AYYA publishing

edice Commodore 64/128

G - PASCAL

GRAFICKÝ PROGRAMOVACÍ JAZYK

PASCAL - G POPIS JAZYKA

1 - UVOD

Jazyk PASCAL-G je jeden z "tiny" kompilátorů. Má jistá omezení oproti standardu tak, jak je definován, navíc má funkce, umožňující jednoduchou manipulaci se sprity, jejich animaci, poměrně jednoduché programování v jemné grafice a zjednodušené programování zvukového výstupu.

Kompilátor PASCALU-G překládá zdrojový text do P-kódu a ten pak interpretuje. Rychlost zpracování je nepoměrně vyšší než v BASICU, je ale menší než u přímého strojového jazyka.

Syntax jazyka PASCAL-G není přesně stejná jako u Wirthovy definice PASCALU, odchylky jsou popsány dále.

Výčet příkazů a jejich interpretace není úplná, pro vlastní pokusy a programování by však mohla být dostačující. V další dokumentaci bude postupně doplňována a měla by končit zpracováním jednak debugeru pro P-kód pro zjednodušení ladění programu, jednak přímého kompilátoru P-kódu, který by ještě několikanásobně zrychlil běh programu.

KONSTRUKCE KOMPILATORU

Kompilátor je uložen na adresách $8000 - $BFFF. Zdrojový text je uložen do adresy $4000, čili je možno pro něj použít až 16 KByte. V oblasti $C000 - $CFFF si kompilátor uchovává tabulky identifikátorů, při běhu programu (RUN) je tato část vyhrazena pro zásobník, tedy jsou v této části ukládány proměnné, pole, pracovní proměnné, adresy návratu aj.

Asi 500 byte v poslední části je vyhrazeno pro tabulky manipulace se sprity, uživateli tedy zbývá pro proměnné pouze 3,5 KByte, na což je třeba dávat pozor při koncipování programu.

Za zdrojový text se standardně ukládá přeložený P-kód. Toto přiřazení je možno měnit příkazem na počátku zdrojového textu, a to:

(* %AXXXX *) ... definice počátku ukládání P-kódu, kde XXXX je konstanta (decimální nebo hexadecimální), která určuje počátek ukládání P-kódu. Konstantu je možno měnit v hranicích od $0800 do $4000.

Zdrojový text je ukládán do paměti ve zkráceném tvaru, tzn. že každé rezervované slovo PASCALU je ukládáno jako jednobytová položka (podobně jako v BASICU). Pracovat se zdrojovým textem je možno pouze editorem, který je součástí kompilátoru, nikoliv editorem BASICU.

3 - PRIKAZY PASCALU - G

A) Typy

PASCAL-G používá a je možno definovat pouze dva typy proměnných. INTEGER, celočíselná proměnná se znaménkem v rozsahu 3 byte, tj. -8388608 až +8388607. Druhým typem je CHAR, což je jednobytová proměnná, známá ze standardu. Typy FLOAT, BOOLEAN, RECORD, SET a vyjmenované v PASCALU-G neexistují.

B) Konstanty

Konstanty je možno definovat v tozsahu typu (INTEGER, CHAR), INTEGER jako celočíselnou hodnotu se znaménkem v definičním rozsahu nebo jako hexadecimální konstantu, což se vyznačí $ před hexadecimální cifrou. Další možností je definovat INTEGER jako řetěz tří znaků v uvozovkách.

Př.: I := $CFFF;

Konstanta typu CHAR je znak v uvozovkách, posloupnost více znaků je nepřípustná v přiřazovacích výrazech, i když je příjmová položka pole typu CHAR.

V některých příkazech (viz dále) je možno používat konstantu typu TEXT, což je znakový řetězec uzavřený v uvozovkách.

Př.: WRITE ("tento text");

C) Operátory

Operátory jsou standardní dle definice jazyka, navíc jsou definovány oproti standardu 3 další binární operátory, a to:

SHL ... bitový posun doleva

Syntax: A SHL N; provede bitový posun o N bitů vlevo (včetně znaménka) u proměnné A. Vytlačované bity jsou ztraceny, vpravo jsou vkládány nuly.

SHR ... bitový posun doprava

Syntax: Stejná jako u SHL, binární posuvy jsou opačné.

XOR ... binární operace EXCUSIVE-OR

Syntax: A XOR B; provede operaci EXCUSIVE-OR na proměnných A a B.

D) Funkce

Oproti standardnímu PASCALU jsou definovány v PASCALU-G z funkcí pro operace s čísly pouze jediná, což je dost podstatné omezení při práci s čísly.

ABS ... funkční hodnota je absolutní hodnota argumentu

Syntax: I:= ABS(I); převede proměnnou I na její absolutní hodnotu.

E) Syntax

Odchylek v syntaxi oproti standardu je několik. Nejvýznamnější jsou uvedeny dále, ostatní je možno vysledovat a opravit z upozornění při kompilaci.

a) Příkaz CASE

Tento příkaz musí být ukončen příkazem bez středníku, za kterým následuje END. Jako návěští příkazu CASE lze užít proměnnou, funkci a konstantu v jakémkoli vyjádření.

