

UŽIVATELSKÁ PŘÍRUČKA

OSOBNÍ MIKRO POČÍTAČ

PMD-85

II. PROGRAMOVACÍ JAZYK BASIC G



TESLA - ELEKTRONICKÉ SOUČÁSTKY

OSOBNÍ MIKROPOČÍTAČ PMD-85

UŽIVATELSKÁ PŘÍRUČKA

II. PROGRAMOVACÍ JAZYK BASIC G



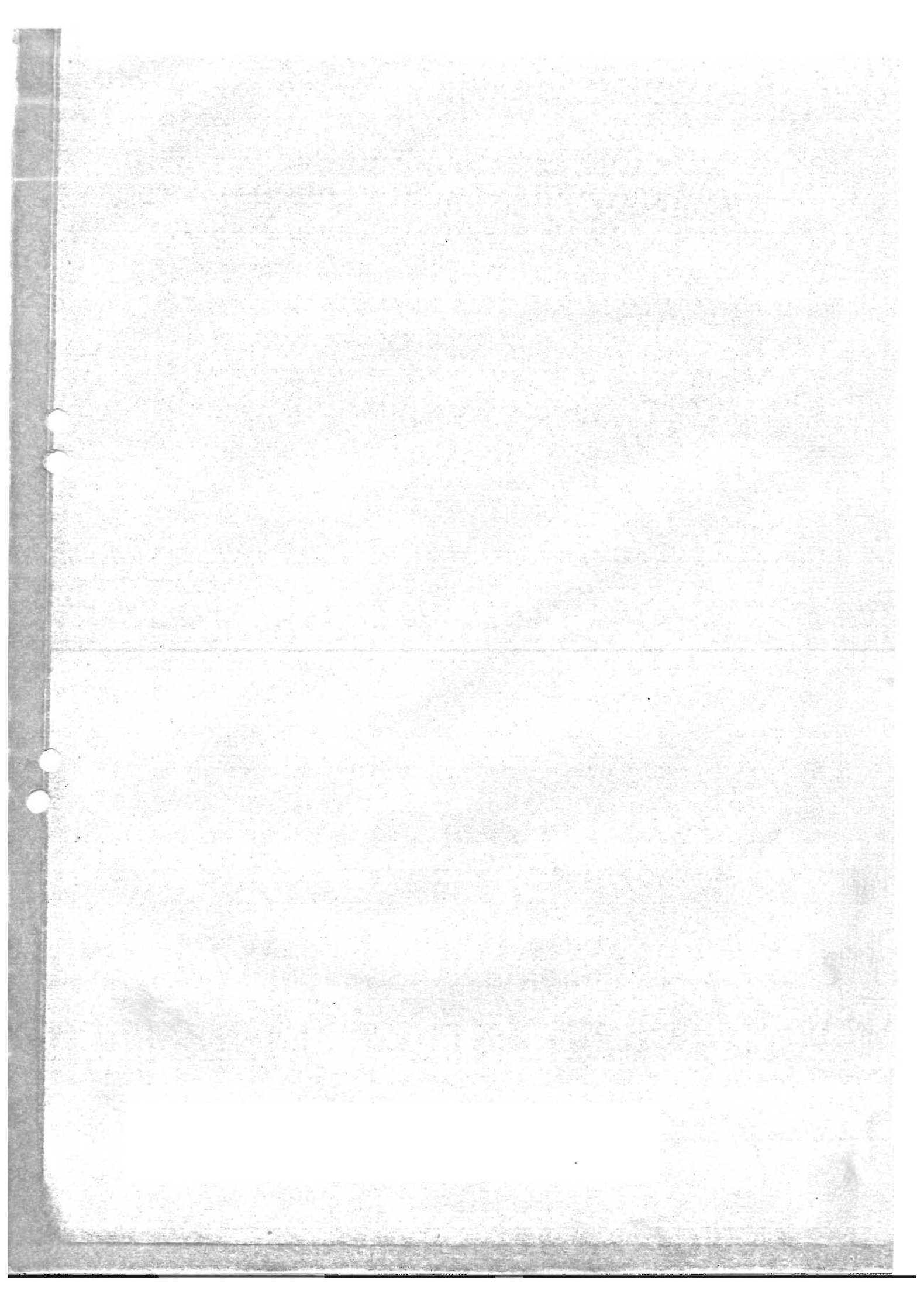
TESLA - ELEKTRONICKÉ SOUČÁSTKY

OBSAH

Přímý mód	1
Programový mód	1
Číslování řádků	2
Vícenásobný příkazový řádek	2
Počet znaků v řádku	2
Číselné formáty	3
Aritmetické proměnné	3
Aritmetické výrazy	4
Matematická hierarchie	5
Pole	6
Řetězce /strings/	7
Funkce	7
Příkazy	9
Řídící příkazy	9
Standardní příkazy	11
DEF	12
DIM	12
DSAVE	12
DLOAD	13
FOR TO NEXT (STEP)	14
DISP	14
GCLEAR	15
GCSUB	15
GOTO	15
IF	16
INPUT	17
LET	17

NEXT	18
ON	18
PAUSE	19
PRINT	20
TAB	20
READ a DATA	21
RESTORE	21
RETURN	22
STOP	23
REM	23
Příkazy pro grafické zobrazení	23
SCALE	24
MOVE	24
AXES	25
PLOT	25
BMOVE	26
BPLOT	26
LABEL	27
FILL	28
Příkazy pro vstup/výstup(I/O).....	29
OUTPUT	29
Druhy kanálů	29
Výstupní kanál-sériový	31
Volba pracovního režimu sériového kanálu	31
Výstupní kanál paralelní (GPIO)	34
Stavová slova pro režim s podporou handshake.....	36
Stavové slovo pro obousměrný I/O režim	37
Výstupní kanál IMS - 2 (HPIB)	37
CONTROL	37
ENTER	39
STATUS	39

Příkazy pro systém 8080	40
ROM	40
USR	41
PEEK	42
POKE	42
OUT	43
IN	43
WAIT	43
CODE	44
Zprávy o chybách	45



"BASIC - G" je graficky orientovaný programovací jazyk BASIC - určený pro počítač PMD 85 a jeho modifikace.

Interpreter Basic - G je uložený buď v programovém ROM modulu nebo na kazetové pásce, odkud je pomocí příkazů operačního systému počítače /"BASIC G" pro ROM modul a "MGLD" pro

magnetofon / zavedený do operační paměti počítače, kde zabírá oblast 9 k byte. Pro uživatelské programy a řetězcové proměnné je vyhrazena oblast přibližně 19 k byte.

Styk operátora s počítačem se uskutečňuje pomocí alfanumerické klávesnice a televizního monitoru, případně televizního přijímače černobílého nebo barevného, který slouží jako alfanumerický a grafický výstup s rastroem 286x256 bodů.

Všechny zprávy a vstupy do BASIC - G interpreteru se realizují prostřednictvím dialogového řádku. Ukončení práce v dialogovém řádku se provádí klávesou EOL.

Interpreter BASIC - G rozlišuje dva způsoby práce

- přímý mód
- programový mód

Přímý mód

Každý vstupní řádek do interpreteru, který nezačíná číslem je považován jako příkaz pro přímé vykonání.

Příklad: PRINT SIN (.12)

Po vykonání příkazu vypisuje BASIC - G hlášení "OK", čímž ukončuje daný příkaz.

Programový mód

Vstupní řádek začínající číslicí se považuje za progr-

nový řádek jazyka BASIC - G a je uložen do jeho programové části. Po vykonání příkazu interpreter odpovídá znakem >, nebo pokud se jedná o vkládání programových řádků pod příkazem AUTO, vypisuje se číslo následujícího řádku.

Příklad: 100 PRINT SIN (.12)

Číslování řádků

Každý programový řádek musí obsahovat své pořadové číslo, pod kterým je evidován v programovém bloku. Toto číslo je celočíselné a musí být v rozsahu 0 až 65535.

Je praktické ukládat jednotlivé programové řádky s inkrementem 5 nebo 10, aby bylo mezi ně možné dodatečně vkládat další.

Vícenásobný příkazový řádek

BASIC - G umožňuje vkládat do jednoho řádku více než jeden příkaz. Jednotlivé příkazy se oddělují dvojtečkou.

Příklad: 10 A=2:B= SIN (A) :PRINT A, B

Poznámka: Při vkládání příkazového řádku není podmínkou vkládání mezer, závisí to na individuálním přístupu operátora.

Počet znaků v řádku

je omezený na 80, přičemž je možné zobrazit pouze prvních 48. Avšak při vkládání, případně editování v dialogovém řádku, je možné zobrazit libovolnou část z 80 znaků posunem dialogového řádku v horizontálním smeru. Při vkládání 70. znaku je

operátor upozorněn akusticky na přibližující se konec řádku.

