Оглавление

1. О библиотеке

Динамически подключаемая 32-х разрядная библиотека функций **km100.dll** предназначена для построения систем учета товародвижения и автоматизации работы оператора на базе ЧПМ **КМ АМС-100**, **КМ АМС-100М** и **КМ АМС-мини 100** в режиме фискального регистратора при помощи персонального компьютера работающего под управлением Windows XP/Windows Vista|Eindows 7. Взаимодействие ПК с ККМ осуществляется по интерфейсу RS232. Соглашения о вызовах - stdcall, соглашения об именах - pascal.

Особенностью реализации данной библиотеки является то, что информацию о предварительно отобранных продаваемых товарах (вместе с сопроводительной информацией: значением скидки/надбавки на чек, суммы оплачиваемых наличных и т.д.) библиотека содержит в себе, управление этой внутренней базой данных организовано при помощи ряда функций, позволяющих добавлять/удалять покупки, получать информацию о содержании базы. Использование такого подхода позволило использовать для принимаемых и возвращаемых функциями параметров простые типы данных (указатель на нуль-терминированную строку (PChar), Integer и Double), что обеспечивает совместимость данной библиотеки с большинством сред разработки приложений для Win32.

Замечания по работе библиотеки можно отправлять на e-mail:

info@vtsoft.ru

Наш сайт в интернет:

vtsoft.ru

2. Ограничения

Программный модуль подготовлен с использованием зарегистрированного продукта корпорации Borland. Среда программирования Delphi.

Библиотека защищена от копирования и не предназначена для свободного использования сверх ограниченного числа инсталляций с поставляемого носителя. Возможно использование данного программного продукта без регистрации в демонстрационном режиме (с ограничением разрядности стоимости покупки до x.xx руб. и ограничением числа покупок до трех в одном чеке), однако декомпиляция и доработка библиотеки является нарушением авторских и имущественных прав и может повлечь за собой судебный иск в соответствии с законами РФ о правовой охране программного обеспечения.

Внимание! Данная библиотека предназначена для использования со следующими ЧПМ:

- КМ АМС-100 с установленной версией ПО 1.0.0 КМ.
- КМ АМС-100М с установленной версией ПО 1.0.1 КМ.
- КМ АМС-мини 100 с установленной версией ПО 1.0.1 КМ.

На ККМ должен быть установлен режим "**Есть сеть**". Сетевой номер ККМ должен быть установлен в значение **1**.

Сетевой номер может быть запрограммирован следующими утилитами:

- КМ АМС-100 — NetNum100K.exe.
- КМ АМС-100М — NetNum100MK_01.exe.
- КМ АМС-мини 100 — NetNum100K.exe.

3. Функции. Общие замечания

По логике выполняемых действий все функции можно разделить на несколько групп:

- функции, обеспечивающие коммуникации;
- функции управления базой описателей товаров;
- функции получения статистики по ККМ;
- функции назначения обработчиков событий;
- функции управления режимом;
- сервисные (дополнительные) функции.

Функции первой группы предназначены для настройки обмена, установления связи с ККМ и освобождения используемого COM порта, по окончании работы с кассовым аппаратом.

Функции управления базой описателей товаров позволяют манипулировать данными, содержащими информацию о покупках, которые попадут в чек, настраивать параметры выводимого чека (чек возврата/продажи, наличный/безналичный расчет, значение скидки/надбавки на чек и т.д.), а также формировать заголовок и окончание чека. Наименования всех функций данной группы предваряется префиксом cb.

Функции получения статистики по ККМ позволяют читать из ЧПМ информацию о произведенных продажах, получать заводской номер ЧПМ, версию ПО ЧПМ и т.п.

Данная версия библиотеки может генерировать события при помощи механизма Windows CallBack, для чего необходимо определить в создаваемом приложении обработчики событий и при помощи функций назначения обработчиков передать в библиотеку указатели на них.

В группе функции управления режимом находится только одна функция, предназначенная для передачи ЧПМ информации о чеке и инициации печати чека.

К группе сервисных функций отнесены функции блокировки/разблокировки клавиатуры ЧПМ, печати строки на принтере ЧПМ, а также функции получения дополнительной информации об ошибках.

Значения, возвращаемые некоторыми функциями, будут актуальными только после установления связи с ККМ, хотя и не будут возвращать сообщения об ошибках. Это связано с тем, что при данной операции один раз происходит вычитывание настроек ЧПМ, необходимых для корректной работы библиотеки. По этой причине настоятельно рекомендуется использовать все функции данной библиотеки (за исключением SetSupplierCode) только после успешного завершения функции ConnectKKM.