Př.:
```
CASE operátor OF
     0: X:=X + Y;
     1: X:=X - Y;
     ABS(X): X:= - X
     END
```

b) Aritmetické výrazy

V aritmetickém součtu nebo rozdílu musí být před a za operátorem + nebo - alespoň jedna mezera. Jde zřejmě o chybu kompilátoru.

c) Kompatibilita typu

Kompilátor zásadně nekontroluje kompatibilitu typu. Všechny typy chápe jako INTEGER, takže je možno psát:
Př.: I:= "G"; i když proměnná I je definována jako INTEGER.
Pozn.: V opačné konverzi (A:=I;), kde A je typu CHAR a I INTEGER, se převod provede tak, že do A se uloží první byte zleva!

d) Parametry podprogramu
Předávané parametry podprogramu jsou vždy hodnoty a tudíž nelze modifikovat proměnné ve volajícím programu.
Př.: VAR I: INTEGER;

```
.
PROCEDURE TEST(K);
BEGIN
K:= 7;
END;
```

nezpůsobí změnu hodnoty proměnné, se kterou byla volána procedura TEST.

5 - REZERVOVANA SLOVA

```
129  GET
130  CONST
131  VAR
132  ARRAY
133  OF
134  PROCEDURE
135  FUNCTION
136  BEGIN
137  END
138  OR
139  DIV
140  MOD
141  AND
142* SHL
143* SHR
144  NOT
145* MEM
146  IF
147  THEN
148  ELSE
149  CASE
150  WHILE
151  DO
152  REPEAT
153  UNTIL
154  FOR
155  TO
```

157 WRITE
158 READ
159* CALL
161 CHAR
162* MEMO
163* CURSOR
164* XOR
165* DEFINESPRITE
166* PLOT
167* GETKEY
168* CLEAR
169* ADDRESS
170* WAIT
171 CHR
172* HEX
173* SPRITEFREEZE
174* CLOSE
175 PUT
223* SPRITE
224* POSITIONSPRITE
225* VOICE
226* GRAPHICS
227* SOUND
228* SETCLOCK
229* SCROLL
230* SPRITECOLLIDE
231* GROUNDCOLLIDE
232* CURSORX
233* CURSORY
234* CLOCK
235* PADDLE
236* SPRITEX
237* JOYSTICK
238* SPRITEY
239* RANDOM
240* ENVELOPE
241* SCROLLX
242* SCROLLY
243* SPRITESTATUS
244* MOVESPRITE
245* STOPSPRITE
246* STARTSPRITE
247* ANIMATESPRITE
248 ABS
249* INVALID
250* LOAD
251* SAVE
252* OPEN
253* FREEZESTATUS
254 INTEGER
255 WRITELN

Číslo je hodnota, pod kterou ukládá rezervované slovo editor, * za číslem značí, že rezervované slovo není součástí definice standardního PASCALU.

6 - POPIS IMPLEMENTOVANYCH FUNKCI

A) Operace s pamětí

Dostup na paměťová místa v celé paměti je umožněn definicí dvou polí, MEM a MEMC. Index pole je přímo absolutní adresou v paměti. Pole MEMO je typu CHAR, tedy reprezentuje 1 byte, pole MEM je typu INTEGER, čili reprezentuje tříbytové pole. Přenos se uskutečňuje s pravidly pro ukládání vícebytových čísel, tedy nejprve je byte s nejnižší hodnotou, na vyšší adrese je střední byte a na další adrese je byte nejvyšší.

Př.: I:= MEM ($D011); přenese do proměnné I (INTEGER) postupně byty z adres $D011, $D012, $D013.

A:= MEMO (1); přenese do proměnné A hodnotu byte na adrese 1.

ADDRESS (proměnná)

Funkce Funkční hodnota: adresa proměnné v argumentu

Argument: identifikátor proměnné

Př.:I:= ADDRESS (I); do proměnné I se uloží její vlastní absolutní adesa v paměti.

Pozn.: Kompilátor PASCALU-G rezervuje pole směrem k nižším adresám, takže ADDRESS (P(0)) je větší než ADDRESS(P(2)).

CALL (INTEGER)

Procedura volání podprogramu kdrkoliv v paměti.

Argument: absolutní adresa

Funkce: vyvolání podprogramu na zvolené adrese.

Př.: CALL ($FFE1) vyvolá podprogram pro test stisku tlačítka STOP.

Pozn.: Vzhledem k tomu, že G-PASCAL používá značné množství proměnných v PAGE 0, je potřeba tento příkaz používat velmi obezřetně.

Návrat do PASCALU je přes instrukci RTS.

B) Operace se znakovým stínítkem

CURSOR (Y-souřadnice,X-souřadnice)

Procedura nastavení kurzoru na zvolené souřadnice.

Argument: Y souřadnice pozice kurzoru

X souřadnice pozice kurzoru

Funkce: nastaví kurzor na zadané souřadnice

Y v oboru (1 25)

X v oboru (1..40)

Př.: CURSOR (25,1); nastaví kurzor (a tím i další zápis na stínítko) na poslední řádek.