Číselné formáty

Každé absolutní číslo, které je v rozsahu

0.1 až 999999

je vytištěno v pevném formátu bez exponentu. Všechna ostatní čísla jsou v následujícím formátu:

SM. MMMMESXX,

kde S je znaménko mantisy nebo exponentu

M je mantisa

X je exponent

E je identifikátor

Rozsah čísel je 1×10^{-38} až 1×10^{38}

Formáty čísel

Příklady:

operátor

počítač

2 192 000

2.192E + 06

- 0

0

108.999

108.999

.0000256789

2.56789 E-05

-3.0

-3

.43E-2

4.3E-03

1/16

.0625

ARITMETICKÉ PROMĚNNÉ

Názvy aritmetických proměnných jsou jednoznakové, písmenné nebo dvojnakové, alfanumerické.

Příklad: X, A, Al, ER

ARITMETICKÉ VÝRAZY

aritmetické výrazy mohou být definovány jako:

1. jednoduché proměnné
2. prvky z definovaného pole
3. BASIC funkce
4. uživatelem definované funkce
5. konstanty
6. aritmetické operátory
7. relační operátory
8. logické operátory

OPERÁTORY PRO ARITMETICKÉ VÝRAZY JSOU:

Symbol	typ	příklad	význam
+	aritm.	$A+B$	sčítání
-	aritm.	$A-B$	odčítání
*	aritm.	$A*B$	násobení
/	aritm.	A/B	dělení
^	aritm.	A^B	umocňování
polarita +	aritm.	$+A$	kladné číslo A
polarita -	aritm.	$-A$	záporné číslo A
NOT	logický	NOT A	dvojkový komplement A
OR	logický	A OR B	logický součet /binárně/
AND	logický	A AND B	logický součin /binárně/
=	relační	$A=B$	A je rovno B
<	relační	$A<B$	A je menší než B
>	relační	$A>B$	A je větší než B
<=	relační	$A<=B$	A je menší nebo rovno B

symbol	typ	příklad	význam
\geq	relační	$A \geq B$	A je větší nebo rovno B
$\langle \rangle$	relační	$A \langle \rangle B$	A není rovno B

MATEMATICKÁ HIERARCHIE

$()$
funkce

nejvyšší

\cup

polarita +, -

\times nebo /

sčítání nebo odčítání

relační operátory

logický NOT

logický AND, OR



Pokud jsou dva operátory ve stejné rovině hierarchie, vykonávání operace jde zleva doprava.

Příklad:

$$20 - A + B \times C \cup D$$

1. $20 - A$

2. $C \cup D$

3. B je násobeno hodnotou získanou v 2.

4. Hodnota ad. 1. je sčítána s hodnotou ad. 3.

Polarita minus a plus má význam pouze na začátku výrazu.

Příklad:

$$-8 + ED$$

$$AA + VG \times (-X/32)$$

Největší prioritu při zpracovávání výrazu mají výrazy v závorkách.

Příklad:

$$20 - ((A + B) \times C) \cup D$$

1. $A + B$
2. Výsledek z 1. je násoben s C
3. Výsledek z 2. je umocněn D
4. Výsledek ad. 3 je odčítán od čísla 20

Poznámka:

na některých místech je nutno používat závorky, protože by se mohlo dojít k jiným vztahům.

Příklad:

$(A + B) / 2 * C$ není totožný s výrazem $(A + B) / (2 * C)$

POLE

Slouží pro ukládání dat /čísla, řetězce/ a pracuje se s nimi jako s proměnnými. Mohou být označeny písmenem nebo číslicí.

Pokud se jedná o řetězcové pole, přibývá ještě znak \$ /dolar/.

BASIC - G automaticky určuje počet prvků max. 11 pro každou dimenzi. V případě větších dimenzí je nutné použít příkaz DIM.

Příklad:

$DIM X (15) , X2 (2,8) , X3 (5,5,5)$

kde X je jednorozměrné pole o 16 prvcích /0 - 15/

X2 je dvojrozměrné pole o 27 prvcích /3x9/

X3 je trojrozměrné pole o 216 prvcích /6 x 6 x 6/

Element prvku v poli může být deklarován jako výraz a BASIC - G si jej přepočítává na celočíselnou hodnotu.

Příklad:

$A(K=11, CT)$

$B(1, 15)$

$B(ABS(I * J + 1) , K * L)$

$A(9. 2, 10. 3)$ je pole $A(9,10)$

Řetězce /strings/

jsou označovány na začátku a na konci značkou ". Délka řetězce může být max. 255 znaků. Řetězcové proměnné mohou být označovány jednoznakově X\$ nebo dvojznakově XR\$, XL\$. BASIC - G tedy rozlišuje tři různé tvary proměnných.

Příklad:

A, A (1), A\$

Řetězcové proměnné mohou být definovány i v polích.

Příklad:

DIM A\$ (10, 10)

znamená, že A\$ obsahuje 10 skupin, přičemž každá skupina obsahuje 10 řetězcových elementů. Řetězcový výraz může být řetězec znaků, řetězec proměnných nebo prvek řetězcového pole /string array element/. Řetězcové výrazy mohou být pouze sčítány, přičemž délka výsledného řetězce nesmí překročit 255 znaků.

Řetězcový výraz může používat všechny relační operátory(=, >, < atd.).

Funkce

BASIC - G obsahuje tyto standartní funkce:

SIN(X) sinus X, kde X je v rad.

COS(X) cosinus X, - " -

TAN(X) tangens X, kde X je v rad.

ATN(X) arcustangens X, kde $\pi/2 = \text{ATN}(X) = \pi/2$

LOG(X) přirozený logaritmus X ($X > 0$)

EXP(X) $(-88 < X < 87)$ x-tá mocnina čísla "e"

SQR(X) odmocnina X ($X > 0$)

ABS(X) absolutní hodnota X

INT (X) celočíslná hodnota X

Mimo tyto aritmetické funkce jsou další:

INP (I) přiřazení byte od portu I

RND (X) generování náhodného čísla v rozsahu 0 - 1

SGN (X) znaménko hodnoty X

FRE (X) volná paměťová oblast pro BASIC - G program a data

SPC (I) vkládání I mezer na printer. Používá se pouze pod příkazem PRINT

TAB (I) tabulátor - použití pouze pod příkazem PRINT

BIT A, B určení hodnoty bitu u A na váze B

A může být v rozsahu 0 - 255

B může být v rozsahu 0 - 7

STATUS A, B načítání hodnoty z I/O kanálu A, a registru B, blíže viz část I/O

INKEY načítání hodnoty KEY klávesy. Při nestlačení se generuje hodnota 255. Při zatlačení KEY klávesy nabývá hodnotu (0 - 10)

PEEK (I) načítání hodnoty byte z paměťového místa I

ASC (X\$) decimální hodnota prvního znaku řetězce X\$

CHR\$ (I) převod decimální hodnoty I (0 - 255) na ASCII hodnotu

FRE (X\$) zjištění volného místa v BASIC - G pro uložení řetězcových proměnných

LEFT\$ (X\$, I) načítání zleva I počet znaků z X\$

RIGHT\$ (X\$, I) načítání zprava I počet znaků řetězce X\$

MID\$ (X\$, I [, K]) načítání části řetězce, který začíná na I-tém znaku a je dlouhý K znaků z řetězce X\$

LEN (X\$) určení délky řetězce X\$

STR\$ (X) změna čísla X na řetězec

VAL (X)	změna řetězce na číslo
USR (X)	volání uživatelské rutiny s adresou X. Parametr se odevzdává přes akumulátor do žádáné proměnné.

