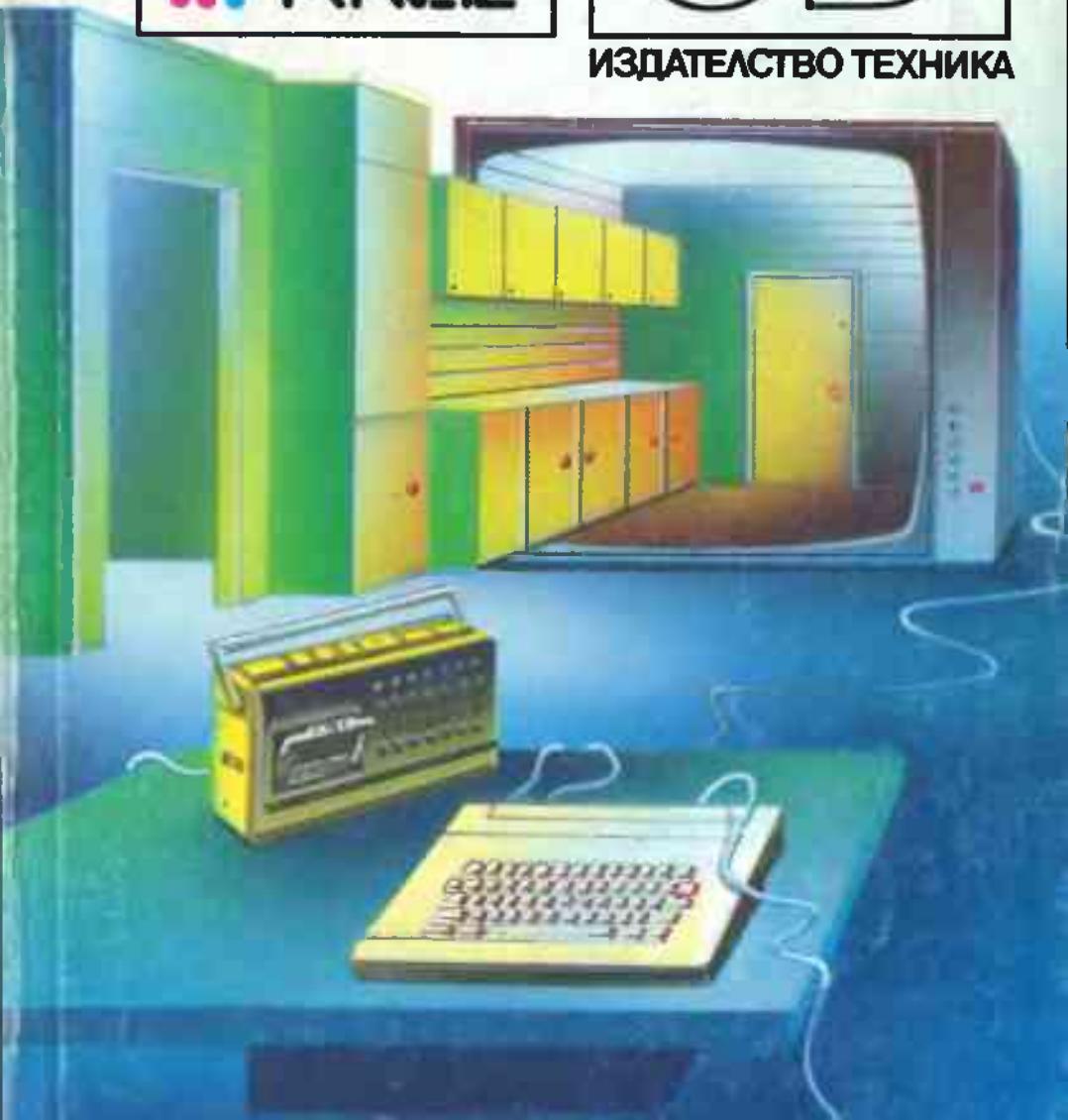


ДОМАШЕН КОМПЮТЪР



ИЗДАТЕЛСТВО ТЕХНИКА



Драги читателю,

Издателство „Техника“ Ви предлага първата у нас книга, придружена от касета с програми за домашния компютър „Правец—8Д“.

За да стимулират ускореното усвояване на нови знания, издателството и НПК по микропроцесорна техника — Правец, поставят началото на творчески контакт с читателите, като обявяват ПРЕМИЯ НА ЧИТАТЕЛЯ. За притежателите на книгата „Домашен компютър Правец—8Д“ премиите са следните:

- 5 първи премии — домашен компютър Правец—8Д;
- 20 втори премии — видеокасета с филм „Компютърът“;
- 20 трети премии — комплект от 4 дискети към книги от поредицата „Микрокомпютърна техника за всички“.

Условието за участие е да попълните вярно теста в талона на края на книгата и да го изпратите до 20 май т. г. на адрес:

ДИ „Техника“, бул. Руски 6, София

Премиите ще бъдат разпределени с помощта на компютър и връчени по време на Международната изложба — панаир на книгата, в София на 6.VI.1986 г.

Желаем Ви успех!

ОРЛИН П. ВЪЛЧЕВ
Инж. ПЕНЧО А. СИРАКОВ
ДИМИТЪР Х. ВАВОВ

ДОМАШЕН КОМПЮТЪР ПРАВЕЦ-8Д

ДЪРЖАВНО ИЗДАТЕЛСТВО
„ТЕХНИКА“
СОФИЯ, 1986

Книгата е помагало за работа с домашния компютър Правец—8Д. Дадени са указания за включването на компютъра, посочени са техническите и програмните му възможности. Основната част от текста е описание на командите и функциите на езика БЕЙСИК за Правец—8Д, подкрепено с примерни програми. Приложени са справочни данни.

Изданието е необходимо на всички читатели, които вече са направили първите стъпки в програмирането и искат да ползват компютъра като другар в игрите, помощник в овладяването на нови знания и средство за разгръщане на творческите им възможности.

Всички програми са записани на касета за битов касетофон.

ПРЕДГОВОР

Изминаха повече от 20 години от разработването на първия български електронен калкулатор. Нашата „ЕЛКА“ беше едно изделие на много високо техническо равнище, с редица оригинални и точни решения. Не случайно тя стана нарицателно в българския език за електронните калкулатори въобще.

Постепенно българската електроника се превърна в съвременен отрасъл, даващ облик на нашата социалистическа промишленост. Народна република България завоюва водещи позиции на голям производител и износител на изчислителна техника сред социалистическите страни.

През последните години светът беше буквально залят от вълната на микрокомпютрите — поредното чудо на микроелектрониката. Достъпна цена, малки размери, повищена надеждност и лесно обслужване — това са причините за масовото им навлизане в институти, производствени предприятия, училища и домове.

Микрокомпютрите се профилираха постепенно в три основни групи:

- персонални компютри с универсално предназначение;
- микрокомпютри за специализирани промишлени и научноизследователски цели;
- домашни компютри за обучение, игри, развлечения и помощ в домакинството.

Вече е реалност и първият български домашен компютър — Правец—8Д. Той е най-малкият член на семейството микрокомпютри Правец, произвеждани в Научно-производствения комбинат по микропроцесорна техника — гр. Правец (до 1985 г. — Приборостроителен завод). За разлика от персоналните компютри Правец—82 и Правец—16, които са създадени в Института по техническа кибернетика и роботика към БАН, домашният компютър е собствена разработка на младежки колектив от Базата за развитие и внедряване към комбината. Цифрата 8 показва разредността на микропроцесора в компютъра, а буквата Д означава домашен.

Типичен представител на съвременните домашни компютри, Правец—8Д предлага функционално развита версия на популярния език за програмиране БЕЙСИК, оперативна памет за 65 536 символа (около 40 страници текст), три звукови канала, графика с голяма разделителна способност и осем цвята. Предвидена е възможност за включване на печатащо и дисково устройство (с гъвкав магнитен диск), които позволяват компютърът да се използува и за кореспонденция, обработка на документи и други професионални цели.

Определението „домашен“ се наложи за най-малките компютри. Типично за тях е използването на битов касетофон като външна памет

и домашен телевизор като монитор, което определя един важен показател — достъпната цена. Касетофонът обаче е специализирано устройство за музика, а не за работа с данни, затова използването му вместо дисково устройство е компромис. По същество това предопределя и целите, за които може да се използува ефективно домашният компютър:

- за обучение по програмиране и различни учебни дисциплини, за езикова подготовка;
- за игри и забава (в компютърното царство най-лесно се влиза чрез чудния свят на игрите);
- в бита, за решаване на някои лични задачи.

Досега компютрите отиваха основно при професионалисти — програмисти, инженери и др. Домашният компютър за първи път става наистина част от битовата техника, но най-сложната и най-взискателната при работа в нашия дом.

Компютри като Правец-8Д привличат основно лецата. Достатъчно е само да се види как малки групи ученици търпеливо чакат за свободно място в компютърните клубове. Когато техни учители са не по-малко ентузиазирани хора, успехът наистина е гарантиран. Един оставаш и след последния час „бате Владко“ може да даде начален тласък на няколко десетки програмисти, като оживи компютрите в едно училище.

От първия български домашен компютър се очаква да стане икономически оправдано средство за обучение и практика по програмиране, средство за компютърно ограмотяване.

ПОДГОТОВКА ЗА РАБОТА С КОМПЮТЪРА

Целият компютър Правец—8Д е събран в една кутия с размери приблизително 35 на 25 см. Постигненията на микроелектрониката позволиха пълната изчислителна мощност да се концентрира само в един чип — микропроцесора. Много малък обем заема и оперативната памет. Добавяме захранване, малък високоговорител за звук, няколко извода за връзка с външни устройства. Всичко това е монтирано и свързано към една платка. Единоплатковите микрокомпютри днес са образец за съвременна техника.

Компютърът Правец—8Д работи с осемразреден микропроцесор СМ 630. Това означава, че данните и числата се обработват на групи от осем бита (един байт). Той може да адресира 65 536 байта оперативна памет, което е прието да се означава като 64 К*. Една четвърт от паметта е запазена за версия на езика БЕЙСИК. Тази памет е постоянна, т. е. нейното съдържание не се губи при изключване на захранването. Самият БЕЙСИК е разширен спрямо известната ни от Правец—82 версия. Той включва няколко специализирани звукови и графични команди. Такива команди са общоприети за домашните компютри. Съвременният малък компютър трябва да има засилени графични и музикални възможности, за да отговори на изискванията на обучението и игрите.

Останалата част от оперативната памет осигурява около 48 К за въвеждане на програми. Разпределението и използването на паметта става по начин, различен от този при Правец—82. Това се налага от факта, че двата компютъра са от различен клас и с различно предназначение. Програми от единия не могат пряко да се изпълняват на другия.

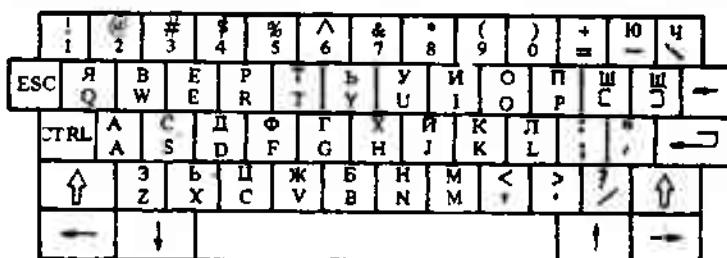
КЛАВИАТУРА

Неразделна част от компютъра е неговата клавиатура. Тя е монтирана на основната платка. През клавиатурата се дават команди и се въвеждат данни. Тя е единственият начин за връзка между човека и компютъра.

Тук е показана клавиатура, която с малки изменения става за повечето видове микрокомпютри. От клавиатурата на Правец—8Д могат да се

* К = 1024 байта.

въвеждат само главни букви на кирилица и латиница. Освен буквите, цифрите от 0 до 9 и специалните символи, като ., ! + и т. н., тя има



няколко специализирани клавиша:

ESC или ОСВ — управляващ клавиш „освобождаване“;
CTRL или МК — управляващ клавиш „машинен контрол“;

DEL или — клавиш за изтриване на символи от экрана;

RTN или ← — клавиш RETURN за преместване на маркера на нов ред и предаване управлението на компютъра;

SHIFT или ↑ — клавиш за избор на горен или долнен регистър, в нашия случай на латиница или кирилица.

Най-често се използва клавишът RETURN. Натискането му обикновено означава край на нашата работа и начало за работа на компютъра. Процесорът на всеки компютър има два основни режима — режим на активна работа и режим на изчакване. Когато компютърът си е свършил работата и изчаква нова команда или данни, той показва това на екрана с едно мигащо квадратче, наречено **маркер** (показалец, курсор). Докато пишем нещо с клавиатурата, съответният текст излиза на екрана, а маркерът се измества символ по символ. Компютърът продължава да чака, докато натиснем клавиша RETURN. Ако сме въвели команда, с това връщаме и управлението на задачата на компютъра. Маркерът изчезва от екрана и се появява отново едва след като компютърът си е свършил работата. Отново е наш ред да дадем нова команда и да натиснем RETURN.

Другите два чисто компютърни клавиша са ESC и CTRL. Общото за тях е, че те не се използват самостоятелно, а само в комбинация с друг клавиш. Разликата е в начинът за задаване на комбинацията. При комбинация с ESC първо трябва да се натисне и отпусне самият ESC, а едва след това да се натисне и отпусне другият клавиш. За CTRL процедурата е друга — той се държи натиснат, докато се натисне и вторият клавиш в комбинацията. Точно така се работи и с главните букви при пишещата машина — клавишът за горен регистър се държи натиснат, докато се натисне клавиш за съответната буква. Произволното натискане на клавиша CTRL не води до никакви промени за компютъра и програмата.

Прието е комбинацията между управляващ клавиш и втори клавиш да се означава така:

CTRL—С, което означава натиснете CTRL и не го отпускайте, докато не натиснете и С;

ESC—С, което означава натиснете и отпуснете ESC, след което натиснете и отпуснете С.

Символите, участващи в комбинация с управляващите клавиши, трябва да са на латиница!

Смисълът на управляващите клавиши е придаването на още две значения на произволен клавиш от клавиатурата. Това се използва от конструкторите на самите компютри, които вграждат стандартно определени функции като комбинация с управляващи клавиши. Така същото CTRL—С се използва почти навсякъде като команда за прекъсване на изпълняваната програма.

Програмистите могат да задават нови комбинации и да изменят съществуващите, естествено без да обръкват работата на системата. Останаха два специализирани клавиша, които обаче имат аналогични функции на познати клавиши от пишещите машини. Единият е клавищът за изтриване на символи от екрана – DEL или ←. Еднократното му натискане връща маркера с една позиция назад, като същевременно изтрива от паметта и екрана последния въведен символ. При пишещата машина има такъв клавищ за връщане с една стълка назад.

Вторият „машинописен“ клавищ е този за избор на горен и долен регистър. Означен като SHIFT, ЛАТ или ♀ при компютъра, той служи за избор на латиница или кирилица, докато при класическата пишеща машина се използва за избор на малки и главни букви в рамките на една азбука.

За разлика от някои други компютри при Правец-8Д няма клавиши за „заключване“ на горния регистър (известен като CAPS LOCK). Компютърът прави това с управляващия символ CTRL—T.

За да приключим с особените клавиши при Правец-8Д, трябва да отбележим четирите клавиша със стрелки, разположени по два от двете страни на клавиша за интервал. Те местят маркера по четирите посоки на екрана, без това да променя съдържанието на паметта и съответно изображението на екрана. Тези клавиши са особено удобни за движение на обект по екрана и се използват интензивно за редактиране.

Обърнете внимание, че клавищът за изтриване на последния въведен символ може да бъде означен с лява стрелка. Той се различава от клавиша с лява стрелка за движение на маркера освен по функции, също и по размер и разположение на клавиатурата.

Ако сте работили с Правец-82, сигурно сте забелязали, че в някои модели на този компютър има лява и дясна стрелка, които се използват за редактиране на текст по екрана. *Стрелките при Правец-8Д не могат да се използват за изтриване и въвеждане, а само за позициониране по екрана.* Дясната стрелка (CTRL—U) при Правец-82 тук е CTRL—A, а лявата (CTRL—H) е клавищът DEL.

Евентуалното сравнение с клавиатурата на по-големия Правец—82 ще покаже липсата на два специални клавиша при домашния компютър. Единият е оцветяваният в червено RESET (или RST). Той се използва за аварийно прекъсване на текущата програма. Маркерът се появява на экрана, програмата остава непроменена в паметта, а компютърът очаква нова команда. Тъй като случайното натискане на RST има неприятен ефект, този клавиш е преместен на долната страна на кутията при Правец—8Д, като умишлено не е леснодостъпен.

Другият клавиш, който наистина липсва, е клавищът за повторение RPT. Натискането му при Правец—82 в комбинация с друг клавиш води до многократно въвеждане на другия избран символ. Това не се налага при домашния компютър. Той осигурява автоматично повторение на кой да е символ, ако съответният му клавиш се задържи натиснат повече от 2 s.

Максималният брой на символите в един ред е 80. Ако въвеждате повече от 75 символа, при 76-ия ще се чуе звън от камбанка. Така компютърът предупреждава за край на реда, аналогично на звънчето на обикновената пишеща машина. Камбанката ще звънне и за 77-ия, 78-ия и 79-ия символ. На мястото на 80-ия символ еcranът ще покаже \ , а маркерът ще се премести в началото на следващия ред.

Функциите на комбинациите на управляващите клавиши с други са описани в приложението.

ПЕРИФЕРИЯ

За нормална работа с компютъра са необходими някои външни устройства, наречени най-общо периферия.

На първо място е устройството, на което ще гледаме нашите команди, изпълнението на задачата, различните резултати и съобщения. Това е видеомониторът, който има пасивна роля в изчислителния процес. На неговия еcran се извежда всичко, изписано от нас с клавиатурата, както и текущото изпълнение на компютърните задачи.

Както отбелязахме вече, домашните компютри са предвидени за работа с обикновен битов телевизионен приемник. Той може да бъде за черно-бяло изображение, но добрият цветови възможности и характеристика приложенията при домашните компютри предполагат цветно изображение. В противен случай няма да се използват пълните възможности на системата PAL, т. е. на всички произвеждани у нас телевизионни приемници за цветно изображение. В комплектацията на компютъра е включен специален кабел за връзка с такъв телевизор. Разнообразието на телевизионни приемници от различни производители и на различни системи може да затрудни връзката с компютъра. Понякога тя е въпрос на подходящ куплунг (извод, съединител), а понякога просто не може да се получи цветно изображение.

Независимо от куплунга за връзка с телевизионен приемник компютърът Правец—8Д има и втори куплунг — за връзка с монитор за цветно изображение (RGB). Мониторите са специално съобразени за работа с компютри и дават по-добро качество на изображението. Работата с текст по целия екран изисква точно изображение във всички части на екрана. Докато при Правец—82 връзката с монитор се реализира с допълнителна платка (модул RGB), при Правец—8Д връзката е осигурена стандартно още на основната платка. Нормално е необходимо кабел да се доставя заедно с монитора.

Оттук нататък ще приемем, че сте свързали **Вашия компютър** с подходящ телевизионен приемник за цветно изображение.

Конфигурацията компютър—монитор осигурява само възможност за начална практика по програмиране — въвеждате отделни команди и програми и проверявате тяхното действие. За да запазите резултатите от своя труд, необходимо Ви е външно устройство за запис. За малкия компютър с ниска цена се очаква това да бъде обикновен битов касетофон. Програми и данни се записват и четат от обикновена касета. Касетофонът осигурява последователен метод на достъп до информациите. Това означава, че за да стигнете до последната програма, трябва да превърнете лентата за първите програми. Този метод ограничава обхвата на ефективно решаваните приложни задачи, като в същото време отлично обслужва работата на компютъра в повечето случаи.

При работа с касетофон трябва да регулирате силата на звука. Данните естествено не възпроизвеждат мелодия, напротив — при четене се чува специфичен шум. Регулирането на звука обаче е много важно. При много голяма и при малка сила на звука информацията ще постъпи изкривена в компютъра, а това ще доведе до грешки в програмите.

Тъй като касетофонът ще бъде основната периферия към Правец—8Д, в следващите няколко страници ще разгледаме най-важните процедури и правила за работа с него.

РАБОТА С КАСЕТОФОН

Приемаме, че вече имате у дома някакъв касетофон. Най-разпространеният и евтин метод за съхраняване на програми при домашния компютър е върху обикновена касетна лента. Това в никакъв случай не е най-удобното, нито най-бързото, нито най-надеждното средство за запазване на програми, но то все още е значително по-евтино от дисковите устройства. В действителност, ако трябва да купувате специално за компютъра, купувайте евтино. Хубаво е касетофонът да има брояч — това улеснява много търсенето на програми.

Стереокасетофони не са необходими — компютърът работи само с един канал.

Практиката изиска да използвате един и същи касетофон колкото се може по-дълго. Често при опит да заредите програми, които са били записани на касетофон, ще срещате проблеми по съвместимост (дори и малки разлики в настройката на главите могат да попречат при четене на една иначе добре записана касета).

Типът на кабела зависи от избрания касетофон. Самият компютър се нуждае от 3- или 7-шифтов DIN-куплунг, който да съответствува на касетофонния вход на компютъра. Повечето устройства за запис на данни използват един DIN-извод, който служи едновременно за запис и четене. Някои от най-простите монокасетофони предлагат само изводи за микрофон и слушалка, което малко усложнява нещата. В този случай са необходими отделни куплунги за вход и изход, като при запис се ползва само връзката MIC, а при четене — само връзка EAR.

Много по-добре е да използвате къси компютърни ленти (от 10 и 15 min) вместо стандартните касети за 60 и 90 min. По-дългата лента затруднява търсенето на дадена програма. По-дългите и по-тънките ленти много по-лесно се разтеглят и влошават критично качеството на записа на данни.

