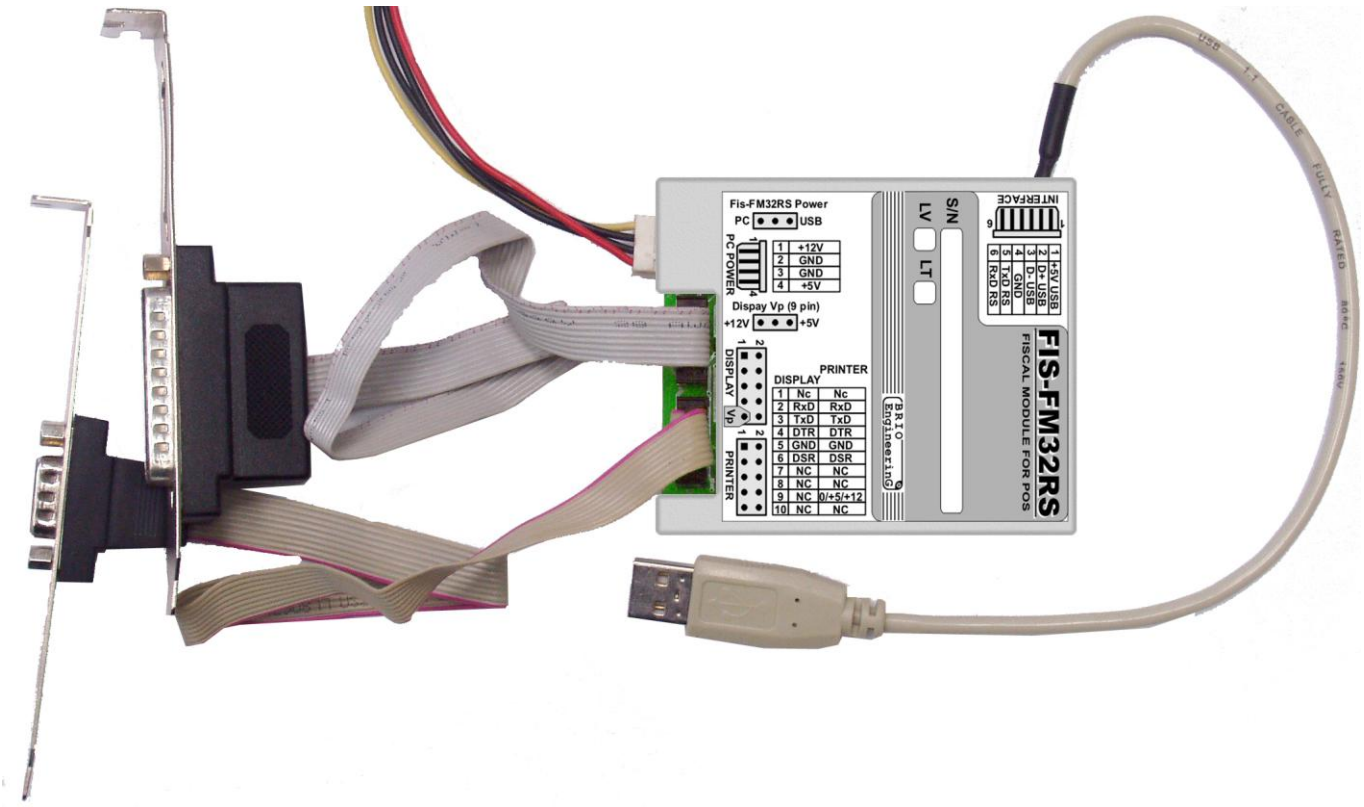
BRIO Fis-FM32

FISCAL MODULE
MODULE PROGRAMMING



BRIO EngineerinG®

O.Halatov

Fiscal module Fis-FM32™ V1.0 for POS systems
Module programming. Rev. 1
BRIO EngineerinG, 2007,
Riga, Latvia.

