Vysoká škola strojní a textilní v Liberci
Fakulta textilní

Martin Moravčík

Grafický výstup na DIGIGRAF 1208 A

Diplomová práce

1994

VYSOKÁ ŠKOLA STROJNÍ A TEXTILNÍ
V LIBERCI
FAKULTA TEXTILNÍ

Obor číslo 31-20-8
ASŘ ve spotřebním průmyslu

Grafický výstup na DIGIGRAF 1208A

Martin Moravčík

KTK ASŘ FT

Vedoucí práce : Ing. Jan Tišer Katedra technické kybernetiky
Konzultant : Ing. Jan Tišer Katedra technické kybernetiky

Rozsah práce

Počet stran : 67
Počet příloh : 5

Datum odevzdání : 30.12. 1994

VYSOKÁ ŠKOLA STROJNÍ A TEXTILNÍ V LIBERCI

Fakulta textilní

Katedra technické kybernetiky Školní rok: 1993/94

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro Martina MORAVČÍKA

obor 31-20-8 Automatizované systémy řízení ve spotřebním průmyslu

Vedoucí katedry Vám ve smyslu zákona č. 172/1990 Sb. o vysokých školách určuje tuto diplomovou práci:

Název tématu:

Grafický výstup na digigraf 1208 A

Zásady pro vypracování:

1) Seznamte se s funkcí digigrafu 1208 A - 3,5 G
2) Vytvořte unitu pro podporu grafického výstupu v TP či BP na digigraf 1208 A
3) Navrhněte a sestavte program pro interpretaci příkazů jazyka HPGL pro digigraf 1208 A

Místopřísežné prohlášení :

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury pod vedením vedoucího a konzultanta.

V Liberci dne 30. 12. 1994

.....

Úvodem bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Janu Tišerovi za cenné rady, připomínky a čas který mi věnoval. Dále pak kolektivu techniků a konstruktérů ze s.p.ELITRON Liberec za hardwarové vyřešení komunikace DIGIGRAFU s PC..

OBSAH

1. ÚVOD

Automatické zobrazovací zařízení (plotr) DIGIGRAF 1208A, s velice slušnými technickými parametry, vytváří trvalý záznam vektorovým způsobem kresby. Proto je velice výhodné používat ho jako grafický výstup systému CAD (Computer Aided Design). Konstruktéři DIGIGRAFU nebrali při konstrukci ohled na celosvětový standard řízení plotrů pomocí grafického jazyka HPGL (Hewlett-Packard Graphics Language), a vybavili ho pouze jednoduchým souborem grafických instrukcí, který ovšem na druhou stranu dostačuje pro vykreslení jakékoliv grafické aplikace.

Proto se nabízí možnost připojení DIGIGRAFU k běžným typům počítačů standardu IBM PC, vytvoření patřičného programového vybavení pro komunikaci s PC a pro podporu grafického výstupu z běžných grafických systémů (AUTOCAD, COREL DRAW, ORCAD).

Následující text je zpracován jako manuál k obsluze DIGIGRAFU, k využití navržené programové jednotky pro podporu grafického výstupu z Turbo Pascalu (popř. Borland Pascalu) na DIGIGRAF a k využití interpretru grafického jazyka HPGL na DIGIGRAF.

2. DIGIGRAF 1208A

2.1. Technický popis

[1]

DIGIGRAF 1208 A je automatický kreslící stůl (plotr) pro vykreslování výkresů, obrázků, grafů, pomocí kreslící hlavy a zhotovování grafických podkladů rytím, pikýrováním, řezání světelným paprskem pomocí technologických záznamových hlav. Mezi technologické hlavy patří hlava rycí, pikýrovací, řezací a světelná.

Zařízení je používáno všude tam, kde je grafická informace požadována jako výsledek výpočtu v počítačích. Potřebné vstupní informace v abecedně číselném tvaru jsou získávány buď ze standardního kanálu pro vstup - výstup JSEP nebo v autonomním režimu z magnetickopáskové jednotky 1/2" nebo z fotoelektrického snímače děrné pásky.

DIGIGRAF má podobu rovinného kreslícího stolu s pevnou vodorovnou kreslící plochou. Obsahuje mechanickou a elektronickou část. Mechanická část je částí nosnou, zajišťující pohyb v obou osách pohonnými jednotkami. Pohonná jednotka je tvořena servopohonem se zabudovaným systémem odměřování polohy. Ovládání DIGIGRAFU je zajišťováno z panelu operátora, umístěného u okraje kreslící plochy. Na panelu operátora jsou umístěny všechny ovládací prvky nutné pro provoz kreslícího zařízení.

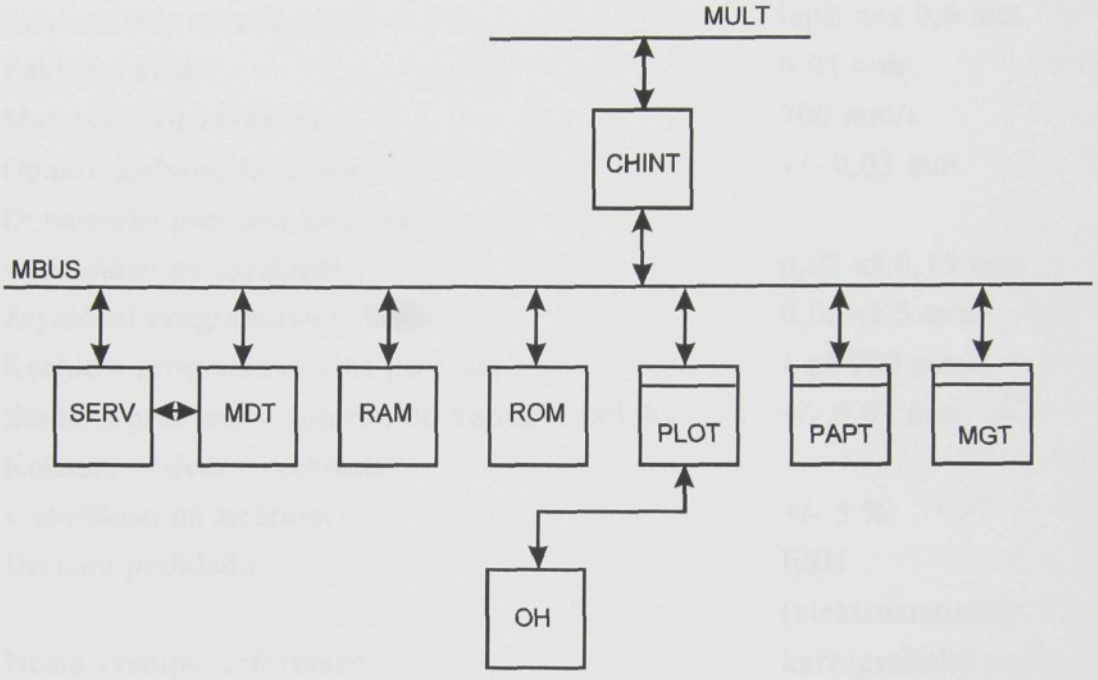
Pohybové ústrojí zajišťuje pohyb ve dvou vzájemně kolmých osách X a Y. Pohyb podélný ve směru osy X je zajišťován dvěma servopohony. Pohyb příčný ve směru osy Y je zajišťován unašečem (vozíkem), s jedním servopohonem. Kreslení je realizováno pomocí pisátek umístěných v kreslící hlavě. Kreslící hlava je připevněna k unašeči (vozíku) a je zaměnitelná za libovolnou technologickou hlavu.

Veškerá elektronika je uložena v podstavci stolu. Podstavec je základní nosnou částí, na které je umístěn rám a jsou zabudovány pevné a výklopné rámy elektroniky. Ve výklopném rámu, umístěném pod panelem operátora, jsou soustředěny veškeré elektronické obvody. Ve zbývajících rámech jsou umístěny zdroje 3,5G, servozesilovač a ovladač pohonů. V základním

provedení je DIGIGRAF vybaven kreslící dvoupisátkovou hlavou pro kuličkové pero STAEDTLER a FISHER a trubičkové pero STAEDTLER.

DIGIGRAF je vybaven elektrostatickým upínáním kreslících podkladů. Blokové schéma DIGIGRAFU 1208 A je na obr.1:

Obr.1.



MDT - rychlý šestnáctibitový speciální procesor
SERV - servisní modul procesorového systému
RAM - pamětí RAM
ROM - paměti ROM
PLOT - řídící elektronika a mechanika kresliče
OH - ovládání hlavy
PAPT - řídící elektronika rychlého snímače děrné pásky
MGT - řídící elektronika magnetopáskové jednotky
CHINT - řídící jednotka pro spojení na multiplexní kanál počítače JSEP
MULT - multiplexní kanál počítačů JSEP
MBUS - sběrnice DGF kreslící části

2.1.1. Technické parametry

Pracovní pole pro kreslení	1189 x 841 mm
Max. velikost nosiče výstupní informace	1400 x 1000 mm
Pracovní poloha kreslení	vodorovná
Rovinnost kreslící desky	lepší než 0,6 mm
Základní krok	0,01 mm
Max.rychlost záznamu	700 mm/s
Opakovatelnost záznamu	+/- 0,03 mm
Dynamická přesnost kreslení v závislosti na zrychlení	0,03 až 0,15 mm
Zrychlení programově volitelné	0,05 až 5 m/s^2
Rychlost programovatelná po 1mm/s	1 až 700 mm/s
Statická přesnost v dojezdu do koncové polohy	+/- 0,03 mm
Kolísání výsledné rychlosti v závislosti na směrnici	+/- 5 %
Upínání podkladů	ESU (elektrostatické)
Nosič výstupní informace	kartografický papír kladívkový papír (ČSN 50 2750) astralonová fólie.

2.2. Obsluha DIGIGRAFU 1208A

Uvedení zařízení do provozu :

Přivedení síťového napětí do bloku ovladače zdrojů se provede zamykatelným vypínačem, který je na panelu operátora označen **SÍŤ**, současně se zapnutím se rozsvítí indikační prvek nad vypínačem. Vlastní zapnutí napájecích zdrojů se provede tlačítkem **ZDROJE**. Po správném zapnutí zdrojů proběhne nulování zařízení, a na displeji svítí kombinace 0000.

Součástí zařízení je vestavěný procesor, který pracuje podle řídícího programu uloženého v paměti typu RAM. Zavedení řídícího programu se musí proto provádět vždy znova při každém zapnutí zařízení.

Následující text je omezen na spolupráci DIGIGRAFU s rychlým snímačem děrné pásky.

Proces zavádění řídícího programu se zahájí stiskem tlačítka **START** po zapnutí zařízení. Stiskem tohoto tlačítka se rozsvítí na displeji kombinace EC07. V tomto stavu je možné pracovat s klávesnicí panelu operátora a provádět následující operace :

1) Číst obsah paměti nebo registrů periferií.
2) Zapisovat do paměti nebo registrů periferií.
3) Nahrávat řídící program do paměti RAM.

Řídící program se nahraje stiskem klávesy **LOAD.** V průběhu zavádění programu svítí na displeji 00F4, což udává zdroj importu řídícího programu (v našem případě snímač děrné pásky). Po zavedení řídícího programu se provede inicializace zařízení (kreslící hlava se přesune do pravého horního rohu kreslící plochy, tj. do hardwarového počátku) a zařízení přejde do stavu ručního řízení. Tento stav je indikován číslem 9999 na displeji, dále svítí tlačítko RYCHLE a všechna tlačítka směrového kříže. Nyní máme možnost pojíždět unašečem ručně pomocí směrového kříže a zadávat příkazy z panelu operátora, které shrnuje tabulka č.1.

Tabulka č.1

Kód	Operandy	Význam
0		Přesun nástroje do bodu 0,0 kreslící plochy (do levého spodního rohu).
1	X,Y	Přesun nástroje do bodu X,Y. Souřadnice se zadávají v mm.
2	X,Y	Posun nástroje o X setin mm ve směru +X a Y setin mm ve směru +Y.
9		Ukončení ručního řízení (přechod do stavu PROVOZ).
10		Zvednutí nástroje.
11		Spuštění nástroje.

12		Blokování spouštění nástroje.
13		Odblokování spouštění nástroje.
14	N	Volba pisátka N.
15	P1,P2,P3,P4	Výměna pořadí pisátek. Při volbě pisátka n z programu se bude kreslit pisátkem Pn.
40	V,A	Volba rychlosti a zrychlení při přejezdech.
41	V,A	Volba rychlosti a zrychlení při kreslení.
50		Zjištění polohy nástroje. Po navolení kódu příkazu se na displeji zobrazí celá část souřadnice X (mm) a dále (po stisknutí tlačítka směrového kříže) desetinná X (setiny mm), celá část Y, desetinná Y.
60	N	Volba měřítka kresby N:1. Příkaz platí do konce výkresu.
90	N	Zařízení čte instrukce, zpracovává, nevykresluje, po zpracování N instrukcí přejde do stavu RUČNÍHO ŘÍZENÍ.
91	N	Obdobný příkaz jako 90 s tím, že po zpracování N instrukcí přejde zařízení do normálního stavu PROVOZ.
7777		Přechod do stavu „Zavádění řídícího programu“.
8888		Stop procesoru.
9999		Ukončení kresleného programu (simulování instrukce „Konec programu“.

Zavádění souboru grafických instrukcí : viz kap. 3.2. Obsluha komunikace.

3. Komunikace PC s DIGIGRAFEM

3.1. Technické zabezpečení komunikace

DIGIGRAF má možnost zavádění vstupních informací z těchto zdrojů :

a) z osmistopé děrné pásky pomocí snímače FS 1503 - EC 6122
b) z magnetopáskové jednotky CM 5300.01, PT 305
c) z kanálů počítače JSEP
d) z minipočítače (u komplexů EC 7941, EC 7942).

Tato technika je ale bohužel v dnešní době několikanásobně překonaná a proto se vyskytla potřeba řídit DIGIGRAF z běžných počítačů standardu IBM PC.

Nabízí se řešení využitím komunikace s PC přes kanál počítače JSEP nebo z minipočítače. Toto řešení ale předpokládá zásah do elektronického vybavení samotného DIGIGRAFU. Proto se technici ze s.p. ELITRON Liberec, kterým bylo hardwarové zabezpečení přenosu dat z PC na DIGIGRAF uloženo, rozhodli pro nejjednodušší a tím pádem i nejlevnější řešení - simulovat snímač děrné pásky.

Elektronický převodník připojený na paralelní port počítače PC (CENTRONICS) upravuje a unifikuje signály z PC na signály stejné jako vysílá snímač děrné pásky. K tomu je potřeba ještě jednoduché programové vybavení, které pošle soubor grafických instrukcí pro DIGIGRAF na paralelní port PC. Tento program se nazývá PLESK.EXE a je tedy stejně jako převodník dílem techniků ze s.p.ELITRON Liberec.

3.2. Obsluha komunikace

a) Po uvedení DIGIGRAFU do provozu (viz kap. 5.2.1. Uvedení zařízení do provozu) se nahraje do DIGIGRAFU řídící program (MIMI.MMM) následujícím příkazem z povelového řádku DOSu : **PLESK MIMI.MMM**. Po zavedení řídícího programu se provede inicializace

zařízení (kreslící hlava se přesune do pravého horního rohu kreslící plochy, tj. do hardwarového počátku) a zařízení přejde do stavu ručního řízení. Tento stav je indikován číslem 9999 na displeji, dále svítí tlačítko RYCHLE a všechna tlačítka směrového kříže. Nyní máme možnost pojíždět unašečem ručně pomocí směrového kříže. Rychlost pojíždění můžeme volit pomocí tlačítek RYCHLE, POMALU a KROK.

b) Pokud máme papír upnutý (tlačítko **ESU**) na desce stolu, nastavíme tímto způsobem počátek souřadného systému tj. najedeme unašečem do bodu, který bude po zavedení programu grafických instrukcí považován za počátek souřadného systému (X=0,Y=0).

Pozn.: Je vhodné označit si polohu kreslícího listu např. pomocí malých proužků samolepících etiket.

c) Poté navolíme na numerické klávesnici číslo **9** (příkaz k přechodu ze stavu RUČNÍHO ŘÍZENÍ do stavu PROVOZ) a stiskneme LOAD. Nyní můžeme vyslat z PC soubor grafických instrukcí (*.DGF) pomocí příkazu **PLESK *JMÉNO*.DGF** z povelového řádku DOSu. DIGIGRAF začne po úspěšném zavedení souboru vykreslovat grafickou aplikaci a na displeji se objevují čísla právě vykonávaných grafických instrukcí. Po vykreslení aplikace zůstane unašeč v místě, kde skončilo vykreslení posledního grafického elementu a přepne se do ručního řízení. Pokud chceme kresbu opakovat nebo vykreslit jinou grafickou aplikaci, postupujeme takto :

Odjedeme unašečem mimo rozsah kreslícího listu (pomocí směrového kříže), vyměníme papír a postupujeme dále podle bodu c).

Pozn.: Je důležité aby byl nový papír umístěn přesně na to samé místo jako papír původní (aby se shodovaly počátky souřadných systémů), jelikož DIGIGRAF považuje počátek souřadného systému nastaveného v bodě b) za platný až do nové inicializace zařízení.

4. Programová jednotka DIGIFRAF

Procedury a funkce implementované v programové jednotce **DIGIGRAF** zajišťují plné využití grafických možností plotru DIGIGRAF 1208 A. Pomocí této jednotky je možno kreslit na plotru grafické elementy a nastavovat jejich atributy stejně tak, jak to dovoluje programová jednotka GRAPH v Turbo Pascalu nebo Borland Pascalu na obrazovku monitoru.

Pro lepší zapamatování a snadnější práci s procedurami a funkcemi jednotky DIGIGRAF byly zachovány stejné názvy procedur a funkcí jako jsou názvy ve výše uvedené programové jednotce GRAPH, s tím rozdílem, že názvům předchází dvojice znaků **D_** . Tedy je-li procedura zajišťující vykreslení kružnice v jednotce GRAPH nazvána **CIRCLE**, pak v jednotce DIGIGRAF má tato procedura název **D_Circle**. Samozřejmě jednotka obsahuje i další procedury přispívající k maximální efektivitě využití plotru.

Všechny grafické procedury pracují v milimetrech nebo stupních a všechny atributy mají předdeklarované konstanty.

Pro snadnější odladění programu v TP či BP využívajícího programovou jednotku DIGIGRAF je umožněna kontrola kresby na monitoru, kterou je po odladění programu možno potlačit (na pomalejších počítačích zpomaluje vlastní generování souboru grafických instrukcí pro plotr). Při kontrole na monitoru dochází k některým omezením ve vykreslování na monitoru, na které je upozorněno v podrobném popisu procedur a funkcí.

