ActiveX driver for fiscal printer EltradeFPAxG2_ZW.dll

Version 2 (Zimbabwe)

Driver for management of ELTRADE fiscal devices. The driver is active X com object and serves for printing of fiscal and non-fiscal receipts on fiscal device.

Definitions

ActiveX is a software framework created by Microsoft that adapts Component Object Model (COM) and Object Linking and Embedding (OLE) technologies.

ActiveX is supported as of Windows 10 and Internet Explorer 11.

ActiveX is not supported in default web browser Microsoft Edge.

An ActiveX driver expose its functionality, specifically properties and methods, via the IDispatch interface to another program on the system. The IDispatch interface is a standard COM interface.

The user can use these methods to perform specific tasks with the fiscal device.

Installation

Use the automated installer to install driver. You can install the component manually – read "Register an ActiveX control"

Identification

Globally-Unique Identifier (GUID) {181E0F5D-D0C0-40E9-8620-2C1191CC00BC}
Library name: EltradeFPAx

Uninstall

Run *unistall.exe* from the working folder of the program. Or use a manual uninstall – read "Register an ActiveX control"

Example:

The example shows the use of ActiveX when selling an item.
Here we use Java Script, but the rules apply to all programming languages in which the driver is used .

```
try {
 try {
  var efp = new ActiveXObject("EltradeFPAx.EltradeFprn");

  efp.GenerateLogFile=1;   //enable logging
  efp.LogFilePath='D://';  //sets log folder

  if(efp.Init(1 115200, 0, 0, 0, "")!=0)
   throw  efp.GetLastError(1); // check for error

  efp.CurrencyID = 2; //set receipt currency

  if (efp.StartBon( 2,                  //rec.type - fiscal
                    1,                  //plu numbering
                    "Cashier 1",        //oper.name
                    "9000000001",       //Invoice no
                    "123456789",        //InvTaxNumber
                    "BG123456789",      //InvVATNumber
                    "Company Name",     //InvCompany
                    "City",             //InvCity
                    "Address",          //InvAddress
                    "MOL",              //InvMOL
                    "Receiver")!=0)
       throw  efp.GetLastError(1);      // failed start receipt

  efp.PLUCurrencyID = 2; //set PLU item currency

  if(efp.AddPlu("1", "PLU number 1", 1.23, 1, 1))!=0) //Add PLU item
   throw  efp.GetLastError(1); // failed add plu item

  if(efp.AddDiscount(1, 10.00))!=0) //add discount
   throw  efp.GetLastError(1); // failed add discount

  if(efp.AddPayment(1, "CASH", 100, 1)!=0) // add payment
   throw  efp.GetLastError(1); // failed add payment

  if(efp.EndBon()!=0)//print receipt
   throw  efp.GetLastError(1);

 } catch (error) {
   alert(error);
 }
finally {
  if( efp && efp !== 'null' && efp !== 'undefined') efp.Close();
}
```

For an advanced example of working with ActiveX, you can view the source of a file **TestFPDriver_ZW.html** located in the driver installation folder .

Description of the entry points:

Function	Result	Entry parameters	Description
Init	**Type: long** Return error code 0 – successful initialization	**ComPort (long)** Indicates the serial interface number, to which the Cash Register is connected. **PortSpeed (long)** Indicates communication speed. Depending on the printer model possible are 19200 and 115200.If the driver does not succeed to connect itself at the specified speed it makes an attempt for connection at the second speed as well. **DevAddress (long)** Indicates the logical number of the printer. Used in case there are more than one device connected in network. By default is 0. **Locale(long)** Not used 1: Bulgarian printer 2: Romanian printer **Language(long)** Not used 0: English 1: Bulgarian **SerialKey (wstring)**	Initializes the connection to the printer. The successful execution of this function is obligatory in order all other functions to operate. By initialization the indicated COM port is opened and is checked whether the printer is ready for work. `long Init(long ComPort, long PortSpeed, long DevAddress, long Locale, long Language, wstring Serial);`