След като поставите нова касета в касетофона, препоръчваме да пренавиете в бърз режим лентата до края, а след това обратно, за да сте сигурни, че е навита стегнато. Повечето касети имат начален маркер, върху който не може да се записва. Преди запис трябва да позиционирате лентата след началния лентов маркер. Всъщност си струва да оставите касетата да се върти без запис приблизително 15 s, защото повечето грешки при запис се дължат на повреда на изложеното на външни влияния начало на лентата (или пък на разтеглянето на първите няколко сантиметра). Заредете брояча на нула и сте готови да започнете.

За да запишете една програма на лента, трябва да извършите следните действия:

1. Проверете дали касетофонът е включен и зареден с чиста и позиционирана касета.
2. Убедете се, че връзката компютър—касетофон е правилна.
3. Проверете на екрана с команда LIST програмата, която ще записвате.
4. Въведете директната команда:
`,SAVE "прg", S`

където *прg* е името на програмата. Името може да има дължина 16 символа, но практиката изиска колкото може по-кратко и лесно за запомняне име.

Компютърът Правец—8Д пише и чете в два различни режима. Автоматично той приема режим на бързо предаване със скорост 2400

bits/s (бода). Ако прибавите S, ще се избере бавният режим със скорост 300 bits/s. Въпреки че и двета метода са достатъчно надеждни, винаги е добре да вземете предпазни мерки и да осигурите едно копие на всяка по-ценна програма, направено в бавен режим.

5. Натиснете клавишите RECORD и PLAY на касетофона.

6. Натиснете клавиш RETURN на компютъра. Когато изпълнението на CSAVE приключи, в текущата позиция на маркера ще се появи обичайното съобщение ГТОВ. Спрете касетофона.

Ако внимателно сте изпълнили описаните действия, сега имате копие от програмата си на касета. Хубаво е да проверите веднага качеството на записа.

Пълните възможности на всички формати на командата CSAVE, както и начинът за проверка на качеството на записа, са описани във втората част на книгата.

Ако искате да заредите от лента програма, която е била записана с команда CSAVE, трябва да извършите следните действия:

1. Проверете дали касетофонът е включен, дали регулаторът за сила на звука е настроен на средно положение и дали регулаторът за тона е в най-високо положение.

2. Проверете връзката касетфон—компютър.

3. Като използвате брояча, позиционирайте лентата в началото на програмата, която ще заредите.

4. Въведете:

CLOAD "prg", S

Името на файла prg трябва да бъде написано точно както при самия запис. Използвайте S само ако програмата е записана при бавен режим. В противен случай компютърът няма да разпознае програмата. Ако се съмнявате за точното име, можете да използвате формата

CLOAD ***

и компютърът ще зареди първата програма, която срещне на лентата.

5. Натиснете клавиш RETURN на компютъра.

6. Натиснете клавиш PLAY на касетофона.

След успешно зареждане компютърът дава съобщението ГТОВ. Спрете касетофона.

Пълните възможности на всички формати на тази команда, както и начините за запис и зареждане на блокове от паметта и на масиви са описани във втората част на книгата.

ГРИЖИ ЗА КАСЕТИТЕ

Работата с касети неизбежно създава проблеми. Загубата на една написана с труд програма е истинско нещастие. Тук даваме някои общи съвети за запазване на касети с програми. Трябва да се подчертава, че ако една лента не е била използвана дори само месец-

два, има опасност от разрушаване на част от записа. Ето някои прости правила, които могат да направят този рисък възможно най-малък:

1. Не записвайте нищо на първите 10—15 s от касетата. По-голяма част от проблемите с обтягането и нарушаването на по-критието стават именно на тази област от лентата.

2. Когато не използвате една касета, винаги я връщайте в кутията.

3. Никога не оставяйте касетата върху телевизор или друг уред, който генерира електромагнитни вълни. Те повреждат съхранените на лентата сигнали.

4. Не докосвайте повърхността на лентата. След работа старательно пренавивайте касетата до края, така че само маркерът да бъде изложен на външни влияния.

5. Преди да записвате върху касета, трябва да пренавиете лентата докрай и след това да я върнете обратно. Така ще осигурите нужното обтягане.

6. Редовно почиствайте главите за запис и четене на касетофона.

7. След като сте записали една програма, проверете дали името на програмата и позицията на брояча са добре документирани на касетата и кутията ѝ. Трябва непременно да запишете в какъв режим е била записана касетата (т. е. бърз или бавен), ако програмата е все още в процес на разработване, отбележете и номера на версията.

8. Когато стигнете до последната версия (оригинал), защитете касетата, за да предотвратите евентуално изтриване по невнимание (за целта махате пластмасовия ограничител на гърба на касетата, с което забранявате записи).

9. Правете копия на всички по-важни програми.

10. През известен интервал пренавивайте лентите дори и да не зареждате нищо от тях. Това ще предотврати намагнитване на съседни секции на лентата.

Някои касетофони могат да накарат компютъра да даде фалшиви съобщения за грешка ERRORS FOUND след изпълнение на CLOAD. Вариантите на водещия сигнал на лентата могат да „поддъжат“ вградените в компютъра проверки и да регистрират грешки. Въпреки съобщението за грешка програмата ще се зареди правилно, но няма да е възможно автоматично стартиране след команда тип CSAVE AUTO.

ДРУГА ПЕРИФЕРИЯ

Освен с телевизор и касетофон, компютърът Правец—8Д може да работи и с други периферни устройства.

На първо място това е печатащото устройство (принтер). За него е предвиден специален куплунг, а на основната платка има вграден па-

паралелен интерфейс тип CENTRONICS. Това означава, че можете да работите с всички устройства, поддържащи този интерфейс и по-специално разпространените у нас матрични печатащи устройства „Петрич“ или японските на фирмата EPSON. Важно е, че Правец-82 и Правец-8Д спазват единен стандарт при кодиране на символите. На практика това означава, че всяко печатащо устройство за Правец-82 със стандартна кирилица може да се ползува при наличие на паралелен интерфейс, при това без специална периферна платка, а свързано направо, само с кабел. Схема на изводите за кабела е дадена в края на тази книга.

На задната част на компютъра има още един куплунг, наричан куплунг за разширение. Той има универсален характер, като позволява свързване на дисково устройство, модем за предаване на данни на разстояние, лостове за игри (джой-стик). За съжаление най-интересното устройство — дисковото, е рядкост за домашните компютри. Такива устройства се предлагат от малко производители в света, цената им е висока, дори по-висока от тази на компютъра. Просто евтините домашни компютри са евтини именно защото не се очаква да работят с дискови устройства. Това естествено се отразява на удобството при работа.

ВКЛЮЧВАНЕ НА КОМПЮТЪРА

Компютърът се свързва към стандартно мрежово напрежение 220 V. Преди това трябва да свържете телевизионния приемник и касетофона. Схема на изводите на домашния компютър е дадена в приложение.

Първо трябва да включите телевизора, за да загрее. Настройте го на 3-ти или 36-и канал. Ако включите и компютъра, на екрана ще се появи светъл надпис:

**МИКРОКОМПЮТЪР ПРАВЕЦ-8Д
BASIC БРВ**

32631 СВ. БАЙТА

ГОТОВ



Преди да започнете работа, регулирайте яркостта и контраста на екрана, докато получите добър образ. Може да се наложи и да превключите на друг канал.

След съобщението за модела на компютъра, версия на БЕЙСИК и

инициалите на разработчика (База за развитие и внедряване към комбината в гр. Правец), компютърът показва обема на свободната памет в байтове (т. е. в брой символи).

Появилата се на екрана дума ГОТОВ означава, че компютърът е готов за работа и очаква инструкции. Мигащото квадратче е маркерът, който показва позицията, в която ще се въведе или изведе следващият символ.

Ако и Вие сте готови за работа, пожелаваме Ви успех!

КОМАНДИ И ФУНКЦИИ

В този раздел е описано използването на всички запазени думи във версията на езика БЕЙСИК за Правец—8Д. За много команди и функции са дадени примери.

В описанията са приети следните означения:

- прг* — име на програма (име на файл);
- усл* — логическо условие;
- изр* — израз;
- оп* — оператор;
- а* — адрес;
- n, m* — числа;
- x, y* — координати;
- з* — символен низ;
- v* — променлива.

За отделни команди и функции се използват и други специфични символи, които са пояснени в самия текст.

Всички означения с малки курсивни букви се заменят с конкретен израз при ползване на команда или функция. Означенията, дадени с главни букви, се изписват без промяна.

Не са задължителни означенията, заградени с квадратни скоби.

Стандартни за БЕЙСИК са и символите # — за шестнадесетични числа, % — за цели числа, \$ — за низ.

Когато програмите се въвеждат в паметта, интерпретаторът на БЕЙСИК замества всяка запазена дума със специален код (*token*), който заема един байт. Тези кодове са дадени след формата на всяка команда или функция.

Правилата при работа с БЕЙСИК дават достатъчна свобода при конкретизиране на отделни параметри и аргументи в команди и функции. Ако изрично не е посочено друго, навсякъде в описаните полета могат да се задават не само числа и променливи, но също така изрази. Работата с компютъра постепенно изгражда необходимите практически навици за отделните групи команди и функции.

КОМЕНТАР

REM

Формат: REM
Код: 157

Въвеждането на коментар към програмата става с думата REM. Ограничения за съдържанието на коментара няма, тъй като той се пренебрегва от интерпретатора.

Включването на REM увеличава заеманата от програмата памет. Независимо от това се препоръчва програмата да съдържа коментари, за да бъде по-четлива. Това се отнася особено за програмистите-непрофесионалисти. Ясната днес програма утре се забравя и много трудно се разчита, затова коментарите наистина пестят ценно време на програмистите.

В професионалното програмиране понякога се практикува следното. Подготвят се два варианта на програмата — един четлив, с коментари и пояснения, и втори — без коментари, по-компактен.

Вместо REM за по-кратко може да се използува апостроф (прим).

10 REM --- REM ---

20 ' ТОВА Е ДРУГ ВИД REM-ОПЕРАТОР

ОПЕРАЦИИ НАД ЦЕЛИ ПРОГРАМИ

NEW

Формат: NEW
Код: 193

Тази команда изтрива от паметта текущата програма и нейните променливи. На практика тя изтрива достъпната за потребителите памет RAM.

Препоръчва се в началото на всяка нова програма еднократно да се използува NEW. Това ще гарантира, че в паметта няма да има „изостанали“ инструкции или данни от предишна програма.

CLEAR

Формат: CLEAR
Код: 189

Командата изчиства стойностите на всички използвани в момента променливи. В показания пример на екрана се извеждат стойностите на 5 променливи — 3 числови и 2 символни. Ред 100 изчиства променливите, а редове 110 и 120 извеждат стойностите им след изчистването (0 за числовите променливи и празни низове — за символните променливи.)

```
10 REM --- CLEAR ---
20 CLS
30 A=5
40 B=10
50 C=20
60 PRINT A,B,C
70 C$="ЕЦ-ВД"
80 B$="ПРАВ"
90 PRINT B$+C$
100 CLEAR
110 PRINT A,B,C
120 PRINT B$+C$
130 REM
140 REM ВИЖ NEW И RUN
```

LIST

Формати: LIST
LIST *n*
LIST *n-m*

Код: 188

Тази команда дава възможност да се „разлисти“ програмата, да се видят нейните оператори. Тя извежда на екрана един ред, поредица от редове или цялата програма.

Да разгледаме различните формати на командата.

Когато се употреби самостоятелно, LIST просто ще изведе на екрана цялата програма. Тъй като всяка по-сериозна програма очевидно е значително по-дълга от един екран, ще се налага да се спират отделни секции от програмата, докато тя преминава през екрана. За Правец—8Д това става, като се натисне клавишът за интервал. Изобразяването на програмата спира и може да се поднови, като се натисне повторно клавишът за интервал.

Когато се използва форматът LIST *n*, на екрана ще се изведе само редът с номер *n*, например

LIST 40

ще изведе на екрана ред 40 от програмата.

Команда LIST *n-m* извежда част от програмата — от ред *n* до ред *m* включително, например

LIST 40-80

ще изведе редове от 40 до 80.

LLIST

Формат: LLIST
Код: 142

Командата е аналогична на LIST, но отпечатва програмата върху хартия на печатащо устройство, вместо да я извежда на экран.

RUN

Формати: RUN
RUN *n*
Код: 152

Използвана самостоятелно като директна команда, RUN започва изпълнението на програмата, която се намира в момента в паметта на компютъра. За разлика от Правец-82, тук след RUN не може да се посочва име на програма. Команда RUN *n*, където *n* е номер на ред, кара компютъра да започне изпълнението на програмата от посочения ред. Ако такъв ред няма, ще се получи съобщение за грешка UNDEFINED STATEMENT ERROR. Командата RUN може да се използува и в тялото на програмата:

```
10 REM --- RUN ---
20 A$—"НАТИСНИ КЛАВИШ ЗА КРАИ":GOSUB
30
30 CLS:PAPER 1
40 X=RND(1)*32+1:Y=RND(1)*20+1
50 PLOT X,Y,"ПРАВЕЦ":WAIT 50
60 V$=KEY$
70 IF V$ THEN END ELSE RUN
80 Z=LEN(A$):FOR L=1 TO Z
90 ' *** ГОРЕН РЕД ***
100 POKE 42999+L,ASC(MID$(A$,L,1))
110 NEXT L
120 RETURN
130 REM
140 REM ВИЖ CONT,END,STOP
```

CLOAD

Формати:

CLOAD "" [S]
CLOAD "prg" [S]
CLOAD "prg", J [S]
CLOAD "prg", V [S]

Код:

182

Командата зарежда програма от касета. Допуска четири формата, които трябва да завършват с S, ако програмата е била записана в бавен режим (slow).

Командата с формат CLOAD "" зарежда в паметта първата срещната програма. Ако не се знаят имената на програмите в дадена касета, тя може да се претърси, като се дава последователно тази команда.

Командата с формат CLOAD "prg" зарежда програма с определено име. Параметърт "prg" може да бъде всяко име с дължина до 16 символа. Практиката налага да се работи с по-кратки и лесни за запомняне имена.

Докато търси програмата, компютърът ще издава съобщението SEARCHING... (търся), а когато я намери и докато я зарежда в оперативната памет — съобщение LOADING (зареждам). Имената на програмите са последвани от В, което означава файл на БЕЙСИК, а блоковете от паметта — от С, файл на машинен език. Намерените файлове, които поради някаква причина не могат да бъдат заредени, се отбелзват със съобщението FOUND....FILENAME (намерена prg програма), докато за правилно зарежданите файлове се издава съобщение LOADING....FILENAME. Ако при зареждането се срещне грешка, компютърът ще издаде съобщение ERRORS FOUND и ще попречи да започне автоматично изпълнение на програмата, ако тя е била записана с команда CSAVE "prg", AUTO.

Ако на касетата е записан блок от паметта като отделен файл, в командата CSAVE се посочват началото и краят на блока. При зареждането на такъв файл обаче, е необходимо да се посочи само името му. При формат CLOAD "prg", J командата добавя втора програма към края на предварително заредена програма. Всички номера на редове във втората програма трябва да бъдат по-големи от най-големия номер на ред в първата. Ако това не е спазено, програмата няма да може да се изпълни правилно, поради дублираните номера на редове. Команда с формат CLOAD "prg", V проверява дали дадена програма (или блок от паметта) е записана правилно на касетата. Програмата е в паметта на компютъра, записва се на касета с команда CSAVE и се зарежда обратно в паметта по обичайния начин, но с операнд V. (Първоначалната програма остава в паметта на компютъра.) Когато зареждането започне, компютърът ще изпише VERIFYING (проверявам) на горния ред на екрана. Той ще зарежда байт по байт и едновременно ще сверява с началната програма. Ако зареждането завърши успешно, в текущата позиция на маркера ще се появи съобщението

0 VERIFY ERRORS DETECTED. Това означава, че програмата е записана правилно на касетата и вече може да се изтрие от паметта (въпреки че винаги е добре да се правят повече от едно копие на всяка програма). Ако такова съобщение не се появии, зареждането може да се повтори, като предварително се регулират тонът и силата на звука на касетофона. Ако и след няколко опита все още няма съобщение 0 VERIFY ERRORS DETECTED, приема се, че записът на програмата не е бил качествен и ще трябва да се повтори.

Командите CSAVE и CLOAD работят в бърз режим, освен ако в края им не се добави операндът за предаване на данни в бавен режим (скоростта на прехвърляне при бърз режим е 2 400 bits/s (бода), а при бавен — 300 bits/s). Ако една програма е била записана в бавен режим, тя може да се прочете само в този режим.

CSAVE

Формати: CSAVE "prg" [S]
 CSAVE "prg".AUTO[S]
Код: 183

Тази команда записва програма (или блок от паметта) върху касета. Тя има няколко формата.

Най-често се използва формат CSAVE "prg", където параметърът *prg* е всяко име на файл с дължина до 16 символа. Тази команда ще запише текущата програма под посоченото име. Подобно на командата CLOAD, и CSAVE може да бъде последвана от операнд S. Така програмата се записва в бавен режим. Компютърът Правец—8Д записва еднакво надеждно и в бавен, и в бърз режим — струва си да се правят копия на всяка по-ценна програма и по двата начина.

Записаните с команда CSAVE "prg".AUTO програми се стартират автоматично, след като се заредят в паметта на компютъра.

Възможно е да се запише на касета и съдържанието на произволен блок от паметта. Блокът се посочва с буквата A, следвана от началния адрес (шестнадесетично или десетично число) и буквата E, следвана от крайния адрес. Като пример може да посочим как се запазват върху касета текстови или графични страници:

за режим HRES:

CSAVE "ПРГ",A40960,E48000

за режим TEXT или LORES:

CSAVE "ПРГ",A48000,E49119

РАБОТА С МАСИВИ

DIM

Формат: DIM *v(n, m,...)*

Код: 147

Командата описва (дефинира) масиви от данни, като запазва в паметта област за елементите на масива. Това улеснява съхраняването на числа или низове като елементи на едномерен (вектор) или многомерен масив. Масивът *v* може да бъде реален, целочислен или символен. Параметрите *n* и *m* определят горната граница на индексите за размери на масива. **Долната граница е винаги нула.** С една команда DIM могат да се опишат и няколко масива, като се разделят със запетай.

Примери:

DIM P(7) описва реален масив с 8 елемента.

DIM A%(3) описва целочислен масив с 4 елемента.

DIM D\$(2) описва символен масив с 3 елемента.

Области с размер до 11 елемента могат да се запазят и без команда DIM. Присвояването на стойност на един елемент, например

LET N(4)=7

автоматично ще дефинира област за масив N, съдържащ 11 елемента — от N(0) до N(10), и ще присвои стойност 7 на петия елемент N(4).

Командата DIM се използва за масиви с по-голям брой елементи. Масивите се оразмеряват винаги в началото на програмата. Веднъж описан с DIM, даден масив не може да бъде променян в рамките на програмата. В противен случай компютърът ще издаде съобщение за грешка REDIM D ARRAY.

В следващата програма се използва DIM, за да се опише масив D\$ от 8 елемента. На всеки елемент се присвоява стойност с помощта на DATA и READ. Накрая стойността се показва на экрана с PRINT.

```
10 REM --- DIM ---
20 DIM D$(7)
30 FOR I=1 TO 6
40 READ D$(I)
50 NEXT
60 DATA "ПРАВЕЦ-8Д.", "ЕН ПР", "НА ", "БУ
4, АЗ, "ИМЕР "
70 FOR A=1 TO 5
80 FOR B=A TO 6
```

```

90 IF D$(A) < D$(B) THEN 110
100 T$=D$(A):D$(A)=D$(B):D$(B)=T$
110 NEXT
120 NEXT
130 FOR I=1 TO 6
140 PRINT D$(I);
150 NEXT
160 END

```

В ред 20 на следващия пример е дефиниран двумерен масив от цели числа. Присвоени са стойности на елементите, които след това са отпечатани в таблица от 4 колони и 5 реда.

```

10 REM --- DIM'2 ---
20 DIM N%(2,4)
30 FOR A=0 TO 3
40 FOR B=0 TO 4
50 N%(A,B)=A*10+B
60 NEXT
70 NEXT
80 FOR K=0 TO 4
90 FOR J=0 TO 3
100 PRINT N%(J,K),
110 NEXT
120 PRINT
130 NEXT
140 END

```

0	10	20	30
1	11	21	31
2	12	22	32
3	13	23	33
4	14	24	34

Дължината на един низ в даден масив може да бъде най-много 255 символа. Теоретично максималният допустим брой измерения на един масив е също 255, а броят на елементите зависи от наличната памет. Трябва да се помни, че масивите бързо „изядват“ паметта и затова не бива да се използват, ако не е необходимо.

STORE

Формат: STORE *v*, "prog"[,*S*]
Код: 130

Командата записва на касета съдържанието на масив *v* като самостоятелен файл с име *прг*. Масивът трябва да бъде предварително описан с команда DIM или приет по премълчаване като стандартен масив от 11 елемента; в противен случай компютърът ще издае съобщение за грешка OUT OF DATA. Могат да се записват реални, целочислени или символни масиви.