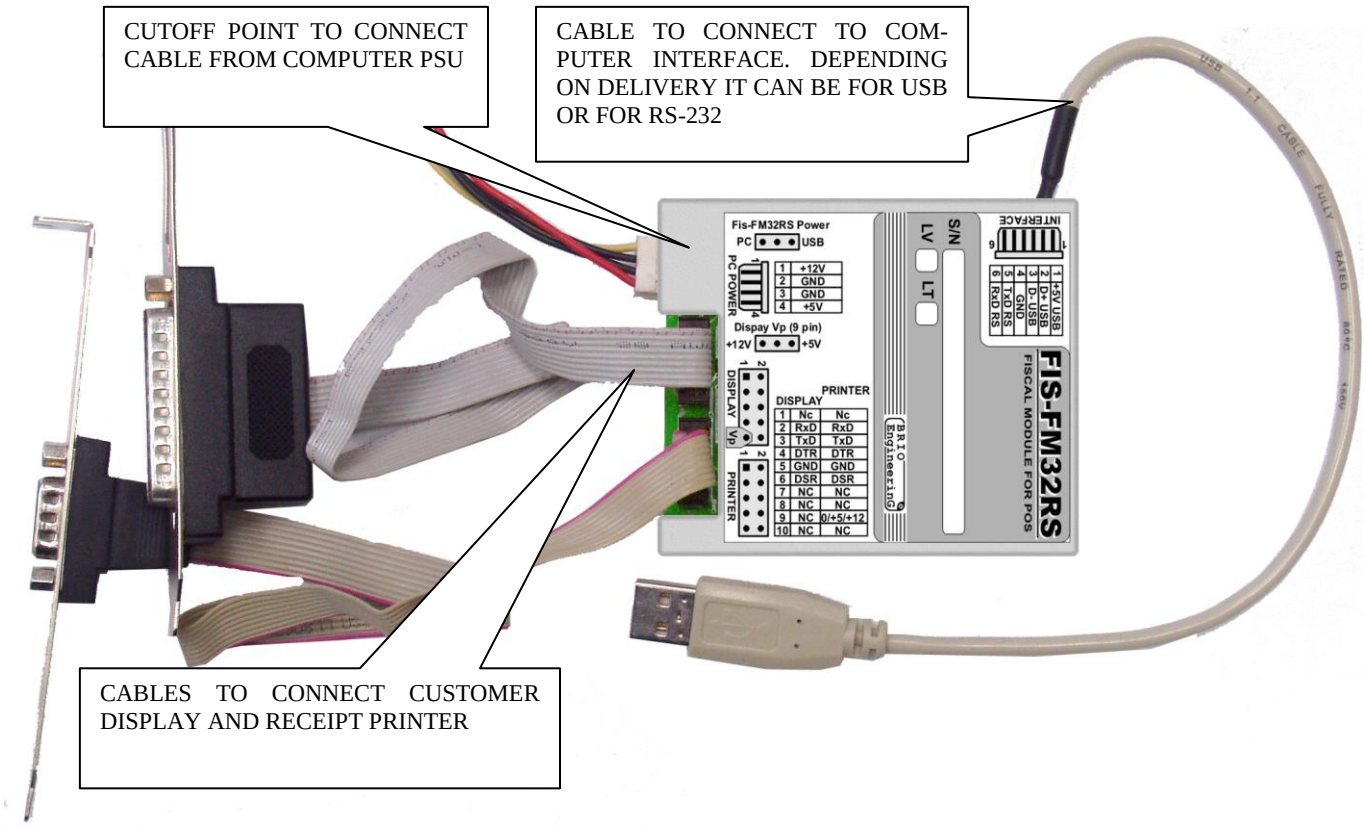
This document contains description, order of working and performance attributes of fiscal module Fis-FM32™. Module is intended for working together with sale POS systems, which are based on PC compatible computers, and completely satisfies the requirements of legislation of the Republic of Latvia.

CONTENTS

1. PERFORMANCE ATRIBUTES

Interface with PC, with automatical choice of interface type.	- USB V1.1, - USB V2.0, - RS-232C (115200 Bits/sec. 8 Bits, Non Parity, 1-Stop, Non flow control)
Type of build in processor	- ARM. Program is closed with “secret byte”
Peak current of consumption	- 100 mA (Max)
Capacity of fiscal memory ROM	- 8 MByte
Capacity of memory for electronic alignment tape	- 4 MByte
Order of exchange PC ↔ Fis-FM32	- Batch exchange by special protocol.
Fiscalization	- Single.
Ceiling amount of Z-reports.	- 8192
Initialization	- Single.
Hardware diagnostics	- Build in system of mistake diagnostics
Защита от стирания Protection from deletion	- Special scheme for blocking deletion command.
Additional interfaces	- RS-232C to connect receipt printer - RS-232C to connect customer display , with an additional power lead (5V or 12V) for 9th lead of cutoff point.
Types of supported printers	Epson-210 Epson-260 Epson-950 CHD- TH582 Samsung- SRP350 Samsung- SRP275 Citizen
Types of supported customer displays	Any, which use interface RS-232C

2. CABLES AND CUTOFF POINTS



RS-232C/USB CONNECTION WITH COMPUTER

Cont.	Signal	Direction	Function
1	DC +5V USB		Supply voltage +5V
2	D + USB		Bus-bar USB
3	D - USB		Bus-bar USB
4	GND.L		Common cable
5	TxD	Exit	Data transfer
6	RxD	Entry	Data receiving

ATTENTION!!! Fiscal module can get power supply from computer PSU as well as from supply voltage from USB cable. Choise is made with a bridge Fis-FM32 POWER PC/USB. But in a case of power supply from USB bus-bar supply voltage to the 9th interface outlet CUSTOMER DISPLAY doesn`t go.