4.1. Nastavení a zjištění parametrů a způsobu kresby

Procedury:

InitDGF	- Inicializace režimu tvorby souboru grafických instrukcí pro plotr
InitMonitor	- Inicializace kontroly kresby na monitoru
CloseDGF	- Ukončí režim tvorby souboru grafických instrukcí pro plotr

CloseMonitor - Ukončí kontrolu kresby na monitoru

D_SetPaperFormat - Nastaví formát papíru (A5 - A0)
D_SetPaperOrient - Nastaví orientaci papíru
D_SetSpeed - Nastaví rychlost pohybu pisátka
D_SetAccel - Nastaví zrychlení pohybu pisátka

Funkce :

D_GetPaperFormat - Vrací aktuální formát papíru
D_GetPaperOrient - Vrací aktuální orientaci papíru
D_GetMaxX - Vrací maximální možnou horizontální souřadnici pro zvolený formát papíru
D_GetMaxY - Vrací maximální možnou vertikální souřadnici pro zvolený formát papíru
D_GetSpeedPD - Vrací aktuální rychlost pohybu spuštěného pisátka
D_GetSpeedPU - Vrací aktuální rychlost pohybu zvednutého pisátka
D_GetAccelPD - Vrací aktuální zrychlení pohybu spuštěného pisátka
D_GetAccelPU - Vrací aktuální zrychlení pohybu zvednutého pisátka

4.2. Nastavení a zjištění atributů grafických prvků

Procedury:

D_SetColor - Nastaví barvu pera
D_SetPen - Nastaví barvu pera
D_SetLineStyle - Nastaví typ čáry
D_SetLineSettings - Definuje uživatelský typ čáry
D_SetTextStyle - Nastaví atributy textu

D_SetTextJustify - Nastaví způsob zarovnávání textu

Funkce :

D_GetColor - Vrací aktuální číslo pisátka
D_GetPen - Vrací aktuální číslo pisátka
D_GetLineSettings - Vrací aktuální typ čáry
D_GetMaxColor - Vrací maximální hodnotu čísla pisátka
D_GetMaxPen - Vrací maximální hodnotu čísla pisátka

4.3. Kreslení grafických prvků

Procedury :

D_Arc - Nakreslí kruhový oblouk od počátečního úhlu ke koncovému úhlu s definovaným středem a poloměrem

D_Circle - Nakreslí kružnici s definovaným středem a poloměrem

D_Ellipse - Nakreslí eliptický oblouk od počátečního úhlu ke koncovému s definovaným středem a délkami poloos

D_Line - Nakreslí čáru, jako spojnici dvou bodů

D_LineTo - Nakreslí čáru z aktuální pozice pisátka do

D_LineRel - Nakreslí čáru z aktuální pozice pisátka do bodu, který je definován relativními souřadnicemi

D_MoveTo - Přesune pisátko do definovaného bodu

D_MoveRel - Přesune pisátko do bodu definovaného relativními souřadnicemi

D_OutText - Vykreslí textový řetězec na aktuální pozici pisátka

D_OutTextXY - Vykreslí textový řetězec na definovanou souřadnici

D_PutPixel - Vykreslí bod na definovanou souřadnici

D_Rectangle	- Vykreslí pravoúhelník definovaný dvěma body
D_XAxis	- Vykreslí horizontální souřadnou osu s definovaným počátkem a diferencí značek
D_YAxis	- Vykreslí vertikální souřadnou osu s definovaným počátkem a diferencí značek

4.4. Podrobný popis procedur a funkcí programové jednotky DIGIGRAF

4.4.1. Procedury a funkce pro nastavení a zjištění parametrů a způsobu kresby

procedura **InitDGF**

Funkce : Inicializuje režim tvorby souboru grafických instrukcí pro plotr DIGIGRAF 1208 A.

Deklarace : InitDGF;

Poznámka : Inicializací režimu tvorby souboru grafických instrukcí pro DIGIGRAF se v našem případě rozumí vytvoření prázdného textového souboru s názvem, který byl specifikován jako parametr procedury *D_SetPath* s příponou DGF. V případě že název nebyl touto procedurou specifikován bude název NONAME.DGF Do tohoto souboru budou během vykonávání následujících grafických procedur postupně ukládány grafické instrukce pro řízení plotru DIGIGRAF 1208 A.

V případě, že tedy požadujeme výstup na plotr, je nutné, aby každý program začínal touto inicializační procedurou.

Je výhodné zařadit tuto inicializaci do programu až po jeho konečném odladění, jelikož její absence značně urychlí vykreslování složitějších grafických sestav na obrazovce monitoru (pokud je samozřejmě kontrola na monitoru nastavena).

Omezení : Na konci programu (v případě použití InitDGF) a při každém dalším volání této procedury je nutné uzavřít výstupní grafický soubor *.DGF pomocí procedury *CloseDGF* a změnit jméno souboru *.DGF pomocí procedury *D_SetPath*. Při nedodržení tohoto postupu dojde k přepsání nebo k chybnému uzavření výstupního souboru *.DGF.

Viz také : *CloseDGF, D_SetPath, InitMonitor*

Příklad :

```
uses DIGIGRAF;
var I : Integer;
begin
    D_SetPath(C:\PLOT\KRUHY.DGF);
    InitDGF;
    for I:=1 to 10 do
    D_Circle(100,100,I*10);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem KRUHY.DGF, který na plotru nakreslí 10 soustředných kružnic o středu /100,100/ a poloměrech od 10 do 100 mm.)

procedura **InitMonitor**

Funkce : Inicializace kontroly kresby na monitoru.

Deklarace : *InitMonitor;*

Poznámka : Procedura inicializuje grafický režim a otevře na obrazovce monitoru okno přizpůsobené aktuální velikosti a orientaci papíru. Při vykonávání programu se veškeré grafické elementy určené k exportu pro plotr zobrazují v patřičném měřítku také na monitoru.

Viz také : CloseMonitor

Příklad :

```
uses DIGIGRAF;
var I : Integer;
begin
    D_SetPath(C:\PLOT\KRUHY.DGF);
    InitDGF;
    InitMonitor;
    for I:=1 to 10 do
    D_Circle(100,100,I*10);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem KRUHY.DGF, který na plotru nakreslí 10 soustředných kružnic o středu /100,100/ a poloměrech od 10 do 100 mm a zároveň provede tuto kresbu na monitoru)

procedura **CloseDGF**

Funkce : Procedura ukončí režim tvorby výstupního textového souboru s grafickými instrukcemi pro plotr.

Deklarace : CloseDGF;

Poznámka : Procedura zapíše na konec souboru *.DGF instrukci (1A), která signalizuje plotru ukončení kresby a přechod do ručního režimu.

Omezení : Tuto instrukci je potřeba zařadit vždy na konec programu, v případě že jí předchází procedura InitDGF. Nedodržením tohoto pravidla dochází k chybnému ukončení souboru grafických instrukcí a tím pádem k nekorektnímu ukončení kresby ,které se projeví tzv."zamrznutím" plotru, který je potřeba znovu inicializovat.

Viz také : *InitDGF;*

Příklad : Viz příklad v *InitDGF;*

procedura **CloseMonitor**

Funkce : Procedura provede uzavření režimu kontroly na monitoru.

Deklarace : CloseMonitor;

Poznámka : Procedura uzavře grafický režim obrazovky.

Viz také : *InitMonitor;*

Příklad : Viz příklad v *InitMonitor.*

procedura **D_SetPaperFormat**

Funkce : Procedura nastaví formát papíru.

Deklarace : D_SetPaperFormat (PF : Byte);

Poznámka : Procedura nastaví zvolený formát papíru jako aktuální pro následující grafické operace. Formát PF je možné nastavit pomocí jedné z následujících předdefinovaných konstant (viz tabulka č.2)

Tabulka č.2

Formát	Hodnota	X max.(mm)	Y max.(mm)
A0	0	841	1189
A1	1	594	841
A2	2	420	594
A3	3	297	420
A4	4	210	297
A5	5	148	210

Maximální rozměry X max. a Y max. platí pro orientaci papíru : Vertical (viz procedura pro nastavení orientace papíru D_SetPaperOrient).

Viz také : *D_GetPaperFormat, D_GetMaxX, D_GetMaxY*

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPaperFormat(A4);
    D_SetPath('C:\DRAW\UHLOPRICKA.DGF');
    InitDGF;
    InitMonitor;
    D_Line(0,0,D_GetMaxX,D_GetMaxY);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem UHLOPRICKA.DGF, který na plotru nakreslí přímku z bodu /0,0/ do bodu definovaného

maximálními rozměry papíru a zároveň provede tuto kresbu na monitoru)

procedura **D_SetPaperOrient**

Funkce : Nastaví orientaci papíru.

Deklarace : D_SetPaperOrient(PO : Byte);

Poznámka : Procedura nastaví zvolenou orientaci papíru jako aktuální orientaci pro následující grafické operace. Orientaci PO je možno zadat pomocí jedné z následujících konstant (viz tabulka č.3).

Tabulka č.3

Orientace	Vertical	Horizontal
Hodnota	1	2

Viz také : *D_GetPaperOrient, D_SetPaperFormat, D_GetPaperFormat, D_GetMaxX, D_GetMaxY*

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPaperFormat(A4);
    D_SetPaperOrient(Horizontal);
    D_SetPath('C:\DRAW\UHLOPRICKA.DGF');
    InitDGF;
    InitMonitor;
    D_Line(0,0,D_GetMaxX,D_GetMaxY);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem UHLOPRICKA.DGF, který na plotru nakreslí přímku z bodu /0,0/ do bodu definovaného maximálními rozměry horizontálně orientovaného papíru formátu A4 a zároveň provede tuto kresbu na monitoru)

procedura **D_SetSpeed**

Funkce : Procedura nastaví rychlost pohybu pisátka.

Deklarace : D_SetSpeed(V0,V1 : Word);

Poznámka : Parametry V0 a V1 určují maximální přípustnou rychlost kreslicí hlavy při zvednutém (V0) nebo spuštěném (V1) pisátku. Rychlosti jsou dány v mm/s. Před zpracováním první instrukce kreslícího programu jsou všem parametrům přiřazeny automaticky počáteční hodnoty.
Počáteční a maximální hodnoty V0 a V1 jsou v tabulce č.4.

Tabulka č.4

Počáteční hodnoty		**Maximální hodnoty**	
V0	**V1**	**$V0_{max}$**	**$V1_{max}$**
400	200	700	700

Všechny hodnoty jsou v mm/s.
Hodnotu V1 je třeba volit s ohledem na kvalitu pisátka. Na přesnost kresby má velmi malý vliv.

Viz také : *D_SetAccel, D_GetSpeedPU, D_GetSpeedPD, D_GetAccelPU, D_GetAccelPD.*

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\CTVEREC.DGF');
```

```
    InitDGF;
    InitMonitor;
    D_SetSpeed(700,700);
    D_Rectangle(0,0,100,100);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem CTVEREC.DGF, který na plotru vykreslí čtverec o stranách 100 mm, a to maximální rychlostí pisátka).

procedura **D_SetAccel**

Funkce : Procedura nastaví zrychlení pohybu pisátka.

Deklarace : D_SetAccel(A0,A1 : Word);

Poznámka : Parametry A0 a A1 určují maximální přípustné zrychlení kreslící hlavy při zvednutém (A0) nebo spuštěném (A1) pisátku. Před zpracováním první instrukce kreslícího programu jsou všem parametrům přiřazeny automaticky počáteční hodnoty. Parametry se zadávají ve stupních 1 až 10, kterým odpovídají zrychlení v mm/s^2 (viz tabulka č.5).

Tabulka č.5

Stupeň zrychlení	Zrychlení v mm/s^2
1	50
2	200
3	450
4	800
5	1250
6	1800
7	2450

8	3200
9	4050
10	5000

Počáteční a maximální hodnoty A0 a A1 jsou v tabulce č.6.

Tabulka č.6

Počáteční hodnoty		**Maximální hodnoty**	
A0	**A1**	**$A0_{max}$**	**$A1_{max}$**
6	4	10	10

Hodnota A1 podstatně ovlivňuje kvalitu kresby. Pro práce s kreslící hlavou se doporučují tyto hodnoty :
Velmi přesná práce - stupeň 3.
Práce s běžnými nároky na přesnost (např. výkresová dokumentace) - stupeň 4.
Práce s malými nároky na přesnost - stupeň 5 až 7.
Hodnota A0 prakticky neovlivňuje přesnost kresby.

Zadané hodnoty není třeba během práce změnit. Plotr automaticky omezuje rychlost kreslení kruhových oblouků a obecných křivek, vytvářených pomocí na sebe navazujících úseček tak, aby radiální zrychlení nepřekročilo zadanou hodnotu.

Viz také : *D_SetSpeed, D_GetSpeedPU, D_GetSpeedPD, D_GetAccelPU, D_GetAccelPD.*

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\CTVEREC.DGF');
    InitDGF;
    InitMonitor;
    D_SetSpeed(700,700);  ( Max.rychlost )
    D_SetAccel(10,10);    ( Max.zrychlení )
    D_Rectangle(0,0,100,100);
```

```
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem CTVEREC.DGF, který na plotru vykreslí čtverec o stranách 100 mm, a to maximální rychlostí a maximálním zrychlením pisátka).

funkce **D_GetPaperFormat**

Funkce : Vrací aktuální formát papíru.

Deklarace : D_GetPaperFormat : Byte;

Poznámka : Funkce vrací hodnotu aktuálního formátu kreslícího listu podle tabulky č.7 :

Tabulka č.7

Formát	A0	A1	A2	A3	A4	A5
Hodnota	0	1	2	3	4	5

Viz také : D_SetPaperFormat, D_SetPaperOrient, D_GetPaperOrient

Příklad :

```
uses DIGIGRAF;
var F : String;
begin
    D_SetPath('C:\DRAW\FORMAT.DGF');
    InitDGF;
    Str(D_GetPaperFormat,F);
    D_OutText(F);
    CloseDGF;
end;
```

,(Program vytvoří soubor s názvem FORMAT.DGF, který na plotru vypíše aktuální formát papíru)

funkce **D_GetPaperOrient**

Funkce : Vrací aktuální orientaci papíru.

Deklarace : D_GetPaperOrient :

Poznámka : Funkce vrací hodnotu aktuální orientace kreslícího listu podle tabulky č.8.

Tabulka č.8

Orientace	Vertical	Horizontal
Hodnota	1	2

Viz také : D_SetPaperOrient, D_SetPaperFormat, D_GetPaperFormat, D_GetMaxX, D_GetMaxY.

