FP3530

(ver.3.0.0.7 – beta version)

Ръководство да употреба

История на продукта

Версия 3.0.0.6

Version	Date	Description
3.0.0.6	02/2019	**Бета версия**! ✔ Нови полета (**properties**): ✔ FR_Invoice – тип: фискална бележка или фактура ✔ FR_UNP – УНП на фискалната бележка/фактура ✔ can_OpenFiscalReceipt – дали може да се отвори фискална бележка ✔ can_OpenInvoiceReceipt – дали може да се отвори фактура ✔ Нови **device-independent** методи: ✔ init_FiscalReceiptValues ✔ open_FiscalReceipt (команда 48) ✔ open_InvoiceReceipt (команда 48 - разширена фискална бележка от тип фактура) ✔ Нови методи във връзка с въвеждането на device-independent метод за команда 48: ✔ логическа група А: ✔ 048_receipt_FiscalOpen_A01: ✔ Отваряне на фискална бележка БЕЗ наличие на поле UNP. ✔ Еквивалентна на 048_receipt_Fiscal_01_Open. ✔ 048_receipt_FiscalOpen_A02: ✔ Отваряне на фактура БЕЗ наличие на поле UNP ✔ Еквивалентна на 048_receipt_Invoice_01_Open ✔ 048_receipt_FiscalOpen_A03 ✔ Отваряне на фискална бележка с наличие на поле UNP ✔ Еквивалентна на 048_receipt_Fiscal_02_Open ✔ 048_receipt_FiscalOpen_A04: ✔ Отваряне на фактура с наличие на поле UNP ✔ Еквивалентна на 048_receipt_Invoice_Open ✔ логическа група Б: ✔ 048_receipt_FiscalOpen_B01: Отваряне на фискална бележка БЕЗ наличие на поле UNP ✔ 048_receipt_FiscalOpen_B02: Отваряне на фактура БЕЗ наличие на поле UNP ✔ 048_receipt_FiscalOpen_B03: ✔ Отваряне на фискална бележка с наличие на поле UNP. ✔ Еквивалентна на 048_receipt_Fiscal_Open ✔ 048_receipt_FiscalOpen_B04: ✔ Отваряне на фактура с наличие на поле UNP ✔ Еквивалентна на 048_receipt_Invoice_Open ✔ bug fix: Проблем в автоматичното разпознаване на модели "SK1-21F" и "SK1-32F"; ✔ bug fix: Отстранен проблем при работа по TCP/IP с устройства в логическа група „А“ ✔ bug fix: Дублиране на имена на функции по HUMAN_NAME_03: ✔ **Промяна на имена в логическа група А**: ✔ 065_info_Get_AdditionalDailyInfo_01: 065_info_Get_AdditionalDailyInfo_01 ✔ 110_info_Get_AdditionalDailyInfo_01: 110_info_Get_AdditionalDailyInfo_02 ✔ 110_info_Get_AdditionalDailyInfo_02: 110_info_Get_AdditionalDailyInfo_03 ✔ 079_info_Print_ShortFMByDTRange: 079_report_FMByDateRange_Short ✔ 111_report_ItemsWithGroup_InRange: 111_report_Items_InRangeByGroup ✔ 080_run_SoundSignal: 080_other_Sound_Signal

- ✔ **Промяна на имена в логическа група Б**:
  - ✔ 065_info_Get_AdditionalDailyInfo_01: 065_info_Get_AdditionalDailyInfo_04
  - ✔ 065_info_Get_AdditionalDailyInfo_02: 065_info_Get_AdditionalDailyInfo_05
  - ✔ 065_info_Get_AdditionalDailyInfo_03: 065_info_Get_AdditionalDailyInfo_06
  - ✔ 065_info_Get_AdditionalDailyInfo_04: 065_info_Get_AdditionalDailyInfo_07
  - ✔ 065_info_Get_AdditionalDailyInfo_05: 065_info_Get_AdditionalDailyInfo_08
  - ✔ 065_info_Get_AdditionalDailyInfo_06: 065_info_Get_AdditionalDailyInfo_09
  - ✔ 065_info_Get_AdditionalDailyInfo_07: 065_info_Get_AdditionalDailyInfo_10
  - ✔ 110_info_Get_AdditionalDailyInfo_03: 110_info_Get_AdditionalDailyInfo_11
  - ✔ 080_other_write_SoundSignal: 080_other_Sound_Signal
- ✔ bug fix:
  - ✔ Команда 110 в логическа група Б няма параметър "ErrorCode"
  - ✔ Команда 112 в логическа група Б няма параметър "ErrorCode"
- ✔ bug fix: Неизлизане от употреба на делимитери, характерни само за дадени, специфични команди;
- ✔ bug fix: Добавени параметри в отговора на команда 70 'CashIn_CashOut'
  - ✔ CashSum, ServIn, ServOut
- ✔ Нови команди в логическа група А:
  - ✔ 034_report_Service_Contracts
  - ✔ 034_info_Service_Contracts
  - ✔ 072_service_Fiscalization
  - ✔ 083_service_Set_DecimalsAndTaxRates
  - ✔ 089_service_Set_ProductionTestArea
  - ✔ 091_service_Set_SerialNumber
  - ✔ 098_service_Set_EIK
  - ✔ 128_service_RAM_Reset
  - ✔ 133_service_Disable_Print
  - ✔ 134_service_Format_KLEN
  - ✔ 135_service_Test_GPRS
  - ✔ 119_klen_Get_Info
  - ✔ 119_klen_Sinfo_W_ByNumber
  - ✔ 119_klen_SInfo_W_ByNumbersRange
  - ✔ 119_klen_SInfo_W_ByDateRange
  - ✔ 119_klen_SInfo_W_GetNext
  - ✔ 119_klen_SInfo_Y_ByNumber
  - ✔ 119_klen_SInfo_Y_ByNumbersRange
  - ✔ 119_klen_SInfo_Y_ByDateRange
  - ✔ 119_klen_SInfo_Y_GetNext
  - ✔ 119_klen_SInfo_V_ByNumber
  - ✔ 119_klen_SInfo_V_ByNumbersRange
  - ✔ 119_klen_SInfo_V_ByDateRange
  - ✔ 119_klen_SInfo_V_GetNext
  - ✔ 119_klen_SInfo_ExV_ByNumber
  - ✔ 119_klen_SInfo_ExV_ByNumbersRange
  - ✔ 119_klen_SInfo_ExV_ByDateRange
  - ✔ 119_klen_SInfo_ExV_GetNext
- ✔ Нови команда в логическа група Б:
  - ✔ 072_service_Fiscalization
  - ✔ 079_report_FMByDateRange_Short
  - ✔ 083_service_Set_DecimalsAndTaxRates
  - ✔ 089_service_Set_ProductionTestArea
  - ✔ 098_service_Set_EIK
  - ✔ 140_clients_Set_ClientData
  - ✔ 140_clients_Del_ClientData
  - ✔ 140_clients_Get_ClientData
  - ✔ 140_clients_Set_SellerName
  - ✔ 140_clients_Get_FirstClientData
  - ✔ 140_clients_Get_NextClientData
  - ✔ 140_clients_Del_AllClientData
  - ✔ 124_klen_FindRange_ByDateTime
  - ✔ 124_klen_FindRange_ByZReports
  - ✔ 124_klen_Get_Info
  - ✔ 125_klen_Prepare_Document
  - ✔ 125_klen_Prepare_DocumentInRange
  - ✔ 125_klen_Get_TextRow
  - ✔ 125_klen_Print_Document
  - ✔ 125_klen_Get_SInfo
- ✔ Корекции в демо програмите:
  - ✔ фикс на някои малки грешки;
  - ✔ редизайн и добавяне на демо страница за **device-independent** методи

Версия 3.0.0.5

Version	Date	Description
3.0.0.5	02/2019	**Бета версия**! ✔ поддръжка на модел DP-15 ✔ new property: Operator_Code ✔ new property: Operator_Code ✔ new property: Operator_Password ✔ new property: Till_Number ✔ new property: St_Reason_Type ✔ new property: St_Doc_Number ✔ new property: St_Doc_DateTime ✔ new property: St_FM_Number ✔ new property: St_Doc_UNP ✔ new property: St_ByInvoice ✔ new property: St_InvoiceNumber ✔ new property: St_Current_UNP ✔ new property: can_OpenStornoReceipt ✔ new property: AllReceipt_Count ✔ new property: StReceipt_Count ✔ new property: FiscalReceipt_Count ✔ new method: init_StornoValues ✔ new method: init_OperatorValues ✔ за устройства от логическа група Б към които е свързан платежен терминал (производство на Датекс) се добавят следните команди: ✔ 053_receipt_PinPadTotal_Amount ✔ 053_receipt_PinPadTotal_AmountTextRow1 ✔ 053_receipt_PinPadTotal_AmountTextRow2 ✔ Забележка: Да се използват единствено при тип на плащане "С" – Плащане с дебитна карта ✔ bug fix: Имената на различните варианти на команда 46 - дублаж, липси и грешки; ✔ Имената се променят съгласно документацията за device-independent метода: ✔ логическа група А: ✔ 046_receipt_Storno_06_Open: 046_receipt_StornoOpen_A01 ✔ 046_receipt_Storno_03_Open: 046_receipt_StornoOpen_A02 ✔ нов метод: 046_receipt_StornoOpen_A03 ✔ 046_receipt_Storno_02_Open: 046_receipt_StornoOpen_A04 ✔ 046_receipt_Storno_05_Open: 046_receipt_StornoOpen_A05 ✔ 046_receipt_Storno_01_Open: 046_receipt_StornoOpen_A06 ✔ 046_receipt_Storno_04_Open: 046_receipt_StornoOpen_A07 ✔ 046_receipt_Storno_Open: 046_receipt_StornoOpen_A08 ✔ 046_receipt_InvoiceStorno_04_Open: 046_receipt_StornoOpen_A09 ✔ 046_receipt_InvoiceStorno_03_Open: 046_receipt_StornoOpen_A10 ✔ нов метод: 046_receipt_StornoOpen_A11 ✔ 046_receipt_InvoiceStorno_02_Open: 046_receipt_StornoOpen_A12 ✔ нов метод: 046_receipt_StornoOpen_A13 ✔ 046_receipt_InvoiceStorno_01_Open: 046_receipt_StornoOpen_A14 ✔ нов метод: 046_receipt_StornoOpen_A15 ✔ 046_receipt_InvoiceStorno_Open: 046_receipt_StornoOpen_A16 ✔ логическа група Б: ✔ 046_receipt_Storno_07_Open: 046_receipt_StornoOpen_B01 ✔ нов метод: 046_receipt_StornoOpen_B02 ✔ 046_receipt_InvoiceStorno_05_Open: 046_receipt_StornoOpen_B03 ✔ нов метод: 046_receipt_StornoOpen_B04 ✔ нов метод: 046_receipt_StornoOpen_B05 ✔ нов метод: 046_receipt_StornoOpen_B06 ✔ липсва метод: 046_receipt_Storno_07_Open ✔ bug fix: 053_receipt_Total_PAmount

Версия 3.0.0.4

Version	Date	Description
3.0.0.4	02/2019	**Бета версия**! ✔ отстранени грешки по текущите команди; ✔ добавена е група „info“ - команди, даващи различна информация от страна на фискалното устройство; ✔ добавена е група „klen“ - команди за четене на клен на фискалното устройство; ✔ нов изброен тип, даващ информация за типа на пакетираното съобщение; ✔ нови “properties”; ✔ поправки в документацията; ✔ поправки в генерирането на сорскод; ✔ добавени енумерирани типове за нови модели;
3.0.0.3	02/2019	**Бета версия**! ✔ отстранени грешки;
3.0.0.2	02/2019	**Бета версия**! ✔ отстранени грешки;

Версия 3.0.0.1

Version	Date	Description
3.0.0.1	01/2019	**Бета версия**! Поддържа промените в протокола на фискалните устройства във връзка с промените в наредба №18 / 2018. Нов интерфейс: ICFD_BGR Поддържа следните модели: ✔ FP-800 ✔ FP-2000 ✔ FP-650 ✔ FMP-10 ✔ SK1-21F ✔ SK1-31F ✔ FP-550 ✔ DP-05 ✔ DP-15 ✔ DP-25 ✔ DP-35 ✔ DP-150 ✔ WP-50
2.0.0.xx	01/2002 - 2018	Поддръжка на всички фискални устройства на Датекс, поддържащи протокола на фискалния принтер. Различните версии и предоставени интерфейси обхващат промените в наредба №18 за описания период, както и логическите модификации на използвания протокол.
1.0.0.xx	01/2001	Поддръжка на принтер FP-3530

Въведение

Този докумемент съдържа информация относно COM Server "FP3530" който на практика представлява ActiveX библиотека за управление на фискалните устройства на Датекс в България. COM сървъра позволява лесна интеграция, употреба и управление от страна на всеки съвременен език за програмиране под OS Windows.

Ако не сте запознати с COM технологията – накратко казано, COM сървъра е обект, който предоставя на клиентския софтуер имплементация различни видове интерфейси. Клиентския софтуер може да използва имплементацията на всеки от предоставените интерфейси. Има два основни типа COM сървъри, “in-process” и „out-of-process”. "FP3530" е „out-of-process” COM сървър. Същността на COM е така наречения неутрален път за имплементиране на обекти. Те могат да бъдат използвани в среди за разработка – различни от използваната за разработката и създаването на предоставяните обекти.

“FP3530” Ви позволява като програмист да управлявате фискалните устройства на Датекс чрез различни видове на транспортния протокол – RS232, USB или TCP/IP (в зависимост от модела).

Поддържани устройства

По-долу е даден списък на устройствата на Датекс които могат да се управляват от “FP3530”.

Device	Type	RS-232	LAN (TCP/IP)
DP-05	ECR	Yes	No
DP-15		Yes	
DP-25	ECR	Yes	No
DP-35	ECR	Yes	No
DP-150	ECR	Yes	No
WP-50	ECR	Yes	No
FP-650	Fiscal printer	Yes	No
FP-2000	Fiscal printer	Yes	Yes
FP-800	Fiscal printer	Yes	Yes
SK1-21F	Fiscal printer	Yes	No
SK1-31F	Fiscal printer	Yes	No

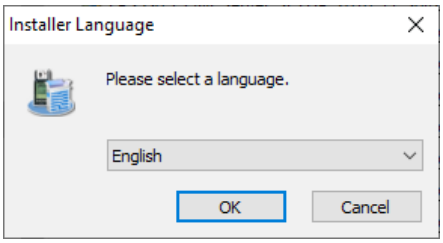
Поддръжка - езици

"FP3530" поддържа два езика – Английски и Български. Клиентските програми могат да изберат/настроят езика на COM сървъра програмно и след това текстовите съобщения за грешки или текста за статуса на устройството (статус битовете) ще се връщат на съответния език. Можете да видите пример за употребата на тези възможности в демо програмите разпространявани заедно с инсталацията.

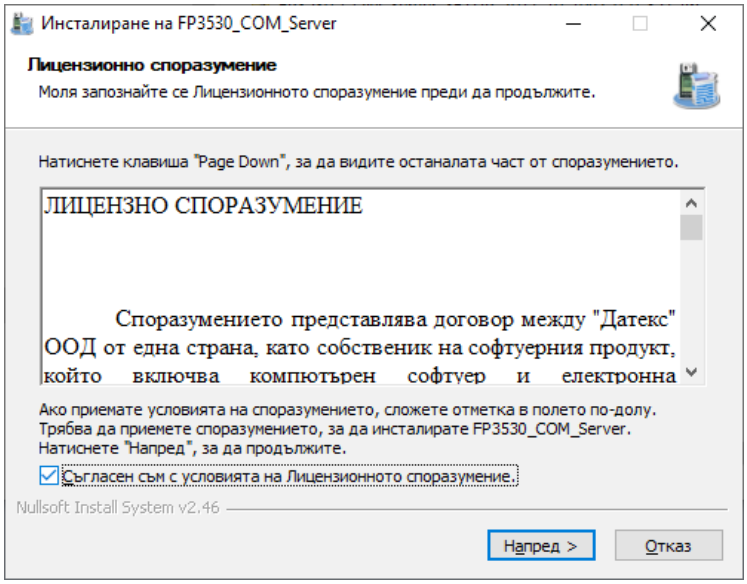
Инсталация

Инсталацията на "FP3530" е лесна. По-долу е описан процеса стъпка по стъпка.

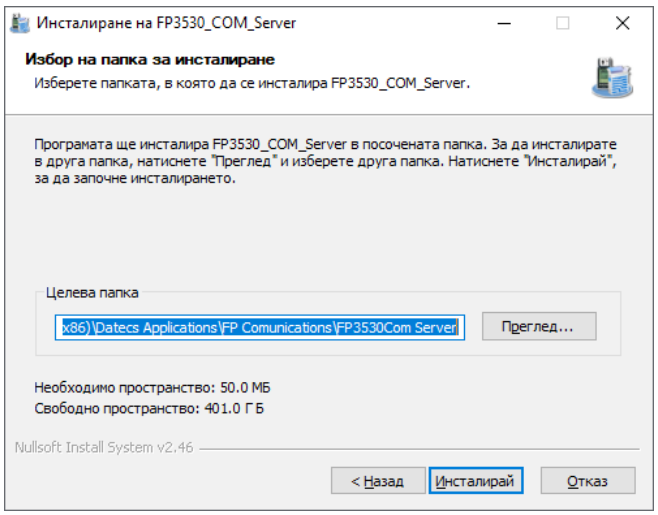
Стартирайте сетъп файла. Ако видите диалогов прозорец от типа на "User account control" (в по-новите версии на Windows), който може да е с различен вид в зависимост от версията на Windows – натиснете бутона "Yes".



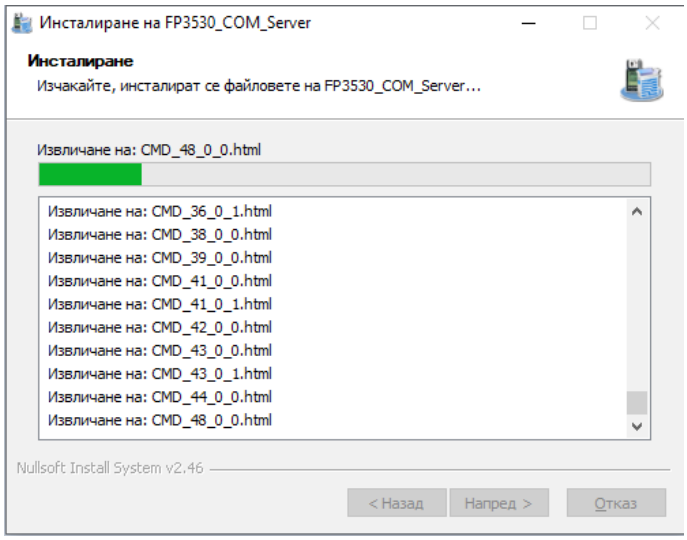
След показването на сплаш картинката, ще видите диалоговия прозорец за избор на език. Изберете предпочитания от Вас език и натиснете бутона "Ok", след което ще видите страницата "License Agreement".



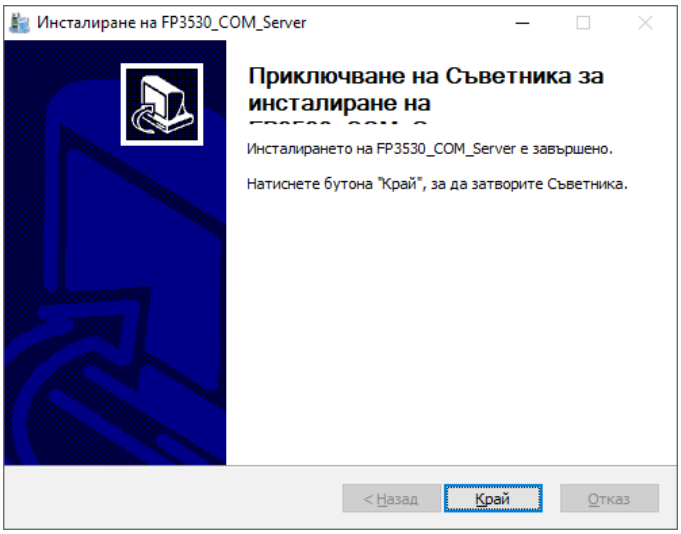
Прочетете лицензното споразумение и ако сте съгласни с условията му - изберете/включете, чекбокса за съгласие и натиснете бутона "Напред".



