"MASSA-K", JSC

Driver for weighing terminals "MASSA-K: Driver R". V3.3

Programmer's Manual
13'th Edition
2020

Table of contents

Introducing

The "MASSA-K: Driver R" is a programming component (driver), developed to work with weighing terminals of R series. At current time there are 4 models of terminals:

- RL and R2L – terminals with labels printing;
- RP and R2P – terminals with labels and receipts printing;
- RA – terminals without printing;
- RC – terminals with receipts printing.

To use this driver, there is a 6.19 or later firmware version must be loaded in terminal.

1. Terminal operation's description

Weighing terminal is a device intended for registration and transmit data about goods and goods flow to accounting systems.

Some of terminal's models supplied with printers and provide to print goods labels and receipt documents of current operation.

Paired with weighing modules (platforms) terminals form a scales, providing registration and accounting of weighting goods (registration scales). Hereinafter, all equipment, which driver supports, will be named as "terminals".

With this driver there are goods, operators, stores, contractors lists and printing templates can be uploaded to terminals. An amount of uploading data determined by actual tasks.

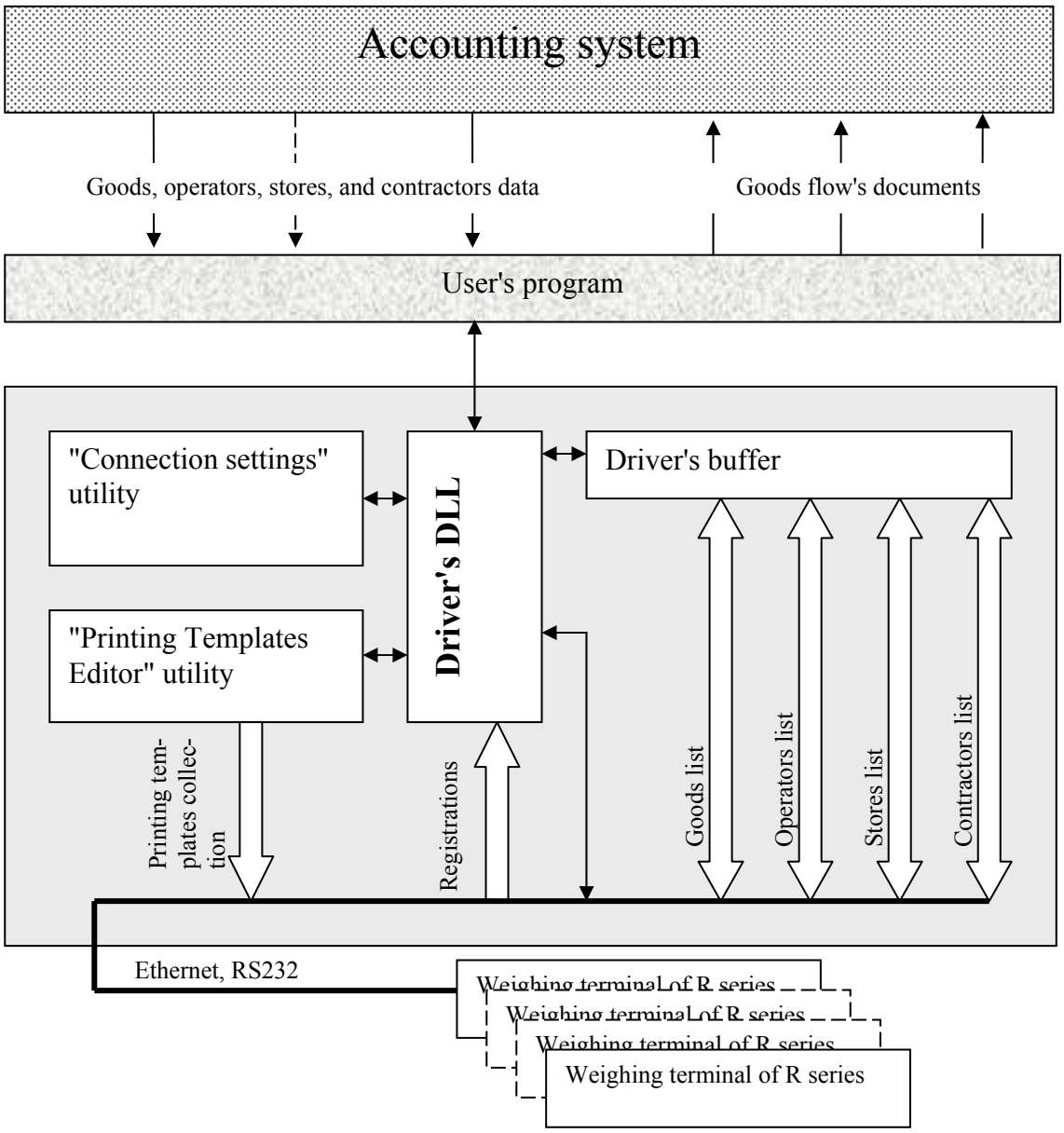
Terminals support six working modes: labeling, acceptance of goods, selling, trading, stock-taking and writing-off. All operations are registered and stored into memory during working with terminal. Terminal provides to store up to 20000 last registrations. Using driver it is possible to read all registrations for further forming a goods flow's documents.

2. Driver's working principle

Driver contains a DLL, printing templates editor and a terminal connection settings utility, as shown on picture 2.1.

2.1. Driver setting

Driver setting is connection parameters setting, and printing templates setting in case of usage terminals with printing ability. All terminals, which will work with driver, must be added to device list. Connection parameters setting is filling of driver's "Device" object properties. Adding terminals to device list and filling properties can be implemented by programming or via "Connection Settings" utility (see chapter 6). Printing templates setting is made via "Printing Templates Editor" utility (see chapter 7).



Picture 2.1. Flowchart of "MASSA-K: Driver R"

2.2. Uploading data to terminals

Goods, operators, stores and contractors lists are uploading through the driver's buffer. By properties and methods of "Goods", "Operator_", "Store" and "Contractor" objects there are a set of items forming for each list. Then, all formed sets uploads to terminals by the "Device" object methods as well as printing templates.

2.3. Downloading data from terminals

Goods, operators, stores and contractors items sets are downloading to the driver's buffer by the "Device" object methods. Then, access to any item can be obtained via properties and methods of "Goods", "Operator_", "Store", "Contractor" objects.

2.4. Downloading registrations from terminals

By the "Transaction" object methods there are properties of any stored registration or registrations group can be accessed for further forming a goods flow's documents.

3. Driver's DLL objects

Object name	Description
Driver	System object
Goods	Using for forming a user's goods parameters list
Operator_	Using for forming a operators names and pass codes list
Store	Using for forming a list of stores (departments) and keypad layouts for each store.
Contractor	Using for forming a contractors list, both buyers and shippers
Transaction	Using for forming a goods flow's documents
Device	Using for uploading and downloading data to/from terminals
LabelEditor	Using for forming printing templates
Settings	Using for operating parameters settings of terminals

4. Objects properties

4.1. Properties data types

Type	Sign	Description
Integer	I	A 32-bits signed integer value
String	S	An Unicode string value
Date	D	DateTime value

4.2. Properties of Driver system object

Property's name	Description	Type	Access	Value	Default value
Version	Driver version	S	R	Up to 32 characters	
Description	Driver description	S	R	Up to 32 characters	
ResultCode	Result code	I	R	See Table 1	
ResultDescription	Result description	S	R		
RaiseException	Generate exception	I	RW	0 – no, 1 – yes	0

ResultCode contains an result code of execution of last operation.
Result code updates in two cases:
1. When writing a value to a property.
2. When method calling.
Result code does not updates when reading a value from the property.
Possible result codes are shown in the Table 1.

Table 1. Result codes

Result code (Driver.ResultCode)	Result description (Driver.ResultDescription)
0	Operation is successful
-1	Driver is busy
-2	File not found
-3	Terminal not found
-4	Connection error
-5	Data transmitting error
-6	Data receiving error
-7	Data analyzing error
-8	Setting mode error
-9	<see note below>
-10	No more space in a driver's buffer
-11	No more memory in the terminal for loading this amount of data
-12	No more items in a list

-13	No more registrations on USB-Flash
-14	Data compression is in progress, please wait
-15	Data compression has been completed
-1000	Wrong property's value
-1001	< see note below >

Note: this error code is not supported by this driver version

ResultDescription.