PŘÍKAZY

V této části bude popsáno pět druhů příkazů:

1. řídicí příkazy
2. standardní příkazy
3. příkazy pro grafiku
4. příkazy pro INPUT/OUTPUT
5. příkazy pro systém 8080

ŘÍDICÍ PŘÍKAZY

NEW	- tento příkaz nuluje všechny uložené příkazové řádky a proměnné
LIST	- tento příkaz provádí výpis uloženého programu na displej od nejbližšího řádku
LIST n	- výpis programu od daného řádku
LIST # kanál;	- výpis programu na zvolený vstupní kanál
RUN	- odstartování programu od nejbližšího řádku, předtím však dojde k nulování proměnných, polí a ukazatele dat
RUN n	- odstartování programu od zvoleného existujícího řádku
GOTO n	- odstartování programu od zvoleného existujícího řádku se zachováním posledního stavu programu /proměnné, pole, atd./
END	- ukončení programu

- CONT - pokračování ve vykonávání programu, který byl přerušem externě od klávesnice STOP nebo programově příkazem **STOP**
- NULL (I) - vložení počtu I nul. znaků po každém řádku. I může být v rozsahu 0 - 72
- LLIST - příkaz používaný k editování programu. Provede listování programu do dialogového řádku. Krokování výpisu se provádí stiskem klávesy **EOL**. Při každém jiném znaku je možno provádět edici řádku, který se ukončí opět jako vstupní řádek do programu.
- SAVE n ["coment "] - uložení programu na pásku kazetového magnetofonu pod pořadovým číslem n, případně je možné vepsat osmiznakový komentář.
- LOAD n - vyhledávání záznamu s pořadovým číslem n z MG pásky, přičemž vycházející záznamy se vypisují do dialogového řádku včetně krátkého akustického doprovodu.
- CHECK n - kontrola záznamu s pořadovým číslem n na kontrolní sumu. V případě chyby hlásí zprávu do dialogového řádku. Tento příkaz je možné použít i na vyhledávání konce všech záznamů pro další zápis.
- ROM n - příkaz, který slouží k přečtení obsahu uživatelské ROM-ky z ROM modulu. Atribut n znamená polohu ROM-0-6. Při nahrání obsahu dané ROM na paměťovou lokalitu 7 000 H se provede i odstartování daného obsahu. Tímto příkazem je možno vykonávat speciální zásahy do interpreteru BASIC - G, doplnit jeho agendu, vykonat rychle uživatelské programy apod. Blíže k jeho modifikaci viz. příkazy pro systém 8080.
- MONIT - opuštění interpreteru a návrat do operačního systému. /z něj je možné opět se vrátit příkazem **JUMP 0000/**

- AUTO - příkaz pro automatické číslování řádků s inkrementem 10.
- AUTO n,m - automatické číslování od řádku n, s inkrementem m. Příkaz AUTO je možné použít i pro zjištění, případně vkládání řádků do programu.
- CLEAR - inicializace proměnných na hodnotu nula
- ? - krátký příkaz sloužící pro vytištění výrazu do dialogového řádku obdobně jako DISP nebo PRINT. Výraz se ruší stiskem libovolné klávesy.

STANDARDNÍ PŘÍKAZY

- BEEP - příkaz pro akustické návěští
- DATA - označení dat nebo řetězcových znaků pro jejich čtení příkazem READ. Jednotlivá data jsou oddělena čárkou. Při interpretaci programu příkaz DATA je ignorován, to znamená, že může být umístěn v libovolné části programu.

Příklad:

10 READ A, B, C, E

30 GOTO 10

50 DATA L, 10, "JACK", . 21

51 DATA -1, 1, "JILL", . 22,

52 DATA X*4+2, SIN (X), DŽ + "WATER",

Poznámka:

Příkaz DATA souvisí s příkazy READ, RESTORE

DEF

Syntax: DEF FNC a (d) = výraz

kde: a je písmenný znak uživatelské proměnné

d je jednoznaková proměnná, která může být zadána i výrazem

Použití: k definování funkcí uživatelem

Příklad:

```
1Ø DEF FNC A (X) = EXP (X ^ 2)
```

```
2Ø Y=Y* FNC A (.1)
```

```
14Ø DEF FNC B (X) = X*P/180
```

```
18Ø/PRINT FNC A (X), FNC B (X)
```

DIM

Syntax: DIM pole (dim 1) , pole (dim 1, dim 2), string pole
(dim 1, dim 2)

Použití: k deklaraci řetězcových a datových polí

Příklad: 1Ø DIM A (5,6), J(20), X(2, 2, 3), BZ (20, 10, 5)

DSAVE

Syntax: DSAVE n; m ["komentář"]

kde: n je výraz, jehož hodnota je celočíselná v rozsahu 0 - 63
a představuje číslo /pořadové/ záznamu.

m je název číselného pole nebo řetězcového pole. Musí být
zapsáno ve formě nejbližšího prvku daného pole. Komentář
je nepovinná část příkazu a umožňuje v rozsahu 8 znaků
označit daný záznam.

Použití: Příkaz DSAVE umožňuje zaznamenat na kazetovou MG pás-
ku obsah všech prvků číselného nebo řetězcového pole,

které je udáno v příkaze. V případě, že příkaz obsahuje pole, které nebylo dimenzováno, interpreter hlásí chybu.

Příklad:

```
10 DIM A$ (10)
200 ? "ZAPNI MG"
110 DSAVE 05; A$ (0) "DATA A$"
115 ? "MG STOP"
```

Poznámka: Záhleví záznamu obsahuje identifikátor "D", který označuje, že se jedná o datový soubor.

DLOAD

Syntax: DLOAD n; m

kde: n, m jsou výrazy obdobné jako u příkazu DSAVE

Při tomto příkazu se kontroluje dimenzovaný prostor pole, které je zaznamenáno na kazetovou pásku. Při nesouhlasu interpreter označuje chybu.

Použití: Příkaz umožňuje naplňování číselného nebo řetězcového pole, /které je dimenzované/ z kazetové MG pásky.

Příklad:

```
10 DIM A (10, 10)
.
.
100 FOR A=5 TO 8
.
120 DLOAD A; A(0, 0)
.
180 NEXT A
```


FOR TO NEXT [STEP]

Syntax: FOR řídicí proměnné = výraz 1 TO výraz 2, STEP výraz 3

kde: výraz 1 - znamená nastavení počáteční hodnoty smyčky /řídicí proměnné/

výraz 2 - znamená konečnou hodnotu smyčky / řídicí proměnné/

výraz 3 - je krokování řídicí proměnné v dané smyčce. Pokud se od tohoto příkazu STEP uustí, krokování je jedna.

Poznámka: Při sestavování více smyček je nutné dbát na správné pořadí ukončení smyček příslušným NEXT příkazem s danou řídicí proměnnou.

Příklad:

```
FOR X
  FOR Y
    FOR Z
      NEXT Z
    NEXT Y
  NEXT X
```

```
FOR X
  GOSUB ← RETURN
NEXT X
```

Každý příkaz musí končit příkazem NEXT.

DISP

Syntax: DISP výraz

Použití: K zobrazení výrazu do dialogového řádku. Jako výraz může být konstanta, řetězcová konstanta, aritmetický

výraz, element z pole. Jeho použití je obdobné jako PRINT.

Příklad: 10 DISP SIN (F), 123*2/5

20 DISP AS (1, 1)

GCLEAR

syntax: GCLEAR

Použití: K vymazání obrazové části

Příklad: 5 GCLEAR

GOSUB

Syntax: GOSUB příkazový řádek

Použití: Přerušování vlastního programu a přechod do podprogramu

BASIC-G. Příkazový řádek musí být dán explicitně, přičemž interpreter ho přepočítává na celočíselnou hodnotu. Příkaz GOSUB může být použit do hloubky až šesti volání.

Příklad:

10 GOSUB 100

GOTO

Syntax: GOTO příkazový řádek

Použití: Pro určení vykonávání následujícího příkazového řádku.

Příkazový řádek musí být zadán explicitně, přičemž interpreter ho přepočítává na integer.

Příklad:

```
190 DATA 10 , -5, -1, 6, 21 -9
```

```
200 READ X
```

```
220 A=SQR (2*X)
```

```
230 PRINT X, A
```

240 GOTO 200 příkaz se bude vykonávat až se načtou všechny hodnoty X.

IF

Syntax: IF výraz THEN příkazový řádek

IF výraz THEN příkaz

Použití: Příkaz pro podmíněné vykonání příkazu THEN na základě pravdivosti výrazu. Pokud je výraz nepravdivý, následuje interpretace následujícího příkazového řádku.

Výraz: Výraz může být aritmetický nebo řetězcový, nebo dva aritmetické, nebo řetězcové výrazy oddělené relačním operátorem.

Příklad:

```
100 IF X+Z=0 THEN 10 : výraz s relačními operátory
```

```
150 IF .01> = SQR (X) GOTO 410
```

```
200 IF A$<>= "YES" GOTO 85
```

15 IF ABS (X) GOTO 410 : výrazy vyhodnocované "nula nebo ne nula".

```
90 IF A+B THEN 27
```

10 IF X+Y =0 THEN LET I=X+Y: pokud X+Y=0, je pravdivý, je vyko-

nán příkaz LET a pokračuje za dalším programovým příkazovým řádkem.

160 IF X THEN IF SIN (X) < .1 GOTO 10:

první IF kontroluje hodnotu X. Pokud je nula, pokračuje za dalším programovým řádkem. Pokud X není nulové, pokračuje za vyhodnocováním příkazu za THEN.

100 IF A>B OR C>D AND E>F THEN STOP

vícenásobný relační výraz: Pokud A>B nebo pokud C>D a E>F, potom se provede příkaz STOP. Pokud ne, pokračuje se příkazem v následujícím řádku.