Příklad :

```
uses DIGIGRAF;
var F : String;
begin
    D_SetPath('C:\DRAW\ORIENT.DGF');
    InitDGF;
    Str(D_GetPaperOrient,F);
    D_OutText(F);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem ORIENT.DGF, který na plotru vypíše aktuální orientaci papíru)

funkce **D_GetMaxX**

Funkce : Vrací maximální možnou horizontální souřadnici (X) pro zvolený formát papíru.

Deklarace : D_GetMaxX : Real ;

Poznámka : Pro stejný formát vrací funkce různé hodnoty v závislosti na orientaci papíru.

Viz také : D_GetMaxY, D_SetPaperFormat, D_GetPaperFormat, D_SetPaperOrient, D_GetPaperOrient.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\MAXX.DGF');
    InitDGF;
    InitMonitor;
    D_Rectangle(0,0,D_GetMaxX,D_GetMaxY);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem MAXX.DGF, který na plotru nakreslí přímku obdelník z bodu /0,0/ do bodu definovaného maximálními rozměry papíru a zároveň provede tuto kresbu na monitoru)

funkce **D_GetMaxY**

Funkce : Vrací maximální možnou vertikální souřadnici (Y) pro zvolený formát papíru.

Deklarace : D_GetMaxY : Byte ;

Poznámka : Pro stejný formát vrací funkce různé hodnoty v závislosti na orientaci papíru.

Viz také : D_GetMaxX, D_SetPaperFomat, D_GetPaperFormat, D_SetPaperOrient, D_GetPaperOrient.

Příklad : Viz příklad v D_GetMaxX.

funkce **D_GetSpeedPD**

Funkce : Vrací aktuální rychlost pohybu spuštěného pisátka.

Deklarace : D_GetSpeedPD : Word ;

Poznámka : Funkce vrací rychlost pisátka v mm/s.

Viz také : D_SetSpeed.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\GSPEED.DGF');
    InitDGF;
    InitMonitor;
    D_SetSpeed(300,400);
    D_Line(0,100,100,100);
    D_SetSpeed(D_GetSpeedPU+100,D_GetSpeedPD+100);
    D_Line(0,110,100,110);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem GSPEED.DGF, který na plotru vykreslí jednu přímku o délce 100 mm aktuální rychlostí, poté druhou přímku rychlostí větší o 100 mm/s).

funkce **D_GetSpeedPU**

Funkce : Vrací aktuální rychlost pohybu zvednutého pisátka.

Deklarace : D_GetSpeedPU : Word ;

Poznámka : Funkce vrací rychlost pisátka v mm/s.

Viz také : D_SetSpeed.

Příklad : Viz příklad D_GetSpeedPD.

funkce **D_GetAccelPD**

Funkce : Vrací aktuální zrychlení pohybu spuštěného pisátka.

Deklarace : D_GetAccelPD : Word;

Poznámka : Funkce vrací zrychlení ve stupních zrychlení podle tabulky č.9.

Tabulka č.9

Stupeň zrychlení	Zrychlení v mm/s^2
1	50
2	200
3	450
4	800

5	1250
6	1800
7	2450
8	3200
9	4050
10	5000

Viz také : D_SetAccel.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\GACCEL.DGF');
    InitDGF;
    InitMonitor;
    D_SetAccel(5,5);
    D_Line(0,100,100,100);
    D_SetAccel(D_GetAccelPU+2,D_GetAccelPD+2);
    D_Line(0,110,100,110);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem GACCEL.DGF, který na plotru vykreslí jednu přímku o délce 100 mm aktuálním zrychlením, poté druhou přímku zrychlením větším o 2 stupně).

funkce **D_GetAccelPU**

Funkce : Vrací aktuální zrychlení pohybu zvednutého pisátka.

Deklarace : D_GetAccelPU : Word;

Poznámka : Funkce vrací zrychlení ve stupních zrychlení podle tabulky č.10.

Tabulka č.10

Stupeň zrychlení	Zrychlení v mm/s^2
1	50
2	200
3	450
4	800
5	1250
6	1800
7	2450
8	3200
9	4050
10	5000

Viz také : D_SetAccel.

Příklad : Viz příklad v D_GetAccelPD.

4.4.2. Procedury a funkce pro nastavení a zjištění atributů grafických prvků

procedura **D_SetColor**

Funkce : Nastaví barvu popř. tloušťku pera.

Deklarace : D_SetColor(Barva : Byte);

Poznámka : Procedura nastaví pisátko specifikované parametrem *Barva* jako pisátko aktuální (aktivní). Veškerá další kresba bude nyní prováděna pouze tímto pisátkem, až do dalšího volání procedury D_SetColor nebo D_SetPen.

Proměnná *Barva* může nabývat hodnot z intervalu [1..MaxPenNum]. Hodnotu MaxPenNum vrací funkce D_GetMaxColor nebo D_GetMaxPen a závisí na technickém vybavení kreslícího zařízení (místo 2-pisátkové hlavy je možnost výměny za hlavu 4-pisátkovou).

Tuto proceduru je vhodné vzhledem ke svému názvu použít v případě, že jsou na kreslící hlavě nasazeny pisátka různých barev a voláním této procedury se provede výběr barvy. Jinak má naprosto stejnou funkci jako procedura D_SetPen.

Viz také : D_SetPen, D_GetColor, D_GetPen, D_GetMaxColor, D_GetMaxPen.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\SCOLOR.DGF');
    InitDGF;
    InitMonitor;
    D_SetColor(1);
    D_Circle(50,50,50);
    D_SetColor(2);
    D_Rectangle(0,0,100,100);
    CloseMonitor;
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem SCOLOR.DGF, který na plotru vykreslí kružnici (pisátkem č.1) vepsanou čtverci (pisátko č.2)).

procedura **D_SetPen**

Funkce : Nastaví tloušťku popř. barvu pera.

Deklarace : D_SetPen(CisPis : Byte);

Poznámka : Procedura nastaví pisátko specifikované parametrem *CisPis* jako pisátko aktuální (aktivní). Veškerá další kresba bude nyní prováděna pouze tímto pisátkem, až do dalšího volání procedury D_SetPen nebo D_SetColor.

Proměnná *CisPis* může nabývat hodnot z intervalu [1..MaxPenNum]. Hodnotu MaxPenNum vrací funkce D_GetMaxPen nebo D_GetMaxColor a závisí na technickém vybavení kreslícího zařízení (místo 2-pisátkové hlavy je možnost výměny za hlavu 4-pisátkovou)

Tuto proceduru je vhodné vzhledem ke svému názvu použít v případě, že jsou na kreslící hlavě nasazeny pisátka různých tlouštěk a voláním této procedury se provede výběr tloušťky pera. Jinak má naprosto stejnou funkci jako procedura D_SetColor.

Viz také : D_SetColor, D_GetColor, D_GetPen, D_GetMaxColor, D_GetMaxPen.

Příklad : Viz příklad v D_SetColor (místo D_SetColor nahraď D_SetPen).

procedura **D_SetLineStyle**

Funkce : Nastavení typu čáry.

Deklarace : D_SetLineStyle (PattNumb : Byte);

Poznámka : Nastaví typ čáry s kterým se budou vykreslovat všechny grafické elementy. Jsou předdeklarovány tyto typy čar (viz tabulka č.11).

Tabulka č.11

Hodnota	Jméno	Typ	Vzorek
0	SolidLn	Plná	
1	DashedLn1	Čárkovaná1	
2	DashedLn2	Čárkovaná2	
3	DashedLn3	Čárkovaná3	
4	CenterLn1	Čerchovaná1	
5	CenterLn2	Čerchovaná2	
6	CenterLn3	Čerchovaná3	
7	CenterLn4	Čerchovaná4	

Vzorek typu čáry lze změnit pomocí procedury D_SetLineSettings.

Omezení : Při kontrole na monitoru se nezobrazí přesný typ čáry, ale pouze tyto 3 typy : plná (0), čárkovaná(1-3), čerchovaná(4-7).

Viz také : D_SetLineSettings, D_GetLineSettings.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\SLINE.DGF');
    InitDGF;
    D_SetLineStyle(DashedLn3);
    D_Circle(100,100,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem SLINE.DGF, který na plotru vykreslí čárkovaně kružnici).

procedura **D_SetLineSettings**

Funkce : Definuje a mění uživatelský typ čáry.

Deklarace : D_SetLineSettings (PattNumb : Byte ; L1, M1, L2, M2, L3, M3, L4, M4 : Real);

Poznámka : Procedura definuje a mění uživatelský typ čáry podle následujících pravidel a omezení :
Parametr *PattNumb* určuje typ čáry, které se změna definice týká.
Vzorek přerušované čáry je definován osmicí čísel *L1* až *L4* a *M1* až *M4*, které udávají délky čárek a mezer v mm.
Počáteční hodnoty jsou uvedeny v tabulce č.12.

Tabulka č.12

Hodnota	Jméno	Typ	L 1	M 1	L 2	M 2	L 3	M 3	L 4	M 4	Vzorek
0	SolidLn	Plná	x	x	x	x	x	x	x	x	————————
1	DashedLn1	Čárkovaná1	5	2	5	2	5	2	5	2	— — — — — —
2	DashedLn2	Čárkovaná2	8	2	8	2	8	2	8	2	—— —— —— ——
3	DashedLn3	Čárkovaná3	2	2	2	2	2	2	2	2	- - - - - - - - - -
4	CenterLn1	Čerchovaná1	8	2	2	2	8	2	2	2	—— - —— - —— -
5	CenterLn2	Čerchovaná2	8	2	2	2	2	2	0	0	—— - - —— - -
6	CenterLn3	Čerchovaná3	8	2	8	2	2	2	0	0	— — - — — -
7	CenterLn4	Čerchovaná4	8	2	8	2	2	2	2	2	— — - - — —

Omezení : Čísla L1 a M1 musí být nezáporná, alespoň jedno musí být kladné. Uživatelsky definovaný typ čáry se nezobrazí při kontrole na monitoru.

Viz také : D_SetLineStyle, D_GetLineSettings.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\SLINESET.DGF');
    InitDGF;
    D_SetLineSettings(DashedLn2,5,8,5,8,5,8,5,8);
    D_SetLineStyle(DashedLn2);
    D_Circle(100,100,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem SLINESET.DGF, který na plotru vykreslí kružnici uživatelským typem čáry - čárkovaná s dlouhými mezerami).

procedura **D_SetTextStyle**

Funkce : Nastaví atributy textu.

Deklarace : D_SetTextStyle (Font : Byte ; Direction, CharSize : Real);

Poznámka : Procedura nastaví znakový font, směr výpisu textu a samotnou velikost textu.

Plotr DIGIGRAF obsahuje jedinou tabulku znaků se třemi typy znakových fontů. Tato tabulka je programově rozdělena na tři tabulky fontů, které je možno pomocí této procedury zvolit jako aktuální (parametr *Font* , viz tabulka č.13).

Tabulka č.13

Jméno fontu	Číslo fontu	Typ fontu
CSFont	1	Český
RUFont	2	Ruský
GRFont	3	Řecký

Přiřazení kláves jednotlivým znakům fontů je uvedeno v příloze č.1.

Parametr *Direction* udává pod jakým úhlem bude text vypisován, tzn. jaký úhel svírá směrnice řádku s osou X. Úhel se zadává ve stupních. Standardní hodnota je 0^{o}.

Parametr *CharSize* udává velikost znaků v mm. Standardní hodnota je 5 mm.

Omezení : Pokud je nastaven jiný font než CSFont, bude text na monitoru ve formě přiřazovacích znaků (viz příloha č.1).

Pokud je nastaven jiný směr výpisu textu než standardní, je kontrolní výpis textu na monitoru směrován od zdola nahoru (tzn. směrnice textu svírá s osou X úhel 90^{o}).

Výška znaků na monitoru není přesně v měřítku k ostatní kresbě. Je to způsobeno chybějící možností nastavovat spojitě výšku znaků Turbo Pascalu.

Výšku znaků je možné volit min. 2 mm.

Viz také : D_SetTextJustify, D_OutText, D_OutTextXY.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\STEXTS.DGF');
    InitDGF;
    D_SetTextStyle(CSFont,45,10);
    D_OutTextXY(100,100,'Sklon 45 stupnu');
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem STEXTS.DGF, který na plotru vypíše text od bodu /100,100/ o výšce znaků 10 mm a pod úhlem 45 stupňů).

procedura **D_SetTextJustify**

Funkce : Nastaví způsob zarovnávání textu.

Deklarace : D_SetTextJustify (Horiz : Word);

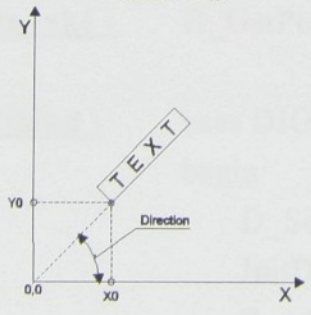
Poznámka : Procedura nastaví způsob horizontálního zarovnávání textu. Zarovnávání se provádí vždy vzhledem k aktuální pozici pisátka. Pro definici zarovnávání lze použít těchto předdefinovaných konstant proměnné *Horiz* (viz tabulka č.14).

Tabulka č.14

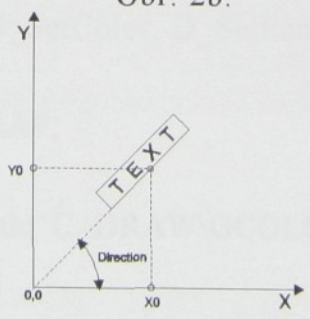
Název konstanty	Hodnota
LeftText	0
CenterText	1
RightText	2

V případě použitím konstanty *LeftText* se text vypisuje od aktuální pozice směrem doprava (při standardním úhlu směru výpisu textů tj. 0^{o}) viz obr.2a. Při použití *CenterText* je text centrován vzhledem k aktuální pozici pisátka (obr.2b) a pro *RightText* je text zarovnán tak, že u aktuální pozice pisátka text končí (obr.2c.).

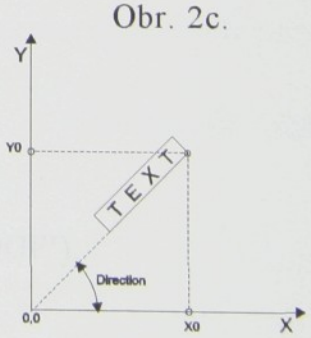
Obr. 2a.



Obr. 2b.



Obr. 2c.



Viz také : D_SetTextStyle, D_OutText, D_OutTextXY.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\STEXTS.DGF');
    InitDGF;
    D_SetTextStyle(CSFont,45,10);
    D_SetTextJustify(CenterText);
    D_OutTextXY(100,100,'Sklon 45 stupnu');
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem STEXTS.DGF, který na plotru vypíše text o výšce znaků 10 mm a pod úhlem 45 stupňů vycentrovaný k bodu /100,100/).

funkce **D_GetColor**

Funkce : Vrací aktuální číslo pisátka.

Deklarace : D_GetColor : Byte ;

Poznámka : Funkce má stejný význam jako D_GetPen. Vzhledem k jejímu názvu je vhodné ji použít při obsazení kreslící hlavy pisátky různých barev (pro lepší srozumitelnost programu).

Viz také : D_GetPen, D_SetColor, D_SetPen.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\GCOLOR.DGF');
    InitDGF;
    Randomize;
    D_SetColor(Random(D_GetMaxColor+1));
```

```
    D_Arc(100,100,0,180,50);
    if D_GetColor=1 then D_SetColor(2)
     else D_SetColor(1);
    D_Arc(100,100,180,360,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem GCOLOR.DGF, který na plotru polovinu kružnice jedním náhodně zvoleným pisátkem a druhou polovinu druhým pisátkem).

funkce **D_GetPen**

Funkce : Vrací aktuální číslo pisátka.

Deklarace : D_GetPen : Byte ;

Poznámka : Funkce má stejný význam jako D_GetColor. Vzhledem k jejímu názvu je vhodné ji použít při obsazení kreslící hlavy pisátky různých tlouštěk (pro lepší srozumitelnost programu).

Viz také : D_GetColor, D_SetPen, D_SetColor.

Příklad : Viz příklad v D_GetColor (místo D_GetColor nahraď D_SetPen).

funkce **D_GetLineSettings**

Funkce : Vrací aktuální typ čáry.

Deklarace : D_GetLineSettings : Byte ;

Poznámka : Vrací číslo aktuálního typu čáry v rozsahu 1 až 7.

Viz také : D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
var Tc,T : Byte;
begin
    D_SetPath('C:\DRAW\GLINESS.DGF');
    InitDGF;
    Tc:=D_GetLineSettings;
    Readln('Zadej typ čáry :',T);
    D_SetLineStyle(T);
    D_Line(0,0,100,100);
    D_SetLineStyle(Tc);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem GLINESS.DGF, který na plotru vykreslí, po zadání typu čáry, přímku tímto typem čáry a obnoví původní nastavení typu čáry).

funkce **D_GetMaxColor**

Funkce : Vrací maximální možnou hodnotu čísla pisátka.

Deklarace : D_GetMaxColor : Byte ;

Poznámka : Hodnota funkce je závislá na typu použité kreslící hlavy. V programu je předdefinována hodnota *MaxPenNum* = 2.
Funkce má stejný význam jako D_GetMaxPen. Vzhledem ke svému názvu je definována pro použití různých barev pisátek.

Viz také : D_GetMaxPen.

Příklad : Viz příklad v D_GetColor.

funkce **D_GetMaxPen**

Funkce : Vrací maximální možnou hodnotu čísla pisátka.

Deklarace : D_GetMaxPen : Byte ;

Poznámka : Tato funkce má naprosto stejný význam jako funkce D_GetMaxColor. Pro svůj název je definována pro použití hlavy s různými tloušťkami pisátek.

Viz také : D_GetMaxColor.

Příklad : Viz příklad v D_GetColor (místo D_GetMaxColor nahraď D_GetMaxPen).

4.4.3. Procedury pro kreslení grafických prvků

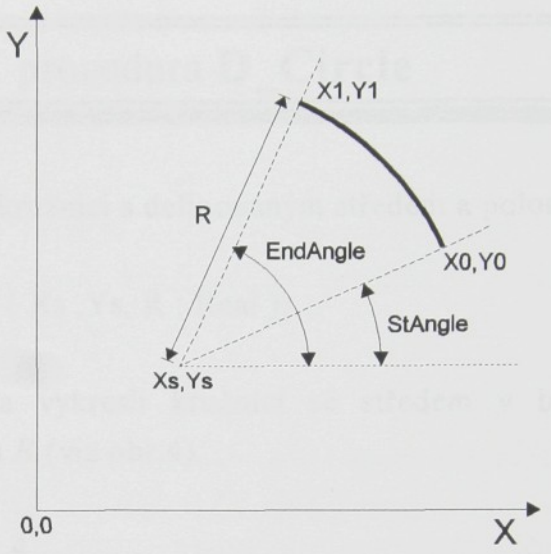
procedura **D_Arc**

Funkce : Nakreslí kruhový oblouk od počátečního úhlu ke koncovému úhlu s definovaným středem a poloměrem.

Deklarace : D_Arc (Xs, Ys, StAngle, EndAngle, R : Real);

Poznámka : Procedura vykreslí kruhový oblouk se středem v bodě (*Xs,Ys*) o poloměru *R*. Kresba oblouku začíná od úhlu *StAngle* a končí v úhlu *EndAngle* (viz obr.3).

Obr. 3



Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen*.
Souřadnice středu (*Xs,Ys*) a velikost poloměru *R* se zadávají v mm. Úhly *StAngle* a *EndAngle* se zadávají ve stupních.
Pisátko se po vykreslení oblouku nastaví na konec oblouku do bodu (Xs+R*Cos(EndAngle),Ys+R*Sin(EndAngle)).

Viz také : D_Circle, D_SetColor, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DARC.DGF');
    InitDGF;
    D_Arc(100,100,90,180,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DARC.DGF, který na plotru vykreslí oblouk se středem v bodě /100,100/ začínající pod úhlem 90° a končící pod úhlem 180° s poloměrem 50mm)

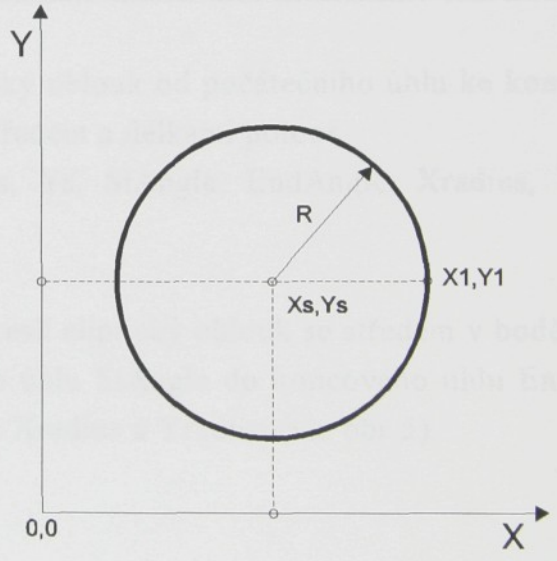
procedura **D_Circle**

Funkce : Nakreslí kružnici s definovaným středem a poloměrem.

Deklarace : D_Circle (Xs ,Ys, R : Real);

Poznámka : Procedura vykreslí kružnici se středem v bodě (*Xs,Ys*) o poloměru *R* (viz obr.4).

Obr.4



Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen.*

Souřadnice středu (*Xs,Ys*) a velikost poloměru *R* se zadává v mm.

Pisátko se po vykreslení kružnice nastaví do bodu (*X1,Y1*).

Viz také : D_Arc, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DCIRCLE.DGF');
    InitDGF;
    D_Circle(100,100,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DCIRCLE.DGF, který na plotru vykreslí kružnici se středem v bodě /100,100/ s poloměrem 50mm)

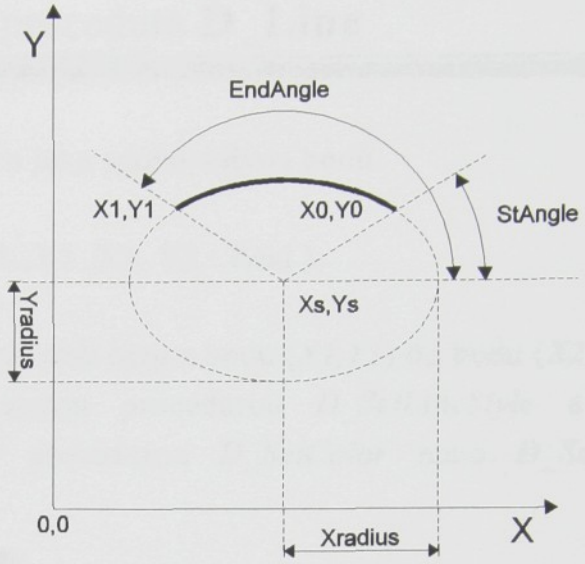
procedura **D_Ellipse**

Funkce : Nakreslí eliptický oblouk od počátečního úhlu ke koncovému s definovaným středem a délkami poloos.

Deklarace : D_Ellipse (Xs, Ys, StAngle, EndAngle, Xradius, Yradius : Real);

Poznámka : Procedura nakreslí eliptický oblouk se středem v bodě (Xs,Ys), od počátečního úhlu StAngle do koncového úhlu EndAngle, s délkami poloos Xradius a Yradius (viz obr.5).

Obr.5



Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen.*

Souřadnice středu (*Xs,Ys*) a délky poloos Xradius,Yradius se zadávají v mm.

Úhly *StAngle* a *EndAngle* se zadávají ve stupních.

Pisátko se po vykreslení oblouku nastaví do bodu (*X1,Y1*).

Viz také : D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DELLIPSE.DGF');
    InitDGF;
    D_Ellipse(100,100,0,180,100,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DELLIPSE.DGF, který na plotru vykreslí horní polovinu eliptického oblouku se středem v bodě /100,100/)

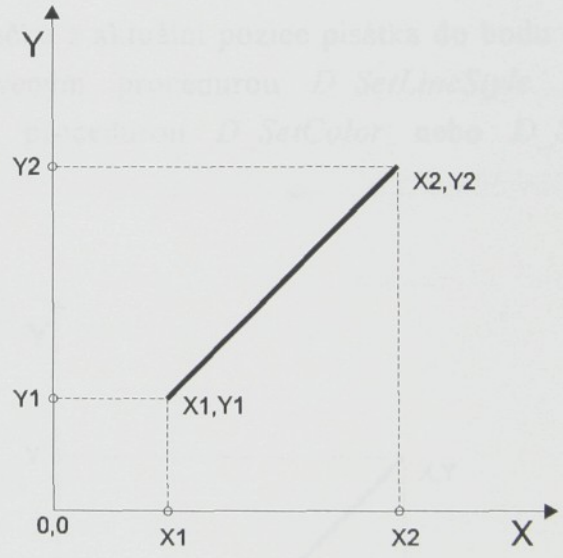
procedura **D_Line**

Funkce : Nakreslí čáru jako spojnici dvou bodů.

Deklarace : D_Line (X1 ,Y1 ,X2, Y2 : Real);

Poznámka : Procedura nakreslí čáru z bodu (*X1,Y1*) do bodu (*X2,Y2*) typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen* (viz obr.6).

Obr.6



Souřadnice *X1,Y1,X2,Y2* se zadávají v mm.
Po vykreslení čáry se pisátko přesune do bodu (*X2,Y2*).

Viz také : D_LineRel, D_LineTo, D_MoveRel, D_MoveTo, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DLINE.DGF');
    InitDGF;
    D_Line(50,50,100,100);
```

```
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DLINE.DGF, který na plotru vykreslí čáru z bodu /50,50/ do bodu /100,100/)

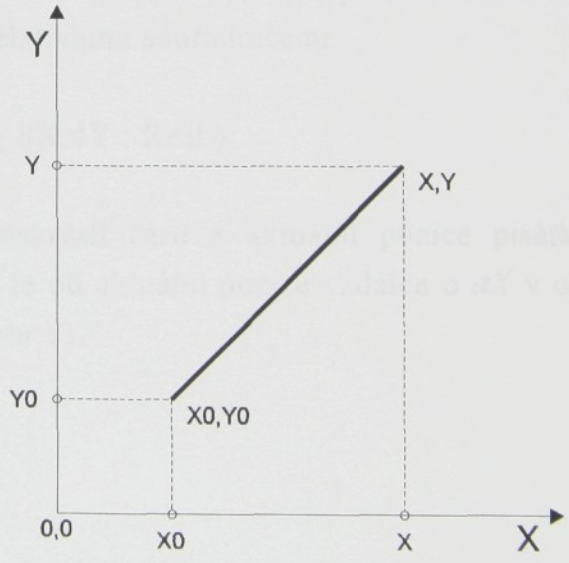
procedura **D_LineTo**

Funkce : Nakreslí čáru z aktuální pozice pisátka do definovaného bodu.

Deklarace : D_LineTo (X,Y : Real);

Poznámka : Nakreslí úsečku z aktuální pozice pisátka do bodu (X,Y) typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen* (viz obr.7).

Obr.7



Souřadnice X,Y se zadávají v mm.

Po vykreslení čáry se pisátko přesune do bodu (X,Y).

Viz také : D_Line, D_LineRel, D_MoveRel, D_MoveTo, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DLINETO.DGF');
    InitDGF;
    D_MoveTo(50,50);
    D_LineTo(100,100);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DLINETO.DGF, který na plotru vykreslí čáru z bodu /50,50/ do bodu /100,100/)

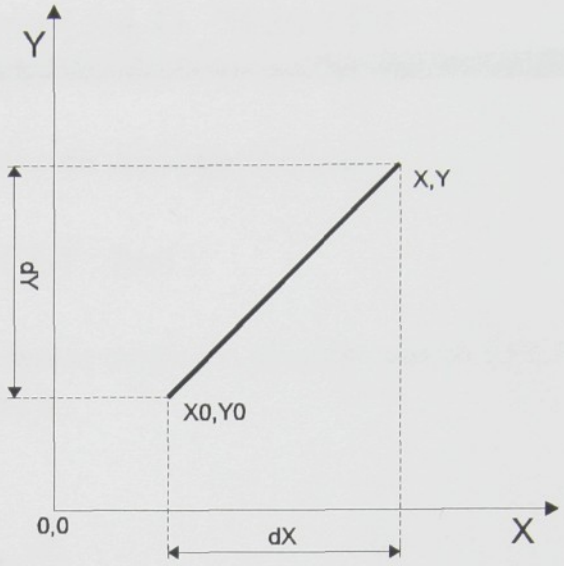
procedura **D_LineRel**

Funkce : Nakreslí čáru z aktuální pozice pisátka do bodu, který je definován relativními souřadnicemi.

Deklarace : D_LineRel (dX,dY : Real);

Poznámka : Procedura nakreslí čáru z aktuální pozice pisátka do bodu (X,Y), který je od aktuální pozice vzdálen o dX v ose X a dY v ose Y (viz obr.8).

Obr.8



Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen*

Diference dX a dY se zadávají v mm.

Viz také : D_Line, D_LineTo, D_MoveRel, D_MoveTo, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DLINEREL.DGF');
    InitDGF;
    D_MoveTo(50,50);
    D_LineRel(50,50);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DLINEREL.DGF, který na plotru vykreslí čáru z bodu /50,50/ do bodu /100,100/)

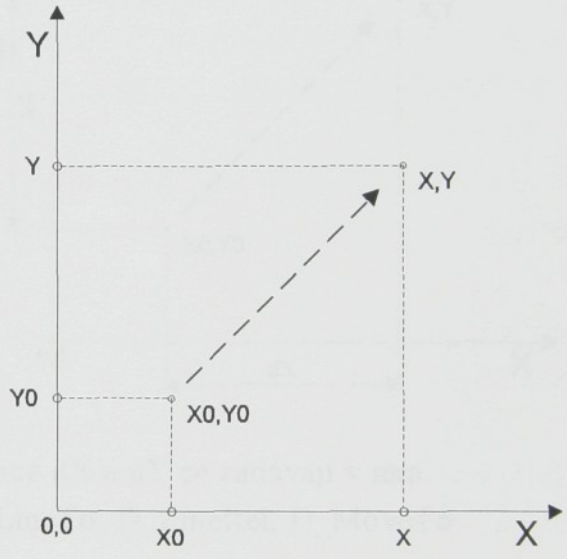
procedura **D_MoveTo**

Funkce : Přesune pisátko do definovaného bodu.

Deklarace : D_MoveTo (X,Y : Real);

Poznámka : Procedura přesune pisátko z aktuální pozice (*X0,Y0*) do bodu (*X,Y*). Viz obr.9.

Obr.9



Souřadnice (*X,Y*) se zadávají v mm.

Viz také : D_Line, D_LineTo, D_LineRel, D_MoveRel.

Příklad : Viz příklad v D_LineTo.

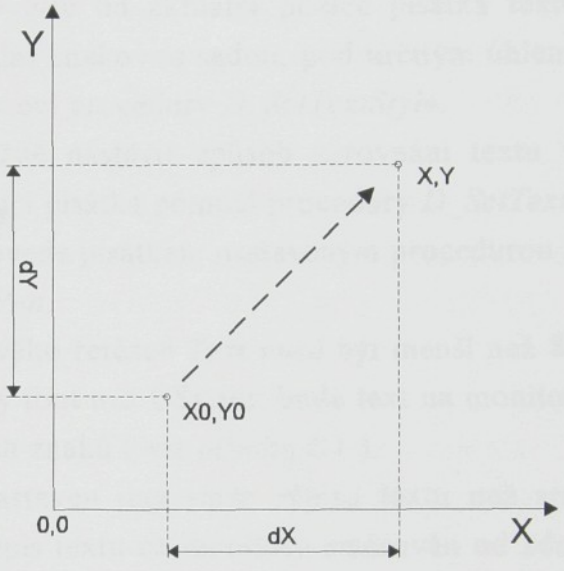
procedura **D_MoveRel**

Funkce : Přesune pisátko do bodu definovaného relativními souřadnicemi.

Deklarace : D_MoveRel (dX,dY : Real);

Poznámka : Procedura přesune pisátko z aktuální pozice *(X0,Y0)* do *(X,Y)*, který je od aktuální pozice vzdálen o *dX* v ose X a *dY* v ose Y (viz obr.10).

Obr.10



Diference dX a dY se zadávají v mm.

Viz také : D_Line, D_LineTo, D_LineRel, D_MoveTo.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DMOVEREL.DGF');
    InitDGF;
    D_MoveRel(50,50);
    D_LineTo(100,100);
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DMOVEREL.DGF, který na plotru vykreslí čáru z bodu /50,50/ do bodu /100,100/)

procedura **D_OutText**

Funkce : Vykreslí textový řetězec na aktuální pozici pisátka.

Deklarace : D_OutText (Text : string);

Poznámka : Procedura vypíše od aktuální pozice pisátka textový řetězec *Text* s aktuální znakovou sadou, pod určitým úhlem a velikostí zadanou pomocí procedury *D_SetTextStyle*.
Dále je možné nastavit způsob zarovnání textu vzhledem k aktuální pozici pisátka pomocí procedury *D_SetTextJustify*.
Výpis se provede pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen*.

Omezení : Délka textového řetězce *Text* musí být menší než 88. Pokud je nastaven jiný font než CSFont, bude text na monitoru ve formě přiřazovacích znaků (viz příloha č.1).
Pokud je nastaven jiný směr výpisu textu než standardní, je kontrolní výpis textu na monitoru směrován od zdola nahoru (tzn. směrnice textu svírá s osou X úhel 90°).
Výška znaků na monitoru není přesně v měřítku k ostatní kresbě. Je to způsobeno chybějící možností nastavovat spojitě výšku znaků Turbo Pascalu.

Viz také : D_OutTextXY, D_SetTextStyle, D_SetTextJustify.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DOUTTEXT.DGF');
    InitDGF;
    D_OutText('DIGIGRAF 1208A');
    CloseDGF;
end;
```