CURSORX

Funkce Funkční hodnota: aktuální sloupec, na kterém se nachází kurzor

Argument: -

Př.: IX:= CURSORX; do proměnné IX uloží číslo sloupce 1..40), ve kterém se nachází kurzor.

CURSORY

Funkce Funkční hodnota: aktuální řádek, na kterém se nachází kurzor

Argument: -
Př.: IY:= CURSORY; do proměnné IY se uloží číslo řádku, kde se nachází kurzor

SCROLL (XP,YP)
Procedura rolování stínítka dle zvolených parametrů
Argument: XP posun stínítka vodorovně (0..7)
YP posun stínítka svisle
Př.: SCROLL (7,7); provede vložení hodnot 7 a 7 do příslušných registrů VIC.

SCROLLX
Funkce Funkční hodnota: počet bodů, o kolik je posunuto stínítko v ose X.
Argument: -

SCROLLY
Funkce Funkční hodnota: počet řádků, o kolik je posunuto stínítko v ose Y.
Argument: -

C) Příkazy jemné grafiky

Rozložení použité paměti v PASCALU-G umožňuje prakticky použít bez problémů pro jemnou grafiku pouze banku 0 s tím, že HIRES-RAM se rezervuje na adresách $2000 - $3FFF; COLOR-RAM od $0400 a pozice od $0800 do $1FFF lze použít pro sprity (bloky 32 až 127).

GRAPHICS (typ1, hodn1, typ2, hodn2 ....)
Procedura Nastavení registru VIC
Argument : Libovolný počet dvojic. První parametr udává typ, druhý hodnotu. Možné kombinace:

Typ	Možná hodnota	Význam
1	0,1	0 znakový modus bit 5 VIC 17 1 bit map modus
2	0,1	0 vypnutí MULTI bit 4 VIC 22 1 zapnutí MULTI
3	0,1	0 vypíná rozšířený kód 1 zapíná rozšířený kód bit 6 VIC 17
4	0,1	0 volba 38 znaků na řádku 1 volba 40 znaků na řádku bit 3 VIC 17
5	0,1	0 volba 24 řádků bit 3 VIC 22 1 volba 25 řádků
6	0,1	1 mazání stínítka bit 4 VIC 17
7	0..3	volba banky dat (a 16 KByte). Volba je prima, čili 0 je 0-$4000, 1 je $4000-$8000 atd. Bity 0,1 či #2. Stnd. hodnota je 0.
8	0..7	volba pozice generátoru znaků nebo pozice HIRES RAM. VIC $ 18 bity 1..3. Stnd. hodnota je 2. Pozice HIRES RAM se musí volit pouze v hodnotách 0 nebo 4.
9	0..15	volba stránky znaku nebo COLOR RAM pro jemnou grafiku. VIC 17 bit 4..7. Stnd. hodnota je 1.
10	0..15	barva znaku RAM $286
11	0..15	barva rámu VIC 32
12	0..15	barva pozadí #0 VIC 33

```
13   0..15 barva pozadí #1      VIC 34
14   0..15 barva pozadí #2      VIC 35
15   0..15 barva pozadí #3      VIC 36
16   0..15 barva spritu #1      VIC 37
17   0..15 barva spritu #2      VIC 38
18   0..15 začátek stránky znakové video (pro editor) RAM $288
```

Pro zobrazení v jemné grafice, multicolor a rozšířenou barevnou grafiku platí stejná pravidla, jak jsou popsána v manuálu pro BASIC. Jediná výjimka je volba banky dat 16K, která je jinak.

Př.: GRAPHICS(1,1,2,0,3,0,7,0,8,4,11,0,12,1); přepne program do jemné grafiky v normálním módu, HIRES RAM je definovaná na adresách $2000-$3FFF, COLOR RAM zůstává na místě původní znakové RAM, tj. $0400-$0800, barva rámu černá, barva pozadí bílá.

```
CLEAR (PB1,PB2)
  Procedura  mazání HIRES RAM a obsazení COLOR RAM
    Argument: PB1 je horní půlbyte COLOR RAM
              PB2 je dolní půlbyte COLOR RAM
```

Pozn.: Příkaz CLEAR pracuje pouze v BIT-MAP módu a nuluje HIRES RAM dle bitu VIC 24 (1,2,3) a dle banky dat, tedy nemusí nulovat MODULO 8KByte, pokud jsou bity VIC 24 špatně nastaveny.

