```
* файл GETKEY
* последната корекция е била на
* 12-юли-1987
* този файл съдържа само една процедура - GETKEY

*****************************************************************
*     Подпрограма 'GETKEY'                                      *
*                                                               *
*    I. Извършвани действия:                                    *
*       Прочитане на 'ASCII' код от клавиатурата.               *
*       Динамично следене състоянието на направленията         *
*         и предприемане на съответни действия според           *
*         приложения граф.                                      *
*       Извеждане на 'таймер' в [S] оставащо време за           *
*          предаване, ако това е разрешено.                     *
*       Инвертиране на пореден символ от предавания текст       *
*          върху екрана 'режим шоу', ако това е разрешено       *
*       Когато, поради смяна на състоянието на някое нап-       *
*          равление, се налага обновяване на съдържанието       *
*          на менюто подпрограмата 'GETKEY' елиминира           *
*          всички извикали я преди това подпрограми и           *
*          прави скок към  последният извикал я екран с         *
*          което го обновява и попада отново в себеси,ако       *
*          това е разрешено.                                    *
*                                                               *
*****************************************************************
*     Ползвани външми подпрограми:                              *
*                                                               *
*      Име          Предназначение                              *
*   PRDEC:          Извежда на екрана цяло десетично число.     *
*   MOVCURS         Използува се в режим 'шоу'.                 *
*   PUSK:           Генерира нов текст от същия вид и го        *
*                   стартира за предаване.                      *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*            Таблица с константни отмествания за                *
*            ровене из буферите и за началните хор.             *
*            позиции за 'таймери'                               *
*                                                               *
MITBL0      DFB  0,$02,$12,$22,$32                             *
MITBL1      DFB  0,$00,$10,$20,$30                             *
HTABTBL     DFB  0,5,15,25,35                                  *
*                                                               *
*****************************************************************

GETKEY      TXA                 ;Запазване на
            PHA                 ;регистрите
            TYA
            PHA

            LDA  #0             ;Нулиране на броячи за
            STA  MILPOS         ;режим 'шоу'
            STA  MIHPOS         ;използувани за пп. MOVCURS
*********************************
* изписване на курсор           *
```

```
*********************************

            LDY   CURSX         ;текуша позиция на курсора
            TYA                 ;да се запази за да бъде
            PHA                 ;възстановена в края.
            LDA   CURSY
            PHA
            DO    ORIC
            LDA   #SP+$80
            ELSE
            LDA   #SP
            FIN

            STA   (BAS),Y       ;Изписване на инверсен интервал
.

            LDX   #5            ;Брой на направленията + 1
            BNE   LOOK          ;Вместо 'JMP' само за първо вли
зане.
                                ;тъй като трябва направленията
да
                                ;се обработят поне веднъж.
GETKEY1     LDX   #5            ;Циклене на клавиатура и направ
ления
            LDA   KBD           ;Натиснат ли е клавиш ?
            BPL   LOOK          ;уви не е ще трябва да прегледа
ме
                                ;направленията !
*****************************************
*      Изтриване на курсор              *
* и нормален край на подпрограмата      *
*****************************************
            PLA                 ;Възстановяване на
            STA   CURSY         ;запомнените позиции
            PLA                 ;на курсора.
            STA   CURSX
            JSR   BASCALC       ;Преизчисляване на базата
            LDY   CURSX

            DO    ORIC          ;Зареждане на
            LDA   #SP           ;символ ' ' за да се
            ELSE                ;изтие курсора.
            LDA   #SP+$80
            FIN

            STA   (BAS),Y       ;Реално изтриване
            PLA                 ;Възстановяване на
            TAY                 ;запомнените преди
            PLA                 ;това регистри
            TAX
            LDA   KBD           ;Прочитане на клавиатурата
JWE         AND   #$7F          ;Маска за прочетен симол
            STA   KBD
            RTS                 ;Нормален край на пп.


WAIT_KEY    LDA   #0            ;Независима пп, която
            STA   KBD           ;извършва само първата функ-
JW          LDA   KBD           ;ция на 'GETKEY' и се изпол-
            BPL   JW            ;зува само в първия екран.
            BMI   JWE

LOOK        DEX                 ;Следващо направление
```

```
                BEQ   GETKEY1       ;Свършиха ли ? тогава давай
                                    ;отначало.
                LDY   MITBL1,X      ;Узнаване на адреса на
                                    ;системния телеграфен буфер
                LDA   TTAB,Y
                STA   UKLT
                INY
                LDA   TTAB,Y
                STA   UKHT
                LDY   #0
                LDA   (UKLT),Y
                STA   MISTATT       ;Край на установяването
                BEQ   NOTIME        ;Ако е 0 не визуализирай
                                    ;време
*****************************************************************
*       Извеждане на 'таймер' в [S] оставащо време за          *
*          предаване, ако това е разрешено.                     *
*****************************************************************
                LDA   CURSX         ;Временно запазване
                PHA                 ;на координатите на
                LDA   CURSY         ;курсора тъй като може да
                PHA                 ;потрябват
                DO    ORIC
                LDA   #27           ;номер на ред
                ELSE                ;на който ще се извежда
                LDA   #23           ;'таймер'
                FIN
                STA   CURSY
                LDY   HTABTBL,X     ;Хор. позиция за даденото
                                    ;направление
                STY   CURSX
                LDY   #LUSTIME      ;Прочитане на текущо време
                LDA   (UKLT),Y      ;подържано от ПП за реално
                PHA                 ;време.
                INY
                LDA   (UKLT),Y
                TAY
                PLA
                JSR   PRDEC:        ;същинско отпечатване
                LDY   CURSX         ;елиминиране на поне
                DO    ORIC          ;два, може би останали от
                LDA   #SP           ;предишния път
                ELSE                ;интервала
                LDA   #SP+$80
                FIN
                STA   (BAS),Y
                INY
                STA   (BAS),Y       ;край на елиминирането
                PLA                 ;възстановяване на временно
                STA   CURSY         ;запазените координати на
                PLA                 ;курсора
                STA   CURSX

NOTIME          LDY   MITBL0,X      ;Установяване на адресите
                LDA   TTAB,Y
                STA   UKLA          ;на потребителския
                INY
                LDA   TTAB,Y        ;и на системния 'ASCII'
```

```
            STA  UKHA
            INY                 ;буфер.
            INY
            INY
            LDA  TTAB,Y
            STA  UKLA1
            INY
            LDA  TTAB,Y         ;Край на установяването.
            STA  UKHA1
*********************************
* Проверка за визуализация т.е.
* режим "шоу"
*********************************

            CPX  WDIR           ;Съвпада ли обработваното
                                ;направление с това което е
                                ;на екрана ?
            BNE  NOSHOW         ;не
            LDA  FLAGSHOW       ;Зададен ли е режим "шоу" ?
            BEQ  NOSHOW         ;не
ЭSEEK       LDY  #2
            LDA  MILPOS         ;дали мойта позиция съвпада
            CMP  (UKLT),Y       ;с позицията в буфера за
            BNE  INCMIPOS       ;предаване ?
            INY                 ;ако не то догонвай !
            LDA  MIHPOS
            CMP  (UKLT),Y
            BNE  INCMIPOS
            BEQ  NOSHOW
INCMIPOS    LDA  MILPOS         ;Дали е върху първият символ
            ORA  MIHPOS         ;ако не е, не установявай
            JSR  MOVCURS        ;началми координати !

NOSHOWO     INC  MILPOS
            BNE  ЭSEEK
            INC  MIHPOS
            BNE  ЭSEEK          ;край на режим "шоу"

*****************************************************************
*
* Начало на последователна проверка на състоянието на
* поредно направление според граф приложен в документацията
*

NOSHOW      LDY  #0
            LDA  (UKLA),Y       ;Статус на сист. "ASCII" буфер
            STA  MISTATA        ;ще ни трябва и за в бъдеше.
            AND  #%00100000     ;Празен ли е той ?
            BEQ  NOEMPTY        ;Не виж за друго.
            LDA  MISTATT        ;А телеграфният предаден ли е
            BNE  LOOK\          ;не тогава го изчакай да свър-
                                ;ши с текущото предаване
            LDA  #%01000000     ;и установи статус 'няма текст
            STA  (UKLA),Y       ;за предаване'.
            STA  (UKLA1),Y
            JSR  NEW            ;Обнови екрана !
LOOK\       JMP  LOOK           ;Виж следващо направление.
```

```
NOEMPTY     LDA  MISTATA
            AND  #%10000000
            BNE  LOOKEND
            LDA  MISTATA
            AND  #%00010000    ;Пауза м/у текстовете
            BNE  WAITTIME
            LDA  MISTATA
            AND  #%00001000    ;Работи с пълен буфер
            BEQ  LOOK1
            LDA  MISTATT
            BNE  LOOK1
            LDY  #LUSTIME
            LDA  #20           ;Установяване на 20 [ESC]
            STA  (UKLT),Y
            LDA  #0
            INY                ;за режим 'таймер'
            STA  (UKLT),Y
            LDY  #0
            LDA  #8            ;и на необходимия статус
            STA  (UKLT),Y      ;за пп 'реално време'
            LDA  #%00010000    ;и за
            STA  (UKLA),Y      ;системен и
            STA  (UKLA1),Y     ;потребителски 'ASCII'
            JSR  NEW           ;Обнови екрана !
LOOK1       JMP  LOOK

LOOKEND     LDA  MISTATT       ;Предаден ли е текушия текст ?
            BNE  LOOK1         ;не
            LDA  #%00000100    ;да тогава установи 'работи с
            STA  (UKLA),Y      ;празен буфер'
            STA  (UKLA1),Y
            JSR  NEW           ;и обнови екрана.
            JMP  LOOK

WAITTIME    LDY  #LUSTIME      ;Младшата част на таймера = 0
            LDA  (UKLT),Y
            BNE  LOOK1         ;не, расейвай се
            LDY  #0
            LDA  (UKLT),Y      ;А дали е в режим 'пауза м/у
            CMP  #8            ;два текста ?
            BNE  LOOK1         ;не, расейвай се
            LDA  WDIR          ;запази 'WDIR'
            PHA
            STX  WDIR          ;Установи # направление за
            JSR  PUSK:         ;пускане и го стартирай !
            LDX  WDIR          ;Въстанови 'WDIR'
            PLA
            STA  WDIR
            JSR  NEW           ;Обнови екрана.
            JMP  LOOK

NEW         LDA  ZABR          ;Разрешено ли е пълното
            CMP  #2            ;обновяване на екрана ?
            BNE  NEWYES        ;Да тогава авариен изход.
            CPX  WDIR          ;А дали да обновя само статуса
            BNE  RTS1          ;не тъй като визуализираното
                               ;направление е друго.
            TXA                ;Да тогава запази 'X'
```

```
            PHA
            LDX   #0
            LDA   CURSX         ;Запази текущите координати
            PHA
            LDA   CURSY         ;на екрана
            PHA
            STX   CURSX         ;установи нови = на 0,0
            STX   CURSY
            INX                 ; 'X' = 1, входен парам. за
            JSR   STATUS:       ;пп 'STATUS:'
            PLA
            STA   CURSY         ;Възстанови запомнените
            PLA
            STA   CURSX         ;координати
            PLA
            TAX                 ;и 'X'
RTS1        RTS                 ;Край.
*****************************************************************
*                                                               *
*           Авариен изход от    'GETKEY'. Тя елиминира          *
*           всички извикали я преди това подпрограми и          *
*           прави скок към  последният извикал я екран с        *
*           което го обновява и попада отново в себеси          *
*                                                               *
*****************************************************************

NEWYES      CMP   #0              ;Разрешено ли е обновяването ?
            BNE   RTS1            ;не, расейвай се
            LDA   WDIR            ;Всички направления ли са на
                                  ;екрана
            BEQ   NEWYES1         ;Да и трябва да бъдат обновени
            CPX   WDIR            ;Само текушото направление
            BNE   RTS1
NEWYES1     LDA   #0
            STA   KBD             ;Нулиране на клавиатурата

* Елиминиране на предишни стекови операции

            PLA
            PLA                   ;На 'JSR NEW'
            PLA
            TAY
            PLA
            TAX                   ;На запазените в началото рег.
            PLA
            PLA                   ;На 'JSR GETKEY'
            PLA
            PLA                   ;На 'JSR WAIT'
            PLA
            PLA                   ;На 'JSR SCREEN'
            LDA   SCREEN+1        ;Установяване на
            PHA
            LDA   SCREEN          ;Текущ адрес за начало на
            PHA                   ;Екран и преход към него !
            RTS

********** край на GETKEY
```

```
* файл  TIME
* последната корекция е била на
* 1о·юли·1987
* Основа част за работа в реално време.
*
*****************************************************************
*
* процедура  'Time'
* -----------------
*
* задача :  Обработка на процеси протичаши в реално време
*
* входни данни :    Състояние на системните телеграфни
*                   буфери
*                   Състояние на буфер за принтер
*
* изходни данни :   Състояние на системните телеграфни
*                   буфери
*                   'ASCII' код прочетен от клавиатурата
*                   в клетка 'KEYOUT'.
*                   Изход към принтер чрез състояние на
*                   буфер за принтер.
*
* ползвани клетки от EQU·зоната : OUTMORS
*                                 KBD (KEYOUT)
* начин на работа : Обработват се 4те направления в
*                   резултат на което се получава изходен
*                   байт 'OUTMORS' който бива изведен към
*                   леч за радиозала. Неговите първи 4 бита
*                   определят кое направление да пищти.
*                   След това се проверява поредната пресе·
*                   чна точка на клавиатурата за натиснат
*                   клавиш. Ако той е натиснат се
*                   предприемат действия за генериране на
*                   съответен 'ASCII' код, който се поставя
*                   в изходния байт 'KEYOUT'.
*                   После се проверява дали е запълнен
*                   буфера за печат и ако това е така и
*                   принтера е готов се изпраща към него
*                   поредния символ.Когато се срешне символ
*                   равен на 0 това означава че буфера е
*                   свършил и извеждането се прекратява,
*                   като едновременно с това се нулира ин·
*                   дикатора за пълен буфер,
*
* външни процедури : няма
*
*****************************************************************

ORIK            KBD

                DO    ORIK
TIMEINIT        =     1000          ;Концтанта за инициализиране
                ORG   $CB00
                ELSE
TIMCINIT        =     1014          ;на таймер на 'VIA'
                ORG   $5B00
                FIN
```

```
ZPAGE           EQU   $60
ABSOLUTE        EQU   $F60
STBL            EQU   $E00
KEYOUT          EQU   $2DF

                DUM   ZPAGE
UL              DS    1             ;работни клетки
UH              DS    1
UKL1            DS    4             ;указатели на пореден
UKH1            DS    4             ;адрес в буфера на нап.
DELL1           DS    4             ;указатели за изчакване на
DELM1           DS    4             ;поредно време за елемент
UKL             DS    1             ;работни клетки
UKH             DS    1
OUTMORS         DS    1             ;изходен байт съдържащ
                                    ;четирите направления
* 1 в тези клетки означава че се чака 7 в статус
WAIT1           DS    4             ;указатели за изчакване на
                                    ;време
TIMEL           DS    1             ;Собствен брояч на секунди
TIMEH           DS    1
MSTATUS         DS    1             ;
UKLP            DS    1             ;Указатели за адресиране на
UKHP            DS    1             ;елемент от принтерски буфер
                DEND

                DUM   ABSOLUTE
SVEA            DS    1             ;Клетки за запазване на
SVEX            DS    1             ;регистрите на 6502
SVEY            DS    1
SVEBYT1         DS    4             ;за запазване на поредния
                                    ;предаван символ
FLAGINC1        DS    4             ;
* дефиниции на клетки за работа с клавиатурата
колона          DS    1             ;Колона на матрицата
ред             DS    1             ;Ред на матрицата
KEYOLD          DS    1             ;Предишен клавиш
KEYSHIFT        DS    1             ;За натиснат регистър
KEYWRK          DS    1             ;Работна
KEYTIME         DS    1             ;Време за повторение
KEYTIMEL        DS    1             ;Време за първоначално
KEYTIMEH        DS    1             ;задържане
PRFLAG          DS    1             ;Флаг за отпечатване на текст
TPFE            DS    4             ;Timer , pause flag enable
PRCOUNT         DS    1             ;
                DEND

* дефиниране на параметрите на даден буфер
STATUS          =     0
TRADRL          =     2
точка           =     4
тире            =     6
паузасим        =     8
паузаго         =     $A
LUSTIME         =     $C
BUFSTART        =     $10
```

```
BUFFRS          =       $A000

                DO      ORIK
VIA             =       $300
                ELSE
VIA             =       $DCA0
                FIN

DRB             =       VIA+0
DRA             =       VIA+1
DDRB            =       VIA+2
DDRA            =       VIA+3
T1CL            =       VIA+4
T1CH            =       VIA+5
T1LL            =       VIA+6
T1LH            =       VIA+7
T2CL            =       VIA+8
T2CH            =       VIA+9
SHIFTR          =       VIA+10
ACR             =       VIA+11
PCR             =       VIA+12
IFR             =       VIA+13
IER             =       VIA+14
DRANH           =       VIA+15

* дефиниции на константи

предаден        =       0
готов           =       4
работисе        =       5
пауза           =       6
възстанов       =       7
TIMER           =       8

*****************************************************************
*
* програма за първоначално инициализиране
*
* да се включи в блока за инициализиране на системата
*
*****************************************************************

INIT            SEI                     ;Забрана на прекъсването
                LDA   #0                ;Начални установявания
                LDX   #3
CL              STA   WAIT1,X           ;Нама време за изчакване
                STA   TPFE,X            ;Нама назначени паузи
                DEX
                BPL   CL
                STA   KEYSHIFT          ;Нама натиснат регистър
                STA   PRFLAG            ;Нама текст за отпечатване
                STA   PRCOUNT
                LDA   #0
                STA   OUTMORS           ;Няма изход за радиозала
                LDA   #$FE
                STA   колона            ;Начална колона
                LDA   #7
                STA   ред               ;и ред за матрицата на
```

```
                                    ;клавиатурата
             LDA   #$FF
             STA   KEYOLD           ;Няма стар клавиш
             STA   DDRA             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             LDA   #%11110111       ;
             STA   DDRB             ;
             LDA   #$40             ;
             STA   IFR              ;
             LDA   #$C0             ;
             STA   IER              ;
             LDA   #%11101110       ;
             STA   PCR              ;
             LDA   #%01000000       ;
             STA   ACR              ;
             LDA   #7               ;
             LDX   #$40             ;
             STA   DRANH            ;
           ->LDY   #$EE             ;  Инициализация на входно-из-
             STY   PCR              ;
           ->LDY   #$DC             ;  ходните порти на 'VIA' и
             STY   PCR              ;
             STX   DRANH            ;  на 'SOUND' процесор за
           ->LDY   #$EC             ;
             STY   PCR              ;  работа с леч за радиозала
           ->LDY   #$DC             ;          клавиатура
             STY   PCR              ;          принтер
             LDA   #<TIMEINIT       ;
             STA   T1LL             ;  както и иницизлизация на
             STA   KEYTIMEL         ;
             LDA   #>TIMEINIT       ;    TIMER1 на 'VIA' за гене-
             STA   T1LH             ;    риране на интервал 1 [ms]
             STA   T1CH             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             STA   KEYTIMEH         ;
             DO    ORIK
             LDA   #<IRQ            ;Инизиализация на вектор
             STA   $245
             LDA   #>IRQ            ;за IRQ за ORIC
             STA   $246
             ELSE
             LDA   #<IRQ            ;Инизиализация на вектор
             STA   $3FE
             LDA   #>IRQ            ;за IRQ за Правец 82
             STA   $3FF
             FIN
             LDA   #10
             STA   KEYTIME          ;10 [ms] за повторение
             LDA   #$8D             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             STA   BUFPRN+1         ; Инициализация на принтер
             STA   BUFPRN           ; 'CR' към него за да изчи-
             LDA   #0               ; сти буфера си.