При записа със STORE се използва същата процедура както при CSAVE. Стандартната скорост при бърз запис 2400 bits/s може по желание да се смени на ниска скорост 300 bits/s (към командата се прибавя операнд *S*). На екрана се появява съобщението SAVING *прг*, последвано от буква, която определя типа на файла: R — за реални числа; I — за цели числа; S — за символен из. В следващата програма е показано как се използват STORE и RECALL. Масивът *A\$* е оразмерен и зареден с низове за демонстрация. След като масивът се запише, променливите се изчистват. Накрая програмата възстановява масива и прочита някои стойности.

```
10 REM --- STORE ---
20 DIM A$(20)
30 FOR F=0 TO 20
40 A$(F) = "НОМЕР " + STR$(F)
50 NEXT
60 PRINT "ПУСНИ КАСЕТОФОНА ЗА ЗАПИС."
70 PRINT "НАТИСНИ КЛАВИШ.":GET A$
80 STORE A$, "МАСИВ", S
90 CLEAR
100 DIM A$(20)
110 PRINT "ПРЕНАВИЙ КАСЕТАТА."
120 PRINT "ПУСНИ КАСЕТОФОНА ЗА ВЪЗПРО
ИЗВЕЖДАНЕ."
130 PRINT "НАТИСНИ КЛАВИШ.":GET A$
140 RECALL A$, "МАСИВ", S
150 PRINT A$(9)
160 PRINT A$(10)
170 END
180 REM
190 REM ВИЖ CLOAD,CSAVE,DIM,RECALL
```

RECALL

Формат: RECALL *v, "прг"[S]*
Код: 131

Тази команда е обратна на STORE. Тя зарежда от касета съдържанието на файл с име *прг*, записан предварително със STORE. Областта, където се зарежда файлът, се определя от масива *v*. Процедурата е същата, както когато се работи с CLOAD. Преди да се използува RECALL, масивът трябва да се оразмери с DIM, в противен случай ще се издаде съобщение за грешка OUT OF DATA. Файлът трябва да бъде от същия тип, като масива *v* (реален, целочислен или символен), със същите размери или по-голям.

Ниска скорост при предаване на данни от касетата може да се определи, като към командата се прибави операнд *S*. Същата скорост трябва да е била използвана и при командата STORE. За пример вж. STORE.

ЛОГИЧЕСКИ ФУНКЦИИ И КОНСТАНТИ

AND

Формат: усл1 AND усл2
Код: 209

Функцията сравнява резултатите от изпълнението на усл1 и усл2 чрез логическо умножение (И). Тя приема стойност – 1 (истина) само ако са изпълнени и двете условия; в останалите случаи стойността ѝ е 0 (неистина). Например, ако $U = 10$, $V = 9$ и $W = 7$, резултатът от логическото действие $U > V$ AND $W < 8$ ще бъде – 1.
 AND може да се използува и в команда IF...THEN.

OR

Формат: усл1 OR усл2
Код: 210

Функцията OR сравнява резултатите от изпълнението на усл1 и усл2 чрез логическо събиране (ИЛИ). Тя получава стойност – 1 (истина),

ако поне едно от условията е изпълнено; в противен случай стойността ѝ е 0 (неистина). Например, ако $U = 10$, $V = 9$ и $W = 7$, резултатът от логическото действие $U < V \text{ OR } W < 8$ ще бъде -1.
OR може да бъде част от оператора IF...THEN, например:

20 IF (A=0) OR (B=1) THEN GOTO 300

Ако поне едно от условията е истина, ще се осъществи безусловен переход към ред 300.

NOT

Формат: NOT усл
Код: 202

Логическата функция NOT действува аналогично на думата НЕ. Тя просто обръща получената от логическия израз стойност „истина“ (-1) в „неистина“ (0) и обратно.

```
10 REM --- NOT ---
20 CLS:A=1
30 INPUT "ЗАДАЙ ЦЯЛО ЧИСЛО";N
40 REPEAT
50 A=A+1
60 IF NOT(A=N) THEN PRINT A
70 WAIT 20:CLS
80 UNTIL A=N
90 PRINT A:EXPLODE
99 REM
100 REM ВИЖ AND, IF, OR
```

TRUE

Формат: TRUE
Код: 239

Вградена системна константа със стойност -1. Тя представя резултата от изчислението на условен израз, който е истина. Стойността за неистина е 0 (вж. FALSE). Всъщност всичко, различно от нула, може да се разглежда като TRUE. Много често за стойност на TRUE се приема 1. При Правец-8D обаче за стойност на константата TRUE с определена -1, т. е. допълнителният код на числото 1111 1111.
Внимание: логическата проверка $TRUE = 1$ ще даде 0 (FALSE), докато $TRUE = -1$ ще даде -1 (TRUE). Двете константи — TRUE и

FALSE, се използват в комбинация, за да направят програмата поясна, най-вече при работа с флагове. В програмирането е прието да наричаме „флагове“ такива битове, които със стойност 0 или 1 показват изпълнението на едно от две възможни условия.

Следващата програма демонстрира използването на TRUE и FALSE за управяване на цикъл REPEAT...UNTIL

```
10 REM --- TRUE ---
20 FLAG=FALSE
30 PRINT "ВЪВЕДИ 4-БУКВЕНА ДУМА"
40 REPEAT
50 INPUT A$
60 IF LEN(A$)=4 THEN FLAG=TRUE
70 UNTIL FLAG=TRUE
80 REM
90 REM ВИЖ FALSE
```

FALSE

Формат: FALSE
Код: 240

Логическа системна константа „неистина“, която има стойност 0. Използува се заедно с логическата константа TRUE („истина“):

```
10 REM --- FALSE ---
20 FOR A=11 TO 20
30 IF A<13 THEN PRINT TRUE ELSE PRINT
FALSE
40 NEXT
50 END
```

TRUE и FALSE онагледяват резултата от оценката на даден логически израз. Например програмата

```
10 REM --- FALSE'2 ---
20 A=10:B=1
30 REPEAT
40 PRINT (A>B):B=B+1
50 UNTIL FALSE
60 REM
70 REM ВИЖ TRUE
```

ще отпечатат 9 пъти – 1 (което е представата на компютъра за истина), преди да престане да бъде вярно условието в ред 40 (т. е. че A е по-голямо от B). Когато A стане равно на B ($10 = 10$), компютърът ще започне да извежда безкраен ред от нули. Въпреки че в ред 50 константата FALSE може да бъде заместена с нула, използването ѝ илюстрира много по-добре какво се извършва в програмата.

МАТЕМАТИЧНИ ФУНКЦИИ

ABS

Формат: ABS (n)
Код: 216

Вградена функция, която изчислява абсолютната стойност на аргумента n. Например ABS(-21) е 21. Тази функция се използва, когато трябва да се осигури положителна стойност на резултата. Например резултатът от израза (A–B) ще бъде положителен, докато A е по-голямо от B, а ABS(A–B) ще бъде винаги положително число.

```
10 REM --- ABS ---
20 A=COS(PI)
30 IF A=-1 THEN PRINT "COS(ПИ)="";A
40 IF ABS(A+1)<1E-9 THEN PRINT "COS(П
И)="";A;"СПОРЕД ПРАВЕЦ-ВД";
50 PRINT " (";A;"=-1)"
60 REM
70 REM ВМЖ SGN
```

```
COS(ПИ) = -.999999999 СПОРЕД ПРАВЕЦ-ВД
(-.999999999 = -1)
```

SGN

Формат: SGN (n)
Код: 214

Вградена функция, която дава:

- – 1, ако аргументът n < 0;
- 0 — ако n = 0;
- 1 — ако n > 0.

В следващата програма се използва функцията SGN, за да се определи знакът на въвежданата стойност. Към резултата се прибавя 2, за да се подадат числата 1, 2, 3 на оператора ON...GOSUB в ред 40, който предава управлението на съответната подпрограма за извеждане знака на числото.

```
10 REM --- SGN ---
20 INPUT "ВЪВЕДИ ЧИСЛО"; N
30 PRINT "ЧИСЛОТО БЕШЕ ";
40 ON SGN(N)+2 GOSUB 100,200,300
50 PRINT "НАТИСНИ КЛАВИШ ЗА ОПИТ С ДР
УГО ЧИСЛО"
60 GET M$
70 GOTO 20
100 PRINT "ОТРИЦАТЕЛНО."
110 RETURN
200 PRINT "НУЛА."
210 RETURN
300 PRINT "ПОЛОЖИТЕЛНО."
310 RETURN
399 REM
400 REM ВИЖ ABS
```

INT

Формат: INT (*n*)
Код: 215

Вградена функция, която определя най-голямото цяло число, по-малко или равно на *n*. Фактически INT превръща едно реално число в цяло. Например програмата

```
10 REM --- INT ---
20 Z=INT(10.747)
30 PRINT Z
```

ще отпечата 10. Трябва да се внимава при работа с отрицателни числа. Например

```
10 REM --- INT'2 ---
20 A=-10.56
30 PRINT INT(A)
```

ще даде, разбира се, -11.

SIN

Формат: SIN (*n*)
Код: 227

Вградена функция, която изчислява синуса на ъгъл, зададен с параметъра *n* в радиани.

Превръщането от градуси в радиани и обратно е просто, като се използва вградената функция PI. Градусите се умножават с коефициент PI/180, за да се превърнат в радиани, а радиантите — с коефициент 180/PI, за да се превърнат в градуси.

Първата програма дава стойностите на синусите на ъгли от 0 до 360° със стъпка 10. Ред 30 превръща градусите в радиани, а ред 40 изобразява ъгъла и резултата от прилагането на SIN към стойността в радиани.

```
10 REM --- SIN ---
20 FOR A=0 TO 360 STEP 10
30 RAD=A*PI/180
40 PRINT A,SIN(RAD)
50 NEXT
60 REM
70 REM ВИН ATN,COS,PI,TAN
```

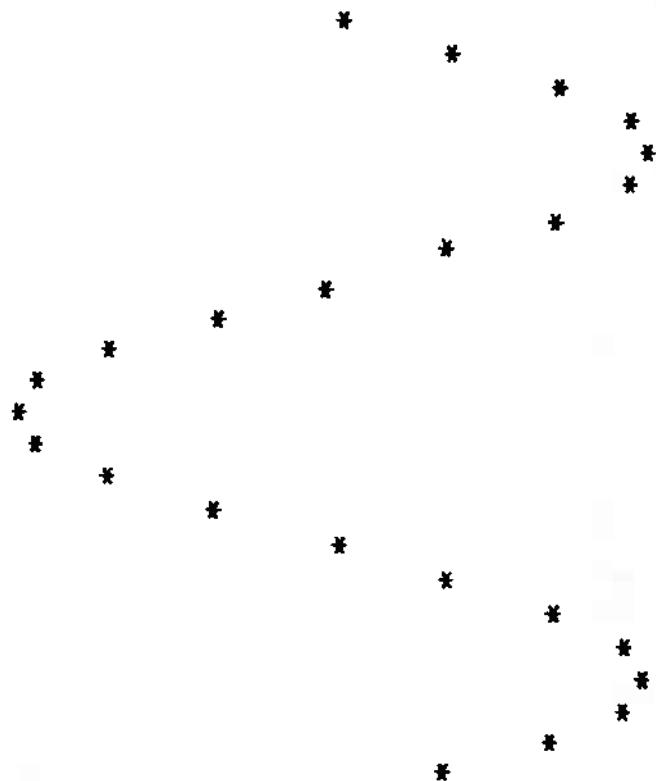
Резултатът ще се движи между 0 и 1, без да достигне нито една от тези стойности, поради неточност на многократните изчисления. Ако се смени ред 40, както е показано по-долу, резултатът ще се закръгли до 5 десетични знака:

```
10 REM --- SIN'2 ---
20 FOR A=0 TO 360 STEP 10
30 RAD=A*PI/180
40 PRINT A,INT(SIN(RAD)*1E4+0.5)/1E4
50 NEXT
```

Следващата програма използва TAB в комбинация със SIN. Тя изчислява позиция за извеждане на звездичка, с която се чертае на екрана кривата на функцията. Тъй като върнатата от SIN стойност варира между 0 и 1, изразът 18*SIN(T) в ред 30 дава стойност между -18 и +18, която се прибавя към позицията на TAB, за да промени разположението на звездата по экрана.

```
10 REM --- SIN'3 ---
20 FOR T=0 TO 8*PI STEP PI/8
```

```
30 PRINT TAB(20+10*SIN(T));"**"
40 NEXT
```



COS

Формат: COS (n)
Код: 226

Вградена функция, която изчислява косинуса на ъгъл, зададен с аргумента n в радиани. Коефициентите за превръщане на радианите в градуси и обратно са дадени в описанието на функцията SIN. Следващият пример показва разликата между кривите на синуса и косинуса:

```
10 REM --- COS ---
20 HIRES : PRINT : PRINT : PRINT
```

```

30 INPUT "КОСИНУС ИЛИ СИНУС (COS/SIN)
";A$
40 INPUT "СТОЙНОСТ (1 ДО 99) ";V
50 CURSET 0,100,3 : DRAW 239,0,1
60 FOR A=40960 TO 49079 STEP 40
70 POKE A,INT(RND(1)*2)+16
80 NEXT
90 FOR A=-PI TO PI STEP .02
100 IF A$="COS" THEN B=COS(A)
110 IF A$="SIN" THEN B=SIN(A)
120 CURSET A*38+120,(B*V)+99,1
130 NEXT
140 PRINT A$" КРИВА":WAIT 100
150 INPUT "ИЗТРИВАНЕ НА ЕКРАНА Д/Н":M
*
160 IF M$="Н" THEN 30
170 GOTO 20
180 REM
190 REM ВИЖ ATN, PI, SIN И TAN

```

TAN

Формат: TAN (n)
Код: 228

Върденена функция, която дава стойността на тангенса на ъгъл, зададен с аргумента *n* в радиани. Резултатът е еквивалентен на SIN (*n*)/COS (*n*). За превръщане на градусите в радиани и обратно вж. описанието на SIN.

Примерната програма просто изобразява таблица със стойности на TAN за ъгли от 0 до 360° (2π радиана).

```

10 REM --- TAN ---
20 DEF FN R(GR)=GR*PI/180
30 FOR D=0 TO 360
40 P=D/20: P%=D/20
50 PRINT D,TAN(FN R(D))
60 IF P=P% THEN WAIT 100
70 NEXT
80 REM
90 REM ВИЖ ATN,COS,SIN,PI

```

ATN

Формат: ATN (*n*)
Код: 229

Вградена функция, която дава стойността на ъгъл с тангенс, равен на аргумента *n*. Стойността винаги е в радиани и се намира в границите от $-\pi/2$ до $\pi/2$. Така ATN(1) има резултат $\pi/4$ или 0.785398162.

```
10 REM --- ATN ---
20 HIRRES
30 INPUT "КООРДИНАТИ (X1,Y1,X2,Y2)":X
1,Y1,X2,Y2
40 CURSET X1,Y1,3
50 DRAW X2,Y2,1:WAIT 100
60 AN=ATN((Y2-Y1)/(X2-X1)):TEXT
70 PRINT "ЪГЪЛЪТ Е"AN"РАДИАНА"
80 PRINT " ИЛИ"AN*180/PI"ГРАДУСА"
90 END
100 REM
110 REM ВИЖ COS, PI, SIN И TAN
```

SQR

Формат: SQR (*n*)
Код: 222

Вградена функция, която изчислява квадратен корен от *n*. Аргументът *n* може да бъде аритметичен израз, но трябва да има положителна стойност, в противен случай ще се издаде съобщение за грешка ILLEGAL QUANTITY.

Тази програма извежда числата от 30 до 20 и техния квадратен корен.

```
10 REM --- SQR ---
20 FOR N=30 TO 20 STEP -1
30 PRINT N,SQR(N)
40 NEXT N

30      5.47722558
29      5.38516481
28      5.29150263
27      5.19615243
```

26	5.09901951
25	5
24	4.89897949
23	4.79583153
22	4.69041576
21	4.5825757
20	4.47213595

EXP

Формат: EXP(*n*)
Код: 225

Вградена функция, която изчислява e^n ($e = 2.7183$). Тя често се използва в комбинация с функцията LN, тъй като EXP (LN(*n*)) дава стойността на антилогаритъма.

```

10 REM --- EXP ---
20 HIRES
30 CURSET 30,30,0
40 DRAW 0,150,1:DRAW 190,0,1
50 FOR N=0 TO 5 STEP .025
60 X=N*40+30
70 Y=180-EXP(N)
80 CURSET X,Y,1
90 NEXT
100 END
110 REM
120 REM ВИЖ LN, LOG

```

LOG

Формат: LOG(*n*)
Код: 232

Вградена функция, която изчислява десетичния логаритъм на *n* (при Правец—82 няма такава функция). Аргументът *n* може да бъде число или израз с положителна стойност, в противен случай LOG(*n*) ще бъде имагинерно число.

```

10 REM -- LOG --
20 CLS
30 INPUT "ВЪВЕДИ ЦИФРА (1-9)";N

```

```

40 FOR A=1 TO 10
50 C=INT(RND(1)*M+1)
60 PRINT "ЛОГАРИТЪМ ОТ ";N*A+C;
70 PRINT " Е ";LOG(N*A+C)
80 NEXT
90 REM
99 REM ВИЖ EXP, LN

```

LN

Формат: $\text{LN}(n)$
Код: 224

Това е една полезна вградена функция на Правец—8Д. Тя изчислява натурални логаритми, или по-точно казано, логаритми с основа e . Обратната функция е $\text{EXP}(\text{LN}(n))$. За операциите с натурални логаритми важат същите правила, както и за операциите с обикновени логаритми. Например изразът

$$\text{EXP}(\text{LN}(x) + \text{LN}(y))$$

ще изчисли произведението на x и y .

RND

Формат: $\text{RND}(n)$
Код: 223

Без тази вградена функция много от компютърните игри щяха да бъдат скучни. Тя осигурява елемент на случайност, като генерира случайно число между 0 и 1 (може да се получи точно 0, но не точно 1). Действието на RND зависи от стойността на аргумента n в скобите. Нормалното използване на функцията е $\text{RND}(1)$. В този случай тя определя случайно число между 0 и 1. Това само по себе си не е особено полезно, както може да се установи, ако се въведе няколко пъти като директна команда $\text{PRINT RND}(1)$.

Обикновено случайните числа трябва да бъдат в определен интервал, за да се симулира например хвърлянето на зарове, или да се осигури координата X в границите между 1 и 30 и др.

Това може да се постигне, като се умножи резултатът от $\text{RND}(1)$ на определено число, което ще промени интервала, например от 0 до 5.999999 (ако се умножава по 6). Ако към резултата се прибави 1 и след това се закръгли до цяло число, наистина ще се получи имитация на хвърляне на зар. Следващата програма показва два еквивалентни метода за закръгляване надолу. Единият използва INT, а другият

просто присвоява стойността на израза $RND(1)*6+1$ на целичислена-
та променлива N%:

```
10 REM --- RND ---
20 A=RND(1)*6+1
30 N%=A
40 R=INT(A)
50 PRINT R
60 PRINT N%
```

Резултатът от RND(0) е по-предсказуем. Функцията дава стойността
на последното случаен число, генерирано преди RND(0):

```
10 REM --- RND'2 ---
20 FOR R=1 TO 50
30 PRINT RND(0)
40 NEXT
```

Понякога е полезно да се генерира няколко пъти една и съща пореди-
ца от случаен числа, например при последователно тестване на про-
грама, ползваща RND. За да се сравняват резултатите от отделни
изпълнения на програмата, трябва да се получава поредица от едни
и същи случаен числа. Това може да се постигне, като се зададе
отрицателна стойност на аргумента на RND. Примерът по-долу по-
казва, че зареждането на генератора с RND(-4) ще създава винаги
една и съща поредица от числа.

```
10 REM --- RND'3 ---
20 FOR K=1 TO 2:PRINT
30 S=RND(-4)
40 FOR F=1 TO 6
50 PRINT RND(1)
60 NEXT F
70 NEXT K
```

PI

Формат: PI
Код: 238

Вградена константа със стойност $\pi = 3.14159265$. В краткия пример по-
долу с PI се изчислява лицето на кръг.

```
10 REM --- PI ---
20 CLS
```

```

30 FOR V=1 TO 5
40 R=INT(RND(1)*30+1)
50 A=PI*R^2
60 PRINT "ПРИ РАДИУС" R "ЛЪЦЕТО НА КРЪГ
A E*A
70 PRINT:PRINT
80 NEXT

```

DEF FN

Формати: DEF FN $v(p)$ = изр

DEF USR a

Код: 184

Командата се използва за дефиниране на аритметични функции. Компютърът разполага с богат набор от вградени функции, но команда DEF FN позволява да се „вграждат“ в програмата нови функции. Така може да се избегне многократното изписване на един и същи израз.

Параметърът v е име на функцията и се подчинява на правилата за имена на променливи. Под това име ще се извиква функцията, като се използва FN $v(p)$ по-нататък в програмата. Скобите са задължителни, като в тях се дава име p , което се използва за аргумент на функцията. Прието е то да се нарича формален параметър, като в действителност е обикновено име на променлива. Следва знак за равенство, след който се описва изразът, който включва p (формалния параметър). Изразът определя как ще се изчислява стойността на функцията за различните стойности на аргумента. В примера стойността на аргумента A се определя в цикъл.

```

10 REM --- DEF FN ---
20 DEF FN M(Z)=Z/6*2
30 FOR A=10 TO 100 STEP 10
40 PRINT FN M(A)
50 NEXT

```

Използува се и друг формат — DEF USR a , за да се дефинира началният адрес a на зададена от потребителя подпрограмма на машинен език (вж. USR):

```

10 REM --- DEF USR ---
20 DEF USR=5120
30 FOR I=0 TO 21
40 READ A: POKE 5120+I,A
50 NEXT

```

```
60 DATA 162,9,189,12,20,157,128,187
65 DATA 202,208,247,96
70 DATA 0, #70, #72, #61, #77, #65
75 DATA #63, #2D, #38, #64
80 REPEAT
90 DUMMY=USR(0)
100 WAIT 50
110 FOR I=48001 TO 48009
120 POKE I,32 : WAIT 10
130 NEXT
140 WAIT 50
150 UNTIL FALSE
```

УПРАВЛЕНИЕ

GOTO

Формат: GOTO *n*
Код: 151

Нормално последователността, в която се изпълнява една програма на БЕЙСИК, се определя от номерата на операторите — от най-малкия към най-големия. Командата GOTO прекъсва тази последователност и предава управлението на оператор с номер *n* (затова се нарича команда за безусловен преход).