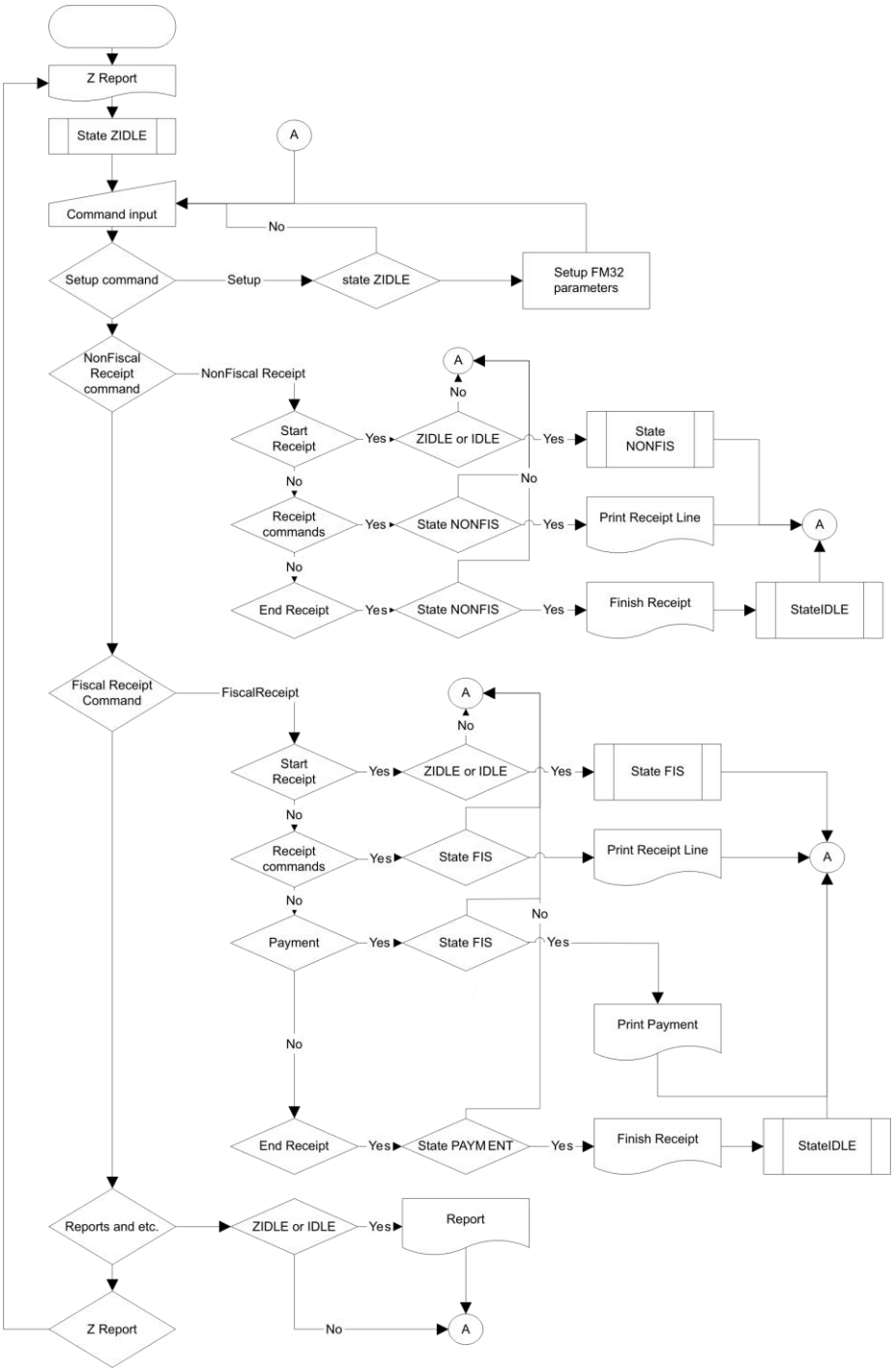
RS-232C PRINTER AND CUSTOMER DISPLAY

				CONTACTS ON CABLES CUTOFF POINTS	
CONT.	DISPLAY	PRINTER	SIGNAL FUNCTION	DB 25 (M)	DB 9 (M)
1	Nc	Nc	Not used		
2	RxD	RxD	Data receiving	3	2
3	TxD	TxD	Data transfer	2	3
4	DTR	DTR		20	4
5	GND	GND	Common	7	5
6	DSR	DSR		6	6
7	Nc	Nc	Not used		
8	Nc	Nc	Not used		
9	Nc	0/+5V/+12V	Not used/ Display power supply	22	9
10	Nc	Nc	Not used		

3. CONDITIONS OF FISCAL MODULE.

Fiscal module can be in five different conditions. Commands can be in progress only in those conditions for which they are allowed. After some command execution fiscal module can change its condition.