             STA   BUFPRN+2         ;
             LDA   #BUFPRN+1        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             STA   UKLP             ;Начален адрес на буфер за
             LDA   #>BUFPRN+1
             STA   UKHP             ;принтер
             CLI
SETTIME      LDA   #1000            ;Инициализация на собствен
             STA   TIMEL            ;брояч на секунди
```

```
                LDA  #>1000
                STA  TIMEH
                RTS                 ;Край на началното
                                    ;инициализиране
*****************************************************************
*
*       Константи за установяване в '0' на битове за
*                 радиозала
*
SETT00          DFB  %11111110,%11111101
                DFB  %11111011,%11110111

*****************************************************************
*
*       Константи за установяване в '1' на битове за
*                 радиозала
*
SETT01          DFB  1,2,4,8

*****************************************************************
*
*       Първоначална група за предаване еквивалент
*                 на 'ЖЖЖ='
*
STARTBL         DFB  $18,$18,$18,$8C,0

*****************************************************************
*
*     Подпрограма за стартиране на текст
*
*****************************************************************

STARTO          LDA  #работисе      ;Установяване на състояние
                STA  (UKL),Y        ;'РАБОТИ СЕ'
                TYA                 ;'Y' винаги = 0
                STA  WAIT1,X        ;Няма време за изчакване
                STA  FLAGINC1,X     ;Не се предава фалшива група
                RTS

*****************************************************************
*
*     Главна програма обработваща прекъсвания.
*
IRQ             SEI                 ;Забрана на прекъсване
                DO   ORIK
                ELSE
                LDA  $45            ;Въстан. на 'A' - само за
                                    ;'Правец 82'
                FIN
                STA  SVEA           ;Запазване на входните
                STY  SVEY
                STX  SVEX           ;регистри
                LDA  TIMEL          ;Ендократно намаляване на
                BNE  IRQ2           ;собствения брояч
                DEC  TIMEH          ;секунди с 1
IRQ1            DEC  TIMEL

                LDX  #3             ;'X' съдържа # на обработвано
```

```
                                    ;направление
IRQ3        LDY   BUFTBL,X          ;Установяване на начален
            LDA   STBL,Y            ;адрес на буфера за
            STA   UKL               ;текущото направление
            INY
            LDA   STBL,Y
            STA   UKH
            LDY   #0                ;Установяване на 'Y' = 0,
                                    ;Много рядко ще се променя !
            LDA   (UKL),Y
            STA   MSTATUS           ;буфер.
IRQ7        LDA   TIMEL             ;Проверка за край на
            ORA   TIMEH             ;собствения брояч
            BNE   IRQ4              ;все още не е изтекла 1 секунда
            LDA   MSTATUS
            CMP   #TIMER            ;В режим 'таймер' ли съм ?
            BEQ   IRQTIMER
            CMP   #9                ;В аесолютна пауза ли съм
            BNE   IRQ6
            JMP   LOOKNEXT          ;да и тогава не прави нищо !

IRQ6        LDA   FLAGINC1,X        ;'ЖЖ=' или 'АР' ли предавам ?
            BNE   IRQ4              ;да, не обработвай таймери
            LDA   TPFE,X            ;разлешен ли е ражим таймери ?
            LSR
            BCS   IRQ4              ;не, расейвай се !
IRQTIMER    LDY   #LUSTIME          ;Достигнал ли е таймерът на
            LDA   (UKL),Y           ;съответното направление
            INY
            ORA   (UKL),Y           ;до нула ?
            BEQ   IRQ4              ;да, расейвай се
            DEY
            LDA   (UKL),Y           ;Намаляване на таймера
            SEC
            SBC   #1
            STA   (UKL),Y
            INY                     ;на текушото направление
            LDA   (UKL),Y
            SBC   #0
            STA   (UKL),Y           ;с единица

IRQ4        LDA   WAIT1,X           ;Изчаква ли се пореден елемент
            BEQ   WRKFO1            ;да. иди и виж колко е
                                    ;останало
            LSR                     ;Очаква ли се 'Въстановяване
                                    ;от пауза
            BCC   NOWAIT7           ;не, тогава поредния елемент е
                                    ;свършил и трябва да обработим
                                    ;следвашия !
            LDA   MSTATUS
            CMP   #въстанов         ;Появил ли се е в буфера
                                    ;статус 'Въстановяване' ?
            BEQ   CONTINUE          ;да, тогава продължавай
            JMP   LOOKNEXT          ;не, чакай да се появи !


CONTINUE    JSR   START0            ;Установяване на статуси
            JMP   START1            ;Продължавай да предаваш от
```

```
                                    ;там докъдето си стигнал, като
                                    ;преди това предай 'ЖЖЖ=' !

NOWAIT7     LDA   DELL1,X       ;Намаляване с 1 на времето за
            BNE   DEC1
            DEC   DELH1,X
DEC1        DEC   DELL1,X       ;изчакване на пореден елемент
            LDA   DELL1,X
            ORA   DELH1,X       ;достигнало ли е до '0'
            BNE   LOOKNXT1      ;не, заначи ще чакаме още.
            STA   WAIT1,X       ;Маркиране на край на
                                ;изчакването
            LDA   FLAGINC1,X    ;'ЖЖЖ=' ли предаваме ?
            BNE   WRKFO1        ;да, не разрешай работата на
                                ;таймерите !
            LDA   TPFE,X        ;не, тогава да им
            AND   #$FE          ;разрешим да
            STA   TPFE,X        ;работят

WRKFO1      LDA   MSTATUS
WRKFO2      CMP   #готов        ;Готов ли е за предаване ?
            BNE   NOSTART       ;не, значи няма да започваме
                                ;сега а ще проверим другите
                                ;възможни състояния.
*****************************************************************
*
*           Начално стартиране на текст
*

START       JSR   STARTO        ;Установяване на параметри
            LDY   #TRADRL       ;Нулиране на
            STA   (UKL),Y       ;брой предадени
            INY
            STA   (UKL),Y       ;символи.
            CLC                 ;Изчисляване на
            LDA   UKL           ;действителен начален
            ADC   #BUFSTART-1   ;адрес на необходимия
            STA   UKL1,X        ;системен телеграфен
            LDA   UKH           ;буфер
            ADC   #0
            STA   UKH1,X
START1      INC   FLAGINC1,X    ;Ще предаваме 'ЖЖЖ='
            LDA   TPFE,X        ;Забрана за работата
            ORA   #1
            STA   TPFE,X        ;на таймера
            LDY   #0
            LDA   STARTBL       ;Прочети първото 'Ж'
            STA   SVEBYT1,X     ;и го подготви за предаване
            JMP   YES1          ;започни предаване

*           Проверка на други възможни състояния

NOSTART     CMP   #предаден     ;Предаден ли е поредния текст
            BNE   YESLOOP       ;не, продължавай проверката
            LDA   OUTMORS       ;да, спри звука за даденото
            AND   SETTOO,X
            STA   OUTMORS       ;направление
            JMP   LOOKNEXT      ;и виж какво става с другите.
```

```
YESLOOP
                CMP   #пауза       ;Появил ли се е сигнал 'Пауза'
                BNE   LOOKREST     ;не, продължавай проверката
                LDA   TPFE,X       ;да, тогава установи
                ORA   #2
                STA   TPFE,X       ;флаг за пауза
                BNE   YES1         ;и продължи да допредаваш
                                   ;групата.
LOOKREST
                CMP   #възстанов   ;'Възстановяване' от пауза ли е
                BNE   LOOKNXT0     ;не, продължавай проверката
                LDA   TPFE,X       ;да, свали флага за пауза
                AND   #%11111101
                STA   TPFE,X
                LDA   WAIT1,X      ;свали и флага за евентуално
                AND   #%11111110
                STA   WAIT1,X      ;започнала пауза
                JMP   YES1         ;и продължи да предаваш
LOOKNXT0        CMP   #работисе    ;Предава ли се в момента
                BEQ   YES1         ;да, е тогава го остави да се
                                   ;предава
LOOKNXT1        JMP   LOOKNEXT     ;авариен JMP поради къси
                                   ;бранчове
YES1
                LDA   OUTMORS      ;Пускане на звука за
                EOR   SETTO1,X     ;даденото направление
                STA   OUTMORS
                AND   SETTO1,X     ;Поредният елемент свършил ли е
                BNE   YES2         ;не още
                LDY   #точка       ;да, изчакай пауза м/у
                BNE   SETDEL1      ;елементи.

YES2            LDA   SVEBYT1,X    ;Извличане на поредния
                ASL   A
                STA   SVEBYT1,X    ;елемент от символа
                BCS   DELTIR       ;Ако е = '1' това е '-'
                LDY   #точка       ;иначе е '.'
                BNE   SETDEL2      ;Установяване на интервал
DELTIR          LDY   #тире        ;за изчакване на
                BNE   SETDEL2      ;поредния елемент

SETDEL1         LDA   SVEBYT1,X    ;Символа като цяло
                CMP   #$80         ;свършил ли е ?
                BEQ   ENDCHR1      ;да, ще извкваш следващ символ

SETDEL2         LDA   (UKL),Y      ;Зареждане на необходимия
                STA   DELL1,X      ;временен интервал за
                INY                ;изчакване на пореден
                LDA   (UKL),Y      ;елемент от символ.
                STA   DELH1,X
                LDA   WAIT1,X      ;И на флаг че поредният
                ORA   #$80         ;интервал трябва да бъде
                STA   WAIT1,X      ;изчакан
                JMP   LOOKNEXT     ;Виж какво става с другите.

ENDCHR1
                LDA   FLAGINC1,X   ;Последен символ ли е предаден
                BMI   SEND1        ;да, тогава предай 'АР' и спри
```

```
                JSR   INCSTBL     ;не, тогава вземи следващия.
                BEQ   ENDGR       ;Ако е '0' значи край на гупа.
                LDY   #паузасим   ;Подготовка за установяване на
                JMP   SETDEL2     ;пауза м/у символи.

ENDGR           LDA   TPFE,X      ;Разрешена ли е проверката за
                AND   #%00000100  ;край на текст ?
                BNE   ENDGR2      ;не
                LDY   #1
                LDA   (UL),Y      ;Символа след последната нула
                                  ;нула ли е ?
                BEQ   SEND0       ;да, значи текста е свършил.

ENDGR2          LDA   TPFE,X      ;Дошла ли е заявка за
                AND   #%00000010  ;режим 'пауза' ?
                BNE   SETWAIT     ;да подготви очакване на
                                  ;статус 'Възстановяване' и влез
                                  ;в режим пауза.

                LDA   MSTATUS     ;
                CMP   #7          ;Проверка за режим 'Пауза'
                BNE   ENDGR3      ;ако не се
                LDA   #работисе   ;поддържа изкуствено
                LDY   #0          ;състояние 'Работи се'
                STA   (UKL),Y     ;(Вероятно е рудиментарен
                TYA               ;остатък и трябва да
                STA   WAIT1,X     ;да се махне)
                JMP   ENDGR3

*****************************************************************
*
*           Установяване на режим 'Пауза'
*

SETWAIT
                LDA   TPFE,X
                AND   #%11111101  ;разр. на нова пауза
                ORA   #%00000001  ;забрана на TIMER
                STA   TPFE,X
                LDY   #1
                STY   WAIT1,X     ;От този момент нататъка само
                                  ;статус 'Възстановяване от пауза

                                  ;ще се анализира от програмата
                BNE   LOOKNEXT    ;Вместо JMP

ENDGR3          JSR   INCSTBL     ;Вземи следващия символ от
                                  ;текста
ENDGR4          LDY   #паузагр    ;и установи изчакване на
                JMP   SETDEL2     ;пауза м/у групи.

SEND0           LDA   #%01010100  ;'AR'
                STA   SVEBYT1,X   ;Последен символ за край на
                                  ;текст.
                DEC   FLAGINC1,X  ;FLAGINC1=$FF, т.е. текста е
                                  ;свършил.
                JMP   ENDGR4      ;Предаване на последния символ

SEND1           LDY   #STATUS     ;Физически край на текста.
```

```
                LDA   #предзаден
                STA   (UKL),Y     ;Подготовка за нов текст
                STY   WAITI,X
                TYA
                STA   FLAGINC1,X

LOOKNEXT        DEX               ;Следващо направление
                BMI   SEEKEY      ;Ако са обработени и 4те
                                  ;направления премини към IIта
                                  ; част на програмата
                JMP   IRQ3


*****************************************************************
*
*           обслужване на клавиатурата
*
*           или II част от програмата.
*
* задача :  Получаване на 'ASCII' код на натиснат клавиш
*
* входни данни : При наличие на строб от клавиатурният
*                мултиплексор за такива се считат клетките
*                'ред' и 'колона'.
*
* изходни данни : 'ASCII' код с вдигнат 7ми бит в клетка
*                 'KEYOUT'.
*
* ползвани клетки от EQU-зоната : KBD (KEYOUT)
*
* начин на работа : Хардуерно клавиатурата е организирана
*                   като матрица от 8 реда и 8 колони.
*                   колоните са свързани към съществувашия в
*                   'SAUND' процесора леч, а редовете към
*                   мултиплексор, който се адресира от 'B'
*                   порта на 'VIA'.
*                   При подаване на активна нула към някой от
*                   битовете на леча и последващо адресиране
*                   на мултиплексора то ако на пресечната
*                   точка има натиснат клави, мултиплексора ше
*                   изработи строб, който може да бъде прочетен
*                   от 3ти бит на 'B' портата на 'VIA'.
*
*****************************************************************


SEEKEY          LDA   ред         ;Необходимо ли е
                CMP   #7          ;адресиране на колона ?
                BNE   саморед     ;не, ще се движим по реда.

*****************************************************************
*                                                               *
*           Стандартно парче от програма за зареждане на        *
*           регистър на 'SAUND' процесор чрез                   *
*           в/и регистър на 'VIA', взета от компютър            *
*           'ORIC'.                                             *
*           входни данни :                                      *
*                        'A' = # на рег. на SAUND проц.         *
```

```
*                        'X' = стойност на записвания байт.*
*                                                          *

                LDA   #$E             ;LATCH на SAUND процесор
                LDX   колона
                STA   DRANH
                LDY   #$EE
                STY   PCR
                LDY   #$DC
                STY   PCR
                STX   DRANH
                LDY   #$EC
                STY   PCR
                LDY   #$DC
                STY   PCR

*                                                          *
*                                                          *
************************************************************

*   сканниране на осем бутона
*
                LDA   ред
саморед         ORA   #$10            ;4ти бит е строб за принтер.
                                      ;той не трябва да пада в '0'
                                      ;докато не се работи с
                                      ; принтера.
                STA   DRB             ;адресиране на мултипл.
                NOP                   ;изчакване на
                NOP                   ;хардуера
                LDA   DRB
                AND   #%00001000      ;Натиснат ли е клавиш ?
                BEQ   NOGETITO        ;не
* натиснат е клавиш                    *
                LDX   #0              ;Изчисляване на позицията
                LDA   колона          ;на активната нула в
MULT            LSR   A               ;'колона'
                BCC   MULTCONT
                INX
                BNE   MULT
MULTCONT        LDA   CODETBL,X       ;'A' = 8 * 'X'
                CLC
                ADC   ред             ;'A' = 'A' + 'ред '
                STA   KEYWRK          ;Позиционен код на натиснатия
                                      ;клавиш.
                JMP   KEYDUMMY        ;Понататъшна обработка.

************************************************************
*
*           Подготовка за сканиране на следващ клавиш
*

NOGETITO        LDX   #$FF
                STX   KEYTIME         ;
                STX   KEYOLD          ;Няма задържан клавиш
                LDA   #<1000          ;Зареждане на
                STA   KEYTIMEL        ;време за първоначално
                LDA   #>1000          ;изчакване при натискане
```

```
                STA   KEYTIMEH    ;на клавиш.
                DEC   ред         ;Подготовка на
                BPL   NOGET       ;следваща точка
NOGETIT         LDA   #7          ;за сканиране
                STA   ред         ;на клавиатурата.
                SEC
                ROL   колона
                BCS   NOGET
                ROL   колона
NOGET           LDA   колона      ;Натиснат ли е
                CMP   #$EF        ;'SHIFT' или 'CNTRL'
                BNE   NOGET1      ;не
                LDA   #0
                STA   KEYSHIFT

* проверка за край на 1 сек. да се оптимизира

NOGET1          LDA   TIMEL
                ORA   TIMEH
                BNE   RTI1
                JSR   SETTIME

RTI1            LDA   OUTMORS     ;Извеждане на приготвеният
                                  ;байт към леч на радиозала.
                                  ;Използува се регистъра
                                  ;'DRANH' за да се запази
                                  ;състоянието на строва за
                                  ;принтер.
                STA   DRANH
                LDA   DRB         ;Строб за
                ORA   #$30
                STA   DRB         ;леч за радиозала

*****************************************************************
*           Работа с принтер
*
*           Или III част от програмата.
*****************************************************************
*
* задача :  Извеждане на определено количество байтове
*           от област на паметта наречена 'Принтерски
*           буфер'
*
* входни данни : Запълнен буфер. Първият байт от него
*                трябва да е <> от 0 и след последния
*                символ трябва да има 0
*
* изходни данни : Движение на главата на принтера
*                 съпроводено със специфичен звук.
*
* ползвани клетки от EQU-зоната : BUFPRN начало на буфера
*
* начин на работа : При наличие на символ <> от 0 в
*                   началото на буфера започва постепенното
*                   извеждане на съдържанието му към
*                   принтера. За всеки изпратен байт се
*                   установава флаг че е неоходимо да се
*                   провери дали принтера е готов преди да
```

```
*                       се изпрати следващия байт.
*                       Процесът продължава докато в буфера се
*                       срещне 0.
* външни процедури : няма
*
*****************************************************************

*
                LDA   BUFPRN      ;'Пълен' ли е буфера
                BEQ   NOPRINT1    ;не, зареди начални указатели
                LDA   PRFLAG      ;Отпечатван ли е символ ?
                BEQ   SAYPRN      ;не, сега трябва да се
                                  ;отпечата.
                LDA   IFR         ;да, отпечатван е и трябва да
                                  ;проверим дали принтера е
                                  ;готов.
                AND   #$02        ;'READY' ?
                BEQ   NOPRINT2    ;все още не
                LDA   #0
                STA   PRFLAG      ;Не е отпечатван символ
                STA   DRA         ;Изчистване на строба
                BEQ   NOPRINT2    ;И изход.
SAYPRN          LDY   #0          ;Четене на пореден байт
                LDA   (UKLP),Y    ;от принтерския буфер.
                BEQ   NOPRINT1    ;Ако е последен не го
                                  ;отпечатвай и ...
                STA   DRA         ;Изход към принтер
                STA   PRFLAG      ;и индикация че е отпечатван
                                  ;символ
                LDA   DRB         ;Строе за принтера
                AND   #%11101111  ;Бит #4 се сваля в '0'
                STA   DRB
                ORA   #$30        ;и веднага след това се
                STA   DRB         ;вдига в '1'.
                INC   UKLP        ;Адресиране на следващия
                BNE   NOPRINT2    ;символ от принтерския
                INC   UKHP        ;буфер
                BNE   NOPRINT2    ;и изход
NOPRINT1        LDA   #BUFPRN+1   ;Установяване на начален
                STA   UKLP        ;адрес за принтерския буфер
                LDA   #>BUFPRN+1  ;Извършва се при всеки край
                STA   UKHP        ;на текста и в инициализацията
                LDA   #0          ;на програмата.
                STA   BUFPRN
NOPRINT2        LDA   SVEA        ;Вистановяване на запомнените
                LDY   SVEY        ;в началото на прекъсването
                LDX   SVEX        ;регистри
                BIT   T1CL        ;Изчистване на флага за
                                  ;прекъсване на таймера на 'VIA'

                CLI               ;Разрешаване на ново
                                  ;прекъсване
                RTI               ;Възврат от прекъсване.


* Таблица за умножение по 8

CODETBL         DFB   0,8,16,24,32,40,48,56
```

```
* Таблица на отместванията на системните телеграфни
* буфери в главната таблица

BUFTBL      DFB  0,$10,$20,$30

*****************************************************************
*
* процедура  INCSTBL
* ------------------
*
* задача :       Извличане на пореден морзов еквивалент
*                за предаване
*
* входни данни : Клетки 'FLAGINC1' указват дали се извлича
*                символ от буфер или от собствената таблица
*                съдържаща '%%%='.
*
* изходни данни : Клетка SVEBIT1, която съдържа символа за
*                 предаване.
*                 Увеличен с 1 брояч на предадени символи
*                 намиращ се в системния тел. буфер.
*
* ползвани клетки от EQU-зоната : няма
*
* начин на работа : Виж коментарите по-долу
*
* външни процедури : няма
*
*****************************************************************


INCSTBL     LDA  FLAGINC1,X ;Фалшива група ли се предава ?
            BNE  INCSLOP    ;да отиди на специална
                            ;обработка.