При желание - посочете папка за инсталиране и натиснете бутона „Инсталирай“



Моля изчакайте, докато приключи инсталацията.



Ако всичко е наред ще видите финалната страница на инсталатора.Натиснете бутона "Finish".

Това е всичко – the "FP3530" е инсталиран. Не трябва да го стартирате или да настройвате нещо. Необходимите настройки на версия 3.0.0.xx се управляват по програмен път, като "FP3530" има собствени настройки по подразбиране. Ако досега не сте използвали този тип интерфейси – разгледайте демо програмите. В някои от тях (на различни езици за програмиране) е показано как може да се управляват настройките. Демо програмите са предоставени както със сорскода си, така и в бинарен вид, така че могат да се използват от крайните потребители за целите на настройката, но екипа на Датекс препоръчва да имплементирате тези няколко метода в клиентските програми, които разработвате. "FP3530" записва сам последните настройки, с които е използван (след успешна връзка с фискалното устройство). Вашия потребителски интерфейс може да се възползва от тази черта в поведението на COM сървъра.

Няколко думи за протокола на ниско ниво

"FP3530" COM сървър енкапсулира имплементацията на описания по-долу протокол на ниско ниво. Не е необходимо да знаете протокола в детайли, но все пак тук предоставяме информация във връзка с това. Понякога – за нуждите на разработката – може да решите да използвате възможностите за проследяване/лог на комуникацията (tracking mode) на сървъра. Ако изпращате „bug report" свързан с това ниво на комуникация - ще е полезно да имате известни познания по темата.

Ще опиша по-долу, две модификации на протокол за комуникация с някои от най-широко разпространените модели фискални устройства, производство на Датекс ООД.

Терминология

В този раздел засягам използваната от мен терминология.

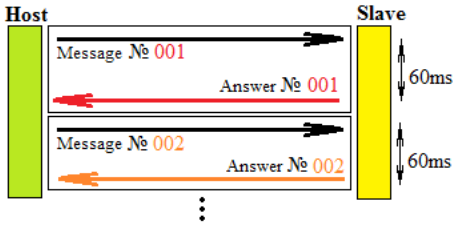
- Значението на бит и байт;
- Бройни системи:
  - Ще наричам за по-кратко десетичната бройна система DEC;
  - Ще наричам за по-кратко шестнайсеттичната бройна система HEX;
- Фискалното устройство ще наричаме за по-кратко ФУ или Slave;
- Компютъра (по-точно софтуера, който изпраща съобщенията до ФУ) ще го наричаме Host;
- Транспортен протокол:
  - Под серийна комуникация между Host и Slave ще разбираме свързаност по RS-232 или по USB;
  - Под БТ (BT) комуникация между Host и Slave ще разбираме свързаност по Bluetooth;
  - Под TCP/IP комуникация между Host и Slave ще разбираме свързаност по TCP/IP;
- Специфични, еднобайтови съобщения:
  - Когато говорим, че Slave е изпратил на/към Host, байт NAK ще разбираме байт със стойност 15H (HEX);
  - Когато говорим, че Slave е изпратил на/към Host, байт SYN ще разбираме байт със стойност 16H (HEX);

Синхронен протокол

Протокола за комуникация с фискалните устройства на Датекс ООД е синхронен.

- Имаме синхронизация в реда на започване, изпращане и получаване на съобщенията;
- Имаме синхронизация по време;
- Имаме снхронизация във формат и размер на съобщенията;

Синхронизация в реда на започване, изпращане и получаване на съобщенията

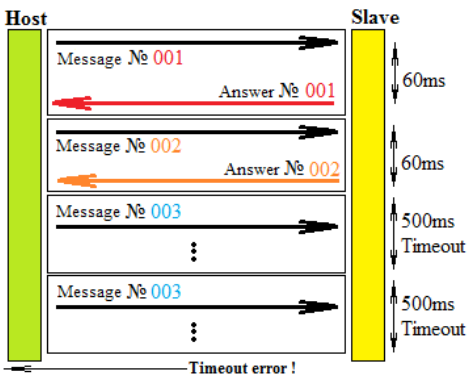


Фискалното устройство (**Slave**) не може да инициира комуникация. Тя се стартира задължително и винаги от програмата, която се намира на съответния компютър (**Host**).

Ако получавате някаква информация от страна на фискалното устройство по комуникационния канал без да сте я инициирали – това обикновено е признак за дефектирало устройство или за неизчистен буфер.

- Host инициира комуникация, като изпраща пакетирано съобщение, съдържащо команда към Slave;
- В зависимост от текущото си състояние Slave извършва или не исканата операция, след което задължително отговаря с пакетирано съобщение (или в някои случаи с единичен байт);
- Host трябва да изчаква отговора от Slave преди да изпрати друго съобщение, освен ако не изминало времето за таймаут (вж. синхронизацията по време);

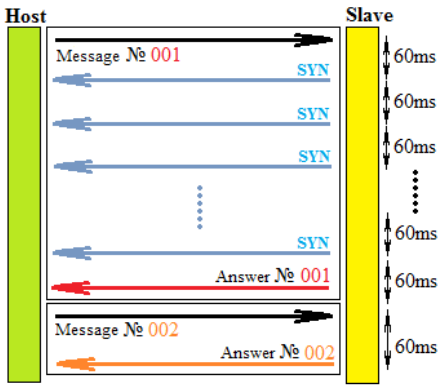
Синхронизация по време



- На всяка от командите идващи от Host – Slave трябва да отговори не по-късно от 60ms (при това или с пакетирано съобщение или със строго определен еднобайтов код);
- Host има 500ms timeout за получаване на отговор от Slave;

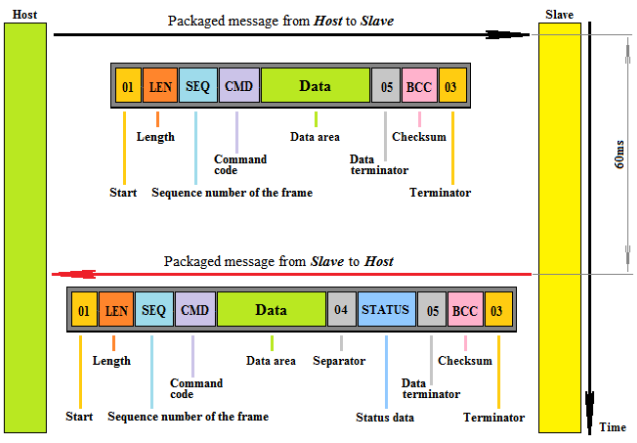
- Ако за 500ms не се получи никакъв отговор, Host трябва да предаде съобщението отново и при това със същия пореден номер и същата команда;
- След два неуспешни опита, Host трябва да индицира, че или няма връзка със Slave или че има хардуерен проблем в устройството;
- Timeout от 500ms се използва в случай, че става дума за комуникация по RS-232 или USB. Ако транспортния протокол е друг (например TCP/IP) - трябва да предвидите по-голям период от време в зависимост от качеството на трасето;

Поддръжка на нормално забавяне на отговора от страна на Slave



В случай, че командата е такава, че Slave ще се забави с отговора си, то Slave започва да изпраща по канала байт SYN на всеки 60ms. Така Slave казва на Host, че е зает в момента, но е напълно изправен и функциониращ. Казва му да изчака още малко.

Синхронизация по формат на съобщенията



- Всички команди и отговори между Host и Slave (освен описаните по-горе едно-байтови съобщения), са пакетирани в строго определен и предварително известен формат;
- "Случайност" се допуска единствено в областта за логически данни, но мястото където се намират тези данни в пакета, както и максималния им размер са строго определени;
- Форматите на пакетираното съобщение от Slave към Host и обратно са строго определени, но "различни", тъй като в отговора от страна на Slave се съдържа и допълнителна информация за статуса на устройството. Дори и ако устройството няма какво да изпрати, като логически данни към Host (т.е.в областта за данни няма нищо), то въпреки това отговаря, за да може програмата (Host) да научи от статус байтовете състоянието на устройството и да провери дали няма някакъв проблем (например, че е

свършила хартията);

- В случай на проблем или шум по трасето на комуникация - напълно възможно е Slave да получи команда с грешна контролна сума (чексума). В този случай Slave отговаря с един байт NAK. С него той казва, че Host трябва да повтори съобщението си без да променя нещо в него - просто да го изпрати отново.

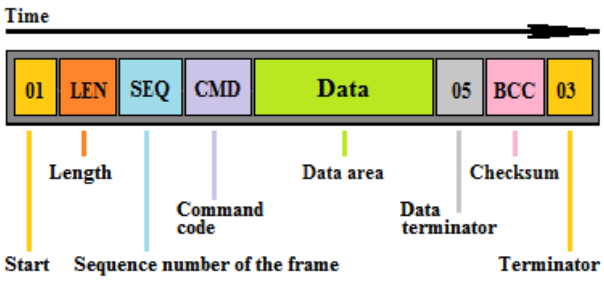
Модификации на протокола

Описаните по-долу две модификации, могат грубо да се разделят на "стар протокол" и "нов протокол". Така наречения "стар протокол", възниква по време, когато най-масовата операционна система е DOS и съществува заради нуждата от съвместимост със стари, но "живи" продукти. При втората модификация, която можем да наричаме за по-кратко "нов протокол" имаме някои съществени подобрения:

- заделя се по-голям интервал на допустимите команди (т.е. предвижда се появата в бъдещето на нови, неописани към момента команди);
- дава се възможността за прехвърляне на по-голям обем данни в рамките на една команда;
- елементи на стандартизация на командите, като резултат от опита на разработчиците на софтуер и на натрупаните изисквания от страна на различни разработчици на софтуер:
  - код на грешка, като първи параметър в отговора на устройството - с общо значение за всички модели, разработени по този протокол;
  - стандартизиране на сепараторите в областта за данни;
  - заделяне на повече статус байтове в отговора от страна на Slave;

И двата протокола са валидни и еднакво популярни, поради което ще опиша и двете модификации. Разликите са посочени в различен цвят. Стойностите са в HEX

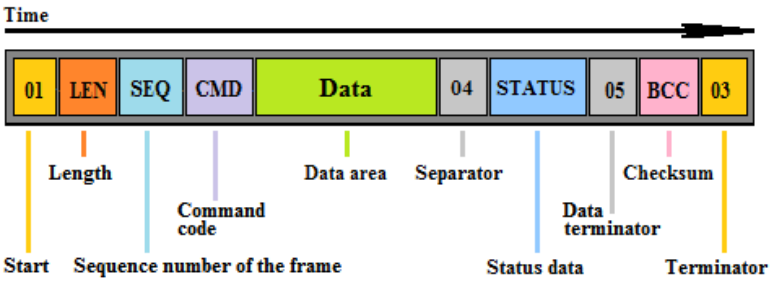
Общо описание на пакетираното съобщение



Packaged message from ***Host*** to ***Slave***

- **01 (Start)** - Стартиращ байт.
  - Дължина: 1 байт;
  - Стойност: 01H;
- **LEN** - Дължина на съобщението. Това са броя байтове от стартиращия байт <01> (**но без него**) до терминатора на данни <05> (**включително**), плюс фиксирано отместване от 20H.
  - Стар протокол:
    - Дължина: 1 байт;

        - Стойности: от 20H до FFH;
    - Нов протокол:
        - Дължина: 2 байта (word), кодирани в 4 байта;
        - Стойности: Тъй като в този протокол възможните стойности са в интервала [0..65535] (DEC), програмата Ви трябва да попълва необходимата стойност в променлива от тип word. Всяка цифра от тези два байта се предава, като към нея се прибави 30H, в резултат на което в пакетираното съобщение за това поле ще участват четири байта. Причината за това кодиране е свързана с вътрешната реализация на проткола във фискалните устройства и не от значение.
            - *Пример: променлива със стойност 1AE3H в пакетираното съобщение се представя като четири байта със стойности съответно 31H,3AH,3EH,33H.*



Packaged message from ***Slave*** to ***Host***

- SEQ - Пореден номер на пакетираното съобщение.
    - Slave записва същия пореден номер в отговора си;
    - Ако Slave получи съобщение със стайност на <SEQ>, същата като при последното получено от него съобщение, то той не извършва действие, а повтаря последното изпратено от него съобщение;
    - Стар протокол:
        - Стойности: от 20H до 7FH;
        - Дължина: 1 byte;
    - Нов протокол:
        - Стойности: от 20H до FFH;
        - Дължина: 1 байт;
- CMD - Номер на командата.
    - Ако Slave получи несъществуващ или невалиден код, той отговаря с пакетирано съобщение с нулева дължина на полето за данни и вдига в единица статус бита за невалиден код на командата;
    - Когато отговаря на дадено съобщение - Slave записва същия номер <CMD> в отговора си;
    - Горната граница на стойността на допустимите команди зависи от конкретното устройство;
    - Стар протокол:
        - Дължина: 1 байт.
        - Стойности: от 20H до 7FH;
    - Нов протокол:
        - Дължина: 2 байта (word), кодирани в 4 байта;
        - Стойности: Тъй като в този протокол възможните стойности са в интервала [0..65535] (DEC), програмата Ви трябва да попълва необходимата стойност в променлива от тип word. Всяка цифра от тези два байта се предава, като към нея се прибави 30H, в резултат на което в пакетираното съобщение за това поле ще участват четири байта. Причината за това кодиране е свързана с вътрешната

реализация на проткола във фискалните устройства и не от значение.
    - Пример: променлива със стойност 1AE3H в пакетираното съобщение се представя като четири байта със стойности съответно 31H,3AH,3EH,33H.
    - Забележка: Списъка с конкретните, позволени и валидни стойности на командите за дадено фискално устройство трябва да се взема от документацията на всяко конкретно устройство. Това, че дадена стойност е възможна - не означава, че се използва или че е позволена.
- **Data** - Логически данни.
    - Форматът и дължината на областта за данни зависи от конкретната команда;
    - Ако командата не изисква изпращане на данни към **Slave** или съответно, ако в отговора към **Host** няма данни, то дължината на това поле е нула;
    - Ако при команда от Host към Slave има синтактична грешка в данните, то Slave отговаря с пакетирано съобщение с нулева дължина на полето за данни и вдига в единица статус бита за синтактична грешка;
    - Дължина (от Host към Slave): 0 – 213 байта;
    - Дължина (от Slave към Host): 0 – 218 байта;
    - Стойности: 09H, 0AH и стойностите в интервала от 20H до FFH;
- **04 (Separator)** - Сепаратор. В пакетираното съобщение от Host към Slave - това е байта, който служи за маркер, отделящ логическите данни от данните за статуса на Slave (статус байтовете).
    - Дължина: 1 байт;
    - Стойност: 04H;
- **STATUS** - Полето за данни, съдържащо информация за текущото състояние на фискалното устройство. В статиите от тук нататък ще наричам байтовете от това поле: "**Статус байтове**";
    - Значението на всеки бит от всеки статус байт, трябва да се вземе от документацията на съответното устройство;
    - Състоянието на даден бит отговаря логически на някакво състояние на устройството. В статиите от тук нататък - ще наричам всеки такъв бит - "статус бит", а когато става дума за състоянието на даден статус ще имаме предвид стойността на съответния статус бит;
    - За съжаление е възможно позицията на даден статус да се мени при различните устройства или дори дадения статус да отсъства поради хардуерни специфики;
    - Като цяло статус битовете могат да бъдат разделени на следните групи и подгрупи:
        - такива, на които може да не се обръща внимание:
            - даден статус бит може да бъде запазен за вътрешна употреба (в зависимост от случая - може да бъде винаги със стойност 0 или 1);
            - даден статус бит може да бъде запазен за бъдеща употреба (в зависимост от случая - може да бъде винаги със стойност 0 или 1);
        - такива, на които задължително трябва да обръщате внимание:
            - Информативни статус битове - носят информация за логическо или физическо състояние на устройството. Някои операции не са позволени в дадени ситуации и поради това е добре да ги следите. Например:
                - Ако принтера не е фискализиран - явно не може да издава фискални бележки;
                - Ако принтера няма хартия - не може да печата, каквито и да било документи;
            - Статус битове за грешка - носят обща информация за грешката, която е възникнала. Например: непозволена команда, синтактична грешка и т.н. Понякога (в стария протокол) единствения начин програмата да вземе

правилно решение за разрешаване на ситуацията е да сглоби информацията от всички налични статуси в отговора от страна на Slave;

- Стар протокол:
  - Дължина: 6 байта;
  - Стойности за всеки от байтовете: от 80H до FFH;
- Нов протокол:
  - Дължина: 8 байта;
  - Стойности за всеки от байтовете: от 80H до FFH

- 05 (Data terminator) - Сепаратор.
  - В пакетираното съобщение от Host към Slave - това е байта, който служи за маркер, отделящ логическите данни от останалата част на пакетираното съобщение.
  - В пакетираното съобщение от Slave към Host - това е байта, който служи за маркер, отделящ статус данните от останалата част на пакетираното съобщение.
    - Дължина: 1 байт;
    - Стойност: 05H;
- **BCC** - Контролна сума (чексума). Сумата включва стойностите от <01> без него до <05> включително. Тъй като в този протокол възможните стойности са в интервала [0..65535] (DEC), програмата Ви трябва да попълва необходимата стойност в променлива от тип word. Всяка цифра от тези два байта се предава, като към нея се прибави 30H, в резултат на което в пакетираното съобщение за това поле ще участват четири байта. Причината за това кодиране е свързана с вътрешната реализация на проткола във фискалните устройства и не от значение.
  - Пример: променлива със стойност 1AE3H в пакетираното съобщение се представя като четири байта със стойности съответно 31H,3AH,3EH,33H.
    - Дължина: 2 байта (word), кодирани в 4 байта;
    - Стойности: от 30H до 3FH.
- **03 (Terminator)** - Байт, маркиращ края на пакетираното съобщение.
  - Дължина: 1 byte.
  - Стойност: 03H.

Употреба на FP3530

Няколко думи във връзка със зависимостта от хардуера

FP3530 и по-специално интерфейс „CFD_BGR“ има смесени характеристики по отношение на зависимостта от харуер и фирмуер. Навсякъде където това е било възможно – методите и техните параметри не зависят от конкретния фирмуер, но за съжаление това не винаги е възможно. Някои минимални или максимални стойности (например брой печатими символи) очевидно зависят от конкретното устройство. В други случаи (поради особеностите на Българския пазар) устройствата „наследяват“ предишни версии на дадения модел.

Когато разработвате Вашия проект – имайте предвид, че интерфейс „CFD_BGR“ ще поддържа всички фискални устройства на Датекс, но като цяло това е зависима от устройството библиотека.

Интерфейс „CFD_BGR“ предоставя необходимите методи и характеристики с които да Ви улесни максимално възможно и продукта Ви да знае във всеки един момент както какъв е модела, така и какво е неговото състояние.

Групи фискални устройства в зависимост от типа на пакетираното съобщение

Интерфейс „CFD_BGR“ предоставя информация за типа на пакетираното съобщение след успешно отваряне на връзката към фискалното устройство. За целта интерфейса предоставя енумериран тип „**TpackagedMessageType**“ и property за четене “**type_PackagedMessage**”. По отношение на типа на пакетираното съобщение – фискалните устройства на Датекс се разделят на две основни групи към днешна дата. Ако използвате интерфейс „CFD_BGR“, Вие няма да се занимавате с пакетиране и разпакетиране на съобщенията на „ниско“ ниво. Практическата и видима за Вас разлика е в броя на статус битовете които фискалното Ви изпраща при всеки отговор на команда. Вижте таблицата по-долу за повече информация.

