.1.2.115	command53Variant0Version1TextRow2:paidMode:amountIn:error:	126
1.1.2.116	command53Variant0Version2TextRow1:paidMode:amountIn:error:	128
1.1.2.117	command53Variant0Version4TextRow1:textRow2:paidMode:amountIn:error:	130
1.1.2.118	command53Variant1Version0AndReturnError:	132
1.1.2.119	command53Variant1Version1TextRow2:error:	133
1.1.2.120	command53Variant1Version2TextRow1:error:	135
1.1.2.121	command53Variant1Version3TextRow1:textRow2:error:	136
1.1.2.122	command54Variant0Version0TextIn:error:	137
1.1.2.123	command56Variant0Version0AndReturnError:	138
1.1.2.124	command58Variant0Version0SignPlu:itemQuantity:error:	138
1.1.2.125	command58Variant0Version1SignPlu:itemQuantity:specialTax:error:	139
1.1.2.126	command58Variant1Version0SignPlu:itemQuantity:sellWithAbsoluteSumDiscount: error:	141
1.1.2.127	command58Variant1Version1SignPlu:itemQuantity:specialTax:sellWithAbsolute- SumDiscount:error:	142
1.1.2.128	command58Variant2Version0SignPlu:itemQuantity:sellWithPercentDiscount- error:	143
1.1.2.129	command58Variant2Version1SignPlu:itemQuantity:specialTax:sellWithPercent- Discount:error:	145
1.1.2.130	command60Variant0Version0AndReturnError:	146
1.1.2.131	command61Variant0Version0TargetDate:targetTime:error:	146
1.1.2.132	command62Variant0Version0AndReturnError:	147
1.1.2.133	command64Variant0Version0AndReturnError:	147
1.1.2.134	command65Variant0Version0AndReturnError:	148
1.1.2.135	command68Variant0Version0AndReturnError:	149
1.1.2.136	command69Variant0Version0ReportTypeOption:error:	149
1.1.2.137	command70Variant0Version0AmountInOut:error:	150
1.1.2.138	command70Variant0Version1AndReturnError:	151
1.1.2.139	command71Variant0Version0AndReturnError:	152
1.1.2.140	command73Variant0Version0StartRecordNumber:endRecordNumber:error:	153
1.1.2.141	command74Variant0Version0AndReturnError:	153
1.1.2.142	command74Variant1Version0AndReturnError:	154
1.1.2.143	command76Variant0Version0AndReturnError:	155
1.1.2.144	command79Variant0Version0StartDate:endDate:error:	156
1.1.2.145	command80Variant0Version0SoundData:error:	156
1.1.2.146	command83Variant0Version0InputMultiplier:inputDecimals:inputCurrency:input- EnabledTaxesArray:inputTaxGroupA:inputTaxGroupB:inputTaxGroupC:inputTax- GroupD:error:	158
1.1.2.147	command83Variant1Version0AndReturnError:	160
1.1.2.148	command84Variant0Version0BarcodeType:barcodeData:error:	161
1.1.2.149	command84Variant0Version1BarcodeType:barcodeData:error:	163

1.1.2.150 command85Variant0Version0AdditionalPaymentTypeOption:inputAdditional-PaymentName:error: . . . . .	164
1.1.2.151 command85Variant0Version1AdditionalPaymentTypeOption:error: . . . . .	165
1.1.2.152 command86Variant0Version0AndReturnError: . . . . .	166
1.1.2.153 command87Variant0Version0AndReturnError: . . . . .	166
1.1.2.154 command90Variant0Version0AndReturnError: . . . . .	167
1.1.2.155 command90Variant0Version1AndReturnError: . . . . .	167
1.1.2.156 command94Variant0Version0StartDate:endDate:error: . . . . .	168
1.1.2.157 command95Variant0Version0StartFiscalRecordNumber:endFiscalRecord-Number:error: . . . . .	169
1.1.2.158 command97Variant0Version0AndReturnError: . . . . .	169
1.1.2.159 command99Variant0Version0AndReturnError: . . . . .	170
1.1.2.160 diagnosticInfoGetAndReturnError: . . . . .	170
1.1.2.161 diagnosticInfoPrintAndReturnError: . . . . .	171
1.1.2.162 getStatusBytesAndReturnError: . . . . .	171
1.1.2.163 nonFiscalReceiptCloseAndReturnError: . . . . .	172
1.1.2.164 nonFiscalReceiptOpenAndReturnError: . . . . .	172
1.1.2.165 nonFiscalReceiptPrintText:error: . . . . .	173
1.1.2.166 removeDelegate: . . . . .	173



# Chapter 1

## Class Documentation

### 1.1 FMP10\_ROU Class Reference

Provides universal access to all supported devices' functions.

Inherits NSObject.

#### Instance Methods

- (void) - [addDelegate:](#)  
*Allows unlimited delegates to be added to a single class instance.*
- (void) - [removeDelegate:](#)  
*Removes delegate, previously added with addDelegate.*
- (bool) - [connectWithStreams:outputStream:error:](#)  
*Connect to the fiscal device using streams.*
- (void) - [disconnect](#)  
*Disconnects from the stream.*
- (NSDictionary \*) - [command38Variant0Version0AndReturnError:](#)  
*26h (38) Opening a non-fiscal receipt.*
- (NSDictionary \*) - [command39Variant0Version0AndReturnError:](#)  
*27h (39) Closing a non-fiscal receipt*
- (bool) - [command41Variant0Version0Switches:error:](#)  
*29h (41) SET MEMORY SWITCHES*
- (bool) - [command41Variant0Version1AndReturnError:](#)  
*29h (41) SET MEMORY SWITCHES*
- (bool) - [command42Variant0Version0InputText:error:](#)  
*2Ah (42) PRINTING OF A FREE NON-FISCAL TEXT*
- (bool) - [command43Variant0Version0ItemIndex:dataValue:error:](#)  
*2Bh (43) SETTING FOOTER AND PRINTING OPTIONS*
- (bool) - [command44Variant0Version0TargetLines:error:](#)  
*2Ch(44) ADVANCING PAPER*
- (NSDictionary \*) - [command45Variant0Version0AndReturnError:](#)  
*2Dh (45) OPENING A RECEIPT FOR 90 DEGREES ROTATED TEXT*
- (bool) - [command46Variant0Version0RotatedTextRow:error:](#)  
*2Eh(46) PRINT 90 DEGREES ROTATED TEXT*
- (NSDictionary \*) - [command47Variant0Version0AndReturnError:](#)  
*2Fh(47) CLOSING A RECEIPT FOR 90 DEGREES ROTATED TEXT*
- (NSDictionary \*) - [command48Variant0Version0OperatorCode:operatorPassword:tillNumber:error:](#)

- 30h(48) OPENING A FISCAL CLIENT'S RECEIPT
- (bool) - [command49Variant0Version0TaxGroup:itemPrice:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version1TaxGroup:itemPrice:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version2TaxGroup:itemPrice:itemQuantity:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version3TaxGroup:itemPrice:itemQuantity:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version4TextRow2:taxGroup:itemPrice:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version5TextRow2:taxGroup:itemPrice:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version6TextRow2:taxGroup:itemPrice:itemQuantity:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version7TextRow2:taxGroup:itemPrice:itemQuantity:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version8TextRow1:taxGroup:itemPrice:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version9TextRow1:taxGroup:itemPrice:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version10TextRow1:taxGroup:itemPrice:itemQuantity:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version11TextRow1:taxGroup:itemPrice:itemQuantity:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version12TextRow1:textRow2:taxGroup:itemPrice:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version13TextRow1:textRow2:taxGroup:itemPrice:specialTax:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version14TextRow1:textRow2:taxGroup:itemPrice:itemQuantity:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant0Version15TextRow1:textRow2:taxGroup:itemPrice:itemQuantity:specialTax-error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version0TaxGroup:itemPrice:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version1TaxGroup:itemPrice:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version2TaxGroup:itemPrice:itemQuantity:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version3TaxGroup:itemPrice:itemQuantity:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version4TextRow2:taxGroup:itemPrice:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version5TextRow2:taxGroup:itemPrice:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES
- (bool) - [command49Variant1Version6TextRow2:taxGroup:itemPrice:itemQuantity:sellWithAbsoluteSumDiscount:error:](#)

- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version7TextRow2:taxGroup:itemPrice:itemQuantity:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version8TextRow1:taxGroup:itemPrice:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version9TextRow1:taxGroup:itemPrice:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version10TextRow1:taxGroup:itemPrice:itemQuantity:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version11TextRow1:taxGroup:itemPrice:itemQuantity:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version12TextRow1:textRow2:taxGroup:itemPrice:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version13TextRow1:textRow2:taxGroup:itemPrice:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version14TextRow1:textRow2:taxGroup:itemPrice:itemQuantity:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant1Version15TextRow1:textRow2:taxGroup:itemPrice:itemQuantity:specialTax:sellWithAbsoluteSumDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version0TaxGroup:itemPrice:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version1TaxGroup:itemPrice:specialTax:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version2TaxGroup:itemPrice:itemQuantity:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version3TaxGroup:itemPrice:itemQuantity:specialTax:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version4TextRow2:taxGroup:itemPrice:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version5TextRow2:taxGroup:itemPrice:specialTax:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version6TextRow2:taxGroup:itemPrice:itemQuantity:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version7TextRow2:taxGroup:itemPrice:itemQuantity:specialTax:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version8TextRow1:taxGroup:itemPrice:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

  - (bool) - [command49Variant2Version9TextRow1:taxGroup:itemPrice:specialTax:sellWithPercentDiscount:error:](#)
- 31h(49) REGISTRATION OF SALES

- (bool) - [command49Variant2Version10TextRow1:taxGroup:itemPrice:itemQuantity:sellWithPercentDiscount:error:](#)  
*31h(49) REGISTRATION OF SALES*
- (bool) - [command49Variant2Version11TextRow1:taxGroup:itemPrice:itemQuantity:specialTax:sellWithPercentDiscount:error:](#)  
*31h(49) REGISTRATION OF SALES*
- (bool) - [command49Variant2Version12TextRow1:textRow2:taxGroup:itemPrice:sellWithPercentDiscount:error:](#)  
*31h(49) REGISTRATION OF SALES*
- (bool) - [command49Variant2Version13TextRow1:textRow2:taxGroup:itemPrice:specialTax:sellWithPercentDiscount:error:](#)  
*31h(49) REGISTRATION OF SALES*
- (bool) - [command49Variant2Version14TextRow1:textRow2:taxGroup:itemPrice:itemQuantity:sellWithPercentDiscount:error:](#)  
*31h(49) REGISTRATION OF SALES*
- (bool) - [command49Variant2Version15TextRow1:textRow2:taxGroup:itemPrice:itemQuantity:specialTax:sellWithPercentDiscount:error:](#)  
*31h(49) REGISTRATION OF SALES*
- (NSDictionary \*) - [command50Variant0Version0StartDate:endDate:error:](#)  
*32h(50) TAX RATES ENTERED DURING THE ACCOUNTED PERIOD*
- (NSDictionary \*) - [command51Variant0Version0ToPrintOption:toDisplayOption:error:](#)  
*33h(51) SUBTOTAL*
- (NSDictionary \*) - [command51Variant0Version1ToPrintOption:toDisplayOption:subtotalWithPercentDiscount:error:](#)  
*33h(51) SUBTOTAL*
- (NSDictionary \*) - [command51Variant0Version2ToPrintOption:toDisplayOption:subtotalWithAbsoluteSumDiscount:error:](#)  
*33h(51) SUBTOTAL*
- (NSDictionary \*) - [command53Variant0Version0PaidMode:amountIn:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant0Version1TextRow2:paidMode:amountIn:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant0Version2TextRow1:paidMode:amountIn:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant0Version4TextRow1:textRow2:paidMode:amountIn:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant1Version0AndReturnError:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant1Version1TextRow2:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant1Version2TextRow1:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (NSDictionary \*) - [command53Variant1Version3TextRow1:textRow2:error:](#)  
*35h(53) CALCULATION OF A TOTAL*
- (bool) - [command54Variant0Version0TextIn:error:](#)  
*36h(54) PRINTING A FREE FISCAL TEXT*
- (NSDictionary \*) - [command56Variant0Version0AndReturnError:](#)  
*38h(56) CLOSING A FISCAL RECEIPT*
- (bool) - [command58Variant0Version0SignPlu:itemQuantity:error:](#)  
*3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM*
- (bool) - [command58Variant0Version1SignPlu:itemQuantity:specialTax:error:](#)  
*3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM*

- (bool) - [command58Variant1Version0SignPlu:itemQuantity:sellWithAbsoluteSumDiscount:error:](#)  
*3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM*
- (bool) - [command58Variant1Version1SignPlu:itemQuantity:specialTax:sellWithAbsoluteSumDiscount:error:](#)  
:  
*3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM*
- (bool) - [command58Variant2Version0SignPlu:itemQuantity:sellWithPercentDiscount:error:](#)  
*3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM*
- (bool) - [command58Variant2Version1SignPlu:itemQuantity:specialTax:sellWithPercentDiscount:error:](#)  
*3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM*
- (bool) - [command60Variant0Version0AndReturnError:](#)  
*3Ch(60) CANCEL FISCAL RECEIPT*
- (bool) - [command61Variant0Version0TargetDate:targetTime:error:](#)  
*3Dh(61) SETTING THE CLOCK - DATE AND HOUR*
- (NSDictionary \*) - [command62Variant0Version0AndReturnError:](#)  
*3Eh (62) READING CURRENT DATE AND HOUR*
- (NSDictionary \*) - [command64Variant0Version0AndReturnError:](#)  
*40h(64) LAST FISCAL CLOSURE DETAILS*
- (NSDictionary \*) - [command65Variant0Version0AndReturnError:](#)  
*41h(65) DAILY TOTALS*
- (NSDictionary \*) - [command68Variant0Version0AndReturnError:](#)  
*44h (68) THE NUMBER OF FREE FIELDS IN THE FISCAL MEMORY*
- (NSDictionary \*) - [command69Variant0Version0ReportTypeOption:error:](#)  
*45h(69) DAILY FINANCIAL REPORT*
- (NSDictionary \*) - [command70Variant0Version0AmountInOut:error:](#)  
*46h(70) INTERNAL DEBITING AND CREDITING (service In and Out)*
- (NSDictionary \*) - [command70Variant0Version1AndReturnError:](#)  
*46h(70) INTERNAL DEBITING AND CREDITING (read only)*
- (bool) - [command71Variant0Version0AndReturnError:](#)  
*47h(71) PRINTING DIAGNOSTIC INFORMATION*
- (bool) - [command73Variant0Version0StartRecordNumber:endRecordNumber:error:](#)  
*49H (73) DETAILED FISCAL MEMORY REPORT BY CLOSURE NUMBER*
- (NSDictionary \*) - [command74Variant0Version0AndReturnError:](#)  
*4Ah(74) READING THE STATUS BYTES*
- (NSDictionary \*) - [command74Variant1Version0AndReturnError:](#)  
*4Ah(74) READING THE STATUS BYTES*
- (NSDictionary \*) - [command76Variant0Version0AndReturnError:](#)  
*4Ch(76) STATUS OF THE FISCAL TRANSACTION*
- (bool) - [command79Variant0Version0StartDate:endDate:error:](#)  
*4Fh(79) SHORT FISCAL MEMORY REPORT BY CLOSURE DATE*
- (bool) - [command80Variant0Version0SoundData:error:](#)  
*50h(80) SOUND SIGNAL*
- (NSDictionary \*) - [command83Variant0Version0InputMultiplier:inputDecimals:inputCurrency:inputEnabled-TaxesArray:inputTaxGroupA:inputTaxGroupB:inputTaxGroupC:inputTaxGroupD:error:](#)  
*53h(83) SETTING THE MULTIPLIER, DECIMALS, CURRENCY NAME AND DISABLED TAXES*
- (NSDictionary \*) - [command83Variant1Version0AndReturnError:](#)  
*53h(83) SETTING THE MULTIPLIER, DECIMALS, CURRENCY NAME AND DISABLED TAXES*
- (bool) - [command84Variant0Version0BarcodeType:barcodeData:error:](#)  
*54h(84) PRINTING A BAR CODE*
- (bool) - [command84Variant0Version1BarcodeType:barcodeData:error:](#)  
*54h(84) PRINTING A BAR CODE*

- (NSDictionary \*) - [command85Variant0Version0AdditionalPaymentTypeOption:inputAdditionalPayment-Name:error:](#)  
*55H(85) DEFINE ADDITIONAL PAYMENT TYPES NAME*
- (NSDictionary \*) - [command85Variant0Version1AdditionalPaymentTypeOption:error:](#)  
*55H(85) DEFINE ADDITIONAL PAYMENT TYPES NAME*
- (NSDictionary \*) - [command86Variant0Version0AndReturnError:](#)  
*56H(86) GET LATEST FISCAL MEMORY RECORD DATE*
- (NSDictionary \*) - [command87Variant0Version0AndReturnError:](#)  
*57H(87) GET SHIFT LENGTH*
- (NSDictionary \*) - [command90Variant0Version0AndReturnError:](#)  
*5Ah(90) RETURNS DIAGONSTIC INFORMATION*
- (NSDictionary \*) - [command90Variant0Version1AndReturnError:](#)  
*5Ah(90) RETURNS DIAGONSTIC INFORMATION*
- (bool) - [command94Variant0Version0StartDate:endDate:error:](#)  
*5Eh(94) DETAILED FISCAL MEMORY REPORT BY CLOSURE DATE*
- (bool) - [command95Variant0Version0StartFiscalRecordNumber:endFiscalRecordNumber:error:](#)  
*5Fh(95) SHORT FISCAL MEMORY REPORT BY CLOSURE NUMBER*
- (NSDictionary \*) - [command97Variant0Version0AndReturnError:](#)  
*61h(97) READING THE SET TAX RATES*
- (NSDictionary \*) - [command99Variant0Version0AndReturnError:](#)  
*63h(99) Reading the TAX REGISTRATION NUMBER*
- (bool) - [command101Variant0Version0OperatorCode:oldOperatorPassword:newOperatorPassword:error:](#)  
*65h(101) SETTING THE OPERATOR'S PASSWORD*
- (bool) - [command102Variant0Version0OperatorCode:operatorPassword:operatorName:error:](#)  
*66h(102) ENTERING OPERATOR'S NAME*
- (NSDictionary \*) - [command103Variant0Version0AndReturnError:](#)  
*67h(103) INFORMATION ON THE CURRENT RECEIPT*
- (bool) - [command105Variant0Version0AndReturnError:](#)  
*68h(105) OPERATOR'S REPORT*
- (NSDictionary \*) - [command107Variant0Version0AndReturnError:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant1Version0TaxGroup:itemPlu:itemGroup:singleItemPrice:replace-Quantity:itemQuantity:itemName:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant2Version0ItemPlu:itemQuantity:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant3Version0ItemPlu:singleItemPrice:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant4Version0AndReturnError:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant4Version1ItemPlu:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant4Version2StartItemPlu:lastItemPlu:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant5Version0TargetItemPlu:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant6Version0TargetItemPlu:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant7Version0TargetItemPlu:error:](#)  
*6Bh(107) DEFINING AND READING ITEMS*
- (NSDictionary \*) - [command107Variant8Version0AndReturnError:](#)

- 6Bh(107) DEFINING AND READING ITEMS
- (NSDictionary \*) - [command107Variant9Version0TargetItemPlu:error:](#)  
6Bh(107) DEFINING AND READING ITEMS
- (NSDictionary \*) - [command107Variant10Version0TargetItemPlu:error:](#)  
6Bh(107) DEFINING AND READING ITEMS
- (NSDictionary \*) - [command107Variant11Version0AndReturnError:](#)  
6Bh(107) DEFINING AND READING ITEMS
- (NSDictionary \*) - [command107Variant12Version0TargetItemPlu:error:](#)  
6Bh(107) DEFINING AND READING ITEMS
- (NSDictionary \*) - [command107Variant12Version1TargetItemPlu:error:](#)  
6Bh(107) DEFINING AND READING ITEMS
- (NSDictionary \*) - [command108Variant0Version0ReportTypeOption:error:](#)  
6Ch (108) EXTENDED DAILY FINANCIAL REPORT
- (bool) - [command109Variant0Version0ReceiptCount:error:](#)  
6Dh(109) PRINTING A DUPLICATE RECEIPT
- (NSDictionary \*) - [command110Variant0Version0AndReturnError:](#)  
6Eh(110) ADDITIONAL DAILY INFORMATION
- (NSDictionary \*) - [command111Variant0Version0PrintOption:startItemNumber:endItemNumber:error:](#)  
6Fh(111) ITEMS REPORT
- (NSDictionary \*) - [command111Variant1Version0PrintOption:startItemNumber:endItemNumber:itemGroup:error:](#)  
6Fh(111) ITEMS REPORT
- (NSDictionary \*) - [command111Variant2Version0PrintOption:error:](#)  
6Fh(111) ITEMS REPORT
- (NSDictionary \*) - [command112Variant0Version0OperatorCode:error:](#)  
70h(112) READING INFORMATION ON THE OPERATOR
- (NSDictionary \*) - [command113Variant0Version0AndReturnError:](#)  
71h(113) READING THE NUMBER OF THE LAST PRINTED DOCUMENT
- (NSDictionary \*) - [command114Variant0Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant1Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant1Version1FiscalRecordNumber1:fiscalRecordNumber2:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant2Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant2Version1FiscalRecordNumber1:fiscalRecordNumber2:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant3Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant3Version1FiscalRecordNumber1:fiscalRecordNumber2:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant4Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant5Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (NSDictionary \*) - [command114Variant6Version0FiscalRecordNumber:error:](#)  
72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD
- (bool) - [command115Variant0Version0RowNumber:rowData:error:](#)  
73h(115) PROGRAMMING A GRAPHIC LOGO
- (NSDictionary \*) - [command115Variant1Version0RowNumber:error:](#)



- 73h(115) PROGRAMMING A GRAPHIC LOGO*
- (NSDictionary \*) - [command120Variant0Version0AndReturnError:](#)
- 78h(120) SWITCHING THE PRINTER OFF*
- (NSDictionary \*) - [command122Variant0Version0AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (NSDictionary \*) - [command122Variant1Version0AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (NSDictionary \*) - [command122Variant1Version1AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (bool) - [command122Variant2Version0AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (bool) - [command122Variant2Version1AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (bool) - [command122Variant3Version0AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (bool) - [command122Variant3Version1AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (bool) - [command122Variant4Version0AndReturnError:](#)
- 7Ah(122) JOURNAL SUPPORT*
- (NSString \*) - **customCommand:data:error:**
- (NSDictionary \*) - [diagnosticInfoGetAndReturnError:](#)
- diagnosticInfoGetAndReturnError - RETURNS DIAGNOSTIC INFORMATION*
- (bool) - [diagnosticInfoPrintAndReturnError:](#)
- diagnosticInfoPrintAndReturnError - PRINTING DIAGNOSTIC INFORMATION*
- (NSDictionary \*) - [getStatusBytesAndReturnError:](#)
- getStatusBytesAndReturnError - READING THE STATUS BYTES*
- (NSDictionary \*) - [nonFiscalReceiptOpenAndReturnError:](#)
- nonFiscalReceiptOpenAndReturnError - Opening a non-fiscal receipt.*
- (bool) - [nonFiscalReceiptPrintText:error:](#)
- nonFiscalReceiptPrintText - PRINTING OF A FREE NON-FISCAL TEXT*
- (NSDictionary \*) - [nonFiscalReceiptCloseAndReturnError:](#)
- nonFiscalReceiptCloseAndReturnError - Closing a non-fiscal receipt*
- (NSDictionary \*) - **fiscalReceiptOpenAndReturnError:**
- (bool) - **fiscalReceiptPrintText:error:**
- (bool) - **fiscalReceiptSellAndReturnError:**
- (NSDictionary \*) - **fiscalReceiptSubtotalAndReturnError:**
- (NSDictionary \*) - **fiscalReceiptTotalAndReturnError:**
- (NSDictionary \*) - **fiscalReceiptCloseAndReturnError:**
- (bool) - **checkAndResolveAndReturnError:**

## Class Methods

- (id) + **sharedInstance**



## Properties

- id [delegate](#)  
*Adds delegate to the class.*
- NSMutableArray \* [delegates](#)  
*Provides a list of currently registered delegates.*
- bool **deviceConnected**
- int **infoPrinterCodePage**
- int **infoMajorNumberSDKVersion**
- int **infoMinorNumberSDKVersion**
- int **infoReleaseNumberSDKVersion**
- int **infoBuildNumberSDKVersion**
- int **infoMaxLogoHigh**
- int **infoMaxLogoWidth**
- int **infoMaxHeaderLinesCount**
- int **infoMaxFooterLinesCount**
- int **infoMaxTransactionsCountInFiscalReceipt**
- int **infoMaxSymbolCountInSellTextRow1**
- int **infoMaxSymbolCountInSellTextRow2**
- int **infoMaxSymbolCountInNonFiscalText**
- int **infoMaxSymbolCountInNonFiscalRotatedText**
- int **infoMaxSymbolCountInFiscalText**
- int **infoMaxDepartmentCount**
- int **infoMaxItemsCount**
- int **infoTaxRatesMaxCount**
- int **infoDisplayCodePage**
- int **infoLastClassErrorCode**
- bool **infoMandatoryZReportEndOfDay**
- bool **infoMandatoryEKLPprintBeforeZReport**
- bool **infoMandatoryEKLSave**
- bool **infoMandatoryMonthlyReport**
- bool **infoMandatoryYearlyReport**
- bool **statusDisplayNotConnected**
- bool **statusTransparentDisplayMode**
- bool **statusFiscalMemoryMissing**
- bool **statusPrintingHeadOverheated**
- bool **statusSeconRollNoPaperPlace**
- bool **statusSeconRollOutOfPaper**
- bool **statusEndOfEKL**
- bool **statusDrawerOpened**
- bool **statusEKLNotEmpty**
- bool **statusEKLPrinted**
- bool **statusEKLNearEnd**
- bool **statusKLENNearEnd**
- bool **statusSecondRollNotEnoughPaper**
- bool **statusAutomaticPaperCutting**
- bool **statusPrintingHeadNotConnected**
- bool **supportEKL**
- bool **supportNRATerminal**
- bool **supportKLEN**
- bool **supportEIK**
- bool **supportServiceContractsInformation**
- bool **supportIOSANumber**
- bool **supportFiscalReceipts**