            LDY  #TRADRL    ;Увеличаване с 1 на указателя
            CLC
            LDA  (UKL),Y
            ADC  #1         ;за брой предадени байтове
            STA  (UKL),Y
            INY
            LDA  (UKL),Y    ;в системния телеграфен
            ADC  #0
            STA  (UKL),Y    ;буфер.
            INC  UKL1,X     ;Увеличаване с 1 на указателя
            BNE  INCSLOP0   ;на пореден морзов код в
            INC  UKH1,X     ;системния телеграфен буфер.
INCSLOP0    LDA  UKL1,X     ;Приравняване към временен
            STA  UL         ;указател, ползуван от всички
            LDA  UKH1,X     ;направления.
            STA  UH
            LDY  #0         ;Преместване на поредния
            LDA  (UL),Y     ;морзов еквивалет в клетка
            STA  SVEBYT1,X  ;от която ще бъде предаван.
                            ;това е необходимо тъй като
                            ;съществува изискване да не се
                            ;променя началния буфер по
                            ;време на предаване а за да се
```

```
                                    ;извлекат елементите 1 по 1 е
                                    ;необходимо да се върти
                                    ;клетката на дясно.
                RTS

INCSLOP         BMI   ENDINC        ;Ако 'FLAGINC' е $FF не
                                    ;увеличавай адреса.
                TAY                 ;'Y' = на пореден символ от
                                    ;'ЖЖЖ='
                INC   FLAGINC1,X    ;Подготовка за следващ символ
                LDA   TPFE,X
                ORA   #%00000100    ;заврана за край на текст
                STA   TPFE,X
                LDA   STARTBL,Y     ;
                PHP                 ;Запазване на статуса
                                    ;Но защо ?
                LDY   #0            ;В цялата останала част от
                                    ;програмата се предполага че
                                    ;'Y' е 0
                PLP
                STA   SVEBYT1,X     ;Виж коментара по горе.
                BNE   ENDINC        ;Последен символ от 'ЖЖЖ='
                STA   FLAGINC1,X    ;да, трябва да установим
                LDA   TPFE,X        ;състояние за нормална
                AND   #%11111011    ;работа на програмата
                STA   TPFE,X        ;И да вземен първия символ
                JSR   INCSLOPO      ;от истинския буфер.
                LDA   #0            ;индикация за край на група
                                    ;необходима е за да има пауза
ENDINC          RTS

KEYDUMMY        LDA   колона
                CMP   #$EF          ;'MC','SF' ?
                BEQ   SETMOVE       ;да, пресметни отместването в
                                    ;в таблицата с 'ASCII' кодове.
                LDA   KEYWRK        ;Същия клавиш ли е
                CMP   KEYOLD        ;натиснат ?
                BEQ   CHEKTIME      ;да, провери дали е изтекло
                                    ;времето за първоначално
                                    ;изчакване
                STA   KEYOLD        ;не, установи го като стар.
                LDA   #<1000        ;зареди ново
                STA   KEYTIMEL      ;време за първоначално изчакван
е
                LDA   #>1000
                STA   KEYTIMEH
TIMENO          LDA   #180          ;Зареди време за изчакване
                STA   KEYTIME       ;при 'REPEAT'
                LDX   KEYSHIFT      ;Изчисляване на
                LDA   MOVETBL,X     ;необходимия индекс
                CLC                 ;за извличане на
                ADC   KEYWRK        ;необходимия
                TAY                 ;'ASCII' код от
                LDA   ASCII,Y       ;таблицата
                STA   KEYOUT        ;и поставянето му в изходния
                                    ;байт
                JMP   NOGET         ;край на подпрограмата

CHEKTIME        LDA   KEYTIMEL      ;Времето за първоначално
```

```
                ORA   KEYTIMEH    ;изчакване нула ли е ?
                BEQ   CHEKTIMEL   ;да, провериа времето за изчакв
ане
                                  ;при 'REPEAT'
                LDA   KEYTIMEL    ;не, тогава
                BNE   CHEKTL1     ;ще го намалим с 1
                DEC   KEYTIMEH
CHEKTL1         DEC   KEYTIMEL
                JMP   NOGET       ;и ще излезем без да
                                  ;генерираме 'ASCII' код.
CHEKTIMEL       LDA   KEYTIME     ;Времето за изчакване при
                                  ;'REPEAT' свършило ли е ?
                BPL   TIMENO      ;да, генерирай 'ASCII' код
                                  ;и зареди ново време за
                                  ;'REPEAT'
                DEC   KEYTIME     ;не, намали го с 1
                JMP   NOGET       ;и излез без да генерираш
                                  ;'ASCII' код
SETMOVE         LDA   ред         ;Установяване на отместване
                STA   KEYSHIFT    ;в 'ASCII' таблицата при
                JMP   NOGETIT     ;натиснат клавиш 'CNTRL'
                                  ;или 'SHIFT'и отместване от
                                  ;реда на тези клавиши за да се
                                  ;установи кой друг е натиснат
                                  ;заедно с него.

*****************************************************************
*
*       Таблица на отместванията в 'ASCII' таблицата при
*       натиснат клавиш 'CNTRL' или ляв или десен 'SHIFT'.
*
*             Използува се факта че хардуерно тези клавиши
*             се намират на един ред и следователно при
*             индикация на натиснат клавиш променливата 'ред',
*             която има стойности от 0 до 7, сочи кой от тях е
*             натиснат.
*

MOVETBL         DFB   0,0,64,0,128,0,0,128

*****************************************************************
*       Таблица на генерираните 'ASCII' кодове
*
*             Използува се за генериране на 'ASCII' код от
*             известен позиционен код на натиснат клавиш.
*             Всички комбинации от клавиши за които няма
*             еквивалент в стандартната 'ASCII' таблица
*             генерират код $A0 т.е. интервал.
*

*   Латиница
ASCII           HEX   B7CACDCBA0D5D9B8
                HEX   CED4B6B9ACC9C8CC
                HEX   B5D2C2DBAECFC7B0
                HEX   D6C6B4BA8BD0C5AF
                HEX   A0A0A0A0A0A0A0A0
                HEX   B19BDAA088FFC18D
                HEX   D8D1B2BB8ADCD3DE
                HEX   B3C4C3DD89C0D7AD
```

```
*   Контроли
                HEX   B78A8D8BA09599B8
                HEX   8E94B6B9AC89888C
                HEX   B592829BAE8F87B0
                HEX   9686B4BAA09085AF
                HEX   A0A0A0A0A0A0A0A0
                HEX   B1A09AA0A09F818D
                HEX   9891B2BBA09C939E
                HEX   B384839DA08097AD
*   Кирилица
                HEX   A7EAEDEBA0F5F9A8
                HEX   EEF4A6A9BCE9E8EC
                HEX   A5F2E2FBBEEFE7B0
                HEX   F6E6A4AAA0F0E5BF
                HEX   A0A0A0A0A0A0A0A0
                HEX   A1A0FAA0A0FFE18D
                HEX   F8F1A2ABA0FCF3FE
                HEX   A3E4E3FDA0E0F7BD

********** Край на файл TIME
```

```
* файл MSGS
* последната корекция е била на
* 08-юли-1987
* дефиниране на символни и други константи

T1          DCI   " "
T2          DCI   "ОРГАНИЗАЦИЯ ЗА СЪДЕЙСТВИЕ НА ОТБРАНАТА "
T3          DCI   "ЦЕНТРАЛЕН СЪВЕТ "
T4          DCI   "БАЗА ПО РАДИОЕЛЕКТРОНИКА "
T5          DCI   "ИНТЕРВАЛ-СЛЕДВАЩ ЕКРАН"
T6          DCI   "СИСТЕМА ЗА ОБУЧЕНИЕ И ТРЕНИРОВКА "
T7          DCI   "НА РАДИОТЕЛЕГРАФИСТИ "
T8          DCI   "ПО 'МЕТОДИЧЕСКО РЪКОВОДСТВО "
T9          DCI   "ЗА ОБУЧЕНИЕ В БНА' "
T10         DCI   "ВЕРСИЯ 1.1                    10.X.1986"
T11         DCI   "1. ОКРЪЖНО"
T12         DCI   "2. РАБОТА В НАПРАВЛЕНИЕ"
T13         DCI   "-------------------------------------"
T14         DCI   "ИЗБЕРЕТЕ:"
T15         DCI   "1. ИЗБОР НА НАПРАВЛЕНИЕ"
T16         DCI   "1-4. ИЗБОР НА НАПРАВЛЕНИЕ"
T17         DCI   "5. ЗАПОЧВАНЕ НА ПРЕДАВАНЕТО"
T18         DCI   "ВЪВ ВСИЧКИ ГОТОВИ НАПРАВЛЕНИЯ"
T19         DCI   "6. ПАУЗА ЗА ВСИЧКИ НАПРАВЛЕНИЯ"
T20         DCI   "7. ПРОДЪЛЖЕНИЕ НА ПРЕДАВАНЕТО"
T21         DCI   "8. КРАЙ НА ПРЕДАВАНЕТО ВЪВ"
T22         DCI   "ВСИЧКИ НАПРАВЛЕНИЯ"
T23         DCI   "9. ВРЪЩАНЕ В ГЛАВНОТО МЕНЮ"
T24         DCI   " С НОМЕР "
T25         DCI   "1. ПРЕДАВАНЕ НА ТЕКСТА"
T26         DCI   "1. ПРЕКРАТЯВАНЕ НА ПРЕДАВАНЕТО"
T27         DCI   "2. ПОДГОТОВКА НА НОВ ТЕКСТ ОТ СЪЩИЯ ВИД"
T28         DCI   "3. ПРОМЯНА НА СКОРОСТТА НА ПРЕДАВАНЕ"
T29         DCI   "4. ВРЪЩАНЕ В МЕНЮТО НА НАПРАВЛЕНИЯТА"
T30         DCI   "5. НОВО КОНФИГУРИРАНЕ НА НАПРАВЛЕНИЕТО"
T31         DCI   "6. ОТПЕЧАТВАНЕ НА ТЕКСТА"
T32         DCI   "7. ИЗБОР НА СТАР ТЕКСТ"
T33         DCI   "8. ИЗТРИВАНЕ НА СЛЕДВАЩИЯ ТЕКСТ"
T34         DCI   "9. НАБЛЮДАВАНЕ НА ПРОЦЕСА НА ПРЕДАВАНЕ"
T35         DCI   " «ОСВ» - ОТКАЗ"
T36         DCI   "НАПРАВЛЕНИЕ:"
T37         DCI   "ТРЕНИРОВЪЧЕН ТЕКСТ"
T38         DCI   "ПРОФИЛ. ТЕКСТ"
T39         DCI   "ДАТЧИК"
T40         DCI   "УПРАЖНЕНИЯ"
T41         DCI   "РАБОТИ С ПЪЛЕН БУФЕР"
T42         DCI   "БРОЙ ГРУПИ ? (10,20,30,50,100):"
T43         DCI   "КИРИЛИЦА"
T44         DCI   "ЛАТИНИЦА"
T45         DCI   "ЦИФРИ"
T46         DCI   "СМЕСЕН"
T47         DCI   "СИЛНО СМЕСЕН"
T48         DCI   "ВИД ТЕКСТ ?:"
T49         DCI   "СКОРОСТ НА ЗНАЦИТЕ ? (ЗН/МИН):"
T50         DCI   "СКОРОСТ НА ТЕКСТА ? (ГР/МИН):"
T51         DCI   "С ПРЕОБЛАДАВАНЕ НА "
T52         DCI   "КОНФИГУРИРАНЕ"
T53         DCI   "НЕКОНФИГУРИРАНО"
```

```
T54         DCI  "ГОТОВО ЗА ПРЕДАВАНЕ"
T55         DCI  "ПАУЗА"
T56         DCI  "ПАУЗА М/У ТЕКСТОВЕ"
T57         DCI  "НЯМА ТЕКСТ ЗА ПРЕДАВ."
T58         DCI  "РАБОТИ С ПРАЗЕН БУФ."
T59         DCI  "1. ИЗУЧАВАНЕ НА:ФОЬП"
T60         DCI  "2. ИЗУЧАВАНЕ НА:ДЛЖК"
T61         DCI  "3. ИЗУЧАВАНЕ НА:ВАРМ"
T62         DCI  "4. ЗАТВЪРДЯВАНЕ. КОНТРОЛНА РАБОТА 1"
T63         DCI  "5. ИЗУЧАВАНЕ НА:ЩНГЪ"
T64         DCI  "6. ИЗУЧАВАНЕ НА:БУСЖ"
T65         DCI  "7. ЗАТВЪРДЯВАНЕ. ИЗУЧАВАНЕ НА:12345"
T66         DCI  "8. ЗАТВЪРДЯВАНЕ. ИЗУЧАВАНЕ НА:67890"
T67         DCI  "9. ЗАТВЪРДЯВАНЕ. КОНТРОЛНА РАБОТА 2"
T68         DCI  "10. ЗАТВЪРДЯВАНЕ. ИЗУЧАВАНЕ НА:ЦЧЮЙ"
T69         DCI  "11. ЗАТВЪРДЯВАНЕ. ИЗУЧАВАНЕ НА:ЯЗХИ"
T70         DCI  "12. ЗАТВЪРДЯВАНЕ. ИЗУЧАВАНЕ НА:ЕТ"
T71         DCI  "13. ЗАТВЪРДЯВАНЕ. КОНТРОЛНА РАБОТА 3"
T72         DCI  "14. ЗАТВЪРДЯВАНЕ. ИЗУЧАВАНЕ НА:?.,"
T73         DCI  "15. УВЕЛИЧАВАНЕ СКОРОСТТА ДО 5ГР/МИН"
T74         DCI  "16. УВЕЛИЧАВАНЕ СКОРОСТТА ДО 6ГР/МИН"
T75         DCI  "17. ОБОБЩАВАЩА КОНТРОЛНА РАБОТА"
T76         DCI  "ТОВА Е ТЕКСТ"
T77         DCI  "ЩЕ БЪДАТ ИЗТРИТИ ВСИЧКИ ТЕКСТОВЕ"
T78         DCI  "ПОТВЪРДЕТЕ (Д/Н) "
T79         DCI  "ПРИНТЕРА Е  ЗАЕТ !!!"
T80         DCI  "<RETURN> - ОТПЕЧАТВАНЕ"
T81         DCI  "ВЪВЕДЕТЕ ТЕКСТ: "
T82         DCI  "ТЕКСТА САМО ЗА ДАДЕНИТЕ СИМВОЛИ"
T83         DCI  "ДА СЕ ГЕНЕРИРА ?(Д/Н): "
T84         DCI  "2. ИЗБОР НА ТЕКСТ ОТ УПРАЖНЕНИЕ"
T85         DCI  "ПЕТОРНО ПОВТОРЕНИЕ НА ЗНАЦИТЕ"
T86         DCI  "СДВОЕНИ ЗНАЦИ       "
T87         DCI  "ДУБЛИРАНЕ НА ЦЯЛА ГРУПА"
T88         DCI  "ПОВТОРЕНИЕ НА ГРУПИТЕ"
T89         DCI  "ОБИКНОВЕН ТЕКСТ"
T90         DCI  "КОНТРОЛЕН ТЕКСТ"
T91         DCI  "ЗАПОЗНАВАНЕ СЪС ЗВУЧ. НА ЗНАЦИТЕ"
T92         DCI  "ЦИФРОВ ТЕКСТ"
T93         DCI  "ДЬЛЖИНА НА ТЕКСТА:"
T94         DCI  "СКОРОСТ:"
T95         DCI  "ГРУПИ"
T96         DCI  "ЗН/МИН."
T97         DCI  "ГР/МИН."
T98         DCI  "ОТ КАСЕТОФОН"
T99         DCI  "САМО ОТ ЗАДАДЕНИТЕ СИМВОЛИ"
T100        DCI  "СМЕСЕН ТЕКСТ"
T101        DCI  "ДУБЛИРАНЕ НА ЗНАЦИТТЕ ОТ ПЪРВА ГРУПА"

**********************************
*
* ДАННИ ЗА УПРАЖНЕНИЯТА
*
**********************************

U1:         DFB  #0
            DFB  %11111010
            DFB  %00010100
```

```
                DFB   #0
U2:             DFB   %10001100
                DFB   %11111010
                DFB   %11101100
                DFB   #0
U3:             DFB   %00100000
                DFB   %11110010
                DFB   %01100110
                DFB   %00010100
U4:             DFB   %11111100
                DFB   #0
                DFB   %10010100
                DFB   %00000111
U5:             DFB   %00100000
                DFB   %11110010
                DFB   %10010100
                DFB   %00000100
U6:             DFB   %00100000
                DFB   %11110010
                DFB   %10101100
                DFB   %00000100
U7:             DFB   %11011100
                DFB   %11010000
                DFB   %10011100
                DFB   %01100100
U8:             DFB   %11011100
                DFB   %11010000
                DFB   %10010100
                DFB   %01100000
U9:             DFB   %11011100
                DFB   #0
                DFB   %10010100
                DFB   %00000111
U10:            DFB   %00000100
                DFB   %11111110
                DFB   %10100100
                DFB   %01100000
U11:            DFB   %01100110
                DFB   %11011010
                DFB   %10110100
                DFB   %01100000
U12:            DFB   %01100110
                DFB   %11010010
                DFB   %01110100
                DFB   %01000000
U13:            DFB   %10101100
                DFB   %0
                DFB   %11100100
                DFB   %01000111
U14:            DFB   %10110100
                DFB   %11000000
                DFB   %10110111
                DFB   %10010000
U15:            DFB   %10100100
                DFB   %0
                DFB   %10101110
                DFB   %01100000
U16:            DFB   %10100100
```

```
                DFB   %0
                DFB   %10100110
                DFB   %01000100
UI7:            DFB   %00000100
                DFB   %0
                DFB   %00000110
                DFB   %01100110

ET1:            DCI   "ФЬЭП"
ET2:            DCI   "ДЛЖК"
ET3:            DCI   "ВЯРМ"
ET4:            DCI   "ЩНГЪ"
ET5:            DCI   "БУСШ"
ET6:            DCI   "12345"
ET7:            DCI   "67890"
ET8:            DCI   "ЦЧЮИ"
ET9:            DCI   "ЯЗХИ"
ET10:           DCI   "ЁТ"
ET11:           DCI   "?.,@"

********** край на MSGS
```

```
* файл EQUS
* последната корекция е била на
* 17-юли-1987

* клетки от нулевата страница, които се ползват от
* библиотеката на компилатора
            DUM   $A0
CURSX       DS    1             ;позиция на курсора
CURSY       DS    1
BUFPOS      DS    1             ;указател за отместването от
                                ;началото на BUFF
ADR         DS    2             ;резултат от GETADR
CSAVE       DS    1             ;за запазване на позицията на
                                ;курсора
ASAVE       DS    1             ;за запазване на регистрите на
YSAVE       DS    1             ;6502
WRK1        DS    1             ;
WRK2        DS    1             ;работни с общо предназначе-
TEMP        DS    2             ;ние. ползват се като локални
TEMP1       DS    2             ;
ACC         DS    1             ;общоприети за запазване на A
YREG        DS    1             ;и Y при влизане в процедура

* клетки от нулевата страница, които се ползват от
* менютата
WDIR        DS    1             ;направление, с което се рабо-
                                ;ти в момента
WRK3        DS    1             ;работни
WRK         DS    2             ;
BAS         DS    2             ;адрес, изчисляван от BASCALC
YOUT        DS    1             ; ?
UKLP        DS    1             ;Ползуват се от 'GETKEY' за
UKHP        DS    1             ;адрес на текуш системен
                                ;'ASCII' буфер.
UKLT        DS    1             ;Същото но за системен
UKHT        DS    1             ;телеграфен буфер
ZABR        DS    1             ;флаг за regraw на екрана
                                ;Ползуват се от 'GETKEY' за
MISTATT     DS    1             ;статус на сист. тел. буфер
MISTATA     DS    1             ;и системен 'ASCII' буфер
SCREEN      DS    2             ;адрес на екрана за regraw
UKLP1       DS    1             ;за потребителски
UKHP1       DS    1             ;'ASCII' буфер
            DEND

* клетки от нулевата страница, които се ползват от
* менютата (продължение)
            DUM   $98
ADUP        DS    2             ;адрес на упражнение
ADP1        DS    2             ;работна за упражнения
            DEND

* preiswoowane na lokalni promen-
* livi na CONV1 modula

            DUM   $FA0
D1          DS    1             ;в тези клетки се получава
D2          DS    1             ;десетичното число при конвер-
```

```
D3          DS   1              ;тирането му от HEX
H1          DS   1              ;при обратното конвертиране
H0          DS   1              ;(DEC->HEX) резултатът е тук
ASAVE1      DS   1              ;работни за GETKEY
CURS+1      DS   1              ;
MyY         DS   1              ;
FLAGSHOW    DS   1              ;
MILPOS      DS   1              ;За 'GETKEY',позиция до която
MIHPOS      DS   1              ;е визуализиран текста в/ч
                                ;екрана в режим 'шоу'
ASTAT       DS   1              ;-за запазване на 6502 реги-
XSTAT       DS   1              ;!стрите и позицията на кур-
YSTAT       DS   1              ;!сора в STAT
CSTAT       DS   1              ;!