Př.: CLEAR (0,1); vymaže HIRES RAM a nastaví celou COLOR RAM na hodnotu bílá barva pro nasazený bit, černá barva pro nulový bit v HIRES RAM, čili po tomto příkazu je celé stínítko černé.

```
PLOT (B,X,Y)
  Procedura  nasazení / nulování bodů v HIRES RAM
  Argument: B - barva bodu hodnota 0..3
            ! V normální grafice na parametr B pouze dvě hodnoty
              0 - mazání bodu
              1 - nastavení bodu
            V multicolor grafice mohou být definovány 4 barvy:
              0 - barva pozadí #0
              1 - horní půlbyte (PB1) COLOR RAM
              2 - dolní půlbyte (PB2) COLOR RAM
              3 - barva znakové COLOR RAM (na adrese $D800)
           X - horizontální souřadnice bodu
               v normální grafice v intervalu 0..319
               v multicolor grafice je povoleno 0..159
           Y - vertikální souřadnice bodu 0..199
```

Př.: I:=0 TO 199 DO PLOT (1,159,I); nakreslí uprostřed stínítka svislou čáru.

D) Operace se sprity

RUNTIME MODUL v PASCALU-G podporuje pohyby a animace spritu nezávisle na programu, tzn., že program pouze pohyb nebo animaci spustí a další běh je odvozen z IRQ a programátor s nemusí starat. Komunikace s programem je dána definování sprité, jejich umístěním a inicializací pohybu nebo animace. Ke zjištění aktuálního stavu, ve kterém se sprite nalézá, slouží registr stavu spritu, který je definován pro každý sprite (1 až 8). Obsazení registru nulou v kterémkoli okamžiku způsobí zastavení pohybu nebo animace, jednička pohyb spouští.

Druhá na programu nezávislá část umožňuje reagovat na kolize předem definovaných spritů tím, že pohyby těchto spritů zastaví. K tomu slouží tzv. registr FREEZE, jehož konstrukce je stejná, jako registr kolizí VIC 30. Při kolizi spritů se testuje, zda je zadaná reakce na kolizi obsažením jedničky v patřičném bitu registru FREEZE.

Registr stavu sprite a zmrazení (FREEZE) jsou nezávislé, takže je možné po kolizi a zastavení spritu nastavením 1 do stavového registru spritu jej znovu uvést do pohybu.

```
DEFINESPRITE (NB,I1,I2,I3,....I21);
```

PROCEDURA generuje a uloží na příslušné místo blok pro zobrazení spritu.

ARGUMENT :NB číslo bloku 0...255. Pozn. Pro banku 0, tj. 0-$4000 je povoleno číslo bloku pouze 32...255
I1,I2 ... proměnné nebo konstanty Integer, jež tvoří bitovoumapu spritu. Každá konstanta vyjadřuje jeden řádek (24 bodů) spritů. Pokud není řádků 21, doplní kompilátor na chybějící řádky 0.

Př:
```
DEFINESPRITE (32,$7c00,$3ff80,$7c00,$3800,$3ff80,
$7ffc0,$dff80,$19ff30,$19ff30,$18fe60,$187cc0,
$18ff80,$fe00,$1ef00,$1ef00,$3c780,$3c780,$783c0,
$783c0,$f01e0);
```
Vytvoří blok 32, tzn. v bance 0 na adrese $0800 pro případné požití jako sprite.

```
SPRITE (SP1,TYP1,HODN1,SP2,TYP2,HODN2.....);
```

PROCEDURA přiřazení čísla bloku ke spritu a definice spritu

ARGUMENT : Libovolný počet trojic parametrů s tímto významem: SP - číslo spritu 1...8; TYP a HODNota s tímto významem:

Typ	Hodnota	Význam
1	0-15	barva spritu
2	0-255	číslo bloku pro sprite
3	0,1	1 sprite multicolor
4	0,1	1 expand v ose X
5	0,1	1 expand v ose Y
6	0,1	1 priorita spritu na obr.
7	0,1	1 aktivuje zobrazení spritu

Př: SPRITE (1,1,7,1,2,32,1,4,1,1,5,1,1,7,1); aktivizuje sprite 1 z bloku 32, ve žluté barvě a expandovaný v obou osách.

```
POSITIONSPRITE (SP,X,Y)
```

PROCEDURA Umístění spritu na obrazovce

ARGUMENTY : SP - číslo spritu 1...8
X - horizontální souřadnice 1...512
Y - vertikální souřadnice 1...256

Př: POSITIONSPRITE (1,100,100); umístí sprite 1 na zvolené souřadnice.

```
SPRITEX (SP)
SPRITEY (SP)
```