This property contains an result description string of last operation.

RaiseException.

Almost all modern development tools (as Delphi, C++, VB, etc.) provide to work with exceptions. It is possible to write more stable programs by using the exception handling mechanism.

This property determines, whether or not to raise exception if error occurs (ResultCode<>0).

4.3. Device object properties

Property's name	Description	Type	Access	Value	Default value
ID	Identifier	I	R	A unique terminal's serial number. If 0 – ID is not set.	0
Connection	Connection interface	S	RW	FLASH COM1, COM2, … TCP/IP [IP-address]:[port], for example "192.168.10.2:5001", where "192.168.10.2" and "5001" are IP-address and port	
Status	Terminal status	I	R	0 – offline, 1 – online	0
LastUpdate	Last data exchange date	D	R		
Mark	Marks terminals to be upload or download	I	RW	0 – not marked, 1 – marked	
WriteType	Sets the write mode from driver's buffer to terminals	I	RW	0 – uploading data 1 – correcting data	0
Name	Terminal name	S	RW		
Weight	Weight value	I	R	Weight, which read from scales	
Stable	Weight stabilization sign	I	R	0 - weight is unstable, 1 - weight is stable	
TareWeight	Tare (package) weight value	I	RW	Tare weight, which set to scales	

Connection, Mark, Name – these properties can be implemented by programming or via "Connection Settings" utility. To call the "Connection Settings" utility use the **ShowProperties()** method**,** see chapter 5.1.

USB-flash must be marked on any terminal before using.

Status – this property sets by **Get(), GetFirst(), GetNext()** methods, see chapter 5.1.

LastUpdate – this property sets if uploading by **WriteGoods(), WriteOperator(), WriteStore()** or **WriteContractor()** methods was successful**,** see chapter 5.1.

WriteType – this property must be set before calling **WriteGoods(), WriteOperator(), WriteStore()** or **WriteContractor()** methods, see chapter 5.1. In case of uploading data (WriteType = 0), the terminal doesn't control a uniqueness of ID's compared with the previous items. All items uploaded earlier, must be previously erased by **DeleteGoods(), DeleteOpera-**

tor(), DeleteStore() or **DeleteContractor() methods** for a corresponding list**,** see chapter 5.1 . This uploading type has a maximum speed. In case of correcting data (WriteType = 1), an item uploading later replaces an item uploaded earlier with the same ID. This type is useful, for example, for goods price changing. In this case only changed goods can be written to the driver's buffer and then to be uploaded to the terminals in correction mode. There is not required to erase previously uploaded items in this uploading mode.

WriteType is taken as equal to 0 while uploading data via USB-flash.

Weight, Stable – these properties set by the **GetWeight()** method, see chapter 5.1.

TareWeight – this property must be set before calling the **SetTare()** method, see chapter 5.1.

4.4. Goods object properties

Property's name	Description	Type	Access	Value	Default value
ID	Identifier	I	RW	4 bytes. A unique number greater than 0	
Code	Goods code	S	RW	Up to 15 characters	
Name	Goods name	S	RW	Up to 250 characters	
GoodsTypeID	Goods type	I	RW	0 – weighting goods 1 – piece goods	0
Price	Goods price, in minor currency unit	I	RW	A number from 0 to 99999999	0
GroupCode	Goods group code	I	RW	A number from 0 to 65535	0
TareWeight	Tare (package) weight, in grams	I	RW	A number from 0 to 99999999	0
UnitWeight	Goods unit weight, in milligrams	I	RW	A number from 0 to 99999999	
Goods properties, that uses for printing					
NameTextAlignmentID	Goods name alignment	I	RW	0,1 (no, yes)	1
Ingredients	Goods ingredients	S	RW	Up to 1500 characters	
BestBefore	Goods best before date	D	RW		
ShelfLife	Goods shelf life, in minutes	I	RW	A number from 0 to 99999999	
AdditionalPercent	A percent of additives (for frozen goods)	I	RW	A number from 0 to 99	0
CertificationCode	Certification code	S	RW	4 characters	
Barcode	Goods barcode	S	RW	A barcode number of 8 or 13 digits. A unique value	
PLU	Goods PLU (for weighting goods only)	I	RW	A number up to 6 digits. A unique value	
BasicUnit	Measuring unit	S	RW	Up to 5 characters	
BarcodePrefix	Barcode prefix	I	RW	A number from 0 to 99	20

ID – a unique number for goods identification.

Code – goods code, a value in a text format. It is allowed to use ID as goods code, translated in a text format.

There are the same requirements for ID and Code of Operator, Store, Contractor objects.

TareWeight – tare (package) weight in grams. This property uses for automatic tare set in the scales.

UnitWeight – goods unit weight in milligrams. If this property is set, then the scales switches in a counting mode automatically when such goods calls.

Name, Ingredients – goods name and ingredients, a values in a text format:

<String1>|<String2>| ……. |<Last string>

Strings are divided by the "|" symbol. Font's type is set in a label's fields.

If required to set an own font for each string, then the "^" symbol and followed two symbols of font's number from "01" to "10" must be added at the beginning of that string.

Internal terminal's fonts.

Font size code	Font size name	Character cell size in pixels (pixel size is 0,125 x 0,125 mm)
01	S1	8 x 16
02	S2	8 x 19
03	S3	8 x 22
04	S4	12 x 20
05	S5	12 x 24
06	M1	16 x 32
07	M2	16 x 38
08	M3	16 x 44
09	M4	24 x 40
10	M5	24 x 48

AdditionalPercent – this property is used for labeling of frozen goods for pure weight counting.

BasicUnit – this property is used for printing a special field with the name of goods measuring unit, for example: box, bunch, etc.

BarcodePrefix – this property is used for printing an own goods barcode prefix, if this parameter is set in the barcode format of printing template. It is useful for dividing goods by groups.

4.5. Operator_ object properties

Property's name	Description	Type	Access	Value	Default value
ID	Identifier	I	RW	4 bytes. A unique number greater than 0	
Code	Operator's code	S	RW	Up to 9 characters. A unique value	
Name	Operator's data	S	RW	Up to 32 characters	
Passcode	Operator's passcode	I	RW	A number from 1 to 99999999	0

4.6. Store object properties

Property's name	Description	Type	Access	Value	Default value
ID	Identifier	I	RW	4 bytes. A unique number greater than 0	
Code	Store's code	S	RW	Up to 9 characters. A unique value	
Name	Store's name	S	RW	Up to 100 characters	
NumberButton1*	Goods ID for fast calling button 1	I	RW	Goods.ID	0
…….....	…….				
NumberButton8*	Goods ID for fast calling button 8	I	RW	Goods.ID	0

NumberButton1..NumberButton8 – these properties set a compliance between a fast calling buttons 1..8 and Goods IDs.

4.7. Contractor object properties

Property's name	Description	Type	Access	Value	Default value
ID	Identifier	I	RW	4 bytes. A unique number greater than 0	
Code	Contractor's code	S	RW	Up to 9 characters. A unique value	
Name	Contractor's name	S	RW	Up to 100 characters	

4.8. Transaction object properties

Property's name	Description	Type	Access	Value	Default value
DeviceID	Terminal's serial number	I	RW	A unique value	
DeviceTransaction-Number	Registration number	I	RW	Each subsequent registration has a number one more higher than the previous registration	
Status	Registration status	I	R	Bits 3..0: "0000" – paid by cash, "0001" – paid by card	
TransactionTypeID	Registration type	I	R	1 – labeling, 2 – acceptance, 3 – selling, 4 – trading, 41 – return, 5 – stock-taking, 6 – write-off, 71 – batch closure, 72 – emergency batch closure, 73 – work shift closure	
TransactionDate	Registration date/time	D	RW		
Weight	Goods net weight	I	R	In grams	
GrossWeight	Goods gross weight	I	R	In grams	
Quantity	Goods quantity	I	R		
Barcode	Barcode for weighting/piece goods or PLU for weighting goods	S	R	Up to 13 numeric characters	
GoodsID	Goods ID	I	R	Identifier from the Goods list	
Price	Goods price	I	R	Price in the minor currency unit	
Discount	Discount	I	R	Given in percents with sign: "-" – discount, "+" – margin	0
Cost	Cost	I	R	Cost in the minor currency unit	
OperatorID	Operator's ID	I	R	Identifier from the Operator list	
OperatingStoreID	Source store's ID	I	R	Identifier from the Store list	
MoveStoreID	Destination store's ID	I	R	Identifier from the Store list	
ContractorID	Contractor's ID	I	R	Identifier from the Contractor list	
ReceiptNumber	Batch (or receipt) number	I	R	This property provides to divide registrations by groups, which is using for further processing. The batch can be closed by operator (by the "Σ" key) for normal closure. If the power was switched off, the emergency closure occurs, as well as operator's or operation's change.	
DocumentCode	Document code	S	R	Up to 15 characters. Sets by operator. For each document one or more batches can be corresponded. It is a not required property.	
WorkShiftNumber	Work shift number	I	R	Uses in trading terminal's mode. Sets by administrator on a Z-report closure.	
Nickname	Nickname	S	R	Temporary unknown goods name, up to 15 characters	