Групи фискални устройства в зависимост от входните данни при изпълнение на различни методи

През годините – фирмуера за различните фискални устройства на Датекс са създавани от различни екипи. Някои от тях външни фирми – по-късно закупени от Датекс. Поради това през годините се получава постепена разходимост в логиката и параметрите на различните модели фискални устройства. Датекс предприема съответните стъпки за постепен преход към стандартизация в предлаганите методи, но тъй като трябва да се спазва и условието за наследяването на функционалността (при всяка промяна в законите има преходен период през който стар и непроменен софтуер трябва да работи едновременно с новия) това не е съвсем лесна задача. Логически, ако разделим фискалните устройства в зависимост от поведението и методите които те предлагат – всяко фискално устройство предлагано от Датекс попада в една от три общи групи. Ако използвате интерфейс „CFD_BGR“, Вие ще

имате достъп както до методите, специфични за дадената група, така и до предлаганите от интерфейса независими от модела методи.

Стойност	Модел	Логическа група „A“	Логическа група „B“	Логическа група „C“	Тип на пакетираното съобщение	Брой статус битове	Стандартизиран код на грешка в отговора	Стандартизиран делимитер
mc_Unknown	Стойност на property “**device_Model**” преди отваряне на връзка към фискалното устройство.							
mc_DP_05	**DP-05**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_15	**DP-15**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_25	**DP-25**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_35	**DP-35**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_150	**DP-150**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_WP_50	**WP-50**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_FP_650	**FP-650**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_800	**FP-800**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_2000	**FP-2000**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_SK_21F	**SK-21F**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_SK_31F	**SK-31F**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FMP_10	**FMP-10**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_550	**FP-550**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_700X	**FP-700X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_DP_25X	**DP-25X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_DP_150X	**DP-150X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_WP_50X	**WP-50X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_WP_500X	**WP-500X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_FMP_55X	**FMP-350X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_FMP_350X	**FMP-55X**	✗	✗	✔	**pmt_02**	**8**	✔	✔

“Thread safe” ?

Един от често задаваните въпроси за COM сървъра е "Thread-safe" or not?".

Термина "Thread-safe" е малко особен. Трябва да знаем какво всъщност значи в този конкретен случай. Нека погледнем дефиницията: "A piece of code is thread-safe if it functions correctly during simultaneous execution by multiple threads."

При директна употребата на методите на този COM сървър - не бива да смятате, че тази библиотека е "Thread-safe"! Когато извиквате метод с директна комуникация към фискалното устройство (например и условно казано, „Изпълни команда ...“) COM сървъра не проверява и не поставя изпълнението на тази команда в опашка. Ако извиквате командите към устройството едновременно от няколко различни нишки (threads) – вероятно изпълнението на командите няма да е коректно. Препоръчваме Ви в зависимост от интерфейса който използвате – да организирате собствена подредба на командите в опашка.

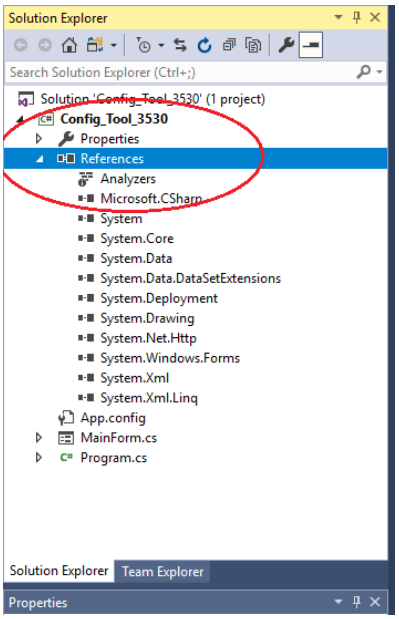
В интерфейс „CFD_BGR“ е заложен механизъм за транзакционно използване на командите и подреждане в опашка. Ако решите да го използвате – използвайте само този подход, защото ако едновременно с това извиквате от друга нишка команда към фискалното устройство – употребата на интерфейса отново няма да е *"Thread-safe"*.

При проектирането на Вашия продукт – вземете тази информация под внимание! Пишете ни – ще помогнем със съвет.

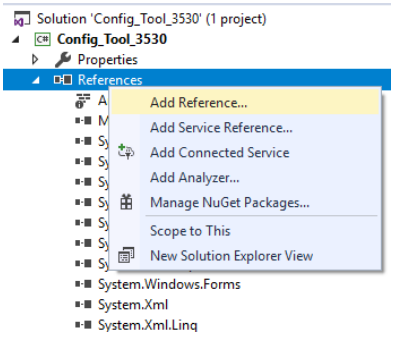
Използване на интерфейс „CFD_BGR“ от Visual Studio 2017 (C#)

Лесно е!

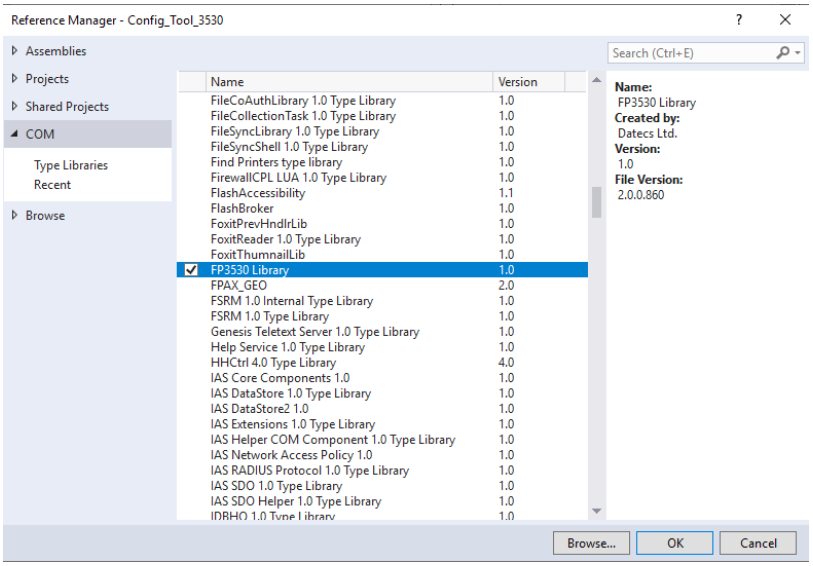
- ✔ Инсталирайте FP3530 COMServer и след това – отидете на "Solution Explorer\ References" в проекта Ви;



- ✔ Натиснете десния бутон на мишката и изберете “Add reference” ;



- ✔ В диалоговия прозорец – изберете COM и намерете FP3530 в списъка. Натиснете бутона “OK”;



✔ Това е всичко – Visual Studio ще добави всички предоставени от сървъра интерфейси и Вие можете да започнете да ги използвате;

```
1  using System;
2  using System.Collections.Generic;
3  using System.Drawing;
4  using System.IO;
5  using System.ComponentModel;
6  using System.Data;
7  using System.Linq;
8  using System.Text;
9  using System.Threading;
10 using System.Windows.Forms;
11 using System.Globalization;
12 using System.Diagnostics;
13 using FP3530;
14 using System.Runtime.InteropServices;
15
16 namespace Config_Tool_3530
17 {
```

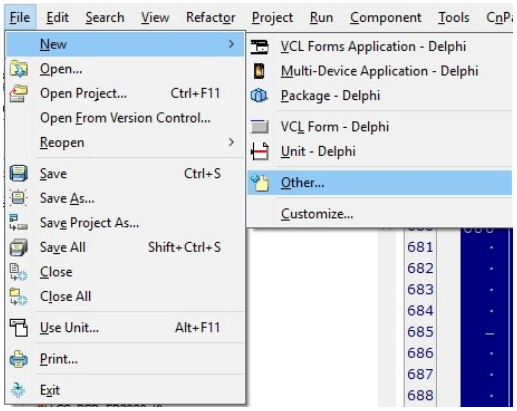
```
16 namespace Config_Tool_3530
17 {
18     public partial class fm : Form
19     {
20         private bool inCommand = false;
21         private bool invalidCharacter1;
22         private string command_Name = "";
23         private bool fWeHaveWaitEvent;
24         private byte fSYNCount;
25         private byte fAppMessCount;
26         private FP3530.CFD_BGR serv;
27         private bool fComServer_Started;
28         private IMyLanguages fLanguage;
29         private bool fDevice_Connected;
30         private int fComPort;
31         private bool fLAN_Connected;
```

✔ Можете да намерите демо програми написани на C# в папките идващи с инсталацията;

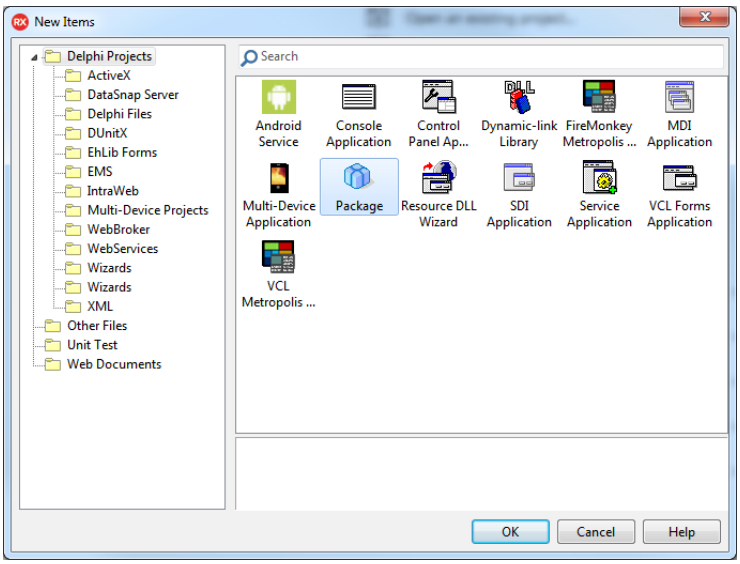
Употреба на интерфейс „CFD_BGR“ от Delphi

В Делфи има повече от един начин за употреба на COM сървър – тук показваме препоръчвания от разработчика (и най-лесен) метод. Имайте впредвид, че при някой версии на Windows е добре да стартирате Delphi като администратор.

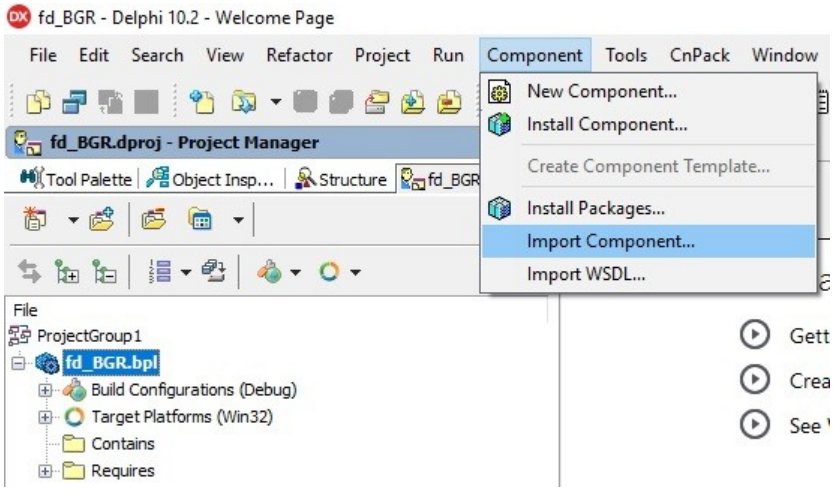
Можете лесно да си направите пакет, в който да добавите „type library”. След компилирането и регистрирането на пакета – ще можете да използвате „CFD_BGR“ като компонент. Описания по-долу пример е направен под Delphi 10.2 но лесно може да се приложи за всички по-стари версии на Delphi – назад до версия Delphi 7 (автора на документа го изпробвал до тази версия).



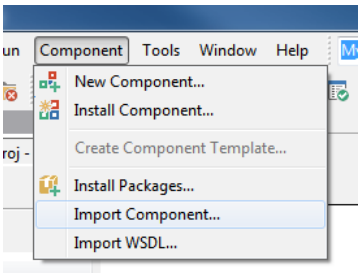
✔ Отидете на “File\New” и изберете “Other”;



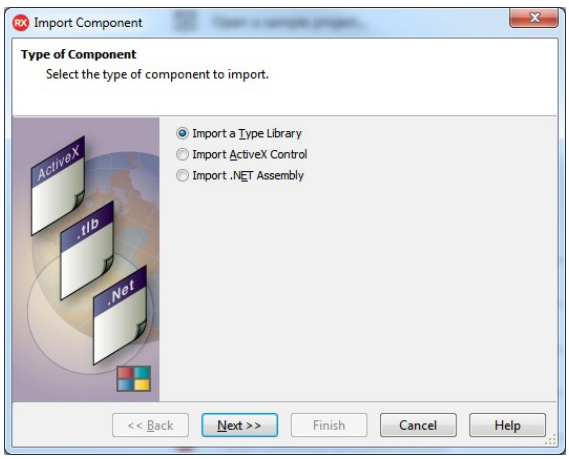
✔ Посочете package и натиснете бутона “OK”;



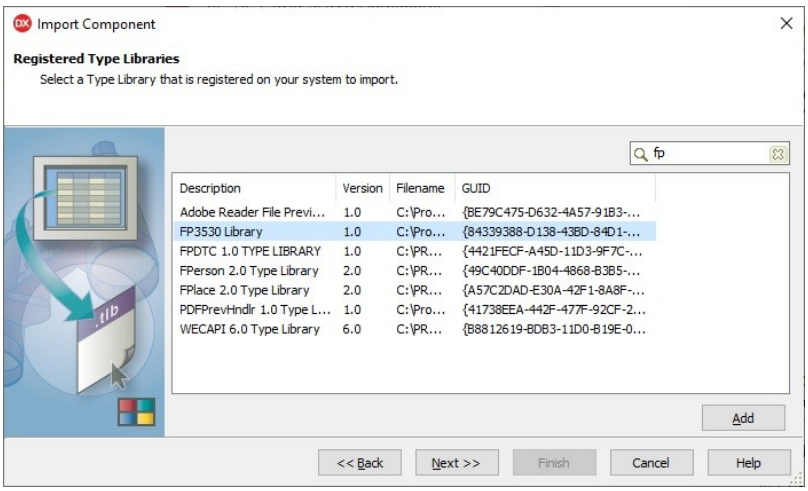
✔ Delphi ще направи нов празен проект за „package” - можете да го запишете в зависимост от предпочитанията си – например като “fd_BGR.dproj” (от фискални устройста – България);



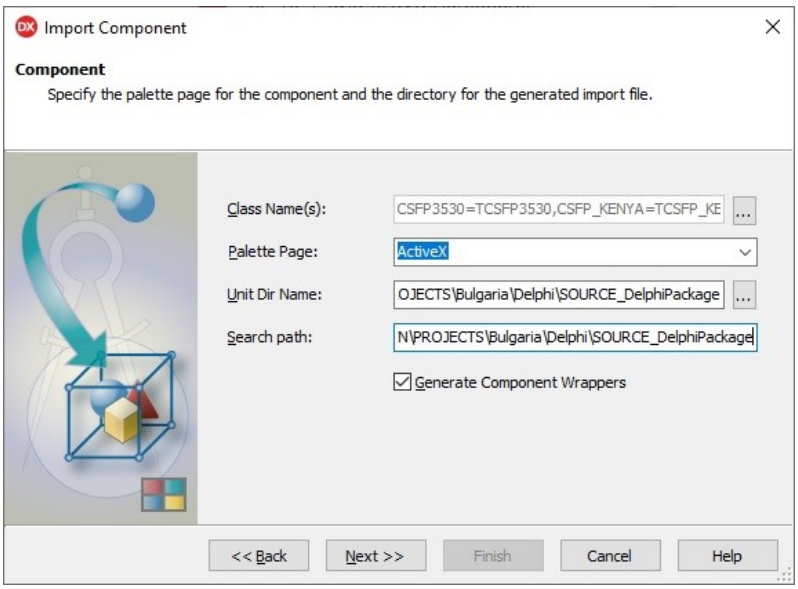
✔ В основното меню изберете “Component\Import Component”;



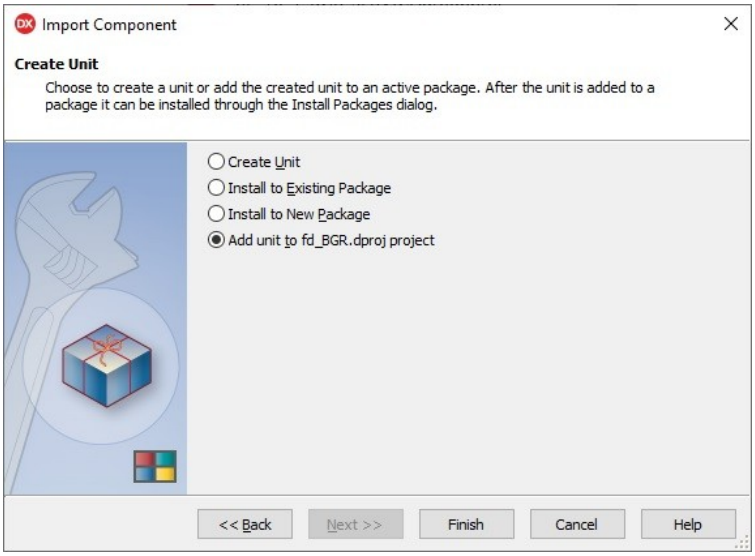
✔ Изберете “Import a Type Library” и натиснете бутона “Next”;



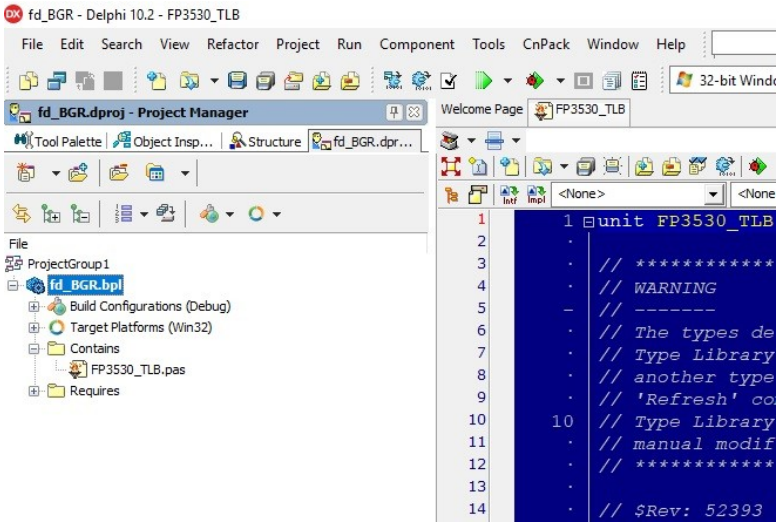
✔ На следващата страница – намерете FP3530 и натиснете бутона “Next”;



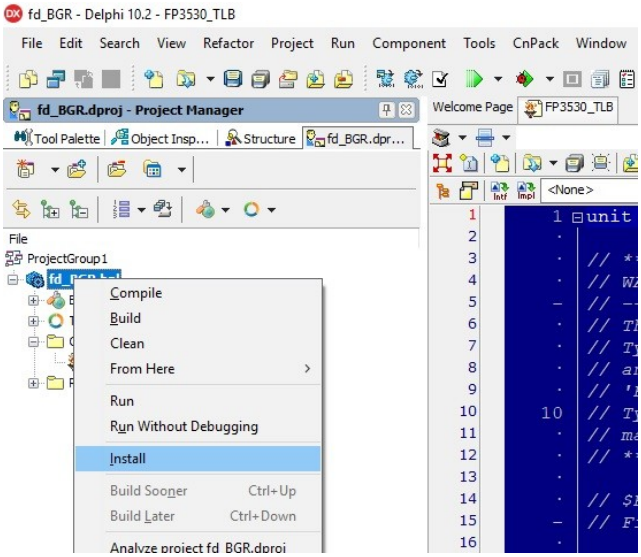
✔ На следващата страница – изберете “Palette page”, “Unit dir name” и активирайте „Generate component wrappers”. Натиснете бутона “Next”;