Почти във всички сериозни книги се подчертава, че честото използуване на оператори GOTO е признак на лошо структурирана програма. Многократното включване на GOTO създава истински хаос в логиката на програмата и затруднява откриването на грешки.

Вместо с абсолютен номер на ред *n* командата работи и с числова променлива или израз, например:

GOTO A

GOTO A+B

Ето и една малка програма:

```
10 REM --- GOTO ---
20 CLS:GOTO 40
30 END
40 GOTO 1000
```

```
50 PRINT "ПРОГРАМА ";
60 GOTO 90
70 PRINT "ЛОМО"
80 GOTO 200
90 PRINT "ДЕМОНСТРИРА ";
100 GOTO 70
200 PRINT "ИЗПОЛЗВАНЕ ";
210 GOTO 1020
300 PRINT "GOTO! "
310 GOTO 30
1000 PRINT:PRINT:PRINT"ТАЗИ ";
1010 GOTO 50
1020 PRINT "НА ";
1030 GOTO 300
2000 REM
2001 REM ВИЖ GOSUB, ON
```

GOSUB

Формат: GOSUB *n*
Код: 155

Това е една от най-полезните команди в БЕЙСИК. Подобно на GOTO тя предава управлението на ред *n* — предизвиква безусловен переход. За разлика от GOTO GOSUB предава управлението временно, докато се изпълни дадена програма. Подпрограмата трябва да завърши с оператор RETURN, който връща управлението в основната програма. Адресът за връщане се записва от GOSUB в специална памет на компютъра, наречена стек (вж. описание на POP).

Нормалната последователност за предаване на управлението между основната програма и няколко подпрограми може да се наруши само с оператора POP. Той работи пряко със съдържанието на стека, като премахва адрес за връщане.

Вместо номер на ред GOSUB може да сочи променлива или арифметичен израз.

Правилното използване на подпрограми може значително да увеличи ефикасността и яснотата на програмата.

```
10 REM --- GOSUB ---
20 CLS:C=0
30 PRINT "ВЪВЕДИ ЦЯЛО ЧИСЛО МЕЖДУ 1 И
22"
40 INPUT N
50 IF N>22 OR N<1 THEN GOSUB 250
```

```

60 IF N<>INT(N) THEN GOSUB 250
70 IF C=1 THEN 20
80 GOSUB 100
90 PRINT F:GOTO 310
100 REM ** ПОДПРОГРАМА **
110 IF N<>1 THEN 140
120 F=1
130 GOTO 180
140 N=N-1
150 GOSUB 100
160 F=F*(N+1)
170 N=N+1
180 RETURN
190 REM ** КРАИ ** 
200 GOTO 310
250 REM ** ПОДПРОГРАМА №.2 **
260 CLS:C=1
270 PRINT "СПАЗВАЙ ИНСТРУКЦИИТЕ!"
280 WAIT 300
290 RETURN
300 REM ** КРАИ №.2 **
310 END
400 REM
410 REM ВИЖ GOTO, ON, RETURN

```

ON

Формат: ON усл GOTO n1, n2,...

Код: 180

ON усл GOSUB n1, n2,...

Тази команда трябва да се комбинира с GOTO или с GOSUB. Тя улеснява разклоняването на програмата. Използва се обикновено като управляваща структура в програми с меню. Така на потребителя се предлагат няколко възможности, които се обработват от различни части на програмата. Това се илюстрира със следния пример:

```

10 REM --- ON ---
20 CLS
30 INPUT "ВЪВЕДИ 1,2 ИЛИ 3";N
40 ON N GOTO 100,200,300
50 REM

```

```

100 PRINT "РЕД 100 ЗАРАДИ N="; N; GOTO 30
150 REM
200 PRINT "РЕД 200 ЗАРАДИ N=2"; GOTO 30
250 REM
300 PRINT "РЕД 300 ЗАРАДИ N=3"; GOTO 30
399 REM
400 REM ВИЖ GOSUB, GOTO, RETURN

```

Ако в ред 30 се въведе 3, управлението ще се предаде на третия номер на ред (300), посочен в ред 40. Ако $n = 2$, програмата ще „скочи“ на втория номер на ред и т. н.

Ако n е по-голямо от броя на посочените редове, програмата просто ще продължи с оператора, който следва след реда, съдържащ ON. Отрицателна стойност на n в INPUT ще предизвика съобщение за грешка. Нецелите стойности се закръгляват автоматично.

IF... THEN... [ELSE]

Формат: IF усл THEN on1; on12;...
 /ELSE on21; on22;.../
Код: 153, 201, [200]

Този формат е известен като структура на решенията. Той се използва за проверка на условия и за управляване на последователността, в която се изпълняват операциите. Елементът ELSE е ограден в квадратни скоби, тъй като не е задължителен в този формат. Оператори могат да бъдат всякакви инструкции на БЕЙСИК при условие, че се събират в максималната дължина на програмния ред. Ако усл е вярно, изпълняват се операторите след THEN. След това управлението се предава на следващия ред, като се прескача ELSE. Операторите след ELSE се изпълняват, ако усл не е вярно. В този случай се прескачат операторите след THEN.

Тази структура на решенията е една от най-силните възможности на БЕЙСИК. Използването ѝ е показано ясно със следния пример:

```

10 REM --- IF/THEN/ELSE ---
20 CLS
30 INPUT "ВЪВЕДИ ИМЕТО СИ"; N$
40 CLS
50 INPUT "МЪЖ ИЛИ ЖЕНА (M/N)"; S$
60 CLS
70 PRINT "ЗДРАВЕИ! ТИ СИ ";
80 IF S$="M" THEN PRINT "ИНТЕЛИГЕНТЕН"
; ELSE PRINT "ПРЕКРАСНА";

```

```

90 PRINT ", "IN$;" !"
100 END
110 REM
120 REM ВИЖ AND, GOTO, NOT, OR, ON

```

GOSUB трябва да се използува в стандартния си формат в конструкцията IF... THEN... ELSE, а операторът GOTO може да се пропусне и да се даде само номер на реда. GOTO може да замества THEN.

FOR... TO... [STEP]... NEXT

Формат: FOR v=x TO y [STEP z]... NEXT v
Код: 141, 195, [203], 144

Голяма част от работата на компютрите е свързана с повтарянето на прости задачи. Организирането на цикли е едно от най-ефективните средства за тази цел. Най-често срещаната реализация на такива конструкции е цикълът FOR...NEXT. Цикълът FOR...NEXT кара компютъра да изпълнява съдържащите се в него операции определен брой пъти, например:

```

10 REM --- FOR ---
20 FOR A=1 TO 3
30 PRINT A
40 NEXT A

```

Този цикъл ще изведе числата 1, 2 и 3. При първото минаване през цикъла компютърът присвоява стойност 1 на променливата A и я извежда на екрана в ред 30. Операторът NEXT A в ред 40 връща управлението към ред 20. Там стойността на A се увеличава с 1 и цикълът се повтаря. Стойността на променливата нараства с 1, докато стане равна на зададената след TO стойност (проверката се прави при NEXT). Тогава цикълът приключва, като изпълнението на програмата продължава с реда след оператора NEXT. За стъпка, различна от 1, трябва да се използува операторът STEP:

```

10 REM --- FOR'2 ---
20 FOR X=5 TO 20 STEP 5
30 PRINT X
40 NEXT

```

Този цикъл ще изведе на екрана числата 5, 10, 15 и 20. NEXT ще изпълни задачата си и при пропуснато име на променлива. Посочването на име все пак е препоръчително — така се постига по-голяма яснота, особено при вграждане на цикли.

Възможно е стойността на STEP да бъде отрицателна, както и десетично число:

```
10 REM --- FOR'3 ---
20 FOR A=20 TO 5 STEP -5
30 PRINT A
40 NEXT A
```

ИЛИ

```
10 REM --- FOR'4 ---
20 FOR F=1 TO 2 STEP .2
30 PRINT F
40 NEXT F
```

Началните и крайните стойности в цикъла, както и стъпката могат да бъдат цели числа, променливи или числови изрази, например:

```
10 REM --- FOR'S ---
20 A=56:C=PI/2
30 FOR X=2^4 TO A/3 STEP C
40 PRINT X
50 NEXT X
60 REM
70 REM ВИЖ REPEAT, UNTIL
```

REPEAT

Формат: REPEAT
Код: 139

При комбиниране с UNTIL тази команда създава цикъл, който кара компютъра да повтаря серия от инструкции, докато се изпълни определено условие. За разлика от циклите FOR...NEXT тук няма брояч, който да нараства. Броячът трябва да се програмира в цикъла, ако е необходим (както е в примера по-долу).

Единственият начин да се излезе от цикъл REPEAT...UNTIL преди нормалното му завършване е с командата PULL.

```
10 REM --- REPEAT ---
20 HIRES:INK 4:PAPER 0
30 A=0
40 REPEAT
50 CURSET 55+A,110-A,3
60 DRAW 2+A,0,1:DRAW 0,A+A,1
```

```
70 A=A+1  
80 UNTIL A>60  
89 REM  
90 REM ВИЖ FOR,NEXT,PULL,UNTIL
```

UNTIL

Формат: UNTIL усл
Код: 140

Командата е част от структурата на цикъла REPEAT...UNTIL. Условният израз се изчислява и ако се установи, че е „истина“, управлението на програмата ще премине към следващия операнд и ще се излезе от цикъла. Ако условието е „неистина“, управлението ще се върне обратно към оператора, следващ REPEAT, който инициализира цикъла. Ако не намери REPEAT, компютърът ще издаде съобщение за грешка ?BAD UNTIL ERROR.

PULL

Формат: PULL
Код: 136

Командата PULL премахва от стека (вж. POP) един адрес за връщане от цикли REPEAT...UNTIL. Действието ѝ е аналогично на действието на POP за конструкцията GOSUB...RETURN.

Практически PULL представлява авариен изход от цикли и се използва при попадане в неясна ситуация.

```
10 REM --- PULL ---  
20 A=5  
30 REPEAT  
40 : B=A  
50 : REPEAT  
60 : PRINT B;  
70 : B=B-1  
80 : IF B<0 THEN PULL:GOTO 100  
90 : UNTIL B=0  
100 :A=A-1  
110 :PRINT  
120 UNTIL A=-5  
130 REM
```

140 REM ВИЖ POP, REPEAT, UNTIL

```
9 8 7 6 5 4 3 2 1  
8 7 6 5 4 3 2 1  
7 6 5 4 3 2 1  
6 5 4 3 2 1  
5 4 3 2 1  
4 3 2 1  
3 2 1  
2 1  
1  
0  
-1  
-2  
-3  
-4
```

RETURN

Формат: RETURN
Код: 156

Тази команда трябва да се използва като последен оператор във всяка подпрограма. Тя показва, че компютърът е стигнал до края на подпрограмата и управлението трябва да се върне към оператора, следващ командата GOSUB.

Адресът за връщане на RETURN може да се промени само с команда POP.

```
10 REM --- RETURN ---  
20 CLS  
30 PRINT "ВЪВЕДИ РАДИУС";  
40 INPUT R  
50 C=2*PI*R:Z=C  
60 GOSUB 200  
70 CLS  
80 PRINT "ОБИКОЛКАТА Е" Z  
90 A=PI*R^2:Z=A  
100 GOSUB 200  
110 PRINT "ЛИЦЕТО Е" Z  
120 GOTO 300
```

```
200 ' ПОДПРОГРАМА ЗА ЗАКРЪГЛЯВАНЕ
210 ' ДО 2 ЗНАКА
220 Z=INT(100*(Z+.005))
230 Z=Z/100
240 RETURN
300 END
399 REM
400 REM ВИЖ GOSUB, POP
```

POP

Формат: POP
Код: 134

Командата премахва адрес за връщане (RETURN—адрес) от стека. При извикване на подпрограма компютърът съхранява адреса за връщане в специална област на паметта, наречена *стек*. Извикването на всяка следваща подпрограма добавя към стека следващ адрес. Когато програмата срещу оператор RETURN, тя взема от стека последния постъпил адрес.

Командата POP е единственото средство за промяна в строгата последователност при изпълнение на вгнездени една в друга подпрограми. Например, ако нашата програма извиква подпрограма A, която от своя страна извиква подпрограма B, естественият ред изисква, след като се изпълни B, управлението да се върне на A, а след като се изпълни и A, да се предаде обратно в основната програма. Командата POP ще позволи преход от B към основната програма (компютърът игнорира най-горния адрес в стека и взема следващия адрес — за връщане от A към програмата).

В много отношения на POP трябва да се гледа като на крайно нежелателна команда. При добре планирана и структурирана програма не би трябвало да се налага авариен изход от вложена подпрограма.

```
10 REM --- POP ---
20 FOR I=-5 TO 5:PRINT I,
30 GOSUB 100
40 PRINT:NEXT
50 END
100 GOSUB 200
110 PRINT X
120 RETURN
200 IF I>0 THEN X=LOG(I) ELSE POP
210 RETURN
```

-5	
-4	
-3	
-2	
-1	
0	
1	7.00890077E-11
2	.301029996
3	.477121255
4	.602059991
5	.698970004

STOP

Формат: STOP
Код: 179

Командата спира изпълнението на програмата, като издава съобщение за прекъсване, в което посочва и номера на реда (BREAK IN ...). Програмата може да се стартира повторно с команда CONT, последвана от RETURN. Командата STOP действува като END, но след END програмата не може да бъде стартирана повторно, както показва следният пример:

```

10 REM --- STOP ---
20 FOR X=1 TO 3
30 PRINT "X=";X
40 PRINT "ПРОГРАМАТА Е СПРЯНА."
50 PRINT "НАПИШИ CONT ЗА ПРОДЪЛЖЕНИЕ.
*
60 STOP
70 NEXT X
80 END
90 PRINT "ТОЗИ РЕД НЯМА ДА СЕ ИЗВЕДЕ,
ЗАЩОТО CONT НЕ ДЕЙСТВА СЛЕД END."
100 REM
110 REM ВИХ END,WAIT,CONT

```

CONT

Формат: **CONT**
Код: 187

Тази директна команда рестартира програмата след прекъсване. При създаване на една програма се налага да се използува **CTRL—C**, за да се спре изпълнението и да се разгледа еcranът или състоянието на дадена променлива. В много случаи изпълнението на програмата трябва да продължи от точката на прекъсване. Ето кога е полезна командата **CONT**.

Командата **CONT** не разрешава рестартиране на програмата, ако някоя нейна част е променена. Тя трябва да се използува *само като директна команда*. Включването ѝ в програма ще предизвика грешка и прекъсване.

END

Формат: **END**
Код: 128

Командата се използува за спиране на програмата. За разлика от **STOP** тя не издава съобщение за номера на последния изпълнен ред (**BREAK IN...**).

Обикновено командата **END** е последен оператор в програмата. В отделни случаи тя се използува и като средство за прекъсване при определено условие, например:

```
10 REM --- END ---
20 REPEAT
30 INPUT "ВЪВЕДИ ЧИСЛО"; NO
40 IF NO<=0 THEN END
50 PRINT "НАТУРАЛЕН ЛОГАРИТЪМ ОТ" NO "Е
*LN(NO)
60 UNTIL FALSE
70 REM
80 REM ВИЖ STOP
```

ОПЕРАЦИИ С НИЗОВЕ

LEN

Формат: LEN (*s*)
Код: 233

Вградена функция, която изчислява броя на символите (дължината) на низ *s*. Например

```
10 REM --- LEN ---
20 J=LEN("ПРАВЕЦ-8Д")
30 PRINT J
```

ще даде 9, защото в низа "ПРАВЕЦ—8Д" има 9 символа, а

```
10 REM --- LEN^2 ---
20 A$="НПК ПО МПТ"
30 PRINT LEN(A$)
```

ще изведе 10, защото този път низът включва 8 символа и 2 интервала.

Тази функция е полезна, например когато трябва да се форматира еcran, който да съдържа въведени от потребителя на програмата низове. Дължината на низовете е променлива и естествено не е известна. Функцията LEN може да се използва и в конструкции FOR...NEXT, когато трябва да се изпълни някаква операция с всяка буква от низа. Например

```
10 FOR I=1 TO LEN(X$)
```

Празен или нулев низ има дължина нула.

LEFT\$

Формат: LEFT\$ (*s, n*)
Код: 244

Вградена функция за работа с низове. Подобно на MID\$ и RIGHTS\$ тя дава възможност да се извличат символи от даден низ. В случая се извличат левите *n* символа от низа *s*. Показаната програма ще отдели и изобрази всички възможни комбинации от символи от низа "ПРАВЕЦ—8Д". Ако *n* е по-голямо от броя на символите в низа, функцията ще върне целия низ.

```
10 REM --- LEFT$ ---  
20 A$="ПРАВЕЦ-8Д"  
30 FOR I=1 TO LEN(A$)  
40 B$=LEFT$(A$, I)  
50 PRINT B$  
60 NEXT I  
70 END  
80 REM  
90 REM ВИЖ RIGHT$, MID$, LEN
```

П
ПР
ПРА
ПРАВ
ПРАВЕ
ПРАВЕЦ
ПРАВЕЦ-
ПРАВЕЦ-8
ПРАВЕЦ-8Д

MID\$

Формат: MID\$(*s, m, n*)
Код: 246

Функцията отделя частта от низа *s*, която започва от *m*-тия символ и съдържа *n* символа. С MID\$ могат да се извлечат група символи от произволно място в един низ.

Ако не се посочи стойност за *n*, функцията отделя всички символи след *m*-тия. Същото се получава и ако се посочи *n*, по-голямо от наличните в низа символи.

Функцията MID\$ има резултат празен низ, ако в низа *s* не съществува *m*-ти символ.

В примера се използва MID\$, за да изчисти водещите и крайните интервали.

```
10 REM --- MID$ ---  
20 A$="      ПРАВЕЦ-8Д      "  
30 IF ASC(A$)=32 THEN A$=MID$(A$, 2):G  
OTO 30  
40 IF ASC(RIGHT$(A$, 1))=32 THEN A$=MI  
D$(A$, 1, LEN(A$)-1):GOTO40
```

```
50 PRINT " "; A$; " "
59 REM
60 REM ВИЖ LEFT$, RIGHT$
```

ПРАВЕЦ-ВД

RIGHT\$

Формат: RIGHT\$ (s, n)
Код: 245

Подобно на MID\$ и LEFT\$ тази функция отделя определени символи от низ. В случая RIGHT\$ отделя десните n символа от низа s.
Например програмата

```
10 REM --- RIGHT$ ---
20 A$="ПРАВЕЦ-ВД"
30 FOR I=1 TO LEN(A$)
40 B$=RIGHT$(A$, I)
50 PRINT SPC(LEN(A$)-LEN(B$)); B$
60 NEXT I
70 END
80 REM
90 REM ВИЖ LEFT$, MID$
```

д
вд
-вд
ц-вд
ец-вд
вец-вд
авец-вд
правец-вд
правец-вд

ще отделя посочения брой символи отляво от низа "Правец-
ВД". Ако n е по-голямо от общия брой символи в даден низ, на экрана
просто ще се изведе целият низ.

STR\$

Формат: STR\$ (n)
Код 234

Тази функция преобразува числови стойности в низ. Тя е обратна на функцията VAL, която пък превръща низ от числови символи в число. Могат да се използват експоненти и шестнадесетични числа, които ще се превърнат в стандартни символи (както обикновено биха се появили на екрана). Низът получава първи символ, който е знак, ако числото е отрицателно, или остава празен, ако числото е положително.

```
10 REM --- STR$ ---
20 N=12.34
30 CLS:PRINT STR$(N):WAIT 15
40 PLOT 10,1,STR$(N):WAIT 45
50 PLOT 0,1,STR$(N):WAIT 15
60 PLOT 10,8,STR$(N):WAIT 15
70 X=-23.5
80 PRINT
90 PRINT STR$(X):WAIT 15
100 PLOT 9,2,STR$(X)
110 PRINT "#АЗ СЕ ИЗВЕЖДА КАТО"STR$(#A3)
120 PRINT "1.345E-4 СЕ ИЗВЕЖДА КАТО"STR$(1.345E-4)
130 PRINT "1.23E2 СЕ ИЗВЕЖДА КАТО"STR$(1.23E2)
```

Тази програма използва PLOT, за да позиционира на екрана извежданите символи (със същия успех може да се използува и PRINT@). Функцията STR\$ е полезна не само с възможността за извеждане на символи през PLOT. Превръщайки дадено число в низ, тя позволява „редактирането“ му след това с набора от специални функции за обработване на низове (напр. MID\$, LEFT\$ и др.). Тя е полезна и когато се позиционират символи за екран в режим HIRES, тъй като с CHAR могат да се позиционират само единични символи. Следващата програма илюстрира това, като използува STR\$, за да получи числото като низ. След това взема последователно всеки символ от низа, намира кода му с ASC и го позиционира на екрана. Същата техника се използува и за елиминиране на първия символ, ако числото е положително.

```
10 REM --- STR$'2 ---
20 HIRES
```

```

30 FOR A=1 TO 10 STEP .5
40 A$=STR$(A)
50 GOSUB 100
60 NEXT
70 PRINT "НАТИСНИ КЛАВИШ ЗА КРАИ.":GET
A$
80 TEXT:END
100 ' ИЗВЕЖДАНЕ НА НИЗА В РЕЖИМ HIRES
110 FOR B=1 TO LEN(A$)
120 CURSET B*6+A*6,A*16,0
130 CHAR ASC(MID$(A$,B)),0,1
140 NEXT
150 RETURN
199 REM
200 REM ВИЖ VAL

```

VAL

Формат: VAL (s)
Код: 235

Функцията преобразува низа *s* в число. Първият символ на низа, който ще се „изчислява“, трябва да започва с интервал, знак минус, тире или число, в противен случай се връща нула. След тези символи или след първото число низът се изчислява до първия нецифров символ. Числата могат да се изразяват и в експоненциална форма, но се обработват във вида, в който се извеждат на экрана. Това може да се провери, като се въведе например 1.23E2 в програмата по-долу. Всеки символ, различен от цифра, ще се игнорира от компютъра, след като в низа са намерени числови символи.

```

10 REM --- VAL ---
20 CLS
30 REPEAT
40 INPUT "ВЪВЕДИ НИЗ";A$
50 PRINT "НИЗЪТ ЗАПОЧВА С ЧИСЛОТО";
60 PRINT VAL(A$)
70 PRINT
80 UNTIL A$="КРАИ"
90 END
95 REM
99 REM ВИЖ ASC,STR$

```

CHR\$

Формат: **CHR\$ (n)**
Код: 237

Функцията има за резултат символ, чийто код ASCII е зададен като аргумент. Аргументът *n* трябва да бъде цяло число в интервала 0—255. Използването на тази функция обаче далеч надхвърля възпроизвеждането на стандартния набор от символи. Дори една бегла справка с пълния списък на кодовете ASCII ще разкрие някои от потенциалните възможности на CHR\$. Например следващата програма използва CHR\$, за да зададе атрибутите, които променят основния цвет и цвета на фона за останалата част на реда, на който се извеждат съобщенията.

```
10 REM ---CHR$---
20 PRINT CHR$(131);
30 PRINT CHR$(144);
40 PRINT CHR$(140);
50 PRINT " ЪЛЪТИ МИГАЩИ БЧКВИ "
60 END
70 REM
80 REM ВИЖ ASC, HEX$, STR$ И VAL
```

Функцията CHR\$ се използува също и при отпечатване, за да изпраща управляващи символи за различните функции и възможности на печатащото устройство.