INPUT

Syntax: INPUT seznam proměnných

kde seznam proměnných je seznam jednotlivých vstupních proměnných, které jsou odděleny čárkou.

Použití: Pro zadávání vstupních dat z klávesnice. Před každým vstupem je obsluha upozorněna krátkým akustickým návěstím. Ukončení vstupu pro danou proměnnou je dano stiskem klávesy EOL.

Příklad:

15 INPUT A\$

20 INPUT A (1, 1), C, A1 (1, 1)

LET

Syntax: LET proměnná = výraz

Použití: Vyhodnocený výraz /jeho hodnota/ je přiřazen proměnné.

Řetězcové výrazy mohou být přiřazeny k řetězcovým proměnným a aritmetické výrazy k aritmetickým proměnným.

Příklad: 10 LET B=C+2.147

pozn.: LET je nepovinné

30 A=A+1

10 LET A\$ = "NOW"

103 B\$=A\$+C\$ + "TIME"

105 E1\$ (3)= E1\$ (2) + A\$

NEXT

Syntax: NEXT řídicí proměnná [řídicí proměnná ...]

Použití: K uzavření smyčky příkazu FOR s danou řídicí proměnnou.

Příklad: 10 FOR I=1 TO 8

12 FOR J=L TO 20 STEP I

13 READ B,I,J

15 NEXT J, I

10 FOR X=1 TO 100

20 FOR Y=1 TO 50

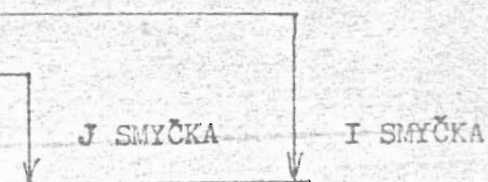
30 LET Z=A (Y)

40 PRINT Z+X

50 IF Z<0 GOTO 70: Ukončení smyčky podmiňujícím

60 NEXT Y příkazem IF

70 NEXT X



ON

Syntax: ON výraz GOSUB seznam příkazových řádků

ON výraz GOTO seznam příkazových řádků

Použití: Na základě hodnoty výrazu / 1 - 255/ se provede vykonání příkazu GOSUB nebo GOTO na příkazový řádek, který se nachází s daným pořadovým číslem v uvedeném seznamu příkazových řádků. Když tyto polohy seznam neobsahuje, bude se interpretovat příkaz na následujícím řádku. Pokud výraz není integer /celočíslná hodnota/, BASIC ∇ G si jej sám převede.

Příklad: 30 ON X-6 GOTO 500, 300, 100

pokud X-6 nemá hodnotu 1, 2, 3 příkaz je ignorován

pokud X-6 má hodnotu 1, provede se skok na příkazový řádek 500.

50 ON INKEY GOSUB 100, 250, 1000

Není-li zatlačena klávesa KEY 1 nebo 2, 3 příkaz je ignorován.

Pokud je zatlačena klávesa např. KEY 2, bude se během jejího stisku vykonávat podprogram na řádku 250.

PAUSE

Syntax: PAUSE [výraz]

kde: výraz může být v oblasti integer o rozsahu 0 - 255.

Použití: Příkaz slouží pro zastavení interpretace programu na určitou dobu uvedenou hodnotou výrazu. Časový inkrement je po 0,1 sek., tedy max. pauza může být 25,5 sek. Neobsahuje-li příkaz výraz, jde vždy o max. časovou pauzu. Přerušeni čekání je možné zrušit klávesou SPACE.

Příklad: 10 PAUSE 20

Program bude zdržen v řádku 10 na dobu 2 sekundy.

PRINT

Syntax: PRINT seznam výrazů

kde: seznam výrazů je seznam číselných proměnných, elementů z pole, řetězcové proměnné, elementy z řetězcového pole, výrazy nebo řetězec ASCII znaků.

Použití: K zobrazení výsledků, hodnot na výstupní zařízení - displej, a to do vymezené části obrazovky.

Print řádek je rozdělen do čtyřech 14 znakových oblastí, které začínají na pozicích 1, 15, 29, 43. Čárka mezi jednotlivými položkami v seznamu výrazů v daném příkazu PRINT indikuje skok k další zóně. Operační systém umožňuje zobrazení pouze 48 znaků na řádek, přičemž další ignoruje a očekává znak na návrat na nový řádek. Tuto okolnost je třeba mít na zřeteli při formátování příkazu PRINT.

Příklad:

10 X = 5

30 PRINT X, (X*2), X^2, "NOW"

1	15	29	43	PRINT pozice
	↓	↓	↓	
5	10	25	NOW	

Pokud je výstupní hodnota delší než délka zóny /14 znaků/, je pokračování PRINT na nejbližší volné zóně. Požadujeme-li PRINT bez uvedených zón, tedy bezprostředně vedle sebe, jednotlivé položky v seznamu výrazů se oddělí středníkem, což způsobí vložení pouze jednoho volného místa při výpisu.

TAB

Syntax: TAB (výraz)

kde: výraz je celočíselná hodnota reprezentující místo začátku výpisu. Vkládá se mezi položky v seznamu výrazů u příkazu PRINT.

Příklad:

```
10 PRINT TAB (5), A, TAB (10), B; D
```

Tzv. spacing

Syntax: SPC (výraz)

kde: výraz je celočíselná hodnota reprezentující počet mezer, které se mají vložit v příkaze PRINT mezi výpisem jednotlivých položek.

READ a DATA

READ

Syntax: READ seznam proměnných

kde: v seznamu proměnných mohou být aritmetické proměnné a prvky pole, řetězcové proměnné nebo prvky z řetězcového pole.

Použití: K přečtení hodnot z datového bloku do daných proměnných nebo polí v příkazu READ. K nastavení datového pole se může použít příkazu RESTORE. Typ proměnné v seznamu READ příkazu musí korespondovat s hodnotami /typ/ v příkazu DATA.

RESTORE

Syntax: RESTORE /příkazový řádek/

kde: příkazový řádek může být zadán explicitně nebo implicitně
a musí být celočíselná hodnota.

Použití: pro nastavení datového bloku k následujícímu příkazu

READ. Když se neuvede příkazový řádek, začíná se prvním
datovým blokem.

Příklad:

```
20 FOR I=1 TO 10
```

```
30 READ B (I)
```

```
40 NEXT I
```

```
50 RESTORE 500*2
```

```
500 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
```

```
505 READ A, B, C
```

```
1000 DATA 10, 11, 12, 13
```

RETURN

Syntax: RETURN

Použití: pro výstup z daného podprogramu. Dalším příkazem, který se bude interpretovat, je příkaz následující za příkazem GOSUB, který způsobil vyvolání tohoto podprogramu.

Příklad:

```
10 X = 5
```

```
11 GOSUB 50
```

```
12 X=7
```

```
13 GOSUB 51
```

```
16 STOP
```



```
50 Y=3*X
51 Z=1.2*EXP (Y)
53 PRINT X,Y
54 RETURN
```

STOP

Syntax: STOP

Použití: k zastavení interpretování programu vypisuje BASIC -G zprávu:

```
*** STOP AT LINE n ***
```

kde: n je číslo řádku STOP příkazu. Systém očekává příkazy z klávesnice.

REM

Syntax: REM text

Použití: k vložení komentářů do programu. Příkazový řádek REM se nevykonává, je pouze vypisován při příkaze LIST.

Příklad:

```
90 REM *** PERSONAL ***
```

PŘÍKAZY PRO GRAFICKÉ ZOBRAZENÍ

Všechny příkazy pro grafické zobrazení se mohou používat v přímém módu a programovém módu. Není možné jich umisťovat více

na jeden programový řádek, ale mohou být umístěny s předcházejícími příkazy jako poslední příkaz na řádku. Protože používají vnitřní promenné X0, X1, X2, X3, Y0, Y1, Y2, Y3, nesmí tyto uživatel používat, protože by mohlo dojít k chybné transformaci proměnných v příkazu SCALE. Zobrazování motivů, grafů se provádí jako negace bodu /bitu/ na dané paměťové lokalitě.

SCALE

Syntax: SCALE x min, x max, y min, y max.

kde: x, y mohou být zadány explicitně nebo implicitně a udávají v jakých hraničních hodnotách se bude zobrazovat.

Použití: Pro nastavení transformace mezi počtem zobrazovaných bodů a hodnotou proměnné, která se má zobrazit.