4. Функции. Возвращаемые ошибки

0 - Операция успешно завершена.
1 - Операция не поддерживается ЧПМ.
2 - Ошибка инициализации порта.
3 - Ошибка интерфейса с ЧПМ.
4 - Данная версия ПО ЧПМ не поддерживается.
5 - ЧПМ не в исходном состоянии режима "Касса".
6 - Необходимо произвести подключение к ЧПМ.
7 - Необходимо вывести контрольную ленту.
8 - Контрольная память оплат не содержит покупок.
9 - Покупка с указанным номером в КЛ отсутствует.
10 - ЧПМ не готова к проведению операции.
11 - Покупка с указанным номером в базе описателей товаров отсутствует.
12 - Недопустимый номер отдела.
13 - Недопустимое значение оплачиваемых наличных.
14 - Недопустимое значение стоимости покупки.
15 - Недопустимое значение количества товара.
16 - База описателей товаров переполнена.
17 - Недопустимое значение скидки/надбавки.
18 - Недостаточно наличных в кассе.
19 - Недостаточно наличных для оплаты покупок.
20 - Получена покупка с нулевой стоимостью.
21 - Заголовок чека не может содержать более 3 строк.
22 - Окончание чека не может содержать более 3 строк.
23 - Строка с таким номером отсутствует в заголовке чека.
24 - Строка с таким номером отсутствует в окончании чека.
25 - На ЧПМ не включен режим "Есть сеть".
26 - Вывод чека был отменен.
27 - Отсутствует бумага в ТПУ.

5. Функции, обеспечивающие коммуникации

5.1. SetSupplierCode

Описание:

procedure SetSupplierCode(Code: PChar);

Принимаемые параметры:

Code - строковое представление кода поставщика.

Процедура предназначена для указания кода поставщика для ККМ, с которой необходимо установить связь. Процедура должна вызываться до попытки установки связи. В результате ее выполнения происходит автоматический вызов процедуры DisconnectKKM.

Пример

```
procedure TDemoForm.ConnectButton1Click(Sender: TObject);
begin
  SetSupplierCode('7463546576');
  if ConnectKKM(1) <> 1 then
    ShowMessage(GetErrorMsg);
end;
```

5.2. ConnectKKM

Описание:

function ConnectKKM(Port: Integer): Integer;

Принимаемые параметры:

Port - номер последовательного порта.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Данная функция используется для настройки интерфейса (как физических параметров линии, так и логики передачи). Функцию необходимо использовать до использования любой другой (но после SetSupplierCode). После однократного использования функции, порт Вашего компьютера будет настроен, и необходимость в применении данной функции в дальнейшем отпадет. Отметим, что сделанные настройки отменятся после завершения приложения, использующего библиотеку. Перед вызовом данной функции ЧПМ должна быть подключена к ПК и включена.

Если Вам потребуется использовать порт не в рамках приложения, использующего

настоящую библиотеку, необходимо предварительно освободить порт функцией DisconnectKKM.

Внимание! При вызове данной функции библиотека пытается установить связь с ЧПМ, сетевой номер которой равен **1**.

Пример

```
procedure TDemoForm.ConnectButton1Click(Sender: TObject);
begin
  SetSupplierCode('7463546576');
  if ConnectKKM(1) <> 1 then         //ЧПМ подключена к COM1
    ShowMessage(GetErrorMsg);
end;
```

5.3. DisonnectKKM

Описание:

procedure DisconnectKKM;

Данная процедура используется при завершении работы с ЧПМ для освобождения последовательного порта.

Пример

```
procedure TDemoForm.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
begin
  DisconnectKKM;
end;
```

6. Функции управления базой описателей товаров

6.1. cbAddTitleLine

Описание:

function cbAddTitleLine(Line: PChar): Integer;

Принимаемые параметры:

Line - указатель на нуль-терминированную строку, содержащую строку заголовка чека.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для добавления строки в заголовок чека и добавляет одну строку в массив строк заголовка, хранящийся в загруженном экземпляре библиотеки. Длина строки заголовка не должна превышать 18 символов для КМ АМС-100/КМ АМС-мини 100 и 24 символа для КМ АМС-100М. Формировать заголовок достаточно один раз за сеанс работы с библиотекой. Если на момент запроса от ЧПМ пользователем не будет определено ни одной строки заголовка, при выводе чека ЧПМ напечатает заголовок, запрограммированный в ней.

Пример

```
procedure TDemoForm.cbAddTitleLineButtonClick(Sender: TObject);
begin
  if cbAddTitleLine(PChar('Начало'))<>1 then
    ShowMessage(GetErrorMsg);
end;
```

6.2. cbGetTitleLinesCount

Описание:

function cbGetTitleLinesCount: Integer;

Возвращаемые значения:

Число строк заголовка чека, определенных пользователем.

Функция предназначена для получения числа строк заголовка чека, определенных пользователем и хранящихся во внутреннем массиве загруженного экземпляра библиотеки.

Пример

```
procedure TDemoForm.cbGetTitleLinesCountButtonClick(Sender:
TObject);
var
  Count: Integer;
begin
  Count:=cbGetTitleLinesCount;
  ShowMessage('Число строк в заголовке чека = '+IntToStr(Count));
end;
```

6.3. cbGetTitleLine

Описание:

function cbGetTitleLine(Index: Integer; var Line: PChar): Integer;

Принимаемые параметры:

Index - номер строки заголовка, которую необходимо получить;

Line - переменная, через которую возвращается указатель на нуль-терминированную строку, содержащую текст запрошенной строки заголовка.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция позволяет получить текст строки заголовка чека, добавленной ранее во внутренний массив загруженного экземпляра библиотеки при помощи функции cbAddTitleLine.