(Program vytvoří soubor s názvem DOUTTEXT.DGF, který na plotru vypíše text 'DIGIGRAF 1208A').

procedura **D_OutTextXY**

Funkce : Vykreslí textový řetězec na definovanou souřadnici.

Deklarace : D_OutTextXY (X,Y : Real ; Text : string);

Poznámka : Procedura vypíše textový řetězec od bodu *(X,Y)* aktuální znakovou sadou, pod určitým úhlem a velikostí zadanou pomocí procedury *D_SetTextStyle*.
Dále je možné nastavit způsob zarovnání textu vzhledem k bodu *(X,Y)* pozici pisátka pomocí procedury *D_SetTextJustify*.
Výpis se provede pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen*.

Omezení : Délka textového řetězce *Text* musí být menší než 88. Pokud je nastaven jiný font než CSFont, bude text na monitoru ve formě přiřazovacích znaků (viz příloha č.1).
Pokud je nastaven jiný směr výpisu textu než standardní, je kontrolní výpis textu na monitoru směrován od zdola nahoru (tzn. směrnice textu svírá s osou X úhel 90°).
Výška znaků na monitoru není přesně v měřítku k ostatní kresbě. Je to způsobeno chybějící možností nastavovat spojitě výšku znaků Turbo Pascalu.

Viz také : D_OutText, D_SetTextStyle, D_SetTextJustify.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DTEXTXY.DGF');
    InitDGF;
    D_OutTextXY(100,100,'DIGIGRAF 1208A');
    CloseDGF;
```

```
end;
```
(Program vytvoří soubor s názvem DTEXTXY.DGF, který na plotru vypíše na pozici /100,100/ text 'DIGIGRAF 1208A').

procedura **D_PutPixel**

Funkce : Vykreslí bod na definovanou souřadnici.

Deklarace : D_PutPixel (X,Y : Real ; Color : Byte);

Poznámka : Procedura vykreslí bod na souřadnici (*X,Y*), pomocí pisátka definovaného parametrem *Color*.

Viz také : D_MoveTo, D_MoveRel, D_Line, D_LineRel, D_LineTo.

Příklad :

```
uses DIGIGRAF,CRT;
begin
    D_SetPath('C:\DRAW\DPIXEL.DGF');
    InitDGF;
    Randomize;
    repeat
     D_PutPixel(Random(100),Random(100),Random(3));
    until KeyPressed;
    CloseDGF;
end;
```
(Program vytvoří soubor s názvem DPIXEL.DGF, který na plotru vykreslí množství náhodně položených bodů).

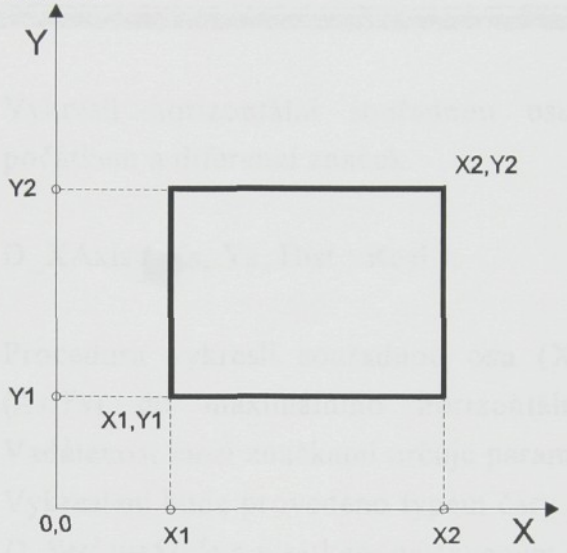
procedura **D_Rectangle**

Funkce : Vykreslí pravoúhelník definovanými dvěma body.

Deklarace : D_Rectangle (X1 ,Y1 ,X2, Y2 : Real);

Poznámka : Bod (*X1,Y1*) definuje první bod pravoúhelníku a bod (*X2,Y2*) definuje druhý, protilehlý bod (viz obr.11).

Obr.11



Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen.*

Souřadnice *X1,Y1,X2,Y2* se zadávají v mm.

Pisátko se po vykonání procedury přesune do bodu *X1,Y1.*

Viz také : D_Line, D_LineTo, D_LineRel, D_MoveRel, D_MoveTo, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DRECTAN.DGF');
    InitDGF;
    D_Circle(50,50,50);
    D_Rectangle(0,0,100,100);
    CloseDGF;
```

```
end;
```
(Program vytvoří soubor s názvem DRECTAN.DGF, který na plotru opíše čtverec kolem kružnice).

procedura **D_XAxis**

Funkce : Vykreslí horizontální souřadnou osu (X) s definovaným počátkem a diferencí značek.

Deklarace : D_XAxis (Xs, Ys, Dist : Real);

Poznámka : Procedura vykreslí souřadnou osu (X) s počátkem v bodě (*Xs,Ys*) do maximálního horizontálního rozměru papíru. Vzdálenost mezi značkami určuje parametr *Dist*.
Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen*.
Souřadnice *Xs,Ys* a diference *Dist* se zadávají v mm.
Pisátko se po vykonání procedury přesune do bodu *Xs,Ys*.

Viz také : D_Line, D_LineTo, D_LineRel, D_MoveRel, D_MoveTo, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad :

```
uses DIGIGRAF;
begin
    D_SetPath('C:\DRAW\DAXIS.DGF');
    InitDGF;
    D_XAxis(10,10,5);
    D_YAxis(10,10,5)
    CloseDGF;
end;
```
(Program vytvoří soubor s názvem DAXIS.DGF, který na plotru vykreslí osy X a Y s roztečí značek 5 mm).

procedura **D_YAxis**

Funkce : Vykreslí vertikální souřadnou osu (Y) s definovaným počátkem a diferencí značek.

Deklarace : D_YAxis (Xs, Ys, Dist : Real);

Poznámka : Procedura vykreslí souřadnou osu (Y) s počátkem v bodě (*Xs,Ys*) do maximálního vertikálního rozměru papíru. Vzdálenost mezi značkami určuje parametr *Dist*.
Vykreslení bude provedeno typem čáry nastaveným procedurou *D_SetLineStyle* a pisátkem nastaveným procedurou *D_SetColor* nebo *D_SetPen*.
Souřadnice *Xs,Ys* a diference *Dist* se zadávají v mm.
Pisátko se po vykonání procedury přesune do bodu *Xs,Ys*.

Viz také : D_Line, D_LineTo, D_LineRel, D_MoveRel, D_MoveTo, D_SetColor, D_SetPen, D_SetLineStyle.

Příklad : Viz příklad v D_XAxis.

5. Interpretr jazyka HPGL

Program *HPGL_DGF.PAS* složí k interpretaci grafických příkazů jazyka HPGL (Hewlett-Packard Graphics Language). pro plotr DIGIGRAF 1208A.

Interpretr je sestaven jako samostatný program, který po zadání patřičných parametrů čte a interpretuje soubor grafických instrukcí jazyka HPGL na soubor grafických instrukcí pro DIGIGRAF 1208A.

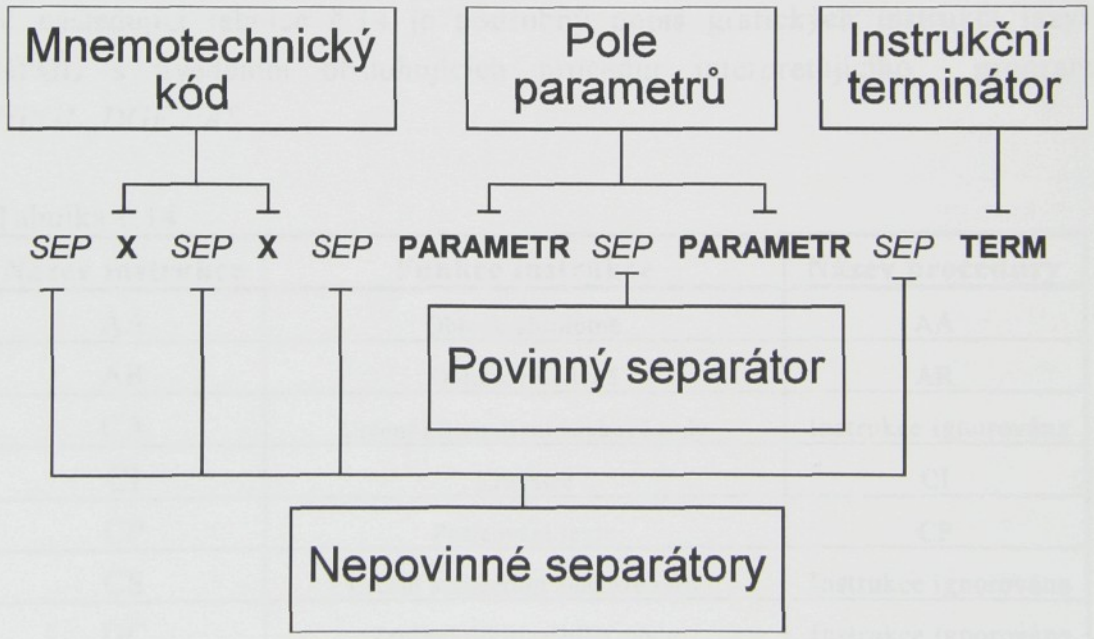
5.1. Popis syntaxe jazyka HPGL

[2]

Grafická instrukce HPGL je tvořena dvoupísmenným mnemotechnickým kódem (velká či malá písmena), za nímž může následovat pole parametrů a případně tzv. "instrukční terminátor". Je-li v instrukci uvedeno více parametrů (číselných), musí být navzájem odděleny alespoň jednou mezerou nebo čárkou, nebo znaménkem + či - , jemuž smějí předcházet čárky či mezery (povinný separátor). Libovolný počet mezer a čárek smí být kromě toho uveden jako nepovinný separátor před, za i mezi písmeny mnemotechnického kódu a před instrukčním terminátorem. Jako instrukční terminátor slouží alespoň jeden středník. Výběr instrukce končí dosažením tohoto terminátoru.

Uvedená syntaxe je znázorněna následujícím schématem (viz obr.12).

Obr.12



Pozn.: Popsaná syntaxe neplatí pro instrukci SM a DT, které jako parametr interpretují první znak bezprostředně následující za mnemotechnickým kódem.

Některé instrukce mají tzv. nepovinné parametry, které - jsou-li vynechány - nabývají předem definovaných standardních hodnot. Některé

instrukce (PA,PR,PU,PD) mohou mít i větší počet parametrů, oddělených navzájem povinným separátorem.

Grafické instrukce lze rozdělit do následujících skupin podle jejich účelu :

- nastavení zapisovače : nastavení standardních hodnot, inicializace, nastavení formátu kreslícího listu
- stanovení jednotek a hranic kresby : práce s referenčními body, nastavení kreslícího okna, uživatelského měřítka a pootočení souřadného systému
- řízení kresby : výběr pera, spuštění a zvednutí pisátka, volba rychlosti kresby, kreslení úseček, oblouků a kružnic
- zvláštní způsob kresby : značky na osách, typ čáry, symbolový mód
- výpis textů : volba znakové sady, velikosti písma, sklonu písma a sklonu řádky
- výstup ze zapisovače : vysílání chybových kódů, stavu zapisovače, okna a dorazů.