4.9. Settings object properties

Property's name	Description	Type	Access	Value
Work mode settings				
Operation	Work mode	I	RW	10 – labeling, manufacturing indication. 11 – labeling, trading indication. 2 – acceptance, 3 – selling, 4 – trading, 5 – stocktaking, 6 – write-off.
Screen	Weight mode screen	I	RW	1 – NET weight 2 – GROSS weight
Registration parameters				
Goods	Goods code	S	RW	Up to 15 symbols. An unique value.
Operator	Operator's code	S	RW	Up to 9 symbols. An unique value.
RegistratorStore	Registrator's store code.	S	RW	Up to 9 symbols. An unique value.
Agent	Supplier's/recipient's code (store or contractor)	S	RW	Up to 9 symbols. An unique value.
Terminal's parameters				
DateTime	Terminal's date and time	D	RW	
GoodsButton1	Goods for button 1	S	RW	Goods code, up to 15 symbols. An unique value.
GoodsButton2	Goods for button 2	S	RW	-//-
...	...	...	...	...
GoodsButton8	Goods for button 8	S	RW	-//-
GoodsButton11	Goods for button 11	S	RW	-//-
GoodsButton12	Goods for button 12	S	RW	-//-
...	...	...	...	...
GoodsButton18	Goods for button 18	S	RW	-//-
GoodsButton21	Goods for button 21	S	RW	-//-
GoodsButton22	Goods for button 22	S	RW	-//-
...	...	...	...	...
GoodsButton28	Goods for button 28	S	RW	-//-
GoodsButton31	Goods for button 31	S	RW	-//-
GoodsButton32	Goods for button 32	S	RW	-//-
...	...	...	...	...
GoodsButton38	Goods for button 38	S	RW	-//-
GoodsButton41	Goods for button 41	S	RW	-//-
GoodsButton42	Goods for button 42	S	RW	-//-
...	...	...	...	...
GoodsButton48	Goods for button 48	S	RW	-//-
GoodsButton51	Goods for button 51	S	RW	-//-
GoodsButton52	Goods for button 52	S	RW	-//-
...	...	...	...	...
GoodsButton58	Goods for button 58	S	RW	-//-
GoodsButton61	Goods for button 61	S	RW	-//-
GoodsButton62	Goods for button 62	S	RW	-//-
...	...	...	...	...
GoodsButton68	Goods for button 68	S	RW	-//-
GoodsButton71	Goods for button 71	S	RW	-//-
GoodsButton72	Goods for button 72	S	RW	-//-
...	...	...	...	...
GoodsButton78	Goods for button 78	S	RW	-//-
GoodsButton81	Goods for button 81	S	RW	-//-
GoodsButton82	Goods for button 82	S	RW	-//-
...	...	...	...	...
GoodsButton88	Goods for button 88	S	RW	-//-
Print settings				

Tape	Paper type	I	RW	0 – separate labels, 1 – continuous tape
Underwinder	Underwinder	I	RW	0 – off 1 – on
LabelFormat	Printing template for the  button	S	RW	OFF - нет LITE – простая этикетка XXXXX – номер этикетки PRO.
LabelFormatZ	Printing template for the  button	S	RW	OFF – off CHECK – checks, reports LITE – LITE (simple) label template XXXXX – PRO label template.
Text	Text for "Terminal text" label's field	S	RW	Text is up to 24 symbols
PriceFormat	Price format	I	RW	0 - integer without comma 1 - a number of 0.0 format 2 - a number of 0.00 format 3 - a number of 0.000 format
Buttons locking				
Lock1		I	RW	0 – unlocked 1 – locked
Lock2		I	RW	0 – unlocked 1 – locked
Lock3		I	RW	0 – unlocked 1 – locked
Lock4		I	RW	0 – unlocked 1 – locked
Lock5		I	RW	0 – unlocked 1 – locked
Lock6		I	RW	0 – unlocked 1 – locked
Lock7		I	RW	0 – unlocked 1 – locked
Lock8		I	RW	0 – unlocked 1 – locked
Lock9		I	RW	0 – unlocked 1 – locked
Lock10		I	RW	0 – unlocked 1 – locked
Lock11		I	RW	0 - unlocked 1 - locked
Lock12		I	RW	0 - unlocked 1 - locked
Options				
OptionMoreLabels	"Printing multiple labels" option	I	RW	0 – option is off 1 – option is on
OptionPacking	"Packing" option	I	RW	0 – option is off 1 – option is on
OptionAutoReset-Goods	"Commodity auto reset" option	I	RW	0 – option is off 1 – option is on
OptionAutoRegistration	"Autoregistering, by scan" option	I	RW	0 – option is off 1 – option is on
OptionDocument	"Source document" option	I	RW	0 – option is off 1 – option is on
OptionMultiple	"Assembly load" option	I	RW	0 – option is off 1 – option is on
OptionFreePrice	"Free price system" option	I	RW	0 – option is off 1 – option is on
OptionTradingScales	"Trade scales" option	I	RW	0 – option is off 1 – option is on
OptionTareFrom-Base	"Tare directory" option	I	RW	0 – option is off 1 – option is on

OptionPartyWeight-Control	"Lot weight value check" option	I	RW	0 – option is off 1 – option is on
OptionScanBCLite	"Barcode scan LITE" option	I	RW	0 – option is off 1 – option is on
OptionTempName	"Temporary name" option	I	RW	0 – option is off 1 – option is on
OptionEan13Ean5	"EAN13+EAN5" option	I	RW	0 – option is off 1 – option is on
Option64GoodsKeys	"Speed dial for 64 items" option	I	RW	0 – option is off 1 – option is on
OptionInternalRe-ceipt	"BuiltIn receipt template" option	I	RW	0 – option is off 1 – option is on
OptionGoodsLabel-sLink	"Template-goods linking" option	I	RW	0 – option is off 1 – option is on
OptionGS1Databar	"EAN128 -> GS1 Databar" option	I	RW	0 – option is off 1 – option is on
Option0102Gtin-Ean13	"(01, 02) GTIN -> EAN13" option	I	RW	0 – option is off 1 – option is on
OptionComparator	"COMPARATOR" option	I	RW	0 – option is off 1 – option is on, comparation range value is in grams 2 – option is on, comparation range value is in kilograms
OptionBarcodeF8F1	"Barcode F8..F1" parameter	I	RW	0 – option is off, barcode F8..F1 parameter prints as "Work shift number" 1 – option is on, barcode F8..F1 parameter prints as "Registration number in a batch" 2 – option is on, barcode F8..F1 parameter prints as "Terminal number" 3 – option is on, barcode F8..F5 parameter prints as "Terminal number", F4..F1 parameter printis as "Registration number in a batch"
OptionLabelCopy	"Reprint" option	I	RW	0 – option is off 1 – option is on