IDLE – Fiscal module turns into this condition after printing next receipt.
ZIDLE - Fiscal module turns into this condition after Z-report execution.
FIS – Condition of fiscal module when registration of new fiscal receipt has already started, but its payment not yet registrate.
NONFIS – Condition when fiscal receipt is already started, but not yet finished.
HARD – Fiscal module turns into this conditions in a case of error working with display or receipt printer.



ALGORITHM OF COMMANDS PASSING

4. FUNCTION LIBRARY FOR WORK WITH BRIO- FM32.

RESET OF FISCAL MODULE CONDITION
int ResetFiscal(void)

RECEIPT PRINTER TYPE SETTING
int SetPrinterType(int printerType)
Used printer types:
- printerEpson210 = 0
- printerCitizeThermal = 1
- printerEpson260 = 2
- printerEpson950 = 3
- printerCHDTH582 = 4,
- printerSRP350 = 10,
- printerSRP275 = 11

RECEIPT PRINTER TYPE SETTING WITH AN INDICATION OF SYMBOL CODING KIND
int SetPrinterTypeEx(int printerType, int coding)
Kinds of symbol coding:
- codingDOS = 0
- codingWin = 1

SETTING INTERVAL BETWEEN LINES ON RECEIPT PRINTER
int SetCompressionMode(int compression)

SETTING DURATION OF IMPULSES TO OPEN CASH DRAWER
int SetTillImpuls(int time1, int time2)

SETTING DATE OF FISCAL MODULE
int SetDate(char * dateString)
Date in format "YYYY.MM.DD"

SETTING TIME OF FISCAL MODULE
int SetTime(char * timeString)
Time in format "HH:MM"

SETTING RECEIPT HEADING
int SetHeader(char * line1, char * line2, char * line3, char * line4)

SETTING RECEIPT HEADING WITH AN INDICATION OF LINE FORMATTING OPERATION FACTORS
int SetHeaderEx(int attr1, char * line1, int attr2, char * line2, int attr3, char * line3, int attr4, char * line4)

SETTING RECEIPT FINISHING LINES WITH AN INDICATION OF LINE FORMATTING OPERATION FACTORS
int SetFooter(int attr1, char * line1, int attr2, char * line2, int attr3, char * line3, int attr4, char * line4)

PERMISSION/PROHIBITION TO PRINT RECEIPT FINISHING LINES
int EnableFooter(int enable)

SETTING ADDITIONAL RECEIPT FINISHING LINES WITH FORMATTING
int SetFooter2(int attr1, char * line1, int attr2, char * line2)

PERMISSION/PROHIBITION TO PRINT ADDITIONAL RECEIPT FINISHING LINES
int EnableFooter2(int enable)

SETTING TAX RATE
int SetVAT(int number, double rate)
int SetVat(int number, double rate)

SETTING CURRENCY OPERATION FACTORS
int SetCurrency(int number, char * name, double rate)

SETTING TITLE OF CASHLESS TYPE OF PAYMENT
int SetCredit(int number, char * name)

PERMISSION/PROHIBITION OF RETURN OPERATION
int AllowGoodsReturn(char * unitNumber)

GETTING INFORMATION FROM FISCAL MODULE
int GetFiscalInfo(int infoType, char * data)

- infoReceiptSumm = 0 – total sum of started receipt
- infoShiftTurnover = 1 – total turnover for shift
- infoReceiptNumber = 2 – current receipt number
- infoUnitNumber = 3 – number of fiscal block
- intUnitVersion = 4 – version of fiscal block
- infoDate = 5 – date in fiscal block
- infoTime = 6 – time in fiscal block
- infoReportNumber = 7 – current number of Z-report
- infoDayReceiptNumber = 8 – number of fiscal receipt for shift
- infoUnitState = 9 – condition of fiscal module
- infoHeader = 10 – lines of receipt heading
- infoFooter = 11 – finishing lines of receipt
- infoFooter2 = 12 – additional finishing lines of receipt
- infoFooterEnabled = 13 – permission to print receipt finishing lines
- infoFooter2Enabled = 14 – permission to print additional receipt finishing lines
- infoCurDescription = 30 – elements of currency table
- infoTax = 40 – elements of tax table
- infoCurCash = 50 – elements of table of money remains in cash drawer
- infoCreditDescription = 60 – elements of table of titles of cashless type of payment
- infoTaxTurnover = 70 – elements of table of turnovers by tax kind
- infoTaxSumm = 80 – elements of table of tax sums by tax kinds

ENTERING OF EXCHANGE MONEY
int MoneyIn(double money)

ENTERING OF EXCHANGE MONEY IN CURRENCY
int MoneyInCurr(int number, double money);
// Extract money from cash drawer

COLLECTION
int MoneyOut(double money)

COLLECTION IN CURRENCY
int MoneyOutCurr(int number, double money)
int MoneyOutCurr2(int number, double money)

OPEN CASH DRAWER
int OpenCashDrawer(void)
int OpenCachDrower(void)

PRINT TAX TABLE ON RECEIPT PRINTER
int PrintVATTable(void)
int PrintVatTable(void)

PRINT CURRENCY TABLE ON RECEIPT PRINTER
int PrintCurrencyTable(void)

PRINT TABLE OF TITLE OF CASHLESS TYPE OF PAYMENT ON RECEIPT PRINTER
int PrintCreditTable(void)

PRINT Z-REPORT
int PrintZReport(void)

PRINT X-REPORT
int PrintXReport(void)

PRINT MINI X-REPORT
int PrintMiniXReport(void)

PRINT SUMMARY PERIODICAL REPORT BY DATES RANGE
int PrintSumPeriodicReport(char * date1, char * date2)
Dates in format "YYYY.MM.DD".