V následující tabulce č.14 je podrobný popis grafických instrukcí jazyka HPGL s uvedením obsluhujících procedur interpretujícího programu *HPGL_DGF.PAS :*

Tabulka č.14

Název instrukce	Funkce instrukce	Název procedury
AA	Oblouk absolutně	AA
AR	Oblouk relativně	AR
CA	Určení alternativní znakové sady	Instrukce ignorována
CI	Kružnice	CI
CP	Pozicování textu	CP
CS	Určení standardní znakové sady	Instrukce ignorována
DC	Zrušení digitizačního módu	Instrukce ignorována
DF	Nastavení standardních hodnot	DF
DI	Směr řádku absolutně	DI
DP	Digitizace bodu	Instrukce ignorována
DR	Směr řádku relativně	DR
DT	Definice textového terminátoru	DT

IM	Vložení podmínkové masky	Instrukce ignorována
IN	Inicializace	DF
IP	Vložení referenčních bodů	Instrukce ignorována
IW	Vložení okénka	Instrukce ignorována
LB	Výpis textu	LB
LT	Typ čáry	LT
OA	Vyslání skutečné pozice a stavu pisátka	Instrukce ignorována
OC	Vyslání požadované pozice a stavu pisátka	Instrukce ignorována
OD	Vyslání digitizovaného bodu a stavu pisátka	Instrukce ignorována
OE	Vyslání kódu programové chyby	Instrukce ignorována
OF	Vyslání faktorů	Instrukce ignorována
OH	Vyslání hranic dorazů	Instrukce ignorována
OI	Vyslání identifikace	Instrukce ignorována
OO	Vyslání náležitostí	Instrukce ignorována
OP	Vyslání referenčních bodů	Instrukce ignorována
OS	Vyslání stavu	Instrukce ignorována
OW	Vyslání okénka	Instrukce ignorována
PA	Úsečka absolutně	PA
PD	Spuštění pisátka	PD
PR	Úsečka relativně	PR
PS	Formát kreslícího listu	Instrukce ignorována
PU	Zvednutí pisátka	PU
RO	Pootočení souřadného systému	Instrukce ignorována
SA	Volba alternativní znakové sady	Instrukce ignorována
SC	Uživatelské měřítko	SC
SI	Absolutní velikost znaku	SI
SL	Sklon znaku	Instrukce ignorována
SM	Symbolový mód	Instrukce ignorována
SP	Výběr pisátka	SP
SR	Relativní velikost znaku	Instrukce ignorována
SS	Volba standardní znakové sady	Instrukce ignorována
TL	Délka značky	Instrukce ignorována
UC	Uživatelův znak	Instrukce ignorována
VS	Volba rychlosti	VS

XT	Značka na ose X	Instrukce ignorována
YT	Značka na ose Y	Instrukce ignorována

Ignorovat většinu instrukcí je nutné z důvodu technického vybavení DIGIGRAFU, hlavně ale nemožností oboustranné komunikace DIGIGRAFU s počítačem. Proto jsou ignorovány všechny instrukce, které zajišťují vyslání informací z plotru na PC. Dále instrukce, které není možné na DIGIGRAFU realizovat (sklon znaku, nastavení referenčních bodů atd.). Tyto instrukce by se za určitých omezení daly simulovat programově, ale tuto možnost jsem vzhledem k jejich postradatelnosti v mém řešení vypustil. Jejich postradatelnost vyplývá z faktu, že programy vygenerované pomocí grafických systémů (AUTOCAD, COREL DRAW, ORCAD) tyto instrukce ani neobsahují a mnou navržený interpretr plně vyhovuje pro jejich interpretaci na DIGIGRAF.

5.2. Popis funkce interpretru

Samotný proces interpretace grafických instrukcí jazyka HPGL na instrukce DIGIGRAFU pomocí programu *HPGL_DGF.PAS* je rozdělen na několik etap :

1) **Lexikální analýza kódu** - spočívá v tom, že se čte text zdrojového souboru a provádí se jeho úprava pro syntaktickou analýzu. Realizuje se zde tak zvané prohledávání a prosévání.

   Prohledávání (scaning) řetězu znaků zdrojového programu se provádí proto, aby byly vyhledány potřebné textové elementy (instrukční terminátory).

   Prosévání (screening) je proces, který se provádí po prohledávání a vynechají se nevýznamné textové elementy (nepovinné separátory tj. mezery a čárky).
2) **Syntaktická analýza kódu** - znamená v našem případě vyjmutí 1 instrukce a následnou kontrolu zda se opravdu o instrukci jedná.
3) **Dekódování typu instrukce** (tzn. rozpoznání jejího mnemotechnického kódu).

4) **Specifikace parametrů** grafické instrukce (tzn. zjištění počtu parametrů a jejich hodnot).
5) **Přepočet a transformace parametrů** grafických instrukcí (tzn. transformace parametrů vzhledem k měřítku kresby atd.).
6) **Vyvolání patřičné obslužné procedury** z programové jednotky DIGIGRAF.PAS (DIGIGRAF.TPU), která zajistí vygenerování instrukce pro DIGIGRAF.

5.3. Spuštění a ovládání interpretru

Program HPGL_DGF.EXE se spouští klasickým způsobem jako každý program pracující pod operačním systémem DOS. Zdrojový program HPGL_DGF.PAS se spouští v prostředí Turbo Pascal nebo Borland Pascal.

Po spuštění program požádá o vložení úplné přístupové cesty a jména interpretovaného souboru grafických instrukcí HPGL. Dále je nutné zadat zdrojový formát papíru (v rozsahu od A0 do A5, a to pomocí čísel 0 až 5) a orientaci papíru (Vertikální = 1 a Horizontální = 2). Při zadání odlišných parametrů než parametrů papíru na který měla být provedena původní kresba ze souboru HPGL, provede program transformaci všech souřadnic na nový formát popř. orientaci papíru (pouze v případě nastaveného uživatelského měřítka instrukcí SC, toto splňuje soubor HPGL exportovaný z COREL DRAW). Prakticky lze toho využít při zvětšování či zmenšování výsledné grafické aplikace.

Jméno vstupního souboru s instrukcemi HPGL je nutno zadat celé tzn. i s příponou, jelikož program rozlišuje mezi dvěmi povolenými příponami *.PLT (AUTOCAD, COREL DRAW) a *.000 (ORCAD). Toto rozlišení je nutné z důvodu odlišného měřítka kreseb v systémech ORCAD a AUTOCAD ,které jsou bez specifikace uživatelského měřítka a tudíž exportují kresbu v jednotkách předvoleného souřadnicového zapisovače.

Program interpretuje správně grafické soubory exportované ze systému AUTOCAD pro nastavený plotr Hewlett-Packard HP 7586 (tento plotr byl zvolen pro možnost kresby do maximálního formátu papíru A0), dále ze systému ORCAD pro nastavenou skupinu plotrů HP a z programu COREL DRAW (WINDOWS) pomocí exportního filtru HPGL. Použití exportních a importních filtrů v programu COREL DRAW dává možnost vykreslení různých formátů grafických dat na DIGIGRAFU.

6. Závěr

Výsledkem řešení je programová jednotka DIGIGRAF v Turbo Pascalu, která zajišťuje grafický výstup z Turbo Pascalu na DIGIGRAF. Tuto jednotku lze libovolně začlenit do uživatelského programu, ve kterém je požadován grafický výstup prostřednictvím kreslícího zařízení DIGIGRAF 1208A. Funkčnost vyvinuté programové jednotky byla úspěšně vyzkoušena na několika grafických aplikacích (viz příloha č.2). Výpis této jednotky je v příloze č.4.

Dále byl sestaven interpretr souborů instrukcí grafického jazyka HPGL na DIGIGRAF. Ukázky interpretovaných grafických souborů jsou v příloze č.3 a výpis interpretru v příloze č.5.

V první části diplomové práce je popsáno automatické kreslící zařízení DIGIGRAF 1208A, jeho funkce a osluha.

V kapitole „Komunikace PC s DIGIGRAFEM“ je popsáno technické zabezpečení a obsluha výše uvedené komunikace.

V další kapitole je popsána programová jednotka HPGL pro výstup z TP nebo BP. Tento popis je dále prohlouben na jednotlivé procedury a funkce této jednotky.

Poslední kapitola „Interpretr jazyka HPGL“ pojednává o syntaxi HPGL, funkci interpretru a je zde uveden návod na obsluhu a využití tohoto interpretru.

Literatura : [1] DIGIGRAF 1208A firemní literatura ZPA NOVÝ BOR

[2] COLORGRAF 0512 Návod k obsluze.

- Mikula Pavel : Referenční manuál PASCAL 7.0
- Miroslav Klobasa : Diplomová práce „Ovládání zapisovače CG 0512."
- Ing.Miroslav Olehla, CSc, Ing.František Král, Ing.Ivan Švarc, CSc, Ing. Jan Tišer : „Programování, programovací jazyky a operační systémy"

ČESKÝ FONT

A : Znak vykreslený DIGIGRAFEM
B : Hodnota znaku v tabulce fontu DIGIGRAFU (hex)
C : Přiřazovací znak
D : Hodnota přiřazovacího znaku v tabulce ASCII

A	B	C	D	A	B	C	D	A	B	C	D
A	$C1	A	65	S	$E2	S	83	k	$92	k	107
B	$C2	B	66	T	$E3	T	84	l	$93	l	108
C	$C3	C	67	U	$E4	U	85	m	$94	m	109
D	$C4	D	68	V	$E5	V	86	n	$95	n	110
E	$C5	E	69	W	$E6	W	87	o	$96	o	111
F	$C6	F	70	X	$E7	X	88	p	$97	p	112
G	$C7	G	71	Y	$E8	Y	89	q	$98	q	113
H	$C8	H	72	Z	$E9	Z	90	r	$99	r	114
I	$C9	I	73	a	$81	a	97	s	$A2	s	115
J	$D1	J	74	b	$82	b	98	t	$A3	t	116
K	$D2	K	75	c	$83	c	99	u	$A4	u	117
L	$D3	L	76	d	$84	d	100	v	$A5	v	118
M	$D4	M	77	e	$85	e	101	w	$A6	w	119
N	$D5	N	78	f	$86	f	102	x	$A7	x	120
O	$D6	O	79	g	$87	g	103	y	$A8	y	121
P	$D7	P	80	h	$88	h	104	z	$A9	z	122
Q	$D8	Q	81	i	$89	i	105				
R	$D9	R	82	j	$91	j	106				

RUSKÝ FONT

A : Znak vykreslený DIGIGRAFEM
B : Hodnota znaku v tabulce fontu DIGIGRAFU (hex)
C : Přiřazovací znak
D : Hodnota přiřazovacího znaku v tabulce ASCII

A	B	C	D	A	B	C	D	A	B	C	D
ъ	$75	₰	156	у	$AD	u	117	И	$CB	I	73
ю	$76	q	113	ж	$AE	h	104	Й	$CC	J	74
б	$78	b	98	в	$AF	v	118	Я	$CD	Å	143
ц	$80	c	99	ь	$B0	¥	157	Л	$CE	L	76
а	$81	a	97	ы	$B1	y	121	К	$D2	K	75
с	$83	s	115	з	$B2	z	122	М	$D4	M	77
е	$85	e	101	ш	$B3	¿	168	О	$D6	O	79
д	$8A	d	100	э	$B4	é	130	Р	$D7	R	82
ф	$8C	f	102	щ	$B5	w	119	П	$DC	P	80
г	$8D	g	103	ч	$B6	ç	135	Т	$E3	T	84
й	$90	j	106	Ъ	$B7	₰	145	Х	$E7	X	88
т	$94	t	116	Ю	$B8	Q	81	У	$EA	U	85
п	$95	p	112	Б	$BA	B	66	Ж	$EC	H	72
о	$96	o	111	Ц	$BB	C	67	Ь	$EE	Æ	146
р	$97	r	114	Д	$BC	D	68	Ы	$EF	Y	89
к	$9A	k	107	Ф	$BE	F	70	З	$FA	Z	90
л	$9B	l	108	Г	$BF	G	71	Ш	$FB	¢	155
м	$9C	m	109	А	$C1	A	65	Э	$FC	É	144
н	$9D	n	110	В	$C2	V	86	Щ	$FD	W	87
я	$A0	å	134	С	$C3	S	83	Ч	$FE	Ç	128
и	$A4	i	105	Е	$C5	E	69				
х	$A7	x	120	Н	$C8	N	78				

ŘECKÝ FONT

A : Znak vykreslený DIGIGRAFEM
B : Hodnota znaku v tabulce fontu DIGIGRAFU (hex)
C : Přiřazovací znak
D : Hodnota přiřazovacího znaku v tabulce ASCII

A	B	C	D	A	B	C	D	A	B	C	D
Δ	$01	D	68	ϑ	$11	v	118	o	$96	o	111
Θ	$02	V	86	ι	$12	i	105	Γ	$BF	C	67
Λ	$03	L	76	κ	$13	k	107	A	$C1	A	65
Ξ	$04	J	74	λ	$14	l	108	B	$C2	B	66
Π	$05	P	80	μ	$15	m	109	E	$C5	E	69
Σ	$06	S	83	ν	$16	n	110	H	$C8	H	72
Φ	$07	F	70	ξ	$17	j	106	I	$C9	I	73
Ψ	$08	W	87	π	$18	p	112	K	$D2	K	75
Ω	$09	G	71	ρ	$19	r	114	M	$D4	M	77
α	$0A	a	97	σ	$1A	s	115	N	$D5	N	78
β	$0B	b	98	τ	$1B	t	116	O	$D6	O	79
γ	$0C	c	99	υ	$1C	y	121	P	$D7	R	82
δ	$0D	d	100	φ	$1D	f	102	T	$E3	T	84
ε	$0E	e	101	χ	$1E	x	120	X	$E7	X	88
ζ	$0F	z	122	ψ	$1F	w	119	Y	$E8	Y	89
η	$10	h	104	ω	$20	g	103	Z	$E9	Z	90

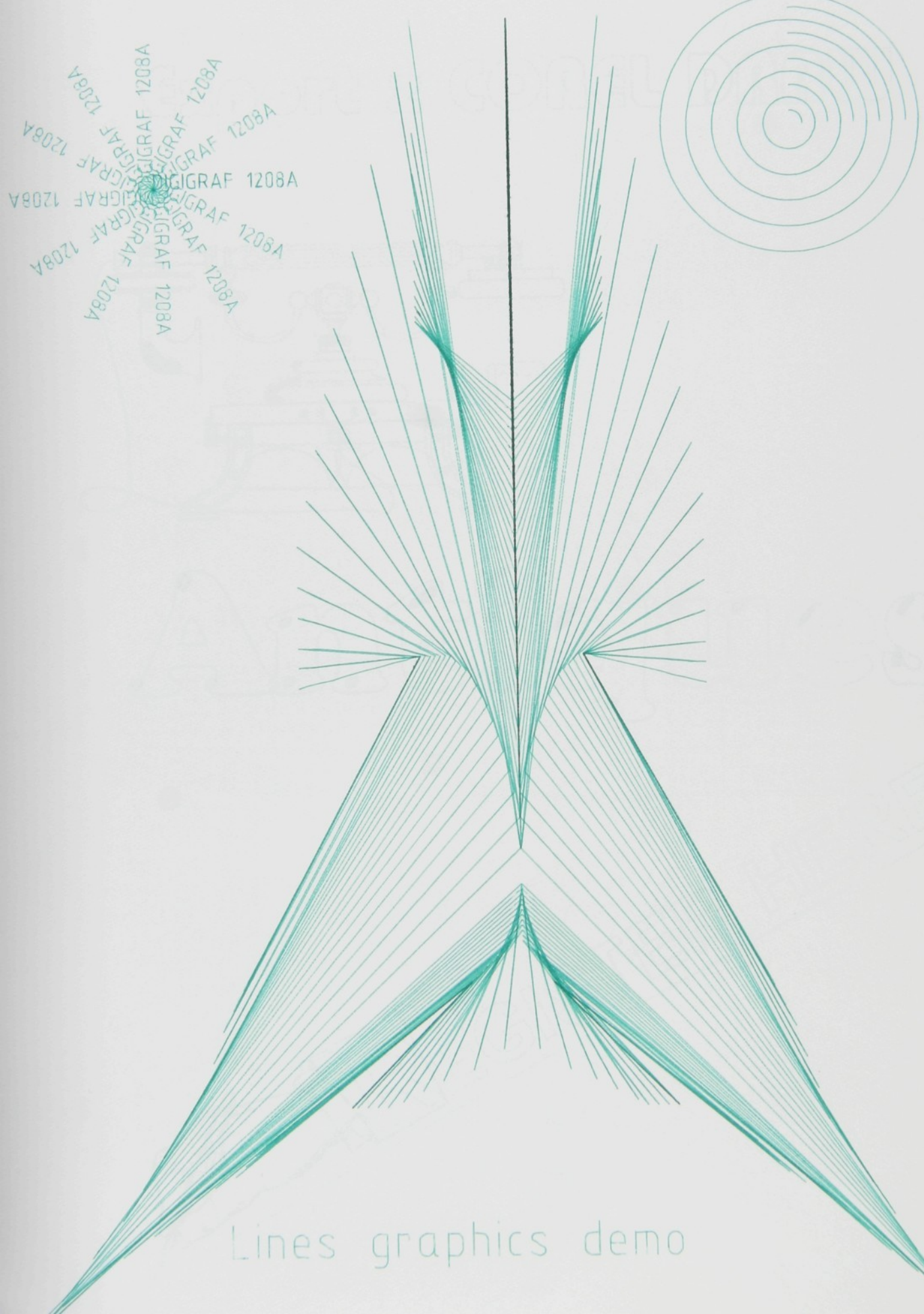
UNIVERSAL FONT

Pozn.: Je aktivní při jakékoliv volbě fontu.