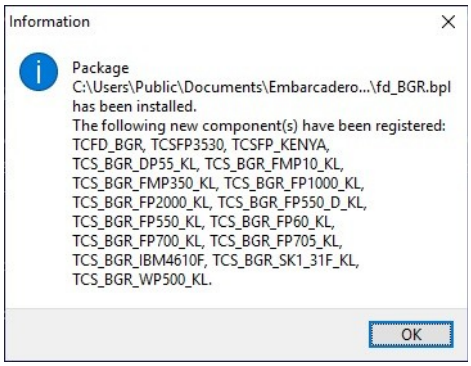
✔ Изберете опция “Add unit to fd_BGR.dproj project” и натиснете бутона “Finish”;



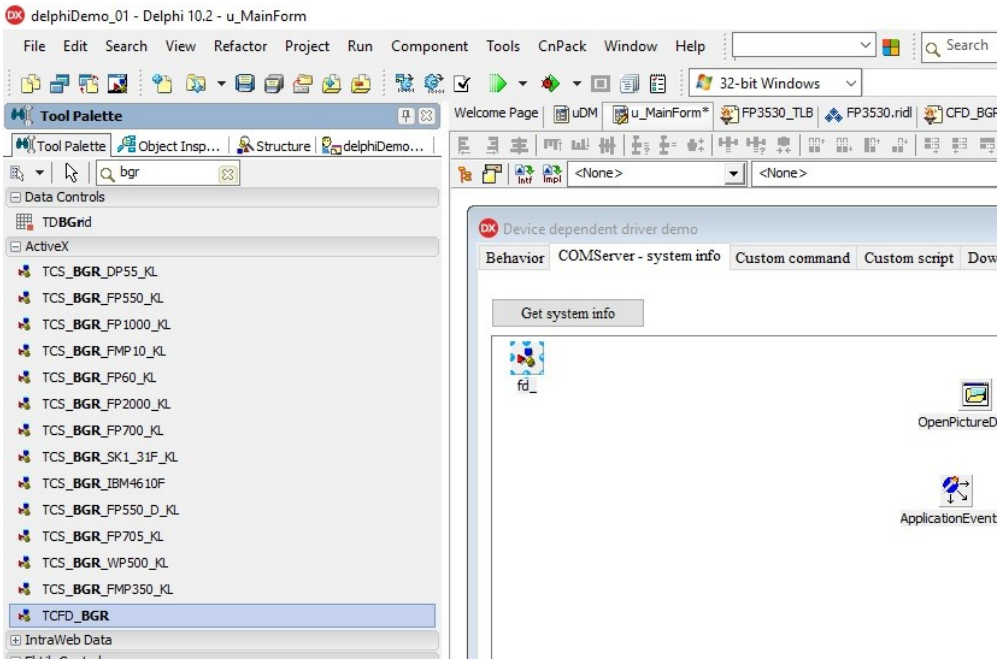
✔ Delphi ще създаде за Вас “FP3530_TLB.pas”. От контекстното меню изберете Save и после Build the project;



✔ Инсталирайте пакета в Delphi IDE и можете да започнете да използвате FP3530 в проекта си;



✔ Можете да намерите новите компоненти в “Tool Palette”;



- ✔ Можете да намерите демо програми със сорскода в инсталационните папки;
- ✔ Това е всичко;

Забележка за демо проекта свързан с интерфейс „CFD_BGR“: е използван отворен сорскод от проекта JVCL/JCL. Сорскода на този проект може да бъде намерен на адрес: https://github.com/project-jedi, https://github.com/TurboPack/SynEdit, *или посредством GetIt Package manager” в по-новите версии на Delphi.*

Енумерирани типове

FP3530 – COM Server предлага всички enumerated types които му трябват вътрешно и които трябва да се използват при саответната употреба на методи или property.

TMyLanguages

Клиентския софтуер може да избере/настрои човешкия език за съобщенията от страна на COM сървъра по програмен начин. Освено възможните текстови съобщения (LastErrorMessage) тази настройка влияе и на текста който библиотеката връща като описание на статус битовете за даденото фискално устройство.

Възможни стойности:

- English
- Bulgarian

Забележка: Обърнете внимание на демо проектите идващи с инсталацията.

TTransportProtocol

Клиентското приложение трябва да настрои типа на комуникация с устройството преди да опита да отвори връзка към него.

Възможни стойности:

- ctc_RS232 – serial communication via RS-232 or USB;
- ctc_TCPIP – TCP/IP communication via LAN/Internet

TDeviceTypes

След успешна връзка с фискалното устройство, интерфейс „CFD_BGR“ настройва съответната характеристика, за да можете да я използвате.

Възможни стойности:

- dtc_FiscalPrinter – фискален принтер;
- dtc_ECR – касов апарат;

TDeviceModel

След успешна връзка с фискалното устройство, клиентските програми могат да получат информация от интерфейс „CFD_BGR“ за модела на фискалното устройство. Това е особено полезно, както заради разликите в хардуера, така и заради разликите в някои от методите на фирмуера.

Възможни стойности:

Стойност	Модел	Логическа група „A“	Логическа група „B“	Логическа група „C“	Тип на пакетираното съобщение	Брой статус битове	Стандартизиран код на грешка в отговора	Стандартизиран делимитер
mc_Unknown	Стойност на property “**device_Model**” преди отваряне на връзка към фискалното устройство.							
mc_DP_05	**DP-05**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_15	**DP-15**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_25	**DP-25**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_35	**DP-35**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_DP_150	**DP-150**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_WP_50	**WP-50**	✗	✔	✗	**pmt_01**	**6**	✗	✗
mc_FP_650	**FP-650**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_800	**FP-800**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_2000	**FP-2000**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_SK_21F	**SK-21F**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_SK_31F	**SK-31F**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FMP_10	**FMP-10**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_550	**FP-550**	✔	✗	✗	**pmt_01**	**6**	✗	✗
mc_FP_700X	**FP-700X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_DP_25X	**DP-25X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_DP_150X	**DP-150X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_WP_50X	**WP-50X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_WP_500X	**WP-500X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_FMP_55X	**FMP-350X**	✗	✗	✔	**pmt_02**	**8**	✔	✔
mc_FMP_350X	**FMP-55X**	✗	✗	✔	**pmt_02**	**8**	✔	✔

TpackagedMessageType

Интерфейс „CFD_BGR“ предоставя информация за типа на пакетираното съобщение след успешно отваряне на връзката към фискалното устройство.

Възможни стойности:

<table>
<tr><th>Стойност</th><th>Описание</th></tr>
<tr><td>pmt_Unknown</td><td>Стойност на property за четене “type_PackagedMessage” преди отваряне на връзка към фискалното устройство.</td></tr>
<tr><td>pmt_01</td><td>Стойност на property за четене “type_PackagedMessage” след отваряне на връзка към фискално устройство от следния списък:
<table>
<tr><th>Модел</th><th>Логическа група „A“</th><th>Логическа група „B“</th><th>Логическа група „C“</th></tr>
<tr><td>FP-800</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>FP-2000</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>FP-650</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>SK1-21F</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>SK1-31F</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>FMP-10</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>FP-550</td><td>✔</td><td>✗</td><td>✗</td></tr>
<tr><td>DP-05</td><td>✗</td><td>✔</td><td>✗</td></tr>
<tr><td>DP-15</td><td>✗</td><td>✔</td><td>✗</td></tr>
<tr><td>DP-25</td><td>✗</td><td>✔</td><td>✗</td></tr>
<tr><td>DP-35</td><td>✗</td><td>✔</td><td>✗</td></tr>
<tr><td>WP-50</td><td>✗</td><td>✔</td><td>✗</td></tr>
<tr><td>DP-150</td><td>✗</td><td>✔</td><td>✗</td></tr>
</table></td></tr>
<tr><td>pmt_02</td><td>Стойност на property за четене “type_PackagedMessage” след отваряне на връзка към фискално устройство от следния списък:
<table>
<tr><th>Модел</th><th>Логическа група „A“</th><th>Логическа група „B“</th><th>Логическа група „C“</th></tr>
<tr><td>FP-700X</td><td>✗</td><td>✗</td><td>✔</td></tr>
<tr><td>DP-25X</td><td>✗</td><td>✗</td><td>✔</td></tr>
<tr><td>DP-150X</td><td>✗</td><td>✗</td><td>✔</td></tr>
<tr><td>WP-50X</td><td>✗</td><td>✗</td><td>✔</td></tr>
<tr><td>WP-500X</td><td>✗</td><td>✗</td><td>✔</td></tr>
<tr><td>FMP-350X</td><td>✗</td><td>✗</td><td>✔</td></tr>
<tr><td>FMP-55X</td><td>✗</td><td>✗</td><td>✔</td></tr>
</table></td></tr>
</table>

TscriptType

Интерфейс „CFD_BGR“ предоставя метод чрез който клиентското приложение може да изпълнява предварително подготвени текстови скриптове. Този тип се използва, за да може скрипт енджина на интерфейса да разбере какъв е „езика“ на скрипта.

Възможни стойности:

Стойност	Описание
• **DS**	Датекс скрипт.

***Забележка**: Датекс си запазва правото да добавя нови скриптови „езици“ и да разширява и подобрява механизма на обработка на скрипта.*

TCodeType

„CFD_BGR“ може да изпълнява команди, подадени от клиентското приложение по три начина:

- Чрез употребата на "execute_Command"
- Чрез употребата на "execute_Script_V1"
- Чрез: "execute_Command_ByName"

В тази версия на „CFD_BGR“ - библиотеката може да генерира примерен сорскод на два програмни езика за всички команди експортирани чрез метода "get_ComandsList".

Възможни стойности:

Стойност	Описание
Delphi	Метода ще генерира сорс код на Delphi (beta)
CSharp	Метода ще генерира сорс код на C# (beta)

***Забележка**: В текущата бета версия на интерфейс „CFD_BGR“, библиотеката генерира сорскод единствено на Pascal (Delphi) и C#. Генерирания сорскод не е „задължителен“, а служи за ориентиране – как да се използва **"execute_Command_ByName"**. Можете да използвате или модифицирате както намерите за добре предложения от нас сорс код. Екипа на Датекс приема препоръки за подобрение, оптимизация и отстраняване на грешки. Ако желаете да ни помогнете, за да добавим и Вашия любим език за програмиране – свържете се с нас. Моля – приемете нашите извинения, ако в генерирания код има грешки към дадения момент.*

Properties

Статус на фискалното устройство

Статус битове за грешка

В зависимост от типа на пакетираното съобщение, текущия статус се определя от отговора и/ или от състоянието на статус битовете на устройството. При първи тип на пакетираното съобщение статуса на устройството се кодира в 6 статус байта. При втория тип на пакетираните съобщения, статуса на устройството се определя от кода на грешката върнат при изпълнение на команда и от състоянието на 8 статус байта. Интерфейс „CFD_BGR“ предоставя „properties” (само за четене) които връщат състоянието на определени статус битове, вдигнати в състояние 1 при грешка. Тези флагове се вдигат от страна на фискалното устройство в случай на грешка или при някои типове повреди в самото фискално устройство.

Вие трябва (в програмата Ви) да четете и използвате статуса на тези properties:

- След успешно установена връзка с фискалното устройство;
- След изпълнение на метод, команда или скрипт;

Property name	Property type	Status [byte,bit]	Human meaning
eSBit_GeneralError_Sharp	Boolean Read only	[0,5]	General error - this is OR of all errors marked with #
eSBit_PrintingMechanism	Boolean Read only	[0,4]	# Failure in printing mechanism.
eSBit_ClockIsNotSynchronized	Boolean Read only	[0,2]	The real time clock is not synchronized.
eSBit_CommandCodeIsInvalid	Boolean Read only	[0,1]	# Command code is invalid.
eSBit_SyntaxError	Boolean Read only	[0,0]	# Syntax error.
eSBit_CommandNotPermitted	Boolean Read only	[1,1]	# Command is not permitted.
eSBit_Overflow	Boolean Read only	[1,0]	# Overflow during command execution.
eSBit_EJIsFull	Boolean Read only	[2,2]	EJ is full.
eSBit_EndOfPaper	Boolean Read only	[2,0]	# End of paper.
eSBit_FM_NotFound	Boolean Read only	[4,6]	Fiscal memory not found or damaged.
eSBit_FM_NotAccess	Boolean Read only	[4,0]	* Error when trying to access data stored in the FM.
eSBit_FM_Full	Boolean Read only	[4,4]	* Fiscal memory is full.
eSBit_GeneralError_Star	Boolean Read only	[4,5]	OR of all errors marked with ‘*’

Статус на устройството – информативни статус битове

В зависимост от типа на пакетираното съобщение, текущия статус се определя от отговора и/ или от състоянието на статус битовете на устройството. При първи тип на пакетираното съобщение статуса на устройството се кодира в 6 статус байта. При втория тип на пакетираните съобщения, статуса на устройството се определя от кода на грешката върнат при изпълнение на команда и от състоянието на 8 статус байта. Интерфейс „CFD_BGR“ предоставя „properties” (само за четене) които връщат състоянието на определени статус битове, вдигнати в състояние 1, но не се смятат за грешка. Това са информативни битове които носят полезна информация за програмата Ви.

Вие трябва (в програмата Ви) да четете и използвате статуса на тези properties:

- След успешно установена връзка с фискалното устройство;
- След изпълнение на метод, команда или скрипт;

Property name	Property type	Status [byte,bit]	Human meaning
iSBit_Cover_IsOpen	Boolean Read only	[0,6]	Cover is open.
iSBit_No_ClientDisplay	Boolean Read only	[0,3]	No client display connected.
iSBit_Receipt_Nonfiscal	Boolean Read only	[2,5]	Nonfiscal receipt is open.
iSBit_EJ_NearlyFull	Boolean Read only	[2,4]	EJ nearly full.
iSBit_Receipt_Fiscal	Boolean Read only	[2,3]	Fiscal receipt is open.
iSBit_Near_PaperEnd	Boolean Read only	[2,1]	Near paper end.
iSBit_LessThan_50_Reports	Boolean Read only	[4,3]	There is space for less than 60 reports in Fiscal memory.
iSBit_Number_SFM_Set	Boolean Read only	[4,2]	Serial number and number of FM are set.
iSBit_Number_Tax_Set	Boolean Read only	[4,1]	Tax number is set.
iSBit_VAT_Set	Boolean Read only	[5,4]	VAT are set at least once.
iSBit_Device_Fiscalized	Boolean Read only	[5,3]	Device is fiscalized.
iSBit_FM_formatted	Boolean Read only	[5,1]	FM is formatted.

Получаване на статус битовете на фискалното устройство чрез употреба методи

Интерфейс „CFD_BGR“ предоставя няколко метода чрез които можете не само да получите състоянието и описанието на всеки статус бит, но и сами да определите или промените поведението на библиотеката по отношение на статус битовете.

Брой на статус битовете за текущото фискално устройство

Освен логически по таблицата с модели на фискалните устройства или от типа на пакетираното съобщение, Вие можете да получите броя на статус битовете от стойността на read only property - „count_StatusBytes“. Коректната стойност се попълва след успешно установяване на връзка с фискалното устройство.

Метод „get_Sbit_State“

Чрез този метод можете да получите текущото състояние на точно определен от Вас статус бит по индекси съответно за байт и за бит.

Състоянието на статус битовете се обновява вътрешно след изпълнението на всяка команда. Например, ако се опитате да изпълните команда за отваряне на бон с неправилно форматирани входни данни – статус бита за синтактична грешка ще се вдигне след обработката на отговора от страна на библиотеката. Можете да го прочетете с този метод многократно и всеки път (до момента на изпълнение на нова команда) статус бита ще е вдигнат в единица(метода връща за удобство True/False съответно за състояние 1/0). Ако след това изпълните валидна команда (например команда 74) – състоянието на този статус бит ще премине в 0, а този метод ще върне False, т.е. нямаме синтактична грешка при последната команда.

Метод „get_Sbit_Description“

Чрез този метод можете да получите текстовото описание на точно определен от Вас статус бит по индекси съответно за байт и за бит. Езика на който се връща текста зависи от избраната настройка.

Метод „get_Sbit_ErrorChecking“

Чрез този метод можете да получите какво ще бъде поведението на библиотеката по отношение на дадения статус бит. Ако метода върне **True** за даден статус бит – това означава, че библиотеката ще смята вдигането му в единица за флаг за грешка. Библиотеката ще върне информация, че съответния метод не е правилно изпълнен.

Метод „set_Sbit_ErrorChecking“

Чрез този метод можете да определите какво да бъде поведението на библиотеката по отношение на даден статус бит. Например, ако желаете програмата да не работи, ако към фискалното устройство не е свързан дисплей – можете да настроите поведението на библиотеката по отношение на статус бита за свързан дисплей. Изпратете стойност **True** за този статус бит и библиотеката ще започне да връща грешка в статус битовете при изпълнението на който и да било метод, ако към фискалното устройство не е свързан дисплей.

Този метод е удобен и по отношение на „неочаквани“ промени в значението на даден статус бит в бъдещи фискални устройства. Вие ще можете да включвате или изключвате вдигането на грешка в зависимост от конкретната ситуация.

Статус битове по групи фискални устройства

Статус байт 0

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
0	7	Резервиран – винаги е 1	Резервиран – винаги е 1	Резервиран – винаги е 0.
	6	Отворен е капакът на принтера	Резервиран – винаги е 0.	Отворен е капакът на принтера
	5	Обща грешка - това е OR на всички грешки, маркирани с ‘#’.	Обща грешка - това е OR на всички грешки, маркирани с ‘#’.	Обща грешка - това е OR на всички грешки, маркирани с ‘#’
	4	(#) Механизмът на печатащото устройство има неизправност	Резервиран – винаги е 0.	(#) Механизмът на печатащото устройство има неизправност
	3	Не е свързан клиентски дисплей	Не е свързан клиентски дисплей.	Не е свързан клиентски дисплей.
	2	Не е сверен часовника.	Не е сверен часовника.	Не е сверен часовника.
	1	(#) Кодът на получената команда е невалиден	(#) Кодът на получената команда е невалиден	(#) Кодът на получената команда е невалиден
	0	(#) Получените данни имат синктактична грешка	(#) Получените данни имат синктактична грешка.	(#) Получените данни имат синктактична грешка.