Funkce funkční hodnota - okamžitá souřadnice X (Y) spritu na obrazovce.

```
    ARGUMENT      : SP - číslo spritu
    Př:  IF  SPRITEX(1)  "je  větší"  100  THEN  SPRITE(1,7,0);
         Jestliže  souřadnice  X spritu číslo 1 překročí  číslo
         100, pak bude sprite zrušen.

SPRITECOLLIDE
    FUNKCE      Funkční hodnota je registr kolizí spritů VIC 30
    ARGUMENT    -
    Př: IF (SPRITECOLLIDE AND 1) THEN
          BEGIN    VOICE(1,1,256,1,3,1,1,4,0,1,5,13,1,6,8,1,7,
          13,   1,14,1);   SOUND(2,160,5,15,6,1);VOICE(1,8,1);
          VOICE(1,8,0);END;
    Při kolizi spritu č. 1 s jiným spritem se ozve detonace.

GROUNDCOLLIDE
    FUNKCE        Funkční hodnota je registr kolize spritu a
         pozadí VIC 31
    ARGUMENT    -
    Př:  IF  GROUNDCOLLIDE  AND 1 = 1 THEN  STOPSPRITE(1);  při
         kolizi spritu 1 s pozadím bude pohyb spritu zastaven.

MOVESPRITE (SP,X,Y,VX.VY,K)
    PROCEDURA        Inicializace nezávilého pohybu  spritu  po
         obrazovce.
    ARGUMENTY      SP - číslo spritu 1...8
                   X - horiz. souřadnice počátku pohybu 1...256
                   Y - vertik. -"-  1...256
                   VX - horiz,. rychlost
                   VY - vertik. rychlost
                   K - doba pohybu sprite
              Obě  rychlosti mohou být i záporné a udávají se  v
         počtech přesuny o jeden bod za 4 sekundy.
              Po  vyčerpání  K  časových  jednotek  se  sprite
         zastaví s jeho status se nastaví na nulu.
         Pozn:  Pohyb spritu je odvozen z časovacího IRQ (16ms)
         a způsob zadávání je poněkud těžkopádný.
              Pro  výpočet  parametrů  přesunu lze použít  tyto
         vztahy:  máme-li sprite v pozici o souřadnicích X1, Y1
         a chceme  jej přesunout na souřadnice X2, Y2 za čas T:
              VX = 4*(X2-X1)/T
              VY = 4*(Y2-Y1)/T
              K = T*64
    Př:  MOVESPRITE  (1,1,100,350,0,256);  sprite  číslo  1  se
         přesune přes obrazovku zleva doprava za 4 sekundy.

    ANIMATESPRITE (SP,K,BL1,BL2,...,BL16);
        PROCEDURA       oživení spritu postupnou výměnou čísla blo
            pro sprite.
        ARGUMENTY      SP - číslo sprite 1...8
                       K - počet čas. jed. (0.016s), které mají pr
                           běhnout mezi dvěma změnami bloku spritu
                       BL1.... -čísla bloku pro oživení (max. 16).
        Př: ANIMATESPRITE (1,16,32,33,34,35); oživení spritu
                 Číslo  1  probíhá cyklickou výměnou bloků 32,  3
            34 a 35. Po bloku 35 následuje opět 32.
```

```
   Př: SOUND (1,0,4,15);
       VOICE (1,1,7493,1,4,8,1,5,0,1,6,15,1,7,6,1,8,1,1,11,1);
         Zazní komorní A flétnovým zvukem.

STOPSPRITE (SP);
    PROCEDURA      nulování registru stavu spritu.
    ARGUMENT       SP - číslo sprite 1...8
    Př:  STOPSPRITE  (1);  sprite, který se  pohybuje  nebo  je
         oživen, se zastaví.

TSPRITE (SP);
    PROCEDURA      nastaví registr spritu na 1
    ARGUMENT       SP - číslo sprite 1...8
    Př:  STARTSPRITE  (1);  sprite, který byl  zastaven  pomocí
         STOP se opět rozbíhá.

SPRITESTATUS (SP);
    FUNKCE         Funkční hodnota registr stavu spritu (0,1).
    ARGUMENT       SP - číslo sprite 1...8
    Př:  REPEAT UNTIL SPRITESTATUS (1); program sojí, dokud  se
         nedokončí pohyb spritu 1.

SPRITEFREEZE (SP);
    PROCEDURA      žádost o zastavení pohybu nebo výměny sprite
         při kolizi.
    ARGUMENT       sprity na jejichž kolizi se má reagovat mají
         v odpovídajících bitech 1, ostatní 0.
    Př:  SPRITEFREEZE (17); způsobí zastavení spritu 1 (1) nebo
         5  (16) při kolizi s jiným spritem nebo navzájem.  Při
         kolizi se registr freeze vynuluje.

FREEZESTATUS
    FUNKCE         Funkční hodnota je registr  VIC  30  logicky
         znásobený  registrem  FREEZE,  čili ve výsledku  je  1
         toho  spritu, jehož kolize byla vyžadována v  registru
         freeze a skutečně nastala.
    ARGUMENTY      -
```

E) Ovládání zvukového výstupu

```
CE (HL1,TYP1,HODN1,HL2,TYP2....)
    PROCEDURA      Nastavení registru SID
    ARGUMENTY      HL - číslo hlasu
                   TYP - typ parametru 1...15
                   HODN - hodnota parametru
```

TYP	HODN	Význam
1	0-65535	Nastavení frekv. oscilátoru
2	0-2047	Nastavení pulsů
3	0, 1	1 filtr zapnut 0 filtr vypnut
4	0-15	attack
5	0-15	decay
6	0-15	sustain
7	0-15	release
8	0, 1	1 spuštění cyklu ADS 0 spuštění release

9	0, 1	1 zapnuta synchronizace osc.
10	0, 1	1 zapnuta kruhová modulace
11	0, 1	1 zapnuta trojúhel. křivka
12	0, 1	1 zapnuta pilová křivka
13	0, 1	1 zapnuta pulsní křivka
14	0, 1	1 zapnuta šumová křivka
15	0, 1	ovládání bitu test