PRINT SUMMARY PERIODICAL REPORT BY NUMBERS RANGE
int PrintSumPeriodicReportByNumber(int number1, int number2)

PRINT PERIODICAL REPORT BY DATES RANGE
int PrintPeriodicReport(char * date1, char * date2)
Dates in format "YYYY.MM.DD".

PRINT PERIODICAL REPORT BY NUMBERS RANGE
int PrintPeriodicReportByNumber(int number1, int number2)

SHOW INFORMATION ON CUSTOMER DISPLAY
int CustomerDisplay(int displayType, char * line1, char * line2)
int CustomerDisplay2(char * line1, char * line2)
int CustomerDisplayPro(char * command)
displayType – type of Display : 1

BEGIN NON-FISCAL RECEIPT
int BeginNonFiscalReceipt(void)

PRINT LINE WITH PACKAGE
int PrintTareItem(char * name, double quantity, double price)

REFUSAL OF PACKING ITEM
int PrintTareItemVoid(char * name, double quantity, double price)

PRINT LINE WITH DEPOSIT
int PrintDepositReceive(char * name, double quantity, double price)

REFUSAL OF DEPOSIT POSITION
int PrintDepositRefund(char * name, double quantity, double price)

PRINTING INFORMATIONAL LINE IN NON-FISCAL RECEIPT
int PrintNonFiscalLine(char * line, int attribute)

FINISH NON-FISCAL RECEIPT
int EndNonFiscalReceipt(void)

BEGIN FISCAL RECEIPT
int BeginFiscalReceipt(void)

PRINTING LINES OF FISCAL RECEIPT
int PrintRecItem(char * name, double quantity, double price, int taxNumber, char * unit)

REFUSAL OF FISCAL RECEIPT LINE
int ItemReturn(char * name, double quantity, double price,
int taxNumber, char * unit, int depart,
double discountPercent, double discountSumm)

int ItemReturnEx(char * name, double quantity, double price,
int taxNumber, char * unit, int depart,
int discountType, double discount)

PRINTING COMMENT LINE IN FISCAL RECEIPT
int PrintCommentLine(char * line, int attribute)

DISCOUNT ON RECEIPT ITEM
int DiscountAdditionForItem(int type, double val)
Тип скидки:
- dtPcnt = 1 – in percents
- dtSumm = 2 – absolute value

DISCOUNT ON RECEIPT
int DiscountAdditionForReceipt(int type, double val)
Тип скидки:
- dtPcnt = 1 – in percents
- dtSumm = 2 – absolute value

FINISH REGISTRATION OF FISCAL RECEIPT
int EndFiscalReceipt(double summCash, double summCredit1, double summCredit2,
double summCredi3, double summCredi4)

```
int EndFiscalReceiptCurr( double summCash, double summCredit1,
                          double summCredit2, double summCredit3,
                          double summCredit4, double summCur1, double summCur2,
                          double summCur3 )
int EndFiscalReceiptCurrEx( double summCash, double summCredit1,
                            double summCredit2, double summCredit3,
                            double summCredit4, double summCredit5,
                            double summCredit6, double summCredit7,
                            double summCredit8,
                            double summCur1, double summCur2,
                            double summCur3 )
```

FINISH REGISTRATION OF RETURN RECEIPT

```
int GoodsReturn( double summCash, double summCredit1,
                 double summCredit2, double summCredit3,
                 double summCredit4 )
int GoodsReturnCurr( double summCash, double summCredit1,
                     double summCredit2, double summCredit3,
                     double summCredit4,
                     double summCur1, double summCur2,
                     double SummCur3 )
int GoodsReturnCurrEx( double summCash, double summCredit1,
                       double summCredit2, double summCredit3,
                       double summCredit4, double summCredit5,
                       double summCredit6, double summCredit7,
                       double summCredit8,
                       double summCur1, double summCur2,
                       double SummCur3 )
```

PRINTING OF RECEIPT COPY

```
int PrintCopyOfLastReceipt( void )
```

PRINTING MESSAGE ON RECEIP PRINTER

```
int PrintErrorMessage( char * message )
```