Příklad:

```
10 SCALE v 1, 1, -1, 1
50 SCALE A, B, 5, 10*PI
```

MOVE

Syntax: MOVE x, y

kde: x, y, jsou výrazy zadané explicitně nebo implicitně a znamenají bod v dané transformaci /SCALE/ se souřadnicemi x, y

Použití: daný příkaz vykonává přesun začátku, "kursoru" do dané pozice

Příklad:

```
10 MOVE 5,5  
20 MOVE A, B
```

AXES

Syntax: AXES x, y

kde: x, y, jsou výrazy zadané explicitně nebo implicitně a znamenají souřadnice průsečíků os.

Použití: daný příkaz umožňuje vykreslit horizontálně čáru bodem y a vertikálně bodem x.

Příklad:

```
10 AXES 0, 0  
50 AXES X, Y
```

PLOT

Syntax: PLOT x, y [, pen]

kde: x, y jsou výrazy, které znamenají souřadnice zobrazovaného bodu. Když výraz pen není nebo obsahuje hodnotu různou od nuly, potom se provede pomocí lineární interpolace propojení zobrazovaného bodu s bodem, který byl zobrazen předcházejícím grafickým příkazem. V případě, že pen má hodnotu 1, uskuteční se vykreslení bodu. Všechny výrazy mohou být zadány explicitně nebo implicitně.

Použití:

10 SCALE -1, 1, -1, 1	: nastavení rozměru obrazovky
20 MOVE 0, 0	: souřadnice kurzoru
30 PLOT 0.5, 0.5	: vykreslení čáry po bod /0.5,0.5/

40 PLOT 0, 0	: smazání vykreslené čáry
50 PLOT 0.5, 0.5, 1	: vykreslení bodu v souřadnici /0.5, 0.5/

BMOVE

Syntax: BMOVE x, y

kde: x, y jsou výrazy zadané explicitně nebo implicitně a udávají polohu kurzoru pro vykreslení hodnoty byte. Výraz x může proto nabýt celočíselné hodnoty v rozsahu 0 - 47 a výraz y v rozsahu 0 - 242.

Použití: příkaz BMOVE nepožaduje nastavení transformačního rozměru příkazem SCALE. Používá se s příkazem BPLOT, kterému slouží jako poloha výpisu /kurzor/.

Příklad:

```
10 BMOVE 15, 15
20 BMOVE X, Y
```

BPLOT

Syntax: BPLOT string, n

kde: string znamená řetězcovou proměnnou nebo prvek z řetězcového pole a n je celočíselná hodnota zadaná explicitně popř. implicitně, určující kolik prvků daného výrazu string se bude zobrazovat vedle sebe. Zobrazování jednotlivých prvků z výrazu string je ve vertikálním směru.

Použití: příkaz BPLOT se používá všude tam, kde je potřeba velmi

rychle vykreslit motiv, který je popsán výrazem string.
Vymazání obrazce se děje opětovným vykonáním příkazu
BPLOT na tomtéž místě.

Příklad: 10 BMOVE 10, 10

20 TŽ = " 7777CCCC"

30 BPLOT TŽ, 2 : vykreslení obrazce o rozměrech
12 x 4 bodů popsaného v řádku 20.

40 BPLOT TŽ, 1 : vykreslení obrazce o rozměrech
6 x 8 bodů.

50 BPLOT TŽ, 3 : vykreslení neúplného obrazce o roz-
měrech 18 x 3, přičemž kurzor se
zastavil na poslední pozici obraz-
ce, protože počet prvků je jen 8.

LABEL

Syntax: LABEL mx, my; seznam výrazů

kde: mx, my jsou výrazy celočíselné hodnoty 1 - 255 a představují,
kolikrát se mají zvětšit /zoom/ v daném směru /x, y/ zobrazované
znaky v seznamu výrazů. Seznam výrazů obsahuje syntax stejnou jako
u příkazu PRINT.

Použití: daný příkaz umožňuje v určeném místě /příkaz MOVE/ zob-
razovat v zadaném násobku bodů znaky, seznam výrazů /kons-
tant/, proměnné prvky z řetězového pole apod., jako je
specifikováno u příkazu PRINT. Zobrazování se děje smě-
rem zdola nahoru.

Příklad:

10 SCALE 0, 255, 0, 242

20 MOVE 0, 200

30 LABEL 2, 2; "TESLA" : vykreslení názvu TESLA od po-
zice bodu /0, 200/ s rozmě-
rem znaků 10 x 14

FILL

Syntax: FILL, mx, my; byte

kde: výrazy mx, my mají podobný význam jako u příkazu LABEL, ale
vztahují se na zobrazování byte, který je zadán explicitně
nebo implicitně a může být pouze v rozsahu 0 - 255 /INTEGER/.

Použití: umožňuje vyplňovat jednoduché obrazce dané jako maska
v hodnotě výrazu byte. Příkaz FILL a příkaz LABEL umož-
ňují při vhodném uspořádání v programu popsat osy vykres-
lené příkazem AXES tak, že příkaz FILL provede požadova-
né delení os a příkaz LABEL je popíše. Dále příkaz FILL
umožňuje vykreslování rovných čar různých typů, čárkove-
ná apod. ve směru x, y. Potom umožňuje provést inverzi
obrazu nad požadovaným místem v SCALE.

Příklad:

10 SCALE 0, 10, 0, 10

20 MOVE 0, 0

30 FILL 255, 242; 1 : inverze obrazu

50 AXES 0, 0

60 FOR I=1 TO 10

7Ø MOVE I, Ø

8Ø FILL 1, 5; 1

vykreslení dílku na ose x

9Ø NEXT I

PŘÍKAZY PRO VSTUP/VÝSTUP I/O

V této části jsou zahrnuty všechny příkazy BASIC - G, které se používají pro vstup a výstup údajů. Všeobecný povel pro výstup údajů je příkaz OUTPUT, pro vstup ENTER. Pokud je možné k počítači připojit více druhů zařízení, je nutné specifikovat správný kanál. Počet kanálů je 8 s adresou Ø - 7. Dále je nutné specifikovat konkrétní výstupní registr. Připojovací místa jsou uvedena v příloze na samostatných místech pro každý kanál.

Povely pro výstup údajů

OUTPUT

Syntax: OUTPUT kr; seznam výrazů

kde: k je číslo kanálu v rozsahu 0 - 7, r je číslo výstupního registru v rozsahu ØØ - 255 a závisí na druhu výstupního kanálu. Seznam výrazů reprezentuje množinu možných výrazů, jak jsou definovány u příkazu PRINT.

DRUHY KANÁLŮ

V následující tabulce jsou zahrnuty kanály, které jsou v interpretaci BASIC - G. Uživatel může definovat rozšíření některých existujících kanálů přemodifikováním příslušných paměťových buněk v tabulce skoků.

TAB. 1

čís. kon.	adresa tab.HEX	Druh kanálu	Adresa exekutivy	Poznámka
0	2088	-	FFFF	neobsazený
1	208A	sériový (V24)	2215	společný s MG
2	208C	-	FFFF	neobsazený
3	208E	-	FFFF	neobsazený
4	2090	paralelní (GPIB)	209F	paralelní
5	2092	-	-	neobsazený
6	2094	-	-	neobsazený
7	2096	IMS-2 (HPIB)	2008	IMS - 2

Například, pokud by bylo třeba umístit na kanál č. 2 řekněme GPIO /programovatelný vstupně-výstupní kanál/, je nutné po hardwareové stránce připojit špičku CS obvodu 8255 na "select" dekodéru I/O viz adresová tabulka a

A7	A6	A5	A4	A3	A2	A1	A0
k dekodéru I/O				A7 = 0; A3=A2=1			

po softwareové stránce vložit na paměťovou buňku 208C nižší hodnotu vektoru exekutivy GPIO, tedy 9F, na následující buňku vyšší hodnotu - 20. Obdobná filosofie platí i pro rozšíření dalších kanálů. V případě, že uživatel potřebuje rozšířit systém o speciální výstupní kanál, je nutno poznat stavbu exekutivy, která je zahrnuta do interpreteru BASIC - G. Jako pomůcka může sloužit příslušná exekutiva, např. pro GPIO.

Uživatele je nutno upozornit, že příslušný kanál je aktivní pouze po jeho inicializování /neplatí pro kanál 7, který je nastaven automaticky vlastními procedurami/. Inicializace se děje programově, vysláním programového kódu na příslušný kanál. Příkaz, který tuto činnost může provést, je CONTROL a bude dále podrobně popsán. Potřebné kódy pro jednotlivé kanály jsou uvedeny v tabulkách. Zjištění stavového slova požadovaného kanálu je možné pro-

vést příkazem STATUS, který bude také dále podrobně popsán.

Jaké stavové hodnoty je možno získat z jednotlivých kanálů je uvedeno v tabulkách u jednotlivých popisů kanálů. Stav příslušného bitu je možno maskovat funkcí BIT.