- bool **supportRotatedFiscalReceipts**
- bool **supportNonFiscalReceipts**
- bool **supportRotatedNonFiscalReceipts**
- bool **supportBluetooth**
- bool **supportWiFi**
- bool **supportTCPIP**
- bool **supportFTP**
- bool **supportHTTP**
- bool **supportSoftwareSwitches**
- bool **supportClientDisplay**
- bool **supportDrawerOpening**
- bool **supportDrawerStatus**
- bool **supportSecondRoll**
- bool **supportAutoCutPaper**
- bool **supportAdditionalMatrixPrint**
- bool **supportPrintingHeadTemperatureControl**
- bool **supportAsynchronousMode**
- bool **supportTransactionsBufferForCommands**
- bool **supportReceiptVoid**
- bool **supportSaleRowVoid**
- bool **supportVoidReceipt**
- bool **supportVoidSale**
- bool **statusEIKSet**
- bool **statusSerialNumberSet**
- bool **statusFMNumberSet**
- bool **statusPrinterFiscalized**
- bool **statusFiscalMemoryFormated**
- bool **statusTaxRatesOk**
- bool **statusLowBattery**
- bool **statusGeneralErrorType1**
- bool **statusGeneralErrorType2**
- bool **statusPrintingHeadFailure**
- bool **statusPrinterClockNotSet**
- bool **statusInvalidCommand**
- bool **statusSyntaxError**
- bool **statusNRATerminalNotRespond**
- bool **statusRamError**
- bool **statusRamCleared**
- bool **statusCommandNotAllowed**
- bool **statusFieldOverflow**
- bool **statusFiscalMemoryFull**
- bool **statusFiscalMemoryReadError**
- bool **statusFiscalMemoryWriteError**
- bool **statusFiscalMemoryReadOnly**
- bool **statusOutOfPaper**
- bool **statusEndOfKLEN**
- bool **statusFiscalReceiptOpened**
- bool **statusNonFiscalReceiptOpened**
- bool **statusRotatedReceiptOpened**
- bool **statusFiscalMemoryNearEnd**
- bool **statusCoverWasOpened**
- bool **statusNotEnoughPaper**
- NSString \* **infoTaxEnabledArray**
- NSString \* **infoServiceEIKNumber**
- NSString \* **infoServiceEndDate**

- NSString \* **infoPrinterName**
- NSString \* **infoFirmwareRevision**
- NSString \* **infoFirmwareDateTime**
- NSString \* **infoSerialNumber**
- NSString \* **infoFiscalModuleNumber**
- bool **infoBluetoothDiscoverable**
- NSString \* **infoMACAddress**
- NSString \* **infoIPAddress**
- NSString \* **infoIOSANumber**
- NSString \* **infoTaxArray**
- NSString \* **infoLastErrorText**
- NSString \* **sellParameterTextRow1**
- NSString \* **sellParameterTextRow2**
- NSString \* **sellParameterTaxGroup**
- NSString \* **sellParameterSpecialTax**
- NSString \* **sellParameterPrice**
- NSString \* **sellParameterQuantity**
- NSString \* **sellParameterPLU**
- NSString \* **sellParameterDepartment**
- NSString \* **sellParameterPercent**
- NSString \* **sellParameterAbsoluteSum**
- NSString \* **sellParameterOperatorCode**
- NSString \* **sellParameterOperatorPassword**
- NSString \* **sellParameterOperatorTillNumber**
- NSString \* **subtotalParameterToPrint**
- NSString \* **subtotalParameterToDisplay**
- NSString \* **subtotalParameterPercent**
- NSString \* **subtotalParameterAbsoluteSum**
- NSString \* **totalParameterTextRow1**
- NSString \* **totalParameterTextRow2**
- NSString \* **totalParameterPaidMode**
- NSString \* **totalParameterAmount**
- NSData \* **statusBytes**

### 1.1.1 Detailed Description

Provides universal access to all supported devices' functions.

In order to use one of the supported accessories in your program, several steps have to be performed:

- Include DTDevices.h and libdtdev.a in your project.
- Go to Frameworks and add ExternalAccessory framework
- Edit your program plist file, add new element and select "Supported external accessory protocols" from the list, then add the protocol names of the accessories you want to connect to:  
 For Linea series: com.datecs.linea.pro.msr and com.datecs.linea.pro.bar  
 For Pinpad: com.datecs.iserial.communication and com.datecs.ppad  
 For iSerial: com.datecs.iserial.communication  
 For ESC/POS printers: com.datecs.printer.escpos

Since this SDK is based on features, the specific device is not that important, for example, if your program relies on barcode scanning, then Linea, Pinpad or the ESC/POS printers can provide that functionality, so you can include all their protocols.

## 1.1.2 Method Documentation

### 1.1.2.1 - (void) addDelegate: (id) newDelegate

Allows unlimited delegates to be added to a single class instance.

This is useful in the case of global class and every view can use addDelegate when the view is shown and removeDelegate when no longer needs to monitor events

#### Parameters

<i>newDelegate</i>	the delegate that will be notified of Linea events
--------------------	--

### 1.1.2.2 - (bool) command101Variant0Version0OperatorCode: (NSString \*) operatorCode oldOperatorPassword:(NSString \*) oldOperatorPassword newOperatorPassword:(NSString \*) newOperatorPassword error:(NSError \*\*) error

65h(101) SETTING THE OPERATOR'S PASSWORD

- Data field:
  1. operatorCode Operator's code (1 to 16)
  2. oldOperatorPassword Old password (4 to 8 digits)
  3. newOperatorPassword New password (4 to 8 digits)
- Response:
  1. None

Sets one of the 16 operator's passwords, which will be demanded upon opening a fiscal receipt. After three erroneous password entries, the printer will block, it must then be switched OFF and ON again to continue operating. After initialization or reset of the operational memory, all 16 passwords are "0000".

#### Parameters

<i>operatorCode</i>	- Operator's code (1 to 16)
<i>oldOperatorPassword</i>	- Old password (4 to 8 digits)
<i>newOperatorPassword</i>	- New password (4 to 8 digits)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

### 1.1.2.3 - (bool) command102Variant0Version0OperatorCode: (NSString \*) operatorCode operatorPassword:(NSString \*) operatorPassword operatorName:(NSString \*) operatorName error:(NSError \*\*) error

66h(102) ENTERING OPERATOR'S NAME

- Data field:
  1. OpCode Operator's code (1 to 16)
  2. Pwd Password (4 to 8 digits)

3. OpName Name of the operator (up to 30 symbols)

- Response:

1. None

Enters one of the 16 operator names. The number and name of the operator are printed at the beginning of each fiscal (clients) receipt. After three erroneous password entries, the printer will block, it must then be switched OFF and ON again to continue operating. After initialization or reset of the operational memory, all 16 password locations are empty.

#### Parameters

<i>operatorCode</i>	- Operator's code (1 to 16)
<i>operator-Password</i>	- Password (4 to 8 digits)
<i>operatorName</i>	- Name of the operator (up to 30 symbols)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

#### 1.1.2.4 - (NSDictionary \*) command103Variant0Version0AndReturnError: (NSError \*\*) error

67h(103) INFORMATION ON THE CURRENT RECEIPT

- Data field: None
- Response:
  1. saleVoidIsPossible Possible/impossible return (sale registration with a negative sign) ['0' / '1']
  2. taxGroupA The sum accumulated under tax A
  3. taxGroupB The sum accumulated under tax B
  4. taxGroupC The sum accumulated under tax C
  5. taxGroupD The sum accumulated under tax D
  6. taxGroupE The sum accumulated under tax E
  7. SpecTax The special tax sum

The command offers information on sums accumulated so far under the different tax groups and whether it is possible to return the registered items sold.

#### Returns

NSDictionary

- KeyValue - @"saleVoidIsPossible" - Possible/impossible return (sale registration with a negative sign) ['0' / '1']
- KeyValue - @"taxGroupA" - The sum accumulated under tax A
- KeyValue - @"taxGroupB" - The sum accumulated under tax B
- KeyValue - @"taxGroupC" - The sum accumulated under tax C
- KeyValue - @"taxGroupD" - The sum accumulated under tax D
- KeyValue - @"taxGroupE" - The sum accumulated under tax E
- KeyValue - @"specialTax" - The special tax sum

### 1.1.2.5 - (bool) command105Variant0Version0AndReturnError: (NSError \*\*) error

#### 68h(105) OPERATOR'S REPORT

- Data field: None
- Response: None

Information on the sales, performed by the operators, is printed out where for each separate operator the following data is printed out: name, individual number, number of fiscal receipts, discharges made, surcharge, sum adjustments and accumulated total sums.

#### Returns

TRUE upon success, FALSE otherwise

### 1.1.2.6 - (NSDictionary \*) command107Variant0Version0AndReturnError: (NSError \*\*) error

#### 6Bh(107) DEFINING AND READING ITEMS

- Data field:
  1. None
- Response:
  1. errorCode - One byte, showing the result from the operation and having the following meaning:
    - (a) 'P' Successful command
    - (b) 'F' Unsuccessful command
  2. maximumItemCount Total programmable article count (13000 for this printer).
  3. currentItemCount Programmed article count.
  4. maximumItemNameLength Maximal article name length (36 for this printer).

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

##### NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- KeyValue - @"maximumItemCount" - Total programmable article count (13000 for this printer).
- KeyValue - @"currentItemCount" - Programmed article count.
- KeyValue - @"maximumItemNameLength" - Maximal article name length (36 for this printer).

1.1.2.7 - (NSDictionary \*) command107Variant10Version0TargetItemPlu: (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the sold item with the greatest number.

- Input data:
  1. *targetItemPlu*
- Responce:
  1. *errorCode*
  2. *itemPlu*
  3. *taxGroup*
  4. *itemGroup*
  5. *singleItemPrice*
  6. *accumulatedSum*
  7. *soldItemQuantity*
  8. *availableItemQuantity*
  9. *itemName*
- *targetItemPlu* Individual number of the item. 9 digits (000000001 to 999999999)
- *errorCode* One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command
- *itemPlu* Individual number of the item. 9 digits (000000001 to 999999999)
- *taxGroup* Tax group - 1 byte
- *itemGroup* Article group. 2 digits (01 - 99).
- *singleItemPrice* Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- *accumulatedSum* Accumulated sum for this article.
- *soldItemQuantity* Accumulated quantity - a floating-point number with 3 decimal places.
- *availableItemQuantity* Available quantity of this article.
- *itemName* The name of the item. Up to 36 symbols.

Note: If the parameter PLU is present, then the first sold article with number lower than or equal to PLU is returned. If missing, PLU=999999999 is assumed.

#### Parameters

<i>targetItemPlu</i>	- Individual number of the item. 9 digits (000000001 to 999999999)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- KeyValue - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)
- KeyValue - @"taxGroup" - Tax group - 1 byte
- KeyValue - @"itemGroup" - Article group. 2 digits (01 - 99).
- KeyValue - @"singleItemPrice" - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- KeyValue - @"accumulatedSum" - Accumulated sum for this article.
- KeyValue - @"soldItemQuantity" - Accumulated quantity - a floating-point number with 3 decimal places.
- KeyValue - @"availableItemQuantity" - Available quantity of this article.
- KeyValue - @"itemName" - The name of the item. Up to 36 symbols.

#### 1.1.2.8 - (NSDictionary \*) command107Variant11Version0AndReturnError: (NSError \*\*) error

##### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the next found sold item. Depending of the starting command, the articles are enumerated in ascending or descending order.

- Input data:
  1. targetItemPlu
- Responce:
  1. errorCode
  2. itemPlu
  3. taxGroup
  4. itemGroup
  5. singleItemPrice
  6. accumulatedSum
  7. soldItemQuantity
  8. availableItemQuantity
  9. itemName
- targetItemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command
- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- taxGroup Tax group - 1 byte



- `itemGroup` Article group. 2 digits (01 - 99).
- `singleItemPrice` Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- `accumulatedSum` Accumulated sum for this article.
- `soldItemQuantity` Accumulated quantity - a floating-point number with 3 decimal places.
- `availableItemQuantity` Available quantity of this article.
- `itemName` The name of the item. Up to 36 symbols.

Note: The process of reading has ended with the last available item.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

##### NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- Key-Value - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)
- Key-Value - @"taxGroup" - Tax group - 1 byte
- Key-Value - @"itemGroup" - Article group. 2 digits (01 - 99).
- Key-Value - @"singleItemPrice" - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- Key-Value - @"accumulatedSum" - Accumulated sum for this article.
- Key-Value - @"soldItemQuantity" - Accumulated quantity - a floating-point number with 3 decimal places.
- Key-Value - @"availableItemQuantity" - Available quantity of this article.
- Key-Value - @"itemName" - The name of the item. Up to 36 symbols.

**1.1.2.9** - (NSDictionary \*) `command107Variant12Version0TargetItemPlu:` (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the first free item.

- Input data: `targetItemPlu`
- Response: `errorCode`
  1. `itemPlu`
- `errorCode` One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)

If the parameter PLU is present, then the first free (not programmed) article with number greater than or equal to PLU is returned. If missing, PLU=1 is assumed.

#### Parameters

<i>targetItemPlu</i>	- Individual number of the item. 9 digits (000000001 to 999999999)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- Key-Value - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)

1.1.2.10 - (NSDictionary \*) command107Variant12Version1TargetItemPlu: (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the last free item.

- Input data: targetItemPlu
- Response: errorCode
  1. itemPlu
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command
- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)

If the parameter PLU is present, then the first free (not programmed) article with number lower than or equal to PLU is returned. If missing, PLU=999999999 is assumed.

#### Parameters

<i>targetItemPlu</i>	- Individual number of the item. 9 digits (000000001 to 999999999)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command

- 'F' Unsuccessful command
- KeyValue - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)

1.1.2.11 - (NSDictionary \*) command107Variant1Version0TaxGroup: (NSString \*) *taxGroup* itemPlu:(NSString \*) *itemPlu* itemGroup:(NSString \*) *itemGroup* singleItemPrice:(NSString \*) *singleItemPrice* replaceQuantity:(NSString \*) *replaceQuantity* itemQuantity:(NSString \*) *itemQuantity* itemName:(NSString \*) *itemName* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

- Input data:
  1. taxGroup
  2. itemPlu
  3. itemGroup
  4. singleItemPrice
  5. replaceQuantity
  6. itemQuantity
  7. itemName
- Response:
  1. ErrorCode
- taxGroup Tax group. One byte ('A', 'B', 'C', 'D' or 'E').
- itemPlu Number of the item (1 to 999999999)
- itemGroup Article group (1 - 99).
- singleItemPrice Singular price - up to 8 meaningful digits.
- replaceQuantity A non-mandatory parameter - one byte with value 'A'. Changes the meaning of the next parameter (Quantity).
- itemQuantity A number with up to 3 decimals - the available quantity of the article. If Replace is present, then the available quantity is replaced with this parameter, otherwise it is added to the old value (if the article is already programmed, of course). Every sale command of this article will decrease this value.
- itemName Name of the item - up to 36 bytes.
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Up to 13000 different items may be programmed and the command will be rejected if a similar item has already been programmed in the memory of printer and sales of this item have been registered.

#### Parameters

<i>taxGroup</i>	- Tax group. One byte ('A', 'B', 'C', 'D' or 'E').
<i>itemPlu</i>	- Number of the item (1 to 999999999)
<i>itemGroup</i>	- Article group (1 - 99).
<i>singleItemPrice</i>	- Singular price - up to 8 meaningful digits.
<i>replaceQuantity</i>	- A non-mandatory parameter - one byte with value 'A'. Changes the meaning of the next parameter (Quantity).

<i>itemQuantity</i>	- A number with up to 3 decimals - the available quantity of the article. If Replace is present, then the available quantity is replaced with this parameter, otherwise it is added to the old value (if the article is already programmed, of course). Every sale command of this article will decrease this value.
<i>itemName</i>	- Name of the item - up to 36 bytes.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
- 'P' Successful command
- 'F' Unsuccessful command

1.1.2.12 - (NSDictionary \*) command107Variant2Version0ItemPlu: (NSString \*) *itemPlu* itemQuantity:(NSString \*) *itemQuantity* error:(NSError \*\*) *error*

## 6Bh(107) DEFINING AND READING ITEMS

- Input data:
  1. *itemPlu*
  2. *itemQuantity*
- Response:
  1. ErrorCode
- *itemPlu* Article number (1 to 999999999).
- *itemQuantity* Quantity correction - a floating-point number with 3 decimal places. Positive number increases the available quantity, negative decreases it.
- *errorCode* One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Changing the quantity is possible, if the article is programmed.

## Parameters

<i>itemPlu</i>	- Article number (1 to 999999999).
<i>itemQuantity</i>	- Quantity correction - a floating-point number with 3 decimal places. Positive number increases the available quantity, negative decreases it.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:

- 'P' Successful command
- 'F' Unsuccessful command

1.1.2.13 - (NSDictionary \*) command107Variant3Version0ItemPlu: (NSString \*) *itemPlu* singleItemPrice:(NSString \*) *singleItemPrice* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Change the price of an item

- Input data:
  1. *itemPlu*
  2. *singleItemPrice*
- Response:
  1. *ErrorCode*
- *itemPlu* Article number (1 to 999999999).
- *singleItemPrice* Singular price - up to 8 meaningful digits.
- *errorCode* One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Changing the price is possible, if the article is programmed and no sales of this article are made in the fiscal receipt (if a fiscal receipt is open).

#### Parameters

<i>itemPlu</i>	- Article number (1 to 999999999).
<i>singleItemPrice</i>	- Singular price - up to 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command

1.1.2.14 - (NSDictionary \*) command107Variant4Version0AndReturnError: (NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Delete all items with non-zero accumulated sums.

- Input data:
  1. None
- Response:

## 1. ErrorCode

- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

NOTE: Delete all items with non-zero accumulated sums.

## Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

## Returns

NSDictionary

- Key/Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command

1.1.2.15 - (NSDictionary \*) command107Variant4Version1ItemPlu: (NSString \*) *itemPlu* error:(NSError \*\*) *error*

6Bh(107) DEFINING AND READING ITEMS

- Input data:
  1. itemPlu
- Response:
  1. errorCode
- itemPlu Deletes article with selected number if there are no accumulated sums.
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Note: Deletes article with selected number if there are no accumulated sums.

## Parameters

<i>itemPlu</i>	- Deletes article with selected number if there are no accumulated sums.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- Key/Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command

- 'F' Unsuccessful command

1.1.2.16 - (NSDictionary \*) command107Variant4Version2StartItemPlu: (NSString \*) *startItemPlu* lastItemPlu:(NSString \*) *lastItemPlu* error:(NSError \*\*) *error*

6Bh(107) DEFINING AND READING ITEMS

- Input data:
  1. startItemPlu
  2. lastItemPlu
- Response:
  1. errorCode
- startItemPlu,lastItemPlu Deletes the articles within a set interval which do not have accumulated sums.
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Note: Deletes the articles within a set interval which do not have accumulated sums.

#### Parameters

<i>startItemPlu</i>	- Starting number
<i>lastItemPlu</i>	- Last number (End of the interval)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
- 'P' Successful command
- 'F' Unsuccessful command

1.1.2.17 - (NSDictionary \*) command107Variant5Version0TargetItemPlu: (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

6Bh(107) DEFINING AND READING ITEMS

Reading Item data

- Input data:
  1. targetItemPlu
- Response:
  1. errorCode
  2. itemPlu
  3. taxGroup

- 4. `itemGroup`
- 5. `singleItemPrice`
- 6. `accumulatedSum`
- 7. `soldItemQuantity`
- 8. `availableItemQuantity`
- 9. `itemName`

- `targetItemPlu` Individual number of the item. 9 digits (000000001 to 999999999)
- `errorCode` One byte, showing the result from the operation and having the following meaning:
  - 1. 'P' Successful command
  - 2. 'F' Unsuccessful command
- `itemPlu` Individual number of the item. 9 digits (000000001 to 999999999)
- `taxGroup` Tax group - 1 byte
- `itemGroup` Article group. 2 digits (01 - 99).
- `singleItemPrice` Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- `accumulatedSum` Accumulated sum for this article.
- `soldItemQuantity` Accumulated quantity - a floating-point number with 3 decimal places.
- `availableItemQuantity` Available quantity of this article.
- `itemName` The name of the item. Up to 36 symbols.

If the item cannot be found, one 'F' byte is returned.

#### Parameters

<i>targetItemPlu</i>	- Individual number of the item. 9 digits (000000001 to 999999999)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

##### NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- Key-Value - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)
- Key-Value - @"taxGroup" - Tax group - 1 byte
- Key-Value - @"itemGroup" - Article group. 2 digits (01 - 99).
- Key-Value - @"singleItemPrice" - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- Key-Value - @"accumulatedSum" - Accumulated sum for this article.
- Key-Value - @"soldItemQuantity" - Accumulated quantity - a floating-point number with 3 decimal places.



- KeyValue - @"availableItemQuantity" - Available quantity of this article.
- KeyValue - @"itemName" - The name of the item. Up to 36 symbols.

1.1.2.18 - (NSDictionary \*) command107Variant6Version0TargetItemPlu: (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

6Bh(107) DEFINING AND READING ITEMS

Returning the data on the first found programmed item.

- Input data:
  1. targetItemPlu
- Responce:
  1. errorCode
  2. itemPlu
  3. taxGroup
  4. itemGroup
  5. singleItemPrice
  6. accumulatedSum
  7. soldItemQuantity
  8. availableItemQuantity
  9. itemName
- targetItemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command
- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- taxGroup Tax group - 1 byte
- itemGroup Article group. 2 digits (01 - 99).
- singleItemPrice Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- accumulatedSum Accumulated sum for this article.
- soldItemQuantity Accumulated quantity - a floating-point number with 3 decimal places.
- availableItemQuantity Available quantity of this article.
- itemName The name of the item. Up to 36 symbols.

Note: If the parameter PLU is present, then the first programmed article with number greater than or equal to PLU is returned. If missing, PLU=1 is assumed. The returned data is similar to the: "command107Variant5Version0"

#### Parameters

<i>targetItemPlu</i>	- Individual number of the item. 9 digits (000000001 to 999999999)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- KeyValue - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)
- KeyValue - @"taxGroup" - Tax group - 1 byte
- KeyValue - @"itemGroup" - Article group. 2 digits (01 - 99).
- KeyValue - @"singleItemPrice" - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- KeyValue - @"accumulatedSum" - Accumulated sum for this article.
- KeyValue - @"soldItemQuantity" - Accumulated quantity - a floating-point number with 3 decimal places.
- KeyValue - @"availableItemQuantity" - Available quantity of this article.
- KeyValue - @"itemName" - The name of the item. Up to 36 symbols.

1.1.2.19 - (NSDictionary \*) command107Variant7Version0TargetItemPlu: (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the programmed item with the greatest number.

- Input data:
  1. targetItemPlu
- Response:
  1. errorCode
  2. itemPlu
  3. taxGroup
  4. itemGroup
  5. singleItemPrice
  6. accumulatedSum
  7. soldItemQuantity
  8. availableItemQuantity
  9. itemName
- targetItemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command
- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- taxGroup Tax group - 1 byte

- `itemGroup` Article group. 2 digits (01 - 99).
- `singleItemPrice` Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- `accumulatedSum` Accumulated sum for this article.
- `soldItemQuantity` Accumulated quantity - a floating-point number with 3 decimal places.
- `availableItemQuantity` Available quantity of this article.
- `itemName` The name of the item. Up to 36 symbols.

Note: If the parameter `PLU` is present, then the first programmed article with number lower than or equal to `PLU` is returned. If missing, `PLU=999999999` is assumed.

#### Parameters

<code>targetItemPlu</code>	- Individual number of the item. 9 digits (000000001 to 999999999)
<code>error</code>	pointer to <code>NSError</code> object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

`NSDictionary`

- Key-Value - `@("errorCode")` - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- Key-Value - `@("itemPlu")` - Individual number of the item. 9 digits (000000001 to 999999999)
- Key-Value - `@("taxGroup")` - Tax group - 1 byte
- Key-Value - `@("itemGroup")` - Article group. 2 digits (01 - 99).
- Key-Value - `@("singleItemPrice")` - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- Key-Value - `@("accumulatedSum")` - Accumulated sum for this article.
- Key-Value - `@("soldItemQuantity")` - Accumulated quantity - a floating-point number with 3 decimal places.
- Key-Value - `@("availableItemQuantity")` - Available quantity of this article.
- Key-Value - `@("itemName")` - The name of the item. Up to 36 symbols.

#### 1.1.2.20 - (NSDictionary \*) command107Variant8Version0AndReturnError: (NSError \*\*) error

##### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the next found programmed item. Depending of the starting command, the articles are enumerated in ascending or descending order.