NRTEXT      DS   1              ;работна за TOCAT
BROJKA      DS   1              ;
            DEND

            DUM  $FC0
GRAC        DS   1              ;
STAT        DS   1              ;
NEWST       DS   1              ;
NAPR        DS   1              ;
SPEEDU      DS   1              ;скорост в упражнение
NORED       DS   1              ;
UPRAV       DS   1              ;упражнение номер
tuprav      DS   1              ;точка от упражнение
KLETKA      DS   1              ;
TYPSYMBXT                       ;тип на текста (цифри,букви..)
KLETK       DS   1              ;
NEXTOB                          ;указател за вида на следваща-
                                ;та обработка
KLET        DS   1              ;
pregovor                        ;при преговор се вземат и пре-
                                ;дишните символи от таблицата
KLB         DS   1              ;
TEMPUPR     DS   1              ;работна за упражнения
VIZIGROUPS  DS   1              ;брой на групи на ред. Заема
                                ;две стойности - 5 и 2
CNTR1       DS   1              ;
CNTR2       DS   1              ;
            DEND

LOLIM       EQU  YSAVE          ;
HILIM       EQU  YREG           ;

BUFER       EQU  $D00           ;буфер за стрингов вход
BUFLEN      EQU  40             ;дължина на ↑ буфер
TTAB        EQU  $E00 TAB.TAB   ;таблица с адресите на буф.
ANSERO      EQU  $E40           ;начало на таблица с отговори

* COMMON област за глобални променливи. Тук се дефинират
* общите променливи, които ще използват и от другите
* три програми - TIME, FORM и ARITM.
BUF1        EQU  $C00           ;входно множество за FORM
BUFPRN      EQU  $A000          ;буфер за принтер
KBD         EQU  $2DF           ;тук се записва въведеният от
                                ;клавиатурата символ
```

```
RNDL        EQU   $4E          ;клетки, в които се пази слу-
RNDH        EQU   RNDL+1       ;чайното число. Следващото чис
                               ;ло се генерира от него.
OUTMORS     EQU   $74          ;в тази клетка се пази състоя-
                               ;нието на LATCH-а за радиоза-
                               ;дата
LUSTIME     EQU   $0C          ;константа за TIMER

* символни константи
SP          EQU   $20          ;SPACE
SS          EQU   $08          ;BACK-SPACE
CR          EQU   $0D          ;CARRIGE-RETURN
ESC         EQU   $1B          ;ESCAPE

* Други константи
BOTTOM_OF_PG EQU 6             ;Ограничения за
TOP_OF_PG   EQU   21           ;екрана

********** Край на EQUS
```

```
* файл CAT.
* последната корекция е била на
* 17-юли-1987
* процедури за записване и четене на данни от каталога

*   използвани съкращения:
*           ПАБ - потребителски ASCII буфер
*           ПТБ - потребителски телграфен буфер

**********************************************************
*
* процедура  TO_CAT
* -----------------
*
* задача :
*   Създаване запис в буфера на каталога за последния
*   генериран текст в съответното направление. Всеки запис
*   има номер. При опит за създаване на запис, с номер,
*   който вече е използван, не се създава нов запис.
*
* входни данни :
*   Параметрите на последния генериран текст в съответното
*   направление, а именно: вид и номер на текста, начална
*   стойност на RND генератора, символи за преобладаване,
*   скорост на предаване зн/мин и гр/мин, брой групи, тип
*   на текста.
*   Клетката WDIR трябва да съдържа номера на
*   обработваното направление.
*
* изходни данни :
*   Запис за последния генериран текст в каталога на
*   съответното направление.
*
* ползвани клетки от EQU-зоната :
*   WDIR       - съдържа номера на направлението, по което
*                се работи.
*   WPR,ADR1   - две двойки клетки, които се използват за
*                указатели при неявна адресация - (...),Y.
*
* външни процедури:
*
*   GETADR: - изчислява адреса на обработвания буфер.
*
**********************************************************

TO_CAT
            LDX   #3          ;Изчислява се адреса на ПАБ
            JSR   GETADR:     ;от обработваното направл.
            LDY   #3          ;Номера на текста
            LDA   (ADR),Y     ;се поставя
            STA   NUM         ;в клетка NUM.
            JSR   SEARCH      ;Установява се мястото за
                              ;записа в каталога.
            BCC   *+3         ;Ако вече има текст с такъв
            RTS               ;номер не се създава нов.

            LDY   #2          ;В A и X регисстри се
            LDA   (ADR),Y     ;зареждат вида...
```

```
                TAX
                INY
                LDA   (ADR),Y     ;...и номера на текста,
                LDY   #1          ;след това се поставят
                STA   (ADR1),Y    ;на първа (номера)...
                TAX
                DEY               ;...и на нулева (вида)
                STA   (ADR1),Y    ;позиция в каталожния запис.

                INY               ;Броя на групите се взема от
                LDA   (ADR),Y     ;таблицата на  ПАБ
                LDY   #$10        ;и се поставя на 16-та ($10)
                STA   (ADR1),Y    ;позиция в каталожния запис.
                LDY   #$E         ;От таблицата на ПАБ в
                LDA   (ADR),Y     ;каталожния запис се
                LDY   #$11        ;прехвърля
                STA   (ADR1),Y    ;типа на текста.

                LDY   #$D
ЗС
                LDA   (ADR),Y     ;От таблицата на ПАБ
                STA   (ADR1),Y    ;в каталожния запис
                DEY               ;се копират
                CPY   #4          ;символите
                BCS   ЗС          ;за преобладаване.

                LDX   #2          ;В клетките ADR и ADR+1 се
                JSR   GETADR:     ;поставя адреса на ПТБ.
                LDY   #$E         ;От таблицата на ПТБ в
                LDA   (ADR),Y     ;каталожния запис се копират
                STA   (ADR1),Y    ;скоростите на текста зн/мин...
                INY
                LDA   (ADR),Y
                STA   (ADR1),Y    ;...и гр/мин.

                LDY   #2          ;В каталожния запис
                LDA   RNDL        ;се копират младшата...
                STA   (ADR1),Y
                INY               ;...и старшата част на
                LDA   RNDH        ;началната стоност на
                STA   (ADR1),Y    ;RND генератора.

                LDA   #0          ;Нулира се първият байт от
                LDY   #$18        ;мястото за следващ запис. Така
                STA   (ADR1),Y    ;се означава края на каталога.
                RTS               ;Край на програмата.

*****************************************************************
*
* процедура  FROM_CAT
* -----------------
*
* задача :
*   Да възстанови началните данни за генериране на текст
*   от каталога на съответното направление.
*
* входни данни :
*   Акумулатора съдържа номера на записа, който ще бъде
```

```
*   използван.
*
* изходни данни :
*   Таблиците на ПАБ и ПТБ за направлението, по което се
*   работи са подготвени за генериране на текст.
*
* ползвани клетки от EQU-зоната :
*   WDIR      - съдържа номера на направлението, по което
*               се работи.
*   ADR,ADR1  - две двойки клетки, които се използват за
*               указатели при неявна адресация - (...),Y.
*
* външни процедури:
*
*   GETADR: - изчислява адреса на обработвания буфер.
*
*****************************************************************

*
FROM_CAT
            STA  NUM         ;В клетка NUM се записва номера
                             ;на записа за обработка.
            JSR  SEARCH      ;Намира се началния му адрес в
                             ;каталога.
            LDY  #0          ;В A и X регистри се записват
            LDA  (ADR1),Y    ;вида...
            TAX
            INY
            LDA  (ADR1),Y    ;...и номера на текста.
            LDY  #3
            STA  (ADR),Y     ;След това се записват на
            TXA              ;втора (вида) и на трета
            DEY              ;(номера) позиция в
            STA  (ADR),Y     ;таблицата на ПАБ.

            LDY  #$10        ;От каталожния запис в
            LDA  (ADR1),Y    ;таблицата на потребителския
            LDY  #1          ;ASCII буфер
            STA  (ADR),Y     ;се записва броя на групите.
            LDY  #$11        ;От каталожния запис в
            LDA  (ADR1),Y    ;таблицата на потребителския
            LDY  #$E         ;ASCII буфер
            STA  (ADR),Y     ;се записва типа на текста.

            LDY  #$D
ЗС
            LDA  (ADR1),Y    ;От каталожния запис в
            STA  (ADR),Y     ;таблицата на потребителския
            DEY              ;ASCII буфер
            CPY  #4          ;се копират
            BCS  ЗС          ;символите за преобладаване.

            LDX  #2          ;В ADR и ADR+1 се поставя
            JSR  GETADR:     ;началния адрес на ПТБ.

            LDY  #$E         ;От каталожния запис в
            LDA  (ADR1),Y    ;таблицата на ПТБ
            STA  (ADR),Y     ;се записват скоростите
```

```
            INY             ;ЗН/МИН...
            LDA  (ADR1),Y
            STA  (ADR),Y    ;...И ГР/МИН.

            LDY  #2         ;От каталожния запис
            LDA  (ADR1),Y   ;се възстановяват
            STA  RNDL       ;млдшата...
            INY
            LDA  (ADR1),Y   ;...и старшата част
            STA  RNDH       ;на RND генератора.
            RTS             ;Край на програмата.

;*****************************************************************
;
; процедура  SEARSH
; -----------------
;
; задача :
;   Да открие в каталога на съответното направление запис,
;   чийто номер е записан вклетка NUM и да запише в клетки
;   ADR1 и ADR1+1 началния му адрес. Ако не съществува
;   запис с такъв номер, ADR1/ADR1+1 сочат първото
;   свободно поле за запис след края на каталога
;   и флаг C=0 (инче C=1).
;
; входни данни :
;   NUM - съдържа номера на търсения текст.
;
; изходни данни :
;   ADR1/ADR1+1 - съдържат началния адрес на търсения текст
;                 или на първото свободно поле след края на
;                 каталога.
;         флаг C - показва дали е на мерен търсения текст:
;                  C=1 - да, C=0 - не.
;
; ползвани клетки от EQU-зоната :
;   WDIR       - съдържа номера на направлението, по което
;                се работи.
;   ADR,ADR1   - две двойки клетки, които се използват за
;                указатели при неявна адресация - (...),Y.
;
;*****************************************************************

;
SEARCH
            LDX  #4         ;В ADR/ADR+1 се поставя
            JSR  GETADR:    ;началния адрес на каталога.
            LDA  ADR        ;След това
            LDX  ADR+1      ;ADR/ADR+1
            STA  ADR1       ;се копират
            STX  ADR1+1     ;в ADR1/ADR1+1.
            LDX  #3         ;В ADR/ADR+1 се поставя
            JSR  GETADR:    ;началния адрес на ПАБ.
JC
            LDY  #0
            LDA  (ADR1),Y   ;Ако поредния запис започва
            BEQ  NOMATCH    ;с 0, значи е достигнат края
                            ;на каталога, а търсения
```

```
                                ;запис не е открит.
            INY                 ;Ако номера на поредния запис
            LDA   (ADR1),Y      ;съвпада с номера на търсения
            CMP   NUM           ;запис то търсенето приключва
            BEQ   MATCH         ;успешно.
            JSR   INC_STEP      ;Съдържанието на ADR1/ADR1+1 се
                                ;променя така, че да сочи
                                ;следващия запис в каталога.
            JMP   ЗС            ;Проверките се извършват отново.
MATCH       SEC                 ;Изход на програмата
            RTS                 ;при флаг C=1.

NOMATCH     CLC                 ;Изход на програмата
            RTS                 ;при флаг C=0.

*****************************************************************
*
* процедура  INC_STEP
* -----------------
*
* задача :
*   Да промени съдържанието на ADR1/ADR1+1 така, че да
*   сочи следващия елемент от каталога.
*
* входни данни :
*   съдържанието на клетките ADR1 и ADR1+1, който се
*   разглеждат като 16-битов регистър.
*
* изходни данни :
*   съдържанието на входните клетки, плюс дължината на
*   един запис в каталога (24 или $18).
*
* ползвани клетки от EQU-зоната :
*   ADR1
*   ADR1+1
*
* начин на работа :
*   Към съдържанието на клетките ADR1 и ADR1+1 16-битово
*   се прибавя 24, т.е. (ADR1+1/ADR1)=(ADR1+1/ADR1)+$0018
*
*****************************************************************

*
INC_STEP
            CLC                 ;Нулиране на преноса.
            LDA   ADR1          ;Към съдържанието на ADR1 се
            ADC   #$18          ;прибавя 24 ($18).
            STA   ADR1          ;
            LDA   ADR1+1        ;Към съдържанието на ADR1+1 се
            ADC   #0            ;прибавя преноса от първото
            STA   ADR1+1        ;събиране (ако има пренос).
            RTS                 ;Край на програмата.

********** край на CAT
```

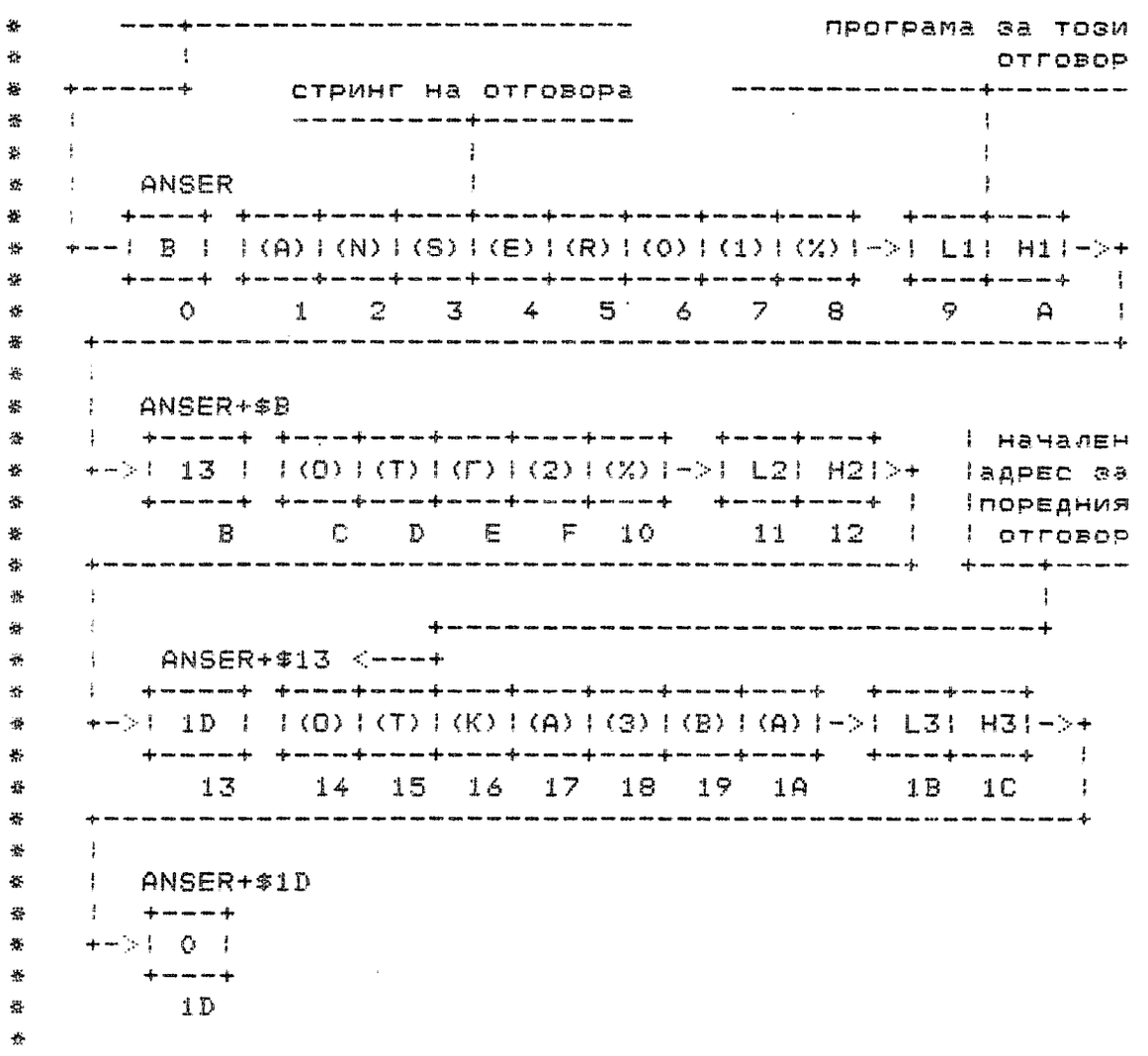
```
* файл LIB.
* последната корекция е била на
* 19-юли-1987
* библиотека на компилатора

*****************************************************************
*
* процедура SCRN
* --------------
*
* задача :
*   обявяване на начало на екран. С нея се нулира таблицата
*   с отговорите. виж ANS_ и WAIT_ . В клетките SCREEN
*   (2б.) се установява адресът, от който започва обявеният
*   екран (в действителност този адрес е с единица по-
*   малък, за да може да се използва преход към него чрез
*   следната примерна програма :
*           LDA SCREEN+1
*           PHA
*           LDA SCREEN
*           PHA
*           RTS
*
* входни данни :
*   явни входни данни не се използват. От стека се изважда
*   адресът, от който е била извикана процедурата и след
*   това се възстановява съсоянието му.
*
* изходни данни :
*   В клетките SCREEN (2 байта) се изчислява адресът, от
*   който започва обявеният с SCRN_ екран. Този адрес се
*   използва при redraw на текущия екран. Нулира се
*   таблицата с отговорите, като в първата й позиция се
*   записва код за край на тази таблица (нула).
*
* ползвани клетки от EQU-зоната :
*   ASAVE      - общопиета за запазване на 6502 акумулаторът
*   SREEN      - изходен резултат от работата на процедурата
*                (2 байта с подредба мл.-ст.)
*   ANSERS     - таблица с обявените отговори (с помощта на
*                процедурата ANS_). Тези отговори се скани-
*                рат когато е активна WAIT_ процедурата
* начин на работа :
*   извлича се възвратният адрес от стека, т.е. адресът,
*   от който е била 'извикана' процедурата SCRN_ плюс 3.
*   Този адрес сочи следващата изпълнима инструкция (в
*   следващия пример това е адреса BBBB).
*
*   AAAA:  JSR  SCRN_
*   BBBB:  LDA  ETIKET
*
*   за да получим стойността на адрес AAAA (счита се, че от
*   този адрес започва екрана, който се обявява с SCRN_)
*   трябва от адресът BBBB да се извади 3 (толкова байта е
*   дълга инструкцията JSR)
*
* външни процедури : няма
*
```

```
*****************************************************************

SCRN_       STA   ASAVE          ;запазва 6502-reg

            PLA                  ;извличане на възвратният ад-
            STA   SCREEN         ;рес от стека и записването му
            PLA                  ;в изходните клетки SCREEN
            STA   SCREEN+1
            PHA                  ;възстановяване на стека
            LDA   SCREEN
            PHA

            LDA   #0             ;нулиране на таблицата с от-
            STA   ANSERS         ;говори

* изчисляване на действителния адрес на екрана (с отчитане
* на 3-те байта за JSR). Това става като от получеият вече
* адрес в SCREEN се извади (16 битово) 3
            LDA   SCREEN         ;взима се младшата част
            SEC
            SBC   #3             ;изважда се броя на байтовете
                                 ;в инструкцията JSR.
            STA   SCREEN         ;записва се полученото

            LDA   SCREEN+1       ;взима се старшата част
            SBC   #0             ;отчита се преносът
            STA   SCREEN+1
            LDA   ASAVE          ;възстановява се акумулаторът
            RTS                  ;край на SCRN_

*****************************************************************
*
* процедура CLS
* -------------
*
* задача :
*   изчиства екрана от текущата позиция на курсора
*   до указаната като входна данна в A. След края на
*   процедурата CURSX=0, а CURSY е равен на указаната
*   позиция
*
* входни данни :
*   Акумолатора. В него трябва да има число между нула и
*   максималния брой на редовете, зависещ от типа на
*   компютъра (той се избира при всяка асемблация). Ако
*   числото е извън допустимите граници се приема
*   максимално възможната стойност.