5. Objects methods

5.1. Device object methods

Method's name	Description	Return value type	Properties	
			Uses	Modifies
Add ()	Add new device to list	I	Connection Name Mark	All properties
Get	Get properties of the device from device list	I	Connection	All properties
GetFirst	Get properties of the first device from device list	I	-	All properties
GetNext	Get properties of the next device from device list	I	-	All properties
Delete	Delete the device from device list	I	Connection	All properties
ShowProperties	Run the "Connection Settings" utility	-	-	-
OpenConnection *	Open connection with the device	I	Connection	-
CloseConnection *	Close connection with the device	I	-	-
DeleteGoods	Delete goods list from the de-	I	Connection	All properties

	vice			
DeleteOperator	Delete operators list from the device	I	Connection	All properties
DeleteStore	Delete stores list from the device	I	Connection	All properties
DeleteContractor	Delete contractors list from the device	I	Connection	All properties
WriteGoods	Upload goods from the buffer to the device	I	Connection WriteType	ID, Status, LastUpdate
WriteOperator	Upload operators from the buffer to the device	I	Connection WriteType	ID, Status, LastUpdate
WriteStore	Upload stores from the buffer to the device	I	Connection WriteType	ID, Status, LastUpdate
WriteContractor	Upload contractors from the buffer to the device	I	Connection WriteType	ID, Status, LastUpdate
WriteLabelEditor	Upload printing templates to the device	I	Connection	ID, Status, LastUpdate
ReadGoods	Download goods from the device to the buffer	I	Connection	ID, Status, LastUpdate
ReadOperator	Download operators from the device to the buffer	I	Connection	ID, Status, LastUpdate
ReadStore	Download stores from the device to the buffer	I	Connection	ID, Status, LastUpdate
ReadContractor	Download contractors from the device to the buffer	I	Connection	ID, Status, LastUpdate
GetWeight	Get current weight from the scales	I	Connection	Weight, Stable
SetTare	Set tare weight to the scales	I	Connection TareWeight	-
GetTare	Get tare value from the scales	I	Connection	TareWeight
DeleteGoodsByID	Delete goods with such ID from goods list	I	Connection Goods.ID	-
DeleteGoodsByCode	Delete goods with such code from goods list	I	Connection Goods.Code	-
DeletGoodsByPLU	Delete goods with such PLU from goods list	I	Connection Goods.PLU	-
DeleteGoodsByBarcode	Delete goods with such barcode from goods list		Connection Goods.Barcode	-
CompressGoods	Compress goods list	I	Connection	-

* Optional methods, are useful for periodically fast weight request, for example, because of short execution time. Without these methods the request execution time is slightly longer, but there is no need to open/close connection manually.

All methods applies to terminals from the list only.

5.2. Common Goods, Operator_, Store, Contractor objects methods

Method's name	Description	Return value type	Properties	
			Uses	Modifies
Add	Add an item to object buffer	I	-	All properties
Clear	Clear object buffer	I	-	All properties
GetFirst	Get properties of the first item from list	I	-	All properties
GetNext	Get properties of the next item from list	I	-	All properties

5.3. Transaction object methods

Method's name	Description	Return	Properties

		value type	Uses	Modifies
Read	Download registration from the terminal	I	DeviceID DeviceTransactionNumber	All properties
ReadLast	Download last registration from the terminal	I	DeviceID	All properties
ReadNextDate	Download next from date registration from terminal	I	DeviceID TransactionDate	All properties
ReadFromTransac-tionNumber	Download registrations from specified number to end from terminal	I	DeviceID DeviceTransactionNumber	All properties
ReadFirstFromFlash	Load first registrations file from USB-Flash to object buffer	I	-	All properties
ReadNextFromFlash	Load next registrations file from USB-Flash to object buffer	I	-	All properties
GetFirst	Get properties of the first registration from buffer	I	-	All properties
GetNext	Get properties of the next registration from buffer	I	-	All properties

USB-flash must be marked on any terminal before using.

5.4. LabelEditor object methods

Method's name	Description	Return value type	Properties	
			Uses	Modifies
ShowSimpleProperties	Run the "Printing Templates Editor" in a simple mode	-	-	-
ShowFullProperties	Run the "Printing Templates Editor" in a full-featured mode	-	-	-

5.5. Settings object methods

Method's name	Description	Return value type	Properties	
			Uses	Modi-fies
SetGoods	Sets goods from goods list	I	Connection, Goods	-
SetLabelFormatW	Sets the label template to the button for weighting goods	I	Connection LabelFormat	-
SetLabelFormatP	Sets the label template to the button for piece goods	I	Connection LabelFormat	-
SetLabelFormatC	Sets the label template to the button for counting goods	I	Connection LabelFormat	-
SetLabelFormatZW	Sets the label template to the Σ for weighting goods	I	Connection LabelFormatZ	-
SetLabelFormatZP	Sets the label template to the Σ for piece goods	I	Connection LabelFormatZ	-
SetLabelFormatZC	Sets the label template to the Σ for counting goods	I	Connection LabelFormatZ	-

SetText	Sets the terminal text	I	Connection Text	-
SetUnderwinder	Set the underwinder mode	I	Connection Underwinder	-
SetMode	Sets the work mode	I	Connection Operation RegistratorStore Agent	-
SetOperator	Sets the operator	I	Connection Operator	-
SetTape	Sets the tape type	I	Connection Tape	-
SetGoodsButton	Set goods buttons	I	Connection, GoodsButton1.. GoodsButton8, GoodsButton11.. GoodsButton18, GoodsButton21.. GoodsButton28, GoodsButton31.. GoodsButton38, GoodsButton41.. GoodsButton48, GoodsButton51.. GoodsButton58, GoodsButton61.. GoodsButton68, GoodsButton71.. GoodsButton78, GoodsButton81.. GoodsButton88	-
SetLock	Set buttons' locks	I	Connection Lock1, Lock2, Lock3, Lock4, Lock5, Lock6, Lock7, Lock8, Lock9, Lock10, Lock11, Lock12	-
SetOption	Set options	I	Connection, OptionMoreLabels, OptionPacking, OptionAutoResetGoods, OptionAutoRegistration, OptionDocument, OptionMultiple, OptionFreePrice, OptionTradingScales, OptionTareFromBase, OptionPartyWeightControl, OptionScanBCLite, OptionTempName, OptionEan13Ean5, Option64GoodsKeys, OptionInternalReceipt, OptionGoodsLabelsLink, OptionGS1Databar, Option0102GtinEan13, OptionComparator, OptionBarcodeF8F1, OptionLabelCopy	-
SetPriceFormat	Sets the price format	I	Connection, PriceFormat	-
SetScreen	Sets the weight mode screen	I	Connection, Screen	-
SetDateTime	Sets the date and time to terminal	I	Connection, DateTime	-

GetDateTime	Reads the date and time from terminal	I	Connection	DateTime
SetDefaults	Set the default settings	I	Connection	-

6. "Connection settings" utility

This utility developed for a visual settings of the Device object properties. To run utility see chapter 5.1.

7. "Printing Templates Editor" utility

This utility developed for a visual forming printing templates list, including:

- "LITE" label template,
- "PROFESSIONAL" labels templates,
- receipt template.

Table 7-1 "PROFESSIONAL" label template's fields

№	Field name	Field's brief description
		Text fields (contains text and numeric values)
Direct text field		
	Direct texts	Texts that do not depend on parameters of the goods and terminal
Goods parameters fields		
	Goods code	Text up to 15 characters
	PLU number	Numeric code of weighting goods, up to 6 digits
	Goods name	Text up to 250 characters
	Goods ingredients	Text up to 1500 characters
	Goods price	Numeric value, per kg, or per pcs.
	Best before date	Text in DD/MM/YY format
	Best before time	Text in HH:MM:SS format
	Goods group code	A number from 0 to 65535
	Barcode prefix	A number from 00 to 99, set in goods parameters
	Measuring unit	Text up to 5 characters
	Net weight	Goods net weight, a number up to 6 digits + decimal separator
	Tare weight	Tare weight, a number up to 6 digits + decimal separator
	Pure weight	Goods net weight without additives (for example, frozen fish weight without surface ice), a number up to 6 digits + decimal separator
	Gross weight	Goods gross weight, a number up to 6 digits + decimal separator
	Unit weight, in grams	Unit weight, in grams, a number up to 6 digits + decimal separator. Uses for piece goods only. A quantity of such goods determines by weighing (counting mode).
	Quantity	A number up to 6 digits
	Quantity of weighings (registrations) in a batch	A number up to 6 digits
	Cost	Numeric value. A number up to 10 digits + decimal separator
	Packing date	Text in DD/MM/YY format
	Packing time	Text in HH:MM:SS format
Goods group parameters fields		
	Reference to the document	Text up to 15 characters
	Work shift number	A number up to 5 digits. Sets by administrator on Z-report closure
	Batch number	A number up to 8 digits. Divides registrations on groups.
Terminal parameters fields		
	Terminal serial number	A number up to 8 digits. Set by manufacturer
	Terminal number	A number up to 8 digits. Sets by operator in the terminal if needed
	Terminal text	Text up to 24 characters. Sets by operator on the terminal if needed
Registrations parameters field		
	Registration number	A number up to 8 digits. A unique registration's number. Increases by 1 after each registration
Stores and contractors parameters fields		
	Source store's name	Text up to 32 characters
	Source store's code	Text value up to 9 characters
	Destination store's name	Text up to 32 characters
	Destination store's code	Text value up to 9 characters