A : Znak vykreslený DIGIGRAFEM
B : Hodnota znaku v tabulce fontu DIGIGRAFU (hex)
C : Přiřazovací znak
D : Hodnota přiřazovacího znaku v tabulce ASCII

A	B	C	D	A	B	C	D	A	B	C	D
	$40		32	°	$55	°	248	,	$6B	,	44
§	$41	$	36	‘	$56	‘	39	%	$6C	%	37
≐	$42	¦	124	‖	$58	‖	186	>	$6E	>	62
≙	$43		127	⊥	$59	⊥	193	?	$6F	?	63
~	$44	~	126	]	$5A	]	93	„	$71	_	95
≈	$45	≈	247	*	$5C	*	42	:	$7A	:	58
≦	$46	≤	243	)	$5D	)	41	=	$7E	=	61
⩽	$47	{	123	;	$5E	;	59	“	$7F	“	34
≧	$48	≥	242	-	$60	-	45	0	$F0	0	48
⩾	$49	}	125	/	$61	/	47	1	$F1	1	49
[	$4A	[	91	<	$62	#	35	2	$F2	2	50
.	$4B	.	46	≻	$63	`	96	3	$F3	3	51
<	$4C	<	60	⊳	$64	»	175	4	$F4	4	52
(	$4D	(	40	□	$65	≡	240	5	$F5	5	53
+	$4E	+	43	⌒	$66	^	94	6	$F6	6	54
!	$4F	!	33	∅	$67	@	64	7	$F7	7	55
&	$50	&	38	√	$68	√	251	8	$F8	8	56
±	$51	±	241	∫	$69	⌠	244	9	$F9	9	57