*
* изходни данни :
*   Конкретни изходни данни от работата на процедурата
*   няма. Резултатът от нея е модификацията на екранната
*   област от RAM-а на компютъра.
*
* ползвани клетки от EQU-зоната :
*   ASAVE, YSAVE  - Общоприети клетки за запазване на 6502-
*                   регистрите
*   CYRSX, CURSY  - Указатели за позицията на курсора
*   MAXLINES      - Максимален брой на редовете за избра-
```

```
*                       ния компютър. Тази променлива се уста-
*                       новява за всяка асемблация или на 24
*                       или на 28 в зависимост от константата
*                       ORIC
*
* начин на работа :
*   Последователно са извиква процедурата CLREOL започвайки
*   от текущата позиция на курсора и завършвайки с указана-
*   та като входна данна Y-позиция. Текущия ред се изчиства
*   от моментната X-позиция, а всички следващи - от начало-
*   то на реда (от позиция 0).
*
* външни процедури :
*   CLREOL    - изчиства (запълва с интервали) реда, указан
*               от CURSY, започвайки от CURSX позицията и
*               завършвайки на физическия му край (40-та
*               позиция).
*
*****************************************************************

CLS_
            PHA                     ;запазва 6502-reg
            STY   YSAVE

* в A е входната данна. Проверява се дали е допустимите
* граници, т.е. дали е в рамките на екрана. Ако е извън
* тях се приема максимало допустимата стийност за избрани-
* ят при асемблирането екран
            CMP   #MAXLINES+1       ;проверява се A в рамките ли е
            BLT   CLS1              ;да, нищо не се прави
            LDA   #MAXLINES         ;ако е по-голямо, зарежда се с
                                    ;максимално допустимата стой-
                                    ;ност за това асемблиране

CLS1        STA   ASAVE             ;клетка съхранявяща Y коорди-
                                    ;натата на последния ред
CLS2        JSR   CLREOL            ;изтрива до края на реда
            LDA   #0
            STA   CURSX             ;нулира се CURSX след всяко
                                    ;изтриване с CLREOL
            INC   CURSY             ;увеличава се брояча на ре-
                                    ;довете
            LDA   CURSY
            CMP   ASAVE             ;достигнат ли е последният ред
            BLT   CLS2              ;не е, изтриването продължава
            BEQ   CLS2

            LDY   YSAVE             ;да, възстановява регистрите
            PLA
            RTS                     ;край на CLS_

*****************************************************************
*
* процедура PRINT
* ---------------
*
* задача :
*   извеждане на екрана, на текущата позиция на курсора,
```

```
*   стринга чийто е указан в A и Y. Стринга се дефинира с
*   DCI "....". Дължината на стринга не може да е по-голяма
*   от 255 байта.
*
* входни данни :
*   A и Y регистрите на 6502. В тях е записан адресът, от
*   който започва стрингът, подлежащ на извеждане върху
*   екрана (в A е записана младшата част, а в Y - стар-
*   шата). Стрингът е дефиниран с DCI "...", от което
*   следва, че всички негови елементи са с бит7 = 1, а
*   последният - с бит7 = 0. По този начин не се дефинира
*   входна данна дължина на стринга.
*
* изходни данни :
*   конкретни изходни данни от работата на процедурата не
*   се получават. Модифицира се екранната област, в която е
*   бил изведен стринга. След приключването на извеждането
*   CYRSX и CURSY сочат следващата позиция от екрана, на
*   която може да продължи извеждането.
*
* ползвани клетки от EQU-зоната :
*   TEMP          - работни клетки от нулевата страница с
*                   общо предназанчение. В тази процедура
*                   се използват за съхранение на адреса на
*                   стринга, който трябва да се извежда на
*                   екрана.
*   CYRSX, CURSY  - Указатели за позицията на курсора
*   MAXLINES      - Максимален брой на редовете за избра-
*                   ния компютър. Тази променлива се уста-
*                   новява за всяка асемблация или на 24
*                   или на 28 в зависимост от константата
*                   ORIC
*
* начин на работа :
*   от базовия адрес (установен като входна данна) започва
*   индексирано извличане (по Y-регистъра) на байтове. След
*   всяки извличане се проверява дали не е достигнат края
*   на стринга за извежане на екрана и ако не е, символът
*   се извежда. Ако е достигнат края на стринга се извежда
*   последния символ, реда, на който е станало извеждането,
*   се истрива до физическия му край и се установяват
*   новите стойности на позицията на курсора.
*
* външни процедури :
*   OUTCHR    - извежда върху екрана на позицията на цурсора
*               символът, чийто код се намира в акумулато-
*               рът. След извеждането CURSX позицията се
*               увеличава с единица, без да се правят про-
*               верки за достигнат край на ред
*   CLREOL    - изчиства (запълва с интервали) реда, указан
*               от CURSY, започвайки от CURSX позицията и
*               завършвайки на физическия му край (40-та
*               позиция).
* входни точки :
*   PRINTO    - обособена входна точка, за същинската проце-
*               дура PRINT, която се използва от PRINT_ и от
*               WAIT_ процедурите на компилаторът.
*
```

```
*****************************************************************

PRINT_
            JSR   PRINTO
            LDA   #0              ;X позиция на курсора в на-
            STA   CURSX           ;чалото на реда
            RTS

PRINTO
            STA   TEMP            ;адресът на стринга се записва
            STY   TEMP+1          ;в TEMP

            LDY   #0              ;начало на стринга

PRINT_1     LDA   (TEMP),Y        ;взима пореден елемент
            BPL   PRINT_2         ;последен ли е ? (на последни-
                                  ;ят бит7 = 0)
            JSR   OUTCHR          ;не е, извежда се на екрана
            INY                   ;следващ елемент
            BNE   PRINT_1         ;Y винаги не е нула за стрин-
                                  ;гове по-къси от 255 символа
PRINT_2
* това е бил последния символ от стринга. Извършват се
* действията за край на процедурата PRINTO
            JSR   OUTCHR          ;извежда последния елемент
            JSR   CLREOL          ;изчиства до края на линията
            INC   CURSY           ;следваща Y позиция на курсора

* проверява се дали следващото извеждане няма да стане
* извън рамките на екрана
            LDA   CURSY
            CMP   #MAXLINES+1     ;последен ред от екрана ли е ?
            BLT   PRINT_3         ;не, продължава

* този ред е бил последният, тогава не се увеличава Y пози-
* цията на курсора с което не се допуска scroll
            LDA   #MAXLINES       ;Y позицията на курсора се
            STA   CURSY           ;приравнява на максимално въз-
                                  ;можната за избрания компютър
PRINT_3     RTS                   ;край на PRINTO

*****************************************************************
*
*   Преди да се запознаем със следващите процедури ще раз-
*   гледаме структурата на таблица с отговорите. Тази таб-
*   лица се попълва от процедурата ANS_ и се ползва от
*   процедурата WAIT_. В нея се обявяват вероятните отго-
*   вори, които ще бъдат валидни в текущия екран. Броя на
*   тези отговори не е ограничен, но таблицата не може да е
*   по-дилга от 255 байта (виж по-надолу). Нулирането на
*   таблицата се извършва при обявяването на наов екран (с
*   процедурата SCRN_).
*
*   таблицата с отговорите има следната структура (започва
*   от адрес ANSER) :
*
*
*      индекс на следващия отговор      адрес на обслужващата
```



**** - под правоъгълниците е означена номерацията на полетата от таблицата спрямо началото (ANSER). В скоби са означени символите, чиито ASCII кодове се намират в тези байтове. Всички символи трябва да са с бит7 = 1, а последният - с бит7 = 0.

**** - разпознават се два служебни символа - CR и ESC. Тяхното директно въвеждане не възможно, защото стринговете се дефинират с DCI "..." и те се заместват със символите % и & съответно.

**** - с индекс 0 се отбелязва края на таблицата. Всички следващи отговоли се провавят към края на таблицата. Тъй като индекса е еднобайтов таблицата не може да е по-дълга от 255 символа.

**** - адресите, от които започва обработката на отговора са дефинирани в два байта с подредба мл. - ст.

```
*****************************************************************

*****************************************************************
*
* процедура
* -----------------
```

```
*
* задача :
*   вмъква възможен отговор в опашката. след края на
*   работата възстановява стека в края на стринга на
*   отговора.
*
* входни данни :
*   извикването на тази процедура се извършва по следния
*   начин (примерна програма) :
*
*          JSR  ANS_
*          DCI  "ОТГОВОР%"
*          JMP  LXXXX          ;след изпълнението на ANS_
*                              ;програмата продължава от тази
*                              ;инструкция
*
*          LDA  ETIKET         ;начало на програмата за обра-
*                              ;ботка на този отговор
*          .........
*          RTS                 ;край на обработващата проц.
*  LXXXX   LDA  ETIKET         ;продължение на главната прог-
*                              ;рама
*
*   по този начин входната данна е записания в стека
*   възвратен адрес (в случая адреса на DCI-то). От този
*   адрес се изчислява адреса за продължение на програмата
*   и началния адрес на процедурата за обработка на обяве-
*   ния отговор.
*
* изходни данни :
*   на върха на стека в края на работата на процедурата
*   трябва да стой коректен възвратен адрес (в горния
*   случай адреса на JMP-а). Заедно с това трябва стринга
*   на този отговор и адреса на процедурата за обработката
*   му трябва да се прехвърли в таблицата с отговорите и тя
*   да се модифицира по нужния начин.
*
* ползвани клетки от EQU-зоната :
*
* начин на работа :
*   от стека се извлича адреса, от който започва стринга за
*   очаквания отговор. След това се намира края на таблица-
*   та с отговорите и там се прехвърля новия стринг. След
*   него се записва адреса на процедурата за обработка на
*   този отговор и се установяват новите индекси на табли-
*   цата (виж описанието й по-нагоре). Накрая се възстанови-
*   ва върха на стека (възвратния адрес се получава като
*   към стария адрес се добави дължината на стринга на
*   отговора).
*
* външни процедури : няма
*
*****************************************************************

* в стека се намира адреса след JSR ANS_. там е поставен стр
инга на
* очаквания отговор (DCI "....").
ANSER_
          STY  YSAVE          ;запазване на 6502-reg
```

```
            PLA                 ;извлича се адреса на стрин-
            STA  TEMP           ;га на отговора и се запис-
            PLA                 ;ва в TEMP (2б.)
            STA  TEMP+1

* търси се края на талицата, т.е. нулев индекс
            LDA  #0             ;начало на таблицата
ANS_1       TAY
            LDA  ANSERS,Y       ;индекс на следващ елемент
            BNE  ANS_1          ;последен ли е ? не е, отново

* краят на таблицата е намерен и в Y е индексът му.
            TYA                 ;индекс на последния елемент
            STA  WRK2
            CLC                 ;в TEMP1 се изчислява факти-
            ADC  #<ANSERS       ;ческия адрес на този елемент
            STA  TEMP1          ;като към получения индекс се
            LDA  #>ANSERS       ;прибавя началия адрес на таб-
            STA  TEMP1+1        ;лицата

* стринга на отговора се прехвърля от адресите, указани в
* (TEMP) на адреси (TEMP1), т.е. от извикващата програма в
* таблицата с отговорите.
            LDY  #1             ;начало на стринга на отговора
ANS_2       LDA  (TEMP),Y       ;изчита елемент от стринга
            STA  (TEMP1),Y      ;записва го в таблицата
                                ;последен ли е ?
            BPL  ANS_3          ;да, край на прехвърлянето
            INY                 ;не е, продължава
            BNE  ANS_2

* прехвърлянето е завършило, в Y е броя на фактически
* прхвърлените символи в таблицата
ANS_3       STY  WRK1           ;запазва се броя на символите
            INY                 ;на този индекс се прибавя 3
            INY                 ;и се получава индекса, от къ-
            INY                 ;дето започва следващия запис
            STY  ASAVE          ;запазва се този индекс

            LDA  #0             ;там се записва 0 т.е. край на
            STA  (TEMP1),Y      ;таблицата

            TAY                 ;Y = 0
            LDA  ASAVE          ;индекса на последния запис в
            CLC                 ;таблицата
            ADC  TEMP1
            SBC  #<ANSERS-1
            STA  (TEMP1),Y      ;записва се в полето за индекс
                                ;на текущия запис
            CLC
            LDA  TEMP           ;в TEMP се получава възвратния
            ADC  WRK1           ;адрес на ANSER_процедурата.
            STA  TEMP           ;той сочи последния символ на
            BCC  ANS_4          ;стринга -> при RTS изпълне-
            INC  TEMP+1         ;нието продължава от следващия
                                ;адрес

ANS_4       LDA  TEMP           ;възвратен адрес младша част
```

```
            LDY   WRK1          ;индекс на полето за запис на
            INY                 ;индекса за текущия елемент от
                                ;таблицата

* установява се индекса за текущия елемент и се установя-
* ва коректно върха на стека
            CLC   .              към възвратния адрес се при-
            ADC   #4            ;бавя дължината на JMP XXXX+1
            STA   (TEMP1),Y     ;и получения адрес се записва

            LDA   TEMP+1        ;в стека се поставя старшата
            PHA                 ;част на възвратния адрес
            ADC   #0            ;преноса от прибавянето на 4
            INY                 ;индекс за запис на ст. байт
            STA   (TEMP1),Y     ;записва се полученият адрес
                                ;в таблицата
            LDA   TEMP          ;възвратен адрес младша част
            PHA                 ;слага се в стека
            LDY   YSAVE         ;възстановява Y регистъра
            RTS                 ;край на ANSER_

*****************************************************************
*
* процедура WAIT
* --------------
*
* задача :
*   тази процедура въвежда стринг от клавиатурата и го
*   проверява дали не съответства на обявен (от ANS_)
*   отвовор, като сканира таблицата с отговорите. Ако по
*   време на въвеждането се достигне до еднаквост на
*   въведения стринг с някой от отговорите се изпълнява
*   преход към процедурата за обработването му. При
*   въвеждането на стринга се обслужват необходимите
*   контролни кодове за осигуряването на възможността за
*   редактиране на въвеждания стринг.
*
* входни данни :
*   A и Y          - адрес на стринга, който се извежда
*                    преди да се започне въвеждането на
*                    отговора.
*   ANSERS         - предварително в таблицата с отговорите
*                    трябва да са обявени допустимите (за
*                    този екран) отговори. За повече под-
*                    робности по таблицата виж описанието на
*                    структурата й при процедурата ANS_
*
* изходни данни :
*   конкретни изходни данни няма. Когато е намерен отговор
*   се изпълнява преход към обслужващата го процедура, но в
*   стека (на върха му) седи началния адрес на процедурата
*   WAIT_. По този начин след завършването на обработката
*   на отговора програмата продилжава своето изпълнение пак
*   от WAIT-а. Ако това не е нужно в края на обработващата
*   програма се модифицира стека (чрез оператора GOTO())
*
* ползвани клетки от EQU-зоната :
*   ASAVE, YSAVE  - Общоприети клетки за запазване на 6502-
```

```
*                       регистрите
*    CYRSX, CURSY   - Указатели за позицията на курсора
*    CSAVE          - за запазване на позицията на курсора в
*                     момента на стартирането на процедурата
*                     WAIT_. Познаването на тази позиция е
*                     необходимо за коректното повторно
*                     изпълнение (напр. при въвеждането на
*                     несъществуващ отговор)
*    TEMP1          - в тези клетки се пази адреса, от който
*                     започва повторното изпълнение на
*                     процедурата WAIT_
*    TEMP           - в тези клетки се записва от процедурата
*                     SCAN адреса, от който започва обслужва-
*                     щата процедура за намерения в таблицата
*                     с отговорите отговор
*    BUFER          - от този адрес започва буфера, в който
*                     се въвежда стринга от потребителят
*    BUFLEN         - максимална дължина на този буфер
*    BUFPOS         - указател за текущата позиция от буфера,
*                     която подлежи на модификация
*
* външни процедури :
*    SETSTACK       - установява адреса на следващата
*                     изпълнима инструкция и го записва в
*                     клетките TEMP1
*    PRINTO         - подмножество на описаната по-горе
*                     процедура PRINT_
*    GETKEY         - процедура за въвеждане на символ от
*                     клавиатурата. Успоредно с очакването на
*                     натискането на клавиш се извършват и
*                     фоновите процеси за автоматичната
*                     работа на системата.
*    OUTCHR         - процедура за извеждане на символ върху
*                     екрана на позиция, указана от
*                     указателите на курсора
*    SCAN_O         - процедура за сканиране на таблицата с
*                     отговорите (виж по-надолу)
*
*****************************************************************

WAIT_
            STA  ASAVE          ;запазване на 6502-reg
            STY  YSAVE
            LDA  CURSY          ;запазване на Y-поз на курсора
            STA  CSAVE
            JSR  SETSTACK       ;тази процедура запомня адреса
* на следващата изпълнима инсрукция в TEMP1 (2 байта). Това
* става като се извлича възвратния адрес от стека. За пове-
* че подробности виж обясненията на самата процедура
* SETSTACK

* от тук започва същинската част на процедурата WAIT_. Тя
* прави следното : очаква въвеждането на клавиш (като се
* извиква външната процедура GETKEY). При всяко въвеждане
* се проверява дали току-що въведения символ не е служебен,
* т.е. със ASCII код по-малък от $20. Ако той е такъв се
* прави нова проверка дали е от тези символи, които се
* разпознават от системата. Тези символи са :
```

```
* CR    - въвежда се в края на стринга, който потребителят
*         въвежда от клавиатурата
* ESC   - използва се в специални случаи за отказ или за
*         връщане на по-горно ниво от йерархичната схема на
*         графа без да се извършва модификация
* BS    - изтрива последния въведен символ от буфера и
*         екрана
*
* Ако въведения символ от клавиатурата не е служебен (т.е.
* ASCII кода му е по-голям от $20) се извършва сканиране на
* таблицата с отговорите (започваща от начален адрес ANSERS
* и със структура описана при процедурата ANS_). Проверява
* се дали въведения до момента стринг (в BUFER) не
* съществува в таблицата с отговорите. Самото сканиране се
* извършва от процедурата SCAN. Ако въведения до момента
* стринг съществува като отговор в таблицата с отговорите
* при връщането от процедурата C=1 и адреса от който
* започва обработката на отговорът се намира в клетките TEMP
* (2 байта). В противен случай C=0.
*
* Ако се въведе CR, и отговорът, който се намира в BUFER,
* не съществува о таблицата на отговорите, се приема, че
* въведения отговор е грешен, не се извежда съобщение за
* грешка, изчиства се реда, от който е започнало предишното
* въвеждане и се започва ново въвеждане. По този начин от
* процедурата WAIT_ се излиза единствено с коректен
* отговор, като се прави JMP към началния адрес на
* процедурата за обработка на този отговор. В стека е
* поставен такъв възвратен адрес, че ако процедурата за
* обработка на отговора завършва с RTS, а не с GOTO()
* изпълнението на програмата ще продължи отново от текущата
* WAIT() инструкция.