Function	Result	Entry parameters	Description
		Not used.	
Close	Do not return result	No entry parameters	Close the connection to the fiscal printer. The command closes the COM port. `void Close();`
GetErrCodeText	**Type: wstring** Convert the error code in text message	**ErrCode (long)** Number of the error returned by the other functions **Language (long)** Language code: 0 - English 1 – Bulgarian	Error code in text message. Supports Bulgarian and English language. `wstring GetErrCodeText(in long ErrCode, in long LngCode);`
GetLastError	**Type: wstring** Detailed information for occurred error	**LngCode (long)**	Return detailed information for the occurred error. `wstring GetLastError(in long LngCode);`
CurrencyID	Property RW Type: long	**CurrencyID(long)** Valid values are 1-5	Sets/Gets currency of the receipt. The function must be performed before StartBon. If the function is not used, the program works with last used currency.
StartBon	Type: long Return error code 0 – successful execution of the command	**BonType** (long) 1 - Non-fiscal receipt 2 - Fiscal receipt 3 - Invoice 4 - Refund invoice 5 - Refund receipt **PLUNumbering (long)** Indicates if the driver shall print the PLU number (barcode).	Command for opening of new cash bon. The type of the required receipt is indicated as parameter. All parameters related to data for Invoice are used only in case that the printed receipt will be of type Invoice. `long StartBon(in long BonType, in long PLUNumbering, in wstring CashierName, in long long InvNumber, in wstring InvTaxNumber, in wstring InvVATNumber, in wstring InvCompany, in wstring InvCity, in wstring InvAddress, in wstring InvMOL, in wstring InvReceiver);`

Function	Result	Entry parameters	Description
		0 – not printed. 1 – printed. **CashierName (wstring)** Cashier name **InvNumber (wstring)** Invoice number (BonType=3) **InvTaxNumber (wstring)** Customer TAX number (BonType=3) **InvVATNumber (wstring)** VAT number of the customer (BonType=3) **InvCompany (wstring)** Company name – Invoice (BonType=3) **InvCity (wstring)** City – Invoice (BonType=3) **InvAddress (wstring)** Address – Invoice (BonType=3) **InvMOL (wstring)** MRP – Invoice (BonType=3) **InvReceiver (wstring)** Recipient – Invoice (BonType=3)	
EndBon	**Type: long** Return error code 0 – successful	No entry parameters	Closes receipt and prints it content `long EndBon();`

Function	Result	Entry parameters	Description
	execution of the command		
PLUCurrencyID	**Type: long** Return error code 0 – successful execution of the command	**CurrecncyID(long)** Valid values are 1-5	Sets PLU item currency. The function must be performed before AddPLU method. If the function is not used, the program works with last used currency
AddPLU	**Type: long** Return error code 0 – successful execution of the command	**Numb (wstring)** PLU number. Ignored by working flag for automatic numbering **Name (wstring)** PLU name – up to 12 symbols for all types of Cash Registers **Price (double)** Unit price (with decimal point) **Quantity (double)** Sold quantity(with decimal point). If the quantity is negative value, operation for returning of products is performed. **VATGroup (long)** Tax group. Integer from 1 to 4.	Command for adding a PLU to the cash receipt. Specific features: Supports operation returning of products (negative quantity is indicated). The sum of the returned PLU must be less than the sum of the bon in order this operation to be performed. It is not possible to generate receipt with negative sum. `long AddPLU(in wstring Numb, in wstring Name, in double Price, in double Quantity, in long VATGroup);`
AddLine	Type: long Return error code 0 – successful execution of the command	**Line (wstring)** Free text **Command (wstring)** Command for determining of the printing font (not working with all	Command for adding a comment text to the receipt. `long AddLine(in wstring Line, in wstring Command, in long WordWrap);`