MODULE FISCALIZATION

```
int Fiscalization( char * date )
```

PROGRAMMING OF REGISTRATION NUMBER (ONLY FOR LITHUANIA)

```
int SetCompanyVATCode( char regNum )
```

PROGRAMMING OF DEPARTMENT TITLE

```
int SetDepartName( int number, char * name )
```

GETTING DATE FROM FISCAL MODULE

```
int GetFiscalData( int infoType, char data[10] )
```

RECORDING OF IDENTIFICATION NUMBER OF SYSTEM

```
int SetId( char * id )
```

READING OF IDENTIFICATION NUMBER OF SYSTEM

```
int GetIdNumber( char * date )
```

READING CHASSIS NUMBER

```
int GetUnitNumber( char * date )
```

GOODS CORRECTION IN RECEIPT

```
int ItemReturnDepart( char * name, double quantity, double price,
                      int taxNumber, char * unit, int depart,
                      double discountPercent, double discountSumm )
```

GOODS SALE IN RECEIPT

```
int PrintRecItemDepart( char * name, double quantity, double price,
                        int taxNumber, char * unit, int depart )
```

RECORDING CHASSIS NUMBER

```
int SetShassi( char * number )
```

5. CODES OF MISTAKES

Fis-FM32 ERRORS	MEANING		COMMENTS
	DEC	HEX	
FM32_OK	0	0x00	No mistakes.
ERR_LENGTH	4	0x04	Inadmissible length of data package.
ERR_DATA	5	0x05	Inadmissible data in package.
ERR_XOR	6	0x06	Wrong control sum of data package.
ERR_ETX	7	0x07	Miss symbol of the end of the package.
ERR_ILLEGAL	16	0x10	Wrong or not supported command.
ERR_IDLE_STATE	17	0x11	Command can not be execute because FM32 is in condition IDLE.
ERR_NONFIS_STATE	18	0x12	Command can not be execute because FM32 is in condition NONFIS.
ERR_FIS_STATE	19	0x13	Command can not be execute because FM32 is in condition FIS.
ERR_HARD_STATE	20	0x14	Error working with external unit.For example, with printer or with custom-er display.
ERR_PARAMETERS	21	0x15	Wrong parameters in package. Inadmissible parameters for command or parameters contain inadmissible value.
ERR_ITEM_DESC_LENGTH	22	0x16	Length of parameter in command exceeds permissible.
ERR_ITEM_OUANTITY	23	0x17	Inadmissible quantity.
ERR_ITEM_PRICE	24	0x18	Inadmissible price.
ERR_VAT	25	0x19	Inadmissible catalogue number. Numbers can be in range from 0 till 4.
ERR_ITEM_DIM	26	0x1A	Inadmissible title of measuring unit. Title can not consist more the of 4 symbols.
ERR_DEFICIENT_PAYMENT	27	0x1B	Money from customer is less than sum of purchase.
ERR_OVERPAYMENT_CREDIT	28	0x1C	Sum of cashless payment exceeds total receipt sum.
ERR_ITEM_DISCOUNT	29	0x1D	Inadmissible discount/extra charge on goods. Possible causes : percent of discount excceds 100% etc.
ERR_DISCOUNT_TYPE	30	0x1E	Inadmissible type of discount/extra charge.
ERR_COMMENT_LENGTH	31	0x1F	Inadmissible length of comment line.
ERR_PRINTER	32	0x20	Printer error.
ERR_DISPLAY	33	0x21	Customer display error.
ERR_PRICE_AMOUNT_OVERFL OW	34	0x22	Price or goods quantity is more than 99999.99
ERR_FLASH_WRITE	35	0x23	Error writing in fiscal memory FM32.
ERR_NOT_FISCAL	36	0x24	FM32 is not fiscalized. Some reports are forbidden.
ERR_BAD_DATE	37	0x25	Wrong data format.
ERR_REPORT_NOT_FOUND	38	0x26	No reports in indicated interval of dates.
ERR_FLASH_ERASE	39	0x27	Fiscal memory FM32 is not accessible for deletion or writing.
ERR_DISCOUNT_RECEIPT	40	0x28	Inadmissible discount/extra charge on goods.
ERR_NO_ITEMS	41	0x29	Discount or extra charge is not attached to the concrete goods.
ERR_CANT_RETURN	42	0x2A	Attempt to correct sale or not yet sold goods.
ERR_OVER_ADDITION_PERCE NT	43	0x2B	Percent of extra charge is too big.
ERR_OVER_DISCOUNT_PERCE NT	44	0x2C	Percent of discount is too big.
ERR_OVER_ADDITION_FIXED	45	0x2D	Absolute extra charge is too big.
ERR_OVER_DISCOUNT_FIXED	46	0x2E	Absolute discount is too big.
ERR_NO_MONEY_FOR_DA	47	0x2F	Not enough cash in cash drawer.
ERR_ZERO_TOTAL	48	0x30	Receipt sum = 0
ERR_PAYMENT_NOT_EQUAL	49	0x31	Return sum exceeds total sum by receipts.
ERR_DEFICIENT_CASH_DRAWER	50	0x32	Not enough cash in cash drawer.
ERR_UNIQUE_FM32_NUMBER	51	0x33	Wrong serial number FM32.
ERR_NOTALLOW_GOODS_RETURN	52	0x34	Goods return is forbidden.
ERR_ALREADYFISCAL	54	0x36	FM32 is already fiscalized.
ERR_NOT_SL_PRN	55	0x37	
ERR_RECEIPT_AMOUNT_OVER FLOW	56	0x38	Sum by receipt exceeds 9900000.
ERR_FLASH_FULL	57	0x39	Fiscal memory FM32 is completely filled up.
ERR_YEAR_VALUE	58	0x3A	Inadmissible year value in date.
ERR_MONTH_VALUE	59	0x3B	Inadmissible month value in date.
ERR_DAY_VALUE	60	0x3C	Inadmissible day value in date.
ERR_PAY_STATE	64	0x40	
ERR_CURRENCY_NUMBER	68	0x44	Inadmissible currency number.
ERR_CURRENCY_RATE	69	0x45	Inadmissible currency rate.
ERR_CURRENCY_NOT_SET	70	0x46	Currency is not set.
ERR_CREDIT_OVERFLOW	72	0x48	Sum of cashless payment exceeds sum by receipts.
ERR_TARE_QUANTITY	73	0x49	Inadmissible quantity of goods package.