*
* Адресът, от който започва следващата инструкция се намира
* в клетките TEMP. Тук се идва или чрез естествено падане
* от горната инструкция (JSR SETSTACK) или чрез изпълняване
* на инструкция RTS

            LDA   TEMP1+1       ;този адрес се поставя в стека
            PHA                 ;при RTS изпълнението започва
            LDA   TEMP1         ; пак от тук (ако не е моди-
            PHA                 ;фициран)

            LDA   CSAVE         ;установява позицията на кур-
                                ;сора такава, каквато е била
            STA   CURSY         ;при активирането на WAIT

WAIT_0      LDY   #BUFLEN       ;нулира на буфера за отговор
            LDA   #0
WAIT_1      STA   BUFER,Y
            DEY
            BPL   WAIT_1
            STY   BUFPOS        ;Y=$FF. Тази начална ппозиция
                                ;означава, че няма вкарани
                                ;символи в буфера
            LDA   #0            ;установява се началото на
            STA   CURSX         ;реда
```

```
            LDY   YSAVE         ;извежда съобщението, специ-
            LDA   ASAVE         ;фицирано в WAIT() инструкци-
            JSR   PRINTO        ;ята. Адреса на това съобщение
                                ;се предава като входна данна
                                ;в регистрите A и Y

WAIT_2      LDA   CSAVE         ;възстановява CURSY позицията,
            STA   CURSY         ;която е променента от PRINTO

* тук се проверява дали X-позицията на корсора не е
* достигнала десния край на екрана. Ако е достигнат се
* връща една позиция и току=що въведения символ не влиза в
* буфера. По този начин се ограничава дължината на
* въвеждания стринг(отговор)
            LDA   CURSX         ;текуща позиция
            CMP   #40           ;реда има 40 колони
            BLT   WAIT_21       ;не е достигнат края, прескача
                                ;се връщането на курсора
            DEC   CURSX         ;достигнат е края на екрана
            DEC   BUFPOS

WAIT_21     JSR   GETKEY        ;въвежда символ

* първо се проверяват служедните символи, които се
* разпознават от системата
            CMP   #BS
            BNE   WAIT_3        ;не е BS, нататък

* обработка на BS команда
            DEC   CURSX         ;връща курсора една пози-
                                ;ция назад
            LDA   #SP           ;и символът на тази позиция
            JSR   OUTCHR        ;се изтрива като на негово
                                ;място се пише SP
            DEC   CURSX         ;курсора е пак там
            DEC   BUFPOS        ;намалява се указателят на
                                ;буфера
* проверява се дали е достигнато началото на буфера при
* изтриването на символ от него
            BPL   WAIT_2        ;не е, продължава въвеждането
            BMI   WAIT_0        ;достигнато е, буфера се ну-
                                ;лира и въвеждането започва
                                ;отново

* следващата проверка за служебен символ
WAIT_3      CMP   #CR
            BEQ   WAIT_4        ;ако е CR се обработва

            CMP   #ESC
            BEQ   WAIT_4        ;ако е CR се обработва
* до тук са проверени всички допустими служебни символи.
* Въвеждането на друг служебен симво с код по-малък от $20
* се възприема като грешка и този символ се игнорира, т.е.
* не се обработва
            CMP   #SP           ;служебен символ ?
            BLT   WAIT_2        ;да - не се обработва

* въведения символ е допустим, извежда се на екрана и се
```

```
* вкарва в буферът
WAIT_4      JSR   OUTCHR        ;извежда се на екрана
            INC   BUFPOS        ;следваща позиция на буфера
            LDY   BUFPOS
            STA   BUFER,Y       ;символа се записва в буфера

            PHA
            JSR   SCAN_O        ;сканира се таблицата с
                                ;отговорите
            PLA
            BCS   WAIT_9        ;отговорът е открит и се из-
                                ;пълнява преход към процедура-
                                ;та за обслужването му
* при сканиранео на таблицата отговорите не открит такъв
* отговор
            CMP   #CR           ;проверява се дали потребите-
                                ;лят е завършил въвеждането
            BEQ   WAIT_ERR      ;завършил го е, но такъв отго-
                                ;вор не е бил намерен. Извърш-
                                ;ва се преход към обработката
                                ;на грешно въведен отговор
            BNE   WAIT_2        ;не е и въвеждането продължава

* отговора е намерен и се изпълнява преход към процедурата
* за обслужването му (адреса, записан в TEMP, се установява
* от SCAN, който го взема от таблицата с отговотите)
WAIT_9      JMP   (TEMP)        ;преход към обработка на отгово
р


* потребителят е завършил въвеждането си (въвел е CR), но
* въведеният от него отговор не е бил намерен в таблицата.
* Въвеждането започва от начало.
WAIT_ERR
            DEC   CURSY         ;възстановява се този указател
            RTS                 ;въвеждането започва от начало

*****************************************************************
*
* процедура SETSTACK
* ------------------
*
* задача :
*   установява адреса на следващата изпълнима инструкция и
*   го записва в клетките TEMP1.
*
* входни данни :
*   съдържанието на върха на стека.
*
* изходни данни :
*   в TEMP1 се записва адреса на следващата изпилнима
*   инструкция.
*
* ползвани клетки от EQU-зоната :
*   TEMP1  - 2 байта
*
* начин на работа :
*   от стека се изваждат най-горните два байта и се
*   записват в клетките TEMP1. След това се възстановява
*   върха на стека като току-що прочетените байтове се
```

```
*    записват отново в него.
*
* външни процедури : няма
*
*****************************************************************

SETSTACK    PLA                 ;изваждане на върха на стека
            STA  TEMP1          ;и записването му в клетките
            PLA                 ;TEMP1
            STA  TEMP1+1
            PHA                 ;възстановяване на стека
            LDA  TEMP1
            PHA
            RTS                 ;край на SETSTACK

*****************************************************************
*
* процедура SCAN_0
* ----------------
*
* задача :
*    тази процедура сравнява получения до тука стринг (в
*    BUFER) с описаните в таблицата на отговорите допустими
*    отговори (описването става с процедурата ANS_). Ако
*    отговора се открие измежу описаните в таблицата, тогава
*    C=1 и адреса, от който започва процедурата за
*    обработката му се записва в клетките TEMP
*
* входни данни :
*    BUFER     - съдържанието на буфера в момента на
*                извикването на процедурата се проверява с
*                обявените в таблицата с отговорите стрингове
*    ANSERS    - таблица с възможните отговори
*
* изходни данни :
*    С флага на 6502 се установява в 1 когато отговора е
*    открит в таблицата или в 0, когато не е открит. Ако е
*    бил намерен такъв отговор (C=1) началния адрес на
*    процедурата за обработванетому се записва в клетките
*    TEMP.
*
* ползвани клетки от EQU-зоната :
*    WRK1      - тук се записва индекса на следващия запис в
*    WRK2      - флаг за последен елемент в стринга
*    ANSERS    - таблица с отговорите, обявени с ANS_. За
*                повече подробности виж описанието на ANS_
*                процедурата
*
* външни процедури : няма
*
*****************************************************************


SCAN_0
            LDY  #0             ;нулиране на индекса за след-
            STY  WRK1           ;щия запис
SCAN_1
            LDY  WRK1           ;текущ запис
```

```
            LDX   #-1           ;в X --> позицията в буфера
            STX   WRK2          ;изчистване на флага за пос-
                                ;леден символ от отговора
                                ;(при бит7=0)
            INX                 ;X = 0

            LDA   ANSERS,Y      ;установяване на индекса на
            TAY                 ;следващия запис
            BEQ   SCAN_ERR      ;този запис последен ли е?
                                ;да --> край
            LDA   WRK1          ;взима се индекса на нача-
                                ;лото на текущия запис
            STY   WRK1          ;запомня се индекса на
                                ;следващия запис
            SEC
            ADC   #<ANSERS
            TAY                 ;Y = на текущия заппис

SCAN_2
* взима се символ от таблицата
            LDA   ANSERS/256*256,Y
            STA   WRK2          ;ако е последен символ, тогава
                                ;бит7=0. Като се запише този
                                ;байт във флага за край ще се
                                ;извърши нулирането му
            AND   #$7F          ;бит7 = 0 на текущия символ
            BEQ   SCAN_55       ;празен символ - сравнението
                                ;продължава

* тук се използва следните условни означения на командни
* символи :
* % - вместо CR
* & - вместо ESC
* използването на тези условни означения се налага от
* невъзможността за въвеждането на командни символи направо
* с псевдооперацията на MERLIN DCI "..."
            CMP   #'%'          ;условно означение на CR
            BNE   SCAN_3        ;ако не е проверката продъл-
                                ;жава
            LDA   #CR           ;ако е % се замества с CR
SCAN_3      CMP   #'&'          ;условно означение на ESC
            BNE   SCAN4         ;ако не е - нататък
            LDA   #ESC          ;ако е & се замества с осв
SCAN4
            CMP   BUFER,X       ;сравнява се със символ от
                                ;буфера
            BNE   SCAN_1        ;не са еднакви, следващ
                                ;запис от таблицата
SCAN5
* текущите символи са еднакви, сравнението продължава до
* достигане на края на някои от стринговете или до
* намирането на различие в тях

            INX                 ;следващ символ от буфера
SCAN_55     INY                 ;следващ символ от таблицата
            BIT   WRK2          ;последен ли е бил сравнения
                                ;току-що символ ?
            BMI   SCAN_2        ;не е, сравнението продължава
```

```
* последен е бил, следователно е намерен отговор.
* Сканирането се прекратява, установява се адрес на
* процедурата за обслужване на този отговор и се вдига
* флага за намерен отговор (C=1)
*
* извличане на адреса на процедурата от таблицата и
* записването му TEMP
            LDA  ANSERS/256*256,Y
            STA  TEMP
            LDA  ANSERS/256*256+1,Y
            STA  TEMP+1

            SEC                 ;C=1 търсенето е успешно
            RTS                 ;край на SCAN_0

SCAN_ERR    CLC                 ;C=0 търсенето не е успешно
            RTS                 ;край на SCAN_0


********** край на файла LIB
```

```
* файл SUBR
* последната корекция е била на
* 17-юли-1987
* процедури с общо предназначение

*****************************************************************
*
* процедура GETADR
* ----------------
*
* задача :
*   процедурата от описаните входни данни установява базо-
*   вият адрес на буфера, ползвайки Таблицата с Адресите на
*   Буферите (ТАБ).
*
* входни данни :
*   WDIR      - номер на направление, с което се извиква та-
*               зи процедура. Обикновенно това е и номера на
*               направлението, с което се работи в момента.
*               Заема стойности в интервала 1..4
*   X-reg     - код, указващ вида на буфера, за който трябва
*               да се извлече адреса. Има следните значения:
*               0 - Системен Телеграфен Буфер (СТБ)
*               1 - Системен ASCII Буфер (САБ)
*               2 - Потребителски Телеграфен Буфер (ПТБ)
*               3 - Потребителски ASCII Буфер (ПАБ)
*
* изходни данни :
*   ADR       - адрес на търсения буфер, 2 байта, мл.-ст.
*
* ползвани клетки от EQU-зоната :
*   ASAVE, YSAVE     - за запазване на 6502-reg
*   TEMP(2b), WRK3   - работни
*   TTAB             - начален адрес на ТАБ
*
* начин на работа :
*   От номера на текущото направление се вади 1 (интервала
*   на изменение вече е 0..3). Полученото число се умножа-
*   ва по 16 (за всяко направление са отделени по 16 байта
*   в ТАБ).  Вида на буфера (X-reg) се умножава по 2 (защо-
*   то всеки адрес заема два байта) и получения резултат се
*   прибавя към умноженото по 16 WDIR-1. По този начин се
*   изчислява отместването от началото на ТАБ.  Към това
*   отместване се прибавя началния адрес на ТАБ и се полу-
*   чава адреса, на който е записан (в два байта) търсения
*   адрес на буфера. ТАБ задилжително трябва да започва от
*   цял адрес (младшата част да е 0)!
*
* външни процедури : няма
*
*****************************************************************
GETADR:
            STY   YSAVE         ;запазва ползуваните регистри
            STA   ASAVE
            TXA                 ;X в A за извършване на арит-
                                ;метични операции
            ASL                 ;X*2
            STA   WRK3          ;запазва се като междинен ре-
```

```
                                    ;зултат
            LDA   WDIR              ;на номера на направление,с
            CLC                     ;което се работи се вади 1
            SBC   #0
            ASL                     ;и се умножава по 16
            ASL
            ASL
            ASL
            ORA   WRK3              ;събира се с отместването, по-
                                    ;лучено от вида на буфера и се
            STA   TEMP              ;запазва като младша част
            LDA   #>TTAB            ;старшата част се взема от
            STA   TEMP+1            ;старшата част на TTAB
* до тук в клетките TEMP и TEMP+1 е изчислен адреса на
* който се намира адреса на търсения буфер. Следва
* изчитането на самия адрес и установяването му в ADR
            LDY   #0
            LDA   (TEMP),Y          ;изчита се младшата част
            STA   ADR               ;записва се
            INY
            LDA   (TEMP),Y          ;старша част
            STA   ADR+1
            LDY   YSAVE             ;възстановяват се ползваните
            LDA   ASAVE             ;регистри
            RTS                     ;край на GETADR


*****************************************************************
*
* процедура CHANGE
* ----------------
*
* задача :
*   разменя адресите на двойките буфери (Телеграфен-ASCII)
*   за системните и потребителските буфери. По този начин
*   буферът, който до сега е бил потребителски е станал
*   системен и обратно. (примерно използване при стартиране
*   на предаването на подготвен текст)
*
* входни данни :
*   WDIR      - номер на направление, с което се извиква та-
*               зи процедура.
*
* изходни данни :
*    модифицира се TAБ
*
* ползвани клетки от EQU-зоната :
*   ASAVE, YSAVE     - за запазване на 6502-reg
*   TTAB             - начален адрес на TAБ
*
* начин на работа :
*
* външни процедури : няма
*
*****************************************************************
CHANGE:
            STA   ASAVE             ;запазват се ползваните
            STY   YSAVE             ;регистри на 6502
```

```
                LDA   WDIR          ;от номера на направлени се
                CLC                 ;изчислява отместването от на-
                SBC   #0            ;чалото на таблицата, и резул-
                ASL                 ;татът се получава в Y, като
                ASL                 ;Y = (WDIR-1)/16 + TTAB(мл)
                ASL
                ASL
                CLC
                ADC   #<TTAB+3
                TAY

*  Y-reg сочи в ТАБ тази позиция  +
*                              +-------------+
*           системен           |      потребителски
*   +-----+-----+-----+-----+  +-----+-----+-----+-----+
*   | 00  | 01  | 02  | 03  |  | 04  | 05  | 06  | 07  |
*   +-----+-----+-----+-----+  +-----+-----+-----+-----+
*

                SEI                 ;по време на рзмяната прекъс-
                                    ;ванията се забраняват
CHANGE_1        LDA   TTAB,Y        ;елемент от системната част
                PHA
                LDA   TTAB+4,Y      ;от потребителската част
                STA   TTAB,Y        ;записва се на мястото на сис-
                                    ;темния
                PLA
                STA   TTAB+4,Y      ;а на мястото на негово място
                                    ;се записва запазения системен
                DEY                 ;указателят се придвижва към
                TYA                 ;началото на полето (виж кар-
                                    ;тинката) и
                AND   #$F           ;младшата му част
                CMP   #$F           ;се проверява дали не станала
                BNE   CHANGE_1      ;нула, т.е. достигнато ли е
                                    ;началото на полето. Ако не
                                    ;достигнато преьвьрлянето про-
                                    ;дължава
                CLI                 ;прекъсванията се разрешават
                RTS                 ;край на CHANGE


* въвеждане на стрингове, въвеж-дане и извеждане на
* десетични числа

*****************************************************************
*
* процедура GETSTR
* ----------------
*
* задача :
*
* входни данни :
*
* изходни данни :
*
* ползвани клетки от EQU-зоната :
*
```

```
* начин на работа :
*
* външни процедури : няма
*
*****************************************************************

GETSTR:
            STA   ASAVE          ;запазва 6502-reg
            JSR   GETSTR
            LDA   ASAVE          ;възстановява регистрите
            RTS

* подпрограма
* ползва се от GETSTR: и GETNUM:

GETSTR      LDX   #BUFLEN        ;в целия BUFER (с дължина
            LDA   #0             ;BUFLEN) се записват 0. Ну-
GETSTR2     STA   BUFER,X        ;лирането се извършва от края
            DEX                  ;на буфера към началото
            BPL   GETSTR2
            STX   BUFPOS         ;X=$FF (-1)

            LDA   CURSX          ;позицията на курсора се за-
            STA   CURSX1         ;пазва за специалните ситуации

GETSTR3     JSR   GETKEY         ;фонова процедура за очакване
                                 ;на клавиш. Извършва и прехо-
                                 ;дите по графа в фонов режим.
            CMP   #BS            ;BackSpace изтрива символа
            BNE   GETSTR4        ;преди курсора от буфера и
                                 ;екрана

* следващите действия се извършват когато е въведен BS
            DEC   CURSX          ;изтрива се символа на предиш-
            LDA   #SP            ;ната позиция на курсора, като
            JSR   OUTCHR         ;на негово място се записва SP
                                 ;OUTCHR увеличава позизията на
                                 ;курсора с единица и затова се
            DEC   CURSX          ;налага намаляването й
            DEC   BUFPOS         ;указател за мястото от буфера
                                 ;където се прави промяната
            BPL   GETSTR3        ;ако BUFPOS>=0 изтриването е
                                 ;коректно
* достигнато е началото на буферът (BUFPOS=-1). В този
* случай се възстановява позицията на курсора такава, какват

* е била при влизането в GETSTR и процедурата започва от
* начало
            LDA   CURSX1         ;възтановяване на позицията на
            STA   CURSX          ;курсора
            JMP   GETSTR         ;процедурата започва от начало

GETSTR4
* въведения символ не е бил BS и се извършва проверка за
* другите служебни символи
            CMP   #CR            ;CR е допустим символ за въве-
            BEQ   GETSTR5        ;ждане в буфера
* до тук са проверени всички служебни символи (т.е. control
* символите с кодове <$20. Всички неразпознати такива
```

```
* символи се считат за погрешни и се игнорират
            CMP   #SP           ;SP=$20. Ако въведеният символ
            BLT   GETSTR3       ;има по-малък код се игнорира

GETSTR5
* въведеният символ (от GETKEY е допустим и се записва в
* буфера. Прави се проверка за достигнат край на реда от
* екрана, на които се извършва въвеждането. По този начин
* стрингът, който се въвежда от GETSTR не може да е по-
* дълъг от 40 позиции.
            JSR   OUTCHR        ;извежда се на екрана
            LDX   CURSX         ;след извеждането CURSX сочи
                                ;следващата позиция от реда,
                                ;на която ще се извежда символ
            DEC   CURSX         ;получава се истинската поз.
            CPX   #39           ;последната колона от реда?
            BGE   GETSTR3       ;да (или по-голяма). Тогава се
                                ;очава въвеждане само на CR
            INC   CURSX         ;не, чак тогава символът се
            INC   BUFPOS        ;записва в буфера и се устано-
            LDX   BUFPOS        ;вяват коректо BUFPOS и CURSX
            STA   BUFER,X       ;

            CMP   #CR           ;с CR се завършва въвеждането
            BNE   GETSTR3       ;не е, въвеждането продължава
* това е бил последния символ, който е въведен в буфера и
* се изпълняват операциите за излизане от процедурата
            LDA   CURSX1        ;възстановяване на позицията
            STA   CURSX         ;на курсора
            LDX   BUFPOS        ;X=брой на въведените символи
            RTS                 ;край на GETSTR

*
*
*****************************************************************
*
* процедура  PRDEC:
* -----------------
*
* задача :
*   Да преобразува входното 16-ично число в 10-ично.
*   Да извежда на екрана, преобразуваното 10-ично число.
*
* входни данни :
*   Регистри A и Y съдържат входното 16-ично число.
*
* изходни данни :
*   В клетки D1,D2 и D3 е записано преобразуваното
*   10-ично число.
*
* ползвани клетки от EQU-зоната :
*   H1,H2     - съдържат 16-ичното число за преобразуване
*   D1,D2,D3 - съдържат преобразуваното 10-ично число
*
* начин на работа :
*   Клетките H1 и H2 се разглеждат като 16 отделни бита.
*   Всеки от тях последователно постъпва като най-младши
*   в кл.D1 (останалите битове на D1,D2 и D3 се изместват
```

```
*    наляво). Преди постъпване на нов бит се прави 10-ична
*    корекция на всеки полубайт на D1,D2 и D3.
*
* външни процедури : няма
*
*****************************************************************

*
PRDEC:
            STA   H1              ;Съдържанието на Y и A се
            STY   H2              ;запсва в клетки H1 и H2.
            STX   ASAVE1          ;Запазва се стойността на X.
            LDA   #0              ;Нулират се клетки D1,D2,D3,
            STA   D1              ;където ще бъде записан
            STA   D2              ;резултата.
            STA   D3
            LDY   #15
JC1
            LDX   #2
JC
            LDA   D1,X            ;Извършва се 10-ична
            AND   #$F0            ;корекция на
            CMP   #$50            ;старшия...
            BCC   *+4
            ADC   #$2F
            PHA
            LDA   D1,X            ;...и младшия полубайт
            AND   #$0F            ;на поредния (от трите
            CMP   #$05            ;работни байта).
            BCC   *+4
            ADC   #$02
            AND   #$0F
            STA   D1,X
            PLA
            ORA   D1,X
            STA   D1,X
            DEX                   ;Пристъпва се към обработка
            BPL   JC              ;на следващия байт.

            ASL   H1              ;Съдържанието на H1 и H2 се
            ROL   H2              ;умножава по 2.
            ROL   D1              ;Съдържанието на D1,D2 и D3 се
            ROL   D2              ;умножава по 2, като преноса
            ROL   D3              ;от предишното умножение заема
                                  ;мястото на най-младшя разряд
                                  ;на D1.