- Input data:
  1. `targetItemPlu`
- Response:
  1. `errorCode`

- 2. itemPlu
- 3. taxGroup
- 4. itemGroup
- 5. singleItemPrice
- 6. accumulatedSum
- 7. soldItemQuantity
- 8. availableItemQuantity
- 9. itemName

- targetItemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- errorCode One byte, showing the result from the operation and having the following meaning:
  - 1. 'P' Successful command
  - 2. 'F' Unsuccessful command
- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- taxGroup Tax group - 1 byte
- itemGroup Article group. 2 digits (01 - 99).
- singleItemPrice Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- accumulatedSum Accumulated sum for this article.
- soldItemQuantity Accumulated quantity - a floating-point number with 3 decimal places.
- availableItemQuantity Available quantity of this article.
- itemName The name of the item. Up to 36 symbols.

Note: The last three commands are used to receive a list of programmed items. The subcommand 'F' or 'L' is followed by 'N' until the response 'F' comes. This means that the process of reading has ended with the last available item.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

##### NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- KeyValue - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)
- KeyValue - @"taxGroup" - Tax group - 1 byte
- KeyValue - @"itemGroup" - Article group. 2 digits (01 - 99).
- KeyValue - @"singleItemPrice" - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).

- KeyValue - @"accumulatedSum" - Accumulated sum for this article.
- KeyValue - @"soldItemQuantity" - Accumulated quantity - a floating-point number with 3 decimal places.
- KeyValue - @"availableItemQuantity" - Available quantity of this article.
- KeyValue - @"itemName" - The name of the item. Up to 36 symbols.

1.1.2.21 - (NSDictionary \*) command107Variant9Version0TargetItemPlu: (NSString \*) *targetItemPlu* error:(NSError \*\*) *error*

#### 6Bh(107) DEFINING AND READING ITEMS

Returning the data on the first sold item.

- Input data:
  1. targetItemPlu
- Responce:
  1. errorCode
  2. itemPlu
  3. taxGroup
  4. itemGroup
  5. singleItemPrice
  6. accumulatedSum
  7. soldItemQuantity
  8. availableItemQuantity
  9. itemName
- targetItemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command
- itemPlu Individual number of the item. 9 digits (000000001 to 999999999)
- taxGroup Tax group - 1 byte
- itemGroup Article group. 2 digits (01 - 99).
- singleItemPrice Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- accumulatedSum Accumulated sum for this article.
- soldItemQuantity Accumulated quantity - a floating-point number with 3 decimal places.
- availableItemQuantity Available quantity of this article.
- itemName The name of the item. Up to 36 symbols.

Note: If the parameter PLU is present, then the first sold article with number greater than or equal to PLU is returned. If missing, PLU=1 is assumed.

#### Parameters

<i>targetItemPlu</i>	- Individual number of the item. 9 digits (000000001 to 999999999)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

### NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command
- Key-Value - @"itemPlu" - Individual number of the item. 9 digits (000000001 to 999999999)
- Key-Value - @"taxGroup" - Tax group - 1 byte
- Key-Value - @"itemGroup" - Article group. 2 digits (01 - 99).
- Key-Value - @"singleItemPrice" - Singular price. A floating-point number - decimal places depend on the count set using command 83 (53h).
- Key-Value - @"accumulatedSum" - Accumulated sum for this article.
- Key-Value - @"soldItemQuantity" - Accumulated quantity - a floating-point number with 3 decimal places.
- Key-Value - @"availableItemQuantity" - Available quantity of this article.
- Key-Value - @"itemName" - The name of the item. Up to 36 symbols.

1.1.2.22 - (NSDictionary \*) command108Variant0Version0ReportTypeOption: (NSString \*) *reportTypeOption* error:(NSError \*\*) *error*

## 6Ch (108) EXTENDED DAILY FINANCIAL REPORT

- Data field:
  1. reportTypeOption
- Response:
  1. fiscalRecordNumber
  2. totalSumForTheDay
  3. totalSumInTaxGroupA,
  4. totalSumInTaxGroupB,
  5. totalSumInTaxGroupC,
  6. totalSumInTaxGroupD,
  7. totalSumInTaxGroupE,
  8. totalSumInSpecialTax
- reportTypeOption Parameter controlling the type of generated report.
  1. '0' A Z-report is printed. The printout ends with inscriptions "NON-FISCAL RECEIPT" if the printer is not fiscalised.
  2. '2' A X-report is generated, i.e., no entry into the fiscal memory is made and no closures are performed. The printout ends with inscription "NON-FISCAL RECEIPT". The same actions may be generated directly from the printer if during switching on the "FEED" button is hold for 2 to 4 seconds.

3. N The presence of this symbol at the end of the data cancels the option to clear the data accumulated on the operators during a Z-report.
4. A The presence of this symbol at the end of the data cancels the option to clear the data about sold article quantities during a Z-report.

- `fiscalRecordNumber` Fiscal closure (Daily report) number - 4 bytes.
- `totalSumForTheDay` The sum of all sales for the day - 12 bytes with a sign.
- `totalSumInTaxGroupX` The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- `totalSumInSpecialTax` Special tax sum (12 bytes with sign).

The command with option '0' (Z-report) must be executed immediately after printing and deleting the electronic journal. If there is information in the journal, the command is not permitted. In the beginning is printed a list of the sold articles with PLU's less or equal than 100.

#### Parameters

<i>reportType-Option</i>	- Parameter controlling the type of generated report. <ul style="list-style-type: none"> <li>• '0' A Z-report is printed. The printout ends with inscriptions "NON-FISCAL RECEIPT" if the printer is not fiscalised.</li> <li>• '2' A X-report is generated, i.e., no entry into the fiscal memory is made and no closures are performed. The printout ends with inscription "NON-FISCAL RECEIPT". The same actions may be generated directly from the printer if during switching on the &lt;FEED&gt; button is hold for 2 to 4 seconds.</li> <li>• N The presence of this symbol at the end of the data cancels the option to clear the data accumulated on the operators during a Z-report.</li> <li>• A The presence of this symbol at the end of the data cancels the option to clear the data about sold article quantities during a Z-report.</li> </ul>
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"fiscalRecordNumber" - Fiscal closure (Daily report) number - 4 bytes.
- Key-Value - @"totalSumForTheDay" - The sum of all sales for the day - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupA" - The total under tax category - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupB" - The total under tax category - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupC" - The total under tax category - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupD" - The total under tax category - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupE" - The total under tax category - 12 bytes with a sign.
- Key-Value - @"totalSumInSpecialTax" - Special tax sum (12 bytes with sign).

1.1.2.23 - (bool) `command109Variant0Version0ReceiptCount:` (NSString \*) *receiptCount* `error:(NSError **) error`

6Dh(109) PRINTING A DUPLICATE RECEIPT

- Data field: `receiptCount`
- Response: None

- `receiptCount` Number of duplicate receipts (only a value of 1 or 2 is accepted!).

The command initiates the printing of a copy of the last closed receipt containing registered sales. Immediately after the tax registration number the inscription "DUPLICATE" is printed out in bold letters. The printer will refuse to execute this command twice.

#### Parameters

<i>receiptCount</i>	- Number of duplicate receipts (only a value of 1 or 2 is accepted!).
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.24 - (NSDictionary \*) `command110Variant0Version0AndReturnError: (NSError **) error`

6Eh(110) ADDITIONAL DAILY INFORMATION

- Data field: None
- Response:
  1. `paidInCash`
  2. `paidWithCreditCart`
  3. `paidWithDebitCard`
  4. `paidWithCheque`
  5. `paidInAdditionalPaymentType1`
  6. `paidInAdditionalPaymentType2`
  7. `paidInAdditionalPaymentType3`
  8. `paidInAdditionalPaymentType4`
  9. `currentFiscalRecordNumber`
  10. `nextFiscalReceiptNumber`
- `paidInCash` Paid in cash
- `paidWithCreditCart` Payment credited
- `paidWithDebitCard` Paid with a debit card
- `paidWithCheque` Paid with a cheque
- `paidInAdditionalPaymentTypeX` Paid with one of the additional payment types ('I', 'J', 'K', 'L').
- `currentFiscalRecordNumber` Current (last) fiscal entry
- `nextFiscalReceiptNumber` Number of the next fiscal receipt

Returns information on distribution of the daily sum according to terms of payment used.



## Returns

## NSDictionary

- KeyValue - @"paidInCash" - Paid in cash
- KeyValue - @"paidWithCreditCard" - Payment credited
- KeyValue - @"paidWithDebitCard" - Paid with a debit card
- KeyValue - @"paidWithCheque" - Paid with a cheque
- KeyValue - @"paidInAdditionalPaymentType1" - Payd with one of the additional payment types ('I').
- KeyValue - @"paidInAdditionalPaymentType2" - Payd with one of the additional payment types ('J').
- KeyValue - @"paidInAdditionalPaymentType3" - Payd with one of the additional payment types ('K').
- KeyValue - @"paidInAdditionalPaymentType4" - Payd with one of the additional payment types ('L').
- KeyValue - @"currentFiscalRecordNumber" - Current (last) fiscal entry
- KeyValue - @"nextFiscalReceiptNumber" - Number of the next fiscal receipt

1.1.2.25 - (NSDictionary \*) command111Variant0Version0PrintOption: (NSString \*) *printOption* startItemNumber:(NSString \*) *startItemNumber* endItemNumber:(NSString \*) *endItemNumber* error:(NSError \*\*) *error*

## 6Fh(111) ITEMS REPORT

Data field:

1. printOption

2. startItemNumber

3. endItemNumber Reponse:

4. errorCode

- printOption Defines the type of information under print. Possible values:
  1. - 'S' Only sold items are printed out. The data on these items include: the individual number, VAT group, group, name, single price, sold quantity and total sum for the day.
  2. - 'P' All programmed items are printed out, containing their number, VAT group, group, name, sold quantity, available quantity and single price.
- startItemNumber First article number (PLU) printed. PLUs less than this are not included in the report. Default: 1.
- endItemNumber Last article number (PLU) printed. PLUs greater than this are not included in the report. Default: 999999999.
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Items are arranged according to their individual numbers. When a Z-report is printed, then the accumulated sums are cleared, if the parameter 'A' is not present in the command line.

## Parameters

<i>printOption</i>	- Defines the type of information under print. Possible values: <ul style="list-style-type: none"> <li>- 'S' Only sold items are printed out. The data on these items include: the individual number, VAT group, group, name, single price, sold quantity and total sum for the day.</li> <li>- 'P' All programmed items are printed out, containing their number, VAT group, group, name, sold quantity, available quantity and single price.</li> </ul>
<i>startItemNumber</i>	- First article number (PLU) printed. PLUs less than this are not included in the report. Default: 1.
<i>endItemNumber</i>	- Last article number (PLU) printed. PLUs greater than this are not included in the report. Default: 999999999.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
- 'P' Successful command
- 'F' Unsuccessful command

1.1.2.26 - (NSDictionary \*) command111Variant1Version0PrintOption: (NSString \*) *printOption* startItemNumber:(NSString \*) *startItemNumber* endItemNumber:(NSString \*) *endItemNumber* itemGroup:(NSString \*) *itemGroup* error:(NSError \*\*) *error*

## 6Fh(111) ITEMS REPORT

Data field:

- *printOption* Defines the type of information under print. Possible values:
  1. - 'S' Only sold items are printed out. The data on these items include: the individual number, VAT group, group, name, single price, sold quantity and total sum for the day.
  2. - 'P' All programmed items are printed out, containing their number, VAT group, group, name, sold quantity, available quantity and single price.
- *startItemNumber* First article number (PLU) printed. PLUs less than this are not included in the report. Default: 1.
- *endItemNumber* Last article number (PLU) printed. PLUs greater than this are not included in the report. Default: 999999999.
- *itemGroup* A number from 1 to 99. If present, only articles from this group are printed, otherwise all articles are printed.
- *errorCode* One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command Reponse:

errorCode

Items are arranged according to their individual numbers. When a Z-report is printed, then the accumulated sums are cleared, if the parameter 'A' is not present in the command line.

## Parameters

<i>printOption</i>	- Defines the type of information under print. Possible values: <ul style="list-style-type: none"> <li>• 'S' Only sold items are printed out. The data on these items include: the individual number, VAT group, group, name, single price, sold quantity and total sum for the day.</li> <li>• 'P' All programmed items are printed out, containing their number, VAT group, group, name, sold quantity, available quantity and single price.</li> </ul>
<i>startItemNumber</i>	- First article number (PLU) printed. PLUs less than this are not included in the report. Default: 1.
<i>endItemNumber</i>	- Last article number (PLU) printed. PLUs greater than this are not included in the report. Default: 999999999.
<i>itemGroup</i>	- A number from 1 to 99. If present, only articles from this group are printed, otherwise all articles are printed.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'P' Successful command
  - 'F' Unsuccessful command

1.1.2.27 - (NSDictionary \*) command111Variant2Version0PrintOption: (NSString \*) *printOption* error:(NSError \*\*) *error*

6Fh(111) ITEMS REPORT

Data field:

1. printOption Reponse:

2. errorCode

- printOption Defines the type of information under print. Possible values:
  1. - 'S' Only sold items are printed out. The data on these items include: the individual number, VAT group, group, name, single price, sold quantity and total sum for the day.
  2. - 'P' All programmed items are printed out, containing their number, VAT group, group, name, sold quantity, available quantity and single price.
- errorCode One byte, showing the result from the operation and having the following meaning:
  1. 'P' Successful command
  2. 'F' Unsuccessful command

Items are arranged according to their individual numbers. When a Z-report is printed, then the accumulated sums are cleared, if the parameter 'A' is not present in the command line.

### Parameters

<i>printOption</i>	- Defines the type of information under print. Possible values: <ul style="list-style-type: none"> <li>• - 'S' Only sold items are printed out. The data on these items include: the individual number, VAT group, group, name, single price, sold quantity and total sum for the day.</li> <li>• - 'P' All programmed items are printed out, containing their number, VAT group, group, name, sold quantity, available quantity and single price.</li> </ul>
Generated on Wed Apr 15 2015 17:57:51 for DTDevices by Doxygen	
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
- 'P' Successful command
- 'F' Unsuccessful command

**1.1.2.28** - (NSDictionary \*) command112Variant0Version0OperatorCode: (NSString \*) *operatorCode* error:(NSError \*\*) *error*

70h(112) READING INFORMATION ON THE OPERATOR

- Data field:
  1. operatorCode
- Response:
  1. receiptCountForOperator
  2. registeredSalesCountForOperator
  3. totalAccumulatedSum
  4. discountCountForOperator
  5. totalDiscounts
  6. surchargeCountForOperator
  7. totalSurcharges
  8. voidCountForOperator
  9. totalVoidSum
  10. operatorName
  11. operatorPassword
- operatorCode Number of the operator (1 to 16)
- receiptCountForOperator Number of fiscal receipts, issued by the operator
- registeredSalesCountForOperator Number of registered sales
- totalAccumulatedSum Total accumulated sum
- discountCountForOperator Number of discounts
- totalDiscounts Total number of discounts
- surchargeCountForOperator Number of surcharges
- totalSurcharges Total number of surcharges made
- voidCountForOperator Number of voids (and corrections of sums)
- totalVoidSum Total sum of the voids
- operatorName Name of the operator
- operatorPassword Operators password. Present only if the printer is in service mode.

The command leads to the reading of the available information, which will be printed out in the operator's report. The sums are returned as floating-point numbers incorporating the currently set number of decimal places.

**Parameters**

<i>operatorCode</i>	- Number of the operator (1 to 16)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- Key-Value - @"receiptCountForOperator" - Number of fiscal receipts, issued by the operator
- Key-Value - @"registeredSalesCountForOperator" - Number of registered sales
- Key-Value - @"totalAccumulatedSum" - Total accumulated sum
- Key-Value - @"discountCountForOperator" - Number of discounts
- Key-Value - @"totalDiscounts" - Total number of discounts
- Key-Value - @"surchargeCountForOperator" - Number of surcharges
- Key-Value - @"totalSurcharges" - Total number of surcharges made
- Key-Value - @"voidCountForOperator" - Number of voids (and corrections of sums)
- Key-Value - @"totalVoidSum" - Total sum of the voids
- Key-Value - @"operatorName" - Name of the operator
- Key-Value - @"operatorPassword" - Operators password. Present only if the printer is in service mode.

**1.1.2.29** - (NSDictionary \*) *command113Variant0Version0AndReturnError: (NSError \*\*) error*

71h(113) READING THE NUMBER OF THE LAST PRINTED DOCUMENT

- Data field: None
- Response: lastDocumentNumber

lastDocumentNumber The number of the last issued document (7 digits)

## Returns

NSDictionary

- Key-Value - @"lastDocumentNumber" - The number of the last issued document (7 digits)

**1.1.2.30** - (NSDictionary \*) *command114Variant0Version0FiscalRecordNumber: (NSString \*) fiscalRecordNumber error:(NSError \*\*) error*

72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

The command returns information on different tax groups for each separate entry and or a selected period of time. Periodic references for longer time periods may take few seconds to process.

- Data field: fiscalRecordNumber
- Response:
  1. errorCode
  2. lastTaxRatesRecordNumber
  3. decimalsCount
  4. enabledTaxArray

5. taxPercentRateA
6. taxPercentRateB
7. taxPercentRateC
8. taxPercentRateD
9. dateAndTime

- fiscalRecordNumber Number of the fiscal memory record.
- errorCode One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- lastTaxRatesRecordNumber Last (active) decimals and VAT rates record number.
- decimalsCount Decimals count for this Z-report record.
- enabledTaxArray Enabled VAT rates mask - 4 bytes with values '0' or '1', where '1' means "enabled".
- taxPercentRateX VAT rate for the corresponding VAT group in percents.
- dateAndTime Date and time of the data in format: DD-MM-YY hh:mm:ss.

#### Parameters

<i>fiscalRecord-Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- KeyValue - @"lastTaxRatesRecordNumber" - Last (active) decimals and VAT rates record number.
- KeyValue - @"decimalsCount" - Decimals count for this Z-report record.
- KeyValue - @"enabledTaxArray" - Enabled VAT rates mask - 4 bytes with values '0' or '1', where '1' means "enabled".
- KeyValue - @"taxPercentRateA" - VAT rate for the corresponding VAT group in percents.
- KeyValue - @"taxPercentRateB" - VAT rate for the corresponding VAT group in percents.
- KeyValue - @"taxPercentRateC" - VAT rate for the corresponding VAT group in percents.
- KeyValue - @"taxPercentRateD" - VAT rate for the corresponding VAT group in percents.
- KeyValue - @"dateAndTime" - Date and time of the data in format: DD-MM-YY hh:mm:ss.

1.1.2.31 - (NSDictionary \*) *command114Variant1Version0FiscalRecordNumber:* (NSString \*) *fiscalRecordNumber*  
*error:(NSError \*\*) error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about total (turnover) sums for the Z-report record. Periodic references for longer time periods may take few seconds to process.

- Data field: *fiscalRecordNumber*
- Response:
  1. *errorCode*
  2. *outputFiscalRecordNumber*
  3. *fiscalReceiptsCount*
  4. *totalTurnoverInTaxGroupA*
  5. *totalTurnoverInTaxGroupB*
  6. *totalTurnoverInTaxGroupC*
  7. *totalTurnoverInTaxGroupD*
  8. *totalTurnoverInTaxGroupE*
  9. *specialTax*
- *fiscalRecordNumber* Number of the fiscal memory record.
- *errorCode* One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- *outputFiscalRecordNumber* Z-report record number.
- *fiscalReceiptsCount* Fiscal receipts count for the day.
- *totalTurnoverInTaxGroupX* Total (turnover) sum for the corresponding VAT group.
- *specialTax* Special tax accumulated sum.

#### Parameters

<i>fiscalRecord-Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- Key-Value - @"outputFiscalRecordNumber" - Z-report record number.
- Key-Value - @"fiscalReceiptsCount" - Fiscal receipts count for the day.

- KeyValue - @"totalTurnoverInTaxGroupA" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupB" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupC" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupD" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupE" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"specialTax" - Special tax accumulated sum.

1.1.2.32 - (NSDictionary \*) command114Variant1Version1FiscalRecordNumber1: (NSString \*) *fiscalRecordNumber1* fiscalRecordNumber2:(NSString \*) *fiscalRecordNumber2* error:(NSError \*\*) *error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about total (turnover) sums for the Z-report record(interval). Periodic references for longer time periods may take few seconds to process.

- Data field: fiscalRecordNumber1
  1. fiscalRecordNumber2
- Response:
  1. errorCode
  2. outputFiscalRecordNumber
  3. fiscalReceiptsCount
  4. totalTurnoverInTaxGroupA
  5. totalTurnoverInTaxGroupB
  6. totalTurnoverInTaxGroupC
  7. totalTurnoverInTaxGroupD
  8. totalTurnoverInTaxGroupE
  9. specialTax
- fiscalRecordNumberX Number of the fiscal memory record.
- errorCode One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- outputFiscalRecordNumber Z-report record number.
- fiscalReceiptsCount Fiscal receipts count for the day.
- totalTurnoverInTaxGroupX Total (turnover) sum for the corresponding VAT group.
- specialTax Special tax accumulated sum.

Note: The data returned is for the period with starting record Closure1 and last record Closure2 for references.

#### Parameters

<i>fiscalRecord-Number1</i>	- Number of the fiscal memory record.
<i>fiscalRecord-Number2</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case of function fails. You can pass nil if you don't want that information



## Returns

## NSDictionary

- KeyValue - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- KeyValue - @"outputFiscalRecordNumber" - Z-report record number.
- KeyValue - @"fiscalReceiptsCount" - Fiscal receipts count for the day.
- KeyValue - @"totalTurnoverInTaxGroupA" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupB" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupC" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupD" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"totalTurnoverInTaxGroupE" - Total (turnover) sum for the corresponding VAT group.
- KeyValue - @"specialTax" - Special tax accumulated sum.

1.1.2.33 - (NSDictionary \*) command114Variant2Version0FiscalRecordNumber: (NSString \*) *fiscalRecordNumber* error:(NSError \*\*) *error*

## 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about net sums for the Z-report record. Periodic references for longer time periods may take few seconds to process.

- Data field: fiscalRecordNumber
- Response:
  1. errorCode
  2. outputFiscalRecordNumber
  3. fiscalReceiptsCount
  4. netAmountInTaxGroupA
  5. netAmountInTaxGroupB
  6. netAmountInTaxGroupC
  7. netAmountInTaxGroupD
  8. netAmountInTaxGroupE
  9. specialTax
- fiscalRecordNumber Number of the fiscal memory record.
- errorCode One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- outputFiscalRecordNumber Z-report record number.
- fiscalReceiptsCount Fiscal receipts count for the day.
- netAmountInTaxGroupX Net sum for the corresponding VAT group.
- specialTax Special tax accumulated sum.

## Parameters

<i>fiscalRecord-Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

NSDictionary

- KeyValue - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- KeyValue - @"outputFiscalRecordNumber" - Z-report record number.
- KeyValue - @"fiscalReceiptsCount" - Fiscal receipts count for the day.
- KeyValue - @"netAmountInTaxGroupA" - Net sum for the corresponding VAT group.
- KeyValue - @"netAmountInTaxGroupB" - Net sum for the corresponding VAT group.
- KeyValue - @"netAmountInTaxGroupC" - Net sum for the corresponding VAT group.
- KeyValue - @"netAmountInTaxGroupD" - Net sum for the corresponding VAT group.
- KeyValue - @"netAmountInTaxGroupE" - Net sum for the corresponding VAT group.
- KeyValue - @"specialTax" - Special tax accumulated sum.

1.1.2.34 - (NSDictionary \*) command114Variant2Version1FiscalRecordNumber1: (NSString \*) *fiscalRecordNumber1* fiscalRecordNumber2:(NSString \*) *fiscalRecordNumber2* error:(NSError \*\*) *error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about net sums for the Z-report record(period). Periodic references for longer time periods may take few seconds to process.

- Data field: fiscalRecordNumber1
  1. fiscalRecordNumber2
- Response:
  1. errorCode
  2. outputFiscalRecordNumber
  3. fiscalReceiptsCount
  4. netAmountInTaxGroupA
  5. netAmountInTaxGroupB
  6. netAmountInTaxGroupC
  7. netAmountInTaxGroupD
  8. netAmountInTaxGroupE
  9. specialTax
- fiscalRecordNumberX Number of the fiscal memory record.

- `errorCode` One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- `outputFiscalRecordNumber` Z-report record number.
- `fiscalReceiptsCount` Fiscal receipts count for the day.
- `netAmountInTaxGroupX` Net sum for the corresponding VAT group.
- `specialTax` Special tax accumulated sum.

#### Parameters

<i>fiscalRecord-Number1</i>	- Number of the fiscal memory record.
<i>fiscalRecord-Number2</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- Key-Value - @"outputFiscalRecordNumber" - Z-report record number.
- Key-Value - @"fiscalReceiptsCount" - Fiscal receipts count for the day.
- Key-Value - @"netAmountInTaxGroupA" - Net sum for the corresponding VAT group.
- Key-Value - @"netAmountInTaxGroupB" - Net sum for the corresponding VAT group.
- Key-Value - @"netAmountInTaxGroupC" - Net sum for the corresponding VAT group.
- Key-Value - @"netAmountInTaxGroupD" - Net sum for the corresponding VAT group.
- Key-Value - @"netAmountInTaxGroupE" - Net sum for the corresponding VAT group.
- Key-Value - @"specialTax" - Special tax accumulated sum.

**1.1.2.35** - (NSDictionary \*) `command114Variant3Version0FiscalRecordNumber:` (NSString \*) *fiscalRecordNumber*  
 error:(NSError \*\*) *error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about VAT sums for the Z-report record. Periodic references for longer time periods may take few seconds to process.