ASC

Формат: **ASC (s)**
Код: 236

Функцията намира десетичната стойност на кода ASCII на първия символ от низа *s*. Всеки символ, достъпен за Правец—8Д, има съответстващ код ASCII.

В примерната програма ASC се използува за контрол при въвеждането на данни. Ред 40 проверява дали първият символ от низа, който е въведен, е в интервала на кодовете между 65 и 90. Тъй като кодовете 65—90 съответствуват на латиница, компютърът ще приеме само тези низове, които започват с буква на латиница.

```
10 REM --- ASC ---
20 CLS
```

```
30 INPUT "ВЪВЕДИ КАКВИ ДА СА СИМВОЛИ"  
1K$:CLS:PRINT:PRINT  
40 IF ASC(K$)>64 AND ASC(K$)<91 THEN  
    PRINT K$  
50 WAIT 100  
60 GOTO 10  
70 REM  
80 REM ВИЖ CHR$, HEX$, STR$ И VAL
```

HEX\$

Формат: HEX\$ (*n*)
Код: 220

Функцията превръща десетичното цяло число *n* в неговия шестнацетишен еквивалент. Например HEX\$(15) ще даде # F (символът # означава шестнадесетична стойност). Десетичното число в снобите може да бъде всяко цяло число между 0 (# 0) и 65535 (# FFFF). Компютърът ще даде съобщение за грешка BAD SUBSCRIPT при опит за превръщане на отрицателни числа или дроби.

ПОМОЩНИ КОМАНДИ И ФУНКЦИИ

PEEK

Формат: PEEK (*a*)
Код: 230

Функцията PEEK извежда стойността на числото, записано в един байт с адрес *a* в паметта. Стойността ще бъде в интервал между 0 и 255. Функцията е обратна на POKE, с която може да се променя съдържанието на един байт в паметта.

```
10 REM --- PEEK ---  
15 CLS  
20 REPEAT  
30 PRINT CHR$(20)  
40 GOSUB 100  
50 WAIT 50  
60 UNTIL FALSE  
70 END
```

```
100 IF PEEK(48039)=83 THEN PRINT@16,3
  !"ЛАТИНИЦА" ELSE PRINT@16,3; "КИРИЛИЦА"
110 RETURN
199 REM
200 REM ВИЖ CALL, DEEK, DOKE, POKE, USR
```

DEEK

Формат: DEEK (*a*)
Код: 231

Тази функция позволява да се разгледа стойността, записана в два байта от паметта с адреси *a* и *a+1*. Тя дава стойност в интервала 0—65 535, запомнена в посочените два байта. Стандартният формат, който използва процесорът 6502 за съхраняване на адреси, са два байта, като първият е младши. DEEK взема стойността, съхранена във втория (стария) байт с адрес *a+1* и я умножава по 256. Към нея прибавя стойността на байта с адрес *a*.
При обяснението на CALL има пример, в който с помощта на DEEK се разглежда съдържанието на адреси от паметта ROM на компютъра. Друг пример за използване на DEEK е възможността да се следи текущото съдържание на HIMEM чрез DEEK (#A6).

POKE

Формат: POKE *a, n*
Код: 185

Командата POKE позволява на програмиста да променя съдържанието на байт в паметта. Тя записва стойността *n* на адрес *a*. Адресът трябва да се намира в интервала 0—65 535, като стойността на *n* е в интервала 0—255. Както адресът, така и стойността на *n* могат да бъдат десетични или шестнадесетични.

Пример за мястото на тази команда е следната програма, която променя състоянието на скрана:

```
10 REM --- POKE ---
20 FOR A=1 TO 5
25 N=ASC(MID$("ОРЛИН", A, 1))
30 POKE 48034+A, N
40 WAIT 100
50 NEXT
60 WAIT 1000
```

```
70 CALL DEEK(#FFFFA)
80 REM
90 REM ВИЖ DEEK, DOKE, PEEK
```

DOKE

Формат: DOKE *a, n*
Код: 138

Тази команда записва цяло число със стойност от 0 до 65 535 в два байта от паметта. Параметърът *a* е адресът на първия от двата байта, в който ще се записва стойността. Параметърът *n* с формат на цяло число е стойността, която ще се записва. Форматът, в който се съхранява, е нормалното за микропроцесора 6502 представяне — младши, следван от старши байт. Например, за да се запише в паметта числото 770, може да се използува командалата:

DOKE 30000,770

Тя ще запише 2 на адрес 30000 и 3 на адрес 30001, тъй като числото 770 се представя като $3 \cdot 256 + 2$.

CALL

Формат: CALL *a*
Код: 191

Командата CALL предава управлението към подпрограмма на машинен език, която започва на адрес *a*. Връщането към БЕЙСИК става с машинната инструкция RTS (връщане от подпрограмма). Използването на тази команда може да доведе до загуба на основната програма, ако адресът е посочен неправилно и не отпраща към началото на подпрограмата.

Ето един пример за потенциалната опасност и сила на тази команда:

CALL DEEK(#FFFC)

Тази команда ще предаде управлението на вектора за „студен старт“ и ще рестартира компютъра, както ако се изключи и включи отново. По подобен начин извикването на адреса, върнат от DEEK (#FFFC), ще предизвика „топъл“ рестарт (както ако се натисне клавиша RESET).

USR

Формат: **USR (a)**
 DEF USR a
Код: **217**

Командата **USR (a)** изпълнява потребителска подпрограма на машинен език, като ѝ предава параметър *a*. Тя позволява достъп до подпрограми на машинен език по време на изпълнение на програми на БЕЙСИК.

Подпрограмата се извика с **USR (a)**. След нейното изпълнение управлението се връща към главната програма (ако естествено сте завършили с **RST**), като резултатът трябва или да се изведе (**PRINT USR (0)**), или да се присвои на променлива (**A=USR(0)**). Ако няма стойности, които трябва да се предадат на подпрограмата на машинен код, тя може да се извика само с **CALL**.

DEF USR a дефинира началния адрес *a* на потребителска подпрограма на машинен език. Този адрес се използва по-късно чрез **USR (a)**.

```
10 REM --- DEF USR ---
20 DEF USR=5120
30 FOR I=0 TO 21
40 READ A: POKE 5120+I,A
50 NEXT
60 DATA 162,9,189,12,20,157,128,187
65 DATA 202,208,247,96
70 DATA 0,#70,#72,#61,#77,#65
75 DATA #63,#2D,#38,#64
80 REPEAT
90 DUMMY=USR(0)
100 WAIT 50
110 FOR I=48001 TO 48009
120 POKE I,32 : WAIT 10
130 NEXT
140 WAIT 50
150 UNTIL FALSE
```

WAIT

Формат: **WAIT n**
Код: **181**

Тази команда прекъсва изпълнението за *n* стотни от секундата. Много съществено е съвместното използуване на **WAIT** с команди за звук.

Така може да се управлява времетраенето на звука за команди PLAY, SOUND и MUSIC.

WAIT може да се използува и за въвеждане на паузи в програмата, за забавяне на изображението на екрана. Важно е да се отбележи, че не може да се прекъсне действието на WAIT през клавиатурата. Изчакване за намеса през клавиатурата може да се постигне с GET и INPUT.

HIMEM

Формат: HIMEM *a*
Код: 158

Командата HIMEM определя адреса *a* като горна граница на паметта, достъпна за програми на БЕЙСИК.

Нормално компютърът сам разпределя колкото е възможно по-ефективно достъпната памет между потребителската програма, нейните променливи и графични страници. При работа с подпрограми на машинен език се налага програмистът да отдели за тях част от паметта за сметка на паметта за програма на БЕЙСИК.

Естествено командата HIMEM трябва да се намира в самото начало на далена програма, като по принцип ползването ѝ винаги трябва да става с повишено внимание.

```
10 REM --- HIMEM ---
20 PRINT"В МОМЕНТА HIMEM Е      "DEEK(#A6)
30 GRAB
40 PRINT"СЛЕД GRAB HIMEM Е      "DEEK(#A6)
50 RELEASE
60 PRINT"СЛЕД RELEASE HIMEM Е "DEEK(#A6)
70 HIMEM 3000
80 PRINT"ИЛИ КОЯ ДА Е СТОЙНОСТ"DEEK(#A6)
90 RELEASE:REM СТАНДАРТЕН HIMEM
100 REM
101 REM ВЪЗN GRAB, RELEASE
```

FRE

Формат: FRE (0)

FRE""

Код: 218

Функцията FRE (0) изчислява броя на свободните байтове от паметта. Функцията FRE"" кара компютъра да извърши т. нар. „прочистване“. Това е просто едно средство да се изчисти паметта от низове (като се започне точно под адреса HIMEM и се продължи в посока надолу). Това е полезно, ако по време на работа с програмата символните променливи се зареждат с нови стойности. Ако новите низове са по-къси, компютърът запазва същия размер на паметта, като допълва по-късите низове с интервали. Ако те са по-дълги, ще трябва да се запазят на друго място, а цялото пространство, което първоначално е било присвоено, ще остане неизползвано. Фактически функцията FRE"" компресира пространството, заето от низове. По този начин се освобождава неизползваното пространство в паметта.

```
10 REM --- FRE ---
20 PRINT FRE(0)
30 A$="A":B$="B"
40 REPEAT
50 A$=A$+A$:B$=B$+B$
60 UNTIL LEN(A$)=128
70 PRINT FRE(0)
80 PRINT "А СЕГА ДА ОСВОБОДИМ ЗАЕТАТА
ПАМЕТ:"
90 A$="":B$=""
100 PRINT FRE("")
```

GRAB

Формат: GRAB

Код: 159

Значителна част от паметта на Правец—8Д е запазена за графичната страница в режим HIRES. Ако се създава дълга програма на БЕЙ-СИК, която не изисква работа в графичен режим, може да се използува и паметта, определена за графика. Това става с помощта на GRAB, която в общия случай се въвежда като директна команда. Тя може да се използва и в тялото на програмата, когато е необходима допълнителна памет за съхраняване на масиви.

След активизирането на командата графичната страница за режим

Hires не може да се използува, докато не се освободи нейната памет (адреси от #9800 до #B400). Това става с RELEASE, която се извежда като директна команда.

```
10 REM --- GRAB ---
20 CLS
30 PRINT "СВОБОДНА ПАМЕТ:"FRE(0)
40 PRINT"АНГАНИРАМЕ ОЩЕ ПАМЕТ С GRAB:
"
50 GRAB
60 PRINTFRE(0)
70 PRINT"А СЕГА ще я ВЪРНЕМ ЗА ИЗПОЛЗ
ВАНЕ В РЕЖИМ HIRES:"
80 RELEASE
90 PRINTFRE(0)
100 REM
110 REM ВИЖ HIRES, RELEASE
```

RELEASE

Формат: RELEASE
Код: 160

Използването на команда GRAB заема областта от паметта, предназначена за графичната страница в режим HIRES. Това се налага при писане на дълги програми, които не използват режима HIRES. Тази памет може да се използува отново за графиката, като се подаде команда RELEASE, която възстановява байтове от #9800 до #B400 на графичната страница.

TRON

Формат: TRON
Код: 132

Командата TRON проследява изпълнението на програмата и помага при откриването на грешки. Тя извежда на екрана номера на всеки изпълнен ред.

Двойката команди TRON и TROFF дава възможност на програмиста да проследи последователността на редовете по време на изпълнение на програмата. В следващия пример има един безкраен цикъл. Командата TRON показва номерата на редовете по реда на изпълнението им.

```
10 REM --- TRON ---
20 CLS
30 TRON
40 FOR A=1 TO 10
50 PRINT A
60 IF A=6 THEN A=1
70 NEXT
80 END
90 REM
99 REM ВМЪКНІ TROFF
```

Ако се въведе ред 65 TROFF, ще се получат номерата на редовете само за първото минаване през цикъла FOR...NEXT.

TROFF

Формат: TROFF
Код: 133

Командата TROFF изключва проследяването на изпълнението, активизирано с TRON.

ВХОДНО-ИЗХОДНИ ОПЕРАЦИИ

INPUT

Формат: INPUT [/s/] v1, v2...
Код: 146

Командата INPUT извежда незадължителния низ *s* на екрана и чака стойности на променливите *v1*, *v2* и т. н. Практически командата позволява да се въвеждат данни от клавиатурата. Тя спира изпълнението на програмата, докато потребителят не въведе дума, буква или число. Ако се въведе например командата

10 INPUT №

тя ще спре програмата, след като изобрази на екрана въпросителен знак. Трябва да се въведат данни и да се натисне клавиша RETURN. В случая вероятно само програмистът може да съобрази какво точно трябва да се въведе. Затова е добре да се вмъкне и някакво съобщение, например:

```
10 PRINT#13,10;"КАК СЕ КАЗВАШ";
20 INPUT NS
30 REM ВИЖ GET,KEY$
```

или

```
10 INPUT "КАК СЕ КАЗВАШ";NS
```

Предимство на първата конструкция е възможността да се позиционира съобщението на произволно място на екрана, докато при втория пример се работи с текущата позиция на маркера.

GET

Формат: GET v
Код: 190

Командата GET спира изпълнението на програмата и чака да се натисне някой клавиши. Стойността на въведенния символ се присвоява на променливата v. Изпълнението на програмата продължава автоматично веднага след като се натисне клавишът (за разлика от команда INPUT, при която трябва да се натисне и RETURN).

GET е много удобна команда при програми с меню, които изискват да се въвежда само по един символ.

Популярните съобщения „ИЗБЕРЕТЕ“ или „НАТИСНЕТЕ ПРОИЗВОЛЕН КЛАВИШ, ЗА ДА ПРОДЪЛЖИТЕ“ използват възможности на командата GET, например:

```
10 REM --- GET ---
20 HIRES :C=RND(1)*6+1:INK C:PAPER 0
30 FOR A=10 TO 50 STEP 10
40 CURSET 50+(A*2),96,3
50 CIRCLE 10+A,2
60 NEXT
70 PRINT "НАТИСНИ КЛАВИШ"
80 GET A$
90 GOTO 20
100 REM
110 REM ВИЖ INPUT, KEY$
```

Променливата след GET може да бъде числова. Тогава допустимите символи ще бъдат само десетте цифри от клавиатурата.

Подобно на GET функцията KEYS също приема един символ, въведен от клавиатурата, но за разлика от нея KEYS не чака, а предава управ-

лението на следващия ред в програмата, ако не е натиснат някакъв клавиш.

KEY\$

Формат: $v\$ = KEY\$$
Код: 241

Функцията KEY\$ е едно от средствата, чрез които Правец—8Д получава информация отвън. Тази вградена функция „претърсва“ клавиатурата за натиснат клавиш и предава към програмата неговия символ. За разлика от GET компютърът не чака задължителен отговор от клавиатурата. Изпълнението на програмата не се прекъсва. Натискането на предварително дефиниран клавиш всъщност може да въведе промени в хода на програмата. Ако не се натисне клавиш, функцията KEY\$ приема „празен“ низ и програмата продължава със следващия оператор.

Функцията KEY\$ е много полезна за някои игри, тъй като подава към програмата стойността на символа от натиснатия клавиш. Така произволни клавиши могат да се използват за управление на „движението“ на экрана.

```
10 REM --- KEY$ ---
20 CLS:X=2
30 PRINT "'J'-НАЛЯВО, 'K'-НАДЯСНО, 'S'
-СТОП"
40 REPEAT
50 V$=KEY$
60 IF V$="J" THEN X=X-3:PLOTX+3,10, "
70 IF V$="K" THEN X=X+3:PLOTX-3,10, "
80 IF X<2 THEN X=2
90 IF X>35 THEN X=35
100 PLOT X,10,"(*)"
110 UNTIL V$="S"
120 PRINT "КРАИ НА ПРИМЕРА"
130 REM
131 REM ВИЖ GET, INPUT
```

READ

Формат: READ $v1, v2, \dots$
Код: 149

Командата READ чете стойности за променливите $v1, v2, \dots$ от списъка със стойности на команди DATA. Тя чете последователно, като запо-

чва от списъка на първия DATA. Не трябва да се програмира четене на повече данни, отколкото действително се съдържат в командите (това ще даде съобщение за грешка OUT OF DATA). Очевидно мястото на READ е критично, докато DATA може да се намира във всяка точка на програмата.

```
10 REM --- READ ---
20 CLS
30 FOR A=1 TO 5
40 READ V,V$
50 PLOT V,10,V$
60 NEXT
70 DATA 2,ПРАВЕЦ,8,-,9,8Д,12,МИКРОПР
ОЦЕСОР,26,6502
80 REM
90 REM ВИЖ DATA,RESTORE
```

DATA

Формат: DATA n1, n2, ...
Код: 145

Командата DATA определя списък от данни, които могат да се четат и присвояват като стойности на променливи с READ. Елементите на списъка n1, n2 и т. н. могат да бъдат числа, символи и низове. Данните трябва да са заградени в кавички, ако съдържат водещи интервали. Отделните елементи в списъка се разделят със запетая. Ако трябва да се включи запетайка в елемент от списъка, тя се загражда в кавички. Командите DATA могат да се разполагат произволно в програмата независимо от мястото, където списъците им се четат от READ. Елементите на списъка се четат отляво надясно. Указателят за четене се връща в началото на списъка само с командата RESTORE. Това се прави, когато се налага повторно четене на данните. В примерната програма данните от редове 70 и 80 се четат в ред 20 и се извеждат на екрана в ред 30:

```
10 REM --- DATA ---
20 FOR I=1 TO 2: FOR J=1 TO 2: READ A
*A
30 PRINT A$,A
40 NEXT
50 NEXT
60 END
70 DATA "ЗДРАВЕИТЕ",1,"ПРОГРАМИСТИ",2
```

```
80 DATA " НА ", 3, "ПРАВЕЦ-8Д", 4
90 REM
100 REM ВИЖ READ И RESTORE
```

RESTORE

Формат: RESTORE
Код: 154

След като се прочете даден елемент от списък, определен с DATA, специален указател се премества към следващия елемент от списъка. Когато се налага в рамките на една програма данните от списъка да се четат повече от един път, се използва команда RESTORE, която връща указателя в началото на списъка (в противен случай компютърът издава съобщение за грешка OUT OF DATA).

```
10 REM --- RESTORE ---
20 HIRES:PAPER 1:INK 0
30 C=1
40 FOR A=1 TO 5
50 IF C=0 THEN WAIT 20:SHOOT
60 READ V
70 CURSET V,40,3
80 FOR B=1 TO 18 STEP B
90 CIRCLE B,C
100 NEXT B
110 NEXT A
120 IF C=0 THEN END
130 RESTORE
140 C=0
150 GOTO 40
160 DATA 28,73,118,163,208
170 REM
180 REM ВИЖ DATA,READ
```

PRINT

Формат: PRINT списък
PRINT @ x, y; списък
Код: 186

Командата PRINT извежда върху екрана посочените елементи. Списъкът от елементи може да се състои от числа, числови променливи,

низове, изрази, които се отделят един от друг със запетая, точка и запетая или просто с интервал.

Ако PRINT се използва самостоятелно, без елементи, компютърът ще изведе на екрана празен ред.

PRINT може да се съкращава на ?.

Използването на запетая или точка и запетая като разделител между отделните елементи в списъка управлява извеждането им на екрана. Разделител точка и запетая запазва позицията на маркера непосредствено след последния изведен елемент. Така PRINT ще продължи да извежда в рамките на същия ред. По същия начин се извежда и ако не се използува разделител. Разделителят ; просто онагледява команда, а в редица случаи трябва да се използува задължително за разграничаване на елементите в даден списък.

Ако PRINT завършва с точка и запетая, следващата команда PRINT ще продължи да извежда на същия ред.

Поставянето на запетайка като разделител между елементите в списъка премества маркера в началото на следващото поле от екрана. При Правец—8Д екранът е разделен (табулиран) на пет полета, всяко с широчина 8 позиции. Запетайката придвижва маркера с едно поле надясно и след това към началото на следващото поле, като по този начин осигурява оставянето поне на един интервал между така форматиранные елементи. Използването на няколко запетайки една след друга премества маркера с няколко полета вдясно.