SOUND (TYP1,HODN1,TYP2,HODN2....)
PROCEDURA řízení SID
ARGUMENT libovolný počet dvojic s obsahem:

TYP	HODN	Význam
1	libov.	Nuluje všechny registry
2	0-2047	Nastavení resonanční frekv. filtru
3	0-65535	čekání v délce hodnoty (jedn. 0,01s)
4	0-15	Hlasitost
5	0-15	Rezonance (SID 23, bity 7 až 4)
6	0, 1	1 dolní propust zapnuta 0 dolní propust vypnuta
7	0, 1	1 pásmová propust zapnuta 0 pásmová propust vypnuta
8	0, 1	1 horní propust zapnuta 0 horní propust vypnuta
9	0, 1	1 vypnutí oscilátoru 3 z výstupu (bit 7 registru SID 24)

Př: SOUND (1,0,4,15); Nejprve vynuluje všechny registry a pak nastaví hlasitost na maximum.

F) Funkce vstupu a výstupu

Vstup a výstup z programu je zajištěn v zásadě procedurami READ a WRITE. Standardně je vstup z klávesnice, výstup na obrazovku. Dále uvedenými příkazy je možno s jistým omezením přesměrovat vstup i výstup na jiné zařízení. Kromě toho lze části paměti ukládat a natahovat dopaměti pomocí příkazu LOAD a SAVE.