- Data field: `fiscalRecordNumber`
- Response:

1. `errorCode`
2. `outputFiscalRecordNumber`
3. `fiscalReceiptsCount`
4. `vatSumInTaxGroupA`
5. `vatSumInTaxGroupB`
6. `vatSumInTaxGroupC`
7. `vatSumInTaxGroupD`

- `fiscalRecordNumber` Number of the fiscal memory record.
- `errorCode` One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- `outputFiscalRecordNumber` Z-report record number.
- `fiscalReceiptsCount` Fiscal receipts count for the day.
- `vatSumInTaxGroupX` VAT sum for the corresponding VAT group.
- `specialTax` Special tax accumulated sum.

#### Parameters

<i>fiscalRecord- Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte with a value of:
  - 'P' The data are valid.
  - 'F' Wrong control sum in the entry. No data available.
  - 'E' The selected entry is empty. No data available.
- Key-Value - @"outputFiscalRecordNumber" - Z-report record number.
- Key-Value - @"fiscalReceiptsCount" - Fiscal receipts count for the day.
- Key-Value - @"vatSumInTaxGroupA" - VAT sum for the corresponding VAT group.
- Key-Value - @"vatSumInTaxGroupB" - VAT sum for the corresponding VAT group.
- Key-Value - @"vatSumInTaxGroupC" - VAT sum for the corresponding VAT group.
- Key-Value - @"vatSumInTaxGroupD" - VAT sum for the corresponding VAT group.

**1.1.2.36** - (NSDictionary \*) `command114Variant3Version1FiscalRecordNumber1:` (NSString \*) *fiscalRecordNumber1* *fiscalRecordNumber2*:(NSString \*) *fiscalRecordNumber2* error:(NSError \*\*) *error*

72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about net sums for the Z-report record(period). Periodic references for longer time periods may take few seconds to process.

- Data field: `fiscalRecordNumber1`
  1. `fiscalRecordNumber2`
- Response:
  1. `errorCode`
  2. `outputFiscalRecordNumber`
  3. `fiscalReceiptsCount`
  4. `vatSumInTaxGroupA`
  5. `vatSumInTaxGroupB`
  6. `vatSumInTaxGroupC`
  7. `vatSumInTaxGroupD`
- `fiscalRecordNumberX` Number of the fiscal memory record.
- `errorCode` One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- `outputFiscalRecordNumber` Z-report record number.
- `fiscalReceiptsCount` Fiscal receipts count for the day.
- `vatSumInTaxGroupX` VAT sum for the corresponding VAT group.
- `specialTax` Special tax accumulated sum.

#### Parameters

<i>fiscalRecord-Number1</i>	- Number of the fiscal memory record.
<i>fiscalRecord-Number2</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- Key-Value - @"outputFiscalRecordNumber" - Z-report record number.
- Key-Value - @"fiscalReceiptsCount" - Fiscal receipts count for the day.
- Key-Value - @"vatSumInTaxGroupA" - VAT sum for the corresponding VAT group.
- Key-Value - @"vatSumInTaxGroupB" - VAT sum for the corresponding VAT group.
- Key-Value - @"vatSumInTaxGroupC" - VAT sum for the corresponding VAT group.
- Key-Value - @"vatSumInTaxGroupD" - VAT sum for the corresponding VAT group.

1.1.2.37 - (NSDictionary \*) *command114Variant4Version0FiscalRecordNumber:* (NSString \*) *fiscalRecordNumber*  
*error:(NSError \*\*) error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Additional information about the Z-report record. Periodic references for longer time periods may take few seconds to process.

- Data field: *fiscalRecordNumber*
- Response:
  1. *errorCode*
  2. *outputFiscalRecordNumber*
  3. *lastTaxRatesRecordNumber*
  4. *lastResetRecordNumber*
  5. *lastElectronicJournalNumber*
  6. *dateAndTime*
- *fiscalRecordNumber* Number of the fiscal memory record.
- *errorCode* One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- *outputFiscalRecordNumber* Z-report record number.
- *lastTaxRatesRecordNumber* Last (active) decimals and VAT rates record number.
- *lastResetRecordNumber* Last RAM reset number for this Z-report record.
- *lastElectronicJournalNumber* Last electronic journal number for this Z-report record.
- *dateAndTime* Date and time of the data in format: DD-MM-YY hh:mm:ss.

#### Parameters

<i>fiscalRecord-Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- Key-Value - @"outputFiscalRecordNumber" - Z-report record number.
- Key-Value - @"lastTaxRatesRecordNumber" - Last (active) decimals and VAT rates record number.
- Key-Value - @"lastResetRecordNumber" - Last RAM reset number for this Z-report record.

- KeyValue - @"lastElectronicJournalNumber" - Last electronic journal number for this Z-report record.
- KeyValue - @"dateAndTime" - Date and time of the data in format: DD-MM-YY hh:mm:ss.

1.1.2.38 - (NSDictionary \*) command114Variant5Version0FiscalRecordNumber: (NSString \*) *fiscalRecordNumber* error:(NSError \*\*) *error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about decimals and VAT rates record. Periodic references for longer time periods may take few seconds to process.

- Data field: *fiscalRecordNumber*
- Response:
  1. *errorCode*
  2. *decimalsCount*
  3. *enabledTaxArray*
  4. *taxPercentRateA*
  5. *taxPercentRateB*
  6. *taxPercentRateC*
  7. *taxPercentRateD*
  8. *dateAndTime*
- *fiscalRecordNumber* Number of the fiscal memory record.
- *errorCode* One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- *decimalsCount* Decimals count for this Z-report record.
- *enabledTaxArray* Enabled VAT rates mask - 4 bytes with values '0' or '1', where '1' means "enabled".
- *taxPercentRateX* VAT rate for the corresponding VAT group in percents.
- *dateAndTime* Date and time of the data in format: DD-MM-YY hh:mm:ss.

#### Parameters

<i>fiscalRecord-Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.

- Key-Value - @"decimalsCount" - Decimals count for this Z-report record.
- Key-Value - @"enabledTaxArray" - Enabled VAT rates mask - 4 bytes with values '0' or '1', where '1' means "enabled".
- Key-Value - @"taxPercentRateA" - VAT rate for the corresponding VAT group in percents.
- Key-Value - @"taxPercentRateB" - VAT rate for the corresponding VAT group in percents.
- Key-Value - @"taxPercentRateC" - VAT rate for the corresponding VAT group in percents.
- Key-Value - @"taxPercentRateD" - VAT rate for the corresponding VAT group in percents.
- Key-Value - @"dateAndTime" - Date and time of the data in format: DD-MM-YY hh:mm:ss.

1.1.2.39 - (NSDictionary \*) command114Variant6Version0FiscalRecordNumber: (NSString \*) *fiscalRecordNumber* error:(NSError \*\*) *error*

#### 72h(114) INFORMATION ON THE FISCAL ENTRY OR A FISCAL PERIOD

Information about RAM reset record. Periodic references for longer time periods may take few seconds to process.

- Data field: *fiscalRecordNumber*
- Response:
  1. *errorCode*
  2. *dateAndTime*
- *fiscalRecordNumber* Number of the fiscal memory record.
- *errorCode* One byte with a value of:
  1. - 'P' The data are valid.
  2. - 'F' Wrong control sum in the entry. No data available.
  3. - 'E' The selected entry is empty. No data available.
- *dateAndTime* Date and time of the data in format: DD-MM-YY hh:mm:ss.

#### Parameters

<i>fiscalRecord-Number</i>	- Number of the fiscal memory record.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"errorCode" - One byte with a value of:
  - - 'P' The data are valid.
  - - 'F' Wrong control sum in the entry. No data available.
  - - 'E' The selected entry is empty. No data available.
- Key-Value - @"dateAndTime" - Date and time of the data in format: DD-MM-YY hh:mm:ss.



1.1.2.40 - (bool) `command115Variant0Version0RowNumber: (NSString *) rowNumber rowData:(NSString *) rowData error:(NSError **) error`

73h(115) PROGRAMMING A GRAPHIC LOGO

- Data field:
  1. rowNumber
  2. rowData
- Response:
  1. None
- rowNumber Shows the line, which is being programmed - a number between 0 and 95
- rowData Graphic data.
  1. Two symbols for each byte of information are entered in the hexadecimal code (Two symbols for every byte).
  2. The length of the data is up to 48 bytes, and if they are less, an automatic addition of "00" follows.

Note: This command offers the option to define a graphic logo with dimensions 48 x 12 mm (384 x 96 dots) designed by the user themselves. The printing of this logo is activated with command 43. It is printed out immediately before the HEADER - at the beginning of each fiscal or non-fiscal receipt. In order to define the whole logo, the command must be executed 96 times - once for each line. After RESET of memory, default logo is active.

#### Parameters

<i>rowNumber</i>	- Shows the line, which is being programmed - a number between 0 and 95
<i>rowData</i>	- Graphic data. Two symbols for each byte of information are entered in the hexadecimal code (Two symbols for every byte). The length of the data is up to 48 bytes, and if they are less, an automatic addition of "00" follows.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.41 - (NSDictionary \*) `command115Variant1Version0RowNumber: (NSString *) rowNumber error:(NSError **) error`

73h(115) PROGRAMMING A GRAPHIC LOGO

- Data field:
  1. rowNumber
- Response:
  1. rowData
- rowNumber Shows the line, which is being programmed - a number between 0 and 95
- rowData Graphic data.
  1. Two symbols for each byte of information are entered in the hexadecimal code (Two symbols for every byte).

2. The length of the data is up to 48 bytes, and if they are less, an automatic addition of "00" follows.

Note: This command offers the option to define a graphic logo with dimensions 48 x 12 mm (384 x 96 dots) designed by the user themselves. The printing of this logo is activated with command 43. It is printed out immediately before the HEADER - at the beginning of each fiscal or non-fiscal receipt. In order to define the whole logo, the command must be executed 96 times - once for each line. After RESET of memory, default logo is active.

#### Parameters

<i>rowNumber</i>	- Shows the line, which is being programmed - a number between 0 and 95
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

##### NSDictionary

- Key-Value - @"rowData" - Graphic data. Two symbols for each byte of information are entered in the hexadecimal code (Two symbols for every byte). The length of the data is up to 48 bytes, and if they are less, an automatic addition of "00" follows.

**1.1.2.42** - (NSDictionary \*) command120Variant0Version0AndReturnError: (NSError \*\*) *error*

#### 78h(120) SWITCHING THE PRINTER OFF

- Data field: No data
- Response: fiscalPrinterAnswer
- fiscalPrinterAnswer Contains the text 'OFF'.

The printer is switched off (eventually after finishing the printing).

#### Returns

##### NSDictionary

- Key-Value - @"fiscalPrinterAnswer" - Contains the text 'OFF'.

**1.1.2.43** - (NSDictionary \*) command122Variant0Version0AndReturnError: (NSError \*\*) *error*

#### 7Ah(122) JOURNAL SUPPORT

Electronic journal information.

- Data field: None
- Response: journalNumber
  1. jnumberAfterLastZ
  2. nextJournalNumber
  3. totalJLinesAfterErase
  4. freeBytesCount
  5. totalBytesCount

journalNumber Journal number jnumberAfterLastZ Journal number after last Z-report nextJournalNumber Next journal number (Subcommand 'N' will get this line number). totalJLinesAfterErase Total journal lines written after last erase. freeBytesCount Free bytes count in el. journal. totalBytesCount Total bytes count in el. journal

## Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

## Returns

NSDictionary

- KeyValue - @"journalNumber" - Journal number
- KeyValue - @"jnumberAfterLastZ" - Journal number after last Z-report
- KeyValue - @"nextJournalNumber" - Next journal number (Subcommand 'N' will get this line number).
- KeyValue - @"totalJLinesAfterErase" - Total journal lines written after last erase.
- KeyValue - @"freeBytesCount" - Free bytes count in el. journal.
- KeyValue - @"totalBytesCount" - Total bytes count in el. journal

1.1.2.44 - (NSDictionary \*) command122Variant1Version0AndReturnError: (NSError \*\*) *error*

7Ah(122) JOURNAL SUPPORT

Get first journal line.

- Data field: None
- Response: errorCode
  1. journalLineText
- errorCode
  1. 'F' No journal line found
  2. 'P' Journal line successfully read
- journalLineText The journal line.

## Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

## Returns

NSDictionary

- KeyValue - @"errorCode" - One byte, showing the result from the operation and having the following meaning:
  - 'F' No journal line found
  - 'P' Journal line successfully read
- KeyValue - @"journalLineText" - The journal line.

1.1.2.45 - (NSDictionary \*) command122Variant1Version1AndReturnError: (NSError \*\*) *error*

7Ah(122) JOURNAL SUPPORT

Get next journal line.

- Data field: None
- Response: errorCode
  1. journalLineText
- errorCode
  1. 'F' No journal line found
  2. 'P' Journal line successfully read
- journalLineText The journal line.

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

NSDictionary

- KeyValue - @" " - One byte, showing the result from the operation and having the following meaning:
  - 'F' No journal line found
  - 'P' Journal line successfully read
- KeyValue - @" " - The journal line.

**1.1.2.46** - (bool) **command122Variant2Version0AndReturnError:** (NSError \*\*) *error*

7Ah(122) JOURNAL SUPPORT

Print electronic journal using half-height font starting from the first journal line.

- Data field: None
- Response: None

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

TRUE upon success, FALSE otherwise

**1.1.2.47** - (bool) **command122Variant2Version1AndReturnError:** (NSError \*\*) *error*

7Ah(122) JOURNAL SUPPORT

Print electronic journal using normal font starting from the first journal line.

- Data field: None

- Response: None

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

TRUE upon success, FALSE otherwise

**1.1.2.48** - (bool) `command122Variant3Version0AndReturnError: (NSError **) error`

7Ah(122) JOURNAL SUPPORT

Continue printing of the electronic journal using half-height font - starting with the first non-printed line.

- Data field: None
- Response: None

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

TRUE upon success, FALSE otherwise

**1.1.2.49** - (bool) `command122Variant3Version1AndReturnError: (NSError **) error`

7Ah(122) JOURNAL SUPPORT

Continue printing of the electronic journal using normal font - starting with the first non-printed line.

- Data field: None
- Response: None

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

TRUE upon success, FALSE otherwise

**1.1.2.50** - (bool) `command122Variant4Version0AndReturnError: (NSError **) error`

7Ah(122) JOURNAL SUPPORT

Erase electronic journal. Before this the journal must be printed!

- Data field: None
- Response: None

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

TRUE upon success, FALSE otherwise

1.1.2.51 - (NSDictionary \*) **command38Variant0Version0AndReturnError:** (NSError \*\*) *error*

26h (38) Opening a non-fiscal receipt.

- Data field:
  1. None
- Response:
  1. Allreceipt The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

The FP performs the following actions:

- Prints the header
- Prints the tax registration number of the seller
- A response is received, which contains Allreceipt

The command cannot be executed, S1.1 is raised if.

- The fiscal memory has not been formatted
- There is an opened fiscal or non-fiscal receipt
- There is no paper
- The clock is not set
- The electronic journal is full

**Returns**

NSDictionary

- Key-Value - @"allReceipt" - The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

### 1.1.2.52 - (NSDictionary \*) command39Variant0Version0AndReturnError: (NSError \*\*) error

27h (39) Closing a non-fiscal receipt

- Data field:
  1. None
- Response:
  1. Allreceipt The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

The FP performs the following actions:

- Prints the footer
- The sequence number, date and hour of document are printed
- "NON-FISCAL RECEIPT" is printed in expanded style.

If the S1.1 flag is raised, the command is not executed because there is no opened non-fiscal receipt.

#### Returns

NSDictionary

- KeyValue - @"allReceipt" - The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

### 1.1.2.53 - (bool) command41Variant0Version0Switches: (NSString \*) switches error:(NSError \*\*) error

29h (41) SET MEMORY SWITCHES

- Data field:
  1. Switches 8 bytes with value '0' or '1' - the configuration switches.
- Response:
  1. None

#### Note:

The command writes to flash memory the switches value, graphics logo, bar code height, print darkness and default drawer pulse length. After RAM reset they are restored with the saved values. If the switches parameter is not present, then the old switches are kept.

#### Parameters

<i>switches</i>	- 8 bytes with value '0' or '1' - the configuration switches.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.54 - (bool) `command41Variant0Version1AndReturnError: (NSError **) error`

#### 29h (41) SET MEMORY SWITCHES

- Data field:
  1. None
- Response:
  1. None

The command writes to flash memory the switches value, graphics logo, bar code height, print darkness and default drawer pulse length. After RAM reset they are restored with the saved values. If the switches parameter is not present, then the old switches are kept.

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.55 - (bool) `command42Variant0Version0InputText: (NSString *) inputText error:(NSError **) error`

#### 2Ah (42) PRINTING OF A FREE NON-FISCAL TEXT

- Data field:
  1. `inputText` A text of 30 symbols (at most). The symbols after 30 are cut off.
- Response:
  1. None

If S1.1 is raised, there is no non-fiscal receipt opened and the text is not printed.

#### Parameters

<i>inputText</i>	- A text of 30 symbols (at most). The symbols after 30 are cut off.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.56 - (bool) `command43Variant0Version0ItemIndex: (NSString *) itemIndex dataValue:(NSString *) dataValue error:(NSError **) error`

#### 2Bh (43) SETTING FOOTER AND PRINTING OPTIONS

- Data field:
  1. `itemIndex`
  2. `dataValue`
- Response:
  1. Entries from the data field



FOOTER consists of 2 lines of text printed at the end of each receipt. HEADER and FOOTER are automatically center aligned.

- itemIndex One symbol having the following meaning:
  1. "6" to "7" "6" and "7" select the first or second FOOTER line.
  2. "B" Set bar code height in pixels (0.125 mm). Possible values from 24 (3 mm) to 240 (30 mm). The barcode is printed with command 84 (54H).
  3. "D" Set print darkness. Possible values:
    - (a) '1': Very low
    - (b) '2': Low
    - (c) '3': Normal
    - (d) '4': Dark
    - (e) '5': Very dark
  4. "E" Disable / enable printing of total receipt sum in EUR when first Total (53) command is executed. Optionally the command sets the valid exchange rate EUR / LEI. Syntax of the data:
    - (a) <Enable>[,Rate]
    - (b) Enable Disable / enable flag. One byte: '0' or '1'.
    - (c) Rate Exchange rate. A float number with up to 5 decimal places and 8 significant digits. If not preset, then old value remains active. If 0.00000, then EUR value is not printed independent of the value of Enable flag.
  5. "L" Height of graphic logo and permission/rejection of the printing of graphic logo immediately before the header. This logo is defined with command 115 (73H).
  6. "Z" Set auto off time in seconds. Possible values from 60 to 3600. Setting 0 disables the auto off.
  7. "I" Gives us the option to read values, set earlier with command 43. After the letter "I" only one more symbol follows which coincides with some of the above.
- dataValue A text string:
  1. If <item> is '0' to '7' - the text of the header / footer line (up to 38 symbols).
  2. If <item> = 'B' - A number - the height of bar code in pixels.
  3. If <item> = 'D' - The print darkness (1 to 5).
  4. If <item> = 'E' - Returns Enable,Rate, where Enable is Disable / enable flag and Rate is current exchange rate EUR / LEI.
  5. If <item> = 'L' Syntax <Height>,<Enabled>
    - (a) Height Graphics logo height in lines (0.125 mm). A number from 8 to 96.
    - (b) Enabled '0' or '1', where '1' means, that logo printing is enabled.
  6. If <item> = 'Z' Syntax <OffTime>

## Parameters

<i>itemIndex</i>	<p>- One symbol having the following meaning:</p> <ol style="list-style-type: none"> <li>1. "6" to "7" "6" and "7" select the first or second FOOTER line.</li> <li>2. "B" Set bar code height in pixels (0.125 mm). Possible values from 24 (3 mm) to 240 (30 mm). The barcode is printed with command 84 (54H).</li> <li>3. "D" Set print darkness. Possible values: <ol style="list-style-type: none"> <li>(a) '1': Very low</li> <li>(b) '2': Low</li> <li>(c) '3': Normal</li> <li>(d) '4': Dark</li> <li>(e) '5': Very dark</li> </ol> </li> <li>4. "E" Disable / enable printing of total receipt sum in EUR when first Total (53) command is executed. Optionally the command sets the valid exchange rate EUR / LEI. Syntax of the data: <ol style="list-style-type: none"> <li>(a) &lt;Enable&gt;[,Rate]</li> <li>(b) Enable Disable / enable flag. One byte: '0' or '1'.</li> <li>(c) Rate Exchange rate. A float number with up to 5 decimal places and 8 significant digits. If not preset, then old value remains active. If 0.00000, then EUR value is not printed independent of the value of Enable flag.</li> </ol> </li> <li>5. "L" Height of graphic logo and permission/rejection of the printing of graphic logo immediately before the header. This logo is defined with command 115 (73H).</li> <li>6. "Z" Set auto off time in seconds. Possible values from 60 to 3600. Setting 0 disables the auto off.</li> <li>7. "I" Gives us the option to read values, set earlier with command 43. After the letter "I" only one more symbol follows which coincides with some of the above.</li> </ol>
<i>dataValue</i>	<p>- A text string:</p> <ol style="list-style-type: none"> <li>1. If &lt;item&gt; is '0' to '7' - the text of the header / footer line (up to 38 symbols).</li> <li>2. If &lt;item&gt; = 'B' - A number - the height of bar code in pixels.</li> <li>3. If &lt;item&gt; = 'D' - The print darkness (1 to 5).</li> <li>4. If &lt;item&gt; = 'E' - Returns Enable,Rate, where Enable is Disable / enable flag and Rate is current exchange rate EUR / LEI.</li> <li>5. If &lt;item&gt; = 'L' Syntax &lt;Height&gt;,&lt;Enabled&gt; <ol style="list-style-type: none"> <li>(a) Height Graphics logo height in lines (0.125 mm). A number from 8 to 96.</li> <li>(b) Enabled '0' or '1', where '1' means, that logo printing is enabled.</li> </ol> </li> <li>6. If &lt;item&gt; = 'Z' Syntax &lt;OffTime&gt;</li> </ol>
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.57 - (bool) `command44Variant0Version0TargetLines: (NSString *) targetLines error:(NSError **) error`

2Ch(44) ADVANCING PAPER

- Data field: targetLines
- Response: None

- `targetLines` Advancing paper measured in text lines. The programmed line count cannot be greater than 99 (1 or 2 bytes).
- If the parameter is not there, the default setting is 1 line.

**Parameters**

<i>targetLines</i>	- Advancing paper measured in text lines. The programmed line count cannot be greater than 99 (1 or 2 bytes). If the parameter is not there, the default setting is 1 line.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

TRUE upon success, FALSE otherwise

### 1.1.2.58 - (NSDictionary \*) `command45Variant0Version0AndReturnError: (NSError **) error`

#### 2Dh (45) OPENING A RECEIPT FOR 90 DEGREES ROTATED TEXT

- Data field: None
- Response: `allRotationReceipt`
- `allRotationReceipt` Rotated receipt number after last Z-report. 4 bytes.

The command opens a non-fiscal receipt for 90 degrees rotated text.

The command will be not executed if:

- A receipt is open (fiscal or non-fiscal).
- Out of paper.
- The clock has to be set.

**Returns**

NSDictionary

- Key-Value - `@ "allRotationReceipt"` - Rotated receipt number after last Z-report. 4 bytes.

### 1.1.2.59 - (bool) `command46Variant0Version0RotatedTextRow: (NSString *) rotatedTextRow error:(NSError **) error`

#### 2Eh(46) PRINT 90 DEGREES ROTATED TEXT

- Data field: Text
- Response: None
- Text The line to be print. Length is up to 100 symbols. It is possible to print bold and underline:
  1. `<Tab>B` Start bold printing.
  2. `<Tab>b` Cancel bold printing.
  3. `<Tab>U` Start underlined printing.
  4. `<Tab>u` Cancel underlined printing.

The command sends text for 90 degrees rotated lines. On the paper can be placed up to 12 rotated lines. Sent lines are accumulated in printer's memory. If the command is executed more than 12 times, then the text is printed and the printer expects new lines or command 47 (close the receipt). The printer detects the longest text line and fill all other lines to the same length with spaces (ASCII 20h). If more than 12 lines are sent, then more than one column of lines is printed. There is no space between the columns, so rotated lines of unlimited length can be printed.

The command is not permitted, if no rotated receipt is open.

#### Parameters

<i>rotatedTextRow</i>	- The line to be print. Length is up to 100 symbols. It is possible to print bold and underline: <ol style="list-style-type: none"> <li>1. &lt;Tab&gt;B Start bold printing.</li> <li>2. &lt;Tab&gt;b Cancel bold printing.</li> <li>3. &lt;Tab&gt;U Start underlined printing.</li> <li>4. &lt;Tab&gt;u Cancel underlined printing.</li> </ol>
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.60 - (NSDictionary \*) **command47Variant0Version0AndReturnError:** (NSError \*\*) *error*

2Fh(47) CLOSING A RECEIPT FOR 90 DEGREES ROTATED TEXT

- Data field:
  1. None
- Response:
  1. allRotationReceipt Rotated receipt number after last Z-report. 4 bytes.

The command closes the receipt. If there are unprinted lines, they are automatically printed before this.

The command will be not executed, if no rotated receipt is open.

#### Returns

NSDictionary

- Key-Value - @"allRotationReceipt" - Rotated receipt number after last Z-report. 4 bytes.