ERR_CREDIT_ID	76	0x4C	Inadmissible ID of cashless payment.
ERR_CREDIT_DESC	77	0x4D	Inadmissible title of cashless payment.
ERR_REFUND_IN_CURRENCY	78	0x4E	Payment for returning goods is possible only in basic currency.
ERR_CREDIT_NOT_SET	79	0x4F	Cashless payment is not determined.
ERR_VAT_AMOUNT	80	0x50	Sales with given tax is lacking.
ERR_OVER_BUF	100	0x64	Repletion of buffer of current receipt goods.
ERR_DEPART_ID	128	0x64	Non-existent depart number.
ERR_DEPART_DESC	129	0x64	Wrong title of department.
ERR_DEPART_SET	130	0x64	Error pre-setting depart parameters.

FiscalUnit.DLL ERRORS	MEANING	COMMENTS
ErrorDLLIllegalPacketStructure	-1	Wrong structure of package marge.
ErrorDLLExecuteCmd	-2	Inner mistake executing command.
ErrorUnknownCommand	16	Non-existent command.
ErrorDidNotSendPacket	160	Not possible to send package.
ErrorDidNotReceivePacket	161	Not possible to receive package.
ErrorBadJournal	162	Wrong control sum of electronic tape.

6. PROGRAMMING OF FISCAL MODULE

6.1. ACCESSIBILITY OF FUNCTIONS

For correct work of fiscal and for prevention from wrong operations, in different fiscal conditions different functions of FiscalUnit.dll. are available.

Possible fiscal conditions are described in paragraph **«Conditions of fiscal module».** By call of function in inappropriate condition is possible appearance of mistakes with numbers 17,18,19 (look «Codes of mistakes»).

To return to IDLE condition or in case of mistake appearance you must call function ResetFiscal().

6.2. ORDER OF USE OF FUNCTION LIBRARY

All functions, except ones connected with receipt (both fiscal and non-fiscal), can be used in the IDLE condition. For functions, which are connected with receipt a typical order of function call can be described as following:

- Call of function «Beginning of receipt». Fiscal condition changes.
- Call of function «Body of receipt»
- Call of function «Finishing of receipt». Fiscal condition returns to IDLE

6.3. EXAMPLE OF HOW TO USE LIBRARY OF FUNCTIONS

Example is written in Delphi.

Load of library:

```
procedure TTraceForm.FormCreate(Sender: TObject);
begin
{You can use FiscalUnit.dll for local connect to FM32
or NetFiscalUnit.dll for using FM32 remote
}

//  if lUse_Fm32Net then
//    FisHandle := LoadLibrary('NetFiscalUnit.dll')
//  else
    FisHandle := LoadLibrary('FiscalUnit.dll');

  if FisHandle>0 then
    nFatalError:=0
  else
    nFatalError:=999;
end;
```