Пример

```
procedure TDemoForm.ReadTitleButtonClick(Sender: TObject);
//Читаем весь заголовок
var
  i, Count: Integer;
  ResultStr: String;
  TempLine: PChar;
  S: String[24];
begin
  Count:=cbGetTitleLinesCount; //Получаем число строк заголовка
  ResultStr:='';
  TempLine:=@S;
```

```
  for i:=1 to Count do
    begin
      if cbGetTitleLine(i, TempLine)<>1 then  //Читаем строку
        begin
          ShowMessage(GetErrorMsg);
          Exit;
        end;
      ResultStr:=ResultStr+String(TempLine)+#13+#10;
    end;
  ShowMessage(ResultStr);
end;
```

6.4. cbClearTitle

Описание:

procedure cbClearTitle;

Процедура служит для удаления всех строк заголовка чека, добавленных ранее во внутренний массив загруженного экземпляра библиотеки при помощи функции cbAddTitleLine.

Пример

```
procedure TDemoForm.cbClearTitleButtonClick(Sender: TObject);
begin
  cbClearTitle;
end;
```

6.5. cbAddBottomLine

Описание:

function cbAddBottomLine(Line: PChar): Integer;

Принимаемые параметры:

Line - указатель на нуль-терминированную строку, содержащую строку окончания чека.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для добавления строки в окончание чека и добавляет одну строку в массив строк окончания, хранящийся в загруженном экземпляре библиотеки. Длина строки заголовка не должна превышать 18 символов для КМ АМС-100/КМ АМС-мини 100 и 24 символа для КМ АМС-100М. Формировать

окончание достаточно один раз за сеанс работы с библиотекой. Если на момент запроса от ЧПМ пользователем не будет определено ни одной строки окончания, при выводе чека ЧПМ напечатает окончание, запрограммированное в ней.

Пример

```
procedure TDemoForm.cbAddBottomLineButtonClick(Sender: TObject);
begin
  if cbAddBottomLine(PChar('Конец'))<>1 then
    ShowMessage(GetErrorMsg);
end;
```

6.6. cbGetBottomLinesCount

Описание:

function cbGetBottomLinesCount: Integer;

Возвращаемые значения:

Число строк окончания чека, определенных пользователем.

Функция предназначена для получения числа строк окончания чека, определенных пользователем и хранящихся во внутреннем массиве загруженного экземпляра библиотеки.

Пример

```
procedure TDemoForm.cbGetBottomLinesCountButtonClick(Sender:
TObject);
var
  Count: Integer;
begin
  Count:=cbGetBottomLinesCount;
  ShowMessage('Число строк в окончании чека = '+IntToStr(Count));
end;
```

6.7. cbGetBottomLine

Описание:

function cbGetBottomLine(Index: Integer; var Line: PChar): Integer;

Принимаемые параметры:

Index - номер строки окончания, которую необходимо получить;

Line - переменная, через которую возвращается указатель на нуль-терминированную строку, содержащую текст запрошенной строки окончания.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция позволяет получить текст строки окончания чека, добавленной ранее во внутренний массив загруженного экземпляра библиотеки при помощи функции cbAddBottomLine.

Пример

```
procedure TDemoForm.ReadBottomButtonClick(Sender: TObject);
//Читаем весь подвал чека
var
  i, Count: Integer;
  ResultStr: String;
  TempLine: PChar;
  S: String[18];
begin
  Count:=cbGetBottomLinesCount;  //Получаем число строк окончания
  ResultStr:='';
  TempLine:=@S;
  for i:=1 to Count do
    begin
      if cbGetBottomLine(i, TempLine)<>1 then  //Читаем строку
        begin
          ShowMessage(GetErrorMsg);
          Exit;
        end;
      ResultStr:=ResultStr+String(TempLine)+#13+#10;
    end;
  ShowMessage(ResultStr);
end;
```

6.8. cbClearBottom

Описание:

procedure cbClearBottom;

Процедура служит для удаления всех строк окончания чека, добавленных ранее во внутренний массив загруженного экземпляра библиотеки при помощи функции cbAddBottomLine.

Пример

```
procedure TDemoForm.cbClearBottomButtonClick(Sender: TObject);
begin
  cbClearBottom;
end;
```

6.9. cbAddSale

Описание:

function cbAddSale(Name: PChar; Price, Qty: Double; Section: Integer): Integer;

Принимаемые параметры:

Name - указатель на нуль-терминированную строку, содержащую наименование товара;

Price - стоимость единицы товара;

Qty - количество товара;

Section - отдел по которому будет продаваться товар.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для добавления информации о продаваемом товаре в базу описателей товаров библиотеки. Допустимо добавление до 40 покупок в один чек. Порядок очистки базы описателей товаров определяется при помощи процедуры cbSetClearBufMode.

Пример

```
procedure TDemoForm.cbAddSaleButtonClick(Sender: TObject);
var
  GoodName: String;
  GoodPrice, GoodQty: Double;
  GoodSection: Integer;
begin
  GoodName:='Новый товар';
  GoodPrice:=12.50;
  GoodQty:=1;
  GoodSection:=1;
  if cbAddSale(PChar(GoodName),GoodPrice, GoodQty, GoodSection) <>
1 then
```

```
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
end;
```

6.10. cbGetSalesCount

Описание:

function cbGetSalesCount: Integer;

Возвращаемые значения:

Число описателей товаров во внутренней базе библиотеки.