Статус байт 1

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
1	7	Резервиран – винаги е 1	Резервиран – винаги е 1	Резервиран – винаги е 1
	6	Вграденият данъчен терминал не отговаря	Вграденият данъчен терминал не отговаря	Резервиран – винаги е 0.
	5	Отворен е служебен бон за печат на завъртян на 90 градуса текст	Има неизпратени документи за повече от настроеното време за предупреждение	Резервиран – винаги е 0.
	4	Отворен сторно бон	Резервиран – винаги е 0.	Резервиран – винаги е 0.
	3	(#) Слаба батерия (Часовникът за реално време е в състояние RESET)	Резервиран – винаги е 0.	Резервиран – винаги е 0.
	2	(#) Извършено е зануляване на оперативната памет	Резервиран – винаги е 0.	-
	1	(#) Изпълнението на командата не е позволено в текущия фискален режим	(#) Изпълнението на командата не е позволено в текущия фискален режим	(#) Изпълнението на командата не е позволено в текущия фискален режим
	0	При изпълнение на командата се е получило препълване на някои полета от сумите. Статус 1.1 също ще се установи и командата няма да предизвика промяна на данните в принтера	При изпълнение на командата се е получило препълване на някои полета от сумите. Статус 1.1 също ще се установи и командата няма да предизвика промяна на данните в ФУ	При изпълнение на командата се е получило препълване на някои полета от сумите

Статус байт 2

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
2	7	Резервиран – винаги е 1	Резервиран – винаги е 1.	Резервиран – винаги е **1**.
	6	Много близък край на КЛЕН (допускат се само определени бонове).	Не се използува	Резервиран – винаги е **0**.
	5	Отворен е служебен бон	Отворен е служебен бон	Отворен е служебен бон
	4	Близък край на КЛЕН (по-малко от 10 MB от КЛЕН свободни).	Близък край на КЛЕН (по-малко от 10 MB от КЛЕН свободни).	Близък край на КЛЕН (по-малко от 10 MB от КЛЕН свободни).
	3	Отворен е фискален бон	Отворен е фискален бон	Отворен е фискален бон
	2	Край на КЛЕН (по-малко от 1 MB от КЛЕН свободни)	Край на КЛЕН (по-малко от 1 MB от КЛЕН свободни).	Край на КЛЕН (по-малко от 1 MB от КЛЕН свободни).
	1	Останала е малко хартия	Резервиран – винаги е 0.	Останала е малко хартия
	0	(#) Свършила е хартията. Ако се вдигне този флаг по време на команда, свързана с печат, то командата е отхвърлена и не е променила състоянието на принтера	(#) Свършила е хартията. Ако се вдигне този флаг по време на команда, свързана с печат, то командата е отхвърлена и не е променила състоянието на ФУ.	(#) Свършила е хартията. Ако се вдигне този флаг по време на команда, свързана с печат, то командата е отхвърлена и не е променила състоянието на ФУ.

Статус байт 3

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
3	7	Резервиран – винаги е 1.	Резервиран – винаги е 1	Резервиран – винаги е 1
	6	Състояние на Sw7	Резервиран – винаги е 0	Резервиран – винаги е 0
	5	Състояние на Sw6	Резервиран – винаги е 0	Резервиран – винаги е 0
	4	Състояние на Sw5	Резервиран – винаги е 0	Резервиран – винаги е 0
	3	Състояние на Sw4	Резервиран – винаги е 0	Резервиран – винаги е 0
	2	Състояние на Sw3	Резервиран – винаги е 0	Резервиран – винаги е 0
	1	Състояние на Sw2	Резервиран – винаги е 0	Резервиран – винаги е 0
	0	Състояние на Sw1	Резервиран – винаги е 0	Резервиран – винаги е 0

Статус байт 4

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
4	7	Резервиран – винаги е 1.	Резервиран – винаги е 1	Резервиран – винаги е 1
	6	Печатащата глава е прегряла	Не се използува.	Фискалната памет липсва или е повредена
	5	OR на всички грешки, маркирани с ‘*’ от байтове 4 и 5	OR на всички грешки, маркирани с ‘*’ от байтове 4 и 5	OR на всички грешки, маркирани с ‘*’ от байтове 4 и 5
	4	(*) Фискалната памет е пълна	(*) Фискалната памет е пълна	(*) Фискалната памет е пълна
	3	Има място за по-малко от 50 записа във ФП	Има място за по-малко от 50 записа във ФП	Има място за по-малко от 50 записа във ФП
	2	Зададени са индивидуален номер на ФУ и номер на ФП	Зададени са индивидуален номер на ФУ и номер на ФП	Зададени са индивидуален номер на ФУ и номер на ФП
	1	Зададен е ЕИК	Зададен е ЕИК	Зададен е ЕИК
	0	(*) Има грешка при запис във фискалната памет	(*) Грешка при запис във фискалната памет	(*) Грешка при запис във фискалната памет

Статус байт 5

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
5	7	Резервиран – винаги е 1	Резервиран – винаги е 1	Резервиран – винаги е 1
	6	Не се използува	Резервиран – винаги е 0	Резервиран – винаги е 0
	5	Грешка при четене от фискалната памет	Резервиран – винаги е 0	Резервиран – винаги е 0
	4	Зададени са поне веднъж данъчните ставки	Зададени са поне веднъж данъчните ставки.	Зададени са поне веднъж данъчните ставки.
	3	Принтерът е във фискален режим	ФУ е във фискален режим.	ФУ е във фискален режим.
	2	(*) Последният запис във фискалната памет не е успешен	Резервиран – винаги е 0	Резервиран – винаги е 0
	1	Фискалната памет е форматирана	ФП е форматирана.	ФП е форматирана.
	0	(*) Фискалната памет е установена в режим READONLY (заключена).	Резервиран – винаги е 0	Резервиран – винаги е 0

Статус байт 6

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
6	7	✗	✗	Резервиран – винаги е 1
	6	✗	✗	Резервиран – винаги е 0
	5	✗	✗	Резервиран – винаги е 0
	4	✗	✗	Резервиран – винаги е 0
	3	✗	✗	Резервиран – винаги е 0
	2	✗	✗	Резервиран – винаги е 0
	1	✗	✗	Резервиран – винаги е 0
	0	✗	✗	Резервиран – винаги е 0

Статус байта е запазен за бъдеща употреба.

Статус байт 7

Байт индекс	Бит индекс	Значение в логическа група „A“	Значение в логическа група „B“	Значение в логическа група „C“
7	7	✗	✗	Резервиран – винаги е 1
	6	✗	✗	Резервиран – винаги е 0
	5	✗	✗	Резервиран – винаги е 0
	4	✗	✗	Резервиран – винаги е 0
	3	✗	✗	Резервиран – винаги е 0
	2	✗	✗	Резервиран – винаги е 0
	1	✗	✗	Резервиран – винаги е 0
	0	✗	✗	Резервиран – винаги е 0

Статус байта е запазен за бъдеща употреба.

Информация за устройството

Property name	Property type	Human meaning
connected_ToDevice	Boolean Read only	True if the connection to the fiscal device is successfully established.
support_RS232	Boolean Read only	True if the device supports communication through RS232 (or USB).
support_TCPIP	Boolean Read only	True if the device supports communication through LAN connector (TCP/IP protocol).
device_Type	TDeviceType Read only	• dt_FiscalPrinter for a fiscal printer; • dt_ECR for a cash register;
device_Number_Serial	WideString/BSTR Read only	Manufacturer number (serial number) of the device.
device_Number_FMemory	WideString/BSTR Read only	Fiscal number of the device.
device_Model	TDeviceModel Read only	• Device model – enumerated value; • mc_Unknown if the server can not recognise the connected device;
device_Model_Name	WideString/BSTR Read only	The model name received from the device.
codePage	Integer/Long Read only	The code page used from the device.
device_Firmware_Revision	WideString/BSTR Read only	Device firmware revision.
device_Firmware_Date	WideString/BSTR Read only	Device firmware date and time.
device_Firmware_CheckSum	WideString/BSTR Read only	Device firmware checksum.

Комуникация

Тази секция съдържа информация относно properties които са свързани с поведението на COM сървъра при опит за отваряне на връзка или по време на комуникация с фискалното устройство.

Property name	Property type	Human meaning
protocol_TransportType	TTransportProtocol Read only	The client application must set the type of the transport protocol with execution of method "set_TransportType" before opening of the connection to the fiscal device. • ctc_RS232 – if connection is via (RS-232/USB) • ctc_TCPIP – if connection is via LAN (TCP/IP)
tcpip_Address	WideString/BSTR Read only	Current IP address of the fiscal device.
tcpip_Port	Integer/Long Read only	Current value of the TCP/IP port from the device side.
rs232_ComPort	Integer/Long Read only	Current COM port value.
rs232_BaudRate	Integer/Long Read only	Current baud rate.
read_TimeOutValue	Word Read/Write	Global timeout value for communication with the device. Default values: • 1000 mSec for RS-232 • 3000 mSec for TCP/IP
exit_ByReadTimeOutIsOn	Boolean Read/Write	If this value is true and if there is no answer for period of time bigger than global timeout value in mSec - “CFD_BGR“ will stop waiting for a response from the fiscal device and will return a timeout error code for current command execution. Default value is true.
rs232_ReadIntervalTimeout	Integer/Long Read only	Part of commtimeouts structure. The maximum time allowed to elapse before the arrival of the next byte on the communications line, in milliseconds. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used. A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the bytes that have already been received, even if no bytes have been received.
rs232_ReadTotalTimeoutMultiplier	Integer/Long Read only	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.
rs232_ReadTotalTimeoutConstant	Integer/Long Read only	Part of commtimeouts structure. A constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes. A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.
rs232_WriteTotalTimeoutMultiplier	Integer/Long Read only	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the number of bytes to be written.
rs232_WriteTotalTimeoutConstant	Integer/Long Read only	Part of commtimeouts structure. A constant used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written. A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations.

rs232_OnOpen_Set_DCB	Boolean Read/Write	When this value is true – „CFD_BGR“ will try to set DCB structure during the opening of the COM port. Default value is true.
rs232_OnOpen_Set_DTR_ToFalse	Boolean Read/Write	When this value is true – „CFD_BGR“ will try to set DTR to false during the opening of the COM port. Clears the DTR (data-terminal-ready) signal. Default value is true.
rs232_OnOpen_Set_RTS_ToFalse	Boolean Read/Write	When this value is true – “CFD_BGR“ will try to set RTS to false during the opening of the COM port. Clears the RTS (request-to-send) signal. Default value is true.
connected_ToLAN	Boolean Read only	A return value of TRUE indicates that either the modem connection is active, or a LAN connection is active and a proxy is properly configured for the LAN. It does not guarantee that a connection to a specific host can be established. A return value of FALSE indicates that neither the modem nor the LAN are connected.

Общо поведение

Property name	Property type	Human meaning
language	TMyLanguages Read only	• Bulgarian • English
trackingMode	Boolean Read only	If the value is true – „CFD_BGR“ will try to save the communication with the fiscal device in a file according to the values of other tracking properties.
trackingMode_Path	WideString/BSTR Read only	The path to the log file when „CFD_BGR“ is in tracking mode.
trackingMode_FileName	WideString/BSTR Read only	If „CFD_BGR“ is in tracking mode – it will try to create and save the log info a text file with this name.
trackingMode_RowLimit	Integer/Long Read only	This value is a limit of the number of text rows into the log file. „CFD_BGR“ accept values between 100 and 5000.
active_OnBeforeScriptExecute	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event before the execution of the script.
active_OnScriptRowExecute	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after execution of the command from the text row into the script.
active_OnAfterScriptExecute	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after the execution of the script.
active_OnFirstProgress_Init	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event before starting an operation which is a loop of commands. It is usable for initialization of progress bar for example.
active_OnFirstProgress_Loop	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after the execution of a command during to the loop of commands.
active_OnFirstProgress_Complete	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after finishing an operation which is a loop of commands.
active_OnSecondProgress_Init	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event before starting an operation which is a loop of commands. It is usable for initialization of progress bar for example.
active_OnSecondProgress_Loop	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after the execution of a command during to the loop of commands.
active_OnSecondProgress_Complete	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after finishing an operation which is a loop of commands.
active_OnSendCommand	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after the sending of packet message to the fiscal device.
active_OnWait	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after receiving SYN byte from the fiscal device.
active_OnReceiveAnswer	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after receiving of packet message from the fiscal device.
active_OnStatusChange	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event after receiving a packet message from the fiscal device.
active_OnError	Boolean Read only	If this value is true – „CFD_BGR“ will fire an event if an error occurs.

Други

Property name	Property type	Human meaning
lastError_Code	Integer/Long Read only	Contains the current value of the last error code of the „CFD_BGR“.
lastError_Message	WideString/BSTR Read only	Contains the current value of the last error message of the „CFD_BGR“.
last_AnswerList	WideString/BSTR Read only	Contains the current value of the last answer from the packaged message received from the fiscal device. The values are separated with CRLF. (bytes with the decimal values 13 and 10. "Carriage Return" and "Line Feed").
download_Path	WideString/BSTR Read only	Contains the current value of the path for download which the „CFD_BGR“ driver will try to use in operations for downloading ANAF files from the fiscal device.
DateRange_StartValue	WideString/BSTR Read/Write	For future usages
DateRange_EndValue	WideString/BSTR Read/Write	For future usages

Събития (Events)

По дизайн събитията които интерфейс „CFD_BGR“ вдига са предназначени за чисто информативни цели. За удобство по отношение на потребителския интерфейс. По време на изпълнение на команда може да бъдат вдигнати повече от едно събития. Не използвайте кодовете за грешка идващи чрез събития за управление на изпълнението на командите! Възможно е да дойдат асинхронно във времето.

OnError

„CFD_BGR“ ще вдигне това събитие при възникване на грешка.

Забележка: Ако изпълнението на командата е неуспешно и метода който използвате връща резултат от тип integer – стойността на "**error_Code**" най-вероятно ще бъде идентична със стойността на резултата от изпълнението на метода. ***Ако изпълнението е успешно – стойността на резултата от метода и стойността на „last error code” ще бъдат равни на нула, а събитието „OnError“ няма да бъде вдигнато.***

Параметри:

Name	Type	Description
error_Code	Integer	The value of the error code depending of the error.
error_Message	WideString	The text error message.

OnBeforeScriptExecute event

„CFD_BGR“ ще вдигне това събитие преди изпълнение на скрипт.
Параметри: **Няма.**

OnScriptRowExecute event

„CFD_BGR“ ще вдигне това събитие след изпълнението на команда от текстова линия в даден скрипт.

Параметри:

Name	Type	Description
row_Index	Integer	The index shows the line number depending on the beginning of the script. The index is zero based.
error_Code	Integer	The value of the error code depending of the error. An error code with value 0 means that there no error during the execution of the command.
input_Value	WideString	The value of the executed text line.
output_Value	WideString	The answer from the device side.

OnAfterScriptExecute event

„CFD_BGR“ ще вдигне това събитие след изпълнението на скрипт.

Параметри: **None**

OnSendCommand event

„CFD_BGR“ ще вдигне това събитие след изпращане на пакетирано съобщение към фискалното устройство.

Параметри:

Name	Type	Description
Command	WideString	The value of the command in decimal.
DateAndTime	WideString	Date and time of execution in format 'dd.mm.yyyy hh:mm:ss:zzz'
repeat_Value	WideString	By the low level communication protocol – if command failed or if fiscal device asked the host to repeat the command, the host must send the same packet message to the fiscal device. The value of this parameter contains the current index of the attempts.
hex_Header	WideString	The header part from the packet message to the fiscal device in hex values.
hex_Data	WideString	The logical data part from the packet message to the fiscal device in hex values.
hex_Footer	WideString	The footer part from the packet message to the fiscal device in hex values.
human_Data	WideString	The logical data part from the packet message to the fiscal device before to convert into a hex values.

OnWait event

„CFD_BGR“ ще вдигне това събитие след получаване на байт със стойност SYN от страна на фискалното устройство.

Параметри:

Name	Type	Description
Value	Byte	Must contain value equal to SYN

OnReceiveAnswer event

„CFD_BGR“ ще вдигне това събитие след получаване на пакетирано съобщение от страна на фискалното устройство.

Параметри:

Name	Type	Description
Command	WideString	The value of the command in decimal.
DateAndTime	WideString	Date and time of execution in format 'dd.mm.yyyy hh:mm:ss:zzz'
repeat_Value	WideString	By the low level communication protocol – if command failed or if fiscal device asked the host to repeat the command, the host must send the same packet message to the fiscal device. The value of this parameter contains the current index of the tries.
hex_Header	WideString	The header part from the packet message from the fiscal device in hex values.
hex_Data	WideString	The logical data part from the packet message from the fiscal device in hex values.
hex_Footer	WideString	The footer part from the packet message from the fiscal device in hex values.
human_Data	WideString	The logical data part from the packet message from the fiscal device as a human text.

OnStatusChange event

„CFD_BGR“ ще вдигне това събитие след получаване на пакетирано съобщение от страна на фискалното устройство. Това събитие е използваемо като тригер. След вдигане на събитието, клиентската програма може да прочете статуса на фискалното устройство от статус битовете (или от статус properties описани по-горе). Това е възможно, защото всички те са вече обновени вътрешно.

Параметри: **Няма**

OnFirstProgress_Init event

„CFD_BGR“ ще вдигне това събитие преди стартиране на операция която се състои от цикъл команди. Удобна е за инициализация на някакъв прогрес (progress bar).

Параметри:

Name	Type	Description
value_Minimum	Integer	The calculated minimum value for the initialization.
value_Maximum	Integer	The calculated maximum value for the initialization.
value_Position	Integer	Current position for the initialization.

OnFirstProgress_Loop event

„CFD_BGR“ ще вдигне това събитие след изпълнението на команда в тялото на вътрешен за библиотеката цикъл.

Параметри:

Name	Type	Description
value_Position	Integer	Current position of the progress.

OnFirstProgress_Complete event

„CFD_BGR“ ще вдигне това събитие след финализиране на вътрешен за библиотеката цикъл.

Параметри: **Няма**

OnSecondProgress_Init event

„CFD_BGR“ ще вдигне това събитие преди стартиране на операция която се състои от цикъл команди. Удобна е за инициализация на някакъв прогрес (progress bar).

Параметри:

Name	Type	Description
value_Minimum	Integer	The calculated minimum value for the initialization.
value_Maximum	Integer	The calculated maximum value for the initialization.
value_Position	Integer	Current position for the initialization.

OnSecondProgress_Loop event

„CFD_BGR“ ще вдигне това събитие след изпълнението на команда в тялото на вътрешен за библиотеката цикъл.

Параметри:

Name	Type	Description
value_Position	Integer	Current position of the progress.

OnSecondProgress_Complete event

„CFD_BGR“ ще вдигне това събитие след финализиране на вътрешен за библиотеката цикъл.

Параметри**: Няма**

Методи

Режим проследяване / Tracking mode

За нуждите на разработките „CFD_BGR“ поддържа вътрешен механизъм за проследяване и записване на комуникацията с фискалното устройство. Ако разработчиците искат да проследят в подробности комуникацията между Host (PC) и Slave (фискалното устройство), те могат да активират този режим. „CFD_BGR“ ще записва комуникацията в двете посоки в даден файл. Ако има съмнение за грешка или проблем – екипа по поддръжката на библиотеката вероятно ще поиска да активирате този режим, да повторите стъпка по стъпка действията довели до проблема и да изпратите лог файла за анализ. Разбира се – добре е преди това самите разработчици да го разгледат – много често проблема се вижда веднага.

set_TrackingMode_RowLimit

Изпълнението на този метод ще настрои горната граница на броя текстови редове които ще се записват в лог файла. „CFD_BGR“ приема стойности между 100 и 5000.

Параметри:

Name	Type	Description
Value	Integer Long	The limit of the number of text rows into the log file. „CFD_BGR“ accept values between 100 and 5000.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_TrackingMode_Path

Изпълнението на този метод ще настрои пътя (папката) където лог файла ще се записва когато „CFD_BGR“ е в режим проследяване.

Параметри:

Name	Type	Description
Value	WideString BSTR	The path to the log file when „CFD_BGR“ is in tracking mode
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_TrackingMode_FileName

Изпълнението на този метод ще настрои името на лог файла който „CFD_BGR“ ще се опита да създаде и използва в режим проследяване.

Параметри:

Name	Type	Description
Value	WideString BSTR	The log file name
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_TrackingMode

Изпълнението на този метод ще активира или деактивира режим проследяване. При стойност true – „CFD_BGR“ ще опитва да записва комуникацията с фискалното устройство във файл и папка съобразно стойностите на описаните по-горе tracking properties.

Параметри:

Name	Type	Description
Value	Boolean	If the value is true, the tracking mode will be activated.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

Комуникация / Communication

set_TransportType

Execution of this function will set the transport protocol type. The client application must set the type of the communication transport protocol before opening of the connection to the fiscal device.

Parameters:

Name	Type	Description
Value	TTransportProtocol	The client application must set the type of the transport protocol before opening the connection to the fiscal device. • ctc_RS232 – if connection is via (RS-232/USB) • ctc_TCPIP – if connection is via LAN (TCP/IP) For current value – check the value of the corresponding property: "protocol_TransportType".
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the “CFD_BGR“ engine.

set_TCPIP

Execution of this function will set the values needed for successful execution of the method "open_Connection" if the chosen protocol type is ctc_TCPIP. The client application must set the type of the communication transport protocol, the IPAddress and the Port before trying to open the connection to the fiscal device.