Function	Result	Entry parameters	Description
		printers) “BOLD” – bold text “DBWIDTH” – double width “DBSTRIKE” – double height **WordWrap (long)** Whether to carry the text over the second row if it is longer.	
AddDiscount	Type: long Return error code 0 – successful execution of the command	**DiscType (long)** Discount/addon type 1 – over last PLU 2 – over sub total 3 – absolute **Value (double)** Value of discount/ surcharge. Values less than zero are considered as discounts. Values greater than zero are considered as addons.	The command adds discount or surcharge to the receipt. By the sign of the value is assessed whether it is a discount or an surcharge. The minus is handled as a discount. There must be at least one PLU item before adding a discount It is not possible to have two consecutive discounts. `long AddDiscount(in long DiscType, in double Value);`
AddBarcode	Type: long Return error code 0 – successful execution of the command	**BType (long)** Barcode type. 1 - EAN 8 2 - EAN13 3 - Code 128 6 - QR code **BHeight (long)** Not used. Field is left for compatibility with the driver for Cash Register. **BHri (long)** Not used. For compatibility only	The command adds barcode to the receipt. May be used as well as in fiscal and in non-fiscal receipts. Not supported by all models of fiscal printers. `long AddBarcode(in long BType, in long BHeight, in long BHri, in BSTR BData);`

Function	Result	Entry parameters	Description
		BData (wstring) Data for barcode. The length and symbol types depend on the type of the barcode.	
AddPayment	Type: long Return error code 0 – successful execution of the command	**Number (long)** Payment number Valid values from 1 to 11 **Name (wstring)** Name of payment Text up to 12 symbols **Ammount (double)** Amount of payment **Cource (double)** Course to payment	Command for adding a payment to the receipt. The fiscal printers support 11 types of payments. `long AddPayment(in long Number, in wstring Name, in double Ammount, in double Cource);`
CopyLastBon	Type: long Return error code 0 – successful execution of the command	No entry parameters	Not used. For compatibility. only `long CopyLastBon();`
GetClock	Type: Double Return clock of the printer	No entry parameters	Return the current date/time of the fiscal printer. `double GetClock();`
SetClock	Type: long Return error code	**Val (double)** Date and time.	Setss the fiscal printer clock. The date/time cannot be a before the last report, recorded in the fiscal memory.

Function	Result	Entry parameters	Description
	0 – successful execution of the command		`long SetClock(in double Val);`
DIsplayPrint	Type: long Return error code 0 – successful execution of the command	**Line1 (wstring)** First row of text **Line2 (wstring)** Second row of text	Print out text message on the customer display (if there is such connected to the printer). The command is executed only outside a receipt. Not used. For compatibility. only `long DisplayPrint(in wstring Line1, in wstring Line2);`
InOutMoney	Type: long Return error code 0 – successful execution of the command	**Ammount (double)** Sum for adding (deduction) Sign determines the operation Negative sum is deducted **PayType (long)** Type of payment for added/deducted sum. Possible values from 1 to 4.	Service entry and taking out of money. Interpreted as money in cash-drawer. The printer prints out service receipt with the result of the operation. The command is executed only outside a receipt. `long InOutMoney(in double Ammount, in long PayType);`
GetLastBonNumber	Type: long Return the number of last printed cash receipt.	No entry parameters	Return the number of last printed receipt. `long GetLastBonNumber();`
GetLastInvoiceNumber	**Type: wstring** Return the number of last printed invoice. If bon is not invoice,	No entry parameters	Return the number of the printed invoice. If the receipt is not invoice, the result is 0. `wstring GetLastInvoiceNumber();`