Функция возвращает число описателей товаров, находящихся на данный момент во внутренней базе загруженного экземпляра библиотеки.

Пример

```
procedure TDemoForm.cbGetSalesCountButtonClick(Sender: TObject);
var
  Count: Integer;
begin
  Count:=cbGetSalesCount;
  ShowMessage('Число описателей товаров в базе библиотеки =
'+IntToStr(Count));
end;
```

6.11. cbGetSale

Описание:

function cbGetSale(Index: Integer; var Name: PChar; var Price, Qty: Double; var Section: Integer): Integer;

Принимаемые параметры:

Index - номер описателя товара в базе данных библиотеки;

Name - переменная, через которую возвращается указатель на нуль-терминированную строку, содержащую наименование товара;

Price - переменная, через которую возвращается стоимость товара;

Qty - переменная, через которую возвращается количество товара;

Section - переменная, через которую возвращается отдел, по которому должен продаваться товар.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для получения описателя товара из внутренней базы данных загруженного экземпляра библиотеки по его порядковому номеру.

Пример

```
procedure TDemoForm.GetCheckButtonClick(Sender: TObject);
var
  S: String[51];
  GoodName: PChar;
  Price, Qty: Double;
  Section: Integer;
  ResultStr: String;
begin
  GoodName:=@S;
  if cbGetSale(1, GoodName, Price, Qty, Section) <> 1 then
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
  ResultStr:=String(GoodName)+'  '+FormatFloat('0.00
''p.''',Price)+
  '  '+FormatFloat('0.000',Qty)+'  '+IntToStr(Section);
  ShowMessage(ResultStr);
end;
```

6.12.cbDeleteSale

Описание:

function cbDeleteSale(Index: Integer): Integer;

Принимаемые параметры:

Index - порядковый номер описателя товара в базе данных библиотеки.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция удаляет указанный описатель товара из базы данных библиотеки,

смещая описатели, расположенные после удаляемого на одну позицию вверх.

Пример

```
procedure TDemoForm.cbDeleteSaleButtonClick(Sender: TObject);
begin
  if cbDeleteSale(1)<>1 then
    ShowMessage(GetErrorMsg);
end;
```

6.13.cbClearSales

Описание:

procedure cbClearSales;

Процедура очищает базу данных описателей товаров.

Пример

```
procedure TDemoForm.cbClearSalesButtonClick(Sender: TObject);
begin
  cbClearSales;
end;
```

6.14.cbSetCreditMode

Описание:

procedure cbSetCreditMode(Mode: Integer);

Принимаемые параметры:

Mode - параметр, определяющий вид платежа:

0 - чек за наличный расчет;

1 - чек за безналичный расчет.

Процедура позволяет установить тип платежа в выводимом чеке. Установки, произведенные данной процедурой будут распространяться на все последующие выводимые чеки. Для изменения типа платежа необходимо вызвать cbSetCreditMode с другим значением принимаемого параметра. Текущее состояние вида платежа можно получить при помощи функции cbGetCreditMode.

Пример

```
procedure TDemoForm.cbSetCreditModeButtonClick(Sender: TObject);
begin
  cbSetCreditMode(1); //Чек за безналичный расчет
end;
```

6.15.cbGetCreditMode

Описание:

function cbGetCreditMode: Integer;

Возвращаемые значения:

Вид платежа в выводимом чеке:

0 - чек за наличный расчет;

1 - чек за безналичный расчет.

Функция предназначена для получения текущего вида платежа, установленного при помощи процедуры cbSetCreditMode.

Пример

```
procedure TDemoForm.cbGetCreditModeButtonClick(Sender: TObject);
var
  ResultStr: String;
begin
  ResultStr:='Будет выведен чек за ';
  if cbGetCreditMode = 1 then
    ResultStr:=ResultStr+'безналичный расчет.'
  else
    ResultStr:=ResultStr+'наличный расчет.';
  ShowMessage(ResultStr);
end;
```

6.16.cbSetDiscountValue

Описание:

function cbSetDiscountValue(Value: Integer): Integer;

Принимаемые параметры:

Value - целочисленное значение процента скидки/надбавки на чек (допустимый диапазон значений от -99 до 99).

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция устанавливает скидку/надбавку на чек в процентном выражении. Сброс установленного значения в ноль при успешном выводе чека определяется процедурой cbSetClearBufMode. Использование данной процедуры имеет смысл только при запрограммированном на ККМ режиме работы со скидками/надбавками. Определить запрограммирован ли указанный режим можнго при помощи функции GetDiscountMode.

Пример

```
procedure TDemoForm.cbSetDiscountValueButtonClick(Sender:
TObject);
begin
  if cbSetDiscountValue(-3) <> 1 then //Установим скидку на чек =
3%
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
end;
```

6.17.cbGetDiscountValue

Описание:

function cbGetDiscountValue: Integer;

Возвращаемые значения:

Процентное выражение значения скидки/надбавки на чек.