1.1.2.61 - (NSDictionary \*) **command48Variant0Version0OperatorCode:** (NSString \*) *operatorCode* **operatorPassword:** (NSString \*) *operatorPassword* **tillNumber:** (NSString \*) *tillNumber* **error:** (NSError \*\*) *error*

30h(48) OPENING A FISCAL CLIENT'S RECEIPT

- Data field:
  1. operatorCode
  2. operatorPassword
  3. tillNumber
- Response:

1. allReceiptNumber
2. fiscalReceiptNumber

- operatorCode Operator's number (1 to 16)
- operatorPassword Operator's password (4 to 8 digits)
- tillNumber Number of point of sale (a whole number of maximum 5 digits)
- allReceiptNumber The number of all issued receipts (fiscal or non-) from the last daily closure up to the moment (4 bytes).
- fiscalReceiptNumber The number of all fiscal receipts from the last daily closure up to the moment (4 bytes).

The FP performs the following actions:

- Prints the HEADER
- Prints the tax registration number
- Prints the number and name of operator as well as the cashier desk number
- Allreceipt and FiscReceipt are returned

The command will not be successful if:

- There is an opened fiscal or non-fiscal receipt
- The maximum number of receipts, as fixed for the day, has already been issued
- The fiscal memory is full
- The fiscal memory is damaged
- No code or operator password, or cashier desk number available
- Less than 2 HEADER lines are programmed
- No VAT registration number available
- Wrong operator password
- The clock needs setting
- Journal is full

After entering three wrong operator's passwords, the printer blocks and must be switched off and on again to restart operation.

#### Parameters

<i>operatorCode</i>	- Operator's number (1 to 16)
<i>operator-Password</i>	- Operator's password (4 to 8 digits)
<i>tillNumber</i>	- Number of point of sale (a whole number of maximum 5 digits)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

## NSDictionary

- KeyValue - @"allReceiptNumber" - The number of all issued receipts (fiscal or non-) from the last daily closure up to the moment (4 bytes).
- KeyValue - @"fiscalReceiptNumber" - The number of all fiscal receipts from the last daily closure up to the moment (4 bytes).

1.1.2.62 - (bool) command49Variant0Version0TaxGroup: (NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* error:(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

## • Data field:

1. taxGroup
2. itemPrice

## • Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

## Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.63 - (bool) `command49Variant0Version10TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. taxGroup
3. itemPrice
4. itemQuantity

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.64 - (bool) `command49Variant0Version11TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity specialTax:(NSString *) specialTax  
error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. taxGroup
3. itemPrice
4. itemQuantity
5. specialTax

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

## Parameters



<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.65 - (bool) command49Variant0Version12TextRow1: (NSString \*) *textRow1* textRow2:(NSString \*) *textRow2* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* error:(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. textRow2
3. taxGroup
4. itemPrice

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. textRow2 A text of up to 30 bytes containing a second line describing the sale.
3. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. itemPrice This is a single price that consists of 8 meaningful digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.66** - (bool) **command49Variant0Version13TextRow1:** (NSString \*) *textRow1* **textRow2:**(NSString \*) *textRow2* **taxGroup:**(NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **specialTax:**(NSString \*) *specialTax* **error:**(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. *textRow1*
  2. *textRow2*
  3. *taxGroup*
  4. *itemPrice*
  5. *specialTax*
- Response: None
  1. *textRow1* A text of up to 30 bytes containing one line of description of sale.
  2. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
  3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
  4. *itemPrice* This is a single price that consists of 8 meaningful digits.
  5. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.

- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.67** - (bool) **command49Variant0Version14TextRow1:** (NSString \*) *textRow1* **textRow2:**(NSString \*) *textRow2* **taxGroup:**(NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **itemQuantity:**(NSString \*) *itemQuantity* **error:**(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow1
  2. textRow2
  3. taxGroup
  4. itemPrice
  5. itemQuantity
- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. textRow2 A text of up to 30 bytes containing a second line describing the sale.

3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result *Price\*Quan* is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result <i>Price*Quan</i> is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.68** - (bool) *command49Variant0Version15TextRow1: (NSString \*) textRow1 textRow2:(NSString \*) textRow2 taxGroup:(NSString \*) taxGroup itemPrice:(NSString \*) itemPrice itemQuantity:(NSString \*) itemQuantity specialTax:(NSString \*) specialTax error:(NSError \*\*) error*

31h(49) REGISTRATION OF SALES

- Data field:

1. *textRow1*
2. *textRow2*
3. *taxGroup*
4. *itemPrice*
5. *itemQuantity*
6. *specialTax*

- Response: None

1. *textRow1* A text of up to 30 bytes containing one line of description of sale.
2. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
6. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.

<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.69 - (bool) **command49Variant0Version1TaxGroup:** (NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **specialTax:**(NSString \*) *specialTax* **error:**(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. taxGroup
  2. itemPrice
  3. specialTax
- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.70 - (bool) `command49Variant0Version2TaxGroup: (NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. itemQuantity

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.71 - (bool) **command49Variant0Version3TaxGroup:** (NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* error:(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. itemQuantity
4. specialTax

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)



- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

**Parameters**

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

TRUE upon success, FALSE otherwise

1.1.2.72 - (bool) command49Variant0Version4TextRow2: (NSString \*) *textRow2* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* error:(NSError \*\*) *error*

**31h(49) REGISTRATION OF SALES**

- Data field:
  1. *textRow2*
  2. *taxGroup*
  3. *itemPrice*
- Response: None

1. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
2. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. *itemPrice* This is a single price that consists of 8 meaningful digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.73 - (bool) **command49Variant0Version5TextRow2:** (NSString \*) *textRow2* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* specialTax:(NSString \*) *specialTax* error:(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow2
  2. taxGroup
  3. itemPrice
  4. specialTax
- Response: None

1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.

- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.74 - (bool) `command49Variant0Version6TextRow2: (NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity error:(NSError **) error`

#### 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow2
2. taxGroup
3. itemPrice
4. itemQuantity

- Response: None

1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.75 - (bool) **command49Variant0Version7TextRow2:** (NSString \*) *textRow2* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* error:(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow2
  2. taxGroup
  3. itemPrice
  4. itemQuantity
  5. specialTax
- Response: None

1. textRow2 A text of up to 30 bytes containing a second line describing the sale.

2. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. *itemPrice* This is a single price that consists of 8 meaningful digits.
4. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- A non-mandatory parameter setting the quantity of items for sale. By default, this is 1.000.- The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.76 - (bool) `command49Variant0Version8TextRow1:(NSString *) textRow1 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice error:(NSError **) error`

### 31h(49) REGISTRATION OF SALES

- Data field:

1. `textRow1`
2. `taxGroup`
3. `itemPrice`

- Response: None

1. `textRow1` A text of up to 30 bytes containing one line of description of sale.
2. `taxGroup` One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. `itemPrice` This is a single price that consists of 8 meaningful digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.77 - (bool) `command49Variant0Version9TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice specialTax:(NSString *) specialTax error:(NSError **) error`

### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow1
  2. taxGroup
  3. itemPrice
  4. specialTax
- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.78 - (bool) `command49Variant1Version0TaxGroup: (NSString *) taxGroup itemPrice:(NSString *) itemPrice sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. sellWithAbsoluteSumDiscount

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information



## Returns

TRUE upon success, FALSE otherwise

1.1.2.79 - (bool) `command49Variant1Version10TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity sellWithAbsoluteSumDiscount:(NSString *)  
sellWithAbsoluteSumDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. taxGroup
3. itemPrice
4. itemQuantity
5. sellWithAbsoluteSumDiscount

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

**1.1.2.80** - (bool) **command49Variant1Version11TextRow1:** (NSString \*) *textRow1* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* sellWithAbsoluteSumDiscount:(NSString \*) *sellWithAbsoluteSumDiscount* error:(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

### • Data field:

1. textRow1
2. taxGroup
3. itemPrice
4. itemQuantity
5. specialTax
6. sellWithAbsoluteSumDiscount

### • Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
6. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.81** - (bool) `command49Variant1Version12TextRow1: (NSString *) textRow1 textRow2:(NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow1
  2. textRow2
  3. taxGroup

- 4. *itemPrice*
- 5. *sellWithAbsoluteSumDiscount*

- Response: None

1. *textRow1* A text of up to 30 bytes containing one line of description of sale.
2. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *sellWithAbsoluteSumDiscount* The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.82 - (bool) `command49Variant1Version13TextRow1: (NSString *) textRow1 textRow2:(NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice specialTax:(NSString *) specialTax sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

### 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. textRow2
3. taxGroup
4. itemPrice
5. specialTax
6. sellWithAbsoluteSumDiscount

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. textRow2 A text of up to 30 bytes containing a second line describing the sale.
3. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. itemPrice This is a single price that consists of 8 meaningful digits.
5. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
6. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

## Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.83 - (bool) **command49Variant1Version14TextRow1:** (NSString \*) *textRow1* **textRow2:**(NSString \*) *textRow2* **taxGroup:**(NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **itemQuantity:**(NSString \*) *itemQuantity* **sellWithAbsoluteSumDiscount:**(NSString \*) *sellWithAbsoluteSumDiscount* **error:**(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. *textRow1*
2. *textRow2*
3. *taxGroup*
4. *itemPrice*
5. *itemQuantity*
6. *sellWithAbsoluteSumDiscount*

- Response: None

1. *textRow1* A text of up to 30 bytes containing one line of description of sale.
2. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
6. *sellWithAbsoluteSumDiscount* The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.

- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWithAbsoluteSumDiscount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.84 - (bool) `command49Variant1Version15TextRow1: (NSString *) textRow1 textRow2:(NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity specialTax:(NSString *) specialTax sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow1
  2. textRow2
  3. taxGroup
  4. itemPrice
  5. itemQuantity
  6. specialTax
  7. sellWithAbsoluteSumDiscount
- Response: None

1. *textRow1* A text of up to 30 bytes containing one line of description of sale.
2. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
6. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
7. *sellWithAbsoluteSumDiscount* The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information



## Returns

TRUE upon success, FALSE otherwise

1.1.2.85 - (bool) **command49Variant1Version1TaxGroup:** (NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **specialTax:**(NSString \*) *specialTax* **sellWithAbsoluteSumDiscount:**(NSString \*) *sellWithAbsoluteSumDiscount* **error:**(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. specialTax
4. sellWithAbsoluteSumDiscount

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
4. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.86 - (bool) **command49Variant1Version2TaxGroup:** (NSString \*) *taxGroup* *itemPrice*:(NSString \*) *itemPrice* *itemQuantity*:(NSString \*) *itemQuantity* *sellWithAbsoluteSumDiscount*:(NSString \*) *sellWithAbsoluteSumDiscount* *error*:(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. itemQuantity
4. sellWithAbsoluteSumDiscount

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
4. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.87** - (bool) **command49Variant1Version3TaxGroup:** (NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* sellWithAbsoluteSumDiscount:(NSString \*) *sellWithAbsoluteSumDiscount* error:(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. itemQuantity
4. specialTax
5. sellWithAbsoluteSumDiscount

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

5. **sellWithAbsoluteSumDiscount** The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.88 - (bool) **command49Variant1Version4TextRow2:** (NSString \*) *textRow2* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* sellWithAbsoluteSumDiscount:(NSString \*) *sellWithAbsoluteSumDiscount* error:(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. *textRow2*

2. taxGroup
3. itemPrice
4. sellWithAbsoluteSumDiscount

- Response: None

1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.89 - (bool) `command49Variant1Version5TextRow2:` (NSString \*) *textRow2* `taxGroup:`(NSString \*) *taxGroup* `itemPrice:`(NSString \*) *itemPrice* `specialTax:`(NSString \*) *specialTax* `sellWithAbsoluteSumDiscount:`(NSString \*) *sellWithAbsoluteSumDiscount* `error:`(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. `textRow2`
2. `taxGroup`
3. `itemPrice`
4. `specialTax`
5. `sellWithAbsoluteSumDiscount`

- Response: None

1. `textRow2` A text of up to 30 bytes containing a second line describing the sale.
2. `taxGroup` One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. `itemPrice` This is a single price that consists of 8 meaningful digits.
4. `specialTax` Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
5. `sellWithAbsoluteSumDiscount` The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

**1.1.2.90** - (bool) `command49Variant1Version6TextRow2: (NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow2
2. taxGroup
3. itemPrice
4. itemQuantity
5. sellWithAbsoluteSumDiscount

- Response: None

1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.

- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWithAbsoluteSumDiscount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.91 - (bool) **command49Variant1Version7TextRow2:** (NSString \*) *textRow2* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* sellWithAbsoluteSumDiscount:(NSString \*) *sellWithAbsoluteSumDiscount* error:(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow2
  2. taxGroup
  3. itemPrice
  4. itemQuantity
  5. specialTax
  6. sellWithAbsoluteSumDiscount
- Response: None
  1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
  2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).



3. *itemPrice* This is a single price that consists of 8 meaningful digits.
4. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
6. *sellWithAbsoluteSumDiscount* The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.92 - (bool) `command49Variant1Version8TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount  
error:(NSError **) error`

### 31h(49) REGISTRATION OF SALES

- Data field:

1. `textRow1`
2. `taxGroup`
3. `itemPrice`
4. `sellWithAbsoluteSumDiscount`

- Response: None

1. `textRow1` A text of up to 30 bytes containing one line of description of sale.
2. `taxGroup` One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. `itemPrice` This is a single price that consists of 8 meaningful digits.
4. `sellWithAbsoluteSumDiscount` The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWith- AbsoluteSum- Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.93 - (bool) `command49Variant1Version9TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice specialTax:(NSString *) specialTax sellWithAbsoluteSumDiscount:(NSString *)  
sellWithAbsoluteSumDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. taxGroup
3. itemPrice
4. specialTax
5. sellWithAbsoluteSumDiscount

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
5. sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

## Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the currently performed sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

### Returns

TRUE upon success, FALSE otherwise

**1.1.2.94** - (bool) **command49Variant2Version0TaxGroup:** (NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **sellWithPercentDiscount:**(NSString \*) *sellWithPercentDiscount* **error:**(NSError \*\*) *error*

### 31h(49) REGISTRATION OF SALES

- Data field:

1. *taxGroup*
2. *itemPrice*
3. *sellWithPercentDiscount*

- Response: None

1. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. *itemPrice* This is a single price that consists of 8 meaningful digits.
3. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)

- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

## Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.95 - (bool) `command49Variant2Version10TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:
    1. textRow1
    2. taxGroup
    3. itemPrice
    4. itemQuantity
    5. sellWithPercentDiscount
  - Response: None
1. textRow1 A text of up to 30 bytes containing one line of description of sale.
  2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
  3. itemPrice This is a single price that consists of 8 meaningful digits.
  4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
  5. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.

- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.96** - (bool) **command49Variant2Version11TextRow1:** (NSString \*) *textRow1* taxGroup:(NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* sellWithPercentDiscount:(NSString \*) *sellWithPercentDiscount* error:(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow1
  2. taxGroup
  3. itemPrice
  4. itemQuantity
  5. specialTax
  6. sellWithPercentDiscount
- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.

2. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. *itemPrice* This is a single price that consists of 8 meaningful digits.
4. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
6. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.97 - (bool) `command49Variant2Version12TextRow1: (NSString *) textRow1 textRow2:(NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. textRow2
3. taxGroup
4. itemPrice
5. sellWithPercentDiscount

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. textRow2 A text of up to 30 bytes containing a second line describing the sale.
3. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. itemPrice This is a single price that consists of 8 meaningful digits.
5. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full



## Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.98 - (bool) **command49Variant2Version13TextRow1:** (NSString \*) *textRow1* **textRow2:**(NSString \*) *textRow2* **taxGroup:**(NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **specialTax:**(NSString \*) *specialTax* **sellWithPercentDiscount:**(NSString \*) *sellWithPercentDiscount* **error:**(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

## • Data field:

1. textRow1
2. textRow2
3. taxGroup
4. itemPrice
5. specialTax
6. sellWithPercentDiscount

## • Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. textRow2 A text of up to 30 bytes containing a second line describing the sale.
3. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. itemPrice This is a single price that consists of 8 meaningful digits.
5. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
6. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.

- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.99 - (bool) **command49Variant2Version14TextRow1:** (NSString \*) *textRow1* **textRow2:**(NSString \*) *textRow2* **taxGroup:**(NSString \*) *taxGroup* **itemPrice:**(NSString \*) *itemPrice* **itemQuantity:**(NSString \*) *itemQuantity* **sellWithPercentDiscount:**(NSString \*) *sellWithPercentDiscount* **error:**(NSError \*\*) *error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow1
  2. textRow2
  3. taxGroup
  4. itemPrice
  5. itemQuantity
  6. sellWithPercentDiscount
- Response: None
  1. textRow1 A text of up to 30 bytes containing one line of description of sale.
  2. textRow2 A text of up to 30 bytes containing a second line describing the sale.

3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
6. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.100 - (bool) *command49Variant2Version15TextRow1*: (NSString \*) *textRow1* *textRow2*:(NSString \*) *textRow2*  
*taxGroup*:(NSString \*) *taxGroup* *itemPrice*:(NSString \*) *itemPrice* *itemQuantity*:(NSString \*) *itemQuantity*  
*specialTax*:(NSString \*) *specialTax* *sellWithPercentDiscount*:(NSString \*) *sellWithPercentDiscount* *error*:(NSError \*\*) *error*

## 31h(49) REGISTRATION OF SALES

- Data field:

1. *textRow1*
2. *textRow2*
3. *taxGroup*
4. *itemPrice*
5. *itemQuantity*
6. *specialTax*
7. *sellWithPercentDiscount*

- Response: None

1. *textRow1* A text of up to 30 bytes containing one line of description of sale.
2. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
3. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
4. *itemPrice* This is a single price that consists of 8 meaningful digits.
5. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result *Price\*Quan* is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
6. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
7. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

## Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.101 - (bool) `command49Variant2Version1TaxGroup: (NSString *) taxGroup itemPrice:(NSString *) itemPrice specialTax:(NSString *) specialTax sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:
    1. taxGroup
    2. itemPrice
    3. specialTax
    4. sellWithPercentDiscount
  - Response: None
1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
  2. itemPrice This is a single price that consists of 8 meaningful digits.
  3. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

4. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.102 - (bool) *command49Variant2Version2TaxGroup: (NSString \*) taxGroup itemPrice:(NSString \*) itemPrice itemQuantity:(NSString \*) itemQuantity sellWithPercentDiscount:(NSString \*) sellWithPercentDiscount error:(NSError \*\*) error*

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. *taxGroup*
  2. *itemPrice*

- 3. *itemQuantity*
- 4. *sellWithPercentDiscount*

- Response: None

1. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. *itemPrice* This is a single price that consists of 8 meaningful digits.
3. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result *Price\*Quan* is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
4. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result <i>Price*Quan</i> is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.103 - (bool) command49Variant2Version3TaxGroup: (NSString \*) *taxGroup* itemPrice:(NSString \*) *itemPrice* itemQuantity:(NSString \*) *itemQuantity* specialTax:(NSString \*) *specialTax* sellWithPercentDiscount:(NSString \*) *sellWithPercentDiscount* error:(NSError \*\*) *error*

### 31h(49) REGISTRATION OF SALES

- Data field:

1. taxGroup
2. itemPrice
3. itemQuantity
4. specialTax
5. sellWithPercentDiscount

- Response: None

1. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
2. itemPrice This is a single price that consists of 8 meaningful digits.
3. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
5. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full



## Parameters

<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.104 - (bool) `command49Variant2Version4TextRow2: (NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow2
  2. taxGroup
  3. itemPrice
  4. sellWithPercentDiscount
- Response: None
  1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
  2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
  3. itemPrice This is a single price that consists of 8 meaningful digits. =
  4. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.105 - (bool) `command49Variant2Version5TextRow2: (NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice specialTax:(NSString *) specialTax sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

#### 31h(49) REGISTRATION OF SALES

- Data field:
    1. *textRow2*
    2. *taxGroup*
    3. *itemPrice*
    4. *specialTax*
    5. *sellWithPercentDiscount*
  - Response: None
1. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
  2. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
  3. *itemPrice* This is a single price that consists of 8 meaningful digits.
  4. *specialTax* Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
  5. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.106 - (bool) `command49Variant2Version6TextRow2: (NSString *) textRow2 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

#### 31h(49) REGISTRATION OF SALES

- Data field:
  1. textRow2
  2. taxGroup
  3. itemPrice
  4. itemQuantity
  5. sellWithPercentDiscount

- Response: None

1. *textRow2* A text of up to 30 bytes containing a second line describing the sale.
2. *taxGroup* One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. *itemPrice* This is a single price that consists of 8 meaningful digits.
4. *itemQuantity* The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result  $\text{Price} \times \text{Quan}$  is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result $\text{Price} \times \text{Quan}$ is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.107 - (bool) `command49Variant2Version7TextRow2: (NSString *) textRow2 taxGroup:(NSString *) taxGroup  
itemPrice:(NSString *) itemPrice itemQuantity:(NSString *) itemQuantity specialTax:(NSString *) specialTax  
sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

### 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow2
2. taxGroup
3. itemPrice
4. itemQuantity
5. specialTax
6. sellWithPercentDiscount

- Response: None

1. textRow2 A text of up to 30 bytes containing a second line describing the sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. itemQuantity The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price\*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
5. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
6. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow2</i>	- A text of up to 30 bytes containing a second line describing the sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>itemQuantity</i>	- The quantity of items for sale. By default, this is 1.000. The length of this parameter is 9 meaningful digits (not more than 3 after the decimal point). The result Price*Quan is rounded up to the set number of digits and cannot be longer than 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

## Returns

TRUE upon success, FALSE otherwise

1.1.2.108 - (bool) `command49Variant2Version8TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

## 31h(49) REGISTRATION OF SALES

- Data field:

1. textRow1
2. taxGroup
3. itemPrice
4. sellWithPercentDiscount

- Response: None

1. textRow1 A text of up to 30 bytes containing one line of description of sale.
2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
3. itemPrice This is a single price that consists of 8 meaningful digits.
4. sellWithPercentDiscount The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- - The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- - The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.

- - If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- - The maximum number of sales for one receipt have already been performed (380)
- - The 35h command has been successfully executed
- - The sum for some of the tax groups has become negative
- - The sum of discounts and surcharges within the same receipt has become negative
- - Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>sellWithPercentDiscount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.109** - (bool) `command49Variant2Version9TextRow1: (NSString *) textRow1 taxGroup:(NSString *) taxGroup itemPrice:(NSString *) itemPrice specialTax:(NSString *) specialTax sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

#### 31h(49) REGISTRATION OF SALES

- Data field:
    1. textRow1
    2. taxGroup
    3. itemPrice
    4. specialTax
    5. sellWithPercentDiscount
  - Response: None
1. textRow1 A text of up to 30 bytes containing one line of description of sale.
  2. taxGroup One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
  3. itemPrice This is a single price that consists of 8 meaningful digits.
  4. specialTax Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

5. *sellWithPercentDiscount* The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.

FP performs the following actions:

- The text, describing sale is printed out together with the price and code of discount or surcharge. If there is a set quantity, the information on it is printed out too.
- The price of the items sold is accumulated to the sums already stored in the operational memory. In case of memory overflow the value of respective bites of the status field will be set.
- If there is a discount or surcharge, it is printed out on a separate line and is then added to a specially maintained registers within the printer. The values for the day are printed out together with the daily financial report.

The command will not be correctly executed if and when: No fiscal receipt has been opened

- The maximum number of sales for one receipt have already been performed (380)
- The 35h command has been successfully executed
- The sum for some of the tax groups has become negative
- The sum of discounts and surcharges within the same receipt has become negative
- Journal is full

#### Parameters

<i>textRow1</i>	- A text of up to 30 bytes containing one line of description of sale.
<i>taxGroup</i>	- One byte containing letter, which indicates the type of tax. There is a restriction, depending on the enabled tax groups (command 83).
<i>itemPrice</i>	- This is a single price that consists of 8 meaningful digits.
<i>specialTax</i>	- Special tax. Has the dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>sellWithPercent-Discount</i>	- The value of discount or surcharge (depending on the sign) in percent over the currently performed sale. Possible values are between - 99.00% and 99.00%, where up to 2 decimal places are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.110 - (NSDictionary \*) *command50Variant0Version0StartDate:* (NSString \*) *startDate* *endDate:*(NSString \*) *endDate* *error:*(NSError \*\*) *error*

32h(50) TAX RATES ENTERED DURING THE ACCOUNTED PERIOD

- Data field: <Start>,<End>
- Response: Data
- Start The starting date of the period - DDMMYY/6 bytes/



- End The end date of the period - DDMMYY /6 bytes/
- Data
  1. 'F' if no tax rates for the period have been found, or in case of error
  2. 'PAA,BB,CC,DD,DDMMYY' if rates have been found, where 'P' means 'PASS' after which the last active rates for the period are listed out as well as the date of their entry.
  3. If there are unused groups (disabled by command 83) for them, instead of rate in percent a 'DT' is returned (Disabled tax).

The command prints a report on the changes made in the decimal points and tax rates during the selected period.