С командата PRINT могат да се вмъкнат управляващи символи, като се използува функцията CHR\$(n) (където n е код ASCII на символ). Управляващите символи заемат по една позиция в картата на паметта на екрана. На самия екран това се изобразява с по една празна позиция (интервал).

В следващия пример се демонстрират различни начини на използване на PRINT:

```
10 REM --- PRINT ---
20 B=2
30 A$="ПРАВЕЦ-8Д"
40 PRINT 1,2,3
50 PRINT A$
60 PRINT "TO BE OR NOT TO BE"
70 PRINT 2*B OR NOT 2*B
80 PRINT
90 PRINT 1;2;3
```

Форматът PRINT@ x, y определя координати (x, y) на позиция от екрана в режим TEXT или LORES, от която да започне извеждане на елементите. По този начин може да се управлява текущата позиция на маркера.

Координатата x е номер на колона, а y — номер на ред. Двете стой-

ности се отделят със запетая и трябва да бъдат последвани от точка и запетая преди списъка на елементите, за който важат същите правила както за обикновения формат на PRINT. Координатите x и y могат да бъдат променливи и числови изрази. Те се закръгляват, ако не са цели числа. Ако стойностите са извън допустимите граници, компютърът ще издаде съобщение за грешка ILLEGAL QUANTITY. Следващата програма чертае окръжност, като използва функциите SIN и COS, за да изчисли позициите на X и Y в PRINT@ :

```
10 REM --- PRINT @ ---  
20 R=13:XC=20:YC=13:CLS  
30 FOR A=0 TO 2*PI STEP PI/50  
40 X=XC+COS(A)*R  
50 Y=YC+SIN(A)*R  
60 PRINT @X,Y;"*"  
70 NEXT  
80 REM  
90 REM ВИЖ LPRINT,SPC,TAB
```

LPRINT

Формат: LPRINT
Код: 143

Командата е аналогична на PRINT, но вместо на екран извежда стойностите към печатащо устройство. Не се допуска формат от типа LPRINT@. В комбинация с управляващи кодове LPRINT може да управлява изхода към различни печатащи устройства. Дължината на реда може да се променя с команда POKE #256, n, където n е дължината на реда в символи.

TEXT

Формат: TEXT
Код: 161

Командата TEXT връща компютъра в стандартен текстов режим (както при началното включване) с 27 реда и 40 колони и стандартния набор от символи. Използването на команда LORES или HIRES активизира графичните режими на екрана, които остават, докато не се отменят съответно с CLS и TEXT.

CLS

Формат: CLS
Код: 148

Тази команда изчиства экрана, оцветява го с текущия цвят на фоновата област и позиционира маркера горе вляво. Добре е всички програми с текст да започват с команда CLS, освен ако има основателни причини предишният текст да остане на экрана. В примерната програма экранът се изчиства два пъти -- първия път -- в ред 20, когато се изчиства и подготвя за извеждане, и втория път -- в ред 70, след като текстът е запълнил экрана (това е същото, както ако се използва **CTRL-L**).

```
10 REM --- CLS ---
20 PAPER 0 : INK 5 : CLS
30 FOR A=0 TO 134
40 PRINT "ПРАВЕЦ-ВД",
50 NEXT
60 WAIT 100
70 CLS
80 WAIT 100
90 GOTO 20
100 REM
110 REM ВИЖ HIRES, LORES И TEXT
```

SPC

Формат: SPC (*n*)
Код: 197

Функцията SPC извежда на экрана *n* интервала. Ако стойността на *n* не е цяло число, тя се закръглява надолу. Могат да се използват и изрази, а тъй като е възможно вътре в скобите да се използват и функции, които работят с низове, SPC се оказва много полезна функция при форматирането. Тя се използва в команди PRINT, за да постави интервали преди или между отделни елементи на PRINT, например:

```
10 REM --- SPC ---
20 FOR F=0 TO 20
30 PRINT "***;SPC(F);***;F;"ИНТЕРВАЛА"
40 NEXT
```

```

** О ИНТЕРВАЛА
* * 1 ИНТЕРВАЛА
* * 2 ИНТЕРВАЛА
* * 3 ИНТЕРВАЛА
* * 4 ИНТЕРВАЛА
* * 5 ИНТЕРВАЛА
* * 6 ИНТЕРВАЛА
* * 7 ИНТЕРВАЛА
* * 8 ИНТЕРВАЛА
* * 9 ИНТЕРВАЛА
* * 10 ИНТЕРВАЛА
* * 11 ИНТЕРВАЛА
* * 12 ИНТЕРВАЛА
* * 13 ИНТЕРВАЛА
* * 14 ИНТЕРВАЛА
* * 15 ИНТЕРВАЛА
* * 16 ИНТЕРВАЛА
* * 17 ИНТЕРВАЛА
* * 18 ИНТЕРВАЛА
* * 19 ИНТЕРВАЛА
* * 20 ИНТЕРВАЛА

```

Тази програма използва цикъл, за да определи броя на интервалите между звездичките, като съобщава колко интервала е оставила. Функцията SPC е полезна като алтернатива на функцията TAB и е гъвкава, както показва следващият пример. Стойността, създадена в цикъла FOR...NEXT, се превръща в низ, като се използва STR\$ в ред 40, а после с този низ се използва LEN, за да върне на ред 50 изчисления брой на интервалите. Това създава еcran, при който всички числа са подредени.

```

10 REM --- SPC'2 ---
20 FOR F=1 TO 19
30 N% = F^2*47
40 N$ = STR$(N%)
50 PRINT SPC(20-LEN(N$));N%
60 NEXT
70 REM
80 REM ВИЖ PRINT,LPRINT,TAB

```

TAB

Формат: TAB (n)
Код: 194

Функцията TAB се използва в команда PRINT, за да позиционира елементите ѝ на определена колона на экрана. Стойността на аргумента *n* определя колоната, към която ще се премести позицията за PRINT. Следващият елемент, който трябва да се изведе, ще следва директно, ако след функцията TAB (*n*) няма нишо или има точка и запетая. Ако след TAB има запетая, поредният елемент ще се изобрази на следващото стандартно поле. Ако числата не са отрицателни, те се изобразяват с водещи интервали. Колоните на экрана са номерирани от 0 до 39, а примерната програма тук илюстрира ефекта от защищените колони:

```
10 REM --- TAB ---
20 FOR F=0 TO 15
30 PRINT TAB(F);F
40 NEXT
50 PRINT "ВЪВЕДИ CTRL-Z И ОПТИАЙ ПАК!"
```

В една команда PRINT може да има повече от една функция TAB. Това е показано в следния пример:

```
10 REM --- TAB'2 ---
20 PRINT TAB(10)10;TAB(20);20
30 PRINT TAB(10);10;TAB(20),-20
40 PRINT TAB(10)"X"TAB(20),"Y"
50 REM
60 REM ВИЖ LPRINT,PRINT,POS,SPC
```

POS

Формати: POS (0)
POS (1)
Код: 219

Функцията POS определя текущата хоризонтална позиция на маркера, получена от последната команда PRINT. Тя не може да се ползува за позиции, определени от PLOT или PRINT @. Форматът POS (0) дава позицията на маркера на экрана, а POS (1) определя хоризонталната позиция на печатащата глава при изпълнение на команда LPRINT.

```

10 REM --- POS ---
20 DIM A(4,2)
40 FOR K=1 TO 4
50 FOR J=1 TO 2
60 A(K,J)=K*RND(1)+J
70 NEXT
80 NEXT:CLS
90 FOR K=1 TO 3:PRINT:NEXT
100 PRINT A(1,1);
110 REPEAT:PRINT" ":";UNTIL POS(0)=23
120 PRINT A(1,2)
129 REM
130 REM ВИЖ PRINT

```

ГРАФИКА

LORES

Формати: LORES 0
 LORES 1
Код: 137

Тази команда установява графичен режим с малка разделителна способност и осигурява два от четирите режима на екрана на Правец—8Д (другите два са TEXT и HIRES). Форматът на екрана става 27 реда по 40 символа:

Командата LORES 0 осигурява екран, подобен на този в режим TEXT, но генерира черен фон, на който се изобразяват светли (бели) символи. Самите символи трябва да се позиционират с PRINT @ или PLOT, тъй като простото използване на PRINT прехвърля целия екран в текстов режим.

Командата LORES 1 действува по абсолютно същия начин с тази разлика, че изобразява на екрана алтернативния набор от символи. Примерната програма демонстрира използването на двета режима на екрана:

```

10 REM --- LORES ---
20 LORES 0:C=0
30 PLOT 2,24,"СИМВОЛЕН НАЕОР В LORES 0"
40 FOR A=32 TO 126
50 X=RND(1)*36+1:Y=RND(1)*22+1
60 C$=CHR$(A)

```

```
70 PLOT X,Y,C$  
80 WAIT 50  
85 NEXT  
90 IF C=1 THEN WAIT 500:CLS:END  
100 PRINT:PRINT"А СЕГА В LORES 1 ":"WAIT  
110 GOSUB 130  
120 GOTO 30  
130 CLS:LORES1:C=1  
140 RETURN  
149 REM  
150 REM ВИД TEXT, HIRES
```

PAPER

Формат: PAPER *n*
Код: 177

Тази команда определя цвета на фона на экрана. Тя действува в режими на голяма и малка разделителна способност. В параметъра *n* на командата се посочва код на един от следните цветове:

- 0 — черно;
- 1 — червено;
- 3 — жълто;
- 4 — синьо;
- 5 — лилаво;
- 6 — светлосиньо;
- 7 — бяло.

За да се определи различен фон за отделните части от экрана, използват се CHR\$ и FILL.

```
10 REM --- PAPER ---  
20 HIRES:INK0  
30 A=0:B=0  
40 FOR V=0 TO 25  
50 A=A+1:B=B+3  
60 IF A>7 THEN A=0  
70 PAPER A:WAIT 20  
80 CURSET 110,100,3  
90 CIRCLE B,1  
100 NEXT V  
110 GOTO 20  
120 REM  
130 REM ВИД LORES,HIRES,INK,TEXT,CHR$
```

INK

Формат: INK *и*
Код: 178

Тази команда работи в режими на голяма и малка разделителна способност. Тя оцветява всичко, което се изписва на екрана според стойността на *и*. Компютърът Правец-8Д предлага 8 цвята:

- 0 — черно;
- 1 — червено;
- 2 — зелено;
- 3 — жълто;
- 4 — синьо;
- 5 — лилаво;
- 6 — светлосиньо;
- 7 — бяло.

Командата INK променя цвета на „мастилото“, с което е изписано всичко на екрана и не може да се използува, за да извежда отделните символи в различни цветове.

```
10 REM --- INK ---
20 HIRES
30 A=0:B=0
40 FOR V=0 TO 25
50 B=B+1:A=A+3
60 IF B>7 THEN B=0
70 INK B
80 CURSET 110,100,3
90 CIRCLE A,1
100 NEXT V
110 GOTO 20
120 REM
130 REM ВИИH LORES,HIRES,PAPER,TEXT,CHR$
```

PLOT

Формат: PLOT *x, y, и*
PLOT *x, y, с*
Код: 135

Тази команда се използва за позициониране на символи на екран с малка разделителна способност.
В първия формат на командата *и* трябва да бъде числовой израз. Изразът

дава код ASCII, а на екрана се извежда съответният символ. Могат да се използват стандартен и алтернативен набор от символи, което позволява да се създават интересни изображения на екрана. Например

10 PLOT 11,11,12:PLOT 15,11,"ПРАВЕЦ-8Д"

ще изобрази на екрана мигащ надпис ПРАВЕЦ—8Д. Практиката ще разкрие възможностите на командата, използвана по този начин. Форматът PLOT x, y, s извежда низа s от позиция (x, y) върху екрана. Координатите както за числовите, така и за символните изрази са между 0—39 за x и 0—26 за y . Командата работи в режими TEXT, LORES 0 и LORES 1. Тя не може да се използува за режим HIRES.

```
10 REM --- PLOT ---
20 CLS:LORES 0
30 PLOT 2,2,"PLOT В РЕЖИМ LORES 0"
40 FOR A=1 TO 23
50 X=RND(1)*31+4:Y=RND(1)*22+3
60 PLOTX,Y,A:PLOT X+2,Y,"ПРАВЕЦ-8Д"
70 WAIT 50:NEXT
80 REM
90 REM ВИЖ CHAR,PRINT
```

SCRN

Формат: SCRN (x, y)
Код: 242

Функцията SCRN определя кода ASCII на символа в избраната позиция на екрана (определена от координатите x и y). Тя се използува само в режими LORES и TEXT. За x и y са валидни и числови изрази, но стойностите трябва да бъдат в интервал 0—39 за x и 0—26 за y ; в противен случай ще се издаде съобщение за грешка ILLEGAL QUANTITY.

Простите примери по-долу илюстрират възможностите на SCRN. Първият използува PRINT@, а вторият — PLOT. Знакът диез (#) се изобразява на екрана в позиция (5,5) и по двата начина. Ред 30 използува SCRN първо да изведе кода ASCII за диез, а после същият израз на SCRN се използува с функция CHR\$. Именно той извежда в позицията, определена по x и y , действителния символ, намерен в паметта.

```
10 CLS
20 PRINT @5,5;"#"
30 PRINT SCRN(5,5),CHR$(SCRN(5,5))
```

```

10 CLS
20 A$=" "
30 PLOT 5,5,A$
40 PRINT SCRN(5,5),CHR$(SCRN(5,5))

```

Функцията SCRN работи също и с повторно дефинирани символи. Програмата от следващия пример ползва звездичката като ракета, изстреляна нагоре по екрана и управлявана наляво и надясно със съответните клавиши за движение на маркера. Повторно дефинираният символ ! оформя цели, които се показват на екрана. Целта се поразява, когато SCRN (X, Y) получи стойност 33 — кола за „!“.

```

10 REM --- SCRN ---
20 CLS:FOR I=1 TO 8
30 READ A:POKE 46343+I,A
40 NEXT:POKE #24E,1:POKE #24F,1
50 PRINT @2,0;"СТРЕЛКА -ИНТЕРВАЛ: <-
И -> ДВИЖЕНИЕ"
60 FOR I=1 TO 10:PRINT PINT(RND(1)*37)
) +2,1;"*"
70 NEXT:PRINT CHR$(17);CHR$(6)
80 REPEAT
90 REPEAT:K$=KEY$:UNTIL K$=" " OR K$=
CHR$(13):IF K$=CHR$(13) THEN 220
100 X=20:Y=26
110 PRINT @X,Y;"*";:SHOOT
120 REPEAT:OX=X
130 IF KEY$=CHR$(8) AND X>2 THEN X=X-
1
140 WAIT 1
150 IF KEY$=CHR$(9) AND X<39 THEN X=X+
1
160 Y=Y-1
170 PRINT @OX,Y+1;" ";
180 IF SCRN(X,Y)=33 THEN PRINT @X,Y;"*";
:EXPLODE:Y=1:GOTO 200
190 PRINT @X,Y;"*"
200 UNTIL Y=1:PRINT @X,Y;" "
210 UNTIL KEY$=CHR$(13)
220 POKE #24E,32:POKE #24F,4:PRINT CH
R$(17);CHR$(6)
230 END
300 DATA 30,33,45,63,45,33,30,0
399 REM
400 REM BMM ASC,PLOT,PRINT @

```

HIRES

Формат: HIRES
Код: 162

Командата избира режим на работа с голяма разделителна способност (хоризонтално 240 точки и вертикално 200 точки). При този режим на екрана се разкриват максимално графичните възможности на компютъра. Правец—8Д предлага серия от специализирани команди, които работят в такъв режим: CURSET, CURMOV, DRAW, FILL, CIRCLE, PATTERN, CHAR и др.

Командата се подава от един от другите два основни режима — текстов (TEXT) и графичен с малка разделителна способност (LORES). Първите 24 реда на екрана се променят във фонова област с черен цвят. Долните три реда остават в нормалния режим TEXT и се използват за изписване на команди и съобщения.

Естествено HIRES заема значителна част от паметта на компютъра. Тази запазена памет може да се ползува за програми на БЕЙСИК с помощта на командата GRAB. Командата RELEASE я освобождава отново за работа в режим HIRES.

За да се разгледат операторите на програма в режим HIRES, първо трябва да се премине в режим TEXT. След това може да се използува LIST и евентуално да се редактира програмата.

В примерната програма командите до FILL включително се изпълняват само в режим HIRES.

```
5 REM --- HIRES ---
10 HIRES
20 CURSET 14,10,3
30 DRAW 0,180,1:DRAW 220,0,1
40 CURMOV -210,-10,3:DRAW -19,19,1
50 FOR B=1 TO 6
60 READ H:GOSUB 100
70 NEXT B
80 GOSUB 400:END
90 REM РИСУВАНЕ НА СТЪЛБОВЕТЕ
100 X=(B*5+2)*6-5:CURSET X,190,3
110 DRAW 0,-H,1:DRAW 10,-10,1:DRAW 12
,0,1:DRAW -10,10,1
120 CURSET X+23,180-H,3
130 FILL H,1,16
140 CURSET X,190-H,3
150 FILL H,2,B+16
160 CURSET X+13,190-H,3
170 FILL H,1,16
```

```

200 RETURN
300 DATA 100,75,90,124,150,170
400 REM ИЗВЕЖДАНЕ НА ТЕКСТ С CHAR
410 A$="ПРОДАЖБИ НА ПРАВЕЦ-ЗД"
420 FOR I=1 TO LEN(A$)
430 CURSET I*6+40,10,3
440 CHAR ASC(MID$(A$,I)),0,1
450 NEXT
460 PATTERN S1:CURSET 40,20,3
470 DRAW 135,0,1
480 RETURN
500 REM
501 'ВИНИ CHAR,CIRCLE,CURMOV,CURSET,
502 ' DRAW,FILL,LORES,TEXT,PAPER,INK

```

Следват групата специализирани графични команди за режим HIRES. Повечето от тях имат последен операнд f , който в зависимост от стойността си има следните функции:

- $f=0$ — установява фонов цвят (този, избран с PAPER);
- $f=1$ — установява основен цвят (избран с INK);
- $f=2$ — обръща цветовете;
- $f=3$ — не извежда върху скрина.

CURSET

Формат: CURSET x, y, f
Код: 170

Тази команда поставя маркера в дадена позиция на екрана в режим HIRES, като x и y са абсолютни координати в рамките на екрана (за x от 0 до 239, а за y от 0 до 199). Както винаги, f е цяло число от 0 до 3 (вж. HIRES). Тъй като командата CIRCLE позиционира центъра на окръжността в текущата позиция на маркера, ето една добра демонстрация за действието на CURSET:

```

10 REM --- CURSET ---
20 HIRES : INK 5 : PAPER 4
30 FOR C=0 TO 100 STEP 20
40 FOR B=1 TO 0 STEP -1
50 CURSET 20,50+C,0
60 FOR A=0 TO 30
70 CIRCLE 10+A,B
80 CURMOV 5,0,0

```

```
90 NEXT A
100 NEXT B
110 NEXT C
120 REM
130 REM ВИЖ CIRCLE,CURMOV,DRAW,HIRES
```

CURMOV

Формат: CURMOV x, y, f
Код: 171

Тази команда се използва само в режим HIRES и премества маркера на нова позиция. Стойностите на x и y са относителни спрямо текущата позиция на маркера. Параметърът f е 0, 1, 2 или 3 (вж. HIRES). Командата е полезна при изобразяване на движение и при повечето от графичните процедури. В примера, даден за CURSET, командата CURMOV премества една окръжност по экрана и след това използва стойност на f , за да я изтрие.

PATTERN

Формат: PATTERN n
Код: 174

PATTERN (шаблон) е команда, която работи само в режим на голяма разделителна способност. Използува се в комбинация с DRAW или CIRCLE.

Обикновено DRAW и CIRCLE изобразяват плътна линия. Въвеждането на PATTERN може да „накъса“ линията на точки или тирета, като шаблонът се определя от цялото число n , което следва PATTERN и е в интервала 0—255. Примерната програма демонстрира пълните възможности на PATTERN.

```
10 REM --- PATTERN ---
20 HIRES:INK 0:PAPER 1
30 FOR A=1 TO 45
40 PATTERN 170-A
50 DRAW 230,0,1
60 CURMOV -230,3,0
70 NEXT
80 REM
90 REM ВИЖ HIRES,DRAW,CIRCLE
```

DRAW

Формат: DRAW x, y, f
Код: 172

Командата DRAW може да се използува само в режим HIRES. Тя работи в комбинация с CURSET и CURMOV и чертае непрекъсната линия от текущата позиция на маркера до точката с координати x и y . Посочените в командата стойности x и y не са абсолютни координати, а дават относителното изместване спрямо текущата позиция на маркера. Стойността на x трябва да бъде в интервала 0—199, а на y — в интервала 0—239. Излизането извън рамките на екрана дава съобщение за грешка. Както обикновено, параметърът f е от 0 до 3 (вж. HIRES).

Комбинирането на DRAW с PATTERN може да превърне непрекъсната линия в пунктирена или щрихова с определена гъстота (вж. PATTERN).