Contractor's name	Text up to 32 characters
Contractor's code	Text value up to 9 characters
Operators parameters fields	
Operator's name	Text up to 32 characters, for example: name, surname
Operator's code	Text value up to 9 characters
Pictures fields	
Picture	Monochrome picture
Certification sign	Certification sign's image with certification code
Standard icon	Standard monochrome icons
Line graphical fields	
Frame	Rectangles of different thickness
Line	Lines of different thickness
Barcodes fields	
EAN13 barcode	13-digit barcode of fixed length for numeric data encoding
EAN13 + EAN5 barcode	EAN5 template configured in terminal
Interleaved 2 of 5 barcode	Barcode of various length for numeric data encoding
ITF-14 barcode	13-digit barcode of fixed length, applied to the transport packaging
Code 39 barcode	Barcode of various length for alphanumeric data encoding
Code 128 barcode	Barcode of various length for alphanumeric data encoding
EAN128 barcode	Barcode for data exchange between different companies
GS1 Databar	Multiline barcode for data exchange between different companies
Datamatrix	Compact 2D-barcode of various size for alphanumeric data encoding
GS1 Datamatrix	Compact 2D-barcode of various size for data exchange between different companies

– highlighted templates and fields are supporting in a simple editor work mode.
All templates and fields are supported in a full-featured mode.

8. Driver initialization

8.1. Microsoft VBA

For the Microsoft Excel 2003:
1. Create new Excel workbook.
2. Select the "Service\Macros\Macroses\" menu item.
3. In a window appeared input the "Macros name" and press the "Create" button. A Visual Basic editor window with created procedure template will be opened.
4. Select the "Tools\References" menu item.
5. In a list appeared select the "TerminalMassaK Library" string and press "Ок".
6. After that, driver is available for programming. For example, objects creations are shown below:

```
Sub DriverConnection()
  Dim Driver As TerminalMassaK.Driver
  Dim Device As TerminalMassaK.Device
  Dim Goods As TerminalMassaK.Goods
  Dim LabelEditor As TerminalMassaK.LabelEditor

  Set Driver = New TerminalMassaK.Driver
  Set Device = New TerminalMassaK.Device
  Set Goods = New TerminalMassaK.Goods
  Set LabelEditor = New TerminalMassaK.LabelEditor
End Sub
```

8.2. Borland Delphi

For Borland Delphi 7:
1. Select the "File\New\Application" menu item. A new project with main unit "Unit1" will be created.
2. Select the "Project\Import Type Library" menu item.
3. In a list appeared select the "TerminalMassaK Library (Version X.X)" string and press the "Create Unit" button. Type library unit will be created.
4. In a "uses" section of "Unit1" specifies the type library unit name: "TerminalMassaK_TLB".
5. After that, driver is available for programming. For example, objects creations are shown below:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, TerminalMassaK_TLB;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Driver: TDriver;
  Device: TDevice;
  Goods: TGoods;
  LabelEditor: TLabelEditor;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Driver:=TDriver.Create(self);
  Device:=TDevice.Create(self);
  Goods:=TGoods.Create(self);
  LabelEditor:=TLabelEditor.Create(self);
end;

end.
```

8.3. Embarcadero RAD Studio C++ Builder

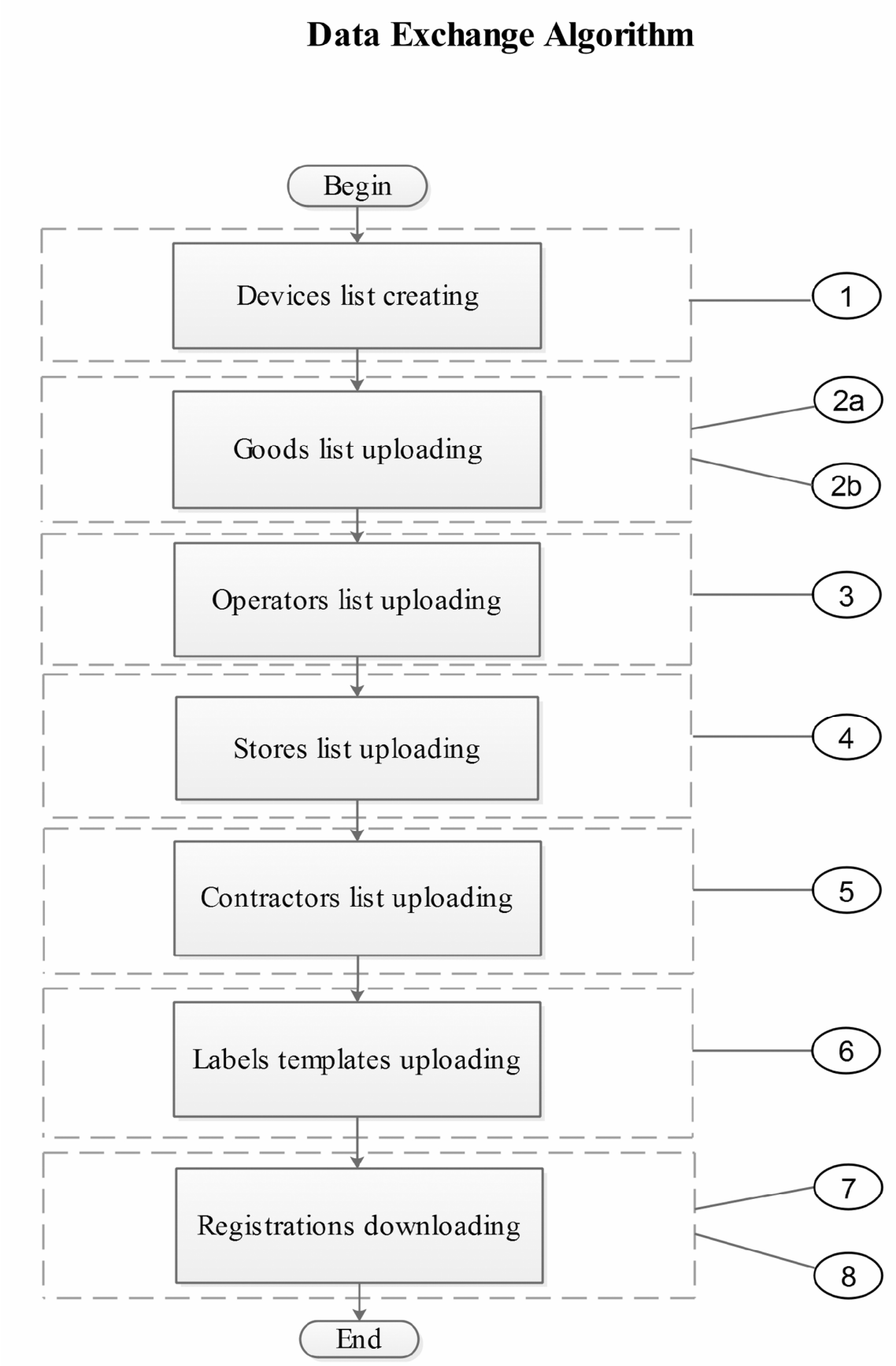
1. Select the "File\New\VCL Forms Application – C++ Builder" menu item. A new project with main unit "Unit1.cpp" will be created.
2. Select the "Component\Import Component" menu item.
3. In a window appeared select the "Import a Type Library" item and press the "Next" button.
4. In a list appeared select the "TerminalMassaK" string and press the "Next" button.
5. In a window appeared press the "Next" button.
6. In a window appeared select the "Add unit to <project name> project" item and press the "Finish" button.
7. In a "Unit1.cpp" include header file of type library:

```
#include "TerminalMassaK_TLB.h"
```

8. After that, driver is available for programming. See example below:

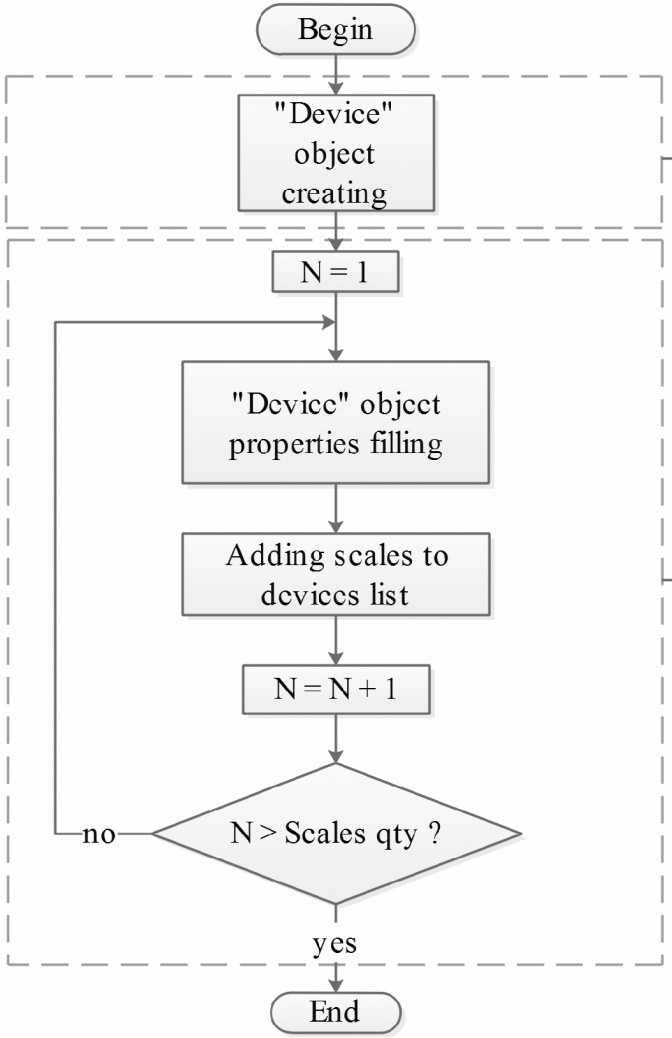
```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "TerminalMassaK_TLB.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{

    CoInitializeEx(NULL, COINIT_APARTMENTTHREADED);

    IDevice *pDeviceDisp = NULL;
    HRESULT hr = CoCreateInstance(CLSID_Device, NULL, CLSCTX_INPROC, IID_IDevice, (void**)&pDeviceDisp);

    IDevice *pDevice     = NULL;
    hr = pDeviceDisp->QueryInterface(IID_IDevice, (void **)&pDevice);
    if(SUCCEEDED(hr))
    {

        hr = pDevice->ShowProperties();

        pDevice->Release();
        pDeviceDisp->Release();
    }
    CoUninitialize();
}
//---------------------------------------------------------------------------
```

Appendix. Driver R operating algorithms.



1

Devices list creating ("Device" object)

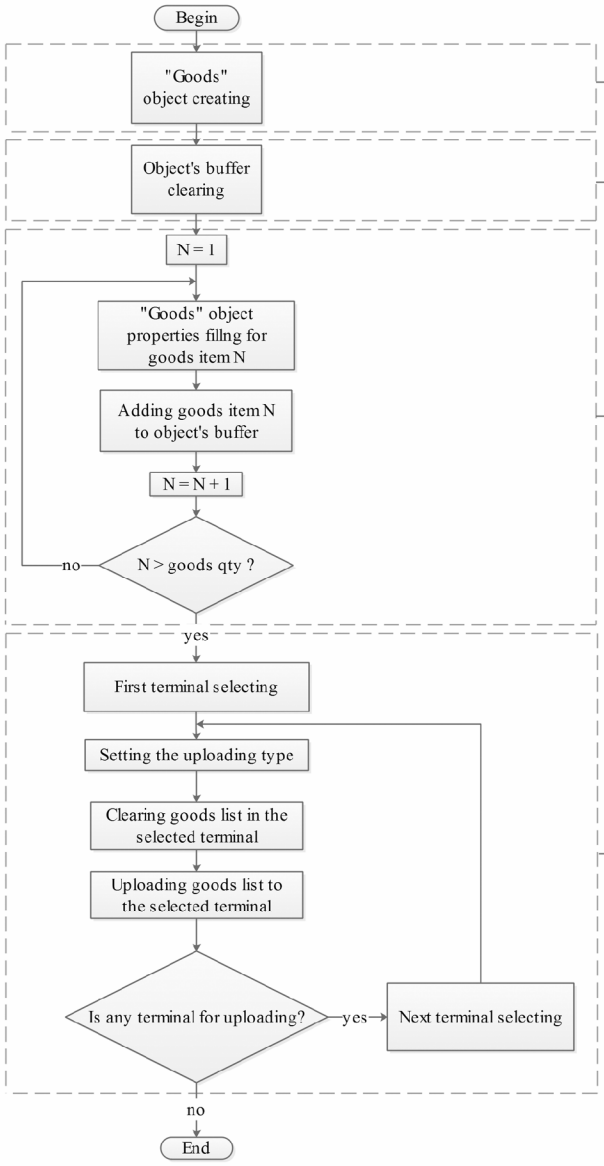



```
if Driver=nil then Driver:=TDriver.Create(nil);
if Device=nil then Device:=TDevice.Create(nil);
```

```
for N:=1 to M do begin
  Device.Connection:=IpAddress+':'+IpPort;
  Device.Name:='Device '+IntToStr(N);
  Device.Mark:=1;
  res:=Device.Add;
  if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
end;
```

2a

Full uploading of goods list ("Goods" object)



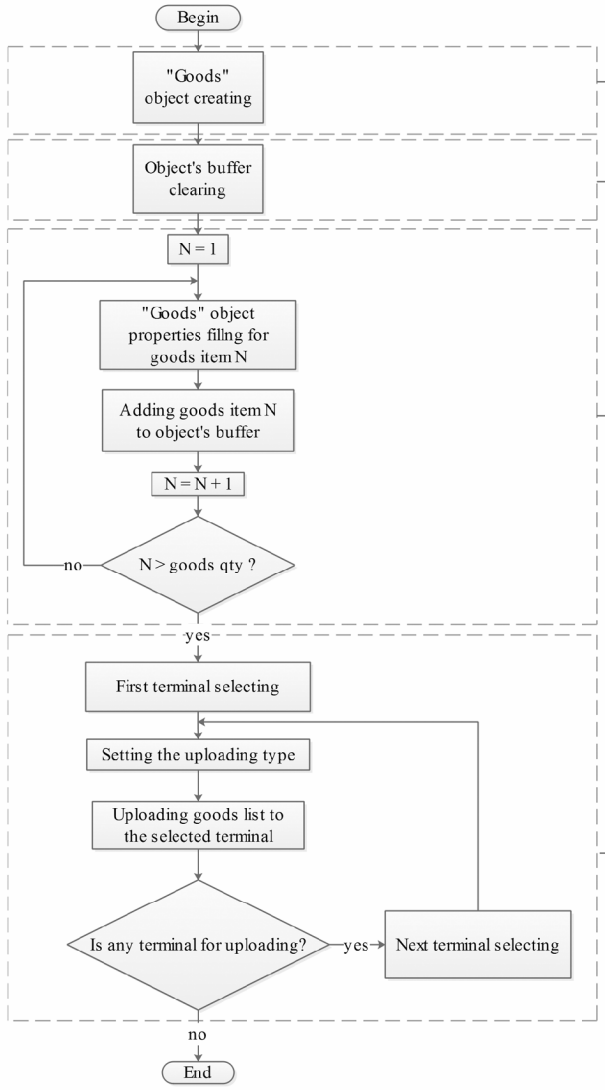

```
if Driver=nil then Driver:=TDriver.Create(nil);
if Goods=nil then Goods:=TGoods.Create(nil);
```

```
Goods.Clear;
```

```
for N:=1 to M do begin
 Goods.ID:=N;
 Goods.GoodsTypeID:=0;
 Goods.Code:=IntToStr(N);
 Goods.Name:='Item '+IntToStr(N);
 ...
 res:=Goods.Add;
 if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
end;
```