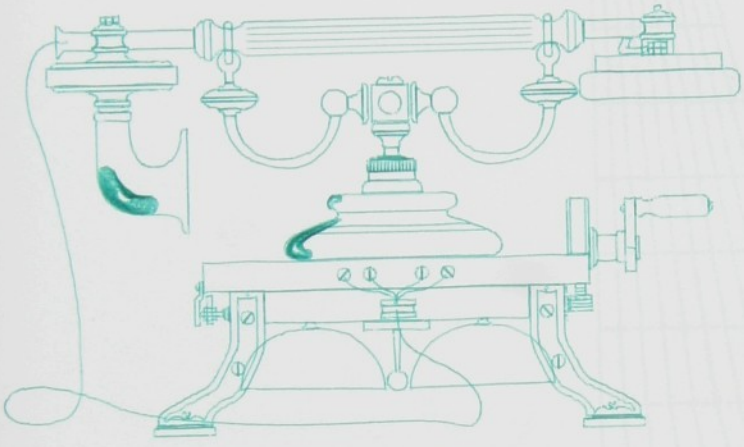
Priloha c.2



Lines graphics demo

Příloha č.3a

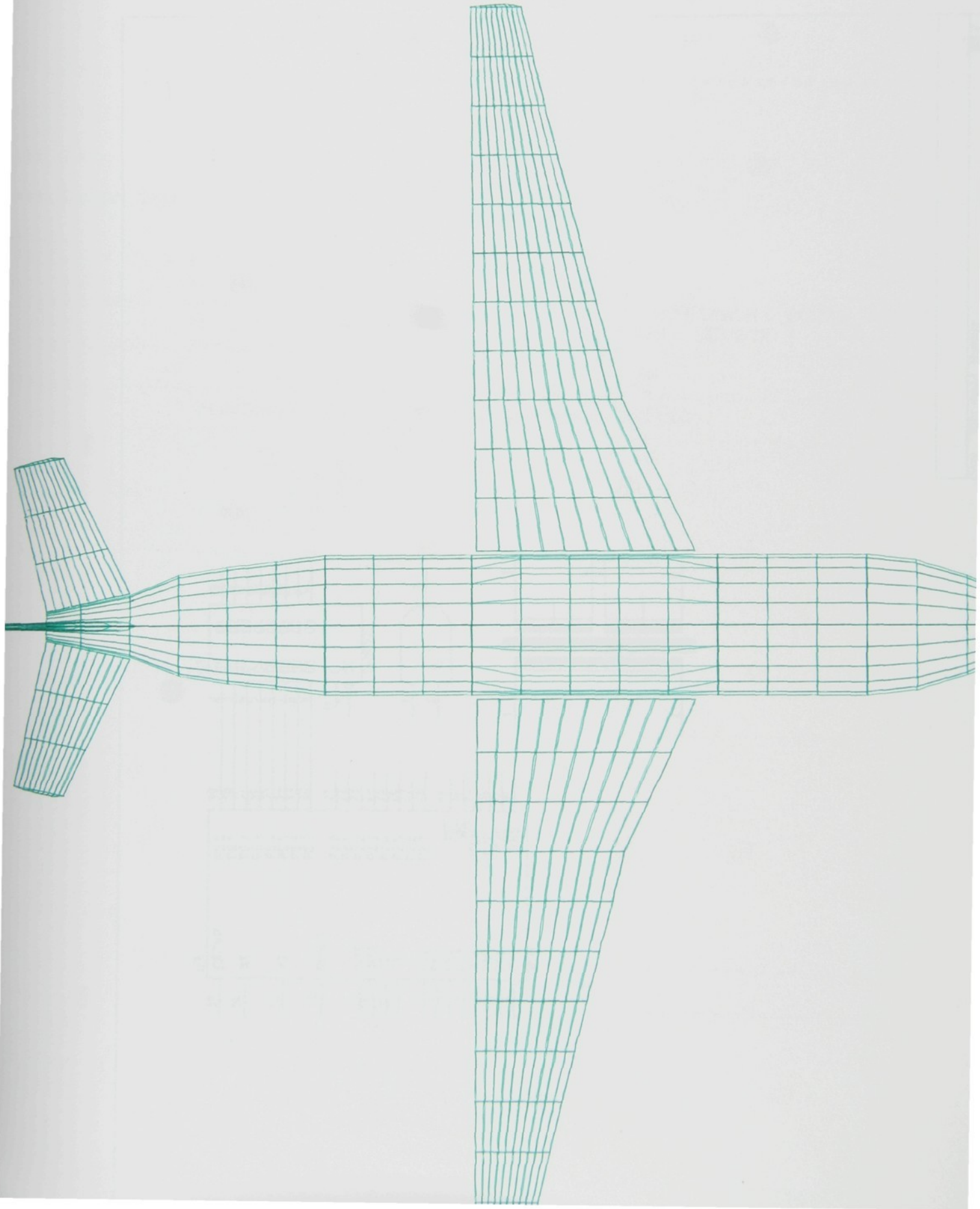
Export z COREL DRAW



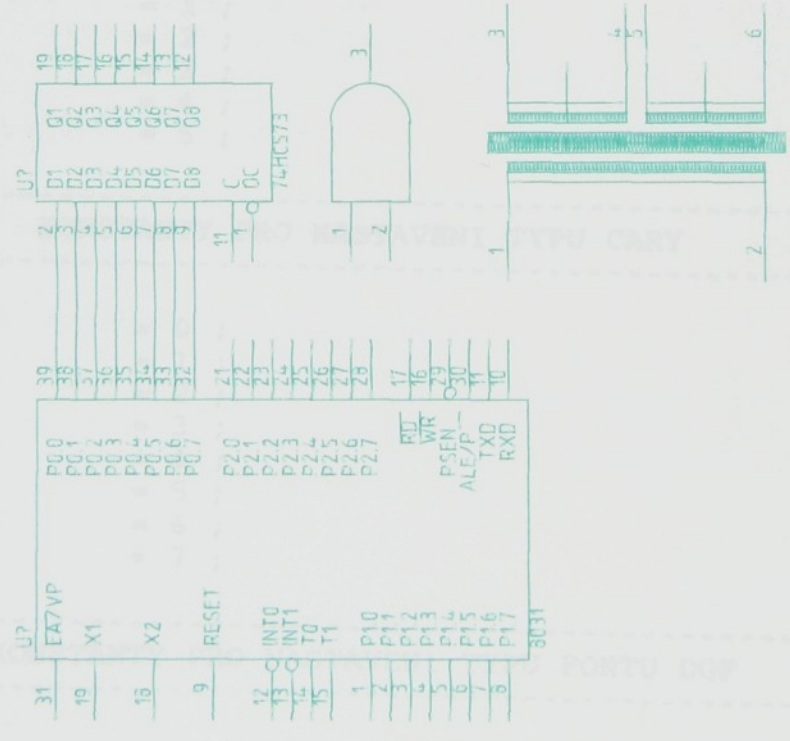
Antiques

PŘÍLOHA č. 3b

EXPORT z AUTOCADU



PŘÍLOHA č. 3c EXPORT z ORCADU



Priloha c.4 }

```
                    { VERZE 1.76 (c) MSM 10.12.1994 }
{*************************************************************************}
{*                                                                       *}
{*                    T U R B O   P A S C A L   U N I T                  *}
{*                                                                       *}
{*                    pro rizeni plotru  DIGIGRAF 1208A                  *}
{*                                                                       *}
{*                    MARTIN MORAVCIK (M.S.M.) (c) 1994                  *}
{*                                                                       *}
{*************************************************************************}

nit DIGIGRAF;

nterface

ses Graph,Dos;

onst
 DGF       : Boolean = False;
 Monitor   : Boolean = False;
 MaxPenNum : Byte    = 2 ;                    { MAXIMALNI POCET PISATEK }
 Path      : PathStr = 'noname.dgf';      { VYSTUPNI SOUBOR }
 Path_Graph : PathStr = 'c:\tp\BGI';

{-------------------------------------------------------------------------}
{               KONSTANTY PRO ORIENTACI A FORMAT PAPIRU                   }
{-------------------------------------------------------------------------}

 Vertical              = 1 ;
 Horizontal            = 2 ;

 A0                    = 0 ;
 A1                    = 1 ;
 A2                    = 2 ;
 A3                    = 3 ;
 A4                    = 4 ;
 A5                    = 5 ;

{-------------------------------------------------------------------------}
{               KONSTANTY PRO NASTAVENI TYPU CARY                         }
{-------------------------------------------------------------------------}

 SolidLn               = 0 ;
 DashedLn1             = 1 ;
 DashedLn2             = 2 ;
 DashedLn3             = 3 ;
 CenterLn1             = 4 ;
 CenterLn2             = 5 ;
 CenterLn3             = 6 ;
 CenterLn4             = 7 ;

{-------------------------------------------------------------------------}
{               KONSTANTY PRO NASTAVENI TYPU FONTU DGF                    }
{-------------------------------------------------------------------------}

 CSFont                = 1 ;
 RUFont                = 2 ;
 GRFont                = 3 ;



{-------------------------------------------------------------------------}
```

```
{ ------------------------ InitMonitor ---------------------------------
  ------------------------------------------------------------------------}

procedure InitMonitor;

{ ------------------------------------------------------------------------
  ------------------------ CloseMonitor --------------------------------
  ------------------------------------------------------------------------}

procedure CloseMonitor;

{ ------------------------------------------------------------------------
  ------------------------ InitDGF -------------------------------------
  ------------------------------------------------------------------------}

procedure InitDGF;

{ ------------------------------------------------------------------------
  ------------------------ CloseDGF ------------------------------------
  ------------------------------------------------------------------------}

procedure CloseDGF;

{ ------------------------------------------------------------------------
  ------------------------D_SetPath -----------------------------------
  ------------------------------------------------------------------------}

procedure D_SetPath ( Cesta : PathStr );

{ ------------------------------------------------------------------------
  ------------------------ D_SetPaperFormat ----------------------------
  ------------------------------------------------------------------------}

procedure D_SetPaperFormat ( PF : byte );

{ ------------------------------------------------------------------------
  ------------------------ D_GetPaperFormat ----------------------------
  ------------------------------------------------------------------------}

function D_GetPaperFormat : byte ;

{ ------------------------------------------------------------------------
  ------------------------ D_SetPaperOrient ----------------------------
  ------------------------------------------------------------------------}

procedure D_SetPaperOrient ( PO : byte );

{ ------------------------------------------------------------------------
  ------------------------ D_GetPaperOrient ----------------------------
  ------------------------------------------------------------------------}

function D_GetPaperOrient : byte ;

{ ------------------------------------------------------------------------
  ------------------------- D_GetMaxX ----------------------------------
  ------------------------------------------------------------------------}

function D_GetMaxX : Real;

{ ------------------------------------------------------------------------
  ------------------------- D_GetMaxY ----------------------------------
  ------------------------------------------------------------------------}

function D_GetMaxY : Real;
```

```
{ ------------------------------------------------------------------------ }
{ ---------------------------- D_SetColor ---------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetColor ( Barva : byte );

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_GetColor ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetColor : byte;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_GetMaxColor ------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetMaxColor : byte;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_SetPen ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_SetPen ( CisPis : byte );

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_GetPen ------------------------------------ }
{ ------------------------------------------------------------------------ }

function D_GetPen : byte;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_GetMaxPen --------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetMaxPen : byte;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_MoveRel ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_MoveRel ( dX,dY : Real );

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_MoveTo ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_MoveTo ( X,Y : Real );

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_Line -------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_Line ( X1,Y1,X2,Y2 : Real );

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_LineRel ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_LineRel ( dX,dY : Real );

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_LineTo ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_LineTo ( X,Y : Real );
```

```
{ ------------------------------------------------------------------------
{ ---------------------------- D_Circle ----------------------------------
{ ------------------------------------------------------------------------

procedure D_Circle ( Xs,Ys,R : Real );

{ ------------------------------------------------------------------------
{ ---------------------------- D_PutPixel --------------------------------
{ ------------------------------------------------------------------------

procedure D_PutPixel ( X,Y : Real ; Color : byte );

{ ------------------------------------------------------------------------
{ ---------------------------- D_Rectangle -------------------------------
{ ------------------------------------------------------------------------

procedure D_Rectangle ( X1,Y1,X2,Y2 : Real );

{ ------------------------------------------------------------------------
{ ------------------------- D_SetTextStyle -------------------------------
{ ------------------------------------------------------------------------

procedure D_SetTextStyle ( Font : Byte ; Direction,CharSize : real );

{ ------------------------------------------------------------------------
{ ---------------------- D_SetTextJustify ----------------------------------
{ ------------------------------------------------------------------------

procedure D_SetTextJustify ( Horiz : word );

{ ------------------------------------------------------------------------
{ ---------------------------- D_OutText ---------------------------------
{ ------------------------------------------------------------------------

procedure D_OutText ( Text : string );

{ ------------------------------------------------------------------------
{ ---------------------------- D_OutTextXY -------------------------------
{ ------------------------------------------------------------------------

procedure D_OutTextXY ( X,Y : Real ; Text : string );

{ ------------------------------------------------------------------------
{ ---------------------------- D_XAxis -----------------------------------
{ ------------------------------------------------------------------------

procedure D_XAxis ( Xs,Ys,Dist : Real );

{ ------------------------------------------------------------------------
{ ---------------------------- D_YAxis -----------------------------------
{ ------------------------------------------------------------------------

procedure D_YAxis ( Xs,Ys,Dist : Real );

{ ------------------------------------------------------------------------
{ ---------------------------- D_Arc -------------------------------------
{ ------------------------------------------------------------------------

procedure D_Arc ( Xs,Ys,StAngle,EndAngle,R : Real );

{ ------------------------------------------------------------------------
{ ---------------------------- D_Ellipse ---------------------------------
{ ------------------------------------------------------------------------
```

```
procedure D_Ellipse ( Xs,Ys,StAngle,EndAngle,Xradius,Yradius : Real );

{ ------------------------------------------------------------------------ }
{ ------------------------- D_SetLineStyle ------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetLineStyle ( PattNumb : byte );

{ ------------------------------------------------------------------------ }
{ ---------------------- D_GetLineSettings ------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetLineSettings : byte;

{ ------------------------------------------------------------------------ }
{ ---------------------- D_SetLineSettings ------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetLineSettings (PattNumb:byte ; L1,M1,L2,M2,L3,M3,L4,M4 : Real);

{ ------------------------------------------------------------------------ }
{ ------------------------- D_SetSpeed ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetSpeed( V0,V1 : Word );

{ ------------------------------------------------------------------------ }
{ ------------------------- D_SetAccel ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetAccel( A0,A1 : Word );

{ ------------------------------------------------------------------------ }
{ ------------------------ D_GetSpeedPU ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetSpeedPU : Word;

{ ------------------------------------------------------------------------ }
{ ------------------------ D_GetSpeedPD ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetSpeedPD : Word;

{ ------------------------------------------------------------------------ }
{ ------------------------ D_GetAccelPU ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetAccelPU : Word;

{ ------------------------------------------------------------------------ }
{ ------------------------ D_GetAccelPD ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetAccelPD : Word;


{ ------------------------------------------------------------------------ }
{ ------------------------ PREVOD DEC -> HEX ----------------------------- }
{ ------------------------------------------------------------------------ }

function DEC_HEX( dekr : real ) : string;

{ ------------------------------------------------------------------------ }
{ ------------------------ OpenMonitor ----------------------------------- }
```

```
---------------------------------------------------------------------------- }

rocedure OpenMonitor;

mplementation

onst

                    PRIRAZOVACI MATICE FONTU DGF                              }

 CSfnt   : array[1..2,1..52] of byte = ((65,66,67,68,69,70,71,72,73,74,75,
 76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,97,98,99,100,101,102,103,
 104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,
 122),($C1,$C2,$C3,$C4,$C5,$C6,$C7,$C8,$C9,$D1,$D2,$D3,$D4,$D5,$D6,$D7,
 $D8,$D9,$E2,$E3,$E4,$E5,$E6,$E7,$E8,$E9,$81,$82,$83,$84,$85,$86,$87,$88,
 $89,$91,$92,$93,$94,$95,$96,$97,$98,$99,$A2,$A3,$A4,$A5,$A6,$A7,$A8,$A9));

 RUfnt   : array[1..2,1..64] of byte = ((156,113,98,99,97,115,101,100,102,
 103,106,116,112,111,114,107,108,109,110,134,105,120,117,104,118,157,121,
 122,168,130,119,135,145,81,66,67,68,70,71,65,86,83,69,78,73,74,143,76,
 75,77,79,82,80,84,88,85,72,146,89,90,155,144,87,128),($75,$76,$78,$80,
 $81,$83,$85,$8A,$8C,$8D,$90,$94,$95,$96,$97,$9A,$9B,$9C,$9D,$A0,$A4,$A7,
 $AD,$AE,$AF,$B0,$B1,$B2,$B3,$B4,$B5,$B6,$B7,$B8,$BA,$BB,$BC,$BE,$BF,$C1,
 $C2,$C3,$C5,$C8,$CB,$CC,$CD,$CE,$D2,$D4,$D6,$D7,$DC,$E3,$E7,$EB,$EC,$EE,
 $EF,$FA,$FB,$FC,$FD,$FE));

 GRfnt   : array[1..2,1..48] of byte = ((68,86,76,74,80,83,70,87,71,97,98,
 99,100,101,122,104,118,105,107,108,109,110,106,112,114,115,116,121,102,
 120,119,103,111,67,65,66,69,72,73,75,77,78,79,82,84,88,89,90),($01,$02,
 $03,$04,$05,$06,$07,$08,$09,$0A,$0B,$0C,$0D,$0E,$0F,$10,$11,$12,$13,$14,
 $15,$16,$17,$18,$19,$1A,$1B,$1C,$1D,$1E,$1F,$20,$96,$BF,$C1,$C2,$C5,$C8,
 $C9,$D2,$D4,$D5,$D6,$D7,$E3,$E7,$E8,$E9));

 UNIfnt : array[1..2,1..54] of byte = ((32,36,124,127,126,247,243,123,242,
 125,91,46,60,40,43,33,38,241,248,39,186,193,93,42,41,59,45,47,35,96,175,
 240,94,64,251,244,44,37,62,63,95,58,61,34,48,49,50,51,52,53,54,55,56,57),
 ($40,$41,$42,$43,$44,$45,$46,$47,$48,$49,$4A,$4B,$4C,$4D,$4E,$4F,$50,$51,
 $55,$56,$58,$59,$5A,$5C,$5D,$5E,$60,$61,$62,$63,$64,$65,$66,$67,$68,$69,
 $6B,$6C,$6E,$6F,$71,$7A,$7E,$7F,$F0,$F1,$F2,$F3,$F4,$F5,$F6,$F7,$F8,$F9));

ar
 VYSTUP          : text ;        {  VYSTUP DO TEXTOVEHO SOUBORU                }
 PenNum          : byte ;        {  CISLO PISATKA = CISLO BARVY                }
 PapForm         : byte ;        {  FORMAT PAPIRU [A0..A5]                     }
 PapOrient       : byte ;        {  ORIENTACE PAPIRU [Horizont..Vertical]      }
 LineStyle       : byte ;        {  TYP CARY                                   }
 D_MaxX,D_MaxY   : real ;        {  MAXIMALNI ROZMERY KRESBY                   }
 SpeedPU         : word ;        {  RYCHLOST KR.HLAVY PRI ZVEDNUTEM PERU       }
 SpeedPD         : word ;        {  RYCHLOST KR.HLAVY PRI SPUSTENEM PERU       }
 AccelPU         : word ;        {  ZRYCHLENI KR.HLAVY PRI ZVEDNUTEM PERU      }
 AccelPD         : word ;        {  ZRYCHLENI KR.HLAVY PRI SPUSTENEM PERU      }
 Lx,Ly           : real ;        {  OBRAZOVE MERITKO !! DO IMPLEMENTATION      }
 Kon             : real ;
 VMaxY           : integer ;
 TextSettings    : TextSettingsType;


 ---------------------------------------------------------------------------- }
 ---------------------- I N I C I A L I Z A C E ----------------------------- }
 ---------------------------------------------------------------------------- }

rocedure D_INIT;

egin
 PenNum:=1;
```

```
 PapForm:=A0;
 D_MaxX:=841;
 D_MaxY:=1189;
 PapOrient:=Vertical;
 LineStyle:=SolidLn;
 TextSettings.Font:=CSFont;
 TextSettings.Direction:=0;
 TextSettings.Charsize:=5;
 TextSettings.Horiz:=LeftText;
 SpeedPU:=400;
 SpeedPD:=200;
 AccelPU:=6;
 AccelPD:=4;
nd;

 ------------------------------------------------------------------------
 ------------------------ OpenMonitor -----------------------------------
 ------------------------------------------------------------------------

rocedure OpenMonitor;

ar
 Gd,Gm     : Integer;
 Xasp,Yasp : Word;

egin
 Gd := Detect;
 InitGraph(Gd,Gm,Path_Graph);
 SetWriteMode(OrPut);
 GetAspectRatio(Xasp,Yasp);
 SetAspectRatio(Xasp,Xasp);
 Lx := D_GetMaxX/GetMaxX;
 Ly := D_GetMaxY/GetMaxY;
 if Lx < Ly then Lx := Ly
 else Ly := Lx;
 SetBkColor(Blue);
 SetLineStyle(SolidLn,0,ThickWidth);
 VMaxY := (GetMaxY+Trunc(D_GetMaxY/Ly)) div 2;
 Bar((GetMaxX-Trunc(D_GetMaxX/Lx)) div 2,
     (GetMaxY-Trunc(D_GetMaxY/LY)) div 2,
     (GetMaxX+Trunc(D_GetMaxX/Lx)) div 2,
                          VMaxY);
 SetColor(Red);
 Rectangle((GetMaxX-Trunc(D_GetMaxX/Lx)) div 2,
           (GetMaxY-Trunc(D_GetMaxY/LY)) div 2,
           (GetMaxX+Trunc(D_GetMaxX/Lx)) div 2,
                                VMaxY);
 SetLineStyle(SolidLn,0,NormWidth);
 SetViewPort((GetMaxX-Trunc(D_GetMaxX/Lx)) div 2,
             (GetMaxY-Trunc(D_GetMaxY/LY)) div 2,
             (GetMaxX+Trunc(D_GetMaxX/Lx)) div 2,
                               VMaxY,True);
 Kon := 1/Lx;
 SetTextStyle(SmallFont,0,Trunc(Kon*5));
 SetTextJustify(LeftText,BottomText);
 MoveTo(0,2*VMaxY-GetMaxY);
nd;

 ------------------------------------------------------------------------
 ------------------------ PREVOD DEC -> HEX -----------------------------
 ------------------------------------------------------------------------

unction DEC_HEX( Dekr : real ) : string;

ar
```

```
H     : string[8];
S     : string[1];
A,I   : Integer;
Dek,E : Longint;
Zap   : Boolean;

egin
E:=268435456;
Zap:=false;

if (Dekr>1189) or (-Dekr>1189) then
  begin
    if Monitor then CloseMonitor;
    if DGF then CloseDGF;
    WriteLn('ERROR : Nepripustna hodnota !!!');
    WriteLn;WriteLn('Stiskni ENTER ...');
    ReadLn;
    Halt(1);
  end;

Dek:=Round(Dekr*100);  { PREPOCET PARAMETRU VYRAZU Z mm NA SETINY mm }
if Dek<0 then
  begin
    Dek:=Dek+E;
    Zap:=True;
  end;
for I:=0 to 7 do
  begin
    A:=Trunc(Dek/E);
    case A of
      0..9 : begin
               Str(A,S);
               Insert(S,H,I+1);
             end;
      10   : Insert('A',H,I+1);
      11   : Insert('B',H,I+1);
      12   : Insert('C',H,I+1);
      13   : Insert('D',H,I+1);
      14   : Insert('E',H,I+1);
      15   : Insert('F',H,I+1);
    end;
    Dek:=Dek-(A*E);
    E:=E div 16;
  end;
  if Zap then
    begin
      Delete(H,1,1);
      Insert('F',H,1);
    end;
DEC_HEX:=H;
nd;


{ ---------------------------------------------------------------------------- }
{ ---------------------- PREVOD HEX -> DEC ----------------------------------- }
{ ---------------------------------------------------------------------------- }

unction HEX_DEC ( Hex : string) : Longint;

ar
 I,Code,C : Integer;
 A        : string;
 E,D      : Longint;

egin
```

```
 E:=268435456;
 D:=0;
 if (Length(Hex)<8) then
   begin                          { DOPLNENI POTREBNYCH NUL NA ZACATEK RETEZCE }
     for I:=1 to (8-length(Hex)) do Hex:=Concat('0',Hex);
   end;
 for I:=0 to 7 do
   begin
     A:=Copy(Hex,I+1,1);
     if (A>='0') and (A<='9') then Val(A,C,Code);
     if A='A' then C:=10;
     if A='B' then C:=11;
     if A='C' then C:=12;
     if A='D' then C:=13;
     if A='E' then C:=14;
     if A='F' then C:=15;
     D:=D+(C*E);
     E:=E div 16;
   end;
 HEX_DEC:=D;
nd;

 ------------------------- I M P O R T _ 0 ----------------------------- }
                    IMPORT SAMOTNE INSTRUKCE                              }
 ------------------------------------------------------------------------ }

rocedure IMPORT_0 (Instrukce : Byte );

egin
 Write(VYSTUP,Chr($2A),Chr(Instrukce));
nd;

 ------------------------- I M P O R T _ 1 ----------------------------- }
                IMPORT JEDNOHO 2-SLABICNEHO OPERANDU                      }
 ------------------------------------------------------------------------ }

rocedure IMPORT_1 (Op : string);

ar
 Op_1,Op_2 : Byte;

egin
 Op_1 := HEX_DEC(Copy(Op,5,2));
 Op_2 := HEX_DEC(Copy(Op,7,2));

 Write(VYSTUP,Chr(Op_1),Chr(Op_2));
nd;

 ------------------------- I M P O R T _ 2 ----------------------------- }
          IMPORT INSTRUKCE A JEDNOHO 2-SLABICNEHO OPERANDU                }
 ------------------------------------------------------------------------ }

rocedure IMPORT_2 (Instrukce : Byte ; Op : string);

ar
 Op_1,Op_2 : Byte;
```

```
egin
 Op_1 := HEX_DEC(Copy(Op,5,2));
 Op_2 := HEX_DEC(Copy(Op,7,2));

 Write(VYSTUP,Chr($2A),Chr(Instrukce),Chr(Op_1),Chr(Op_2));
nd;


 ------------------------- I M P O R T _ 4 ----------------------------- }

              IMPORT INSTRUKCE A DVOU 2-SLABICNYCH OPERANDU                }

 ------------------------------------------------------------------------ }

rocedure IMPORT_4 (Instrukce : Byte ; Op1,Op2 : string);

ar
 Op1_1,Op1_2,
 Op2_1,Op2_2 : Byte;

egin
 Op1_1 := HEX_DEC(Copy(Op1,5,2));
 Op1_2 := HEX_DEC(Copy(Op1,7,2));
 Op2_1 := HEX_DEC(Copy(Op2,5,2));
 Op2_2 := HEX_DEC(Copy(Op2,7,2));

 Write(VYSTUP,Chr($2A),Chr(Instrukce),Chr(Op1_1),Chr(Op1_2),
                                      Chr(Op2_1),Chr(Op2_2));
nd;


 ------------------------- I M P O R T _ 8 ----------------------------- }

              IMPORT INSTRUKCE A DVOU 4-SLABICNYCH OPERANDU                }

 ------------------------------------------------------------------------ }

rocedure IMPORT_8 (Instrukce : Byte ; Op1,Op2 : string);

ar
 Op1_1,Op1_2,Op1_3,Op1_4,
 Op2_1,Op2_2,Op2_3,Op2_4   : Byte;

egin
 Op1_1 := HEX_DEC(Copy(Op1,1,2));
 Op1_2 := HEX_DEC(Copy(Op1,3,2));
 Op1_3 := HEX_DEC(Copy(Op1,5,2));
 Op1_4 := HEX_DEC(Copy(Op1,7,2));

 Op2_1 := HEX_DEC(Copy(Op2,1,2));
 Op2_2 := HEX_DEC(Copy(Op2,3,2));
 Op2_3 := HEX_DEC(Copy(Op2,5,2));
 Op2_4 := HEX_DEC(Copy(Op2,7,2));

 Write(VYSTUP,Chr($2A),Chr(Instrukce),Chr(Op1_1),Chr(Op1_2),
                                      Chr(Op1_3),Chr(Op1_4),
                                      Chr(Op2_1),Chr(Op2_2),
                                      Chr(Op2_3),Chr(Op2_4));
d;


 ------------------------- I M P O R T _ 16 ---------------------------- }

              IMPORT INSTRUKCE A CTYR 4-SLABICNYCH OPERANDU                }
```

```
{ ------------------------------------------------------------------------ }

procedure IMPORT_16 (Instrukce : Byte ; Op1,Op2,Op3,Op4 : string);

var
  Op1_1,Op1_2,Op1_3,Op1_4,
  Op2_1,Op2_2,Op2_3,Op2_4,
  Op3_1,Op3_2,Op3_3,Op3_4,
  Op4_1,Op4_2,Op4_3,Op4_4   : Byte;

begin
  Op1_1 := HEX_DEC(Copy(Op1,1,2));
  Op1_2 := HEX_DEC(Copy(Op1,3,2));
  Op1_3 := HEX_DEC(Copy(Op1,5,2));
  Op1_4 := HEX_DEC(Copy(Op1,7,2));

  Op2_1 := HEX_DEC(Copy(Op2,1,2));
  Op2_2 := HEX_DEC(Copy(Op2,3,2));
  Op2_3 := HEX_DEC(Copy(Op2,5,2));
  Op2_4 := HEX_DEC(Copy(Op2,7,2));

  Op3_1 := HEX_DEC(Copy(Op3,1,2));
  Op3_2 := HEX_DEC(Copy(Op3,3,2));
  Op3_3 := HEX_DEC(Copy(Op3,5,2));
  Op3_4 := HEX_DEC(Copy(Op3,7,2));

  Op4_1 := HEX_DEC(Copy(Op4,1,2));
  Op4_2 := HEX_DEC(Copy(Op4,3,2));
  Op4_3 := HEX_DEC(Copy(Op4,5,2));
  Op4_4 := HEX_DEC(Copy(Op4,7,2));

  Write(VYSTUP,Chr($2A),Chr(Instrukce),Chr(Op1_1),Chr(Op1_2),
                                       Chr(Op1_3),Chr(Op1_4),
                                       Chr(Op2_1),Chr(Op2_2),
                                       Chr(Op2_3),Chr(Op2_4),
                                       Chr(Op3_1),Chr(Op3_2),
                                       Chr(Op3_3),Chr(Op3_4),
                                       Chr(Op4_1),Chr(Op4_2),
                                       Chr(Op4_3),Chr(Op4_4));
end;


{ ------------------------------------------------------------------------ }
{ ------------------------ InitMonitor ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure InitMonitor;

begin
  if Monitor=False then
    begin
      OpenMonitor;
      Monitor:=True;
    end;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------ CloseMonitor ---------------------------------- }
{ ------------------------------------------------------------------------ }

procedure CloseMonitor;

begin
```

```
  if Monitor=True then
    begin
      CloseGraph;
      Monitor:=False;
    end;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------- InitDGF ------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure InitDGF;

begin
  if DGF=False then
    begin
      Assign(VYSTUP,Path);
    {  Erase(VYSTUP); }
      Rewrite(VYSTUP);
      DGF:=True;
    end;
end;



{ ------------------------------------------------------------------------ }
{ ------------------------- CloseDGF ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure CloseDGF;
begin
     if DGF=True then begin
                         DGF:=False;
                         D_MoveTo(0,0);
                         Write(VYSTUP,Chr($1A)); {  KONEC PROGRAMU PRO DGF }
                         Close(VYSTUP);
                       end;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------D_SetPath ---------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetPath ( Cesta : PathStr );

begin
  Path:=Cesta;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------- D_SetPaperFormat ---------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetPaperFormat ( PF : byte );

begin

   if (PF>=0) and (PF<6) then
   begin
     PapForm:=PF;
     case PF of

       A0 : begin
              D_MaxX:=841;
```

```
                D_MaxY:=1189;
            end;
      A1 : begin
                D_MaxX:=594;
                D_MaxY:=841;
            end;
      A2 : begin
                D_MaxX:=420;
                D_MaxY:=594;
            end;
      A3 : begin
                D_MaxX:=297;
                D_MaxY:=420;
            end;
      A4 : begin
                D_MaxX:=210;
                D_MaxY:=297;
            end;
      A5 : begin
                D_MaxX:=148;
                D_MaxY:=210;
            end;

    end;
 end
 else
 begin
   if Monitor then CloseMonitor;
   if DGF then CloseDGF;
   WriteLn('ERROR : NEPRIPUSTNY FORMAT PAPIRU !!!');
   WriteLn;WriteLn('Stiskni ENTER ...');
   ReadLn;
   Halt(1);
 end;

nd;

 ------------------------------------------------------------------------------ }
 ------------------------- D_GetPaperFormat ---------------------------------- }
 ------------------------------------------------------------------------------ }

unction D_GetPaperFormat : Byte ;

egin
 D_GetPaperFormat := PapForm;
nd;

 ------------------------------------------------------------------------------ }
 ------------------------- D_SetPaperOrient ---------------------------------- }
 ------------------------------------------------------------------------------ }

rocedure D_SetPaperOrient ( PO : Byte );

ar
 PP : real;

egin
 if PO <> PapOrient then begin
                           PP:=D_MaxX;
                           D_MaxX:=D_MaxY;
                           D_MaxY:=PP;
                           PapOrient:=PO;
                         end;
nd;
```

```
{ ------------------------------------------------------------------------------
{ ------------------------ D_GetPaperOrient ------------------------------------
{ ------------------------------------------------------------------------------

function D_GetPaperOrient : Byte ;

begin
  D_GetPaperOrient := PapOrient;
end;

{ ------------------------------------------------------------------------------
{ ---------------------------- D_GetMaxX ---------------------------------------
{ ------------------------------------------------------------------------------

function D_GetMaxX : real;

begin
  D_GetMaxX:=D_MaxX;
end;

{ ------------------------------------------------------------------------------
{ ---------------------------- D_GetMaxY ---------------------------------------
{ ------------------------------------------------------------------------------

function D_GetMaxY : real;

begin
  D_GetMaxY:=D_MaxY;
end;

{ ------------------------------------------------------------------------------
{ ---------------------------- D_SetColor --------------------------------------
{ ------------------------------------------------------------------------------

procedure D_SetColor ( Barva : Byte );

begin
  if DGF then
    begin
      PenNum:=Barva;
      IMPORT_2($10,DEC_HEX(Barva/100));
    end;
  if Monitor then SetColor(Barva+2);
end;


{ ------------------------------------------------------------------------------
{ ---------------------------- D_SetPen ----------------------------------------
{ ------------------------------------------------------------------------------

procedure D_SetPen ( CisPis : Byte );

begin
  if DGF then
    begin
      PenNum:=CisPis;
      IMPORT_2($10,DEC_HEX(CisPis/100));
    end;
  if Monitor then
    begin
      if CisPis=1 then SetColor(Red);
      if CisPis=2 then SetColor(Green);
    end;
end;
```

```
{ ------------------------------------------------------------------------ }
{ ---------------------------- D_MoveRel ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_MoveRel ( dX,dY : real );

begin
  if DGF then
    begin
      IMPORT_8($01,DEC_HEX(dX),DEC_HEX(dY));
    end;
  if Monitor then MoveRel(Trunc(Kon*dX),-Trunc(Kon*dY))
end;


{ ------------------------------------------------------------------------ }
{ ---------------------------- D_MoveTo ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_MoveTo ( X,Y : real );

begin
  if DGF then
    begin
      IMPORT_8($03,DEC_HEX(X),DEC_HEX(Y));
    end;
  if Monitor then Moveto(Trunc(Kon*X),2*VMaxY-GetMaxY-Trunc(Kon*Y));
end;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_Line -------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_Line ( X1,Y1,X2,Y2 : real );


begin
  if DGF then
    begin
      IMPORT_8($03,DEC_HEX(X1),DEC_HEX(Y1));
      IMPORT_8($07,DEC_HEX(X2),DEC_HEX(Y2));
    end;
  if Monitor then Line(Trunc(Kon*X1),2*VMaxY-GetMaxY-Trunc(Kon*Y1),
                       Trunc(Kon*X2),2*VMaxY-GetMaxY-Trunc(Kon*Y2));
end;


{ ------------------------------------------------------------------------ }
{ ---------------------------- D_Circle ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_Circle ( Xs,Ys,R : real );

var
 X,Y : real ;

begin
 if DGF then
   begin
     X:=Xs+R;
     Y:=Ys;
     D_MoveTo(X,Y);
     IMPORT_16($0B,DEC_HEX(X),DEC_HEX(Y),DEC_HEX(Xs),DEC_HEX(Ys));
   end;
```

```
  if Monitor then Circle(Trunc(Kon*Xs),2*VMaxY-GetMaxY-Trunc(Kon*Ys),
                         Trunc(Kon*R));
end;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_GetColor ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetColor : Byte;

begin
  D_GetColor:=PenNum;
end;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_GetPen ------------------------------------ }
{ ------------------------------------------------------------------------ }

function D_GetPen : Byte;

begin
  D_GetPen:=PenNum;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------- D_GetMaxColor ---------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetMaxColor : Byte;

begin
  D_GetMaxColor:=MaxPenNum;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------- D_GetMaxPen ----------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetMaxPen : Byte;

begin
  D_GetMaxPen:=MaxPenNum;
end;

{ ------------------------------------------------------------------------ }
{ ---------------------------- D_LineRel ----------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_LineRel ( dX,dY : real );

begin
  if DGF then
    begin
      IMPORT_8($05,DEC_HEX(dX),DEC_HEX(dY));
    end;
  if Monitor then LineRel(Trunc(Kon*dX),-Trunc(Kon*dY))
end;

{ ------------------------------------------------------------------------ }
```

```
{ ---------------------------- D_LineTo ---------------------------------
{ -----------------------------------------------------------------------

procedure D_LineTo ( X,Y : real );

begin
  if DGF then
    begin
      IMPORT_8($07,DEC_HEX(X),DEC_HEX(Y));
    end;
  if Monitor then LineTo(Trunc(Kon*X),2*VMaxY-GetMaxY-Trunc(Kon*Y));
end;


{ ----------------------------------------------------------------------- }
{ ---------------------------- D_PutPixel ------------------------------- }
{ ----------------------------------------------------------------------- }

procedure D_PutPixel ( X,Y : real ; Color : Byte );

begin
  if DGF then
    begin
      if Color<>PenNum then          { ZMENA PISATKA NA POZADOVANOU HODNOTU }
      IMPORT_2($10,DEC_HEX(Color/100));

      IMPORT_8($03,DEC_HEX(X),DEC_HEX(Y));
      IMPORT_0($20);                 { OZNACENI BODU }

      if Color<>PenNum then          { ZMENA PISATKA NA PUVODNI HODNOTU }
      IMPORT_2($10,DEC_HEX(PenNum));
    end;
  if Monitor then PutPixel(Trunc(Kon*X),2*VMaxY-GetMaxY-Trunc(Kon*Y),Color+2)
end;


{ ----------------------------------------------------------------------- }
{ ---------------------------- D_Rectangle ------------------------------- }
{ ----------------------------------------------------------------------- }

procedure D_Rectangle ( X1,Y1,X2,Y2 : real );

begin
  if DGF then
    begin
      IMPORT_8($03,DEC_HEX(X1),DEC_HEX(Y1));
      IMPORT_8($07,DEC_HEX(X1),DEC_HEX(Y2));
      IMPORT_8($07,DEC_HEX(X2),DEC_HEX(Y2));
      IMPORT_8($07,DEC_HEX(X2),DEC_HEX(Y1));
      IMPORT_8($07,DEC_HEX(X1),DEC_HEX(Y1));
    end;
  if Monitor then Rectangle(Trunc(Kon*X1),2*VMaxY-GetMaxY-Trunc(Kon*Y1),
                            Trunc(Kon*X2),2*VMaxY-GetMaxY-Trunc(Kon*Y2));
end;

{ ----------------------------------------------------------------------- }
{ ---------------------------- D_SetTextStyle ---------------------------- }
{ ----------------------------------------------------------------------- }

procedure D_SetTextStyle ( Font : Byte ; Direction,CharSize : real );

var
  SinText,CosText : real;
```

```
begin
  if DGF then
    begin
      TextSettings.Font:=Font;
      TextSettings.Direction:=trunc(Direction);
      TextSettings.CharSize:=trunc(CharSize);
      SinText:=Sin(Direction*Pi/180)*16384;
      CosText:=Cos(Direction*Pi/180)*16384;
      IMPORT_4($1D,DEC_HEX(SinText/100),DEC_HEX(CosText/100));
      IMPORT_2($1C,DEC_HEX(CharSize));
    end;
  if Monitor then
      if Direction = 0 then
        SetTextStyle(SmallFont,HorizDir,trunc(Kon*CharSize/1.7))
      else
        SetTextStyle(SmallFont,VertDir,trunc(Kon*CharSize/1.7));

    {    SetUserCharSize(trunc(Kon*(18-CharSize)/0.9),trunc(Kon*CharSize/1.5)
                         trunc(Kon*(18-CharSize)/0.9),trunc(Kon*CharSize/1.5)

    }
end;

{ ------------------------------------------------------------------------ }
{ ---------------------- D_SetTextJustify ------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetTextJustify ( Horiz : Word );

begin
  if DGF then TextSettings.Horiz:=Horiz;
  if Monitor then SetTextJustify(Horiz,BottomText);
end;

{ ------------------------------------------------------------------------ }
{ ------------------------- D_OutText ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_OutText ( Text : string );

var
  I,J   : Word;
  Ok,Vk : Boolean;
  L,L1  : Byte;
begin
  if DGF then
    begin
      Vk:=False;
      L:=Length(Text);
      if L>88 then
      begin
        if Monitor then CloseMonitor;
        if DGF then CloseDGF;
        WriteLn('ERROR : DELKA RETEZCE MUSI BYT MENSI NEZ 88 !!!');
        WriteLn;WriteLn('Stiskni ENTER ...');
        ReadLn;
        Halt(1);
      end;
      L1:=L;
      if Odd(L) then
      begin                          { Doplneni na sudy pocet  }
        L1:=L+1;                     {    znaku v retezci      }
        Vk:=True;
      end;
      IMPORT_2($1F,DEC_HEX(L1/100));
```

```
    for I:=1 to L do
      begin
        Ok:=False;
        if TextSettings.Font=CSFont then
          begin
            for J:=1 to 52 do
              begin
                if Ord(Text[I])=CSfnt[1,J] then
                  begin
                    Write(VYSTUP,Chr(CSfnt[2,J]));
                    Ok:=True;
                  end;
              end;
          end;

        if TextSettings.Font=RUFont then
          begin
            for J:=1 to 64 do
              begin
                if Ord(Text[I])=RUfnt[1,J] then
                  begin
                    Write(VYSTUP,Chr(RUfnt[2,J]));
                    Ok:=True;
                  end;
              end;
          end;

        if TextSettings.Font=GRFont then
          begin
            for J:=1 to 48 do
              begin
                if Ord(Text[I])=GRfnt[1,J] then
                  begin
                    Write(VYSTUP,Chr(GRfnt[2,J]));
                    Ok:=True;
                  end;
              end;
          end;

        if Ok=False then
          begin
            for J:=1 to 54 do
              begin
                if Ord(Text[I])=UNIfnt[1,J] then
                  begin
                    Write(VYSTUP,Chr(UNIfnt[2,J]));
                    Ok:=True;
                  end;
              end;
            if Ok=False then
            Write(VYSTUP,Chr($70));
          end;
      end;
    if Vk then Write(VYSTUP,Chr($00));
  end;
if Monitor then OutText(Text);
d;

{------------------------------------------------------------------------------}
{------------------------- D_OutTextXY ----------------------------------------}
{------------------------------------------------------------------------------}

ocedure D_OutTextXY ( X,Y : real ; Text : string );

gin
```

```
  D_MoveTo(X,Y);
  D_OutText(Text);
end;


{ ------------------------------------------------------------------------
{ -------------------------- D_XAxis -------------------------------------
{ ------------------------------------------------------------------------

procedure D_XAxis ( Xs,Ys,Dist : real );

var
  I  : Integer;
  Xd : real;
begin
  D_MoveTo(Xs,Ys);I:=0;
  repeat
    Xd:=Xs+I*Dist;
    D_Line(Xd,Ys+0.01*D_GetMaxY,Xd,Ys-0.01*D_GetMaxY);
    I:=I+1;
  until (Xs+I*Dist) >= D_GetMaxX;
  D_Line(Xs,Ys,Xd,Ys);
  D_MoveTo(Xs,Ys);
end;


{ ------------------------------------------------------------------------
{ -------------------------- D_YAxis -------------------------------------
{ ------------------------------------------------------------------------

procedure D_YAxis ( Xs,Ys,Dist : real );

var
  I  : Integer;
  Yd : real;
begin
  D_MoveTo(Xs,Ys);I:=0;
  repeat
    Yd:=Ys+I*Dist;
    D_Line(Xs-0.01*D_GetMaxX,Yd,Xs+0.01*D_GetMaxX,Yd);
    I:=I+1;
  until (Ys+I*Dist) >= D_GetMaxY;
  D_Line(Xs,Ys,Xs,Yd);
  D_MoveTo(Xs,Ys);
end;


{ ------------------------------------------------------------------------
{ -------------------------- D_Arc ---------------------------------------
{ ------------------------------------------------------------------------

procedure D_Arc ( Xs,Ys,StAngle,EndAngle,R : real );

var
  X1,Y1,X2,Y2,PomAngle : real;

begin
  if DGF then
    begin
      X1:=Xs+R*Cos(StAngle*Pi/180);
      Y1:=Ys+R*Sin(StAngle*Pi/180);
      X2:=Xs+R*Cos(EndAngle*Pi/180);
      Y2:=Ys+R*Sin(EndAngle*Pi/180);
      D_MoveTo(X1,Y1);
```

```
      if StAngle<EndAngle then
        IMPORT_16($0B,DEC_HEX(X2),DEC_HEX(Y2),DEC_HEX(Xs),DEC_HEX(Ys));

      if StAngle>EndAngle then
        IMPORT_16($0F,DEC_HEX(X2),DEC_HEX(Y2),DEC_HEX(Xs),DEC_HEX(Ys));
    end;
  if Monitor then
    begin
      if StAngle>EndAngle then
        begin
          PomAngle:=StAngle;
          StAngle:=EndAngle;
          EndAngle:=PomAngle;
        end;
      Arc(Trunc(Kon*Xs),2*VMaxY-GetMaxY-Trunc(Kon*Ys),
          Trunc(StAngle),Trunc(EndAngle),Trunc(Kon*R));
    end;
end;


{ ------------------------------------------------------------------------ }
{ ------------------------- D_Ellipse ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_Ellipse ( Xs,Ys,StAngle,EndAngle,Xradius,Yradius : real );

var
  X,Y,Angle : real;
  A,Count   : integer;

begin
  Count:=0;
  for A:=round(StAngle) to round(EndAngle) do
    begin
      if A=round(EndAngle) then
      begin
        A:=round(EndAngle);
        Count:=4;
      end;
      X:=XRadius*cos(A*Pi/180);
      Y:=YRadius*sin(A*Pi/180);
      X:=X+Xs;
      Y:=Y+Ys;
      if A=round(StAngle) then D_Moveto(X,Y);
      Inc(Count);
      if Count=5 then
      begin                { JEMNOST KRESBY = 5 STUPNU }
        D_LineTo(X,Y);
        Count:=0;
      end;
    end;
nd;


{ ------------------------------------------------------------------------ }
{ ------------------------- D_SetLineStyle ------------------------------- }
{ ------------------------------------------------------------------------ }

rocedure D_SetLineStyle ( PattNumb : byte );

egin
  if DGF then
    begin
      LineStyle:=PattNumb;
```

```
      IMPORT_2($13,DEC_HEX(PattNumb/100));
    end;
  if Monitor then
    begin
      if PattNumb=0 then SetLineStyle(SolidLn,0,NormWidth);
      if (PattNumb>0) and (PattNumb<4) then
                        SetLineStyle(DashedLn,0,NormWidth);
      if (PattNumb>3) and (PattNumb<8) then
                        SetLineStyle(CenterLn,0,NormWidth);
    end;
end;

{ ------------------------------------------------------------------------ }
{ ---------------------- D_GetLineSettings ------------------------------- }
{ ------------------------------------------------------------------------ }

function D_GetLineSettings : byte;

begin
  D_GetLineSettings:=LineStyle;
end;

{ ------------------------------------------------------------------------ }
{ ---------------------- D_SetLineSettings ------------------------------- }
{ ------------------------------------------------------------------------ }

procedure D_SetLineSettings (PattNumb:byte ; L1,M1,L2,M2,L3,M3,L4,M4 : Real);

begin
  if DGF then
    begin
      IMPORT_0($14);
      IMPORT_1(DEC_HEX(PattNumb/100));
      IMPORT_1(DEC_HEX(L1));
      IMPORT_1(DEC_HEX(M1));
      IMPORT_1(DEC_HEX(L2));
      IMPORT_1(DEC_HEX(M2));
      IMPORT_1(DEC_HEX(L3));
      IMPORT_1(DEC_HEX(M3));
      IMPORT_1(DEC_HEX(L4));
      IMPORT_1(DEC_HEX(M4));
    end;
end;


{ ------------------------------------------------------------------------ }
{ ------------------------ D_SetSpeed ------------------------------------ }
{ ------------------------------------------------------------------------ }

procedure D_SetSpeed( V0,V1 : Word );

begin
  if DGF then
    begin
      if (V0>0) and (V0<701) then SpeedPU:=V0;
      if (V1>0) and (V1<701) then SpeedPD:=V1;
      IMPORT_4($11,DEC_HEX(V0/100),DEC_HEX(V1/100));     { mm/s }
    end;
end;


{ ------------------------------------------------------------------------ }
{ ------------------------ D_SetAccel ------------------------------------ }
{ ------------------------------------------------------------------------ }
```

```
procedure D_SetAccel( A0,A1 : Word );

begin
  if DGF then
    begin
      if (A0>0) and (A0<11) then AccelPU:=A0;
      if (A1>0) and (A1<11) then AccelPD:=A1;
      IMPORT_4($12,DEC_HEX(A0/100),DEC_HEX(A1/100));   { stupen zrychleni }
    end;
end;


{ ------------------------------------------------------------------------- }
{ ------------------------ D_GetSpeedPU ----------------------------------- }
{ ------------------------------------------------------------------------- }

function D_GetSpeedPU : Word;
begin
  D_GetSpeedPU:=SpeedPU;
end;

{ ------------------------------------------------------------------------- }
{ ------------------------ D_GetSpeedPD ----------------------------------- }
{ ------------------------------------------------------------------------- }

function D_GetSpeedPD : Word;
begin
  D_GetSpeedPD:=SpeedPD;
end;

{ ------------------------------------------------------------------------- }
{ ------------------------ D_GetAccelPU ----------------------------------- }
{ ------------------------------------------------------------------------- }

function D_GetAccelPU : Word;
begin
  D_GetAccelPU:=AccelPU;
end;

{ ------------------------------------------------------------------------- }
{ ------------------------ D_GetAccelPD ----------------------------------- }
{ ------------------------------------------------------------------------- }

function D_GetAccelPD : Word;
begin
  D_GetAccelPD:=AccelPD;
end;


{ ------------------------------------------------------------------------- }
{ ------------------------------------------------------------------------- }
{ --------------------- H L A V N I  P R O G R A M ------------------------ }
{ ------------------------------------------------------------------------- }
{ ------------------------------------------------------------------------- }

begin
  D_INIT;
end.
```