Функция возвращает процентное выражение значения скидки/надбавки на чек, установленное при помощи cbSetDiscountValue.

Пример

```
procedure TDemoForm.cbGetDiscountValueButtonClick(Sender:
TObject);
begin
  ShowMessage('Скидка/Надбавка на чек =
'+IntToStr(cbGetDiscountValue)+'%');
end;
```

6.18.cbSetReturnMode

Описание:

procedure cbSetReturnMode(Mode: Integer);

Принимаемые параметры:

Mode - параметр, определяющий тип выводимого чека:

0 - чек продажи;

1 - чек возврата.

Процедура позволяет установить тип выводимого чека. Установки произведенные данной процедурой будут распространяться на все последующие выводимые чеки. Для изменения типа выводимого чека необходимо вызвать cbSetCreditMode с другим значением принимаемого параметра.

Пример

```
procedure TDemoForm.cbSetReturnModeButtonClick(Sender: TObject);
begin
  cbSetReturnMode(1); //1 - чек возврата товара
end;
```

6.19.cbGetReturnMode

Описание:

function cbGetReturnMode: Integer;

Возвращаемые значения:

Тип выводимого чека:

0 - чек продажи;

1 - чек возврата.

Функция предназначена для получения текущего типа выводимого чека, установленного при помощи процедуры cbSetReturnMode.

Пример

```
procedure TDemoForm.cbGetReturnModeButtonClick(Sender: TObject);
var
  ResultStr: String;
begin
  ResultStr:='Будет выведен чек ';
  if cbGetReturnMode = 1 then
    ResultStr:=ResultStr+'возврата.'
  else
```

```
    ResultStr:=ResultStr+'продажи.';
  ShowMessage(ResultStr);
end;
```

6.20.cbSetCash

Описание:

function cbSetCash(Value: Double): Integer;

Принимаемые параметры:

Value - сумма наличных, переданных кассиру покупателем, для оплаты покупок.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для установки значение оплачиваемых наличных. Используется обычно для того, чтобы ЧПМ при выводе чека подсчитала сдачу. Если в этом нет необходимости, то следует передать в качестве принимаемого параметра нулевое значение. Порядок обнуления установленного значения оплачиваемых наличных при успешном выводе чека определяется при помощи процедуры cbSetClearBufMode.

Внимание! На данный момент функция не поддерживается ЧПМ.

Пример

```
procedure TDemoForm.cbSetCashButtonClick(Sender: TObject);
begin
  cbSetCash(100); //Сумма, полученная от покупателя
end;
```

6.21.cbGetCash

Описание:

function cbGetCash: Double;

Возвращаемые значения:

Сумма наличных, переданных кассиру покупателем, для оплаты покупок.

Функция необходима для получения значения оплачиваемых наличных, установленного при помощи функции cbSetCash.

Пример

```
procedure TDemoForm.cbGetCashButtonClick(Sender: TObject);
begin
  ShowMessage('Оплачивыемые наличные = '+FormatFloat('0.00' ''
p.''',cbGetCash));
end;
```

6.22. cbSetClearBufMode

Описание:

procedure cbSetClearBufMode(Mode: Integer);

Принимаемые параметры:

Mode - параметр, определяющий порядок очистки базы описателей товаров:

0 - не очищать;

1 - очищать.

Процедура позволяет настроить порядок очистки базы описателей товаров после успешного вывода чека. При передаваемом параметре равном 1 после успешного вывода чека из базы будут удалены все описатели товаров, а значение оплачиваемых наличных (установленное функцией cbSetCash) и процентное выражение скидки/надбавки на чек (установленное функцией cbSetDiscountValue) будут сброшены в ноль. По умолчанию база описателей очищается после успешного вывода чека.

Пример

```
procedure TDemoForm.cbSetClearBufModeButtonClick(Sender: TObject);
begin
  cbSetClearBufMode(1); //Очищать базу описателей после вывода
чека
end;
```

6.23. cbGetClearBufMode

Описание:

function cbGetClearBufMode: Integer;

Возвращаемые значения:

Значение параметра, определяющего порядок очистки базы описателей товаров:

0 - не очищать;

1 - очищать.

Пример

```
procedure TDemoForm.cbGetClearBufModeButtonClick(Sender: TObject);
var
  ResultStr: String;
begin
  ResultStr:='После успешного вывода чека база описателей ';
  if cbGetClearBufMode = 1 then
    ResultStr:=ResultStr+'будет очищена.'
  else
    ResultStr:=ResultStr+'не будет очищена.';
  ShowMessage(ResultStr);
end;
```

6.24.cbGetSum

Описание:

function cbGetSum: Double;

Возвращаемые значения:

Текущее значение суммы по чеку.

Функция позволяет получить текущее значение суммы по чеку с учетом скидок/надбавок. Сумма подсчитывается по описателям, находящимся в базе данных библиотеки и изменяется при проведении любой операции над базой: добавление/удаление описателя, изменение значения скидки/надбавки на чек и т.д.