```
if Device=nil then Device:=TDevice.Create(nil);
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
  Device.Get;
  if (Device.Status=1) then begin
   Device.WriteType:=0;
   Device.DeleteGoods;
   res:=Device.WriteGoods;
   if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
  end;
 end;
 i:=Device.GetNext;
end;
```

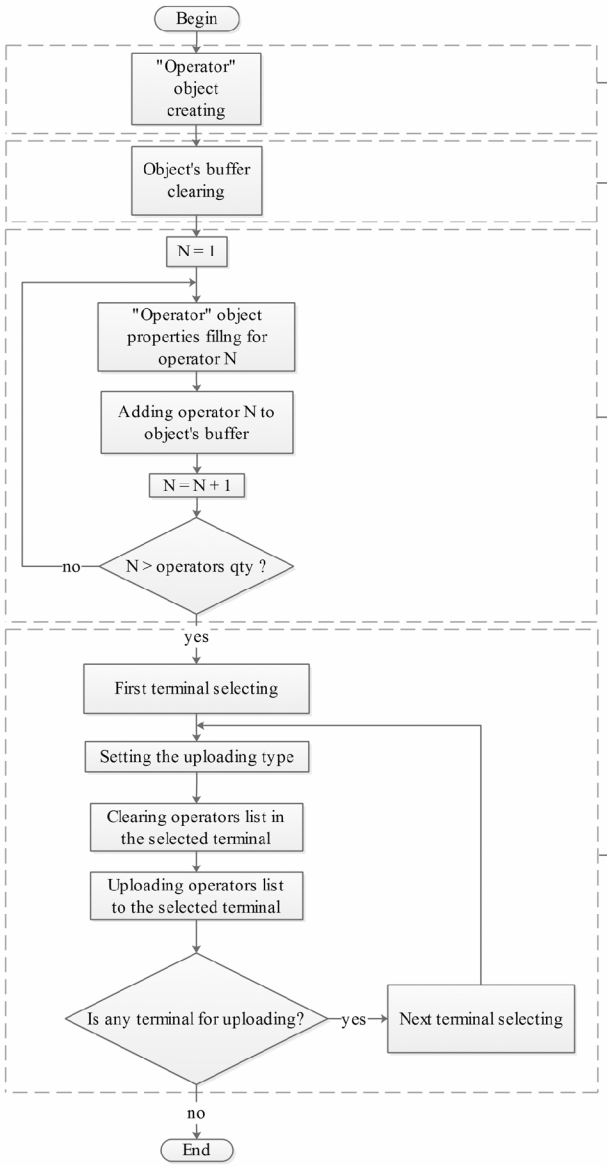
2b **Partly uploading of goods list ("Goods" object)**




```
if Driver=nil then Driver:=TDriver.Create(nil);
if Goods=nil then Goods:=TGoods.Create(nil);

Goods.Clear;

for N:=1 to M do begin
 Goods.ID:=N;
 Goods.GoodsTypeID:=0;
 Goods.Code:=IntToStr(N);
 Goods.Name:='Item'+IntToStr(N);
 ...
 res:=Goods.Add;
 if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
end;

if Device=nil then Device:=TDevice.Create(nil);
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
  Device.Get;
  if (Device.Status=1) then begin
   Device.WriteType:=1;
   res:=Device.WriteGoods;
   if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
  end;
 end;
 i:=Device.GetNext;
end;
```

3

Uploading of operators list ("Operator" object)



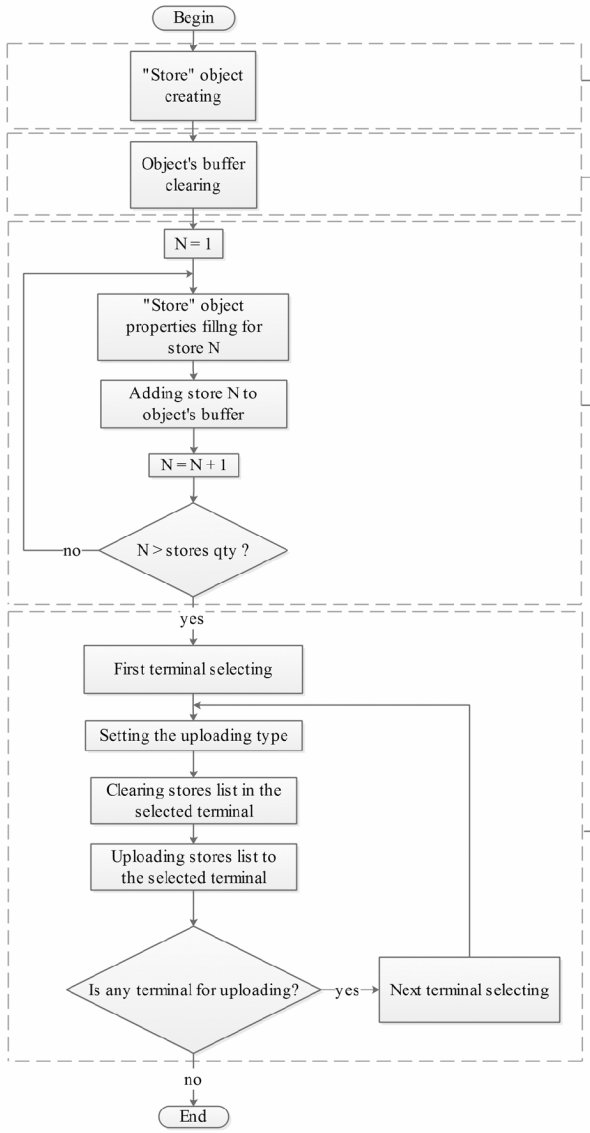

```
if Driver=nil then Driver:=TDriver.Create(nil);
if Operator=nil then Operator:=TOperator.Create(nil);
```

```
Operator.Clear;
```

```
for N:=1 to M do begin
  Operator.ID:=N;
  Operator.Passcode:=N;
  Operator.Code:=IntToStr(N);
  Operator.Name:='Operator '+IntToStr(N);
  res:=Operator.Add;
  if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
end;
```

```
if Device=nil then Device:=TDevice.Create(nil);
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
  Device.Get;
  if (Device.Status=1) then begin
    Device.WriteType:=0;
    Device.DeleteOperator;
    res:=Device.WriteOperator;
    if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
  end;
 end;
 i:=Device.GetNext;
end;
```

4 **Uploading of stores list ("Store" object)**



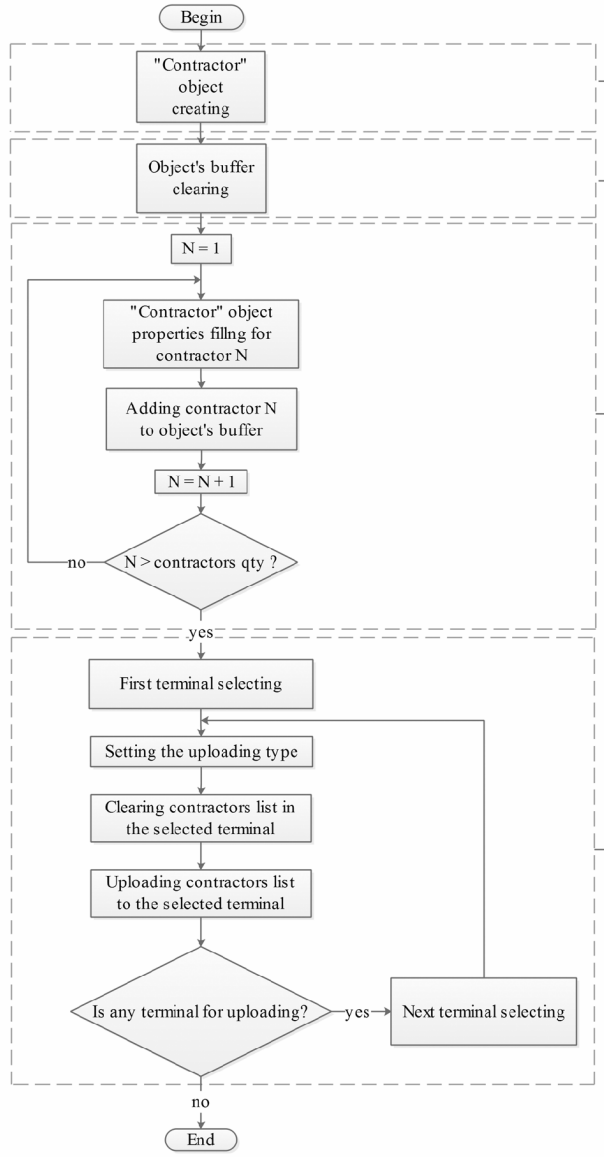