Parameters:

Name	Type	Description
IPAddress	WideString BSTR	The IP Address of the fiscal device.
Port	Integer / Long	The Port number for TCP/IP communications of the fiscal device.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_RS232

Execution of this function will set the values needed for successful execution of the method "open_Connection" if the chosen protocol type is ctc_RS232. The client application must set the type of the communication transport protocol, the ComPort and the BaudRate before trying to open the connection to the fiscal device.

Parameters:

Name	Type	Description
ComPort	Integer / Long	The ComPort.
BaudRate	Integer / Long	The baud rate – must be the same as into the fiscal device.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_RS232_Timeouts

Execution of this function will set the values of the commtimeouts structure. Do not use it if you don't know how or if „CFD_BGR“ is working properly with the default values.

Parameters:

Name	Type	Description
ReadIntervalTimeout	LongWord	Part of commtimeouts structure. The maximum time allowed to elapse before the arrival of the next byte on the communications line, in milliseconds. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used. A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the bytes that have already been received, even if no bytes have been received.
ReadTotalTimeoutMultiplier	LongWord	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.
ReadTotalTimeoutConstant	LongWord	Part of commtimeouts structure. A constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes. A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.
WriteTotalTimeoutMultiplier	LongWord	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the number of bytes to be written.
WriteTotalTimeoutConstant	LongWord	Part of commtimeouts structure. A constant used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written. A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the “CFD_BGR“ engine.

rs232_COMPortList

This method returns a list of visible for “CFD_BGR“ engine COM ports. If your COM port is not here – most probably the „CFD_BGR“ engine isn‘t using it properly. After the adding of new USB device or hardware (new COM port) – this method must be executed again to refresh the info of the client application.

Parameters:

Name	Type	Description
Result	WideString BSTR	This method returns a list of visible for „CFD_BGR“ engine COM ports.

open_Connection

This function opens a connection to the fiscal device according to the other communication properties set before the execution of the command.

Parameters:

Name	Type	Description
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

close_Connection

This function closes the connection to the fiscal device.

Parameters:

Name	Type	Description
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

Error support

You can find a full list of the possible errors returned from the “CFD_BGR“ engine or from the fiscal device in the document: "ErrorCodes.xls". This document is located in the folder DOCUMENTATION, which is a sub-folder of the root of the installation folder.

get_ErrorMessageByCode

The client applications can get the error message by given code.

Parameters and result:

Name	Type	Description
Value	Integer/Long	The value of the code.
Result	WideString BSTR	The error message for a given code.

get_Sbit_State

The client applications can get the state of any given status bit after each answer of the fiscal device.

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Result	Boolean	The result = true – mean that the status bit is raised. *Example:* *if get_SBit_State(0,0) then ShowMessage('Syntax error!');*

get_Sbit_Description

The client applications can get the description text for any given status bit.

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Result	WideString BSTR	The description text for the given status bit. *Example: ShowMessage(get_SBit_Description(0,0));*

get_Sbit_ErrorChecking

The client applications can get the description text for any given status bit.

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Result	Boolean	If the result of the function is true - this means that the „CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

set_Sbit_ErrorChecking

The client application can change behavior of the „CFD_BGR“ engine according to the needs of the project. If you think that the raising of some status bit is not an error – you can switch off its checking.

Note: Change of the checking mean that only the “CFD_BGR“ engine will stop checking given status. The fiscal device will continue raising the given status bit and will continue to return errors less than zero if command is not apropriate or if there is a syntax error, for example. ***This function is developed for the needs of service applications and We do not recomend you to use it if you are not sure what we are talking about.*** *One more time: If the fiscal device "thinks" that given command is an error – it will not execute the command.*

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Value	Boolean	True or False if you want to switch on or switch off the checking of given status bit.
Result	Boolean	If the result of the function is true - this mean that the „CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

Behavior

The client applications can change the behavior of the „CFD_BGR“ in some aspects. For example they can activate or deactivate the raising of the events in the different cases.

set_ScriptEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnBeforeScriptExecute**;
- **OnScriptRowExecute**;
- **OnAfterScriptExecute**;

Parameters and result:

Name	Type	Description
active_OnBeforeScriptExecute	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnScriptRowExecute	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnAfterScriptExecute	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the „CFD_BGR“ engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_FirstProgressEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnFirstProgress_Init**;
- **OnFirstProgress_Loop**;
- **OnFirstProgress_Complete**;

Parameters and result:

Name	Type	Description
active_OnFirstProgress_Init	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnFirstProgress_Loop	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnFirstProgress_Complete	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the „CFD_BGR“ engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_SecondProgressEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnSecondProgress_Init**;
- **OnSecondProgress_Loop**;
- **OnSecondProgress_Complete**;

Parameters and result:

Name	Type	Description
active_OnSecondProgress_Init	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnSecondProgress_Loop	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnSecondProgress_Complete	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the „CFD_BGR“ engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

set_CommunicationEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnSendCommand**;
- **OnWait**;
- **OnReceiveAnswer**;
- **OnStatusChange**;
- **OnError**;

Parameters and result:

Name	Type	Description
active_OnSendCommand	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnWait	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnReceiveAnswer	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnStatusChange	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnError	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the „CFD_BGR“ engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the „CFD_BGR“ engine.

Services

cancel_Loop

With execution of this method – you can tell the „CFD_BGR“ engine to stop the loop process and after that to exit the current execution of the method.

Parameters: **None**

Result: **None**

Note: For fufure usages

upload_Logo

This method sends a logo file to the fiscal device. The file must be with proper size, monochrome and with “bmp” extension. The fiscal devices not accept images different than bmp.

During the execution of the methods „CFD_BGR“ will fire more than one time many of the events, so if the uploading of the logo files from client application must be accelerated as a process – the application can deactivate some of the events in the beginning. After the execution of the method the application can switch them back on to an active state.

Parameters:

Name	Type	Description
FileName	WideString BSTR	The path and file name to the logo file.
Result	WideString BSTR	A system information according to the input search value.

Note: For fufure usages

SystemInfo

get_SystemInfoSearchList

This method returns a list of system values which can be used as a search parameter into the method "**get_SystemInfo**". These two methods are suggested for the convenience of programmers.

Parameters:

Name	Type	Description
Result	WideString BSTR	A list of system values which can be used as a search parameter with the method "get_SystemInfo".

get_SystemInfo

The programmers can use this method as it is convenient. It shows information about the current PC configuration, folders and so on in the way that the “CFD_BGR“ sees the system.

Parameters:

Name	Type	Description
SearchValue	WideString BSTR	One of the items returned from the method: “**get_SystemInfoSearchList**”.
Result	WideString BSTR	A system information according to the input search value.

Общи методи

execute_Command

Common method for sending commands to the fiscal device. The execution of the command can fire one or more of the following events if they are not switched off programmatically:

- OnSendCommand;
- OnWait;
- OnReceiveAnswer;
- OnStatusChange;
- OnError;

Parameters and result:

Name	Type	Description
Command	Integer/Long	The value of the command in the decimal number system.
input_Value	WideString BSTR	The logical data of the command – formatted according to the protocol of the fiscal devices. You can find a detailed description of the commands to the fiscal device in the documents: • In English: FP_Protocol_EN.pdf • In Bulgarian: FP_Protocol_BG.pdf These documents are located in the folder DOCUMENTATION which is a sub-folder of the root of the installation folder.
output_Value	WideString BSTR	Text which contains the logical data from the answer from the fiscal device. The answer is formatted according to the protocol of the fiscal devices. You can parse the answer with your own parser or use the parsed answer from the property "last_AnswerList".
Result	Boolean	If the result of the function is true - this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

execute_Script_V1

Common method for sending a set of commands to the fiscal device. The client application can execute pre-prepared text scripts through this method. The execution of the command can fire one or more of the following events if they are not switched off programmatically:

- OnBeforeScriptExecute;
- OnScriptRowExecute;
- OnAfterScriptExecute;
- OnSendCommand;
- OnWait;
- OnReceiveAnswer;
- OnStatusChange;
- OnError;

Parameters and result:

Name	Type	Description
scriptType	TScriptType	This value is used from the script engine. "CFD_BGR" must know the language of the incoming script. Possible values: • DS
input_Value	WideString BSTR	The script text which contains the complette set of needed commands. Format of the text row: <***cmd***>,<***logical data for the command***> • cmd – the value of the command in the decimal number system; • logical data of the command – formatted according to the protocol of the fiscal devices. You can find a detailed description of the commands to the fiscal device in the documents: • In English: FP_Protocol_EN.pdf • In Bulgarian: FP_Protocol_BG.pdf These documents are located in the folder DOCUMENTATION which is a sub-folder of the root of the installation folder.
Result	Boolean	If the result of the function is true - this mean that the "CFD_BGR" engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_ComandsList

The “CFD_BGR“ engine contains an internal description of the commands for the given fiscal device as a collection of objects and the client applications can use them. This method returns a list with “human names” of the commands from the protocol of the fiscal device. In this version of the “CFD_BGR“ engine – the programmers can choose from three different names for each of the commands and use them according to their own preferences.

Parameters:

Name	Type	Description
Index	Integer/Long	A zero based index for the internal search machine. Use the method with values of index between 0 and 2.
Result	WideString BSTR	This method returns a list of command names accordingly to the used index value.

Format description of the “human names”:

- index [0]: <**cmd**>_<**group_name**>_<**name**>. *Example: 038_receipt_NonFiscal_Open;*
- index [1]: <**group_name**>_<**name**>. *Example: receipt_NonFiscal_Open;*
- index [2]: <**name**>. *Example: NonFiscal_Open;*

In some cases the “human names” for index 1 and 2 can be identical. You can easily find the detailed description for a given command in the documentation of the low level protocol by searching for a given **cmd** value and, of course, the **name** value.

get_CommandInfo

By using this method the client applications can get from the “CFD_BGR“ engine some general information about the command and its parameters. Look at the source code of the demo programs for a demonstration of the method. This method is developed for the programmers as a fast way of getting information about the command and its input or output parameters. Of course – all of these parameters are described in the low level protocol documentation but here you can found information on the limit values of the parameters used by the “CFD_BGR“ engine.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	The one of the "human names" of the given command exported from the method “get_CommandsList”.
Value	WideString BSTR	Text with a common information of the command and its input or output parameters.
Result	Integer/Long	If the result of the function is true – this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_InputParams_Count

Through this method – the programmers can receive the count of the input parameters for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Value	Integer/Long	The count of the input parameters for a given command.
Result	Integer/Long	If the result of the function is true - this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_InputParams_Names

Through this method – the programmers can receive the list of the input parameter names for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Value	WideString BSTR	The list of input parameter names accordingly to the used “**command_Name**” value.
Result	Integer/Long	If the result of the function is true - this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

set_InputParam_ByIndex

If you want to use the method "**execute_Command_ByName**" you must ensure that all input parameters of the command are set properly before the execution of the command. One of the ways to do that is to use this method.

Notes:

- *In some cases the value of the input parameter must be empty. For example, look at command 255 in the case of reading of the value from fiscal device.*
- *In other cases the input parameter is optional – so depending on the case, the input parameter may or may not need to be empty.*
- ***You must be sure that you set the values of all input parameters properly. The "CFD_BGR" engine keeps the input parameters into the memory so in some cases you probably do not want to execute the method with no set new values (using the old values).***

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Index	Integer/Long	A zero based index related to the input parameters list of this command.
Value	WideString BSTR	The value of the input parameter. Check the low level protocol to be sure that you will fill the value of this parameter properly.
Result	Integer/Long	If the result of the function is true - this mean that the "CFD_BGR" engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

set_InputParam_ByName

If you want to use the method "**execute_Command_ByName**" you must ensure that all input parameters of the command are set properly before the execution of the command. One of the ways to do that is to use this method.

Notes:

- *In some cases the value of the input parameter must be empty. For example look at command 255 in the case of reading of the value from fiscal device.*
- *In other cases the input parameter is optional – so according to the case, the input parameter may or may not need to be empty.*
- ***You must be sure that you set the values of all input parameters properly. The "CFD_BGR" engine keeps the input parameters in the memory so in some cases you probably do not want to execute the method with no set new values (using the old values).***

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Name	WideString BSTR	The input parameter name.
Value	WideString BSTR	The value of the input parameter. Check the low level protocol to be sure that you will fill the value of this parameter properly.
Result	Integer/Long	If the result of the function is true - this mean that the "CFD_BGR" engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

execute_Command_ByName

The "CFD_BGR" engine contains an internal description of the commands for the given fiscal device as a collection of objects and the client applications can use them. Set the input parameters of the command properly – before the execution of the method.

The execution of the command can fire one or more of the following events if they are not switched off programmatically:

- OnSendCommand;
- OnWait;
- OnReceiveAnswer;
- OnStatusChange;
- OnError*;*

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Result	Integer/Long	If the result of the function is true - this mean that the "CFD_BGR" engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

After the successful execution of the method – you can receive the output parameters by using one of the following methods:

- get_OutputParam_ByIndex;
- get_OutputParam_ByName;
- use the values from last_AnswerList;

Note: Any of the commands exported into the "human oriented" command names list can be executed with your own input text via method "execute_Command". You can execute each of the commands from the low level protocol of the fiscal device via method "execute_Command". During the development of the client application ***you have more than one way to execute the commands*** *if:*

- *something goes wrong;*
- *something is not clear;*
- *the command is not exported for the execution by name;*

get_OutputParams_Count

Through this method – the programmers can receive the count of the output parameters for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Value	Integer/Long	The count of the output parameters for a given command.
Result	Integer/Long	If the result of the function is true - this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_OutputParams_Names

Through this method – the programmers can receive the list of the output parameter names for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	The one of the "human names" of the given command exported from the method “**get_CommandsList**”.
Value	WideString BSTR	The list of output parameter names accordingly to the used “**command_Name**” value.
Result	Integer/Long	If the result of the function is true - this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_OutputParam_ByIndex

After the successful execution of the method “**execute_Command_ByName**” – you can receive the output parameters by using this method.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Index	Integer/Long	A zero based index related to the output parameters list of this command.
param_Value	WideString BSTR	The value of the output parameter.
Result	Integer/Long	If the result of the function is true – this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_OutputParam_ByName

After the successful execution of the method “**execute_Command_ByName**” – you can receive the output parameters by using this method.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Name	WideString BSTR	The output parameter name.
param_Value	WideString BSTR	The value of the output parameter.
Result	Integer/Long	If the result of the function is true – this mean that the “CFD_BGR“ engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command

		execution.

generate_SourceCode

In this version of the "CFD_BGR" engine – the programmers can use this method to generate a source code which gives an example of the usage of method "**execute_Command_ByName**". The source code can be used and modified as is comfortable for the programmer and without any license restrictions. In simple terms – it is free as a beer. Of course, this source code is a generated from our AI example so if you choose to use it – please ensure that it is implemented and works well on a real fiscal device.

Currently the "CFD_BGR" engine can generate a source code on two languages for all commands which are exported from the method "**get_ComandsList**".

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
code_Type	TCodeType	Possible parameter values: • Delphi; • CSharp;
Result	Integer/Long	If the result of the function is true – this mean that the "CFD_BGR" engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

***Note**: In this version of "CFD_BGR" – the driver generate source code only on Pascal (Delphi) and C#. If you find a bug it would be nice to send an email to the authors of the COM server. We would be glad to fix or resolve the issue in the next version of the engine.*

Машинно-независими методи (device-independent methods)

Отваряне на фискален бон

За да се направи отварянето на фискален бон независимо спрямо изискванията на различните устройства – в интерфейс „CFD_BGR“ са добавени *(Read/Write)* properties които трябва да бъдат попълнени преди извикването на самия метод.

В следващата таблица са изброени поретата (properties) и техните характеристики:

<table>
<tr><th>Име</th><th>Пояснение</th><th>Задължително поле в логическа група „A“</th><th>Задължително поле в логическа група „B“</th><th>Задължително поле в логическа група „C“</th></tr>
<tr><td>Operator_Code</td><td>Код на оператор.
<table>
<tr><th>Логическа група</th><th>Стойност</th><th>По подразбиране</th></tr>
<tr><td>A</td><td>от 1 до 16</td><td>1</td></tr>
<tr><td>B</td><td>от 1 до 30</td><td>1</td></tr>
<tr><td>C</td><td>от 1 до 30</td><td>1</td></tr>
</table></td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>Operator_Password</td><td>Операторска парола:
<table>
<tr><th>Логическа група</th><th>Стойност</th><th>По подразбиране</th></tr>
<tr><td>A</td><td>4 до 8 цифри</td><td>0000</td></tr>
<tr><td>B</td><td>1 до 8 цифри</td><td>1</td></tr>
<tr><td>C</td><td>1 до 8 цифри</td><td>1</td></tr>
</table></td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>Till_Number</td><td>Номер на касово място. /цяло число от 1 до 99999/
Ако клиентския софтуер не зададе стойност, сървъра ще се опита да отвори сторно бон със стойност 1.</td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>FR_UNP</td><td>УНП на фискалния документ. (Уникален номер продажба)</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>FR_Invoice</td><td>Определя дали ще се отваря фискален бон или разширена клиентска бележка (фактура).
<table>
<tr><th>Стойност</th><th>Значение</th></tr>
<tr><td>0</td><td>Фискален бон</td></tr>
<tr><td>1</td><td>Разширена клиентска бележка (фактура).</td></tr>
</table></td><td>✗</td><td>✗</td><td>✗</td></tr>
</table>

Полетата оцветени в зелено са задължителни. Фискалните устройства позволяват да бъде отварян фискален бон без подаване на УНП, заради съвместимост със стар софтуер, но за всеки софтуер деклариран като отговарящ на наредба №18 полето е задължително.

Имайте предвид, че отварянето на фактура е възможно само и единствено, ако във фискалното устройство е зададен допустим интервал от фактури и брояча на фактурите не е достигнал края му.

init_FiscalReceiptValues

Метода инициализира всички properties необходими за отварянето на фискален бон (или фактура). Където това е възможно – метода попълва стойности по подразбиране.

- ✔ Поле „***can_OpenFiscalReceipt***“ става със стойност True;
- ✔ Поле „***can_OpenInvoiceReceipt***“ става със стойност False.

Този метод се изпълнява автоматично – веднага след успешно отваряне на фискален бон (или фактура), така че стойностите да са инициализирани за следващата употреба, но клиентското приложение може да го вика и самостоятелно – преди попълването на правилните за случая стойности.

can_OpenFiscalReceipt

Това поле е само за четене. Ако „***can_OpenFiscalReceipt***“ е със стойност **False** – това означава, че не са попълнени необходимите за отваряне на фискален бон стойности или че има отворен документ и до неговото затваряне не може да се отвори нов.

Пример за употреба - клиентското приложение може да отвори фискален бон, ако е попълнило стойности в полета:

- ✔ Operator_Code;
- ✔ Operator_Password;
- ✔ Till_Number;
- ✔ FR_UNP;

Погледнете таблицата с необходими за отваряне на фискален бон полета по-горе. Общото правило е че ако сте попълнили поне едно поле от даден цвят – трябва да попълните и останалите полета от същия цвят, за да стане пропърти „***can_OpenFiscalReceipt***“ със стойност **True** и съответно метод „***open_FiscalReceipt***“ да може да бъде изпълнен.

При попълване на горните полета, сървъра проверява дали в съответните полета въобще е попълнено нещо и само в някои случаи може да провери дали стойността е правилна.

Пропърти „***can_OpenFiscalReceipt***“ със стойност ***True*** не означава, че отварянето на фискален бон ще премине успешно, а че вече сте попълнили достатъчно на брой полета за опит за отваряне чрез съответната комбинация от задължителни и опционални параметри.

can_OpenInvoiceReceipt

Това поле е само за четене. Ако „***can_OpenInvoiceReceipt***“ е със стойност **False** – това означава, че не са попълнени необходимите за отваряне на фискален бон (тип фактура) стойности или че има отворен документ и до неговото затваряне не може да се отвори нов.