            DEY                   ;Пристъпва се към обработка на
            BPL   JC1             ;следващия бит (от 16-те).

            LDA   D2              ;Следва
            BEQ   PRDEC1          ;извеждане
            JSR   PRACC1          ;на D2
PRDEC1      LDA   D1              ;и D1
            JSR   PRACC           ;на екрана.
            LDX   ASAVE1          ;Възстановява се сойността
                                  ;на X регистър.
            RTS                   ;Край на програмата.
```

```
*
*
*****************************************************************
*
* процедура GETNUM
* ----------------
*
* задача :
*
* входни данни :
*
* изходни данни :
*
* ползвани клетки от EQU-зоната :
*
* начин на работа :
*
* външни процедури : няма
*
*****************************************************************

*

GETNUM:

            STX   ASAVE
            STA   LOLIM
            INY
            STY   HILIM

* въвеждат се символи в буфера и се проверяват дали са само
*  цифри. ако не са -> въвеждането се повтаря отново

DCONVO
            JSR   GETSTR
            CPX   #0
            BNE   DCONVLOOP
            LDX   ASAVE
            DEC   HILIM
            LDA   HILIM
            RTS

DCONVLOOP
            DEX
DCONV       LDA   BUFER,X
            CMP   #':'
            BGE   DCONVO
            CMP   #'0'
            BLT   DCONVO
            DEX
            BPL   DCONV

            LDA   #0              ;Нулират се клетки D1,D2,D3,
            STA   D1              ;където ще бъде записано
            STA   D2              ;въведеното 10-ично число.
            STA   D3
            LDX   #0
DC1
```

```
                LDA   BUFER,X       ;Ако поредния (от въведените)
                CMP   #CR           ;символ е CR, се пристъпва към
                BEQ   DH            ;преобразуване на числото.
                AND   #$0F          ;Ако поредния символ е число,
                ASL                 ;младшия полубайт на ASCII
                ASL                 ;кода му се премества
                ASL                 ;на мястото на старшия.
                ASL
                LDY   #3
JC2
                ASL                 ;След това се премества на
                ROL   D1            ;мястото на младшия полубайт
                ROL   D2            ;на D1 (останалите полубайтове
                ROL   D3            ;на D1,D2 и D3 се изместват на
                                    ;ляво).
                DEY                 ;Цикъла се изпълнява 4 пъти
                BPL   JC2           ;(1 полубайт съдържа 4 бита).
                INX                 ;Пристъпва се към обработка на
                JMP   JC1           ;следващия въведен символ.
DCONVO\         JMP   DCONVO

* В клетки D3/D2/D1 е въведено 10-ично число, което ртябва
* да бъде преобразувано в 16-ично.
* Това става като всички битове на D3/D2/D1 и H2/H1 се
* изместват надясно (най-младшия бит на D3/D2/D1 става
* най-старши на H2/H1). След това се извършва 16-ична
* корекция на всички полубайтове на D3/D2/D1.
* Действието се извършва 16 пъти (резултата е 16 битов).

DH
                LDA   #0            ;Нулират се клетките H1 и H2,
                STA   H1            ;където ще бъде записан
                STA   H2            ;резултата.
                LDY   #15
JC1
                LSR   D3            ;Всички битове на D3,D2,D1,H2
                ROR   D2            ;и H1 се изместват надясно.
                ROR   D1
                ROR   H2
                ROR   H1
                LDX   #2
JC
                LDA   D1,X          ;Извършва се 16-ична
                AND   #$F0          ;корекция на
                CMP   #$50          ;старшия...
                BCC   *+4
                SBC   #$30
                PHA
                LDA   D1,X          ;...и младшия полубайт
                AND   #$0F          ;на поредния (от трите
                CMP   #$05          ;работни байта).
                BCC   *+4
                SBC   #$03
                AND   #$0F
                STA   D1,X
                PLA
                ORA   D1,X
                STA   D1,X
```

```
            DEX                 ;Пристъпва се към обработка
            BPL  ЭC             ;на следващия байт.

            DEY                 ;Пристъпва се към обработка на
            BPL  ЭC1            ;следващия бит (от 16-те).

* изходната данна е в H1 и H2,
* се ползва само младшата част -
* в а се зарежда H1

            LDA  H1

* проверява се дали е в очаква-
* ните граници

            LDX  ASAVE
            LDY  H2
            BNE  DCONVO\
*въведеното число е > 256

            CMP  HILIM
            BGE  DCONVO\
            CMP  LOLIM
            BLT  DCONVO\

* изходната данна е добра, край

            RTS


*****************************************************************
*
* процедура  'MOVCURS'
* ------------------
*
* задача :      Инвертиране на текст в/у екрана.
*               Всяко поредно извикване предизвиква
*               инвертиране на един символ и подготовка за
*               следващия.
*               Ползува се от 'GETKEY' в режим 'шоу'.
*
* входни данни :  При 'A' = 0 , започва от началото на
*                 екрана. В всеки друг случай продължава
*                 с поредния символ, като 'CURSX' и 'CURSY'
*                 трябва да имат актуални стойностти.
*
* изходни данни : Пореден инвертиран символ в/у екрана.
*
* ползвани клетки от EQU-зоната : 'CURSX', 'CURSY',
*                                 'BAS'   , 'VIZIGROUPS'
*
* начин на работа :  Ако при извикване 'A' = 0 тогава
*                    се установяват начални координати
*                    на екрана а именно 'X' = 0 и 'Y' =
*                    на константата 'BOTTOM_OF_PG'.
*                    След това се изчислява актуален адрес
*                    от екрана чрез подпрограмата 'BASCALC'
*                    и се инвертира символа намиращ се на
```

```
*                       този адрес.
*                       После се подготвя следващата позиция
*                       за инвертиране чрез увеличаване на
*                       'CURSX'. Ако при това е достигнат
*                       края на реда (определя се от
*                       състоянието на 'VIZIGROUPS'), се
*                       увеличава 'CURSY' и се нулира 'CURSX'.
*                       'VIZIGROUPS' е необходима поради
*                       различният брой символи изписвани на
*                       един ред в зависимост от броя на
*                       символи в група ( 5 или 10 ).
*
* външни процедури : 'BASCALC' - за ичисляване на адрес от
*                       екрана чрез съдържанието на 'CURSX' и
*                       'CURSY'.
*
*****************************************************************

MOVCURS     BNE   MVCR1           ;не установява позиции
                                  ;т.е това е поредният символ
                                  ;от екрана.
            LDA   #0              ;Ако е отначало да зареди
            STA   CURSX           ;позицията (адреса) от която
                                  ;да започнем.
            LDA   #BOTTOM_OF_PG
            STA   CURSY
MVCR1       JSR   BASCALC         ;текушо изчисление
            LDY   CURSX
            LDA   (BAS),Y         ;инвертиране на символ

            DO    ORIC            ;в ORIC вдигнат ст.бит
            ORA   #$80
            ELSE
            AND   #$7F            ;в APPLE свален ст.бит
            FIN

            STA   (BAS),Y         ;край на инвертирането
            INC   CURSX           ;Проверка за край на ред
            LDA   VIZIGROUPS      ;и евентуално прехвърляне на
            CMP   #3              ;нов ред в зависимост от
            BEQ   CHECK32         ;броя групи на ред.
            LDA   #30
            BNE   CHECMOVE
CHECK32     LDA   #33
CHECMOVE    CMP   CURSX           ;Неоходим ли е нов ред ?
            BNE   MVCREND         ;не
            LDA   #0              ;да, тогава установи 'X' = 0
            STA   CURSX
            INC   CURSY           ;и 'Y' = 'Y' + 1.
MVCREND
RTS01       RTS                   ;И край.

*****************************************************************
*
* процедура STATUS
* ----------------
*
* задача :
```

```
*   извеждане на статуса на напр.
*
* входни данни :
*
* изходни данни :
*
* ползвани клетки от EQU-зоната :
*
* начин на работа :
*
* външни процедури : няма
*
*****************************************************************
*
*  преди да разгледаме програмата за извеждане на статуса
*  на направлението нека разгледаме какво представлява
*  самия статус. Успоредно с показването на примерния
*  статус ще извършим и условната номерация на полетата в
*  които се извеждат съобщенията на статуса
*
*  един промерен статус би изглеждал така :
*
*          0         1   2 3 4   5 6   7 8   9
*    A     >НАПРАВЛЕНИЕ >1      >готово за предаване
*    B     >профил. текст   >С НОМЕР  >01
*    C     >цифри с преобладаване на  >abc
*    D     >ДЪЛЖИНА НА ТЕКСТА  >100    >ГРУПИ
*    E     >СКОРОСТ >250  >ЗН/МИН    >50 >ГР/МИН
*          0         1   2 3 4   5 6   7 8 9
*
*  използвани са следните означения :
*  главни букви - текст, който се извежда винаги, незави-
*                 имо от параметрите на текста в буфера
*  малки букви  - текст с променлив характер в зависимост
*                 от съсоянието на буфера
*  >            - с този символ е означено началото на
*                 полето, в което се извършва извеждането на
*                 някакво съобщение. Над всеки такъв символ
*                 (в същата колона) има число, което показа
*                 номера на колоната
*  редовете са номерирани с главните латински букви (от A
*  до E). По този начин номерът на полето се образува като
*  към буквата, указваща реда в който се намира полето,
*  прибавим условния номер на колоната. Напр. "НАПРАВЛЕНИЕ"
*  е A0 поле, "50 ГР/МИН" е написано в полета E7 и E9
*  съответно.
*
*  в тази процедура се използва макроса PR. Ето неговия
*  пълен текст :
*  PR        MAC
*            LDA  #<>1     ;взима се младшият байт от адреса
*            LDY  #>>1     ;след това старшия
*            JSR  PRN      ;извиква се процедурата за писане
*            <<<
*

STATUS:
            STA  ASTAT        ;ACC
```

```
                STY   YSTAT          ;Y REG
                STX   XSTAT          ;X REG

                STA   NUM
                LDA   CSAVE
                STA   CSTAT          ;CYRSY
                LDA   CURSY
                STA   CSAVE

                CPX   #3
                BLT   STO
                BEQ   STO

                JSR   SEARCH
                BCC   RTS01          ;няма такъв
                LDA   ADR1
                STA   ADR
                LDA   ADR1+1
                STA   ADR+1
                JMP   ST1

STO             JSR   GETADR:
ST1             JSR   CLRLN          ;CLS(0-5)
                INC   CURSY
                LDA   CURSY
                CLC
                SBC   #4
                CMP   CSAVE
                BNE   ST1
                >>>   PR.T13
                STA   CURSX
                INC   CURSY

                LDA   CSAVE
                PHA
                LDA   CURSY
                STA   CSAVE
                PLA
                STA   CURSY

                LDA   XSTAT
                CMP   #4
                BGE   ST8
                >>>   PR.T36         ;?"напр "
                LDA   #13
                STA   CURSX
                LDA   WDIR
                ORA   #$30           ;число
                JSR   OUTCHR         ;извежда WDIR
                LDA   #16            ;извежда статус на направление
                STA   CURSX
                LDY   #0
                LDA   (ADR),Y
                CMP   #6
                BNE   ST2
                >>>   PR.T55
                BEQ   ENDST\
```

```
ST2         CMP  #2            ;некофигурианно
            BNE  ST3
            >>>  PR.T53
            BEQ  ENDST\

ST3         CMP  #$40          ;няма текст за предавне
            BNE  ST4
            >>>  PR.T57
ENDST\      JMP  ENDSTAT

ST4         CMP  #4            ;готово за предаване
            BNE  ST5
            >>>  PR.T54

ST5         CMP  #8            ;работи с пълен буфер
            BEQ  ST51
            CMP  #$80
            BNE  ST6
ST51        >>>  PR.T41

ST6         CMP  #$10          ;пауза м/у текст
            BNE  ST7
            >>>  PR.T56

ST7         CMP  #$20          ;работи с празен буфер
            BNE  ST8
            >>>  PR.T58

ST8
* втори ред на статус (вид на текста)
            LDA  XSTAT
            AND  #%00000111
            LSR
            LSR
            TAX

            INC  CURSY
            STX  CURSX
            CPX  #0
            BEQ  ST_

            LDY  #0
            LDA  ASTAT
            JSR  PRDEC:
            LDA  #4
            STA  CURSX

            TXA
            ASL
            ASL
            STA  CURSX

ST_         LDY  TBLSTAT+0,X
            LDA  (ADR),Y
            BNE  ST11
            BEQ  ENDST\

ST11        CMP  #1
```

```
                BNE   ST12          ;тренировъчен
                >>>   PR.T37
                JMP   TYPE

ST12            CMP   #2            ;профилактичен
                BNE   ST13
                >>>   PR.T38
                JMP   TYPE

ST13            CMP   #3            ;датчик
                BNE   ST14
                >>>   PR.T39

ST14            CMP   #4
                BNE   ST20
                >>>   PR.T40

ST20            LDY   CSAVE         ;извеждане дължина на текста
                DEY
                DEY
                DEY
                TXA
                ASL
                ASL
                STA   CURSX
                STY   CURSY
                >>>   PR.T93
                INC   CURSX
                LDY   TBLSTAT+2,X
                LDA   (ADR),Y
                LDY   #0
                JSR   PRDEC:
                INC   CURSX
                >>>   PR.T95

                TXA
                ASL
                ASL
                STA   CURSX
                INC   CURSY

* извежда скоростите

                CPX   #1
                BEQ   SPEEDS
                LDX   XSTAT
                DEX
                JSR   GETADR:

SPEEDS          >>>   PR.T94
                INC   CURSX
                LDY   #$F
                LDA   (ADR),Y
                PHA
                DEY
                LDA   (ADR),Y
                LDY   #0
                JSR   PRDEC:
```

```
                INC   CURSX
                >>>   PR.T96
                TAY
                PLA
                JSR   PRDEC:
                INC   CURSX
                >>>   PR.T97
ENDSTAT
                STA   CURSX
                LDA   CSAVE
                STA   CURSY
                LDA   CSTAT
                STA   CSAVE
                LDA   ASTAT
                LDX   XSTAT
                LDY   YSTAT
                SEC                 ;SET LUBO FLAG FOR CATALOG
                CPX   #4
                BLT   RTS02
                DEC   CURSY
RTS02           RTS


* подпрограма за STATUS
TYPE
                CPX   #1
                BEQ   TYPE1

                INC   CURSX
                >>>   PR.T24        ;с номер
                LDY   #3
                LDA   (ADR),Y
                LDY   #0
                JSR   PRDEC:

TYPE1           TXA
                ASL
                ASL
                STA   CURSX
                INC   CURSY
                LDY   TBLSTAT+4,X
                LDA   (ADR),Y
                CMP   #1            ;кирилица ?
                BNE   TY1
                >>>   PR.T43

TY1             CMP   #2            ;латиница
                BNE   TY2
                >>>   PR.T44

TY2             CMP   #3            ;цифри
                BNE   TY3
                >>>   PR.T45

TY3             CMP   #4            ;смесен
                BNE   TY4
                >>>   PR.T46
```

```
TY4         CMP  #5            ;силно смесен
            BNE  TY5
            >>>  PR.T47

TY5         CMP  #6            ;от касетофон
            BNE  TY6
            >>>  PR.T98

TY6         CMP  #7            ;само от зададените
            BNE  TY10
            >>>  PR.T99
            BEQ  TY11

TY10        INC  CURSX
            LDY  #4
            LDA  (ADR),Y
            BEQ  TYEND
            >>>  PR.T51

TY11        LDY  #4

TY12        LDA  (ADR),Y
            BEQ  TYEND
            JSR  OUTCHR
            INY
            BNE  TY12

TYEND       JMP  ST20

*****************************************************************
*
* процедура PRN
* -----------------
* подпрограма за STATUS
*
* задача :
*
* входни данни :
*
* изходни данни :
*
* ползвани клетки от EQU-зоната :
*
* начин на работа :
*
* външни процедури : няма
*
*****************************************************************

PRN         STA  TEMP
            STY  TEMP+1
            LDY  #0
PRN1        LDA  (TEMP),Y
            BPL  PRN2
            JSR  OUTCHR
            INY
            BNE  PRN1
PRN2        JSR  OUTCHR
```

```
            LDA  #0
            RTS

TBLSTAT     DFB  $E,$11,1,$10,2,0


*****************************************************************
*
* процедура  CONV
* ---------------
*
* задача :
*
* входни данни :
*
* изходни данни :
*
* ползвани клетки от EQU-зоната :
*
* начин на работа :
*
* външни процедури : няма
*
*****************************************************************

CONV:
            LDX  #3
            PHA
            LSR
            LSR
            LSR
            LSR
            JSR  E1:
            INX
            PLA
E1:
            AND  #$F
            ORA  #$B0
            CPX  #3
            BNE  PROVER:
            CMP  #"0"
            BNE  PROVER:
            LDA  #$80
PROVER:
            STA  EXTPRG:,X
            RTS

********** край на SUBR
```

```
* файл ARITM
* последната корекция е била на
* 15-юли-1987
* процедури за изчисление на временните характеристики на
* текст, подлежащ на предаване

              LST  OFF
              EXP  OFF

ORIC          KBD
              DO   ORIC
              ORG  $C600
TEST          =    0
              ELSE
              ORG  $5600
TEST          KBD
              FIN

*****************************************************************
*
* процедура  ARITM
* ----------------
*
* задача :
*   Да изчисли времетраенето на елементите 'точка',
*   'тире', 'пауза между символи', 'пауза между групи'
*   за конкретния текст.
*
* входни данни :
*   В А-регистър е записан номера на обработваното
*   направление.
*   В таблицата на ПТБ са записани скоростите зн/мин и
*   гр/мин.
*
* изходни данни :
*   Времетраенето на елементите 'точка', 'тире', 'пауза
*   между символи', 'пауза между групи' за конкретния
*   текст.
*
* ползвани клетки от EQU-зоната :
*   ADR,ADR1  - две двойки клетки, които се използват за
*               указатели при неявна адресация - (...),Y.
*
* начин на работа :
*   Превръщат се символте,групите и елементарните 'точки'
*   в  обработвания текст. След това се изчислява броя на
*   елементарните паузи по формулата Np = 3*Ns + 2*Ng - 5,
*   тъй като паузата между два символа има продължителност
*   3 елементарни паузи, а паузата между две групи - 5. От
*   формулата 3*Ns + 2*Ng се получава броя на паузите след
*   всеки символ (3) и след всяка група (3 - след послед-
*   ния знак + 2, за да се отделят групите с 5 ел. паузи),
*   но тъй като паузата след последната група на текста не
*   се включва във времетраенето на текста - от получената
*   стойност се изважда 5.
*   Изчислява се времетраенето на текста при скорост 60
*   зн/мин (TU). След това - при скорост 60 гр/мин (TT) и
*   тяхната разлика (TP). На базата на TU се изчислява
```

```
*    времетраенето на ел. точка. Тъй като се използва
*    целочислена аритметика се получава загуба на точност.
*    Това се взема под внимание и при изчисляването на ел.
*    пауза (на базата на TP) и така общото времетраене на
*    текста се запазва.