Function	Result	Entry parameters	Description
	the value is 0		
GetSerialNumber	**Type: wstring** Return ECR serial number	No entry parameters	Return the fiscal printer serial number. `wstring GetSerialNumber();`
GetFiscalNumber	**Type: wstring** Return the fiscal memory number of the ECR.	No entry parameters	Return the fiscal memory number. `wstring GetFiscalNumber();`
GetDriverVersion	Type: wstring Return the version of FP driver	No entry parameters	Return the version of FP driver. `wstring GetDriverVersion();`
GetDriverVendor	Type: wstring Return the name of driver vendor.	No entry parameters	Return the name of the driver vendor. `wstring GetDriverVendor();`
GetPrinterModel	Type: wstring Return the model of the fiscal printer.	No entry parameters	Return the model of the fiscal printer. `wstring GetPrinterModel();`
GetBonStatus	Type: long Return error code	No entry parameters	Return the status of the cash receipt. TotalAmnt : Total sum; PayedAmnt : Paid sum;

Function	Result	Entry parameters	Description
	0 – successful execution of the command		BonNumber : Receipt number; InvoiceNumber : Invoice number; Transactions : Number of transactions in the receipt `long GetBonStatus(inout double TotalAmnt, inout double PayedAmnt, inout long BonNumber, inout wstring InvoiceNumber, inout long Transactions);`
GetFiscalStatus	Type: long Return error code 0 – successful execution of the command	No entry parameters	Return the fiscal status of the printer. FiscalMode (long): Fiscalized DecimalPosition (long): Decimal point position FMRecordCount (long): Number of records in fiscal memory FMChangesCount (long): Number of changes in fiscal data FMTurnover (double): Total turnover in fiscal memory FMTurnoverVAT (double): VAT sum in fiscal memory FMLastRecordDate (double): Date of last record in fiscal memory `long GetFiscalStatus(inout long FiscalMode, inout long DecimalPosition, inout long FMRecordCount, inout long FMChangesCount, inout double FMTurnover, inout double FMTurnoverVAT, inout double FMLastRecordDate);`
GetHardwareStatus	Type: long Return error code 0 – successful execution of the command	No entry parameters	Return the hardware status of the printer. The status is consecution of zeros and ones in text form. One mean YES; zero NO. Description of positions: 1:Journal near-end paper flag 2:Receipt near-end paper flag 3:Journal end of paper flag 4:Receipt end of paper flag

Function	Result	Entry parameters	Description
			5:Receipt cover, Journal cover or 6:Journal platen is OPEN 7:Printing is being STOPPED by paper-end or paper-near end 8:Error flag 9:Drawer 2 interface connector: HIGH 10:Drawer 1 interface connector: HIGH 11:Auto cutter: Error 12:Unrecoverable error occurred 13:Auto recoverable error occurred The command is not supported by all printer versions long GetHardwareStatus(inout wstring Status);
GetXReportTotal	**Type: long** Return error code 0 – successful execution of the command	No entry parameters	Return information for accumulated daily turnover. TOTAL (Double): total sum TOTAL_VAT (Double): total VAT NbCustomers (long): number of customers NbDiscounts (long): number of discounts NbAdds (long): number of addons NbRefunds (long): number of refunds NbVoids (long): number of voids TlDiscounts (Double): discounts sum TlAdds (Double): addons sum TlRefunds (Double): refunds sum TlVoids (Double): voids sum `long GetXReportTotal(inout double TOTAL, inout double TOTAL_VAT, inout long NbCustomers, inout long NbDiscounts, inout long NbAdds, inout long NbRefunds, inout long NbVoids, inout double TlDiscounts, double TlAdds, inout double`

Function	Result	Entry parameters	Description
			`TlRefunds, double TlVoids);`
GetXReportTotalSh	Type: long Return error code 0 – successful execution of the command	No entry parameters	Return information for accumulated daily turnover – short version. TOTAL (Double): total sum TOTAL_VAT (Double): total VAT `long GetXReportTotal(inout double TOTAL, inout double TOTAL_VAT);`
GetXReportPayments	Type: long Return error code 0 – successful execution of the command	No entry parameters	Return information for accumulated daily turnover, types of payment and service entered and taken out sums TOTAL (Double) – total sum TOTAL_VAT (Double) – total VAT TotalPy1 (double): Payment sum 1 TotalPy2 (double): Payment sum 2 TotalPy3 (double): Payment sum 3 TotalPy4 (double): Payment sum 4 InPy1 (double): Sum entered payment 1 OutPy1 (double): Sum taken out payment 1 InPy2 (double): Sum entered payment 2 OutPy2 (double): Sum taken out payment 2 InPy3 (double): Sum entered payment 3 OutPy3 (double): Sum taken out payment 3 InPy4 (double): Sum entered payment 4 OutPy4 (double): Sum taken out payment 4 NbIn (long): Number of entries NbOut (long): Number of taking out `long GetXReportPayments(inout double TOTAL,`