O jednotlivých kanálech podrobněji:

Výstupní kanál - sériový

Syntax: OUTPUT 1, seznam výrazů

Tento kanál je realizován obvodem USART MHB 8251, který vzhledem k tomu, že je přepínatelný s interfejsem kazetového magnetofonu, má pevně přiřazenou vzorkovací frekvenci 1200 Hz /případně 2400 Hz/, což je nutné mít na zřeteli při volbě přenosové rychlosti. Sériový kanál je programovatelný, což znamená, že před příkazem OUTPUT je nutné zvolit jeho pracovní režim.

Volba pracovního režimu sériového kanálu

se provede vysláním dvou údajů. První reprezentuje nastavení režimu USART a je odlišný pokud jde o asynchronní nebo synchronní přenos dat. Z následujících dvou tabulek je možné sestavit potřebný kód - decimální číslo, které se vypočte jako součet z povinných parametrů.

Asynchronní režim tab. č. 2

Počet Stop Bit			Hodnota parity - ENABLE			Počet znaků				BAUD			
1	2	3	sudá	lichá	uvolněná	5	6	7	8	1x	16x	64x	sync MOD
64	128	192	32	0	16	3	4	8	12	1	2	3	0

Synchronní režim tab. č. 3

ZNAK	SYNC	IDENTIFIKACE SYNC		PARITA			POČET ZNAKŮ			
		SYNDET je INPUT	SYNDET je OUTPUT	sudá	lichá	uvolnění	5	6	7	8
128	Ø	64	Ø	32	Ø	16	Ø	4	8	12

Příklad: Požadujeme naprogramovat sériový kanál jako asynchronní, počet stop bitů má být 1; bez parity, počet znaků 7 bitů a přenosová rychlost 1200 bit/sek./1200 Hz/. Potom obdržíme podle tabulky pro asynchronní režim následující:

$$64 + \emptyset + 8 + 1 = 73$$

toto číslo bude vysláno na USART příkazem CONTROL jako první. Dále USART požaduje řídící instrukci, kterou musíme sestavit tak, aby to vyhovovalo našim požadavkům. Potřebný kód sestavíme obdobně z tabulky č. 4.

Tabulka č. 4

tzv. HUNT MODE JEN PRO SYNC MODE	RESET INTERNAL	"REQUEST" VYSLÁNÍ RTS = log Ø	RESET ERROR	SBRK-BREAK NORMÁLNĚ = Ø 8 = T x D = log Ø	PŘIJÍMANÍ UVOVNIT	TERMINAL "READY" DTR = log Ø	VYSLÁNÍ UVOVNIT
128	64	32	16	8	4	2	1

Stavové slovo ze sériového kanálu je osmibitové a obsahuje tyto údaje:

DSR	SYNDET	FE	OE	PE	TxEMPTY	RxRDY	TxRDY
-----	--------	----	----	----	---------	-------	-------

kde: DSR - DATA - SET READY - indikuje úroveň log. 0, je připravený módem.

SYNDET - pouze pro synchronní režim a indikuje, že USART našel SYNC znak.

FE - pouze pro asynchronní režim a indikuje chybějící STOP bit.

Nuluje se vysláním RESET ERROR v řídicí instrukci USART.

OE - OVERRUN ERROR - znamená, že systém nepřevzal včas data a dochází k přepřínování USARTu. Nuluje se obdobně jako příznak FE.

PE - PARITY ERROR - chyba v paritě. Nuluje se obdobně jako příznak FE.

TxEMPTY - TRANSMITTER EMPTY - používá se k indikaci konce přenosu pro módem.

RxRDY - RECEIVER READY - indikuje, že systém si může převzít přijatá data. Nejčastěji se používá jako "HANDSHAKE".

TxRDY - TRANSMITTER READY - indikuje, že systém může poslat další data na vyslání. Nejčastěji se používá jako "HANDSHAKE".

Pro zjištění stavového slova ze sériového kanálu se použije příkaz STATUS.

Příklad pro OUTPUT:

100 OUTPUT 1 ; AŽ : vyslání obsahu řetězce AŽ na sériový kanál.

Poznámka: Před každou indikací kanálu je nutné ho interně vynulovat.

Výstupní kanál paralelní (GPIO)

Je realizován programovatelným paralelním stykovým obvodem MHB 8255 A.

Syntax: OUTPUT 4r; seznam výrazů

kde: r znamená konkrétní výstupní registr a může být v rozsahu

00 - 07. Kanál GPIO je rozdělen na dvě rovnocenné skupiny.

Skupina 0 - v této skupině se nachází PORT A a část PORT C s vyššími linkami.

Skupina I - sem patří PORT B a druhá dolní část PORT C.

Jednotlivé skupiny a porty A, B, C mohou být programovány v různých režimech (jednoduchý) vstup/ výstup; obousměrný vstup- výstup, s podporou "HANDSHAKE". Výběr jednotlivé skupiny nebo portu je dán hodnotou r, pro kterou platí následující tabulce :

	PORT	režim	r	CONTROL CODE
SKUPINA 0	A	jednoduchý výstup /vstup/	00	128
	C /vyšší/	jednoduchý výstup /vstup/	02	128
	A	výstup /vstup/ s podporou handshake	03	160
	A	obousměrný vstup/výstup + handshake	05	192
SKUPINA I	B	jednoduchý výstup/vstup	01	128
	C /nižší/	jednoduchý výstup /vstup/	02	128
	B	výstup /vstup/ s podporou handshake	04	132

Tab. č. 5

Kódy pro $r = 06$ a 07 nejsou obsazeny a uživatel jim může přiřadit speciální výstupní /nebo vstupní/ exekutivu tak, že uloží příslušný vektor exekutivy na následující paměťové buňky /nejprve se ukládá nižší hodnota/.

VÝSTUP	r	adresa tabulky
	06	20 E6, 20 E7
	07	20 E9, 20 EA

VSTUP	r	adresa tabulky
	06	21 F8, 21 F9
	07	21 FB, 21 FC

Takto je možné např. rozšířit za pomoci obou skupin styk pro BCD výstup apod.

Režim jednotlivých skupin či portů je nutné naprogramovat příkazem CONTROL. Tabulka č. 5 obsahuje kombinaci skupin 0, I když jsou všechny porty A, B, C ve funkci výstupu, ale je možné volit i jiné kombinace. Řídící slovo - kód pro volbu GPIO kanálu je dáno v následující tabulce, která se získá sčítáním jednotlivých požadovaných úkonů kanálu.

Tab. č. 6

NASTAVENÍ ŘÍDÍCÍHO SLOVA	SKUPINA 0							SKUPINA I						
	REŽIM			PORT A		PORT C vyšší bity		REŽIM		PORT B		PORT C nižší bity		
	JEDNODUCHÝ (MOD = 0)	S PODPOROU HANDSHAKE (MOD = 1)	OBOUSMĚRNÝ I/O + HANDSHAKE (MOD = 2)	VSTUP	VÝSTUP	VSTUP	VÝSTUP	JEDNODUCHÝ (MOD = 0)	S PODPOROU (MOD = 1)	VSTUP	VÝSTUP	VSTUP	VÝSTUP	
	128	0	32	64	16	0	8	0	4	2	0	1	0	

Příklad: Je třeba naprogramovat kanál GPIO tak, aby skupina 0 pracovala v režimu výstup s podporou dvoudrátového handshake a ve skupině I port B jako vstupní.

Řídicí slovo = $128 + 32 + 0 + 0 + 2 = 162$

Tento kód bude zapsán příkazem CONTROL do řídicího registru GPIO kanálu.

Kanál GPIO poskytne při režimu /MOD 1 a MOD 2/ stavové slovo, které je možné programově příkazem STATUS a BIT testovat na požadované hodnoty.

Stavová slova pro režim s podporou handshake

Tab. č. 7 - vstupní konfigurace

PC7	PC6						
I/O	I/O	IBF ₀				IBF _I	
7	6	5	4	3	2	1	0

kde: PC 7 je volná linka pro vstup/výstup

PC 6 je volná linka pro vstup/výstup

IBF₀ je indikace, že vstupní registr portu B je naplněn

IBF_I je indikace, že vstupní registr portu A je naplněn

Tab. č. 8 - výstupní konfigurace

		PC 5	PC 4		
$\overline{\text{OBF}}_0$		I/O	I/O	$\overline{\text{OBF}}_I$	

kde: $\overline{\text{OBF}}_0$ indikuje, že výstupní registr portu A je naplněn

PC 5, PC 4 - volné linky pro vstup/výstup

$\overline{\text{OBF}}_I$ indikuje, že výstupní registr portu B je naplněn

Stavové slovo pro obousměrný I/O režim

Tab. 8. 9



kde: \overline{OBF}_0 a IBF_0 mají předcházející význam.