Пример

```
procedure TDemoForm.cbGetSumButtonClick(Sender: TObject);
begin
  ShowMessage('Сумма по чеку = '+
              FormatFloat('0.00' '' р.''',cbGetSum));
end;
```

7. Функции получения статистики по ККМ

7.1. GetCashSum

Описание:

function GetCashSum(var Sum: Double): Integer;

Принимаемые параметры:

Sum - переменная, через которую возвращается сумма наличных в кассе.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для получения суммы наличных в кассе, накопленных за текущую смену.

Пример

```
procedure TDemoForm.GetCashSumButtonClick(Sender: TObject);
var
  TempVar: Double;
begin
  if GetCashSum(TempVar) <> 1 then
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
  ShowMessage('Сумма наличных в кассе = '+FormatFloat('0.00
''р.''',TempVar));
end;
```

7.2. GetKKMNum

Описание:

function GetKKMNum(var KKMNum: Integer): Integer;

Принимаемые параметры:

KKMNum - переменная, через которую возвращается заводской номер ЧПМ.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для получения заводского номера ЧПМ, с которой на данный момент установлено соединение.

Пример

```
procedure TDemoForm.GetKKMNumButtonClick(Sender: TObject);
var
  TempVar: Integer;
begin
  if GetKKMNum(TempVar) <> 1 then
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
  ShowMessage('Заводской номер ККМ = '+IntToStr(TempVar));
end;
```

7.3. GetKLNum

Описание:

function GetKLNum(var KLNum: Integer): Integer;

Принимаемые параметры:

KLNum - переменная, через которую возвращается номер текущей контрольной ленты.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция позволяет получить номер текущей контрольной ленты в ЧПМ.

Пример

```
procedure TDemoForm.GetKLNumButtonClick(Sender: TObject);
var
  TempVar: Integer;
begin
  if GetKLNum(TempVar) <> 1 then
    begin
      ShowMessage(GetErrorMsg);
      Exit;
```

```
    end;
  ShowMessage('Номер текущей контрольной ленты =
'+IntToStr(TempVar));
end;
```

7.4. GetNI

Описание:

function GetNI(var NI: Double): Integer;

Принимаемые параметры:

NI - переменная, через которую возвращается значение необнуляемого итога.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для получения значения необнуляемого итога по ЧПМ.

Пример

```
procedure TDemoForm.GetNIButtonClick(Sender: TObject);
var
  TempVar: Double;
begin
  if GetNI(TempVar) <> 1 then
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
  ShowMessage('Необнуляемый итог = '+FormatFloat('0.00
''р.''',TempVar));
end;
```

7.5. GetKKMVers

Описание:

function GetKKMVers: Integer;

Возвращаемые значения:

Номер версии ПО ККМ.

Функция возвращает номер версии ПО ЧПМ без разделительных точек, т.е. для

версии 1.0.1 будет возвращено значение 101.

Пример

```
procedure TDemoForm.GetKKMVersButtonClick(Sender: TObject);
begin
  ShowMessage('Версия ПО ККМ = '+FloatToStr(GetKKMVers/10));
end;
```

7.6. ReadSaleFromKL

Описание:

function ReadSaleFromKL(SaleNum: Integer; var Operation, Section, Cashier, Discount, Credit, StartCheck, H, M: Integer; var Sum: Double): Integer;

Принимаемые параметры:

SaleNum - номер покупки, хранящейся в контрольной памяти оплат и по которой необходимо получить информацию;

Operation - переменная, через которую возвращается код операции:

1 - продажа;

2 - возврат;

3 - внесение суммы;

4 - снятие выручки;

Section - переменная, через которую возвращается значение отдела по которому был продан товар;

Cashier - переменная, через которую возвращается номер кассира, выводишего чек;

Discount - переменная, через которую возвращается процентное выражение скидки/надбавки с которой был продан товар;

Credit - переменная, через которую возвращается признак безналичнго расчета:

1 - товар был продан за безналичный расчет;

0 - товар был продан за наличные деньги;

StartCheck - переменная, через которую возвращается признак начала нового чека:

1 - запись является первой в чеке;

0 - запись не является первой в чеке;

H - переменная, через которую возвращается значение часа вывода чека;

M - переменная, через которую возвращается значение минуты вывода чека;

Sum - переменная, через которую возвращается сумма операции (без учета

скидки/надбавки);

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для получения информации по покупкам или операциям внесения/снятия сумм, хранящимся в контрольной памяти оплат. Возвращаемые функцией данные позволяют получить более подробную информацию по пробитым на ККМ покупкам, чем данные представленные в X - отчете.