Пример за употреба - клиентското приложение може да отвори фискален бон, ако е

попълнило стойности в полета:

- ✔ Operator_Code;
- ✔ Operator_Password;
- ✔ Till_Number;
- ✔ FR_UNP;
- ✔ FR_Invoice (със стойност 1)

При попълване на горните полета, сървъра проверява дали в съответните полета въобще е попълнено нещо и само в някои случаи може да провери дали стойността е правилна.

Пропърти „***can_OpenInvoiceReceipt***“ със стойност ***True*** не означава, че отварянето на фискален бон (тип фактура) ще премине успешно, а че вече сте попълнили достатъчно на брой полета за опит за отваряне чрез съответната комбинация от задължителни и опционални параметри. Честа срещана грешка – всички полета са попълнени правилно, но не е зададен интервал от номера за фактура (команда 66) или той е изчерпан.

open_FiscalReceipt и open_InvoiceReceipt

Ако „***can_OpenFiscalReceipt***“ е със стойност ***True*** и стойностите са правилни – COM сървъра ще може да изпълни метод „**open_FiscalReceipt**“. Ако „***can_OpenInvoiceReceipt***“ е със стойност ***True*** и стойностите са правилни – COM сървъра ще може да изпълни метод „**open_InvoiceReceipt**“.

И в двата случая – отварянето на фискален бон все още зависи от фискалното устройство. Например ако в момента няма хартия – метода ще върне статус бит грешка, а съответния статус бит ще бъде вдигнат след отговора от страна на устройството.

След успешно отваряне на фискален бон – всички properties имащи отношение към метода се инициализират вътрешно. Не трябва да забравяте, че трябва да попълвате правилните стойности преди изпълнението на методите.

За логически групи „А“ и „Б“ - метода попълва (read only) properties:

- ✔ ***AllReceipt_Count*** - Броят на всички издадени бонове (фискални и служебни) от последното приключване на деня до момента;
- ✔ ***FiscalReceipt_Count*** - Броят на всички издадени фискални бонове от последното приключване на деня до момента;

За логическа група „С“ - метода попълва стойността на поле **GlobalCounter**;

При успешно изпълнение и отваряне на фискален бон – резултата от изпълнението на метода е със стойност **0**.

При неуспешно изпълнение – резултата съвпада по стойност със стойността на property “***lastError_Code***”.

При отваряне на фискална бележка от тип фактура трябва да се знае, че ако поради някаква причина се наложи да я анулирате (команда 60), при следващото отваряне на фактура устройствата от тип „А“ ще използват същия номер. При устройства от логическа група „Б“ и „С“ отварянето на фактура изхабява текущия номер и при следващо отваряне ще се използва

следващия номер от интервала.

По-долу се вижда какъв документ се отваря при съответната комбинация от въведени полета, както и коя от device-dependent командите се изпълнява вътрешно от страна на COM сървъра.

ID	Operator_Code	Operator_Password	Till_Number	FR_UNP	FR_Invoice	Име на команда	can_OpenFiscalReceipt	can_OpenInvoiceReceipt	Тип документ
A01	✔	✔	✔	✗	**0**	048_receipt_FiscalOpen_A01	✔	✗	Касов бон
A02	✔	✔	✔	✗	**1**	048_receipt_FiscalOpen_A02	✗	✔	Фактура
A03	✔	✔	✔	✔	**0**	048_receipt_FiscalOpen_A03	✔	✗	Касов бон с УНП
A04	✔	✔	✔	✔	**1**	048_receipt_FiscalOpen_A04	✗	✔	Фактура с УНП
B01	✔	✔	✔	✗	**0**	048_receipt_FiscalOpen_B01	✔	✗	Касов бон
B02	✔	✔	✔	✗	**1**	048_receipt_FiscalOpen_B02	✗	✔	Фактура
B03	✔	✔	✔	✔	**0**	048_receipt_FiscalOpen_B03	✔	✗	Касов бон с УНП
B04	✔	✔	✔	✔	**1**	048_receipt_FiscalOpen_B04	✗	✔	Фактура с УНП
C01	✔	✔	✔	✗	**0**	048_receipt_FiscalOpen_C01	✔	✗	Касов бон
C02	✔	✔	✔	✗		048_receipt_FiscalOpen_C02	✗	✔	Фактура
C03	✔	✔	✔	✔		048_receipt_FiscalOpen_C03	✔	✗	Касов бон с УНП
C04	✔	✔	✔	✔		048_receipt_FiscalOpen_C04	✗	✔	Фактура с УНП

Софтуерните продукти съобразени с промените в наредба №18 трябва да издават фискални документи с УНП. Останалите методи са оставени за съвместимост или при употреба на фискални устройства от логическа група „А“. В случая на фискални устройства от логическа група „А“ имаме характерна особеност. Ако поне веднъж е издаден фискален документ с УНП – фискалното устройство го запомня и ако използвате команда за отваряне на фискална бележка без да сте попълнили УНП – фискалното устройство ще генерира само следващ УНП номер. За генериране на този УНП, фискалното устройство използва собствения си сериен номер, номера на оператора от поле „**Operator_Code**“ и стойността на брояча от УНП, увеличена с единица.

Отваряне на сторно бон

Както бе обяснено по-горе, фискалните устройства, предлагани от Датекс могат да се разделят на три логически групи. За да се направи отварянето на сторно бон независимо спрямо логиката и изискванията на различните устройства – в интерфейс „CFD_BGR“ са добавени *(Read/Write)* properties които трябва да бъдат попълнени преди извикването на самия метод.

В следващата таблица са изброени наличните properties и техните характеристики:

<table>
<tr><th>Име</th><th>Пояснение</th><th>Задължително поле в логическа група „A“</th><th>Задължително поле в логическа група „B“</th><th>Задължително поле в логическа група „C“</th></tr>
<tr><td>Operator_Code</td><td>Код на оператор.<table><tr><th>Логическа група</th><th>Стойност</th><th>По подразбиране</th></tr><tr><td>A</td><td>от 1 до 16</td><td>1</td></tr><tr><td>B</td><td>от 1 до 30</td><td>1</td></tr><tr><td>C</td><td>от 1 до 30</td><td>1</td></tr></table></td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>Operator_Password</td><td>Операторска парола:<table><tr><th>Логическа група</th><th>Стойност</th><th>По подразбиране</th></tr><tr><td>A</td><td>4 до 8 цифри</td><td>0000</td></tr><tr><td>B</td><td>1 до 8 цифри</td><td>1</td></tr><tr><td>C</td><td>1 до 8 цифри</td><td>1</td></tr></table></td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>Till_Number</td><td>Номер на касово място. /цяло число от 1 до 99999/
Ако клиентския софтуер не зададе стойност, сървъра ще се опита да отвори сторно бон със стойност 1.</td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>St_Reason_Type</td><td>Сървъра очаква цяло число със стойност от 0 до 2.<table><tr><th>Стойност</th><th>Значение</th></tr><tr><td>0</td><td>Операторска грешка (по подразбиране)</td></tr><tr><td>1</td><td>Връщане / Замяна / Рекламация</td></tr><tr><td>2</td><td>Намаление на данъчната основа</td></tr></table></td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>St_Doc_Number</td><td>Номер на документа (глобален) който се сторнира.
(цяло положително число от 1 до 9999999)</td><td>✔</td><td>✔</td><td>✔</td></tr>
<tr><td>St_Doc_DateTime</td><td>Дата и час на сторнирания документ.
Формат „DDMMYYhhmmss“.</td><td>✗</td><td>✔</td><td>✔</td></tr>
<tr><td>St_FM_Number</td><td>Номер на фискалната памет на ФУ от което е издаден бона, който се сторнира.</td><td>✗</td><td>✔</td><td>✔</td></tr>
<tr><td>St_Doc_UNP</td><td>УНП на документа, който се сторнира.</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>St_ByInvoice</td><td><table><tr><th>Стойност</th><th>Значение</th></tr><tr><td>0</td><td>Сторниране по фискален бон (по подразбиране)</td></tr><tr><td>1</td><td>Сторниране по фактура</td></tr></table></td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>St_InvoiceNumber</td><td>Номер на фактурата която се сторнира. (1 - 9999999999)</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>St_Reason</td><td><table><tr><th>Логическа група</th><th>Стойност до...</th></tr><tr><td>A</td><td>Причина за сторниране до 30 символа</td></tr><tr><td>B</td><td>Причина за сторниране до 42 символа</td></tr><tr><td>C</td><td>Причина за сторниране до 42 символа</td></tr></table></td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>St_Current_UNP</td><td>УНП на самия сторно бон. (Само в логическа група A)</td><td>✗</td><td>-</td><td>-</td></tr>
</table>

Полетата, оцветени в зелено са минимално изискуемите за съответната логическа група устройства, но имайте предвид, че употребата само на минимално изискуемите properties има

своите особености:

- Ако в логическа група „A“ попълните само минимума данни - ФУ ще търси издадения бон в собствената си контролна лента. Ако не го намери, командата приключва неуспешно. Ако го намери, данните се попълват автоматично и се издава сторно бон с всички присъстващи в оригиналния фискален бон данни;
- Ако в логически групи „B“ и „C“ попълните само минимума данни – това означава, че заявявате, че се сторнира документ, за който няма издаден УНП номер;
- Според наредба №18 – сторниране през фискално устройство е възможно до седмо число на следващия месец. Ако подадете дата несъобразена с тази особеност – фискалното устройство ще откаже отварянето на сторно бон;

init_StornoValues

Метода инициализира всички properties необходими за отварянето на сторно бон. Където това е възможно – метода попълва стойности по подразбиране.

Пропърти „***can_OpenStornoReceipt***“ става със стойност False.

Този метод се изпълнява автоматично – веднага след успешно отваряне на стороно бон. Така че стойностите да са инициализирани за следващата употреба, но клиентското приложение може да го вика и самостоятелно – преди попълването на правилните за случая стойности.

can_OpenStornoReceipt

Това поле е само за четене. „***can_OpenStornoReceipt***“ става със стойност **False** след инициализиране с метод „***init_StornoValues***“ и остава с тази стойност до момента в който клиентското приложение попълни всички необходими за отварянето на сторно бон стойности.

Пример за употреба:

Ако клиентското приложение е попълнило стойности в полета:

- ✔ Operator_Code;
- ✔ Operator_Password;
- ✔ Till_Number;
- ✔ St_Reason_Type;
- ✔ St_Doc_Number;

Ако фискалното устройство е модел в логическа група „A“ - стойността на „***can_OpenStornoReceipt***“ ще стане **True,** тъй като това са минимално изискуемите полета за отваряне на сторно бон в тази група.

Ако фискалното устройство е модел в логическа група „B“ или „C“ - стойността на „***can_OpenStornoReceipt***“ ще остане в стойност **False**, тъй като в тези групи се изисква

попълването на минимум още две полета.

Погледнете таблицата с необходими за отваряне на сторно бон полета по-горе. Общото правило е че ако сте попълнили поне едно поле от даден цвят – трябва да попълните и останалите полета от същия цвят, за да стане пропърти „***can_OpenStornoReceipt***“ със стойност **True** за устройството към което сте свързани в момента и съответно метод „***open_StornoReceipt***“ да може да бъде изпълнен.

Това означава, че в момента в който попълните минимално изискуемите полета стойността ще стане **True**, а при попълването на следващото поле е възможно стойността да стане отново **False** до попълването на соътветния набор от стойности.

При попълване на горните полета, сървъра проверява дали в съответните полета въобще е попълнено нещо и само в някои случаи може да провери дали стойността е правилна.

Пропърти „***can_OpenStornoReceipt***“ със стойност ***True*** не означава, че отварянето на сторно бон ще премине успешно, а че вече сте попълнили достатъчно на брой полета за опит за отваряне чрез съответната комбинация от задължителни и опционални параметри.

open_StornoReceipt

Поведението на COM сървъра в зависимост от въведените данни по отношение на сторно бона, може да се види по-долу.

Ако „***can_OpenStornoReceipt***“ е със стойност ***True*** и стойностите са правилни – COM сървъра ще изпълни метода, но отварянето на сторно бон все още зависи от фискалното устройство.

Ако клиентското приложение е посочило като причина нещо различно от операторска грешка, фискалното устройство ще провери дали в чекмеджето има достатъчно наличност (логически) и ако няма – ще откаже операцията.

Ако фискалното устройство е в група А и попълнените стойности са минимално изискуемите (до „***St_Doc_Number***“ вкл.) - фискалното устройство ще търси издадения бон в контролната си лента. Ако не го намери, командата приключва неуспешно. Ако го намери, данните се попълват автоматично и се издава сторно бон с всички присъстващи в оригиналния фискален бон данни.

След успешно отваряне на сторно бон – всички properties имащи отношение към метода се инициализират вътрешно с цел предпазване от последващо сторниране с неверни данни.

Метода попълва (read only) properties:

- ✔ ***AllReceipt_Count*** - Броят на всички издадени бонове (фискални и служебни) от последното приключване на деня до момента;
- ✔ ***StReceipt_Count*** - Броят на всички издадени фискални СТОРНО бонове от последното приключване на деня до момента;

При успешно изпълнение и отваряне на сторно бон – резултата от изпълнението на метода е със стойност **0**.

При неуспешно изпълнение – резултата съвпада по стойност със стойността на property “***lastError_Code***”.

„**St_Reason**“ е текстово property в което можете да посочите причината за отварянето на сторно бона. Това поле е независим, опционален параметър в група устройства А. В тази група може да бъде добавен независимо към всяка валидна комбинация от стойности, така че в зависимост от това дали полето е попълнено или не е възможно COM сървъра да изпрати една или друга комбинация от входни параметри към фискалното устройство.

Само в логическа група "А" можете да посочите УНП на самия сторно бон като го попълните в текстово property “**St_Current_UNP**“. Това поле също е независим, опционален параметър и може да бъде добавен независимо към всяка валидна комбинация от стойности. В зависимост от това дали полето е попълнено или не е възможно COM сървъра да изпрати една или друга комбинация от входни параметри към фискалното устройство.

При опит за отваряне на сторно бон – метода проверява вътрешно наличността и валидността на стойностите по входните полета. Ако има недостатъчен брой попълнени полета или някоя от стойностите не отговаря на ограниченията – метода ще върне като резултат: „***-43144***“ (Not enough input parameters).

Стойността на property “**can_OpenStornoReceipt**” ще стане **True** в комбинациите от правилно попълнени стойности на входните данни, описани по-долу.

Затварянето на сторно бон става само след плащане на **точна сума в брой**.

Логическа група „A“

ID	Operator_Code	Operator_Password	Till_Number	St_ByInvoice	St_InvoiceNumber	St_Current_UNP	St_Reason_Type	St_Doc_Number	St_Doc_UNP	St_Doc_DateTime	St_FM_Number	St_Reason	Тип сторниран документ
A01	✔	✔	✔				✔	✔					Касов бон
A02	✔	✔	✔			✔	✔	✔					
A03	✔	✔	✔				✔	✔				✔	
A04	✔	✔	✔			✔	✔	✔				✔	
A05	✔	✔	✔				✔	✔	✔	✔	✔		
A06	✔	✔	✔			✔	✔	✔	✔	✔	✔		
A07	✔	✔	✔				✔	✔	✔	✔	✔	✔	
A08	✔	✔	✔			✔	✔	✔	✔	✔	✔	✔	
A09	✔	✔	✔	✔	✔		✔	✔					Фактура
A10	✔	✔	✔	✔	✔	✔	✔	✔					
A11	✔	✔	✔	✔	✔		✔	✔				✔	
A12	✔	✔	✔	✔	✔	✔	✔	✔				✔	
A13	✔	✔	✔	✔	✔		✔	✔	✔	✔	✔		
A14	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		
A15	✔	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	
A16	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	

***Забележки**:*

- *Колона „ID“ е добавена към таблицата с цел по-добро описание, а също така в случай, че се съобщава за грешка, проблем или просто при въпрос към поддържащия екип;*
- *Ако не желаете да използвате метод „open_StornoReceipt“, а желаете да използвате „execute_Command_ByName“, то колона „ID“ може да бъде свързана с имената на методите за сторниране в метод „execute_Command_ByName“ по следния примерен начин – на „ID = A08” съответстват на имена на методи:*
  - *046_receipt_StornoOpen_A08*
  - *receipt_StornoOpen_A08*
  - *StornoOpen_A08*
- *Ако се извършва сторно по фактура – отново е необходимо между командите за тотал и затваряне на бележката да се подаде команда за отпечатване на информация за крайния клиент;*

Логическа група „B“

ID	Operator_Code	Operator_Password	St_Doc_UNP	Till_Number	St_Reason_Type	St_Doc_Number	St_Doc_DateTime	St_FM_Number	St_ByInvoice	St_InvoiceNumber	St_Reason	Тип сторниран документ
B_01	✔	✔		✔	✔	✔	✔	✔				Касов бон
B_02	✔	✔	✔	✔	✔	✔	✔	✔				
B_03	✔	✔		✔	✔	✔	✔	✔	✔	✔		Фактура
B_04	✔	✔		✔	✔	✔	✔	✔	✔	✔	✔	
B_05	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		
B_06	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	

***Забележки**:*

- *Колона „ID“ е добавена към таблицата с цел по-добро описание, а също така в случай, че се съобщава за грешка, проблем или просто при въпрос към поддържащия екип;*
- *В тази логическа група устройства – сторно бона няма собствен УНП номер;*
- *Ако не желаете да използвате метод „open_StornoReceipt“, а желаете да използвате „execute_Command_ByName“, то колона „ID“ може да бъде свързана с имената на методите за сторниране в метод „execute_Command_ByName“ по следния примерен начин – на „ID = A08” съответстват на имена на методи:*
  - *046_receipt_StornoOpen_B01*
  - *receipt_StornoOpen_B01*
  - *StornoOpen_B01*
- *Ако се извършва сторно по фактура – отново е необходимо между командите за тотал и затваряне на бележката да се подаде команда за отпечатване на информация за крайния клиент;*

Логическа група “С”

ID	Operator_Code	Operator_Password	Till_Number	St_ByInvoice	St_InvoiceNumber	St_Current_UNP	St_Reason_Type	St_Doc_Number	St_Doc_UNP	St_Doc_DateTime	St_FM_Number	St_Reason
C_01	✔	✔	✔				✔	✔		✔	✔	
C_02	✔	✔	✔				✔	✔	✔	✔	✔	
C_03	✔	✔	✔	✔	✔		✔	✔		✔	✔	
C_04	✔	✔	✔	✔	✔		✔	✔		✔	✔	✔
C_05	✔	✔	✔	✔	✔		✔	✔	✔	✔	✔	
C_06	✔	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔

***Забележки**:*

- *Колона „ID“ е добавена към таблицата с цел по-добро описание, а също така в случай, че се съобщава за грешка, проблем или просто при въпрос към поддържащия екип;*
- *В тази логическа група устройства – сторно бона няма собствен УНП номер;*
- *Ако се извършва сторно по фактура – отново е необходимо между командите за тотал и затваряне на бележката да се подаде команда за отпечатване на информация за крайния клиент*

Машинно-зависими методи (device-dependent methods)

Интерфейс “**CFD_BGR**” предоставя повечето от методите, зависими от съответния модел. Пропуснати са някои методи които се използват при производството, за целите на сервизирането или които не представляват интерес от гледна точка на ежедневната практика.

Най-удобния начин за употреба на машинно-зависим метод е чрез изпълнението на методи:

- ✔ „***set_InputParam_ByName***“;
- ✔ „***execute_Command_ByName***“;
- ✔ „***get_OutputParam_ByName***“

Имената на съответните методи могат да се получат освен от документацията и от самия COM сървър чрез методи:

- ✔ „***get_CommandsList***“;
- ✔ „***get_CommandInfo***“;
- ✔ „***get_InputParams_Count***“;
- ✔ „***get_InputParams_Names***“

Всеки от предоставените методи, чиято реализация не е удобна или не работи както трябва може да бъде „заобиколен“ чрез реализация на собствена имплементация използвайки метод „***custom_Command***“.