Příklad:

500 OUTPUT 403; "COMPUTER"

Řetězec znaků bude vyslán na kanál GPIO a komunikace bude s podporou handshake.

Výstupní kanál IMS - 2 (HFIB)

Syntax: OUTPUT 7 adr; seznam výrazů

kde: adr. znamená adresu zařízení na IMS-2

Použití: výstup výrazů na zařízení se standardní sběrnicí IMS-2.

Příklad:

100 OUTPUT 701; SIN (X)

Hodnota výrazu SIN(X) bude vyslána na zařízení s adresou 01, které je nastaveno jako "LISTENER".

Poznámka: Tento kanál není třeba inicializovat, ale taktéž je k němu možné přistupovat s příkazy CONTROL, STATUS.

CONTROL

Syntax: CONTROL k, adr; výraz [,výraz ...]

kde: k je číslo kanálu I/O

adr. je adresa registru, do kterého se má zapsat výraz nebo pořadí výrazů.

Výraz může být zadán explicitně nebo implicitně a představuje celočíselnou hodnotu v rozsahu 0 - 255.

V následující tabulce jsou uvedeny všechny možné adresy kanálů, přístupné těmito příkazy.

Tab. č. 10

kanál	registr /PORT/	adr.
1	datový řídící	0 1
4	A B C řídící	0 1 2 3
7	A B C řídící	0 1 2 3*

Použití: Příkaz CONTROL je možno používat k jednoduchému ovládní registrů příslušného kanálu včetně jeho inicializace.

Příklad:

10 CONTROL 4,3; 160

20 OUTPUT 4,3; AŽ

Příklad znázorňuje inicializaci kanálu 4, a to skupiny 0, pro výstup dat s podporou handshake /viz tab. č. 5/. V řádku 20 se vykonává přenos dat /řetězec AŽ/ na kanál 4, skupina 0, port A.

40 CONTROL 1,1; 64 ... interní nulování

50 CONTROL 1,1; 73, 37 ... inicializace

inicializace kanálu sériového - 1 podle tabulek č. 3, 4

100 CONTROL 4,0; 255

Nastavení výstupního registru kanálu 4 port A na hodnotu 255 (FFH).

ENTER

Syntax: ENTER kr; seznam proměnných

kde: k je číslo kanálu

r je konkrétní registr v kanálu

seznam proměnných je množina proměnných obdobně definovaná jako u příkazu INPUT.

"k" a "r" se určují tímto způsobem jako u příkazu OUTPUT.

Pro uživatele je nutné znát, že max. počet přijatých znaků v rámci jedné proměnné může být 80 a tento blok musí být ukončen znakem návrat vozu /CR/ a nový řádek /LF/, v opačném případě může dojít ke zničení celého obsahu paměti.

Použití: pro vstup dat ze zvoleného kanálu do seznamu proměnných.

Poznámka: příkaz ENTER je možné použít pouze po inicializování kanálu!

Příklad:

13 CONTROL 4, 3; 128 + 0 + 16 + 0 + 2 + 0

Nastavení kanálu 4 pro vstup dat do portu A v módu 0 a ve skupině I nastavení portu B jako vstupního v jednoduchém režimu /mód 0/.

20 ENTER 400; X3 vstup dat do X3 přes registr 00 port A

50 ENTER 401,X vstup dat do X přes registr 01 port B

STATUS

Syntax: STATUS k, adr

kde: adr. mají obdobný význam jako u povelu CONTROL /kanál - adre-

se registru v kanálu/.

Použití: pro jednoduché přiřazení hodnoty obsahu zvoleného registru z příslušného kanálu do zvolené proměnné. Příkaz se může použít pouze po inicializování příslušného kanálu příkazem CONTROL.

Příklad:

10 CONTROL 4,3; 146	inicializace kanálu 4
20 A=STATUS 4,0	přiřazení obsahu portu A proměnné A
40 PRINT BIT(STATUS 4,0), 7	tisk hodnoty sedmého bitu portu A kanálu 4

PŘÍKAZY PRO SYSTÉM 8080

V této části se nacházejí příkazy, kterými můžeme komunikovat se systémem 8080.

ROM

Syntax: ROM n

kde: n je celočíselné číslo, výraz, jehož hodnota je v rozsahu 0 - 6.

Použití: Tento příkaz se používá pro zavedení obsahu zvolené paměti ROM / 1 K byte/ na paměťové místo 7000 H. Potom se provede start tohoto programu od adresy 7000 H. Návrat do BASICu na interpretování následujícího příkazu je instrukcí RET /kód C9/. Touto filosofií je

možné provést přemodifikování části interpreteru, jeho doplnění o speciální exekutivy apod. Pomocí příkazu ROM je možné volat i standardní programy, popřípadě obsadit KEY klávesy speciálními údaji. Tento příkaz je možné použít pouze ve verzi BASIC - G, který je uložen v ROM modulu.

Příklad:

10 ROM 1 přenesení obsahu uživatelské ROM z pozice 1
do lokace 7000 a jeho odstartování

USR /funkce/

Syntax: USR /výraz/

kde: výraz je decimální /integer/ hodnota, která může být zadána implicitně nebo explicitně a je v rozsahu 0 - 32767.

Pro plný rozsah adresace se používá negativní hodnota jako rozdíl k číslu 65536.

Použití: uživatelem sestavená exekutiva /podprogram/, který je umístěn v paměti na adrese dané hodnotou výrazu, se vykoná ve strojovém kódu, přičemž parametr z této exekutivy /obsah akumulátoru/ je možné přenést do BASICu, a to do proměnné zadané v příkaze. Pro ukládání uživatelských podprogramů je k dispozici oblast od 7000 - 7E00 H /28672 decimálně/. Tyto binární programy mohou být zavedeny z MG pásky /z operačního systému/ nebo příkazem ROM.

Příklad:

10 ROM 0
20 U=USR (29440)
30 PRINT U

Příklad znázorňuje načítání binárního programu z ROM pozice 0 a jeho odstartování /na adrese 28672 se nachází kód C9 /RET//. Potom se provede interpretování binárního programu na adrese 29000 decimálně, přičemž obsah akumulátoru se přenesení do BASICu a přiřadí se k proměnné U, která se v dalším příkaze zobrazuje.

PEEK /funkce/

Syntax: PEEK /výraz/

kde: výraz má obdobný význam jako u příkazu USR

Použití: načítání obsahu paměťové buňky na adrese dané hodnotou výrazu a přiřazení k proměnné, která se specifikuje v daném příkaze.

Příklad:

1000 X = PEEK /36864 - 65536/

POKE

Syntax: POKE I, J

kde: I a J jsou integer výrazy. I má obdobný význam jako u příkazu USR výraz. J je integer hodnota z oblasti 0 - 255, a může být zadána jako výraz. Tato hodnota se ukládá na paměťové místo adresované I.

Příklad:

10 POKE 28672, 25

OUT

Syntax: OUT I, J

kde: I a J jsou integer výrazy v oblasti hodnot 0 - 255. I je číslo portu a J je decimální hodnota výrazu.

Použití: Výstup dat na adresovaný port.

Příklad:

10 OUT 0, 255

IN /funkce/

Syntax: IN I

kde: I znamená decimální hodnotu /integer/ v rozsahu 0 - 255.

Použití: přečtení a přiřazení obsahu adresovaného vstupního portu k dané proměnné.

Příklad:

10 A=IN 0

WAIT

Syntax: WAIT I, J [,K]

kde: I, J, K jsou integer čísla z oblasti 0 - 255

Použití: K zastavení interpretování programu pokud není splněna podmínka hodnoty čteného portu I s bitem daným maskou hodnotou J. /nenulový obsah/. Pokud je specifikován i výraz K, potom se provede exclusive OR s touto hodnotou.

CODE

Syntax: CODE seznam řetězcových proměnných /může být přímo proměnná nebo prvek z pole.

Použití: používá se interpretování programu ve strojovém /binárním/ kódu, který je přímo zahrnutý v samotném programu BASIC - G. Programy v binárním tvaru se nacházejí jako řetězce znaků ASCII v příslušné řetězové proměnné /nebo prvek z pole/. Takto je možné sestavit jednoduché exekutivy v rozsahu až 40 byte. Posledním znakem v řetězci musí být kód RET /C9/. V řetězci je možné používat i volání jiných podprogramů, dále je možno v rámci podprogramu provádět skoky, ale je nutné mít na paměti, že program se bude vykonávat v paměťové oblasti 7F00 H.

Příklad:

10 A\$ = "3E413234C1CDO085C9"

20 CODE A\$, A\$

vykonání téhož programu
A\$ dvakrát.