Printing fiscal receipt:

```
      //lReturn – Flag, which means sale or return
      Function SendReceipt(lReturn:boolean):integer;
      var
         sName, sUnit:string;
         nQuant, nPrice, nCash,nCred1,nCred2,nCred3,nCred4:double;
         nVat, nTotDisc, nDisc:integer;
         PrintRecItem_proc: function(sN:string;nQuant,nPrice:double;
              nVat:integer;sUnit:string):integer;stdcall;
         BeginFiscalReceipt_proc: function():integer;stdcall;
         PrintCommentLine_proc: function(_Line : string; _attrib : integer):integer;stdcall;
         DiscountAdditionForItem_proc: function(_type : integer; _amount : double):integer;stdcall;
         DiscountAdditionForReceipt_proc: function(ntype:integer;amount:double):integer;stdcall;
         EndReceipt_proc: function(_Pay,_K1,_K2,_K3,_K4:double):integer;stdcall;
      begin
        //Set variables for receipt
        sName:='Item1';
        sUnit:='Kg.';
        nQuant:=3;
        nPrice:=1.18;
        nDisc:=10;
        nTotDisc:=5;
        nVat:=0;//0..3
        nCash:=3.03;
        nCred1:=0;
        nCred2:=0;
```

```
  nCred3:=0;
  nCred4:=0;
  //Get address of receipt beginning procedure
  @BeginFiscalReceipt_proc := GetProcAddress(FisHandle, 'BeginFiscalReceipt');
  result:=BeginFiscalReceipt_proc();//и вызываем
  if Result<>0 then
    exit;
  //comment
  @PrintCommentLine_proc := GetProcAddress(FisHandle, 'PrintCommentLine');
  result:=PrintCommentLine_proc('This is fiscal receipt!',65);//65-маска форматирования текста
  if Result<>0 then
    exit;
  //Goods in receipt
  @PrintRecItem_proc := GetProcAddress(FisHandle, 'PrintRecItem');
  result:=PrintRecItem_proc(sName,abs(nQuant),abs(nPrice),nVat,sUnit);
  if Result<>0 then
    exit;
  {PrintRecItem together with  DiscountAdditionForItem
  can repeat one after another for each goods in receipt.
  DiscountAdditionForItem is not obligatory to use
  }
  //Discount on position
  @DiscountAdditionForItem_proc := GetProcAddress(FisHandle, 'DiscountAdditionForItem');
  //First parameter – type of discount. 1 - percents
  //Negative number – discount. Positive number – extra charge
  result:=DiscountAdditionForItem_proc(1,-1*nDisc);
  if Result<>0 then
    exit;
  //Discount on receipt. Not obligatory to use.
  @DiscountAdditionForReceipt_proc :=
    GetProcAddress(FisHandle, 'DiscountAdditionForReceipt');
  // First parameter – type of discount. 1 - percents
  result:=DiscountAdditionForReceipt_proc(1,-1*nTotDisc);
  if Result<>0 then
    exit;
  //Depending on parameter (flag) we load receipt with a command
  //sale or return
  if lReturn then
  //During returning sum of money must coincide with receipt sum and ALL positions
  //of goods must have negative quantity
    @EndReceipt_proc := GetProcAddress(FisHandle, 'GoodsReturn')
  else
    @EndReceipt_proc := GetProcAddress(FisHandle, 'EndFiscalReceipt');
  result:=EndReceipt_proc(nCash,nCred1,nCred2,nCred3,nCred4);
end;
```

Printing non-fiscal receipt:

```
Function SendNonFisReceipt():integer;
var
  NonFiscalReceipt_proc: function():integer;stdcall;
  PrintNonFiscalLine_proc: function(_Line : string; _attrib : integer):integer;stdcall;
begin
  // Get address of non-fiscal receipt beginning procedure
  @NonFiscalReceipt_proc := GetProcAddress(FisHandle, 'BeginNonFiscalReceipt');
  result:=NonFiscalReceipt_proc();
  if Result<>0 then
    exit;
  //Print 2 lines
  @PrintNonFiscalLine_proc := GetProcAddress(FisHandle, 'PrintNonFiscalLine');
  result:=PrintNonFiscalLine_proc('Line1',65);//65-маска форматирования текста
  if Result<>0 then
    exit;
  result:=PrintNonFiscalLine_proc('Line2',65);//65-маска форматирования текста
  if Result<>0 then
    exit;

  //Finish non-fiscal receipt
```

```
    @NonFiscalReceipt_proc := GetProcAddress(FisHandle, 'EndNonFiscalReceipt');
    Result:=NonFiscalReceipt_proc();
    if Result<>0 then
      exit;
  end;
```

Unloading of library: FreeLibrary(FisHandle);

Unloading of library is obligatory because it is necessary for correct work of fiscal.

7. EC DECLARATION OF CONFORMITY

EC
Atbilstības deklarācija

Mes, SIA "BRIO ZRF" ar pilnu atbildību, deklarējam, kas tālāk nosauktais produkts atbilst prasībām, kas noteiktas

Zemsprieguma direktīvā
73/23/EEC ar papildinājumiem 93/68/EEC

Produkta kategorija: Fiskālais modulis

Modeļa nosaukums: BRIO Fis-FM32

Piemērotie standarti: EN60950, EN55022, EN55010

CE zīmes piestiprināšanas gads: 2006 g.

Rīga , 12.10.2006

Oļegs Halatovs , Direktors