WRITE (SEZNAM)
WRITELN (SEZNAM)
PROCEDURA Výstup textové informace
ARGUMENTY Seznam jednoduchých proměnných, funkcí, literálů a výrazů.
Pozn: Procedura WRITELN se liší od procedury WRITE pouze tím, že po výstupu seznamu vystoupí ještě znak CR (13), což způsobí skok kurzoru na začátek nového řádku. Procedura WRITELN nemusí mít žádný argument.
Př: WRITELN; vystoupí pouze znak CR
Př: WRITE ("A=",A); Z programu vystoupí nejprve text v uvozovkách a poté hodnota proměnné A.
Bez zadání konverse jsou výstupy proměnných vždy vnumerickém, dekadickém tvaru bez vedoucích nul. Pokud proměnné následují v seznamu bezprostředně za sebou, nejsou odděleny na výstupu žádnou mezerou.

```
HEX(I)
    FUNKCE          Funkční hodnota je znakové hexadecimální
                    zobrazení proměnné I. Funkcilze použít pouze jako
                    argument procedury WRITE, resp. WRITELN.
    ARGUMENT        I - jednoduchá proměnná lib. typu, výraz
                    funkce. Ta se konvertuje do šesti hexadecimálních
                    znaků.
    Př: I:=$800F; WRITE(HEX(I)); na obr. se vypíše 00800F

CHR(A)
    FUNKCE          Funkční hodnotou je znaková reprezentace
                    nejnižšího bytu proměnné A.
    ARGUMENT        A - jednoduchá proměnná, výraz, funkce.
    Př: VAR A; ARRAY(8) OF CHAR;
        FOR I:=0 TO 7 DO WRITE (CHR(A(I)));WRITELN;
             Na obrazovku se vypíše 8 prvků pole "A" jako
        znakový řetězec. Po výpisu skočí kurzor na počátek
        nového řádku.

READ (SEZNAM)
    PROCEDURA       Načtení textové informace
    ARGUMENT        Seznam jednoduchých proměnných nebo pole
                    typu CHAR
                    Pozn: Vstup je obdobný BASIC příkazu INPUT. Znaky
                    jsou akceptovány až po nalezení znaku CR. V případě,
                    že v seznamu je více proměnných, musí být CR za
                    každou proměnnou. Při načítání do proměnné typu
                    INTEGER se kontroluje numeričnost a rozsah. Při chybě
                    se vloží do proměnné 0.
                    Při načítání do pole typu CHAR jsou jednotlivé
                    znaky ukládány v pořadí rostoucího indexu. Vkládání
                    je možno ukončit znakem CR (RETURN). zi tento znak se
                    uloží do pole a zbytek zůstane nezměněn. Maximální
                    délka je 196 znaků.
    Př:     READ (I,A); Vstup z klávesnice. Nejprve se vkládá
            číslo do proměnné I, RETURN ukončí vkládání do I a
            následující znak se uloží do A. RETURN opět uončí
            vkládání. Po vložení informace do proměnné se kurzor
            přesune na další řádek.

OPEN (LOG,DEV,CHAN,NAZEV)
    PROCEDURA       Otevření souboru pro vstup nebo výstup
    ARGUMENTY       LOG - logické číslo souboru
                    DEV - číslo zařízení
                              1 - datassette
                              2 - RS232
                              8 - disc drive
                    CHAN - číslo kanálu 2...127
                    NAZEV - název souboru v uvozovkách
            Pozn:  Procedura OPEN pracuje identicky jako v BASICu
    Př: OPEN(3,8,3,"TEXTY,S,W); otevření výstup. soub. "TEXTY"
        OPEN(15,8,15,"S:BLOCK"); zrušení souboru "BLOCK"

CLOSE (LOG,DEV,CHAN,NAZEV)
    PROCEDURA       Uzavření souboru
    ARGUMENTY       jako proc. OPEN
            Pozn: Pracuje jako v BASICu.
```

```
GET (LOG)
    PROCEDURA       Přesměrování vstupu na jiné zařízení
    ARGUMENT        LOG - logické číslo souboru
                        - 0 vstup z klávesnice
         Pozn:      Soubor (LOG) musí být před příkazem GET
         otevřen příkazem OPEN. Po přesměrování je READ
         prováděno z tohoto souboru (viz popis READ). Soubor
         musí být znakový a sekvenční.
    Př: VAR VETA:ARRAY(100) OF CHAR;
        .
        .
        OPEN(3,8,8,"VSTUPY");
        GET(3); READ (VETA);
        IF INVALID = 0 THEN
        REPEAT
          I:=0; REPEAT
                  WRITE (CHR (VETA(I)));
                UNTIL VET(I) "je různé" 13;
        READ (VETA);
        UNTIL INVALID = 0
             Program přečte soubor vstupy a po jednotlivých
         záznamech jej vypíše na obrazovku.

PUT (LOG)
    PROCEDURA       Přesměrování vstupu na jiné zařízení
    ARGUMENT        LOG - logické číslo souboru
                        -  0  vrací vstup na klávesnici a výstup
                           na obrazovku
    Pozn: Soubor (LOG) musí být před příkazem PUT otevřen. Po
         přesměrování je WRITE a WRITELN prováděno do
         definovaného souboru.

INVALID
    FUNKCE          Funkční hodnota: obsah stavové proměnné (v
         BASICu označované ST)
    ARGUMENT        -
    Pozn: Význam jednotlivých bitů proměnné ST viz manuál
         BASICu. Po vyvolání funkce INVALID se proměnná ST
         vynuluje.

SAVE (DEV,ADRFROM,ADRTO,NAZEV);
    PROCEDURA       Uložení definované části paměti na vnější
         médium
    ARGUMENTY       DEV - číslo zařízení
                    ADRFROM - počáteční adresa, odkud se
                              ukládání započne
                    ADRTO - koncová adresa + 1
                    NAZEV - název souboru v uvozovkách
    Př: VAR XX: ARRAY (100) OF CHAR;
        SAVE(8,ADDRESS(XX(49)),ADDRESS(XX(0))+1,"ARRAY");
             Příkaz způsobí uložení padesáti prvků zpočátku
         pole XX.
```

```
GET (LOG)
    PROCEDURA       Přesměrování vstupu na jiné zařízení
    ARGUMENT        LOG - logické číslo souboru
                        - 0 vstup z klávesnice
         Pozn:      Soubor (LOG) musí být před příkazem GET
         otevřen příkazem OPEN. Po přesměrování je READ
         prováděno z tohoto souboru (viz popis READ). Soubor
         musí být znakový a sekvenční.
    Př: VAR VETA:ARRAY(100) OF CHAR;
        .
        .
        OPEN(3,8,8,"VSTUPY");
        GET(3); READ (VETA);
        IF INVALID = 0 THEN
        REPEAT
          I:=0; REPEAT
                  WRITE (CHR (VETA(I)));
                UNTIL VETA(I) "je různé" 13;
        READ (VETA);
        UNTIL INVALID = 0
            Program přečte soubor vstupy a po jednotlivých
         záznamech jej vypíše na obrazovku.

PUT (LOG)
    PROCEDURA       Přesměrování vstupu na jiné zařízení
    ARGUMENT        LOG - logické číslo souboru
                        - 0 vrací vstup na klávesnici a výstup
                          na obrazovku
    Pozn: Soubor (LOG) musí být před příkazem PUT otevřen. Po
         přesměrování je WRITE a WRITELN prováděno do
         definovaného souboru.

INVALID
    FUNKCE          Funkční hodnota: obsah stavové proměnné (v
         BASICu označované ST)
    ARGUMENT        -
    Pozn: Význam jednotlivých bitů proměnné ST viz manuál
         BASICu. Po vyvolání funkce INVALID se proměnná ST
         vynuluje.

SAVE (DEV,ADRFROM,ADRTO,NAZEV);
    PROCEDURA       Uložení definované části paměti na vnější
         médium
    ARGUMENTY       DEV - číslo zařízení
                    ADRFROM - počáteční adresa, odkud se
                              ukládání započne
                    ADRTO - koncová adresa + 1
                    NAZEV - název souboru v uvozovkách
    Př: VAR XX: ARRAY (100) OF CHAR;
        SAVE(8,ADDRESS(XX(49)),ADDRESS(XX(0))+1,"ARRAY");
            Příkaz způsobí uložení padesáti prvků zpočátku
         pole XX.

LOAD (DEV,ADR,VER,NAZEV)
    PROCEDURA       Načtení bloku z vnějšího média do paměti
```

ARGUMENTY DEV - číslo zařízení
ADR - adresa, odkud se začne ukládat
VER - 0 LOAD
1 VERIFY
NAZEV - název souboru v uvozovkách

Př. LOAD (8,ADDRESS(XX(99)),0,"ARRAY");
V předchozím příkladu uloženou informaci načte program do stejného pole XX, ale pozice indexu jsou 50 - 99.