#### Parameters

<i>startDate</i>	- StartDate The starting date of the period - DDMMYY/6 bytes/
<i>endDate</i>	- EndDate The end date of the period - DDMMYY /6 bytes/
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

##### NSDictionary

- Key-Value - @"fResult" - 'F' if no tax rates for the period have been found, or in case of error 'P' means 'PASS' after which the last active rates for the period
- Key-Value - @"aA" - The last active rates for the period are listed out as well as the date of their entry. If there are unused groups (disabled by command 83) for them, instead of rate in percent a 'DT' is returned (Disabled tax).
- Key-Value - @"bB" - The last active rates for the period are listed out as well as the date of their entry. If there are unused groups (disabled by command 83) for them, instead of rate in percent a 'DT' is returned (Disabled tax).
- Key-Value - @"cC" - The last active rates for the period are listed out as well as the date of their entry. If there are unused groups (disabled by command 83) for them, instead of rate in percent a 'DT' is returned (Disabled tax).
- Key-Value - @"dD" - The last active rates for the period are listed out as well as the date of their entry. If there are unused groups (disabled by command 83) for them, instead of rate in percent a 'DT' is returned (Disabled tax).
- Key-Value - @"dMMYY" - Day Month Year

1.1.2.111 - (NSDictionary \*) command51Variant0Version0ToPrintOption: (NSString \*) toPrintOption toDisplayOption:(NSString \*) toDisplayOption error:(NSError \*\*) error

#### 33h(51) SUBTOTAL

- Data field: toPrintOption,toDisplayOption
- Response: subTotal,taxGroupA,taxGroupB,taxGroupC,taxGroupD,taxGroupE,specialTax
- toPrintOption One byte, which if '1' the sum of the subtotal will be printed out.
- toDisplayOption One byte, which if '1' the sum of the subtotal will be displayed out.

- SubTotal The sum accumulated for the current fiscal receipt (10 bytes).
- taxGroupA The sum over tax group A /10 bytes/
- taxGroupB The sum over tax group B /10 bytes/
- taxGroupC The sum over tax group C /10 bytes/
- taxGroupD The sum over tax group D /10 bytes/
- taxGroupE The sum over tax group E - VAT exempt /10 bytes/
- specialTax The sum over special tax /10 bytes/

The sum of all sales registered in the fiscal receipt is calculated. The calculated total sum and the accumulated separate sums for each tax group are returned to the PC.

#### Parameters

<i>toPrintOption</i>	- One byte, which if '1' the sum of the subtotal will be printed out.
<i>toDisplayOption</i>	- One byte, which if '1' the sum of the subtotal will be displayed out.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"subTotal" - The sum accumulated for the current fiscal receipt (10 bytes).
- KeyValue - @"taxGroupA" - The sum over tax group A /10 bytes/
- KeyValue - @"taxGroupB" - The sum over tax group B /10 bytes/
- KeyValue - @"taxGroupC" - The sum over tax group C /10 bytes/
- KeyValue - @"taxGroupD" - The sum over tax group D /10 bytes/
- KeyValue - @"taxGroupE" - The sum over tax group E - VAT exempt /10 bytes/
- KeyValue - @"specialTax" - The sum over special tax /10 bytes/

1.1.2.112 - (NSDictionary \*) **command51Variant0Version1ToPrintOption:** (NSString \*) *toPrintOption* **toDisplayOption:** (NSString \*) *toDisplayOption* **subtotalWithPercentDiscount:** (NSString \*) *subtotalWithPercentDiscount* **error:** (NSError \*\*) *error*

33h(51) SUBTOTAL

- Data field: toPrintOption,toDisplayOption,subtotalWithPercentDiscount
- Response: subTotal,taxGroupA,taxGroupB,taxGroupC,taxGroupD,taxGroupE,specialTax
- toPrintOption One byte, which if '1' the sum of the subtotal will be printed out.
- toDisplayOption One byte, which if '1' the sum of the subtotal will be displayed out.
- subtotalWithPercentDiscount The value of discount or surcharge in percent over the sum accumulated so far.
- subTotal The sum accumulated for the current fiscal receipt (10 bytes).
- taxGroupA The sum over tax group A /10 bytes/
- taxGroupB The sum over tax group B /10 bytes/
- taxGroupC The sum over tax group C /10 bytes/

- taxGroupD The sum over tax group D /10 bytes/
- taxGroupE The sum over tax group E - VAT exempt /10 bytes/
- specialTax The sum over special tax /10 bytes/

The sum of all sales registered in the fiscal receipt is calculated. The calculated total sum and the accumulated separate sums for each tax group are returned to the PC.

#### Parameters

<i>toPrintOption</i>	- One byte, which if '1' the sum of the subtotal will be printed out.
<i>toDisplayOption</i>	- One byte, which if '1' the sum of the subtotal will be displayed out.
<i>subtotalWithPercentDiscount</i>	- The value of discount or surcharge in percent over the sum accumulated so far.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"subTotal" - The sum accumulated for the current fiscal receipt (10 bytes).
- KeyValue - @"taxGroupA" - The sum over tax group A /10 bytes/
- KeyValue - @"taxGroupB" - The sum over tax group B /10 bytes/
- KeyValue - @"taxGroupC" - The sum over tax group C /10 bytes/
- KeyValue - @"taxGroupD" - The sum over tax group D /10 bytes/
- KeyValue - @"taxGroupE" - The sum over tax group E - VAT exempt /10 bytes/
- KeyValue - @"specialTax" - The sum over special tax /10 bytes/

1.1.2.113 - (NSDictionary \*) command51Variant0Version2ToPrintOption: (NSString \*) *toPrintOption* toDisplayOption:(NSString \*) *toDisplayOption* subtotalWithAbsoluteSumDiscount:(NSString \*) *subtotalWithAbsoluteSumDiscount* error:(NSError \*\*) *error*

33h(51) SUBTOTAL

- Data field: toPrintOption,toDisplayOption,subtotalWithAbsoluteSumDiscount
- Response: subTotal,taxGroupA,taxGroupB,taxGroupC,taxGroupD,taxGroupE,specialTax
- toPrintOption One byte, which if '1' the sum of the subtotal will be printed out.
- toDisplayOption One byte, which if '1' the sum of the subtotal will be displayed out.
- subtotalWithAbsoluteSumDiscount The value of discount as absolute value (up to 8 digits).

The subtotal will be printed out.

- subTotal The sum accumulated for the current fiscal receipt (10 bytes).
- taxGroupA The sum over tax group A /10 bytes/
- taxGroupB The sum over tax group B /10 bytes/
- taxGroupC The sum over tax group C /10 bytes/
- taxGroupD The sum over tax group D /10 bytes/

- taxGroupE The sum over tax group E - VAT exempt /10 bytes/
- specialTax The sum over special tax /10 bytes/

The sum of all sales registered in the fiscal receipt is calculated. The calculated total sum and the accumulated separate sums for each tax group are returned to the PC.

#### Parameters

<i>toPrintOption</i>	- One byte, which if '1' the sum of the subtotal will be printed out.
<i>toDisplayOption</i>	- One byte, which if '1' the sum of the subtotal will be displayed out.
<i>subtotalWith-AbsoluteSum-Discount</i>	- The value of discount as absolute value (up to 8 digits).
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

##### NSDictionary

- KeyValue - @"subTotal" - The sum accumulated for the current fiscal receipt (10 bytes).
- KeyValue - @"taxGroupA" - The sum over tax group A /10 bytes/
- KeyValue - @"taxGroupB" - The sum over tax group B /10 bytes/
- KeyValue - @"taxGroupC" - The sum over tax group C /10 bytes/
- KeyValue - @"taxGroupD" - The sum over tax group D /10 bytes/
- KeyValue - @"taxGroupE" - The sum over tax group E - VAT exempt /10 bytes/
- KeyValue - @"specialTax" - The sum over special tax /10 bytes/

1.1.2.114 - (NSDictionary \*) **command53Variant0Version0PaidMode:** (NSString \*) *paidMode* amountIn:(NSString \*) *amountIn* error:(NSError \*\*) *error*

#### 35h(53) CALCULATION OF A TOTAL

- Data field:
  1. paidMode
  2. amountIn
- Response:
  1. paidCode
  2. amountOut
- paidMode Code indicating the terms of payment. It may have the following values:
  1. 'P' Payment in cash
  2. 'N' Payment via credit
  3. 'C' Payment in cheques
  4. 'D' Payment with a debit card
  5. 'I' Programmable payment 1
  6. 'J' Programmable payment 2
  7. 'K' Programmable payment 3

## 8. 'L' Programmable payment 4

- *amountIn* The sum tendered (up to 10 meaningful symbols)
  1. Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
- *paidCode* One byte - resulting from the execution of the command
  1. 'F' Error
  2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
  3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
  4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
  5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
- *amountOut* Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

## Parameters

<i>paidMode</i>	- Code indicating the terms of payment. It may have the following values: <ul style="list-style-type: none"> <li>• 'P' Payment in cash</li> <li>• 'N' Payment via credit</li> <li>• 'C' Payment in cheques</li> <li>• 'D' Payment with a debit card</li> <li>• 'I' Programmable payment 1</li> <li>• 'J' Programmable payment 2</li> <li>• 'K' Programmable payment 3</li> <li>• 'L' Programmable payment 4</li> </ul>
<i>amountIn</i>	- The sum tendered (up to 10 meaningful symbols). Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

**Notes:**

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.
  - KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

**1.1.2.115** - (NSDictionary \*) **command53Variant0Version1TextRow2:** (NSString \*) *textRow2* paidMode:(NSString \*) *paidMode* amountIn:(NSString \*) *amountIn* error:(NSError \*\*) *error*

**35h(53) CALCULATION OF A TOTAL**

- Data field:
  1. textRow2
  2. paidMode
  3. amountIn
- Response:
  1. paidCode
  2. amountOut
- textRow2 A text of 30 bytes containing the second line
- paidMode Code indicating the terms of payment. It may have the following values:
  1. 'P' Payment in cash
  2. 'N' Payment via credit
  3. 'C' Payment in cheques
  4. 'D' Payment with a debit card
  5. 'I' Programmable payment 1
  6. 'J' Programmable payment 2
  7. 'K' Programmable payment 3
  8. 'L' Programmable payment 4
- amountIn The sum tendered (up to 10 meaningful symbols)

1. Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
- `paidCode` One byte - resulting from the execution of the command
    1. 'F' Error
    2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
    3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
    4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
    5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
  - `amountOut` Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>textRow2</i>	- A text of 30 bytes containing the second line
<i>paidMode</i>	- Code indicating the terms of payment. It may have the following values: <ul style="list-style-type: none"> <li>• 'P' Payment in cash</li> <li>• 'N' Payment via credit</li> <li>• 'C' Payment in cheques</li> <li>• 'D' Payment with a debit card</li> <li>• 'I' Programmable payment 1</li> <li>• 'J' Programmable payment 2</li> <li>• 'K' Programmable payment 3</li> <li>• 'L' Programmable payment 4</li> </ul>
<i>amountIn</i>	- The sum tendered (up to 10 meaningful symbols). Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

**Notes:**

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

- KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

**1.1.2.116** - (NSDictionary \*) **command53Variant0Version2TextRow1:** (NSString \*) *textRow1* paidMode:(NSString \*) *paidMode* amountIn:(NSString \*) *amountIn* error:(NSError \*\*) *error*

**35h(53) CALCULATION OF A TOTAL**

- Data field:
  1. textRow1
  2. paidMode
  3. amountIn
- Response:
  1. paidCode
  2. amountOut
- textRow1 A text of 30 bytes containing the first line
- paidMode Code indicating the terms of payment. It may have the following values:
  1. 'P' Payment in cash
  2. 'N' Payment via credit
  3. 'C' Payment in cheques
  4. 'D' Payment with a debit card
  5. 'I' Programmable payment 1
  6. 'J' Programmable payment 2
  7. 'K' Programmable payment 3
  8. 'L' Programmable payment 4
- amountIn The sum tendered (up to 10 meaningful symbols)



1. Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
- `paidCode` One byte - resulting from the execution of the command
    1. 'F' Error
    2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
    3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
    4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
    5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
  - `amountOut` Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>textRow1</i>	- A text of 30 bytes containing the first line
<i>paidMode</i>	- Code indicating the terms of payment. It may have the following values: <ul style="list-style-type: none"> <li>• 'P' Payment in cash</li> <li>• 'N' Payment via credit</li> <li>• 'C' Payment in cheques</li> <li>• 'D' Payment with a debit card</li> <li>• 'I' Programmable payment 1</li> <li>• 'J' Programmable payment 2</li> <li>• 'K' Programmable payment 3</li> <li>• 'L' Programmable payment 4</li> </ul>
<i>amountIn</i>	- The sum tendered (up to 10 meaningful symbols). Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

**Notes:**

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

- KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

**1.1.2.117** - (NSDictionary \*) **command53Variant0Version4TextRow1:** (NSString \*) *textRow1* **textRow2:**(NSString \*) *textRow2* **paidMode:**(NSString \*) *paidMode* **amountIn:**(NSString \*) *amountIn* **error:**(NSError \*\*) *error*

35h(53) CALCULATION OF A TOTAL

- Data field:

1. textRow1
2. textRow2
3. paidMode
4. amountIn

- Response:

1. paidCode
2. amountOut

- textRow1 A text of 30 bytes containing the first line
- textRow2 A text of 30 bytes containing the second line
- paidMode Code indicating the terms of payment. It may have the following values:
  1. 'P' Payment in cash
  2. 'N' Payment via credit
  3. 'C' Payment in cheques
  4. 'D' Payment with a debit card
  5. 'I' Programmable payment 1
  6. 'J' Programmable payment 2
  7. 'K' Programmable payment 3
  8. 'L' Programmable payment 4

- **amountIn** The sum tendered (up to 10 meaningful symbols)
  1. Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
- **paidCode** One byte - resulting from the execution of the command
  1. 'F' Error
  2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
  3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
  4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
  5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
- **amountOut** Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>textRow1</i>	- A text of 30 bytes containing the first line
<i>textRow2</i>	- A text of 30 bytes containing the second line
<i>paidMode</i>	- Code indicating the terms of payment. It may have the following values: <ul style="list-style-type: none"> <li>• 'P' Payment in cash</li> <li>• 'N' Payment via credit</li> <li>• 'C' Payment in cheques</li> <li>• 'D' Payment with a debit card</li> <li>• 'I' Programmable payment 1</li> <li>• 'J' Programmable payment 2</li> <li>• 'K' Programmable payment 3</li> <li>• 'L' Programmable payment 4</li> </ul>
<i>amountIn</i>	- The sum tendered (up to 10 meaningful symbols). Depending on the code, the sums are accumulated in different registers and may be recovered in the daily report.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.
  - KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

1.1.2.118 - (NSDictionary \*) command53Variant1Version0AndReturnError: (NSError \*\*) error

35h(53) CALCULATION OF A TOTAL

Payment in cash!!!

- Data field: None
- Response:
  1. paidCode
  2. amountOut
- paidCode One byte - resulting from the execution of the command
  1. 'F' Error
  2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
  3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
  4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
  5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
- amountOut Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
  2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.
- KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

1.1.2.119 - (NSDictionary \*) command53Variant1Version1TextRow2: (NSString \*) textRow2 error:(NSError \*\*) error

35h(53) CALCULATION OF A TOTAL

Payment in cash!!!

- Data field:
  1. textRow2
- Response:
  1. paidCode
  2. amountOut
- textRow2 A text of 30 bytes containing the second line

- paidCode One byte - resulting from the execution of the command
  1. 'F' Error
  2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
  3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
  4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
  5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
- amountOut Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>textRow2</i>	- A text of 30 bytes containing the second line
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyVal - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.

2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

- KeyVal - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

1.1.2.120 - (NSDictionary \*) command53Variant1Version2TextRow1: (NSString \*) *textRow1* error:(NSError \*\*) *error*

35h(53) CALCULATION OF A TOTAL

Payment in cash!!!

- Data field:
  1. textRow1
- Response:
  1. paidCode
  2. amountOut
- textRow1 A text of 30 bytes containing the first line
- paidCode One byte - resulting from the execution of the command
  1. 'F' Error
  2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
  3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
  4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
  5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.
- amountOut Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>textRow1</i>	- A text of 30 bytes containing the first line
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

**Notes:**

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

- KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

1.1.2.121 - (NSDictionary \*) command53Variant1Version3TextRow1: (NSString \*) *textRow1* textRow2:(NSString \*) *textRow2* error:(NSError \*\*) *error*

**35h(53) CALCULATION OF A TOTAL**

Payment in cash!!!

- Data field:
  1. textRow1
  2. textRow2
- Response:
  1. paidCode
  2. amountOut
- textRow1 A text of 30 bytes containing the first line
- textRow2 A text of 30 bytes containing the second line
- paidCode One byte - resulting from the execution of the command
  1. 'F' Error
  2. 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
  3. 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
  4. 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
  5. 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.



- amountOut Up to 9 digits with a sign. Depends on PaidCode.

This command starts the calculation of the sums from fiscal receipt, the printing of the sum with a special font. An additional text may also be printed. When the command has been successfully executed a further command for opening a cash drawer is activated.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - The accumulated sum is negative,
- - If some of the accumulated sums under taxation (tax group) is negative.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

#### Parameters

<i>textRow1</i>	- A text of 30 bytes containing the first line
<i>textRow2</i>	- A text of 30 bytes containing the second line
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"paidCode" - One byte - resulting from the execution of the command
- 'F' Error
- 'E' The calculated sub-total sum is negative. Payment is withheld and Amount will contain a negative sub-total.
- 'D' If the paid sum is less than the sum on the receipt. The residual sum due for payment is returned to Amount
- 'R' When the paid sum is greater than the sum on the receipt. A message "CHANGE" will be printed out and the change will be returned to Amount.
- 'I' An error has occurred because the sum under one of the tax groups is negative. The current subtotal is returned to Amount.

Notes:

1. After the successful completion of the command, fiscal printer will not perform the commands 49 and 51 within the opened receipt, although it can still perform command 53.
2. The codes of error 'E' and 'I' will never appear in this version of the printer because commands 49 and 52 (registering a sale) do not accept negative sums.

- KeyValue - @"amountOut" - Up to 9 digits with a sign. Depends on PaidCode.

1.1.2.122 - (bool) command54Variant0Version0TextIn: (NSString \*) *textIn* error:(NSError \*\*) *error*

36h(54) PRINTING A FREE FISCAL TEXT

- Data field: textIn

- Response: None
- `textIn` Up to 30 bytes text

A fiscal receipt must be opened because, in the opposite case, the text will not be printed and the S1.1. flag is raised. If the text is longer than 30 symbols, the redundant letters are cut off.

#### Parameters

<i>textIn</i>	- Up to 30 bytes text
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

#### 1.1.2.123 - (NSDictionary \*) `command56Variant0Version0AndReturnError: (NSError **) error`

##### 38h(56) CLOSING A FISCAL RECEIPT

- Data field: No data
- Response: `allReceipt`, `fiscalReceipt`
- `allReceipt` All issued receipts from the last daily closure up to the moment
- `fiscalReceipt` All issued fiscal receipts from the last daily closure up to the moment

The accumulated sums from the fiscal receipt are added to the daily sums in the registries of operational memory.

The command will not be successful if:

- - No fiscal receipt has been opened,
- - Command 53 (35h) has failed,
- - The sum paid under command 53 is less than the total sum on the fiscal receipt.

#### Returns

NSDictionary

- Key-Value - @"allReceipt" - All issued receipts from the last daily closure up to the moment
- Key-Value - @"fiscalReceipt" - All issued fiscal receipts from the last daily closure up to the moment

#### 1.1.2.124 - (bool) `command58Variant0Version0SignPlu: (NSString *) signPlu itemQuantity:(NSString *) itemQuantity error:(NSError **) error`

##### 3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM

- Data field:
  1. `signPlu`
  2. `itemQuantity`

- Response:
  1. None
- *signPlu* The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
- *itemQuantity* The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (\*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.

The fiscal printer performs the following operations:

- - The name, price and tax group of the item is read from items list, programmed in the printer.
- - Prints out the name of the item, selected quantity and singular price. The second printed line contains the final price together with the letter, designating the tax group from which the sale will be performed. The registries for accumulated sums and item quantities are updated.
- - The price of the item is added to the accumulated sums in the registries of operational memory. In case of overflow, the respective bytes from the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is added in specially selected registries in the printer. The values from the whole day will be printed together with the daily financial report.

The command will not be successful if:

- - No item has been programmed under the given number,
- - No fiscal receipt has been opened,
- - The maximum number of sales for one receipt (380) has already been registered.
- - The command 35h has been successfully executed,
- - The sum under one or more of the tax groups has turned out negative,
- - The sum of surcharges and discounts within the receipt has remained negative,
- - The journal is full

#### Parameters

<i>signPlu</i>	- The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
<i>itemQuantity</i>	- The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.125 - (bool) `command58Variant0Version1SignPlu: (NSString *) signPlu itemQuantity:(NSString *) itemQuantity specialTax:(NSString *) specialTax error:(NSError **) error`

3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM

- Data field:
  1. signPlu
  2. itemQuantity
  3. specialTax
- Response:
  1. None
- signPlu The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
- itemQuantity The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (\*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
- specialTax The dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.

The fiscal printer performs the following operations:

- - The name, price and tax group of the item is read from items list, programmed in the printer.
- - Prints out the name of the item, selected quantity and singular price. The second printed line contains the final price together with the letter, designating the tax group from which the sale will be performed. The registries for accumulated sums and item quantities are updated.
- - The price of the item is added to the accumulated sums in the registries of operational memory. In case of overflow, the respective bytes from the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is added in specially selected registries in the printer. The values from the whole day will be printed together with the daily financial report.

The command will not be successful if:

- - No item has been programmed under the given number,
- - No fiscal receipt has been opened,
- - The maximum number of sales for one receipt (380) has already been registered.
- - The command 35h has been successfully executed,
- - The sum under one or more of the tax groups has turned out negative,
- - The sum of surcharges and discounts within the receipt has remained negative,
- - The journal is full

#### Parameters

<i>signPlu</i>	- The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
<i>itemQuantity</i>	- The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
<i>specialTax</i>	- The dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
<i>error</i>	pointer to NSError object, where error information is stored in case of function fails. You can pass nil if you don't want that information

**Returns**

TRUE upon success, FALSE otherwise

1.1.2.126 - (bool) `command58Variant1Version0SignPlu: (NSString *) signPlu itemQuantity:(NSString *) itemQuantity sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

**3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM**

- Data field:
  1. signPlu
  2. itemQuantity
  3. sellWithAbsoluteSumDiscount
- Response:
  1. None
- signPlu The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
- itemQuantity The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (\*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
- sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the current sale. Up to 8 significant digits.

The fiscal printer performs the following operations:

- - The name, price and tax group of the item is read from items list, programmed in the printer.
- - Prints out the name of the item, selected quantity and singular price. The second printed line contains the final price together with the letter, designating the tax group from which the sale will be performed. The registries for accumulated sums and item quantities are updated.
- - The price of the item is added to the accumulated sums in the registries of operational memory. In case of overflow, the respective bytes from the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is added in specially selected registries in the printer. The values from the whole day will be printed together with the daily financial report.

The command will not be successful if:

- - No item has been programmed under the given number,
- - No fiscal receipt has been opened,
- - The maximum number of sales for one receipt (380) has already been registered.
- - The command 35h has been successfully executed,
- - The sum under one or more of the tax groups has turned out negative,
- - The sum of surcharges and discounts within the receipt has remained negative,
- - The journal is full

**Parameters**

<i>signPlu</i>	- The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
<i>itemQuantity</i>	- The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the current sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.127 - (bool) `command58Variant1Version1SignPlu: (NSString *) signPlu itemQuantity:(NSString *) itemQuantity specialTax:(NSString *) specialTax sellWithAbsoluteSumDiscount:(NSString *) sellWithAbsoluteSumDiscount error:(NSError **) error`

#### 3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM

- Data field:
  1. signPlu
  2. itemQuantity
  3. specialTax
  4. sellWithAbsoluteSumDiscount
- Response:
  1. None
- signPlu The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
- itemQuantity The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (\*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
- specialTax The dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
- sellWithAbsoluteSumDiscount The value of discount or surcharge (depending on the sign) over the current sale. Up to 8 significant digits.

The fiscal printer performs the following operations:

- - The name, price and tax group of the item is read from items list, programmed in the printer.
- - Prints out the name of the item, selected quantity and singular price. The second printed line contains the final price together with the letter, designating the tax group from which the sale will be performed. The registries for accumulated sums and item quantities are updated.
- - The price of the item is added to the accumulated sums in the registries of operational memory. In case of overflow, the respective bytes from the status field will be set.

- - If there is a discount or surcharge, it is printed out on a separate line and is added in specially selected registries in the printer. The values from the whole day will be printed together with the daily financial report.

The command will not be successful if:

- - No item has been programmed under the given number,
- - No fiscal receipt has been opened,
- - The maximum number of sales for one receipt (380) has already been registered.
- - The command 35h has been successfully executed,
- - The sum under one or more of the tax groups has turned out negative,
- - The sum of surcharges and discounts within the receipt has remained negative,
- - The journal is full

#### Parameters

<i>signPlu</i>	- The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-" ).
<i>itemQuantity</i>	- The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
<i>specialTax</i>	- The dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the current sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.128 - (bool) `command58Variant2Version0SignPlu: (NSString *) signPlu itemQuantity:(NSString *) itemQuantity sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

#### 3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM

- Data field:
  1. signPlu
  2. itemQuantity
  3. sellWithPercentDiscount
- Response:
  1. None
- signPlu The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-" ).

- **itemQuantity** The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (\*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
- **sellWithPercentDiscount** The value of surcharge or discount (depending on the symbol) in percent over the current sale. Possible values are between -99.00% to 99.00%. Up to 2 digits after the decimal point are acceptable.

The fiscal printer performs the following operations:

- - The name, price and tax group of the item is read from items list, programmed in the printer.
- - Prints out the name of the item, selected quantity and singular price. The second printed line contains the final price together with the letter, designating the tax group from which the sale will be performed. The registries for accumulated sums and item quantities are updated.
- - The price of the item is added to the accumulated sums in the registries of operational memory. In case of overflow, the respective bytes from the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is added in specially selected registries in the printer. The values from the whole day will be printed together with the daily financial report.