Ето една интересна демонстрация на DRAW:

```
10 REM --- DRAW ---
20 HIRES:PAPER 6:INK 1
30 FOR A=10 TO 230 STEP 10
40 CURSET A,0,0
50 DRAW 0,199,1
60 WAIT 20:PING
70 NEXT
80 FOR Y=10 TO 190 STEP 10
90 CURSET 0,Y,0
100 DRAW 239,0,1
110 WAIT 20:PING
120 NEXT
130 REM
140 REM ВИЖ CIRCLE, CURMOV, CURSET,
150 REM HIRES И PATTERN.
```

CIRCLE

Формат: CIRCLE r, f
Код: 173

В режим на голяма разделителна способност тази команда чертае окръжност с център в текущата позиция на маркера. Обикновено CIRCLE се използува в комбинация с команда CURSET. Ако част от окръжността се окаже извън екрана, компютърът ще издаде съоб-

щение за грешка. Дължината на радиуса r се задава в брой точки (1—99). В примерната програма се чертаят различни окръжности в различни позиции на маркера. След това начертаното се заличава с $f=0$ (фонов цвят) (вж. HIRES).

```
10 REM --- CIRCLE ---
20 HIRES
30 FOR A=0 TO 60 STEP 5
40 FOR B=1 TO 0 STEP -1
50 CURSET 80+A,100,0
60 CIRCLE A+9,B
70 NEXT B
80 NEXT A
90 END
100 REM
110 REM ВИЖ CURMOV,CURSET,DRAW,HIRES,
      И PATTERN
```

CHAR

Формат: CHAR n, m, f
Код: 176

Командата CHAR изписва един символ с код ASCII, равен на n , от набора символи, определен от m (0 за стандартен или 1 за алтернативен набор), като горният ляв край на символа се разполага в текущата точка на екрана в режим HIRES.

(За параметъра f вж. HIRES.) Аналогични команди за режими LORES са PRINT, PLOT и INPUT.

Командата е особено полезна при графики или игри. Тя не премества маркера, но в комбинация с CURMOV и CURSET дава възможност да се позиционира текст или потребителски символи с точност една точка.

```
10 REM --- CHAR ---
20 HIRES : C=0
30 FOR A=0 TO 150 STEP 50
40 CURSET 40+A,90,3
50 DRAW 20,0,2 : DRAW 0,20,2
60 DRAW-20,0,2 : DRAW 0,-20,2
70 CURMOV 7,6,3
80 CHAR 49+C,0,2
90 C=C+1
```

```
100 NEXT
110 REM
120 REM ВИЖ CURMOV,CURSET,HIRES,PLOT
```

FILL

Формат: FILL x, y, n
Код: 175

Командата FILL се използва само в режим HIRES. Тя дава възможност да се запълват блокове от по 6 точки от екрана. Екранът в режим HIRES има 200 реда и 240 точки на ред. Така един ред за FILL съдържа 40 блока.

Командата запълва у реда по x байта със стойността n — двоичен шаблон за всеки блок. Поради това у е в интервала от 1 до 199, x — от 1 до 40, а и трябва да бъде между 0 и 255.

Командата FILL може да се използува, за да оцвети отделни части от екрана на компютъра с различни цветове. Тя е удобна за запълване на малки фигури с неправилна форма; много бързо запълва големи правоъгълни фигури. За начало на описания блок (горен ляв ъгъл) се приема позицията на маркера. След запълване на блока маркерът остава в долния му ляв ъгъл. Примерната програма показва FILL в действие.

```
10 REM --- FILL ---
20 PAPER 0
30 HIRES:PRINT CHR$(17)
40 REPEAT
50 X=INT(RND(1)*231)
60 Y=INT(RND(1)*175+7)
70 A=INT(RND(1)*(230-X)/6+1)
80 D=INT(RND(1)*(182-Y)+8)
90 CURSET X+6,Y-7,0:FILL D+7,A,16
100 CURSET X,Y,0:FILL D,A,(17+RND(1)*7)
110 UNTIL FALSE
120 REM
130 REM ВИЖ CURSET, CURMOV, HIRES
```

POINT

Формат: POINT (*x, y*)
Код: 243

Функцията проверява цвета на екрана в режим HIRES в точка с координати (*x, y*) и има стойност 0, ако цветът е избран с команда PAPER, и —1, ако е избран от INK. Допустимите стойности за *x* са от 0 до 239, а за *y* — от 0 до 199.

Функцията POINT се използва често в програми за игри, за да открива сблъскване между два обекта. Следващата програма по-долу предлага една такава проста демонстрация:

```
10 REM --- POINT ---
20 HIRES:PAPER4:INK0
30 FOR B=5 TO 175
40 CURSET 200,B,2:CHAR 124,0,2
50 NEXT B
60 A=5
70 REPEAT
80 A=A+8
90 CURSET A,90,0
100 CHAR 127,0,1:WAIT 5
110 CHAR 127,0,0
120 C=POINT(A+11,90)
130 UNTIL C=-1
140 EXPLODE
149 REM
150 REM ВИЖ HIRES
```

МУЗИКАЛНИ ВЪЗМОЖНОСТИ

За разлика от Правец—82, домашният компютър Правец—8Д притежава специализиран триканален музикален процесор. Той предлага 3 специализирани музикални команди (SOUND, MUSIC и PLAY), които позволяват да се генерира удивително широка гама от музикални звуци. Други четири команди възпроизвеждат вграден звук (ZAP, SHOOT, PING, EXPLODE).

ZAP

Формат: ZAP
Код: 165

Това е команда за издаване на вграден звук. Той имитира изстрел с някое от оръжията на бъдещето и се използва при игри. За разлика от другите команди за вградени звуци (PING, SHOOT и EXPLODE) ZAP не изисква да се използува WAIT при управление на звука и може да се подава многократно.

```
10 REM --- ZAP ---
20 FOR F=1 TO 7
30 PAPER F
40 ZAP
50 NEXT
98 REM
99 REM ВИЖ EXPLODE,SHOOT,PING
```

SHOOT

Формат: SHOOT
Код: 163

Това е команда за вграден звук, който имитира изстрел на пушка. Ако се използват няколко последователни команди SHOOT, те трябва да са разделени с WAIT, за да се чуе звукът и да не се застъпват отделните изстрели. Ако се изпълни програмата

```
10 REM --- SHOOT ---
20 FOR C=1 TO 6
30 SHOOT
50 NEXT
```

ще се чуе същият звук, както ако се изпълни само командалата SHOOT, тъй като цикълът завършва преди края на първия звук. За да се чутят 6 отделни изстрела, трябва да се прибави WAIT:

```
10 REM --- SHOOT'2 ---
20 FOR C=1 TO 6
30 SHOOT
40 WAIT 30
50 NEXT
60 REM
70 REM ВИЖ EXPLODE,PING,ZAP
```

PING

Формат: PING
Код: 166

Това е команда за вграден звук, имитиращ звука на камбанка. Той може да се използва като сигнал или в програми за игри. В комбинация с WAIT тази команда генерира ефектна поредица от звуци. Действието на командата може да се имитира, като се използва CTRL—G (или при опит за въвеждане на повече от 80 символа в един програмен ред).

```
10 REM --- PING ---
20 PAPER 1:INK0:CLS:PRINT
30 PRINT CHR$(140) "ИЗБЕРИ: "
40 PRINT:PRINT:PRINT
50 PRINT CHR$(147) "1 КОТЕ":PING
60 WAIT 30
70 PRINT CHR$(148) "2 ХАМСТЕР":PING
80 WAIT 30
90 PRINT CHR$(146) "3 ПИЛЕ":PING
100 REM
110 REM ВИИ EXPLODE, SHOOT, ZAP
```

EXPLODE

Формат: EXPLODE
Код: 164

Това е команда за вграден звук, който имитира експлозия. Използува се най-често при игри, но може да бъде полезна и при редица сериозни програми. За многократно повторение на експлозии трябва творчески да се използват и закъснения (с команда WAIT).

```
10 REM --- EXPLODE ---
20 HIRES:PAPER 3:INK 4
30 FOR B=0 TO 95 STEP 4
40 CURSET 120,2+B,0
50 GOSUB 200
60 CURSET 120,190-B,0
70 GOSUB 200
80 NEXT
90 EXPLODE
```

```

100 FOR K=1 TO 10
110 PAPER 4:INK 3
120 WAIT K
130 PAPER 3:INK 4
140 WAIT K
150 NEXT
160 WAIT 200
170 GOTO 20
180 END
200 FOR I=1 TO 3
210 CHAR 100,1,1,CURMOV 6,0,0
220 NEXT
230 RETURN
240 REM
250 REM BOSH PING, SHOOT, ZAP

```

MUSIC

Формат: MUSIC *c,o,n,v*
Код: 168

Командата е типична за съвременните домашни компютри и се използва за генериране на музикален изход. Тя управлява четири параметъра:

c — канал (от 1 до 3);
o — октава (от 0 до 7);
n — нота (от 1 до 12);
v — сила на звука (от 0 до 15).

Това означава, че се генерира нота *n* от октава *o* по канал *c* със сила *v*. Четирите стойности трябва да бъдат разделени със запетайки. Възможностите на командата очевидно са големи, но използването ѝ не е толкова просто. Обикновено MUSIC се комбинира с PLAY (команда, която оформя самия звук и управлява броя на действуващи звукови канали).

Тук даваме малък пример. В края на програмата е използвана команда PLAY 0,0,0,0, за да изключи звука.

```

10 REM --- MUSIC ---
20 FOR O=0 TO 6
30 FOR N=1 TO 12
40 MUSIC 1,O,N,10
50 PLAY 3,0,7,0
60 WAIT 30

```

```
70 NEXT N
80 NEXT O
90 PLAY 0,0,0,0
100 EXPLODE
109 REM
110 REM WITH PLAY, SOUND
```

PLAY

Формат: PLAY *t,s,e,d*
Код: 169

Това е една от сложните звукови команди, които предлагат възможности за звук и музика, далеч надхвърлящи стандартните при персоналните компютри. Естествено команди като PLAY и SOUND изискват чисто музикални познания и практика в допълнение към желанието за програмиране.

Компютърът Правец-8Д е снабден с три звукови канала, като PLAY е командата, с която се определя комбинацията на тези канали. Трите канала могат да издават едновременно три различни звука.

Форматът на командата позволява избор на:

- t* — изходни канали за тон (от 0 до 7);
- s* — шумови канали за звук (от 0 до 7);
- e* — тип моделиране на звука (от 0 до 7);
- d* — продължителност (от 0 до 32 767).

Ефектът от различните комбинации на канали се разбира след малко упражнения. Следващата таблица улеснява използването на командата. Колонката вляво представлява стойността на *t* или *s*, а колонката вдясно показва кои канали се активизират от тази стойност:

<i>t</i> или <i>s</i>	комбинация от канали
0	няма включени канали
1	включен е канал 1
2	включен е канал 2
3	включени са канали 1 и 2
4	включен е канал 3
5	включени са канали 1 и 3
6	включени са канали 2 и 3
7	включени са всички канали — 1, 2 и 3

Третият параметър на командата — *e*, може би е най-труден за разбиране. Той представлява цяло число, моделиращо „формата“ на звука, създаван от компютъра. С него се определя ритъмът за промяна на силата на звука. Има например специфичен формат за рязко повишенна сила (бърза атака) и бавно затихване на звука. Друг формат започва направо със силен звук, който постепенно затихва. Общо може да избираме 7 формата (от 1 до 7), които моделират различно звука. Ако

променливата *e* има стойност 1 или 2, ще се моделира звук с фиксирана дължина, докато стойностите от 3 до 7 генерират продължителен звук от различен тип. Веднъж зададен с PLAY, форматът *e* определя всички следващи звуци. Последният параметър, *d* (0—32767), определя продължителността на звука в хиляди от секундата за разлика от командата WAIT, където се задават стотни от секундата.

За да се изключат звуковите канали след ползване на MUSIC или SOUND, трябва да се подаде команда PLAY с нулеви параметри (PLAY 0,0,0).

В показаната програма се изprobват различни комбинации от параметри на PLAY.

```
10 REM --- PLAY ---
20 CLS:PLAY 0,0,0,0
30 CLS:PRINT:PRINT
40 INPUT"ВЪВЕДИ ЦИФРА (0-7) ЗА МОДЕЛИ
РАНЕ";E
50 IF E<0 OR E>7 THEN 40
60 INPUT"ИЗБЕРИ ПРОДЪЛЖИТЕЛНОСТ (1-65
535)";D
70 PRINT"ТОВА Е P L A Y 1, 0,"E","D
80 SOUND 1,500,0
90 PLAY 1,0,E,D
100 PRINT"НАТИСНИ КЛАВИШ ЗА СПИРАНЕ Н
А ЗВУКА"
110 GET A$:GOTO 20
120 REM
130 REM ВИЖ MUSIC,SOUND
```

SOUND

Формат: SOUND *c,p,v*
Код: 167

Компютърът Правец—8Д има специализиран звуков чип. Командата SOUND е третата истински музикална команда заедно с MUSIC и PLAY. Тя активизира първи, втори или трети канал и генерира по него чист звук (съответно при *c*=1, 2 или 3) или шум (при *c*=4, 5 или 6) с период, определен от *p*, и сила, определена от *v*.

Посоченият период *p* управлява тона на създадения звук. Допустими са стойности от 0 до 65 535, но полезната интервал за *p* е от 0 до 4000. Това се демонстрира със следната програма:

```
10 PLAY 1,0,0,0
20 FOR P=0 TO 32767
```

30 SOUND 1,P,8

40 NEXT

Силата на звука e започва от 0 (което активизира параметъра e (моделиране на звука) на командата PLAY), минава през 1 (тихо) и достига 15 (силно). Нормално се работи със стойности от 3 до 5. Този параметър не действува без наличието на съответна команда PLAY. Командата SOUND може да се използува без PLAY за активизиране на канали — в този случай се работи с канал за тон 1.

Следващите няколко примера дават известна представа за гъвкавостта на командата SOUND. Независимо че основната структура на програмите е една и съща, те създават много различни звуци.

Първата програма има команда PLAY, която активизира първи канал за шум, следвана от команди SOUND, разделени с WAIT. Каналът се избира с 4 (което е равно на лърви канал за шум), периодът се променя според стойността на променливата на цикъла p , а за сила на звука се използват два различни параметъра:

```
10 REM --- SOUND'2 ---
20 FOR P=1 TO 3
30 PLAY 0,1,1,250
40 SOUND 4,8+P,10
50 WAIT 5
60 SOUND 4,3+P,6
70 WAIT 20
80 PLAY 0,0,0,0
90 NEXT P
100 GOTO 20
110 REM
120 REM ВИД MUSIC,PLAY,WAIT
```

Параметърът за моделиране на звука от командата PLAY действува, когато силата на звука в SOUND е нула. Ако го се промени в редове 40 и 60 на нашия пример (съответно на SOUND 4,8+P,0 и SOUND 4,5+P,0) ще чуете разликата при активизирана стойност на e . (Същото важи и за следващите два примера.)

Във втората програма е променен само каналът на звука в PLAY и SOUND — избран е канал за звук 1.

```
10 REM --- SOUND'3 ---
20 FOR P=1 TO 3
30 PLAY 1,0,1,250
40 SOUND 1,8+P,10
50 WAIT 5
60 SOUND 1,3+P,6
```

```
70 WAIT 20
80 PLAY 0,0,0,0
90 NEXT P
100 GOTO 20
110 REM
120 REM ВИМ MUSIC,PLAY,WAIT
```

В последния пример е премахната команда PLAY от ред 30. Така SOUND се използва без команда PLAY, въпреки че накрая даваме PLAY 0,0,0,0, за да се изключи звукът.

```
10 REM --- SOUND'4 ---
20 FOR P=1 TO 3
40 SOUND 1,8+P,10
50 WAIT 5
60 SOUND 1,3+P,6
70 WAIT 20
80 PLAY 0,0,0,0
90 NEXT P
100 GOTO 20
110 REM
120 REM ВИМ MUSIC,PLAY,WAIT
```

ПРИЛОЖЕНИЯ

СЪОБЩЕНИЯ ЗА ГРЕШКИ

Съвременният компютър има развита система за диагностика. По време на изпълнение на програмата интерпретаторът издава различни съобщения. Част от тях поясняват работата на програмата, други показват грешки — например в синтаксиса на отделни команди. Неспазването на техническите и програмните ограничения на компютъра също се „забелязва“ от интерпретатора и той издава съответни съобщения. Най-трудно се откриват логически грешки — при тях програмата се изпълнява, но просто не дава очакваните резултати.

Съобщението за грешка съдържа информация за нейния характер и номера на реда, където тя е открита.

Следват съобщенията за грешки при Правец—8Д, подредени по азбучен ред. Анализирани са накратко причините за възникването им.

?BAD SUBSCRIPT ERROR

Съобщението се издава при опит да се използува несъществуващ елемент от масив. За дефинираните по премълчаване масиви това означава, че индексите на элемента са по-големи от 10. Когато масивът е описан с DIM, съобщението показва, че индексите са извън обявените граници.

?BAD UNTIL ERROR

Грешката се проявява при опит да се изпълни команда UNTIL, преди която не е изпълнена съответната команда REPEAT.

?CAN'T CONTINUE ERROR

Програмата е спряла и не може да продължи с команда CONT. Командата може да се използува само след спиране с CTRL—С или STOP при положение, че не са внесени промени в програмата.

?DISPTYPE MISMATCH ERROR

Опит да се изпълни команда за извеждане на екрана в неправилен режим. Например DRAW или CHAR са използвани в режим TEXT или LORES, или пък PLOT и PRINT са използвани в режим HIRES.

?DIVISION BY ZERO ERROR

Опит да се дели на нула. Проверете дали няма променливи със стойност 0 или недефинирани променливи.

?FORMULA TOO COMPLEX ERROR

Това съобщение се появява при много сложни формули. Разделете израза на по-малки части.

?ILLEGAL DIRECT ERROR

В директен режим е използвана команда, която може да се задава само в програма (например INPUT или GET).

?ILLEGAL QUANTITY ERROR

Използван е параметър, чиято стойност е извън допустимия интервал. Проверете точно:

- какви са валидните интервали в дефинициите на параметрите;
- стойностите, които са резултат от изчисление на изрази и участват като параметри в посочения ред;
- всяко използване на INT, както и автоматичното закръгляване (при превръщането на реални в цели числа).

Това съобщение се получава и при присвояване на стойност, по-голяма от 32 767 или по-малка от -32 768.

?NEXT WITHOUT FOR ERROR

Програмата е срешила NEXT, за което няма съответстващо FOR...TO. Причината е просто пропуснато FOR...TO, некоректно посочена променлива на цикъла или неправилно предаване на управлението в цикъла.

?OUT OF DATA ERROR

Програмата е инструктирана да чете с READ несъществуваща поредица от елементи, т. е. има прекалено много команди READ или прекалено малко елементи, посочени в DATA.

?OUT OF MEMORY ERROR

Съобщението означава, че е надхвърлена оперативната памет, предназначена за съхраняване на данни, програми на ГЕЙСИК, променливи и др. В някои случаи решение може да бъде освобождаването на неизползваниите области с команда FREE.

Същото съобщение се получава и ако в програмата се използват по-вече от 16 вложени един в друг цикли или подпрограми.

?OVERFLOW ERROR

Това е съобщение за препълване. То се получава, когато резултатът от изчисленията е прекалено голямо число.

Най-голямото число, което Правец-8Д може да обработи, е приблизително 1.7E38, а най-малкото е 2.93E-39.

?REDIM*D ARRAY ERROR

Съобщението се предизвиква от опит за повторно дефиниране на да-

ден масив в рамките на една програма независимо от това, дали маси-
вът е бил дефиниран с DIM или по премълчаване.

?REDO FROM START

Използвана е команда INPUT, която очаква от клавиатурата само
числови данни. Съобщението се издава, ако въведете текст. Управле-
нието се връща към INPUT, за да опитате повторно въвеждане.

?RETURN WITHOUT GOSUB ERROR

Програмата е срещнала команда RETURN, без да е обработила пред-
варително съответна команда GOSUB.

?STRING TOO LONG ERROR

Допустимата максимална дължина на един низ е 255 символа. Низът,
който е въведен или е получен при обединяване на низове, е по-дълъг.

?SYNTAX ERROR

Неправилен формат на команда. Интерпретаторът на БЕЙСИК не я
разпознава, като в общия случай има синтактична грешка, например
неправилно написани запазени думи, липсваща или неправилна пунк-
туация и т. н.

?TYPE MISMATCH ERROR

Тази грешка се появява, когато на чисрова променлива или функция
е присвоен низ или обратното. Съобщението показва опит за пряк
обмен на различни типове данни.

?UNDEF'D STATEMENT ERROR

Опит за предаване на управлението на несъществуващ номер на ред.

?UNDEF'D FUNCTION ERROR

Съобщение за недефинирана функция.

Програмата е срещнала израз, включващ изчисляване на потребител-
ска функция, която не е определена предварително с команда DEF
FN.

ТАБЛИЦА НА КОДОВЕТЕ ASCII (КОИ—7)

Кодове 0—31

Код	Действие	Въвеждане от клавиатурата
0	NULL	CTRL-@
1	копиране на символ	CTRL-A
2		CTRL-B
3	прекъсване	CTRL-C
4	двойно високи символи	CTRL-D
5		CTRL-E
6	озвучена клавиатура	CTRL-F
7	звуков сигнал (BEL)	CTRL-G
8	маркер наляво (BS)	CTRL-H
9	маркер надясно (TAB)	CTRL-I
10	маркер надолу (LF)	CTRL-J
11	маркер нагоре (VT)	CTRL-K
12	изчистване на екран (FF)	CTRL-L
13	RETURN	CTRL-M
14	изчистване на ред	CTRL-N
15	блокиране на екран	CTRL-O
16	управлява печата (SYN)	CTRL-P
17	маркер включен	CTRL-Q
18		CTRL-R
19	екран	CTRL-S
20	горен регистър (CAPS LOCK)	CTRL-T
21		CTRL-U
22		CTRL-V
23		CTRL-W
24	цифориране на ред (CAN)	CTRL-X
25		CTRL-Y
26		CTRL-Z
27	ESC (ESCAPE)	CTRL-[
28		CTRL-\
29		CTRL-]
30		CTRL-^
31		

Кодове 32—127

При режим LORES 0 се изобразяват стандартните символи, при LORES 1 — алтернативните. Следва таблица на стандартното изобразяване.