GETKEY
FUNKCE Funkční hodnota je kód stisknuté klávesy nebo 0, pokud nebyla žádná klávesa stisknuta
ARGUMENT -
Př: REPEAT A:=GETKEY UNTIL A "je různé" 0;
Program čeká ve smyčce na stisk klávesy. Hodnotu stisknuté klávesy uloží do A. Totéž je možno naprogramovat lépe:
READ(A);

6) Různé funkce

WAIT (N)
PROCEDURA čekání na specifikovaný řádek obrazovky
ARGUMENT N - číslo řádku registr VIC 11 a 12
pro C64 evropská verze 0...311
- " - americká verze 0..261

Př. WAIT (245); SCROLL(SCROLLX,(SCROLLY+1)MOD7);
Program vyčká do zatemění obrazovky a pak roluje obrazovku o jeden řádek.

Pozn. Interpretr PASCALu není dostatečně rychlý, aby bylo možno používat tohoto příkazu k přepínání grafiky a textu.

SETCLOCK(HH,MM,SS,D)
PROCEDURA Nastavení vnitřních hodin na reálný čas
ARGUMENT HH - hodiny
MM - minuty
SS - sekundy
D - desetiny

Př. SETCLOCK (12,0,0,0);
Nastavení vnitřních hodin na poledne. Pro hodiny platí stejné omezení jako pro BASIC proměnnou TI$.

CLOCK (I)
FUNKCE Funkční hodnota BYTE vnitřních hodin dle argumentu
ARGUMENT 1 - funkční hodnota jsou desetiny sekundy
2 - - " - sekundy
3 - - " - minuty
4 - - " - hodiny

Př. WRITELN("stiskni tlačítko");
T1:=CLOCK(1)*10+CLOCK(0);
REPEAT A:=GETKEY UNTIL A "je různé" 0;
WRITELN("reakční čas =",CLOCK(1)*10+CLOCK(0)-T1,"des. sekundy");
Po výpisu program sejme čas a čeká na stisk tlačítka klávesnice. Poté sejme čas znovu a vypíše reakční čas.

PADDLE (N)

FUNKCE — Funkční hodnota je obsah převodníku SID 25, 26

ARGUMENT — N - číslo registru 1, 2

JOYSTICK (N)

FUNKCE — Funkční hodnota je sejmutý stav joysticku. Dle jednotlivých bitů:

0 - vpřed
1 - vzad
2 - vlevo
3 - vpravo
4 - FIRE

ARGUMENT — N - číslo joysticku 1 nebo 2

Př. IF JOYSTICK (2) AND $10 THEN WRITELN ("FIRE");

RANDOM

FUNKCE — Funkční hodnota je obsah registru SID 27

ARGUMENT —

Pozn. Tato funkce dává nenulovou hodnotu pouze v případě, že je spuštěn třetí oscilátor šumovým signálem s blokovaným signálem.

Př. VOICE(3,1,1000,3,14,1,3,8,1); SOUND(9,1);
I:=RANDOM + RANOM*256;
Do proměnné I se uloží náhodné číslo 0 - 65535

ENVELOPE

FUNKCE — Funkční hodnota je obsah registru SID 28

ARGUMENT —

Pozn. Tato funkce vrací číslo, jež udává velikost obálky ADSR, generované třetím oscilátorem. Ten musí být spuštěn s blokovaným výstupem.

7) Poznámky k editoru, kompilaci a j.

Všechna činnost v PASCALU 6 je řízena pomocí MENU a je dostatečně srozumitelná. Pozornost je třeba věnovat editoru, ve kterém není možno dělat opravy přímo na obrazovce. Navíc se řádky interně nečíslují, takže každé vložení řádku způsobí změnu čísel všech řádků, které následují za vloženým.

V některých případech je třeba u číselnýchargumentů použít hranaté závorky.

Editor má vlastní malý HELP soubor a po pár pokusech je práce s ním poměrně snadná, i když málo elegantní.

Každý výpis a to jak v módu EDIT, tak ve všech ostatních lze pozastavit stisknutím mezerníku, opětným stisknutím se výpis na obrazovce znovu pouští. To platí i pro RUN vlastního programu.

PASCAL 6 má možnost trasování a debugger, který je orientován výhradně na přeložení P-code, nikoli zdrojový text. Bez znalosti P-code je trasování i debugger nepoužitelný. Trasování a debugger (který vypisuje prováděný P-code spolu s obsahem stacku, tracer vypisuje pouze prováděný P-code) lze spustit v kterémkoli příkazu READ/WRITE klávesami:

CTRL + T — spouští tracer
CTRL + D — spouští debugger
CTRL + N — vypíná zapnutý ladící prostředek