The command will not be successful if:

- - No item has been programmed under the given number,
- - No fiscal receipt has been opened,
- - The maximum number of sales for one receipt (380) has already been registered.
- - The command 35h has been successfully executed,
- - The sum under one or more of the tax groups has turned out negative,
- - The sum of surcharges and discounts within the receipt has remained negative,
- - The journal is full

#### Parameters

<i>signPlu</i>	- The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-" ).
<i>itemQuantity</i>	- The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
<i>sellWithPercent-Discount</i>	- The value of surcharge or discount (depending on the symbol) in percent over the current sale. Possible values are between -99.00% to 99.00%. Up to 2 digits after the decimal point are acceptable.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information



## Returns

TRUE upon success, FALSE otherwise

1.1.2.129 - (bool) `command58Variant2Version1SignPlu: (NSString *) signPlu itemQuantity:(NSString *) itemQuantity specialTax:(NSString *) specialTax sellWithPercentDiscount:(NSString *) sellWithPercentDiscount error:(NSError **) error`

## 3Ah(58) REGISTERING THE SALE OF AN PROGRAMMED ITEM

- Data field:
  1. signPlu
  2. itemQuantity
  3. specialTax
  4. sellWithPercentDiscount
- Response:
  1. None
- signPlu The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
- itemQuantity The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (\*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
- specialTax The dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report an periodical reports, if not 0.
- sellWithPercentDiscount The value of surcharge or discount (depending on the symbol) in percent over the current sale. Possible values are between -99.00% to 99.00%. Up to 2 digits after the decimal point are acceptable.

The fiscal printer performs the following operations:

- - The name, price and tax group of the item is read from items list, programmed in the printer.
- - Prints out the name of the item, selected quantity and singular price. The second printed line contains the final price together with the letter, designating the tax group from which the sale will be performed. The registries for accumulated sums and item quantities are updated.
- - The price of the item is added to the accumulated sums in the registries of operational memory. In case of overflow, the respective bytes from the status field will be set.
- - If there is a discount or surcharge, it is printed out on a separate line and is added in specially selected registries in the printer. The values from the whole day will be printed together with the daily financial report.

The command will not be successful if:

- - No item has been programmed under the given number,
- - No fiscal receipt has been opened,
- - The maximum number of sales for one receipt (380) has already been registered.
- - The command 35h has been successfully executed,

- - The sum under one or more of the tax groups has turned out negative,
- - The sum of surcharges and discounts within the receipt has remained negative,
- - The journal is full

#### Parameters

<i>signPlu</i>	- The individual number of the item - a whole number between 1 and 999999999 (not more than 9 digits)(with sign "+" or "-").
<i>itemQuantity</i>	- The quantity of the items for sale with a default value of 1.000. Length cannot be longer than 8 meaningful digits (not more than 3 after the decimal point). The resulting singular price (*Quan) is rounded up to the set number of digits after the decimal point and also cannot be greater than 9 meaningful digits.
<i>specialTax</i>	- The dimensions of a single price. This value is subtracted from the single price before applying any VAT calculations and is printed on a separate line in the receipt. The accumulated value is stored in the fiscal memory and is printed in the Z-report and periodical reports, if not 0.
<i>sellWith-AbsoluteSum-Discount</i>	- The value of discount or surcharge (depending on the sign) over the current sale. Up to 8 significant digits.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.130 - (bool) `command60Variant0Version0AndReturnError: (NSError **) error`

3Ch(60) CANCEL FISCAL RECEIPT

- Data field: None
- Response: None

The command cancels an open fiscal receipt. All sales in the receipt are discarded. The message "===CANCELLED===" is printed and then the receipt is closed as non-fiscal. The command is not permitted, if command 53 (Total) is already executed for this receipt.

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.131 - (bool) `command61Variant0Version0TargetDate: (NSString *) targetDate targetTime:(NSString *) targetTime error:(NSError **) error`

3Dh(61) SETTING THE CLOCK - DATE AND HOUR

- Data field:
  1. targetDate
  2. targetTime
- Response: None

You cannot set a date, which is earlier than the date of the last entry into the fiscal memory of device and the capacity of this memory includes the year 2099. After RESET of memory, this command must be executed - otherwise, the normal functioning of device cannot be resumed. The printer's real-time clock must always be set correctly.

#### Parameters

<i>targetDate</i>	- DD-MM-YY
<i>targetTime</i>	- HH:MM[:SS]
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.132 - (NSDictionary \*) command62Variant0Version0AndReturnError: (NSError \*\*) error

3Eh (62) READING CURRENT DATE AND HOUR

- Data field: None
- Response:
  1. outputText DD-MM-YY HH:MM:SS

#### Returns

NSDictionary

- Key-Value - @"outputText" - DD-MM-YY HH:MM:SS

1.1.2.133 - (NSDictionary \*) command64Variant0Version0AndReturnError: (NSError \*\*) error

40h(64) LAST FISCAL CLOSURE DETAILS

- Data field: None
- Response:
  1. errorCode
  2. receiptCount
  3. totalSumInTaxGroupA
  4. totalSumInTaxGroupB
  5. totalSumInTaxGroupC
  6. totalSumInTaxGroupD
  7. totalSumInTaxGroupE
  8. specialTaxSum
  9. fiscalRecordDate
- errorCode:
  1. 'P' Successful command. Data present after ',' symbol.
  2. 'F' Can't read last record. No data present.

- receiptCount Receipt count
- totalSumInTaxGroupX VAT group total (12 bytes with sign).
- specialTax Special tax sum (12 bytes with sign).
- fiscalRecordDate Closure date in format DDMMYY.

#### Returns

##### NSDictionary

- KeyValue - @"errorCode" - 'P' Successful command. Data present after ',' symbol. 'F' Can't read last record. No data present.
- KeyValue - @"receiptCount" - Receipt count
- KeyValue - @"totalSumInTaxGroupA" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupB" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupC" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupD" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupE" - VAT group total (12 bytes with sign).
- KeyValue - @"specialTax" - Special tax sum (12 bytes with sign).
- KeyValue - @"fiscalRecordDate" - Closure date in format DDMMYY.

1.1.2.134 - (NSDictionary \*) command65Variant0Version0AndReturnError: (NSError \*\*) error

#### 41h(65) DAILY TOTALS

- Data field: None
- Response:
  1. totalSumInTaxGroupA
  2. totalSumInTaxGroupB
  3. totalSumInTaxGroupC
  4. totalSumInTaxGroupD
  5. totalSumInTaxGroupE
  6. specialTax
- totalSumInTaxGroupX VAT group total (12 bytes with sign).
- specialTax Special tax sum (12 bytes with sign).

#### Returns

##### NSDictionary

- KeyValue - @"totalSumInTaxGroupA" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupB" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupC" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupD" - VAT group total (12 bytes with sign).
- KeyValue - @"totalSumInTaxGroupE" - VAT group total (12 bytes with sign).
- KeyValue - @"specialTax" - Special tax sum (12 bytes with sign).

1.1.2.135 - (NSDictionary \*) command68Variant0Version0AndReturnError: (NSError \*\*) error

44h (68) THE NUMBER OF FREE FIELDS IN THE FISCAL MEMORY

- Data field: None
- Response:
  1. logicalFiscalRecordsCount
- logicalFiscalRecordsCount The number of logical locations for fiscal entries (4 bytes)

The number of free fields in the fiscal memory, reserved for saving information from the daily report.

#### Returns

NSDictionary

- KeyValue - @"logicalFiscalRecordsCount" - The number of logical locations for fiscal entries (4 bytes)

1.1.2.136 - (NSDictionary \*) command69Variant0Version0ReportTypeOption: (NSString \*) reportTypeOption error:(NSError \*\*) error

45h(69) DAILY FINANCIAL REPORT

- Data field:
  1. reportTypeOption
- Response:
  1. fiscalRecordNumber
  2. totalSumForTheDay
  3. totalSumInTaxGroupA,
  4. totalSumInTaxGroupB,
  5. totalSumInTaxGroupC,
  6. totalSumInTaxGroupD,
  7. totalSumInTaxGroupE,
  8. totalSumInSpecialTax
- reportTypeOption Parameter controlling the type of generated report.
  1. '0' A Z-report is printed. The printout ends with inscriptions "NON-FISCAL RECEIPT" if the printer is not fiscalised.
  2. '2' A X-report is generated, i.e., no entry into the fiscal memory is made and no closures are performed. The printout ends with inscription "NON-FISCAL RECEIPT". The same actions may be generated directly from the printer if during switching on the "FEED" button is hold for 2 to 4 seconds.
  3. N The presence of this symbol at the end of the data cancels the option to clear the data accumulated on the operators during a Z-report.
  4. A The presence of this symbol at the end of the data cancels the option to clear the data about sold article quantities during a Z-report.
- fiscalRecordNumber Fiscal closure (Daily report) number - 4 bytes.
- totalSumForTheDay The sum of all sales for the day - 12 bytes with a sign.

- `totalSumInTaxGroupX` The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- `totalSumInSpecialTax` Special tax sum (12 bytes with sign).

The command with option '0' (Z-report) must be executed immediately after printing and deleting the electronic journal. If there is information in the journal, the command is not permitted.

#### Parameters

<i>reportType-Option</i>	- Parameter controlling the type of generated report. <ul style="list-style-type: none"> <li>• '0' A Z-report is printed. The printout ends with inscriptions "NON-FISCAL RECEIPT" if the printer is not fiscalised.</li> <li>• '2' A X-report is generated, i.e., no entry into the fiscal memory is made and no closures are performed. The printout ends with inscription "NON-FISCAL RECEIPT". The same actions may be generated directly from the printer if during switching on the &lt;FEED&gt; button is hold for 2 to 4 seconds.</li> <li>• N The presence of this symbol at the end of the data cancels the option to clear the data accumulated on the operators during a Z-report.</li> <li>• A The presence of this symbol at the end of the data cancels the option to clear the data about sold article quantities during a Z-report.</li> </ul>
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

##### NSDictionary

- Key-Value - @"fiscalRecordNumber" - Fiscal closure (Daily report) number - 4 bytes.
- Key-Value - @"totalSumForTheDay" - The sum of all sales for the day - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupA" - The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupB" - The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupC" - The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupD" - The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- Key-Value - @"totalSumInTaxGroupE" - The totals under all tax categories - A, B, C, D and E - 12 bytes with a sign.
- Key-Value - @"totalSumInSpecialTax" - TotF The total under tax category - 12 bytes with a sign.

**1.1.2.137** - (NSDictionary \*) `command70Variant0Version0AmountInOut:` (NSString \*) *amountInOut* error:(NSError \*\*) *error*

46h(70) INTERNAL DEBITING AND CREDITING (service In and Out)

- Data field:
  1. `amountInOut`
- Response:
  1. `exitCode`
  2. `sumInCashRegister`

- 3. totalForAllInputs
- 4. totalForOutputs

- amountInOut The sum, which will be registered (up to 9 bytes). Depending on the sign of the digit, this sum is interpreted either as credit or debit (serveIn or serveOut).
- exitCode:
  1. 'P' The order has been completed. If the ordered sum is not 0, the printer will print an interior receipt for registering the operation.
  2. 'F' The order has been canceled. This happens if:
    - (a) - the cash sum available is less than the ordered interior credit (serveIn),
    - (b) - there is an opened fiscal and non-fiscal receipt.
- sumInCashRegister Available cash. Apart from this command, the sum grows after each payment in cash.
- totalForAllInputs The sum from all commands "Interior credit"
- totalForOutputs The sum from all commands "Interior debit"

Changes the content of the cash availability register. Depending on the sign of the sum in question, it is accumulated in the register for interior debit-credit. The information is not saved in the fiscal memory of device and is accessible until the performance of the daily closure. It is printed out at the command 69 (45h) and at the generation of the daily report without closure from the printer itself. At successful completion of this command, the drawer "kick-out" function is automatically activated.

#### Parameters

<i>amountInOut</i>	- The sum, which will be registered (up to 9 bytes). Depending on the sign of the digit, this sum is interpreted either as credit or debit (serveIn or serveOut).
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- Key-Value - @"exitCode" - 'P' The order has been completed. If the ordered sum is not 0, the printer will print an interior receipt for registering the operation. 'F' The order has been canceled. This happens if:
  1. - the cash sum available is less than the ordered interior credit (serveIn),
  2. - there is an opened fiscal and non-fiscal receipt.
- Key-Value - @"sumInCashRegister" - Available cash. Apart from this command, the sum grows after each payment in cash.
- Key-Value - @"totalForAllInputs" - The sum from all commands "Interior credit"
- Key-Value - @"totalForOutputs" - The sum from all commands "Interior debit"

1.1.2.138 - (NSDictionary \*) command70Variant0Version1AndReturnError: (NSError \*\*) error

46h(70) INTERNAL DEBITING AND CREDITING (read only)

- Data field: None
- Response:

1. exitCode
  2. sumInCashRegister
  3. totalForAllInputs
  4. totalForOutputs
- amountInOut The sum, which will be registered (up to 9 bytes). Depending on the sign of the digit, this sum is interpreted either as credit or debit (serveIn or serveOut).
  - exitCode:
    1. 'P' The order has been completed. If the ordered sum is not 0, the printer will print an interior receipt for registering the operation.
    2. 'F' The order has been canceled. This happens if:
      - (a) - the cash sum available is less than the ordered interior credit (serveIn),
      - (b) - there is an opened fiscal and non-fiscal receipt.
  - sumInCashRegister Available cash. Apart from this command, the sum grows after each payment in cash.
  - totalForAllInputs The sum from all commands "Interior credit"
  - totalForOutputs The sum from all commands "Interior debit"

Changes the content of the cash availability register. Depending on the sign of the sum in question, it is accumulated in the register for interior debit-credit. The information is not saved in the fiscal memory of device and is accessible until the performance of the daily closure. It is printed out at the command 69 (45h) and at the generation of the daily report without closure from the printer itself. At successful completion of this command, the drawer "kick-out" function is automatically activated.

#### Returns

##### NSDictionary

- KeyValue - @"exitCode" - 'P' The order has been completed. If the ordered sum is not 0, the printer will print an interior receipt for registering the operation. 'F' The order has been canceled. This happens if:
  1. - the cash sum available is less than the ordered interior credit (serveIn),
  2. - there is an opened fiscal and non-fiscal receipt.
- KeyValue - @"sumInCashRegister" - Available cash. Apart from this command, the sum grows after each payment in cash.
- KeyValue - @"totalForAllInputs" - The sum from all commands "Interior credit"
- KeyValue - @"totalForOutputs" - The sum from all commands "Interior debit"

1.1.2.139 - (bool) command71Variant0Version0AndReturnError: (NSError \*\*) error

#### 47h(71) PRINTING DIAGNOSTIC INFORMATION

- Data field: No data
- Response: None

The command initiates the generation of an interior receipt containing diagnostic information as follows:

- - Prints the date and version of the employed software,
- - Prints the control sum of the employed firmware,



- - Prints the serial port's band rate,
- - Prints out the status of memory switches,
- - Prints emergency time after power supply cut-off,
- - Prints the number, date and hour of the last reset of the RAM (if there is such),
- - Prints the current temperature of the two printer heads,
- - Prints the overall number of fields in the fiscal memory and the number of the free fields,
- - Prints the current date and hour.

The command will not be executed when there is an open receipt in progress or when the paper roll has finished. It may also be activated by pressing the "FEED" button while power on for less than 2 seconds.

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.140 - (bool) `command73Variant0Version0StartRecordNumber: (NSString *) startRecordNumber endRecordNumber:(NSString *) endRecordNumber error:(NSError **) error`

#### 49H (73) DETAILED FISCAL MEMORY REPORT BY CLOSURE NUMBER

- Data field:
  1. startRecordNumber
  2. endRecordNumber
- Response: None
- startRecordNumber The number of the starting fiscal entry - 4 bytes
- endRecordNumber The number of the ending fiscal entry - 4 bytes

The command leads to printing of a detailed report of the fiscal memory from one selected number to another.

#### Parameters

<i>startRecord-Number</i>	- The number of the starting fiscal entry - 4 bytes
<i>endRecord-Number</i>	- The number of the ending fiscal entry - 4 bytes
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.141 - (NSDictionary \*) `command74Variant0Version0AndReturnError: (NSError **) error`

#### 4Ah(74) READING THE STATUS BYTES

- Data field: None

- Response:
  1. statusByte0 Status byte 0
  2. statusByte1 Status byte 1
  3. statusByte2 Status byte 2
  4. statusByte3 Status byte 3
  5. statusByte4 Status byte 4
  6. statusByte5 Status byte 5

All printer buffers must be printed out first.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

NSDictionary

- KeyValue - @"statusByte0" - statusByte0
- KeyValue - @"statusByte1" - statusByte1
- KeyValue - @"statusByte2" - statusByte2
- KeyValue - @"statusByte3" - statusByte3
- KeyValue - @"statusByte4" - statusByte4
- KeyValue - @"statusByte5" - statusByte5

1.1.2.142 - (NSDictionary \*) command74Variant1Version0AndReturnError: (NSError \*\*) *error*

#### 4Ah(74) READING THE STATUS BYTES

- Data field: None
- Response:
  1. statusByte0 Status byte 0
  2. statusByte1 Status byte 1
  3. statusByte2 Status byte 2
  4. statusByte3 Status byte 3
  5. statusByte4 Status byte 4
  6. statusByte5 Status byte 5

The status is returned immediately (default).

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

NSDictionary

- KeyValue - @"statusByte0" - statusByte0

- KeyValue - @"statusByte1" - statusByte1
- KeyValue - @"statusByte2" - statusByte2
- KeyValue - @"statusByte3" - statusByte3
- KeyValue - @"statusByte4" - statusByte4
- KeyValue - @"statusByte5" - statusByte5

1.1.2.143 - (NSDictionary \*) command76Variant0Version0AndReturnError: (NSError \*\*) *error*

#### 4Ch(76) STATUS OF THE FISCAL TRANSACTION

- Data field: None
- Repsonse:
  1. receiptIsOpened
  2. countOfRegisteredSales
  3. lastFiscalReceiptAmount
  4. lastFiscalReceiptTender

The command will return the information on the current state of the sum due for payment by client.

- receiptIsOpened One byte:
  1. '1' - if a fiscal or a non-fiscal receipt has been opened (which can be understood from the status bytes);
  2. '0' - if there is no opened receipt;
- countOfRegisteredSales The number of sales registered on the current or last fiscal receipt - 4 bytes.
- lastFiscalReceiptAmount The sum from the last fiscal receipt - 9 bytes with a sign.
- lastFiscalReceiptTender The sum tendered against the current or the last receipt - 9 bytes with a sign.

The command supports the PC application's ability to monitor the status and, if needed, to restore and complete an already started fiscal operation, which has been interrupted on emergency or out of time - for example, as a result of a power failure.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

NSDictionary

- KeyValue - @"receiptIsOpened" - One byte, which is
  - '1' - if a fiscal or a non-fiscal receipt has been opened (which can be understood from the status bytes);
  - '0' - if there is no opened receipt;
- KeyValue - @"countOfRegisteredSales" - The number of sales registered on the current or last fiscal receipt - 4 bytes.
- KeyValue - @"lastFiscalReceiptAmount" - The sum from the last fiscal receipt - 9 bytes with a sign.
- KeyValue - @"lastFiscalReceiptTender" - The sum tendered against the current or the last receipt - 9 bytes with a sign.

1.1.2.144 - (bool) **command79Variant0Version0StartDate:** (NSString \*) *startDate* **endDate:**(NSString \*) *endDate* **error:**(NSError \*\*) *error*

#### 4Fh(79) SHORT FISCAL MEMORY REPORT BY CLOSURE DATE

- Data field:
  1. startDate
  2. endDate
- Response: None
- startDate Starting date - 6 bytes (DDMMYY)
- endDate End date - 6 bytes (DDMMYY)

The command generates printing out of a short periodic financial report.

#### Parameters

<i>startDate</i>	- Starting date - 6 bytes (DDMMYY)
<i>endDate</i>	- End date - 6 bytes (DDMMYY)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.145 - (bool) **command80Variant0Version0SoundData:** (NSString \*) *soundData* **error:**(NSError \*\*) *error*

#### 50h(80) SOUND SIGNAL

- Data field:
  1. soundData
- Response: None

If the input data looks like: <Hz>,<mSec> where Hz and mSec are integer numbers, then a sound signal with frequency Hz (100-5000) and duration mSec milliseconds (50-2000) is generated. In all other cases data must be in format, similar to the one used for writing notes and can be of any length up to 218 bytes. The first invalid character cancels the command. SoundData format is a sequence of the following subcommands:

- Notes of the scale: One latine letter with value from 'A' to 'G'.
  1. 'C' - Do
  2. 'D' - Re
  3. 'E' - Mi
  4. 'F' - Fa
  5. 'G' - Sol
  6. 'A' - La
  7. 'B' - Si
  8. If immediately after the note comes character '#', then the note is higher in pitch by a semitone (sharp).  
If immediately after the note comes character '&', then the note is lower in pitch by a semitone (flat).

- Pause: Character space (ASCII 20h).
- After a note or pause there can be one or a few bytes, which specify the duration. Valid are characters from '0' to '5', they have the following meaning:
  1. '0' basic duration of a note/pause
  2. '1' basic duration \* 2
  3. '2' basic duration \* 4
  4. '3' basic duration \* 8
  5. '4' basic duration \* 16
  6. '5' basic duration \* 32
  7. If there are a few durations one after another they are summed up.
- Going to higher scale: character '+'.
- Going to lower scale: character '-'.
- Specifying tempo: character '^', followed by a number. The number specifies the percentage: duration of notes and intervals to basic duration. Values:
  1. '1' 200 %
  2. '2' 175 %
  3. '3' 140 %
  4. '4' 120 %
  5. '5' 100 %
  6. '6' 80 %
  7. '7' 60 %
  8. '8' 50 %
  9. '9' 40 %
- Return to scale 1 (it is default). Character '@'. Tone 'La' in it is 440 Hz.

#### Parameters

<i>soundData</i>	-
------------------	---

If the input data looks like: <Hz>,<mSec> where Hz and mSec are integer numbers, then a sound signal with frequency Hz (100-5000) and duration mSec milliseconds (50-2000) is generated. In all other cases data must be in format, similar to the one used for writing notes and can be of any length up to 218 bytes. The first invalid character cancels the command. SoundData format is a sequence of the following subcommands:

- Notes of the scale: One latine letter with value from 'A' to 'G'.
  1. 'C' - Do
  2. 'D' - Re
  3. 'E' - Mi
  4. 'F' - Fa
  5. 'G' - Sol
  6. 'A' - La
  7. 'B' - Si
  8. If immediately after the note comes character '#', then the note is higher in pitch by a semitone (sharp).  
If immediately after the note comes character '&', then the note is lower in pitch by a semitone (flat).

- Pause: Character space (ASCII 20h).
- After a note or pause there can be one or a few bytes, which specify the duration. Valid are characters from '0' to '5', they have the following meaning:
  1. '0' basic duration of a note/pause
  2. '1' basic duration \* 2
  3. '2' basic duration \* 4
  4. '3' basic duration \* 8
  5. '4' basic duration \* 16
  6. '5' basic duration \* 32
  7. If there are a few durations one after another they are summed up.
- Going to higher scale: character '+'.
- Going to lower scale: character '-'.
- Specifying tempo: character '^', followed by a number. The number specifies the percentage: duration of notes and intervals to basic duration. Values:
  1. '1' 200 %
  2. '2' 175 %
  3. '3' 140 %
  4. '4' 120 %
  5. '5' 100 %
  6. '6' 80 %
  7. '7' 60 %
  8. '8' 50 %
  9. '9' 40 %
- Return to scale 1 (it is default). Character '@'. Tone 'La' in it is 440 Hz.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.146** - (NSDictionary \*) **command83Variant0Version0InputMultiplier:** (NSString \*) *inputMultiplier* *inputDecimals*:(NSString \*) *inputDecimals* *inputCurrency*:(NSString \*) *inputCurrency* *inputEnabledTaxesArray*:(NSString \*) *inputEnabledTaxesArray* *inputTaxGroupA*:(NSString \*) *inputTaxGroupA* *inputTaxGroupB*:(NSString \*) *inputTaxGroupB* *inputTaxGroupC*:(NSString \*) *inputTaxGroupC* *inputTaxGroupD*:(NSString \*) *inputTaxGroupD* *error*:(NSError \*\*) *error*

53h(83) SETTING THE MULTIPLIER, DECIMALS, CURRENCY NAME AND DISABLED TAXES

- Data fields:
  1. inputMultiplier
  2. inputDecimals
  3. inputCurrency

4. inputEnabledTaxesArray
5. inputTaxGroupA
6. inputTaxGroupB
7. inputTaxGroupC
8. inputTaxGroupD

- Response:

1. outputMultiplier
2. outputDecimals
3. outputCurrency
4. outputEnabledTaxesArray
5. outputTaxGroupA
6. outputTaxGroupB
7. outputTaxGroupC
8. outputTaxGroupD

- (input/output)Multiplier A multiplier between 0 and 3 which shows the degree of 10 before multiplying it times the input or output value (at present deactivated and out of use).
- (input/output)Decimals One byte with a value 0 or 2 and shows the exact place of the decimal point.
- (input/output)Currency The currency name. Up to 6 bytes.
- (input/output)EnabledTaxesArray 4 bytes with value '0' or '1', corresponding to VAT groups 'A', 'B', 'C', 'D'. '0' means disabled VAT group, '1' - enabled VAT group.
- (input/output)TaxGroupX The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).

If nothing is entered in the data field, the FP returns the currently valid values. Even when only one of the parameters must be changed, the rest must be entered too. The fiscal memory has a fixed capacity for a set number of entries, and for that reason the command can be performed not more than 20 times after the fiscalisation. Before the fiscalisation the data are hold in RAM only and may be changed without limitations. The command may be executed only before the first fiscal receipt for the day.