ZPRÁVY O CHYBÁCH (ERROR MESSAGES)

BASIC - G podává zprávu o chybě přímo do dialogového řádku, a to ve dvou formách podle toho, zda se jedná o interpretování programového řádku nebo řádku při přímém režimu vykonávání.

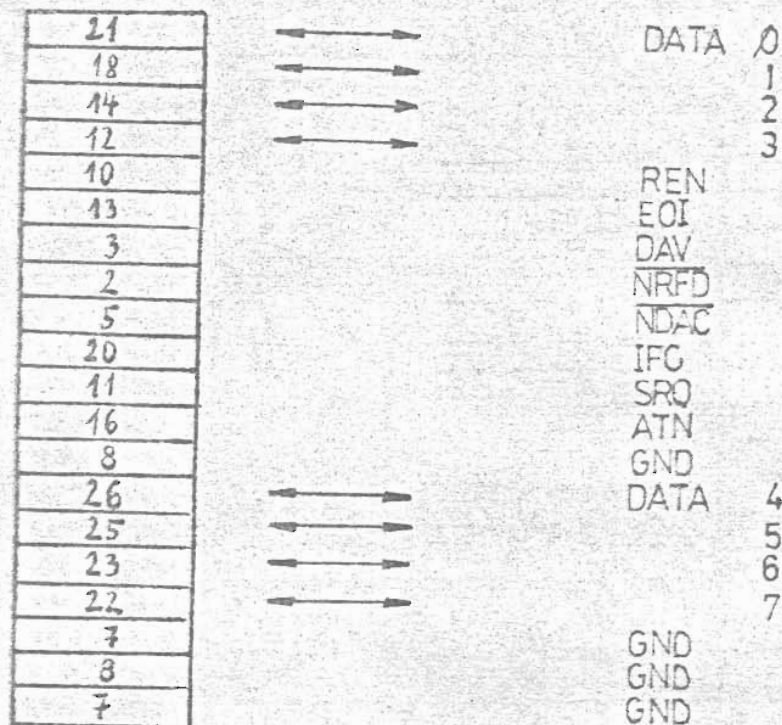
*** ERROR MESSAGE *** přímý mód

*** ERROR MESSAGE IN LINE n*** v běžícím programu

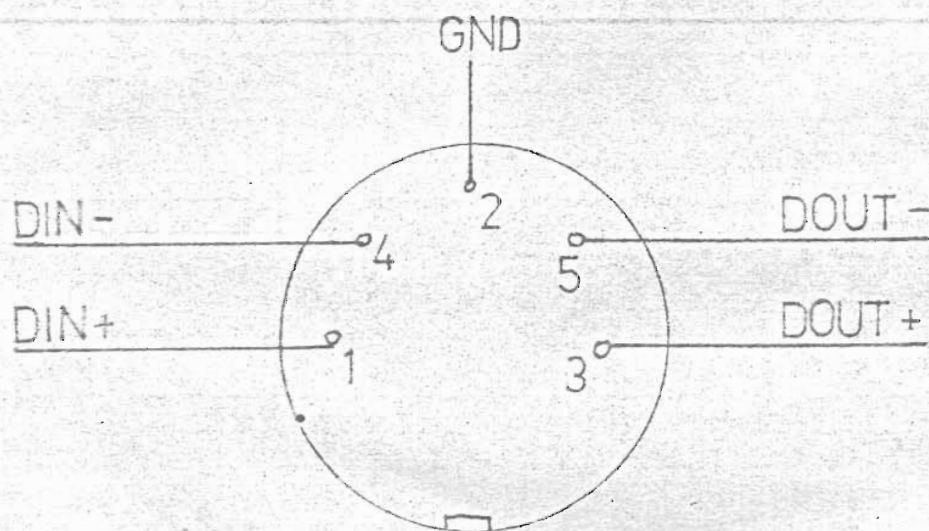
Všechny zprávy jsou v anglických zkratkách a obsahují 10 znaků.

- | | |
|-----------------------|--|
| - <u>SYNTAX ERR</u> | - chybně zadáný příkaz |
| - <u>ENC. PARAM.</u> | - parametr funkce je nesprávný |
| - <u>SUBSCR. RNC.</u> | - výraz subscript je mimo definovaný rozsah např. příkazem DIM |
| - <u>ONLY IN FG</u> | - příkaz je možno použít pouze v programovém módu INP, DEB, INPUT |
| - <u>OVERFLOW</u> | - překročený výpočtový rozsah
FLOATING POINT NUMBER |
| - <u>DV BY ZERO</u> | - dělení nulou |
| - <u>TYPE CONV</u> | - chybný převod např. převod mezi numerickou proměnnou do řetězové proměnné |
| - <u>CAN'T CONT</u> | - chybný povel CONT pro pokračování v interpretování programu. Před tímto povelom byl použit povel, který způsobil změnu v konfiguraci programu. |
| - <u>NO FOR STM</u> | - chybně realizovaná programová smyčka
FOR TO NEXT |
| - <u>ARR ALLOC</u> | - chybná dimenze pole nebo dvakrát dimenzované pole |

- DATA EXHAU
- PG TOO BIG
- STRING LONG
- NO STR. SPC
- NUMB. NONEX
- STR ALGRT
- STOP
- RETURN ERR
- INPUT ERR
- FILE BOUND
- FILE EPROR
- FIELD LCST
- FILE SMALL
- chybějící data pro povel READ
- přeplnění paměťového prostoru.
Program je nutno zkompresovat vynecháním mezer a použitím vícenásobných příkazů na jeden programový řádek.
- délka řetězce přesahuje 255 znaků
- žádné volné místo pro string
- neexistující programový řádek
GOTO, GOSUB THEN
- řetězcový výraz je v tomto případě příliš dlouhý nebo složitý. Je nutné ho rozdělit na dva popřípadě na více jednoduchých příkazů.
- zastavený program
- použitý příkaz RETURN bez korespondujícího příkazu GOSUB.
- chyba při vkládání dat např. nečíselný znak do číselné proměnné.
- překročený max. počet záznamů
- chyba při přenosu dat
- překročený žádaný počet vstupních údajů např. při INPUT
- datový soubor z MG pásky nekoresponduje prostorově s dimenzováním pole.

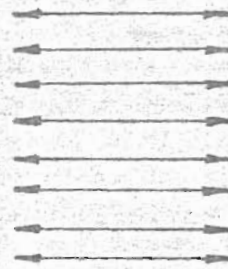
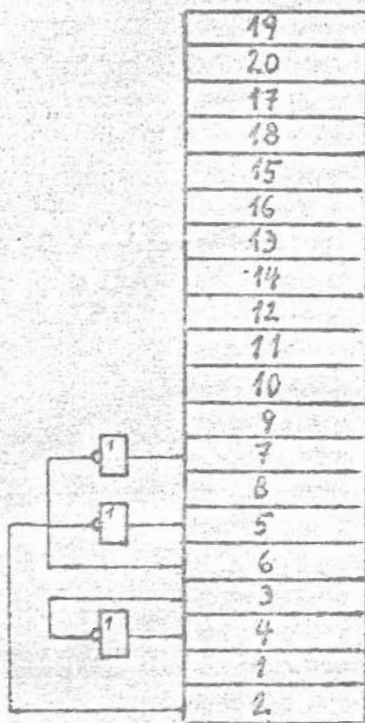


KONEKTOR PRO IMS (HP-IB)



KONEKTOR V 24

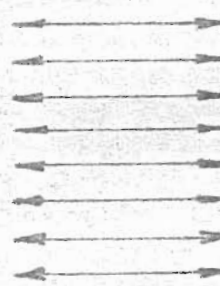
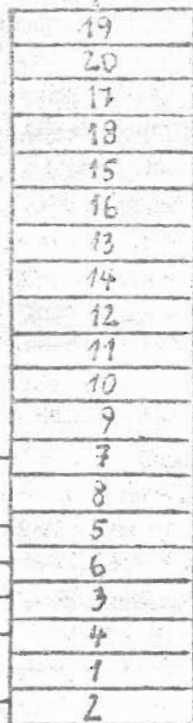
SKUPINA Ø



PC 4
PC 5
PC 7
PC 6

OVLÁDÁNÍ SMĚRU DAT, LOG Ø = DATA VSTUP

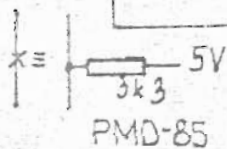
GND



PC 0
PC 1
PC 3
PC 2

OVLÁDÁNÍ SMĚRU DAT, LOG Ø = DATA VSTUP

GND



UŽIVATEL

Vypracoval : Ing. Kišš Roman
k. p. TESLA Piešťany