Код	Символ	Код	Символ	Код	Символ
32	интервал	64	¤	96	ю
33	!	65	А	97	а
34	"	66	В	98	б
35	#	67	С	99	ц
36	\$	68	Д	100	д
37	%	69	Е	101	е
38	&	70	Ф	102	ф
39	.	71	Г	103	г
40	(72	И	104	и
41)	73	Ј	105	ј
42	*	74	Ќ	106	ќ
43	+	75	Љ	107	љ
44	,	76	М	108	м
45	-	77	Н	109	н
46	.	78	О	110	о
47	/	79	П	111	п
48	0	80	Р	112	р
49	1	81	Q	113	q
50	2	82	С	114	с
51	3	83	Т	115	т
52	4	84	У	116	у
53	5	85	Џ	117	џ
54	6	86	Ѡ	118	ѡ
55	7	87	Ѡ	119	ѿ
56	8	88	Х	120	х
57	9	89	Ѡ	121	ѿ
58	:	90	Ѡ	122	ѿ
59	;	91	Ѡ	123	ѿ
60	<	92	Ѡ	124	ѿ
61	=	93	Ѡ	125	ѿ
62	>	94	Ѡ	126	ѿ
63	?	95	Ѡ	127	DEL

УПРАВЛЯВАЩИ КОДОВЕ ESCAPE

Натискането на клавиша ESC подготвя компютъра за въвеждане на управляващ символ. При Правец-8Д така се определя режимът за извеждане на символите в рамките на един ред. За всеки ред от екрана може да се използува различен режим за изобразяване на символите — с различен цвет, на различен фон, мигащи, с двойна височина и т. н. Вторият символ, набиран след ESC, определя атрибута за следващите го символи. Комбинацията от ESC и управляващ символ ще наричаме код ESCAPE.

По-долу са дадени управляващите кодове ESCAPE. Те се въвеждат направо от клавиатурата или от програма (като се използува PRINT CHR\$(27); CHR\$(*n*), където *n* е кодът на втория символ). В средата на таблицата са дадени стойности на атрибутите, които могат да се ползват с командите PLOT (за режим LORES) и FILL (за режим HIRES). Ако в тях се задава число вместо символ, интерпретаторът на БЕЙСИК го разглежда като код ASCII. Така с кодове от 0 до 23 могат пряко да се задават различни атрибути като последен параметър. *Избегайте да използвате кодове от 24 до 31.* Те са свързани със синхронизацията на кадровата честота на монитора.

Атрибутите се задават в началото или в средата на всеки ред. С тях се определя цветът на „мастилото“ (самите символи) и „хартията“ (фона, на който те се извеждат).

Код	Атрибут	Режим
ESCAPE @	0	черно мастило
ESCAPE A	1	червено мастило
ESCAPE B	2	зелено мастило
ESCAPE C	3	жълто мастило
ESCAPE D	4	синьо мастило
ESCAPE E	5	лилаво мастило
ESCAPE F	6	светлосиньо мастило
ESCAPE G	7	бяло мастило
ESCAPE H	8	стандартна азбука
ESCAPE I	9	алтернативна азбука
ESCAPE J	10	стандартна азбука с двойна височина
ESCAPE K	11	алтернативна азбука с двойна височина
ESCAPE L	12	стандартна азбука — мигаш режим
ESCAPE M	13	алтернативна азбука — мигаш режим
ESCAPE N	14	стандартна азбука с двойна височина — мигаш режим
ESCAPE O	15	алтернативна азбука с двойна височина мигаш режим
ESCAPE P	16	черна хартия
ESCAPE Q	17	червена хартия
ESCAPE R	18	зелена хартия

ESCAPE S	19	жълта хартия
ESCAPE T	20	синя хартия
ESCAPE U	21	лилава хартия
ESCAPE V	22	светлосиня хартия
ESCAPE W	23	бяла хартия

ОПЕРАТИВНА ПАМЕТ

Разпределение на паметта за Правец-8Д



Забележка: Всички числа в таблицата са шестнадесетични.

Оперативната памет е *постоянна* (ROM) и *динамична* (RAM).

Постоянната памет съдържа интерпретатор на БЕЙСИК и системни подпрограми. Тя заема старшите адреси на оперативната памет от #C000 до #FFFF. За Правец—8Д тази памет включва 16384 байта, от които се чете практически непрекъснато. Програмистът не може да променя, но може да чете нейното съдържание.

Адресите от #0000 до #BFFF е разположена памет RAM, организирана, както следва:

- 5 страници по 256 К за системни нужди в младшите адреси;
- потребителска памет (работна област за програми);
- памет за стандартен и алтернативен набор от символи;
- памет за графика с голяма разделителна способност.

Ще разгледаме малко по-подробно организацията на паметта RAM, с която практически работи потребителят.

Страница 0 съдържа текуща информация за работата на микропроцесора (например адреси за начало и край на изпълняваната програма на БЕЙСИК, указатели към паметта с променливи и т. н.).

Страница 1 съдържа адреси за управление на подпрограми. Прието е тя да се нарича *стек-памет*. В нея последователно се съхраняват и извличат адресите за връщане от цикли и подпрограми. Последният постъпил адрес първи напуска стека.

В страница 2 се пазят някои системни променливи, необходими за проследяване на операциите в БЕЙСИК (например позиция на маркера, текущ регистър за кирилица или латиница, включен или изключен звуков сигнал и др.).

Страница 3 съдържа адреси за вход—изход между компютъра и външни устройства, както и адреси за обмен на данни между отделните системни елементи в компютъра.

Страница 4 запазва адреси от #0400 до #04FF за системни нужди.

След петте системни страници следват адреси за разполагане на потребителска програма — от #0500 до #97FF. Паметта се запълва от ниските адреси нагоре, като непосредствено след края на програмата започва областта за стойностите на всички дефинирани променливи. Изображението на екрана се описва също в част от паметта RAM. За разлика от стандартния текстов режим (TEXT) екранът за режим с голяма разделителна способност (HIRES) изисква допълнителен обем памет. Тази област (от #9800 до #B3FF) може да се придава към потребителска програма с команда GRAB. Така обаче не може да се ползва режим HIRES.

Обратната команда — RELEASE, възстановява областта като памет за экрана в режим HIRES.

В адреси от #B400 до #BB7F следват описанията на двата набора от символи за Правец—8Д.

Стандартният набор съхранява 128 символа в област от 1024 байта.

Алтернативният набор дава 112 символа, за които с запазена област от 896 байта. Всеки символ заема 8 последователни байта, в които се определя матрицата от точки, изобразяваща символа на экрана. Дефинираните в алтернативния набор символи могат да бъдат изменени от потребителя, като се записват различни стойности в адреса на паметта, където е описан съответният символ. Така се оформят нови специални символи, създава се различен шрифт.

След двета набора от символи се намира областта, запазена за описание на экрана (от # BB80 до # BFDF). Накрая остава малка област с допълнителна памет за програми на машинен език и вход--изход (от # BFE0 до # BFFF).

БЕЙСИК предлага няколко специализирани команди за работа с паметта. Например HIMEM определя най-високия адрес, достъпен за програма на БЕЙСИК. Командите POKE и DOKE, функциите PEEK и DEEK служат за четене и запис в клетки на паметта.

Подпрограми и адреси в постоянната памет

Записаните в постоянната памет подпрограми могат да се извикват от програми на БЕЙСИК с команда CALL. За повечето от тях ще бъде достатъчно да заредите съответните регистри на микропроцесора, преди да се изпълни переход към тях с инструкция JSR. Параметрите на подпрограми за графика и звук се записват в област от паметта с начален адрес # 2E0. Параметрите се представят като 16-битови числа със знак. Адресът на областта с параметри се означава с PARAMS + x. След изпълнението си подпрограмата поставя код на грешка в байта с адрес PARAMS + 0. Преди извикване на подпрограма с CALL в PARAMS + 0 следва да се запише 0. Всички адреси по-долу са шестнадесетични. Всички подпрограми ползват акумулатора A и индексни регистри X и Y (т. е. променят тяхното съдържание) освен ако изрично не е посочено друго.

VDU Адрес: F77C

Извежда символ на экрана и придвижва маркера с 1 позиция надясно.

Параметри: X = код на извежданния символ

Резултати: няма

Ползвани регистри: няма

STOUT Адрес: F865

Извежда съобщение на реда за състояние на экрана (на най-горния ред -- адреси от BB80 до BBA7). Съобщението трябва да завърши със символ NULL.

Параметри: A = адрес на съобщението (младши байт)

Y = адрес на съобщението (старши байт)

X = хоризонтална позиция за първия символ

Резултати: X = следваща позиция

GTORKB Адрес: EB78

Четене на символ от клавиатурата. Символът се получава с текущата скорост на повторение, определена от байтове на адреси # 24E и # 24F. Байт # 24E е закъснението, обикновено 30 ms, преди клавишът да започне да се повтаря. При постоянно натиснат клавиш честотата на повторение се задава в # F (обикновено също 30 ms). За да получите символите с максимална скорост, запишете единици и в двета байта (24E и 24F).

Параметри: няма

Резултати: А = код ASCII на символа

Флаг за знак +

Флаг за знак -

Ползвани регистри: няма

PRTCHR Адрес: F5C1

Извежда един символ към печатащото устройство.

Параметри: А = код ASCII на символа

Резултати: няма

OUTLED Адрес: E74A

Извежда начална синхронизираща поредица (9 символа с код ASCII # 16,SYN)към касетофон.

Параметри: няма

Резултати: няма

Забележка. Скоростта за обмен с касетофона се задава от байт # 24D. Скоростта е висока (2400 bits/s) при стойност 0 и ниска (300 bits/s) при стойност, по-голяма от 0.

GETSYN Адрес: E735

Чете синхронизираща поредица от байтове от касетофон.

Параметри: няма

Резултати: няма

Ползвани регистри: А, X

OUTBYT Адрес: E65E

Записва един байт на касетофон.

Параметри: А = код ASCII на символа

Резултати: няма

Ползвани регистри: А

RDBYTE Адрес: E6C9

Чете един байт от касетофон.

Параметри: няма

Резултати: А = код ASCII на прочетения символ

Ползвани регистри: А

CURSET Адрес: F0C8

Параметри: PARAMS+1: стойност на X
PARAMS+3: стойност на Y
PARAMS+5: стойност на F

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

CURMOV Адрес: FDFD

Параметри: PARAMS+1: стойност на X
PARAMS+3: стойност на Y
PARAMS+5: стойност на F

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

DRAW Адрес: F110

Параметри: PARAMS+1: стойност на X
PARAMS+3: стойност на Y
PARAMS+5: стойност на F

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

Ползвани регистри: A, X

CHAR Адрес: F12D

Параметри: PARAMS+1: код ASCII на символ
PARAMS+3: код на азбуката; 0 за стандартна 1 за алтернативна

PARAMS+5: стойност на F

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

CIRCLE Адрес: F37F

Параметри: PARAMS+1: радиус
PARAMS+3: стойност на F

Резултати: PARAMS+0 — код за грешка 1 при стойности извън допустимия интервал

PATRN Адрес: F11D

Параметри: PARAMS+1: стойност на шаблона PATTERN

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

Ползвани регистри: A

POINT Адрес: F1C8

Параметри: PARAMS+1: стойност на X
PARAMS+3: стойност на Y

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

PARAMS+1: 0 — фон;
1 — основен цвят

FILL *Адрес:* F268

Параметри: PARAMS+1: брой редове
PARAMS+3: брой клетки
PARAMS+5: стойност

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

PAPER *Адрес:* F204

Параметри: PARAMS+1: цвят

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

INK *Адрес:* F210

Параметри: PARAMS+1: цвят

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

PING *Адрес:* FA97

Достъпна по адрес.

Забележка. Всички програми за звук използват същите процедури за предаване на параметри както графичните програми.

SHOOT *Адрес:* FA85

Достъпна по адрес.

EXPLD *Адрес:* FACB

Достъпна по адрес.

ZAP *Адрес:* FAE1

Достъпна по адрес:

KBBEEP *Адрес:* FB14

Достъпна по адрес. Предизвиква звуков сигнал при натискане на клавиш.

CONTBP *Адрес:* FB2A

Достъпна по адрес. Предизвиква звуков сигнал при CTRL.

SOUND *Адрес:* FB40

Параметри: PARAMS+1: канал

PARAMS+3: период на звука

PARAMS+5: сила на звука

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

MUSIC Адрес: FC18

Параметри: PARAMS+1: канал (1—3)

PARAMS+3: октава (0—7)

PARAMS+5: нота (1—12)

PARAMS+7: сила на звука (0—15)

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

PLAY Адрес: FBD0

Параметри: PARAMS+1: канал за тон

PARAMS+3: канал за шум

PARAMS+5: модулиране

PARAMS+7: период на мудилирането

Резултати: PARAMS+0 — код на грешка 1 при стойности извън допустимия интервал

W8912 Адрес: F590

Въвежда данни от регистър X в музикалния синтезатор. Подпрограмата поддържа клавиатурата достъпна.

Параметри: A = обръщение към синтезатора

X = изходни данни

Резултати: няма

Адреси в страница 0 и страница 2

Страница 0

LINWID	#0031	— широчина на реда
TXTTAB	#009A—#009B	— начален адрес на програмата
VARTAB	#009C—#009D	— начален адрес на областта за променливи
ARYTAB	#009E—#009F	— начален адрес на областта за масиви
STREND	#00AD—#00A1	— краен адрес на областта за низове
MEMSIZ	#00A6—#00A7	— краен адрес на динамичната памет (HIMEM)
CHRGET	#00E2—#00E7	— получава символ от програмата
CHRGOT	#00E8	— инструкция LDA
TXTPRT	#00E9	#0DEA— указател към интерпретирания символ
SKPSPC	#00EB	#00EE— подпрограма за прескачане на интервали
QNUM	#00EF	#00F8— проверка за цифра
CHRRTS	#00F9	— инструкция RTS

Страница 2

KEYAD	#0208	— адрес на натиснатия клавищ
KBSTAT	#0209	— избор на горен регистър #A4=лявSHIFT #A7=десенSHIFT
CAPLCK	#020C	— #80=само горен регистър
PAT	#0213	— параметър PATTERN за команди CIRCLE и DRAW
CURX	#0219	— координата x за графичен режим
CURY	#021A	— координата y
GRA	#021F	— режим на экрана: 1=HIRES, 0=TEXT/LORES
XVDU	#238	— переход към подпрограма VDU
XGETKY	#23B	— переход към подпрограма за въвеждане от клавиатура (GTORKB)
XPRTCH	#23E	— переход към подпрограма за извеждане на печатащо устройство (PRTCCHR)
XSTOUT	#241	— переход към подпрограма за извеждане на реда за състояние (STOUT)
INTFS	#244	— переход към подпрограма за обработване на прекъсвания
NMIJP	#247	— переход към подпрограма за обработване на немаскируеми прекъсвания
INTSL	#24A	— връщане от подпрограма за обработване на прекъсвания
TSPEED	#24D	— скорост на четене — запис на касетофон: 0 — 2 400 bits/s; >0 — 300 bits/s
KBDLY	#24E	— време на изчакване преди автоматично повторение на натиснат клавищ
KBPRT	#24F	— честота на повторение на натиснатия клавищ
PWIDTH	#256	- широчина на ред за печатащото устройство (стандартно 80)
VWIDTH	#257	широкина на ред за экрана (стандартно 40)
CURROW	#268	текущ ред на маркера
MODEO	#26A	битовете в гози байт дефинират текущото състояние на различни функции: бит инструкция 7 не се използва 6 не се използва 5 1=защитени колони 0 и 1 на экрана 4 1=последният изведен символ е ESC 3 1=изключва звуковия сигнал 2 не се използува

		1	= включено VDU
		0	= включен маркер
BGND	#026B	—	код на цвета на фона (хартия) + 16
FGND	#026C	—	код на основния цвят (мастило)
CURON	#0270	—	флаг за включване — изключване на маркера
CURINV	#0271	—	флаг за смяна на маркера
TIMER1	#0272—#0273	—	таймер на клавиатурата
TIMER2	#0274—#0275	—	таймер на маркера
TIMER3	#0276—#0277	—	16-битов таймер със стъпка 0.01 s; използва се от WAIT
VDUL2	#0278—#0279	—	адрес на втория ред от екрана
VDUL1	#027A—#027B	—	адрес на първия ред от екрана
VDUCH	#027C—#027D	—	брой на символите за изместяване при скрол, обикновено 26 реда × 40 символа
NOROWS	#027E	—	брой редове на екрана
ICHAR	#02DF	—	код на последния натиснат клавиш
PARAMS	#02E0	—	буфер за параметри на подпрограми за графика и звук

Забележка. Таймерът отчита времето в посока намаляване (обратно на хронометъра).

СХЕМИ НА КУПЛУНГИ



КУПЛУНГ ЗА РАЗШИРЕНИЕ

+5V	A10	A9	A8	A7	A6	A5	D5	A2	A1	A0	A3	D2	RW	10	02	MAP
33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
GND	A11	A12	A13	A14	A15	D7	A4	D4	D3	D6	D1	Dd	IRQ	I/O	RST	ROMDIS

КУПЛУНГ ЗА ПЕЧАТАЩО УСТРОЙСТВО

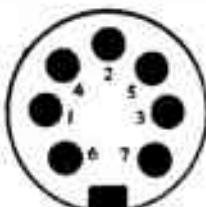
ACK	D7	D6	D5	D4	D3	D2	D1	D0	STB
19	17	15	13	11	9	7	5	3	1
20	18	16	14	12	10	8	7	5	2
↑	↑	↑	↑	↑	GND	↑	↑	↑	↑

КУПЛУНГ ЗА МОНИТОР



1. ЧЕРВЕНО
2. ЗЕЛЕНО
3. СИНЬО
4. СИНХРОН
5. ЗЕМЯ

КУПЛУНГ ЗА КАСЕТОФОН



1. ИЗХОД
2. ЗЕМЯ
3. ВХОД
4. ЗВУК
- 5.
6. РЕЛЕ
- 7.

ЛИТЕРАТУРА

1. Сираков, П. А., О. П. Вълчев. Ключ за компютър. С., Народна младеж, 1985.
2. Петров, П. БЕЙСИК за персонални компютри. С., НУЦ, 1985.

АЗБУЧЕН УКАЗАТЕЛ

ABS 29	IF...THEN...[ELSE] 42	REM 17
AND 26	INK 75	REPEAT 44
ASC 55	INPUT 63	RESTORE 67
ATN 34	INT 30	RETURN 46
CALL 58	KEY\$ 65	RIGHTS 52
CHAR 82	LEFT\$ 50	RND 36
CHR\$ 55	LEN 50	RUN 20
CIRCLE 81	LIST 19	SCRN 76
CLEAR 18	LLIST 20	SGN 29
CLOAD 21	LN 36	SHOOT 85
CLS 70	LOG 35	SIN 31
CONT 49	LORES 73	SOUND 89
COS 32	LPRINT 69	SPC 70
CSAVE 22	MID\$ 51	SQR 34
CURMOV 80	MUSIC 87	STOP 48
CURSET 79		STORE 25
DATA 66		STR\$ 53
DEEK 57	NEW 18	TAB 72
DEF FN 38	NOT 27	TAN 33
DIM 23	ON 41	TEXT 69
DOKE 58	OR 26	TROF 63
DRAW 81		TRON 62
END 49	PAPER 74	TRUE 27
EXP 35	PATTERN 80	
EXPLODE 86	PEEK 56	UNTIL 45
FALSE 28	PI 37	USR 59
FILL 83	PING 86	VAL 54
FOR...TO...[STEP]...NEXT 43	PLAY 88	
FRE 61	PLOT 75	WAIT 59
GET 64	POINT 84	
GOSUB 40	POKE 57	ZAP 85
GOTO 39	POP 47	
GRAB 61	POS 72	
HEX\$ 56	PRINT 67	
HIMEM 60	PULL 45	
HIRES 78	READ 65	
	RECALL 26	
	RELEASE 62	

СЪДЪРЖАНИЕ

ПРЕДГОВОР	5
ПОДГОТОВКА ЗА РАБОТА С КОМПЮТЪРА	7
Клавиатура	7
Периферия	10
Работа с касетофон	11
Грижи за касетите	13
Друга периферия	14
Включване на компютъра	15
КОМАНДИ И ФУНКЦИИ	17
Коментар	17
Операции над цели програми	18
Работа с масиви	23
Логически функции и константи	26
Математични функции	29
Управление	39
Операции с низове	50
Помощни команди и функции	56
Входно-изходни операции	63
Графика	73
Музикални възможности	84
ПРИЛОЖЕНИЯ	92
Съобщения за грешки	92
Таблица на кодовете ASCII (КОИ--7)	95
Управляващи кодове ESCAPE	97
Оперативна памет	98
Схеми на куплунги	107
ЛИТЕРАТУРА	107
АЗБУЧЕН УКАЗАТЕЛ	108

Домашен компютър Правец—8Д

Автори: *Орлин Петров Вълчев*
инж. Пенчо Ангелов Сирakov
Димитър Христов Вавов

Рецензенти: *Здравко Божилов Василев*
ст. н. с. к. т. и. инж. Пламен Иванов Вачков

Националност българска

Първо издание

95331 15231

Код 03 3202—34—86

Изд. № 15514

Научен редактор инж. Нина Денева

Художник Любомир Михайлов

Художествен редактор Мария Димитрова

Технически редактор Иван Георгиев

Коректор Цветанка Лулчева

Дадена за набор на 20.II.1986 г.

Подписана за печат м. март 1986 г.

Излязла от печат м. март 1986 г.

Формат 60 × 90/16

Печ. коли 7

Изд. коли 7

УИК 6,78

Тираж 50 000 + 106

Цена 1,50 лв.

Държавно издателство „Техника“, бул. Руски 6, София

Държавна печатница „Георги Димитров“ — София

ТАЛОН

към книгата

**„ДОМАШЕН КОМПЮТЪР
ПРАВЕЦ— 8Д“**





1,50 mn.