Пример

```
procedure TDemoForm.ReadSaleFromKLButtonClick(Sender: TObject);
var
  Section, Credit, Discount, Cashier, StartCheck, H, M, Operation:
Integer;
  Sum: Double;
  TempStr: String;
begin
  NumEditForm.InfoLabel.Caption:='Укажите номер покупки';
  NumEditForm.ShowModal;
  DemoForm.Refresh;
  if ReadSaleFromKL(NumEditForm.SpinEdit.Value, Operation,
Section, Cashier, Discount, Credit, StartCheck, H, M, Sum) <> 1
then
    begin
      ShowMessage(GetErrorMsg);
      Exit;
    end;
  TempStr:='Начало чека - ';
  if StartCheck = 1 then
    TempStr:=TempStr + 'да' + #13#10
  else
    TempStr:=TempStr + 'нет' + #13#10;
  TempStr:=TempStr + 'Операция - ';
  case Operation of
    1: TempStr:=TempStr + 'продажа' + #13#10;
    2: TempStr:=TempStr + 'возврат' + #13#10;
```

```
    3: TempStr:=TempStr + 'внесение суммы' + #13#10;
    4: TempStr:=TempStr + 'снятие выручки' + #13#10;
  end;
  TempStr:=TempStr + 'Время вывода чека - ' + FormatFloat('00', H)
+ ':' + FormatFloat('00', M) + #13#10;
  TempStr:=TempStr + 'Кассир № ' + IntToStr(Cashier) + #13#10;
  TempStr:=TempStr + 'Отдел № ' + IntToStr(Section) + #13#10;
  TempStr:=TempStr + 'Тип оплаты - ';
  if Credit = 1 then
    TempStr:=TempStr + 'БЕЗНАЛ' + #13#10
  else
    TempStr:=TempStr + 'НАЛ' + #13#10;
  TempStr:=TempStr + 'Скидка/Надбавка = ' + IntToStr(Discount) +
'%' + #13#10;
  TempStr:=TempStr + 'Сумма без скидки/надбавки = ' +
FormatFloat('0.00', Sum);
  ShowMessage(TempStr);
end;
```

7.7. GetDiscountMode

Описание:

function GetDiscountMode: Integer;

Возвращаемые значения:

1 - режим работы со скидками/надбавками на ЧПМ установлен;

0 - режим работы со скидками/надбавками на ЧПМ не установлен.

Функция позволяет узнать, был ли запрограммирован на ККМ режим работы со скидками/надбавками. Поскольку установки читаются из ККМ при установлении с ней связи, то функция будет возвращать актуальное значение только после успешного завершения функции ConnectKKM.

Пример

```
procedure TDemoForm.GetDiscountModeButtonClick(Sender: TObject);
begin
  if GetDiscountMode = 1 then
    ShowMessage('Режим работы со скидками на ККМ установлен.')
  else
```

```
    ShowMessage('Режим работы со скидками на ККМ не установлен.')
end;
```

8. Функции назначения обработчиков событий

8.1. SetChPrepareEvent

Описание:

procedure SetChPrepareEvent(Ptr: TAppCheckPrepare);

Объявление используемого типа данных:

type TAppCheckPrepare = procedure(Progress: Integer); stdcall;

Принимаемые параметры:

Ptr - параметр процедурного типа (указатель на процедуру принимающую один параметр типа Integer);

Процедура служит для назначения обработчика события, генерируемого библиотекой при подготовке чека. Передаваемый параметр является указателем на процедуру, принимающую один параметр типа Integer, которая будет вызываться из библиотеки. Получаемый обработчиком параметр Progress содержит значение прогресса подготовки чека в процентах. Следует обратить внимание на то, что соглашение о вызове назначаемого обработчика должно быть stdcall.

Пример

```
// Объявление процедуры назначения обработчика
  procedure SetChPrepareEvent(Ptr: TAppCheckPrepare); stdcall;
external 'chon100k.dll';

     .
     .
     .
// Объявление обработчика события
procedure OnCheckPrepare(Progress: Integer); stdcall;

     .
     .
     .
// Реализация обработчика
procedure OnCheckPrepare(Progress: Integer);
begin
  if Progress = 0 then
    begin
      DemoForm.ProgressBar.Position:=0;
      DemoForm.ProgressBar.Visible:=True;
```

```
    end;
  DemoForm.ProgressBar.Position:=Progress;
  if Progress = 100 then
    begin
      Sleep(300);
      DemoForm.ProgressBar.Visible:=False;
    end;
end;
     .
     .
     .
// Назначение обработчика
  SetChPrepareEvent(OnCheckPrepare);
```

8.2. SetCloseCheckEvent

Описание:

procedure SetCloseCheckEvent(Ptr: TAppEvent);

Объявление используемого типа данных:

type TAppEvent = procedure; stdcall;

Принимаемые параметры:

Ptr - параметр процедурного типа (указатель на процедуру не принимающую параметров);

Процедура служит для назначения обработчика события, генерируемого библиотекой при успешном завершении вывода чека. Передаваемый параметр является указателем на процедуру, не принимающую параметров, которая будет вызываться из библиотеки. Следует обратить внимание на то, что соглашение о вызове назначаемого обработчика должно быть stdcall.

Пример

```
// Объявление процедуры назначения обработчика
  procedure SetCloseCheckEvent(Ptr: TAppEvent); stdcall; external
'chon100k.dll';
     .
     .
     .
// Объявление обработчика события
procedure OnCloseCheck; stdcall;
     .
     .
```

```
      .
// Реализация обработчика
procedure OnCloseCheck;
begin
  ShowMessage('Чек успешно выведен.');
end;
      .
      .
      .
// Назначение обработчика
  SetCloseCheckEvent(OnCloseCheck);
```

8.3. SetErrorEvent

Описание:

procedure SetErrorEvent(Ptr: TAppError);

Объявление используемого типа данных:

type TAppError = procedure (ErrorCode: Integer; ErrorMsg: PChar); stdcall;

Принимаемые параметры:

Ptr - параметр процедурного типа (указатель на процедуру принимающую параметр типа Integer и указатель на нуль-терминированную строку);

Процедура служит для назначения обработчика события, генерируемого библиотекой при возникновении ошибки. Передаваемый параметр является указателем на процедуру, принимающую параметр типа Integer и указатель на нуль-терминированную строку. Указанная процедура будет вызываться из библиотеки и получать код возникшей ошибки и ее текстовую интерпретацию. Следует обратить внимание на то, что соглашение о вызове назначаемого обработчика должно быть stdcall.