*
* външни процедури :
*    COUNT  - преброява елементите на обработвания текст
*    MULT:  - 16-битово умножение
*    MULT4: - 32-битово умножение
*    DIV:   - 16-битово деление
*    DIV4:  - 32-битово деление
*
*****************************************************************


ZPAGE          EQU  $80
ABSOLUTE       EQU  $F80

               DUM  ZPAGE
BAS            DS   2
TXT            DS   2
A1L            DS   1
A1H            DS   1
A2L            DS   1
A2H            DS   1
RL             DS   1
RH             DS   1
BL             DS   1
BH             DS   1
               DEND

               DUM  ABSOLUTE
NG             DS   2
NS             DS   2
NE             DS   2
NP             DS   2
TU             DS   2
TT             DS   2
TP             DS   2
CT             DS   2
C_             DS   2
CP             DS   2
CS             DS   2
CG             DS   2
R3             DS   1
R4             DS   1
B3             DS   1
B4             DS   1
A1             DS   1
A2             DS   1
A3             DS   1
A4             DS   1
               DEND

TABTAB         =    $E00          ;TABL OF TABL

TB             =    1
```

```
* 'TB' се измерва в милисекунди

**********************************
*                                *
*          макрокоманди          *
*                                *
**********************************

                DO   0

LOD             MAC
                LDA  ]2
                STA  ]1
                <<<

LDD             MAC
                >>>  LOD.]1;#<]2
                >>>  LOD.1+]1;#>]2
                <<<

MOV             MAC
                >>>  LOD.]1;]2
                >>>  LOD.1+]1;1+]2
                <<<

ADC             MAC
                LDA  ]2
                ADC  ]1
                STA  ]1
                <<<

SUB             MAC
                LDA  ]1
                SBC  ]2
                STA  ]1
                <<<

SB              MAC
                LDA  ]1
                SBC  ]2
                <<<

ORB             MAC
                LDA  ]2
                ORA  ]1
                STA  ]1
                <<<

ASL             MAC
                ASL  ]1
                ROL  ]1+1
                <<<

ROL             MAC
                ROL  ]1
                ROL  ]1+1
                <<<
```

```
LSR             MAC
                LSR   ]1+1
                ROR   ]1
                <<<

ROR             MAC
                ROR   ]1+1
                ROR   ]1
                <<<

PR              MAC
                JSR   $FD8E
                LDA   #<STR
                LDY   #>STR
                JSR   $DB3A
                JMP   CONT
STR             ASC   ]1
                HEX   00
CONT
                LDX   ]2
                LDA   ]2+1
                JSR   $ED24
                <<<

                FIN

************************************
*                                  *
*     Начало на основната          *
*                                  *
*           програма               *
*                                  *
************************************

                JMP   BEGIN

TBL
                DA    TABTAB+$04 ;Таэи таблица се използва при
                DA    TABTAB+$14 ;изчисляване началните адреси
                DA    TABTAB+$24 ;на телеграфните буфери за
                DA    TABTAB+$34 ;четирите направления.

BEGIN
                SEC              ;Установяване начлните адреси
                SBC   #1         ;на ПТБ и неговата таблица.
                ASL
                TAX
                >>>   MOV.TXT;TBL,X
                LDY   #0
                >>>   LOD.BAS;(TXT),Y
                INY
                >>>   LOD.BAS+1;(TXT),Y

                CLC              ;В BAS е записан адреса на
                LDA   BAS        ;таблицата на ПТБ (тя има
                ADC   #$10       ;дължина 16 т.е. $10 байта)
                STA   TXT        ;В TXT е записан адреса на
```

```
                LDA   BAS+1       ;самия текст (той се намира
                ADC   #0          ;след таблицата).
                STA   TXT+1
                JSR   COUNT       ;Извършва се преброяване на
                                  ;елементите на текста.

*   изтисляване на N_пауза (NP)
*   NP = 3*NS + 2*NG - 5

                >>>   LDD.A1L;3
                >>>   MOV.A2L;NS
                JSR   MULT:
                >>>   MOV.NP;RL   ;NP = 3*NS
                >>>   LDD.A1L;2
                >>>   MOV.A2L;NG
                JSR   MULT:       ;RL/RH = 2*NG
                CLC
                >>>   ADC.NP;RL   ;NP = NP+2*NG
                >>>   ADC.NP+1;RH
                SEC
                >>>   SUB.NP;#5   ;NP = NP-5
                >>>   SUB.NP+1;#0

*   изтисляване на т_усл. (TU)
*   TU = 60*NS / скорост на символ

                >>>   LDD.A1L;60
                >>>   MOV.A2L;NS
                JSR   MULT:
                >>>   MOV.A1L;RL ;A1L/A1H = 60*NS
                LDY   #$E         ;A2L/A2H = скорост на символ
                >>>   LOD.A2L;(BAS),Y
                >>>   LOD.A2H;#0
                JSR   DIV:
                >>>   MOV.TU;RL   ;TU = A1 / A2

*   изтисляване на т_текст (TT)
*   TT = 60*NG / скорост на група

                >>>   LDD.A1L;60
                >>>   MOV.A2L;NG
                JSR   MULT:
                >>>   MOV.A1L;RL ;A1L/A1H = 60*NG
                LDY   #$F         ;A2L/A2H = скорост на група
                >>>   LOD.A2L;(BAS),Y
                >>>   LOD.A2H;#0
                JSR   DIV:
                >>>   MOV.TT;RL   ;TT = A1 / A2

*   изтисляване на С_тотка (CT)
*   CT = 1000*TU / NE*TB


                CLC
                >>>   ADC.NE;NP   ;NE = NE+NP
                >>>   ADC.NE+1;NP+1

                >>>   LDD.A1L;1000
```

```
                >>>  MOV.A2L;TU
                JSR  MULT4:      ;A1/A2/A3/A4 = 1000*TU
                >>>  MOV.A1L;NE
                >>>  LDD.A2L;TB
                JSR  MULT:
                >>>  MOV.A1L;RL  ;A1L/A1H = NE*TB
                JSR  DIV4:
                >>>  MOV.CT;RL   ;CT = A1/A2/A3/A4 / A1L/A1H

*   изчисляване на T_пауза (TP)
*   TP = TT - (NE*CT / 1000)

                >>>  MOV.A1L;NE
                >>>  MOV.A2L;CT
                JSR  MULT4:      ;A1/A2/A3/A4 = NE*CT
                >>>  LDD.A1L;1000
                JSR  DIV4:       ;RL/RH/R3/R4 = NE*CT / 1000
                >>>  MOV.TP;TT
                SEC
                >>>  SUB.TP;RL   ;TP = TT - RL/RH
                >>>  SUB.TP+1;RH

*   изчисляване на C_тире (C_)
*   C_ = 3*CT

                >>>  LDD.A1L;3
                >>>  MOV.A2L;CT
                JSR  MULT:
                >>>  MOV.C_;RL

*   изчисляване на C_пауза (CP)
*   CP = 1000*TP / NP*TB + CT

                >>>  LDD.A1L;1000
                >>>  MOV.A2L;TP
                JSR  MULT4:      ;A1/A2/A3/A4 = 1000*TP
                >>>  MOV.A1L;NP
                >>>  LDD.A2L;TB
                JSR  MULT:       ;RL/RH = NP*TB
                >>>  MOV.A1L;RL
                JSR  DIV4:       ;RL/RH/R3/R4 = 1000*TP / NP*TB
                >>>  MOV.CP;RL
                CLC
                >>>  ADC.CP;CT   ;CP = RL/RH + CT
                >>>  ADC.CP+1;CT+1

*   изчисляване на C_символ (CS)
*   CS = 3*CP

                >>>  LDD.A1L;3
                >>>  MOV.A2L;CP
                JSR  MULT:
                >>>  MOV.CS;RL

*   изчисляване на C_група (CG)
*   CG = 5*CP

                >>>  LDD.A1L;5
```

```
                >>>  MOV.A2L;CP
                JSR  MULT:
                >>>  MOV.CG;RL

*  Поставяне на резултатите в таблицата на ПТБ.

*  Записва се параметъра 'времетраене на точка'.
                LDY  #4
                >>>  LOD.(BAS),Y;CT
                INY
                >>>  LOD.(BAS),Y;CT+1

*  Записва се параметъра 'времетраене на тире'.
                LDY  #6
                >>>  LOD.(BAS),Y;C_
                INY
                >>>  LOD.(BAS),Y;C_+1

*  Записва се параметъра 'пауза между символи'.
                LDY  #8
                >>>  LOD.(BAS),Y;CS
                INY
                >>>  LOD.(BAS),Y;CS+1

*  Записва се параметъра 'пауза между групи'.
                LDY  #$A
                >>>  LOD.(BAS),Y;CG
                INY
                >>>  LOD.(BAS),Y;CG+1

*  Записва се параметъра 'времетраене на текста'.
                LDY  #$C
                >>>  LOD.(BAS),Y;TT
                INY
                >>>  LOD.(BAS),Y;TT+1

*  Статуса на буфера се установява в състояние
*  'готов за предаване'.

                LDY  #0
                >>>  LOD.(BAS),Y;#4

* контролна визуализация на резултатите

                DO   TEST

                >>>  PR."групи  :";NG
                >>>  PR."символи:";NS
                >>>  PR."NE     =";NE
                >>>  PR."NP     =";NP
                >>>  PR."TU     =";TU
                >>>  PR."TT     =";TT
                >>>  PR."TSP    =";TP
                >>>  PR."TP     =";CP
                >>>  PR."CT     =";CT
                >>>  PR."C_тире =";C_
                >>>  PR."C_симв.=";CS
                >>>  PR."C_група=";CG
```

```
            >>>  PR."T_ов]о =";TT

            FIN

            CLC
            RTS

**********************************
*                                *
*      подпрограми               *
*      -----------               *
*                                *
**********************************

*****************************************************************
*
* процедура  COUNT
* ----------------
*
* задача :
*   Да бреброй групите,символите и елементарните точки
*   в обраротвания морзов текст.
*
* входни данни :
*   TXT/TXT+1 - указател към началото на морзовия текст.
*
* изходни данни :
*   NG/NG+1 - брой групи
*   NS/NS+1 - брой символи
*   NE/NE+1 - брой елементарни точки
*
* ползвани клетки от EQU-зоната : няма
*
* начин на работа :
*   Всяка група от морзовия текст завършва с 0. Група,
*   която започва с 0, означава край на текста. Символите
*   се съхраняват в отделни байтове. Всеки байт съдържа
*   нули и единици. Нулите се тълкуват като 'точки', а
*   единиците - като 'тирета'. Тирето се тъкува като три
*   точки. Между тиретата и точките в един морзов символ
*   се оставя пауза, с продължителност една 'точка'.
*   На базата на тази информация COUNT преброява групите,
*   символите и елементарните точки в обраротвания морзов
*   текст.
*
* външни процедури : няма
*
*****************************************************************

*
COUNT
            LDA  #0          ;Нулиране на използваните
            STA  NG          ;променливи:
            STA  NG+1        ;брой групи
            STA  NS
            STA  NS+1        ;брой символи
            STA  NE
            STA  NE+1        ;брой елементарни точки
```

```
                STA   NP
                STA   NP+1          ;брой елементарни паузи
ЗС1
                LDY   #0            ;Ако поредната група започва
                LDA   (TXT),Y       ;с 0, значи това е края на
                BEQ   END1          ;морзовия текст.
ЗС2
                LDA   (TXT),Y       ;Ако поредния символ в групата
                BEQ   NEXTG         ;е 0, значи това е край на гр.
                LDX   #0            ;В X-регистър ще се нструпва
                                    ;броя на елементарните точки
                                    ;в поредния морзов символ.
ЗС3
                ASL                 ;B7 на A се прехвърля в C.
                BEQ   CONT1         ;Ако A=0, значи всички битове
                                    ;в A са обработени.
                BCC   *+4           ;Ако C=0 -> X=X+2 (точка плюс
                INX                 ;пауза след нея).
                INX                 ;Ако C=1 -> X=X=4 (три точки
                INX                 ;плюс пауза след тях).
                INX
                JMP   ЗС3           ;Обработката на морзовия
                                    ;символ не е завършена.
CONT1
                DEX                 ;Паузата след последния
                                    ;елемент от морзовия
                                    ;символ се пренебрегва.
                TXA
                CLC
                ADC   NE            ;Получения брой точки се
                STA   NE            ;натрупва в NE/NE+1.
                >>>   ADC.NE+1;#0
                INC   NS            ;Броя на символите се
                BNE   NEXTS         ;увеличава с 1.
                INC   NS+1
NEXTS
                INY                 ;Пристъпва се към обработка
                JMP   ЗС2           ;на следващия морзов символ.
NEXTG
                INC   NG            ;Броя на групите
                                    ;се увеличава с 1.
                TYA
                SEC                 ;Към TXT/TXT+1 се прибавя Y,
                ADC   TXT           ;за да сочи началото на
                STA   TXT           ;следващата група.
                >>>   ADC.TXT+1;#0
                JMP   ЗС1           ;Пристъпва се към обработка
                                    ;на следващата група.
END1
                RTS                 ;Край на процедурата.


*****************************************************************
*
* процедура  MULT:
* ----------------
*
* задача :
```

```
*    Да умножи 16-битово A1L/A1H и A2L/A2H .
*
* входни данни :
*    A1L/A1H - множимо
*    A2L/A2H - множител
*
* изходни данни :
*     RL/RH  - резултат
*
* ползвани клетки от EQU-зоната : няма
*
* начин на работа :
*    Използва се стандартен метод за умножение с изместване
*    на резултата и множителя наляво и натрупване на
*    множимото.
*
* външни процедури : няма
*
*****************************************************************
*
MULT:
              LDA  #0
              STA  RL           ;Нулират се клетките за
              STA  RH           ;резултата.

              LDY  #15
JCYCLE1
              >>>  ASL.RL       ;Резултата се измества с 1
                                ;бит наляво.
              >>>  ASL.A2L      ;Множителя се измества с 1
                                ;бит наляво (най-старшия бит
                                ;постъпва в C).
              BCC  NEXTM        ;Ако C=0 не се извършва
                                ;натрупване на множимото към
                                ;резултата.
              CLC
              >>>  ADC.RL;A1L   ;Множимото се прибавя към
              >>>  ADC.RH;A1H   ;резултата.
NEXTM
              DEY               ;Цикъла се изпълнява
              BPL  JCYCLE1      ;16 пъти.
              RTS               ;Край на процедурата.


*****************************************************************
*
* процедура  MULT4:
* -----------------
*
* задача :
*    Да умножи 32-битово A1L/A1H и A2L/A2H .
*
* входни данни :
*    A1L/A1H - множимо
*    A2L/A2H - множител
*
* изходни данни :
*    A1/A2/A3/A4 - 32-битов резултат
```

```
*
* ползвани клетки от EQU-зоната : няма
*
* начин на работа :
*   Използва се стандартен метод за умножение с изместване
*   на резултата и множителя наляво и натрупване на
*   множимото.
*
* външни процедури : няма
*
*****************************************************************
*
MULT4:
              LDA  #0
              STA  A1           ;Нулират се клетките за
              STA  A2           ;резултата.
              STA  A3
              STA  A4

              LDY  #15
JC
              ASL  A1           ;Резултата се измества с 1
              ROL  A2           ;бит наляво.
              ROL  A3
              ROL  A4

              >>>  ASL.A2L      ;Множителя се измества с 1
                                ;бит наляво (най-старшия бит
                                ;постъпва в C).
              BCC  LBL1         ;Ако C=0 не се извършва
                                ;натрупване на множимото към
                                ;резултата.
              CLC
              >>>  ADC.A1;A1L   ;Множимото се прибавя към
              >>>  ADC.A2;A1H   ;резултата.
              >>>  ADC.A3;#0
              >>>  ADC.A4;#0
LBL1
              DEY               ;Цикъла се изпълнява
              BPL  JC           ;16 пъти.
              RTS               ;Край на процедурата.


*****************************************************************
*
* процедура  DIV:
* ---------------
*
* задача :
*   Да раздели 16-битово A1L/A1H на A2L/A2H .
*
* входни данни :
*   A1L/A1H - делимо
*   A2L/A2H - делител
*
* изходни данни :
*    RL/RH  - резултат
*
```

```
* ползвани клетки от EQU-зоната : няма
*
* начин на работа :
*   Значещите разряди на делителя се изместват в
*   най-лявата част на разрядната решетка (в клетки BL/BH
*   се пази информация за изместването на делителя от
*   началната му позиция). След това започва същинската
*   част на аритметичната операция деление.
*   Сравняват се числата от разрядните решетки на делимото
*   и делителя. Ако е възможно от делимото да се извади
*   делителя - резултата от изваждането се поставя в
*   разрядната решетка на делимото, а на съответната
*   позиция в решетката на резултата от делението се
*   записва 1 (в противен случай се записва 0, а резултата
*   от изваждането се пренебрегва).
*   След това делителя се измества с 1 разряд надясно
*   (изместването се отбелязва в BL/BH) и отново се
*   извършва сравнение.
*   Действията се извършват докато делителя заеме
*   първоначалното си положение.
*
* външни процедури : няма
*
*****************************************************************
*
DIV:
              LDA   #0
              STA   RL            ;Нулират се клетките за
              STA   RH            ;резултата.
              STA   BH            ;В BL/BH се записва $0001.
              LDA   #1
              STA   BL

              LDA   A2H           ;Ако най-старшият разряд от
              BMI   DIVISION      ;разрядната решетка на
                                  ;делителя е 1 - не се
                                  ;извършва изместване.
              ORA   A2L           ;Ако разрядната решетка на
              BEQ   END           ;делителя не съдържа единици -
                                  ;делението не се извършва.

JCYCLE
              >>>   ASL.BL        ;Значещите разряди в решетката
              >>>   ASL.A2L       ;на делителя се изместват в
              BPL   JCYCLE        ;най-лявата и част (степента
                                  ;на изместване се отбелязва в
                                  ;BL/BH).

DIVISION
JCYCLE
              SEC
              >>>   SB.A1L;A2L    ;Делителя се изважда от
              TAX                 ;делимото (резултата остава
              >>>   SB.A1H;A2H    ;в A и X регистри).
              BCC   NEXTD         ;Ако се получи пренос -
                                  ;резултата се пренебрегва.
              STX   A1L           ;Резултата се записва в
              STA   A1H           ;решетката на делимото.
```

```
                >>>   ORB.RL;BL   ;На съответната позиция в
                >>>   ORB.RH;BH   ;решетката на резултата се
                                  ;записва 1.
NEXTD
                >>>   LSR.A2L     ;Делителя се измества с 1 бит
                >>>   LSR.BL      ;надясно (изместването се
                                  ;отбелязва в BL/BH).
                BCC   JCYCLE      ;Ако делителя не е достигнал
                                  ;началната си позиция -
                                  ;действията се повтарят.
END
                RTS               ;Край на процедурата.

;*****************************************************************
;
; процедура  DIV4:
; ----------------
;
; задача :
;   Да раздели 32-битово A1/A2/A3/A4 на A1L/A1H/A2L/A2.
;
; входни данни :
;   A1/A2/A3/A4     - делимо
;   A1L/A1H/A2L/A2  - делител
;
; изходни данни :
;   RL/RH/R3/R4     - 32-битов резултат
;
; ползвани клетки от EQU-зоната : няма
;
; начин на работа :
;   Действието на DIV4: е аналогично на това на DIV:, но
;   при DIV4: регистрите са 32-битови (а не 16-битови).
;
; външни процедури : няма
;
;*****************************************************************
;
DIV4:
                LDA   #0          ;Нулират се клетките за
                STA   RL          ;резултата.
                STA   RH
                STA   R3
                STA   R4
                STA   A2L         ;Нулират се двата старши
                STA   A2H         ;байта на делителя и на
                STA   B3          ;буфера, в който се отбелязва
                STA   B4          ;изместването на делителя.
                STA   BH
                LDA   #1          ;В BL/BH се записва $0001.
                STA   BL

                LDA   A1L         ;Ако разрядната решетка на
                ORA   A1H         ;делителя не съдържа единици -
                BEQ   END         ;делението не се извършва.

JC
                >>>   ASL.BL      ;Значещите разряди в решетката
```

```
                >>>   ROL.B3       ;на делителя се изместват в
                >>>   ASL.A1L      ;най-лявата и част (степента
                >>>   ROL.A2L      ;на изместване се отбелязва в
                BPL   JC           ;BL/BH/B3/B4).

JC
                SEC
                >>>   SB.A1;A1L    ;Проверява се дали е възможно
                >>>   SB.A2;A1H    ;коректно изваждане на-
                >>>   SB.A3;A2L    ;делителя от делимото.
                >>>   SB.A4;A2H
                BCC   DLBL         ;Ако се е получил пренос -
                                   ;изваждането не се извършва.
                >>>   SUB.A1;A1L   ;Делителя се изважда от
                >>>   SUB.A2;A1H   ;делимото (резултата остава
                >>>   SUB.A3;A2L   ;в решетката на делимото).
                >>>   SUB.A4;A2H
                >>>   ORB.RL;BL    ;На съответната позиция в
                >>>   ORB.RH;BH    ;решетката на резултата се
                >>>   ORB.R3;B3    ;записва 1.
                >>>   ORB.R4;B4
DLBL
                >>>   LSR.A2L      ;Делителя се измества с 1 бит
                >>>   ROR.A1L      ;надясно (изместването се
                >>>   LSR.B3       ;отбелязва в BL/BH/B3/B4).
                >>>   ROR.BL
                BCC   JC           ;Ако делителя не е достигнал
                                   ;началната си позиция -
                                   ;действията се повтарят.
                RTS                ;Край на процедурата.

********** край на ARITM
```