#### Parameters

<i>inputMultiplier</i>	- A multiplier between 0 and 3 which shows the degree of 10 before multiplying it times the input or output value (at present deactivated and out of use).
<i>inputDecimals</i>	- One byte with a value 0 or 2 and shows the exact place of the decimal point.
<i>inputCurrency</i>	- The currency name. Up to 6 bytes.
<i>inputEnabled-TaxesArray</i>	- 4 bytes with value '0' or '1', corresponding to VAT groups 'A', 'B', 'C', 'D'. '0' means disabled VAT group, '1' - enabled VAT group.
<i>inputTaxGroupA</i>	- The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
<i>inputTaxGroupB</i>	- The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
<i>inputTaxGroupC</i>	- The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
<i>inputTaxGroupD</i>	- The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

##### NSDictionary

- Key-Value - @"outputMultiplier" - A multiplier between 0 and 3 which shows the degree of 10 before multiplying it times the input or output value (at present deactivated and out of use).

- KeyValue - @"outputDecimals" - One byte with a value 0 or 2 and shows the exact place of the decimal point.
- KeyValue - @"outputCurrency" - The currency name. Up to 6 bytes.
- KeyValue - @"outputEnabledTaxesArray" - 4 bytes with value '0' or '1', corresponding to VAT groups 'A', 'B', 'C', 'D'. '0' means disabled VAT group, '1' - enabled VAT group.
- KeyValue - @"outputTaxGroupA" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
- KeyValue - @"outputTaxGroupB" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
- KeyValue - @"outputTaxGroupC" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
- KeyValue - @"outputTaxGroupD" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).

1.1.2.147 - (NSDictionary \*) command83Variant1Version0AndReturnError: (NSError \*\*) error

53h(83) SETTING THE MULTIPLIER, DECIMALS, CURRENCY NAME AND DISABLED TAXES

- Data fields: None
- Response:
  1. outputMultiplier
  2. outputDecimals
  3. outputCurrency
  4. outputEnabledTaxesArray
  5. outputTaxGroupA
  6. outputTaxGroupB
  7. outputTaxGroupC
  8. outputTaxGroupD
- (input/output)Multiplier A multiplier between 0 and 3 which shows the degree of 10 before multiplying it times the input or output value (at present deactivated and out of use).
- (input/output)Decimals One byte with a value 0 or 2 and shows the exact place of the decimal point.
- (input/output)Currency The currency name. Up to 6 bytes.
- (input/output)EnabledTaxesArray 4 bytes with value '0' or '1', corresponding to VAT groups 'A', 'B', 'C', 'D'. '0' means disabled VAT group, '1' - enabled VAT group.
- (input/output)TaxGroupX The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).

If nothing is entered in the data field, the FP returns the currently valid values. Even when only one of the parameters must be changed, the rest must be entered too. The fiscal memory has a fixed capacity for a set number of entries, and for that reason the command can be performed not more than 20 times after the fiscalisation. Before the fiscalisation the data are hold in RAM only and may be changed without limitations. The command may be executed only before the first fiscal receipt for the day.



## Returns

## NSDictionary

- Key/Value - @"outputMultiplier" - A multiplier between 0 and 3 which shows the degree of 10 before multiplying it times the input or output value (at present deactivated and out of use).
- Key/Value - @"outputDecimals" - One byte with a value 0 or 2 and shows the exact place of the decimal point.
- Key/Value - @"outputCurrency" - The currency name. Up to 6 bytes.
- Key/Value - @"outputEnabledTaxesArray" - 4 bytes with value '0' or '1', corresponding to VAT groups 'A', 'B', 'C', 'D'. '0' means disabled VAT group, '1' - enabled VAT group.
- Key/Value - @"outputTaxA" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
- Key/Value - @"outputTaxB" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
- Key/Value - @"outputTaxC" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).
- Key/Value - @"outputTaxD" - The VAT rate for the corresponding VAT group in % with up to 2 decimals (0.00 to 99.00).

1.1.2.148 - (bool) command84Variant0Version0BarcodeType: (NSString \*) *barcodeType* barcodeData:(NSString \*) *barcodeData* error:(NSError \*\*) *error*

## 54h(84) PRINTING A BAR CODE

The command prints a bar code. It is permitted in an open fiscal or non-fiscal receipt. Bar code is printed centered. Note: Below the bar code is printed the information as text.

- Data field:
  1. barcodeType
  2. barcodeData
- Response: None
- barcodeType Bar code type. One byte with value:
  1. '1' EAN8. Data contain only digits and is 7 bytes long. Check sum is calculated by the printer.
  2. '2' EAN13. Data contain only digits and is 12 bytes long. Check sum is calculated by the printer.
  3. '3' Code 128. Data contain symbols with ASCII from 32 to 127. Length is from 9 to 18 symbols (depends on the contents - maximum length is possible if all symbols are digits). Check sum is calculated by the printer.
  4. '4' ITF (Interleaved 2 of 5). Data contain only digits.
  5. '5' ITF (Interleaved 2 of 5). Data contain only digits. Check sum is calculated and printed by the printer.
- barcodeData
  1. - EAN8 bar code. Data contains only digits and is 7 bytes long. The check sum is automatically calculated and printed.
  2. - EAN13 bar code. Data contains only digits and is 12 bytes long. The check sum is automatically calculated and printed.
  3. - Code128 bar code. Data contains symbols with ASCII codes between 32 and 127. Data length is between 16 and 32 symbols
  4. (depends on the content - the maximum length is if all symbol are digits). The check sum is automatically calculated and printed.

5. - Interleaved 2 of 5 bar code. Data contains only digits and is up to 28 bytes long. No check sum is calculated and printed.
6. - Interleaved 2 of 5 bar code. Data contains only digits and is up to 27 bytes long. The check sum is automatically calculated
7. and printed.

If data length is wrong or invalid characters are used, the "Syntax error" status is set and nothing is printed. Bar code length is set using command 43.

#### Parameters

<i>barcodeType</i>	- Bar code type. One byte with value: <ul style="list-style-type: none"> <li>• '1' EAN8. Data contain only digits and is 7 bytes long. Check sum is calculated by the printer.</li> <li>• '2' EAN13. Data contain only digits and is 12 bytes long. Check sum is calculated by the printer.</li> <li>• '3' Code 128. Data contain symbols with ASCII from 32 to 127. Length is from 9 to 18 symbols (depends on the contents - maximum length is possible if all symbols are digits). Check sum is calculated by the printer.</li> <li>• '4' ITF (Interleaved 2 of 5). Data contain only digits.</li> <li>• '5' ITF (Interleaved 2 of 5). Data contain only digits. Check sum is calculated and printed by the printer.</li> </ul>
<i>barcodeData</i>	- BC_Data

- - EAN8 bar code. Data contains only digits and is 7 bytes long. The check sum is automatically calculated and printed.
- - EAN13 bar code. Data contains only digits and is 12 bytes long. The check sum is automatically calculated and printed.
- - Code128 bar code. Data contains symbols with ASCII codes between 32 and 127. Data length is between 16 and 32 symbols
- (depends on the content - the maximum length is if all symbol are digits). The check sum is automatically calculated and printed.
- - Interleaved 2 of 5 bar code. Data contains only digits and is up to 28 bytes long. No check sum is calculated and printed.
- - Interleaved 2 of 5 bar code. Data contains only digits and is up to 27 bytes long. The check sum is automatically calculated
- and printed.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

## Returns

TRUE upon success, FALSE otherwise

1.1.2.149 - (bool) `command84Variant0Version1BarcodeType: (NSString *) barcodeType barcodeData:(NSString *) barcodeData error:(NSError **) error`

## 54h(84) PRINTING A BAR CODE

The command prints a bar code. It is permitted in an open fiscal or non-fiscal receipt. Bar code is printed centered.

- Data field:
  1. `barcodeType`
  2. `barcodeData`
- Response: None
- `barcodeType` Bar code type. One byte with value:
  1. '1' EAN8. Data contain only digits and is 7 bytes long. Check sum is calculated by the printer.
  2. '2' EAN13. Data contain only digits and is 12 bytes long. Check sum is calculated by the printer.
  3. '3' Code 128. Data contain symbols with ASCII from 32 to 127. Length is from 9 to 18 symbols (depends on the contents - maximum length is possible if all symbols are digits). Check sum is calculated by the printer.
  4. '4' ITF (Interleaved 2 of 5). Data contain only digits.
  5. '5' ITF (Interleaved 2 of 5). Data contain only digits. Check sum is calculated and printed by the printer.
- `barcodeData`
  1. - EAN8 bar code. Data contains only digits and is 7 bytes long. The check sum is automatically calculated and printed.
  2. - EAN13 bar code. Data contains only digits and is 12 bytes long. The check sum is automatically calculated and printed.
  3. - Code128 bar code. Data contains symbols with ASCII codes between 32 and 127. Data length is between 16 and 32 symbols
  4. (depends on the content - the maximum length is if all symbol are digits). The check sum is automatically calculated and printed.
  5. - Interleaved 2 of 5 bar code. Data contains only digits and is up to 28 bytes long. No check sum is calculated and printed.
  6. - Interleaved 2 of 5 bar code. Data contains only digits and is up to 27 bytes long. The check sum is automatically calculated
  7. and printed.

If data length is wrong or invalid characters are used, the "Syntax error" status is set and nothing is printed. Bar code length is set using command 43.

## Parameters

<i>barcodeType</i>	- Bar code type. One byte with value: <ul style="list-style-type: none"> <li>• '1' EAN8. Data contain only digits and is 7 bytes long. Check sum is calculated by the printer.</li> <li>• '2' EAN13. Data contain only digits and is 12 bytes long. Check sum is calculated by the printer.</li> <li>• '3' Code 128. Data contain symbols with ASCII from 32 to 127. Length is from 9 to 18 symbols (depends on the contents - maximum length is possible if all symbols are digits). Check sum is calculated by the printer.</li> <li>• '4' ITF (Interleaved 2 of 5). Data contain only digits.</li> <li>• '5' ITF (Interleaved 2 of 5). Data contain only digits. Check sum is calculated and printed by the printer.</li> </ul>
<i>barcodeData</i>	- BC_Data

- - EAN8 bar code. Data contains only digits and is 7 bytes long. The check sum is automatically calculated and printed.
- - EAN13 bar code. Data contains only digits and is 12 bytes long. The check sum is automatically calculated and printed.
- - Code128 bar code. Data contains symbols with ASCII codes between 32 and 127. Data length is between 16 and 32 symbols
- (depends on the content - the maximum length is if all symbol are digits). The check sum is automatically calculated and printed.
- - Interleaved 2 of 5 bar code. Data contains only digits and is up to 28 bytes long. No check sum is calculated and printed.
- - Interleaved 2 of 5 bar code. Data contains only digits and is up to 27 bytes long. The check sum is automatically calculated
- and printed.

## Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

## Returns

TRUE upon success, FALSE otherwise

1.1.2.150 - (NSDictionary \*) **command85Variant0Version0AdditionalPaymentTypeOption:** (NSString \*) **additionalPaymentTypeOption** **inputAdditionalPaymentName:(NSString \*) inputAdditionalPaymentName** **error:(NSError \*\*) error**

55H(85) DEFINE ADDITIONAL PAYMENT TYPES NAME

- Data field:
  1. additionalPaymentTypeOption:
    - (a) 'I' Additional payment 1
    - (b) 'J' Additional payment 2
    - (c) 'K' Additional payment 3

(d) 'L' Additional payment 4

2. inputAdditionalPaymentName: Comment text of the payment. Up to 31 bytes. If not present, the current name is returned.

- Response:

1. outputText:

(a) 'P' No error.

(b) 'F' Name longer than 31 bytes.

The command defines the comment text, printed before the additional (programmable) payments. The command is not permitted after the first fiscal receipt for the day.

#### Parameters

<i>additional-PaymentType-Option</i>	- - 'I' Additional payment 1 • 'J' Additional payment 2 • 'K' Additional payment 3 • 'L' Additional payment 4
<i>inputAdditional-PaymentName</i>	- Comment text of the payment. Up to 31 bytes. If not present, the current name is returned.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

NSDictionary

- KeyValue - @"outputText" - - 'P' No error.
- 'F' Name longer than 31 bytes.

1.1.2.151 - (NSDictionary \*) command85Variant0Version1AdditionalPaymentTypeOption: (NSString \*) additionalPaymentTypeOption error:(NSError \*\*) error

55H(85) DEFINE ADDITIONAL PAYMENT TYPES NAME

- Data field:

1. additionalPaymentTypeOption:

(a) 'I' Additional payment 1

(b) 'J' Additional payment 2

(c) 'K' Additional payment 3

(d) 'L' Additional payment 4

- Response:

1. outputAdditionalPaymentName: Comment text of the payment. Up to 31 bytes. If not present, the current name is returned.

#### Parameters

<i>additional-PaymentType-Option</i>	-
--------------------------------------	---

- 'I' Additional payment 1
- 'J' Additional payment 2
- 'K' Additional payment 3
- 'L' Additional payment 4

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

NSDictionary

- Key-Value - @"outputAdditionalPaymentName" - Comment text of the payment. Up to 31 bytes. If not present, the current name is returned.

1.1.2.152 - (NSDictionary \*) command86Variant0Version0AndReturnError: (NSError \*\*) *error*

56H(86) GET LATEST FISCAL MEMORY RECORD DATE

- Data field:
  1. None
- Response:
  1. lastFiscalMemoryDate Date of last (latest) record in the fiscal memory in format:DD-MM-YYYY

**Returns**

NSDictionary

- Key-Value - @"lastFiscalMemoryDate" - Date of last (latest) record in the fiscal memory in format:DD-MM-YYYY

1.1.2.153 - (NSDictionary \*) command87Variant0Version0AndReturnError: (NSError \*\*) *error*

57H(87) GET SHIFT LENGTH

- Data field: None
- Response: secondsAfterFirstReceipt
  - secondsAfterFirstReceipt Seconds count after the first fiscal receipt for the day.

Opening fiscal receipt will be blocked if this count is > 86400 (24 hours).

**Returns**

NSDictionary

- Key-Value - @"secondsAfterFirstReceipt" - Seconds count after the first fiscal receipt for the day.

## 1.1.2.154 - (NSDictionary \*) command90Variant0Version0AndReturnError: (NSError \*\*) error

5Ah(90) RETURNS DIAGONSTIC INFORMATION

- Data field:
  1. None
- Response:
  1. printerName Fiscal device name.
  2. firmwareRevision The version of the software program - 4 bytes.
  3. firmwareDate The date of the software program DDMmmYY - 7 bytes.
  4. firmwareTime Hour of the software program HHMM - 4 bytes.
  5. checkSum The EPROM control sum - a 4 bytes string in the hexadecimal code.
  6. softwareSwitches The configuration switches from Sw1 to Sw8 - an 8 bytes string with '0' or '1'.
  7. serialNumber The serial number - 8 bytes.
  8. fiscalMemoryNumber Number of the fiscal module - 8 bytes.

## Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

## Returns

NSDictionary

- KeyValue - @"printerName" - Fiscal device name.
- KeyValue - @"firmwareRevision" - The version of the software program - 4 bytes.
- KeyValue - @"firmwareDate" - The date of the software program DDMmmYY - 7 bytes.
- KeyValue - @"firmwareTime" - Hour of the software program HHMM - 4 bytes.
- KeyValue - @"checkSum" - The EPROM control sum - a 4 bytes string in the hexadecimal code.
- KeyValue - @"softwareSwitches" - The configuration switches from Sw1 to Sw8 - an 8 bytes string with '0' or '1'.
- KeyValue - @"serialNumber" - The serial number - 8 bytes.
- KeyValue - @"fiscalMemoryNumber" - Number of the fiscal module - 8 bytes.

## 1.1.2.155 - (NSDictionary \*) command90Variant0Version1AndReturnError: (NSError \*\*) error

5Ah(90) RETURNS DIAGONSTIC INFORMATION

- Data field:
  1. None
- Response:
  1. printerName Fiscal device name.
  2. firmwareRevision The version of the software program - 4 bytes.
  3. firmwareDate The date of the software program DDMmmYY - 7 bytes.
  4. firmwareTime Hour of the software program HHMM - 4 bytes.
  5. checkSum The EPROM control sum - a 4 bytes string in the hexadecimal code.

6. `printerIsDiscoverable` '0' or '1'. The printer is bluetooth discoverable if the value is 1.
7. `serialNumber` The serial number - 8 bytes.
8. `fiscalMemoryNumber` Number of the fiscal module - 8 bytes.

**Parameters**

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

## NSDictionary

- Key-Value - @"printerName" - Fiscal device name.
- Key-Value - @"firmwareRevision" - The version of the software program - 4 bytes.
- Key-Value - @"firmwareDate" - The date of the software program DDMmmYY - 7 bytes.
- Key-Value - @"firmwareTime" - Hour of the software program HHMM - 4 bytes.
- Key-Value - @"checksum" - The EPROM control sum - a 4 bytes string in the hexadecimal code.
- Key-Value - @"printerIsDiscoverable" - '0' or '1'. The printer is bluetooth discoverable if the value is 1
- Key-Value - @"serialNumber" - The serial number - 8 bytes.
- Key-Value - @"fiscalMemoryNumber" - Number of the fiscal module - 8 bytes.

1.1.2.156 - (bool) `command94Variant0Version0StartDate: (NSString *) startDate endDate:(NSString *) endDate error:(NSError **) error`

## 5Eh(94) DETAILED FISCAL MEMORY REPORT BY CLOSURE DATE

- Data field:
  1. `startDate`
  2. `endDate`
- Response: None
- `startDate` Starting date of selected fiscal entry - 6 bytes DDMMYY
- `endDate` End date of the fiscal entry - 6 bytes DDMMYY

This command prints out a detailed financial report of the period between two selected dates.

**Parameters**

<i>startDate</i>	- Starting date of selected fiscal entry - 6 bytes (DDMMYY)
<i>endDate</i>	- End date of the fiscal entry - 6 bytes (DDMMYY)
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

**Returns**

TRUE upon success, FALSE otherwise



1.1.2.157 - (bool) `command95Variant0Version0StartFiscalRecordNumber: (NSString *) startFiscalRecordNumber endFiscalRecordNumber:(NSString *) endFiscalRecordNumber error:(NSError **) error`

#### 5Fh(95) SHORT FISCAL MEMORY REPORT BY CLOSURE NUMBER

- Data field:
  1. `startFiscalRecordNumber`
  2. `endFiscalRecordNumber`
- Response: None
- `startFiscalRecordNumber` Starting number of fiscal entry
- `endFiscalRecordNumber` End number of fiscal entry

The command starts the calculation and the printing of a short periodic financial report.

#### Parameters

<i>startFiscal-RecordNumber</i>	- Starting number of fiscal entry
<i>endFiscal-RecordNumber</i>	- End number of fiscal entry
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

1.1.2.158 - (NSDictionary \*) `command97Variant0Version0AndReturnError: (NSError **) error`

#### 61h(97) READING THE SET TAX RATES

- Data field: None
- Response:
  1. `taxGroupA`
  2. `taxGroupB`
  3. `taxGroupC`
  4. `taxGroupD`
- `taxGroupA` Current tax rate A
- `taxGroupB` Current tax rate B
- `taxGroupC` Current tax rate C
- `taxGroupD` Current tax rate D

#### Returns

NSDictionary

- Key-Value - @"taxGroupA" - Current tax rate A
- Key-Value - @"taxGroupB" - Current tax rate B
- Key-Value - @"taxGroupC" - Current tax rate C
- Key-Value - @"taxGroupD" - Current tax rate D

### 1.1.2.159 - (NSDictionary \*) command99Variant0Version0AndReturnError: (NSError \*\*) error

63h(99) Reading the TAX REGISTRATION NUMBER

- Data field: None
- Response:
  1. vatNumber
- vatNumber The VAT number as a text (up to 14 bytes).

#### Returns

NSDictionary

- Key-Value - @"vatNumber" - The VAT number as a text (up to 14 bytes).

### 1.1.2.160 - (NSDictionary \*) diagnosticInfoGetAndReturnError: (NSError \*\*) error

diagnosticInfoGetAndReturnError - RETURNS DIAGONSTIC INFORMATION

- Data field:
  1. None
- Response:
  1. printerName Fiscal device name.
  2. firmwareRevision The version of the software program - 4 bytes.
  3. firmwareDate The date of the software program DDMmmYY - 7 bytes.
  4. firmwareTime Hour of the software program HHMM - 4 bytes.
  5. checkSum The EPROM control sum - a 4 bytes string in the hexadecimal code.
  6. softwareSwitches The configuration switches from Sw1 to Sw8 - an 8 bytes string with '0' or '1'.
  7. serialNumber The serial number - 8 bytes.
  8. fiscalMemoryNumber Number of the fiscal module - 8 bytes.

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

#### Returns

NSDictionary

- Key-Value - @"printerName" - Fiscal device name.
- Key-Value - @"firmwareRevision" - The version of the software program - 4 bytes.
- Key-Value - @"firmwareDate" - The date of the software program DDMmmYY - 7 bytes.
- Key-Value - @"firmwareTime" - Hour of the software program HHMM - 4 bytes.
- Key-Value - @"checkSum" - The EPROM control sum - a 4 bytes string in the hexadecimal code.
- Key-Value - @"softwareSwitches" - The configuration switches from Sw1 to Sw8 - an 8 bytes string with '0' or '1'.
- Key-Value - @"serialNumber" - The serial number - 8 bytes.

- KeyValue - @"fiscalMemoryNumber" - Number of the fiscal module - 8 bytes.

#### 1.1.2.161 - (bool) diagnosticInfoPrintAndReturnError: (NSError \*\*) error

diagnosticInfoPrintAndReturnError - PRINTING DIAGNOSTIC INFORMATION

- Data field: No data
- Response: None

The command initiates the generation of an interior receipt containing diagnostic information as follows:

- - Prints the date and version of the employed software,
- - Prints the control sum of the employed firmware,
- - Prints the serial port's band rate,
- - Prints out the status of memory switches,
- - Prints emergency time after power supply cut-off,
- - Prints the number, date and hour of the last reset of the RAM (if there is such),
- - Prints the current temperature of the two printer heads,
- - Prints the overall number of fields in the fiscal memory and the number of the free fields,
- - Prints the current date and hour.

The command will not be executed when there is an open receipt in progress or when the paper roll has finished. It may also be activated by pressing the "FEED" button while power on for less than 2 seconds.

#### Returns

TRUE upon success, FALSE otherwise

#### 1.1.2.162 - (NSDictionary \*) getStatusBytesAndReturnError: (NSError \*\*) error

getStatusBytesAndReturnError - READING THE STATUS BYTES

- Data field: None
- Response:
  1. statusByte0 Status byte 0
  2. statusByte1 Status byte 1
  3. statusByte2 Status byte 2
  4. statusByte3 Status byte 3
  5. statusByte4 Status byte 4
  6. statusByte5 Status byte 5

The status is returned immediately (default).

#### Parameters

<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information
--------------	--

**Returns**

NSDictionary

- KeyValue - @"statusByte0" - statusByte0
- KeyValue - @"statusByte1" - statusByte1
- KeyValue - @"statusByte2" - statusByte2
- KeyValue - @"statusByte3" - statusByte3
- KeyValue - @"statusByte4" - statusByte4
- KeyValue - @"statusByte5" - statusByte5

#### 1.1.2.163 - (NSDictionary \*) nonFiscalReceiptCloseAndReturnError: (NSError \*\*) error

nonFiscalReceiptCloseAndReturnError - Closing a non-fiscal receipt

- Data field:
  1. None
- Response:
  1. Allreceipt The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

The FP performs the following actions:

- Prints the footer
- The sequence number, date and hour of document are printed
- "NON-FISCAL RECEIPT" is printed in expanded style.

If the S1.1 flag is raised, the command is not executed because there is no opened non-fiscal receipt.

**Returns**

NSDictionary

- KeyValue - @"allReceipt" - The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

#### 1.1.2.164 - (NSDictionary \*) nonFiscalReceiptOpenAndReturnError: (NSError \*\*) error

nonFiscalReceiptOpenAndReturnError - Opening a non-fiscal receipt.

- Data field:
  1. None
- Response:
  1. Allreceipt The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

The FP performs the following actions:

- Prints the header
- Prints the tax registration number of the seller
- A response is received, which contains Allreceipt

The command cannot be executed, S1.1 is raised if.

- The fiscal memory has not been formatted
- There is an opened fiscal or non-fiscal receipt
- There is no paper
- The clock is not set
- The electronic journal is full

#### Returns

NSDictionary

- Key-Value - @"allReceipt" - The number of all issued receipts (fiscal and non-fiscal) from the last daily closure on (4 bytes).

**1.1.2.165** - (bool) nonFiscalReceiptPrintText: (NSString \*) *text* error:(NSError \*\*) *error*

nonFiscalReceiptPrintText - PRINTING OF A FREE NON-FISCAL TEXT

- Data field:
  1. inputText A text of 30 symbols (at most). The symbols after 30 are cut off.
- Response:
  1. None

If S1.1 is raised, there is no non-fiscal receipt opened and the text is not printed.

#### Parameters

<i>inputText</i>	- A text of 30 symbols (at most). The symbols after 30 are cut off.
<i>error</i>	pointer to NSError object, where error information is stored in case function fails. You can pass nil if you don't want that information

#### Returns

TRUE upon success, FALSE otherwise

**1.1.2.166** - (void) removeDelegate: (id) *newDelegate*

Removes delegate, previously added with addDelegate.

#### Parameters

<i>newDelegate</i>	the delegate that will be no longer be notified of Linea events
--------------------	---