Пример

```
// Объявление процедуры назначения обработчика
  procedure SetErrorEvent(Ptr: TAppCheckPrepare); stdcall;
external 'chon100k.dll';
      .
      .
      .
// Объявление обработчика события
procedure OnError(ErrorCode: Integer; ErrorMsg: PChar); stdcall;
      .
      .
      .
// Реализация обработчика
```

```
procedure OnError(ErrorCode: Integer; ErrorMsg: PChar);
begin
    ShowMessage('chon100.dll сгенерировала сообщение об
ошибке.'+#13+#10+'Код ошибки = '+IntToStr(ErrorCode)+'  Текст:
'+ErrorMsg);
end;
     .
     .
     .
// Назначение обработчика
  SetErrorEvent(OnError);
```

9. Функции управления режимом

9.1. SendSales

Описание:

function SendSales: Integer;

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция передает ЧПМ информацию о чеке из базы описателей товаров и инициирует вывод чека на ЧПМ.

Пример

```
procedure TDemoForm.SendSalesButtonClick(Sender: TObject);
begin
  if SendSales = 1 then
    ShowMessage('Чек успешно выведен.')
  else
    ShowMessage(GetErrorMsg);
end;
```

10. Сервисные функции

10.1. CheckRegKass

Описание:

function CheckRegKass: Integer;

Возвращаемые значения:

1 - в случае нахождения ЧПМ в режиме "Касса";

0 - в случае нахождения ЧПМ в режиме отличном от режима "Касса".

Функция позволяет определить находится ли ЧПМ в режиме "Касса". Ошибка, возникающая при потере интерфейса не обрабатывается, т.е. в случае отсутствия интерфейса с ЧПМ функция вернет 0. Если необходимо обработать ошибку интерфейса, то следует воспользоваться любой функцией получения статистики по ЧПМ, которая данную ошибку обрабатывает или попытаться переустановить соединение с ЧПМ при помощи функции ConnectKKM.

Пример

```
procedure TDemoForm.CheckRegKassButtonClick(Sender: TObject);
begin
  if CheckRegKass = 1 then ShowMessage('ККМ в режиме "Касса"')
  else ShowMessage('ККМ не в режиме "Касса"');
end;
```

10.2. Lock

Описание:

function Lock: Integer;

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для блокировки клавиатуры ЧПМ.

Пример

```
procedure TDemoForm.LockButtonClick(Sender: TObject);
begin
  Lock;
  ShowMessage(GetErrorMsg);
end;
```

10.3. KKMPrintStr

Описание:

function KKMPrintStr(Str: PChar): Integer;

Принимаемые параметры:

Str - указатель на нуль-терминированную строку, которую необходимо напечатать.

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для печати произвольной строки длиной до 17 символов (КМ АМС-100, КМ АМС-мини 100) или 23 символов (КМ АМС-100М) на принтере ЧПМ. ЧПМ при печати любой строки, при помощи данной функции, будет автоматически предварять выводимую строку символом "#".

Пример

```
procedure TDemoForm.KKMPrintStrButtonClick(Sender: TObject);
begin
  if KKMPribtStr('Test String') <> 1 then
ShowMessage(GetErrorMsg);
end;
```

10.4. UnLock

Описание:

function UnLock: Integer;

Возвращаемые значения:

1 - в случае успешного завершения;

0 - в случае возникновения ошибки.

Функция предназначена для разблокирования клавиатуры ЧПМ, заблокированной ранее при помощи функции Lock.

Пример

```
procedure TDemoForm.UnLockButtonClick(Sender: TObject);
begin
  UnLock;
  ShowMessage(GetErrorMsg);
end;
```

10.5. GetErrorCode

Описание:

function GetErrorCode: Integer;

Возвращаемые значения:

Код ошибки.

Функция возвращает код последней возникшей ошибки. Сообщение об ошибке можно получить при помощи функции GetErrorMsg.

Пример

```
procedure TDemoForm.GetErrorCodeButtonClick(Sender: TObject);
begin
  ShowMessage(IntToStr(GetErrorCode));
end;
```

10.6. GetErrorMsg

Описание:

function GetErrorMsg: PChar;

Возвращаемые значения:

Указатель на нуль-терминированную строку, содержащую сообщение об ошибке.

Функция возвращает текстовую интерпретацию последней возникшей ошибки. Для получения кода ошибки воспользуйтесь функцией GetErrorCode.

Пример

```
procedure TDemoForm.GetErrorMsgButtonClick(Sender: TObject);
begin
  ShowMessage(GetErrorMsg);
end;
```