Function	Result	Entry parameters	Description
			`inout double TOTAL_VAT, inout double TotalPy1, inout double TotalPy2, inout double TotalPy3, inout double TotalPy4, inout double InPy1, inout double OutPy1, inout double InPy2, inout double OutPy2, inout double InPy3, inout double OutPy3, inout double InPy4, inout double OutPy4, inout long NbIn, inout long NbOut);`
GetXReportPaymentsEx	Type: long Return error code 0 – successful execution of the command	No entry parameters	Return information for accumulated daily turnover, types of payment and service entered and taken out sums TOTAL (Double) – total sum TOTAL_VAT (Double) – total VAT TotalPy1 (double): Payment sum 1 TotalPy2 (double): Payment sum 2 TotalPy3 (double): Payment sum 3 TotalPy4 (double): Payment sum 4 TotalPy5 (double): Payment sum 5 TotalPy6 (double): Payment sum 6 TotalPy7 (double): Payment sum 7 TotalPy8 (double): Payment sum 8 InPy1 (double): Sum entered payment 1 OutPy1 (double): Sum taken out payment 1 InPy2 (double): Sum entered payment 2 OutPy2 (double): Sum taken out payment 2 InPy3 (double): Sum entered payment 3 OutPy3 (double): Sum taken out payment 3 InPy4 (double): Sum entered payment 4 OutPy4 (double): Sum taken out payment 4 InPy5 (double): Sum entered payment 5 OutPy5 (double): Sum taken out payment 5

Function	Result	Entry parameters	Description
			InPy6 (double): Sum entered payment 6 OutPy6 (double): Sum taken out payment 6 InPy7 (double): Sum entered payment 7 OutPy7 (double): Sum taken out payment 7 InPy8 (double): Sum entered payment 8 OutPy8 (double): Sum taken out payment 8 NbIn (long): Number of entries NbOut (long): Number of taking out `long GetXReportPayments(inout double TOTAL, inout double TOTAL_VAT, inout double TotalPy1, inout double TotalPy2, inout double TotalPy3, inout double TotalPy4, TotalPy5, TotalPy6, TotalPy7, TotalPy8, inout double InPy1, inout double OutPy1, inout double InPy2, inout double OutPy2, inout double InPy3, inout double OutPy3, inout double InPy4, inout double OutPy4, inout double InPy5, inout double OutPy5, inout double InPy6, inout double OutPy6, inout double InPy7, inout double OutPy7, inout double InPy8, inout double OutPy8, inout long NbIn, inout long NbOut);`
GetXReportVAT	Type: long Return error code 0 – successful execution of the command	No entry parameters	Return information for accumulated daily turnover by tax groups. TOTAL (double): total sum TOTAL_VAT (double): total VAT TotalTaxA (double): sum tax group A TotalTaxB (double): sum tax group B TotalTaxC (double): sum tax group C