```
if Driver=nil then Driver:=TDriver.Create(nil);
if Store=nil then Store:=TStore.Create(nil);
```

```
Store.Clear;
```

```
for N:=1 to M do begin
 Store.ID:=N;
 Store.Code:=IntToStr(N);
 Store.Name:='Store '+IntToStr(N);
 res:=Store.Add;
 if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
end;
```

```
if Device=nil then Device:=TDevice.Create(nil);
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
  Device.Get;
  if (Device.Status=1) then begin
   Device.WriteType:=0;
   Device.DeleteStore;
   res:=Device.WriteStore;
   if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
  end;
 end;
 i:=Device.GetNext;
end;
```

5 **Uploading of contractors list ("Contractor" object)**




```
if Driver=nil then Driver:=TDriver.Create(nil);
if Contractor=nil then Contractor:=TContractor.Create(nil);
```

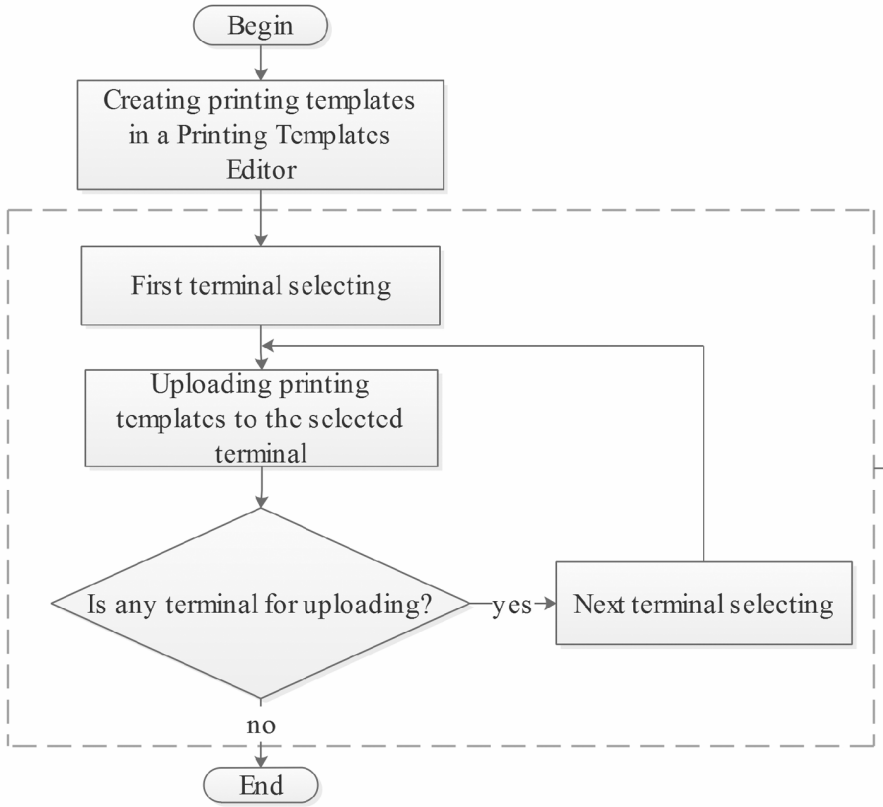
```
Contractor.Clear;
```

```
for N:=1 to M do begin
 Contractor.ID:=N;
 Contractor.Code:=IntToStr(N);
 Contractor.Name:='Contractor '+IntToStr(N);
 res:=Contractor.Add;
 if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
end;
```

```
if Device=nil then Device:=TDevice.Create(nil);
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
  Device.Get;
  if (Device.Status=1) then begin
   Device.WriteType:=0;
   Device.DeleteContractor;
   res:=Device.WriteContractor;
   if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
  end;
 end;
 i:=Device.GetNext;
end;
```

6

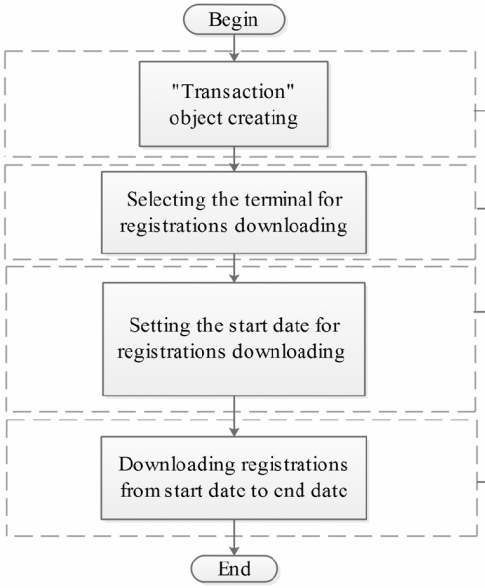
Uploading of printing templates




```
if Driver=nil then Driver:=TDriver.Create(nil);
if Device=nil then Device:=TDevice.Create(nil);
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
  Device.Get;
  if (Device.Status=1) then begin
   res:=Device.WriteLabelEditor;
   if res<>0 then WriteLn('Ошибка: '+Driver.ResultDescription);
  end;
 end;
 i:=Device.GetNext;
end;
```

7

Downloading registrations for the dates period ("Transaction" object)




```
if Device=nil then Device:=TDevice.Create(nil);
if Transaction=nil then Transaction:=TTransaction.Create(nil);
```

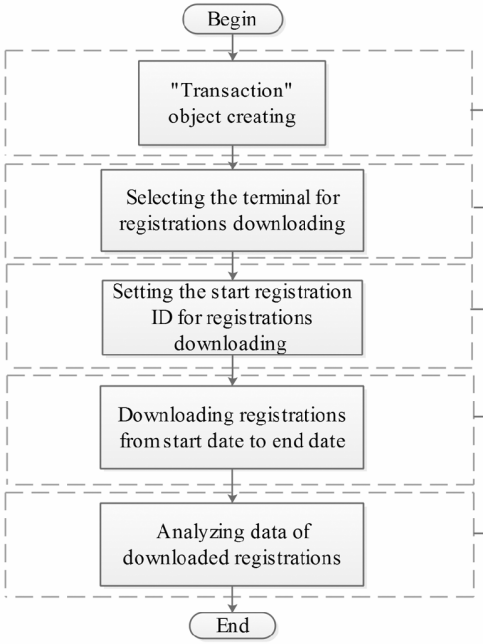
```
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
   i:=Device.Get;
   if Device.Status=1 then begin
     if Device.ID=DeviceID then break;
   end;
 end;
 i:=Device.GetNext;
end;
Transaction.DeviceID:=Device.ID;
```

```
Transaction.DateTime:=StartDateTime;
```

```
res:=Transaction.ReadNextDate;
if res=0 then begin
  res:=Transaction.ReadFromTransactionNumber;
  if res=0 then begin
    i:=Transaction.GetFirst;
    while i=0 do begin
      if Transaction.DateTime>EndDateTime then break;
     // <код обработки параметров текущей регистрации>
      i:=Transaction.GetNext;
    end;
  end;
end;
```

8

Downloading registrations from specified ID ("Transaction" object)




```
if Device=nil then Device:=TDevice.Create(nil);
if Transaction=nil then Transaction:=TTransaction.Create(nil);
```

```
i:=Device.GetFirst;
while i=0 do begin
 if Device.Mark=1 then begin
   i:=Device.Get;
   if Device.Status=1 then begin
    if Device.ID=DeviceID then break;
   end;
 end;
 i:=Device.GetNext;
end;
Transaction.DeviceID:=Device.ID;
```

```
Transaction.DeviceTransactionNumber:=StartTransactionNumber;
```

```
res:=Transaction.ReadFromTransactionNumber;
```

```
if res=0 then begin
  i:=Transaction.GetFirst;
  while i=0 do begin
    // <код обработки параметров текущей регистрации>
    i:=Transaction.GetNext;
  end;
end;
```