Като цяло – интерфейса предоставя три начина за извикване по име:

- По пълно име на командата, включващо номера на командата, името на групата и името на самия метод (*Пример: „**100_display_Show_Text**“*);
- По име на група и име на метод (*Пример: „**display_Show_Text**“*);
- По име на метод (*Пример: „**Show_Text**“*);

За да се избегне повторение или двузначност – при някои методи имената на втория и третия вариант съвпадат. *Пример: „**033_display_Clear**“, „**display_Clear**“ и „**display_Clear**“.* За по-голяма яснота от човешка гледна точка машинно-зависимите методи са описани по-долу не по номер на команда, а по групи.

Команди за управление на външен дисплей

Изчистване на дисплея

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
033_display_Clear	✔	✔	✔
display_Clear	✔	✔	✔
display_Clear	✔	✔	✔

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>Няма параметри.</td></tr>
<tr><td>B</td><td>Няма параметри.</td></tr>
<tr><td>C</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Output</td><td>✔</td><td>ErrorCode</td><td>Код на грешка.
Стойност = 0 – успешно изпълнение.
Стойност < 0 – код на грешка.</td></tr>
</table>
</td></tr>
</table>

Показване на текст на долния ред на външен дисплей

Може да се използва с метод „*execute_Command_ByName*“:

Име	Група „A“	Група „B“	Група „C“
035_display_Show_LowerLine	✔	✔	✔
display_Show_LowerLine	✔	✔	✔
Show_LowerLine	✔	✔	✔

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>TextData</td><td>Текст до 20 символа, който се изпраща директно към дисплея.</td></tr>
<tr><td>Output</td><td>-</td><td>-</td><td>-</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>TextData</td><td>Текст до 20 символа, който се изпраща директно към дисплея.</td></tr>
<tr><td>Output</td><td>-</td><td>-</td><td>-</td></tr>
</table>
</td></tr>
<tr><td>C</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>TextData</td><td>Текст до 20 символа, който се изпраща директно към дисплея.</td></tr>
<tr><td>Output</td><td>✔</td><td>ErrorCode</td><td>Код на грешка.
Стойност = 0 – успешно изпълнение.
Стойност < 0 – код на грешка.</td></tr>
</table>
</td></tr>
</table>

Показване на текст на горния ред на външен дисплей

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
047_display_Show_UpperLine	✔	✔	✔
display_Show_UpperLine	✔	✔	✔
Show_UpperLine	✔	✔	✔

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>TextData</td><td>Текст до 20 символа, който се изпраща директно към дисплея.</td></tr>
<tr><td>Output</td><td>-</td><td>-</td><td>-</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>TextData</td><td>Текст до 20 символа, който се изпраща директно към дисплея.</td></tr>
<tr><td>Output</td><td>-</td><td>-</td><td>-</td></tr>
</table>
</td></tr>
<tr><td>C</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>TextData</td><td>Текст до 20 символа, който се изпраща директно към дисплея.</td></tr>
<tr><td>Output</td><td>✔</td><td>ErrorCode</td><td>Код на грешка.
Стойност = 0 – успешно изпълнение.
Стойност < 0 – код на грешка.</td></tr>
</table>
</td></tr>
</table>

Показване на дата и час на външен дисплей

На долния ред на дисплея се показват текущите дата и час на ФУ във формат DD-MM-YY HH:MM:SS.

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
063_display_Show_DateTime	✔	✔	✔
display_Show_DateTime	✔	✔	✔
Show_DateTime	✔	✔	✔

Параметри:

Група A

Тип	Задълж.	Име	Описание
Input	✗	✗	✗
Output	✗	✗	✗

Група B

Тип	Задълж.	Име	Описание
Input	✗	✗	✗
Output	✗	✗	✗

Група C

Тип	Задълж.	Име	Описание
Input	✗	✗	✗
Output	✔	ErrorCode	Код на грешка. Стойност = 0 – успешно изпълнение. Стойност < 0 – код на грешка.
	✔	DateTime	Дата и час във формат: "DD-MM-YY hh:mm:ss DST" DD - Ден; MM - Месец; YY - Година; hh - Час; mm - Минута; ss - Секунда; DST - Text "DST" (ако текущото време е лятно часово време)

***Забележка**: Командата не се използва за мобилните фискални устройства.*

Показване на текст на външен дисплей

Текст до 40 символа, който се изпраща към дисплея. Ако е необходимо да се предадат ASCII символи по-малки от 20h (управляващи поредици) те се увеличават с 40h и се предхождат от 10h (DLE).

***Пример**: за да се предаде 1Bh,4Bh,00h в полето за данни се записва 10h,5Bh,4Bh,10h,40h.*

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
100_display_Show_Text	✔	✔	✗
display_Show_Text	✔	✔	✗
Show_Text	✔	✔	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>Текст до 40 символа, който се изпраща към дисплея.</td></tr>
<tr><td>Output</td><td>✗</td><td>✗</td><td>✗</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>Текст до 40 символа, който се изпраща към дисплея.</td></tr>
<tr><td>Output</td><td>✗</td><td>✗</td><td>✗</td></tr>
</table>
</td></tr>
<tr><td>C</td><td>Командата не се поддържа от устройствата в тази група.</td></tr>
</table>

Нефискален бон

Отваряне на нефискален бон

ФП извършва следните действия:

- Отпечатва се HEADER;
- Отпечатва се ЕИК на продавача;
- Връща се отговор;

Командата не може да се изпълни, ако:

- Фискалната памет не е форматирана;
- Има отворен фискален бон;
- Вече е отворен служебен бон;
- Часовникът не е сверен;

Може да се използва с метод „*execute_Command_ByName*“:

Име	Група „A“	Група „B“	Група „C“
038_receipt_NonFiscal_Open	✔	✔	✗
receipt_NonFiscal_Open	✔	✔	✗
NonFiscal_Open	✔	✔	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>Output</td><td>✔</td><td>AllReceipt</td><td>Броят на всички издадени бонове (фискални и служебни) от последното приключване на деня до момента /4 байта/.</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>Output</td><td>✔</td><td>AllReceipt</td><td>Броят на всички издадени бонове (фискални и служебни) от последното приключване на деня до момента /4 байта/.</td></tr>
</table>
</td></tr>
<tr><td>C</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td rowspan="2">Output</td><td>✔</td><td>ErrorCode</td><td>Код на грешка.
Стойност = 0 – успешно изпълнение.
Стойност < 0 – код на грешка.</td></tr>
<tr><td>✔</td><td>SlipNumber</td><td>Текуща стойност на глобалния брояч (1...9999999)</td></tr>
</table>
</td></tr>
</table>

Затваряне на нефискален бон

ФП извършва следните действия:

- Отпечатва се FOOTER;
- Отпечатва се поредния номер, датата и часа на документа;
- Отпечатва се с широк печат “СЛУЖЕБЕН БОН”;
- Връща се отговор;

Ако е вдигнат S1.1 командата не е изпълнена защото в момента не е отворен служебен бон.

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
038_receipt_NonFiscal_Open	✔	✔	✗
receipt_NonFiscal_Open	✔	✔	✗
NonFiscal_Open	✔	✔	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>Output</td><td>✔</td><td>AllReceipt</td><td>Броят на всички издадени бонове (фискални и служебни) от последното приключване на деня до момента /4 байта/.</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td>Output</td><td>✔</td><td>AllReceipt</td><td>Броят на всички издадени бонове (фискални и служебни) от последното приключване на деня до момента /4 байта/.</td></tr>
</table>
</td></tr>
<tr><td>C</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>✗</td></tr>
<tr><td rowspan="2">Output</td><td>✔</td><td>ErrorCode</td><td>Код на грешка.
Стойност = 0 – успешно изпълнение.
Стойност < 0 – код на грешка.</td></tr>
<tr><td>✔</td><td>SlipNumber</td><td>Текуща стойност на глобалния брояч (1...9999999)</td></tr>
</table>
</td></tr>
</table>

Отпечатване на свободен текст в служебен бон

Свободен текст за печат. В началото и края на реда се отпечатва символът '#'.

Може да се използва с метод „*execute_Command_ByName*“:

Име	Група „A“	Група „B“	Група „C“
042_receipt_NonFiscal_Text	✔	✔	✗
receipt_NonFiscal_Text	✔	✔	✗
NonFiscal_Text	✔	✔	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>InputText</td><td>До 40 символа текст.</td></tr>
<tr><td>Output</td><td>✗</td><td>✗</td><td>Няма данни в отговора.</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>InputText</td><td>До 40 символа текст.</td></tr>
<tr><td>Output</td><td>✗</td><td>✗</td><td>Няма данни в отговора.</td></tr>
</table>
</td></tr>
<tr><td>C</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>InputText</td><td>До 62 символа текст.</td></tr>
<tr><td>Output</td><td>✔</td><td>ErrorCode</td><td>Код на грешка.
Стойност = 0 – успешно изпълнение.
Стойност < 0 – код на грешка.</td></tr>
</table>
</td></tr>
</table>

Отпечатване на параметризиран текст в служебен бон

Свободен текст за печат. В началото и края на реда се отпечатва символът '#'.

Може да се използва с метод „*execute_Command_ByName*“:

Име	Група „A“	Група „B“	Група „C“
042_receipt_PNonFiscal_Text	✔	✔	✗
receipt_PNonFiscal_Text	✔	✔	✗
PNonFiscal_Text	✔	✔	✗

Параметри в група А

Група	Описание на параметрите

A

Тип	Задълж.	Име	Описание
Input	✔	Height	Цяло число от 0 до 3. 0 – 32 точки (4 mm) височина, по-високи букви 1 – 32 точки (4 mm) височина, нормални букви 2 – 24 точки (3 mm) височина 3 – 16 точки (2 mm) височина
Input	✔	Flags	Една до 3 букви: 'B', 'H' или 'I'. Всяка може да се появи най-много веднъж. B – Bold (Удебелено) H – High (Двойна височина) I – Italic (Наклонено)
Input	✔	InputText	До 40 символа текст.
Output	✗	✗	Няма данни в отговора.

Параметри в група Б

Група	Описание на параметрите

B

Тип	Задълж.	Име	Описание
Input	✔	Height	Цяло число от 1 до 3. 1 – 32 точки (4 mm) височина, нормални букви 2 – 24 точки (3 mm) височина 3 – 16 точки (2 mm) височина
Input	✔	InputText	До 40 символа текст.
Output	✗	✗	Няма данни в отговора.

Параметри в група C

Група	Описание на параметрите

Тип	Задълж.	Име	Описание
Input	✔	**InputText**	До 62 символа текст.
Input	✔	**Bold**	0 = normal text 1 = print bold text
Input	✔	**Italic**	0 = normal text 1 = print italic text
Input	✔	**Height**	0 = normal height 1 = double height 2 = half height
Input	✔	**UnderLine**	0 = normal text 1 = print underlined text
Input	✔	**Alignment**	0 = left alignment 1 = center alignment 2 = right alignment
Output	✔	**ErrorCode**	Код на грешка. Стойност = 0 – успешно изпълнение. Стойност < 0 – код на грешка.

Отваряне на служебен бон за завъртян на 90 градуса текст

Командата отваря служебен бон, в който може да се печата завъртян на 90 градуса текст. Командата няма да се изпълни, ако:

- Има отворен някакъв бон (служебен или фискален);
- Няма хартия;
- Не е сверен часовникът;

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
122_receipt_NonFiscalRotated_Open	✔	✗	✗
receipt_NonFiscalRotated_Open	✔	✗	✗
NonFiscalRotated_Open	✔	✗	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>Без входни данни.</td></tr>
<tr><td>Output</td><td>✔</td><td>RotRec</td><td>Поредният номер на отворения завъртян на 90 градуса бон за деня.</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>Няма такава функционалност.</td></tr>
<tr><td>C</td><td>Няма такава функционалност.</td></tr>
</table>

Печат на завъртян на 90 градуса текст

Командата служи за печат на завъртян на 90 градуса текст. Хартиената лента побира до 18 реда текст (12 при тясна хартиена лента). Изпратените редове текст се натрупват в паметта на принтера по реда на изпращането им. Ако командата се изпълни повече от тринадесет пъти след отварянето на бона, то натрупаната информация се отпечатва и принтерът очаква нови текстови редове или команда 124 (затваряне на бона). При отпечатването на информацията принтерът определя най-дългия от пратените редове и допълва останалите до същата дължина. Ако се изпратят повече от 18 (12) реда, то се отпечатва повече от една колона текст. Между двете колони няма никаква междина, така че с подходящо подбрани данни могат да се получат редове с неограничена дължина. Командата няма да се изпълни, ако не е отворен бон за печат на завъртян служебен текст.

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
123_receipt_NonFiscalRotated_Text	✔	✗	✗
receipt_NonFiscalRotated_Text	✔	✗	✗
NonFiscalRotated_Text	✔	✗	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✔</td><td>InputText</td><td>Съдържанието на поредния ред от текст, който искаме да отпечатаме.
Дължината му е до 100 символа.</td></tr>
<tr><td>Output</td><td>✗</td><td>✗</td><td>Няма изходни данни</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>Няма такава функционалност.</td></tr>
<tr><td>C</td><td>Няма такава функционалност.</td></tr>
</table>

Възможен е печатът на част от текста с атрибути удебелено (Bold) и подчертано (Underline). Те се включват и изключват съответно със следните тагове

Таг	Пояснение
<Tab>B	Стартира удебелен печат.
<Tab>b	Прекратява удебеления печат.
<Tab>U	Стартира подчертан печат. Може да се използува като разделителна хоризонтална линия на таблица.
<Tab>u	Прекратява подчертания печат.
<Tab>O	Стартира печат с черта най-отгоре. Може да се използува като разделителна горна хоризонтална линия на таблица.
<Tab>o	Прекратява печата с черта най-отгоре.
<Tab>A	Вмъква начална вертикална линия за таблици.
<Tab>Z	Вмъква крайна вертикална линия за таблици.
<Tab>T	Добавя нулев (непечатащ се) ред. Използува се за печат на хоризонтална линия в началото на таблицата (преди първия завъртян ред). Следващите символи не се печатат, но ако има включен атрибут "подчертаване", той предизвиква печат на линия с дебелина 2 точки.

Затваряне на служебен бон за завъртян на 90 градуса текст

Командата затваря бона. Ако има не отпечатани редове, те се отпечатват автоматично преди затварянето му. Командата няма да се изпълни, ако не е отворен бон за печат на завъртян служебен текст.

Може да се използва с метод „***execute_Command_ByName***“:

Име	Група „A“	Група „B“	Група „C“
124_receipt_NonFiscalRotated_Close	✔	✗	✗
receipt_NonFiscalRotated_Close	✔	✗	✗
NonFiscalRotated_Close	✔	✗	✗

Параметри:

<table>
<tr><th>Група</th><th>Описание на параметрите</th></tr>
<tr><td>A</td><td>
<table>
<tr><th>Тип</th><th>Задълж.</th><th>Име</th><th>Описание</th></tr>
<tr><td>Input</td><td>✗</td><td>✗</td><td>Без входни данни.</td></tr>
<tr><td>Output</td><td>✔</td><td>RotRec</td><td>Поредният номер на отворения завъртян на 90 градуса бон за деня.</td></tr>
</table>
</td></tr>
<tr><td>B</td><td>Няма такава функционалност.</td></tr>
<tr><td>C</td><td>Няма такава функционалност.</td></tr>
</table>

Obsolete interfaces (Остарели интерфейси)

Описаните по-долу интерфейси се смятат за остарели от страна на екипа в Датекс и тяхното развитие е прекратено. Ако все пак решите да продължите да ги използвате – за някои от моделите ще Ви трябва консултация по отношение на сторно боновете и статус битовете които се вдигат в този случай.

Моля пишете ни за повече подробности, ако имате проблем – екипа на Датекс ще се опита да го разреши.

Obsolete interfaces
ICSFP3530
ICSFP_KENYA
ICS_BGR_DP55_KL
ICS_BGR_FP550_KL
ICS_BGR_FP1000_KL
ICS_BGR_FMP10_KL
ICS_BGR_FP60_KL
ICS_BGR_FP2000_KL
ICS_BGR_FP700_KL
ICS_BGR_SK1_31F_KL
ICS_BGR_IBM4610F
ICS_BGR_FP550_D_KL
ICS_BGR_FP705_KL
ICS_BGR_WP500_KL
ICS_BGR_FMP350_KL

Отказ от отговорност

Информацията в този документ подлежи на промяна без предизвестие и не представлява ангажимент от страна на Датекс. Датекс не носи никаква отговорност, ако информацията в този документ е непълна или неточна.

Въпреки това, тъй като подобренията на продуктите стават достъпни, ние ще положим всички усилия да предоставяме актуална информация за продуктите, описани в този документ.

Как да изпращаме съобщения за грешки

За да разрешим проблема – често ние трябва да симулираме същите условия като тези при Вас или Вашия клиент. Моля – изпратете ни имейл със следната информация:

- ✔ Версията на COM сървъра, както и кой интерфейс използвате;
- ✔ Информация относно Windows и системата – можете да използвате информацията върната от метода “get_SystemInfo” или от някоя от демо програмите;
- ✔ Информация относно модела на фискалното устройство. Би било хубаво да отпечатате и сканирате диагностичната информация от самото устройство;
- ✔ Типа на транспортния протокол и ако използвате преходник от USB към RS-232 – информация за модела и драйвера;
- ✔ Ако използвате или става дума за метод “execute_Command” – изпратете ни стойността на командата, както и стойностите на входноизходните данни, ако има такива;
- ✔ Ако използвате или става дума за “execute_Script_V1” – изпратете ни стойността на скрипта;
- ✔ Ако е възможно:
  - активирайте „**tracking mode**” до **true**;
  - опишете възникването на проблема стъпка по стъпка;
  - изпратете ни log файла който генерира самия COM сървър с включен „**tracking mode**”;
- ✔ Ако е възможно – запишете и ни изпратете стойностите на статус битовете след като изпълнението на командата пропадне. Не изпълнявайте други команди, а също не четете други пропъртита след неуспешното изпълнение, защото това може да доведе до препокриване или заличаване на проблема. Просто вземете стойността им и я запишете;
- ✔ Изпратете ни и стойността на последната грешка (last error code);
- ✔ Ако използвате метода “**execute_Command_ByName**” – изпратете ни цялата необходима информация (стойностите на входно/изходните параметри), защото те ще ни трябват при симулирането на същите условия.
- ✔ Всъщност винаги ни изпращайте стойностите на входно/изходните параметри;
- ✔ **Ако ни изпращате сорскод – моля не ни изпращайте сорса на целия проект!** Ние не предлагаме такъв тип поддръжка!

Поддръжката която оказваме е само чрез имейл! Моля не изисквайте какъвто и да било конферентен тип поддръжка.

Ние знаем, че проекта Ви е важен за Вас и клиентите Ви. Повярвайте ни – Вашият успех е важен и за нас. Ще се радваме да помогнем, но поради спецификата на темата, политиката на компанията е да оказва помощ на разработчиците единствено в писмена форма – най-често и предимно чрез имейл.

Контакти

*Ако откриете грешка – изпратете ни електронно съобщение. Ние ще се радваме да отстраним проблема в следващи версии на имплементацията на интерфейс“**CFD_BGR**“. Ако сте програмист – чувствайте се свободен да питате за различните аспекти и начини за употреба на COM сървъра. Ако имате препоръки или желания за доработка на интерфейс „**CFD_BGR**“ ще се радваме да го обсъдим. Пишете ни – ще се опитаме да помогнем.*

Можете да изпращате имейлите със своите съобщения и въпроси до:

dobrin@datecs.bg