Function	Result	Entry parameters	Description
			TotalTaxD (double): sum tax group D TotalTaxE (double): sum tax group E TotalTaxF (double): sum tax group F TotalTaxG (double): sum tax group G TotalTaxH (double): sum tax group H `long GetXReportVAT(inout double TOTAL, inout double TOTAL_VAT, inout double TotalTaxA, inout double TotalTaxB, inout double TotalTaxC, inout double TotalTaxD, inout double TotalTaxE, inout double TotalTaxF, inout double TotalTaxG, inout double TotalTaxH);`
RunFMBlockReport	Type: long Return error code 0 – successful execution of the command	StartBlock (long) Number of memory start block (From 1 to end block) EndBlock (long) Number of memory end block (From start block to the number of records) Detailed (long) Flag indicating whether the report to be detailed	Start printing of report from the printer fiscal memory. The command works only in case the printer is fiscalized. The command is executed only outside a receipt. `long RunFMBlockReport(in long StartBlock, in long EndBlock, in long Detailed);`
RunFMDateReport	Type: long Return error code 0 – successful execution of the command	StartDate (long) Start date of report EndDate (long) End date of report	Start printing of report from the printer fiscal memory. The command works only in case the printer is fiscalized. The command is executed only outside a receipt. `long RunFMDateReport(in double StartDate, in double EndDate, in long Detailed);`

Function	Result	Entry parameters	Description
		Detailed (long) Flag indicating whether the report to be detailed	
RunXReport	Type: long Return error code 0 – successful execution of the command	No entry parameters	Start daily report without zeroing. The command is executed only outside a receipt. `long RunXReport();`
RunZReport	Type: long Return error code 0 – successful execution of the command	No entry parameters	Start daily report with zeroing. The command is executed only outside a receipt. `long RunZReport();`
RunPLUReport	Type: long Return error code 0 – successful execution of the command	ZeroFlag (long): Flag indicating whether the report to be voided (0-NO; 1-YES) PrintFlag (long): Flag indicating whether the report to be printed (0-NO; 1-YES)	Start report by PLUs. The command is executed only outside a receipt. `long RunPLUReport(in long ZeroFlag, in long PrintFlag);`
RunPrinterTest	Type: long Return error code 0 – successful	No entry parameters	Start printing of test receipt. The command is executed only outside a receipt. `long RunPrinterTest();`

Function	Result	Entry parameters	Description
	execution of the command		

SaveEJournalToDisk	Type: long
	
	Property
		No entry parameters
		(0-NO; 1-Yes)
			Get or Set whether electronic journal will be written to the disk.
			`long Get_SaveEJournalToDisk();`
			`Set_SaveEJournalToDisk(in long Value);`

EJournalPath	Type: wstring
	
	Property
		No entry parameters	Get or Set path to store the electronic journal, when saved to disk.
			
			`wstring EJournalPath();`
			`EJournalPath(in wstring Value);`

GenerateLogFile	Type: long Property	No entry parameters (0-NO; 1-Yes)	Get or Set whether log file shall be created
LogFilePath	Type: wstring Property	No entry parameters	Get or Set the path of the log file
CashDrawerPuls	Type: long Property		Pulse width to open cash drawer. If is set to zero – drawer will not opened. Default is set to 15[ms]

Description of the error codes:

Table 2	
Code	Description
1	„Error COM port initialization” Impossibility to open the selected COM port. Occur by attempting to open non-existing port or when the port is occupied by another program.
2	„Error communication with printer” Occur by irregularities in communication protocol. Wrong packets, check sums, printer version, etc.
3	„No connection to printer” Occur when the printer does not respond to the sent command (when is switched off or is just missing).
4	„Error induced by the printer” Occur when the printer itself returns error message.(No paper, reset ram, overfilled fiscal memory ...)Detailed information for the error to be found from **GetLastError**
5	Error induced by the driver Occur by program error
6	Invalid registration key Not apply to the driver for fiscal printer
7	The command is not supported by the printer Depend on the version of the fiscal printer
8	Invalid input data
100	Other error Information for the error to be found from **GetLastError**

In all cases of occurred error, the function **GetLastError** returns message that gives information for the error. Even the above-mentioned errors give only the frame but not the specific reason. In order to find out the specific reason it is necessary to call out the function **GetLastError**.

When an error occurs, the program creates a text file *„AXDriverErrLog.txt* “containing a log of the events that caused the error. By default a log is created in a sub folder „Log “of the program's working folder. The user can change the location of the file through the property **GenerateLogFile**