PŘÍLOHA č. 5

```
program Interpretr_HPGL_to_DGF;

uses DIGIGRAF,DOS,CRT;

const
  Path : PathStr = 'Noname.plt'; { Cesta na graf.soubor *.PLT }

  Vertical              = 1 ;     { Konstanty pro orientaci a format papiru }
  Horizontal            = 2 ;

  A0                    = 0 ;
  A1                    = 1 ;
  A2                    = 2 ;
  A3                    = 3 ;
  A4                    = 4 ;
  A5                    = 5 ;

type
  UDType  = (Up,Down);
  ModType = (AbsMod,RelMod);
var                HPGLfile : Text;
   X0,Y0,KonX,KonY,VZ,VZOld : Real;      { Pom. graf. konst. + Vyska znaku }
                 Xact,Yact : Real;       { Aktualni pozice pisatka         }
                       Pen : UDType;     { Stav pera (UP/DOWN)             }
                    KodGI  : string[2];  { Kod graficke instrukce          }
                    ParGI  : string;     { Parametr graficke instrukce     }
          Format,Orientace : Byte;       { Format a orientace papiru       }
               FXMax,FYMax : Integer;    { Max. rozmery papiru dle formatu }
                     Music : Integer;
                     Cesta : PathStr;    { Cesta na soubor *.PLT,*.000,*.DGF
                   Adresar : DirStr;
                     Jmeno : NameStr;
                   Pripona : ExtStr;
                       SSC : Boolean;    { Indikator provedeni intrukce SC }
                      SMod : ModType;    { Souradnicovy mod (Absol./Rel.   }
                      Term : Byte;       { Textovy terminator              }
                     Orcad : Boolean;
{ ------------------------------------------------------------------------
{ ------------------------------ Form ------------------------------------
{ ------------------------------------------------------------------------

procedure Form ( Fo : Byte ) ;

begin
     case Fo of

      A0 : begin
             if Orientace=Vertical then
               begin
                 FXMax:=841;
                 FYMax:=1189;
               end
             else
               begin
                 FXMax:=1189;
                 FYMax:=841;
               end;
           end;
      A1 : begin
             if Orientace=Vertical then
               begin
                 FXMax:=594;
                 FYMax:=841;
               end
             else
```

```
            begin
              FXMax:=841;
              FYMax:=594;
            end;
        end;
    A2 : begin
           if Orientace=Vertical then
             begin
               FXMax:=420;
               FYMax:=594;
             end
           else
             begin
               FXMax:=594;
               FYMax:=420;
             end;
        end;
    A3 : begin
           if Orientace=Vertical then
             begin
               FXMax:=297;
               FYMax:=420;
             end
           else
             begin
               FXMax:=420;
               FYMax:=297;
             end;
        end;
    A4 : begin
           if Orientace=Vertical then
             begin
               FXMax:=210;
               FYMax:=297;

             end
           else
             begin
               FXMax:=297;
               FYMax:=210;
             end;
        end;
    A5 : begin
           if Orientace=Vertical then
             begin
               FXMax:=148;
               FYMax:=210;
             end
           else
             begin
               FXMax:=210;
               FYMax:=148;
             end;
        end;

  end;
end;

{ ---------------------------------------------------------------------------
  ---------------------------- PocetPar -------------------------------------
  --------------------------------------------------------------------------- }

function PocetPar( RsGI : string ; Pozice : Integer ) : Byte;

var  J,N    : Integer;
```

```
    P,SP   : Byte;
begin
  ParGI:=Copy(RsGI,Pozice+2,Length(RsGI)-Pozice-1);
  SP:=0;
  if ParGI=';' then N:=0                          { 44 = , }
  else                                            { 32 =   }
  begin                                           { 43 = + }
    N:=1;                                         { 45 = - }
    SP:=32;
    for J:=1 to Length(ParGI) do
      begin
        P:=Ord(ParGI[J]);
        if (P=44) or (P=32) or ((P=43) and (SP<>32)) or ((P=45) and (SP<>32))
          then Inc(N);
        if (P=45) and (SP=44) then Dec(N);
        SP:=P;
      end;
  end;
  PocetPar:=N;
end;

{ ------------------------------------------------------------------------- }
{ ------------------------------ Par -------------------------------------- }
{ ------------------------------------------------------------------------- }

function Par ( ParNum : Integer ) : real;

var Code,I,J,K,P : Integer;
          SP,DP : Byte;
          Param : String;
           ParI : real;
begin
  SP:=32;
  J:=1;
  for I:=1 to ParNum do
    begin
      Param:='';
      K:=1;
      P:=Ord(ParGI[J]);
      DP:=Ord(ParGI[J+1]);

      repeat
            Insert(ParGI[J],Param,K);
            Inc(J);
            Inc(K);
            SP:=P;
            P:=Ord(ParGI[J]);
            DP:=Ord(ParGI[J+1]);

      until (P=59) or ((P=44) and (DP<>45)) or (P=32)
                   or ((P=43) and (SP<>32)) or ((P=45) and (SP<>32));


    end;
  if Param[1]=',' then Delete(Param,1,1);
  if Param[Length(Param)]=',' then Delete(Param,Length(Param),1);
  Val(Param,ParI,Code);
  Par:=ParI;
end;

  ------------------------------------------------------------------------- }
  ------------------------------ AA --------------------------------------- }
  ------------------------------------------------------------------------- }

procedure AA ( RG : string ; Od : Integer ) ;
```

```
var          PocPar : Byte;
       Alfa1,Alfa2 : real;
           Xs,Ys,R : real;
begin

  PocPar:=PocetPar(RG,Od);
  Xs:=(X0+Par(1))/KonX;
  Ys:=(Y0+Par(2))/KonY;
  if Xact=Xs then Alfa1:=90
  else
    Alfa1:=ArcTan((Yact-Ys)/(Xact-Xs))*180/Pi;
  Alfa2:=Alfa1+Par(3);
  R:=Sqrt(Sqr(Xact-Xs)+Sqr(Yact-Ys));

  D_Arc(Xs,Ys,Alfa1,Alfa2,R);
  Xact:=Xs+R*Cos(Alfa2*Pi/180);
  Yact:=Ys+R*Sin(Alfa2*Pi/180);
  D_MoveTo(Xact,Yact);

end;

{ ------------------------------------------------------------------------- }
{ ------------------------------ AR ----------------------------------------- }
{ ------------------------------------------------------------------------- }

procedure AR ( RG : string ; Od : Integer ) ;

var          PocPar : Byte;
       Alfa1,Alfa2 : real;
           Xs,Ys,R : real;
begin
  PocPar:=PocetPar(RG,Od);
  Xs:=Xact+(Par(1))/KonX;
  Ys:=Yact+(Par(2))/KonY;
  if Xact=Xs then Alfa1:=90
  else
    Alfa1:=ArcTan((Yact-Ys)/(Xact-Xs))*180/Pi;
  Alfa2:=Alfa1+Par(3);
  R:=Sqrt(Sqr(Xact-Xs)+Sqr(Yact-Ys));

  D_Arc(Xs,Ys,Alfa1,Alfa2,R);
  Xact:=Xs+R*Cos(Alfa2*Pi/180);
  Yact:=Ys+R*Sin(Alfa2*Pi/180);
  D_MoveTo(Xact,Yact);
end;

{ ------------------------------------------------------------------------- }
{ ------------------------------ CI ----------------------------------------- }
{ ------------------------------------------------------------------------- }

procedure CI ( RG : string ; Od : Integer ) ;

var   PocPar : Byte;
         R : Real;
begin
  PocPar:=PocetPar(RG,Od);
  R:=Par(1)/KonX;
  D_Circle(Xact,Yact,R);
  D_MoveTo(Xact,Yact);
end;


{ ------------------------------------------------------------------------- }
{ ------------------------------ DI ----------------------------------------- }
```

```
{ ------------------------------------------------------------------------

procedure DI ( RG : string ; Od : Integer ) ;

var   PocPar : Byte;
      Alfa   : Real;
begin
  PocPar:=PocetPar(RG,Od);
  if PocPar>0 then
    begin
      Alfa:=(ArcTan(Par(2)/Par(1)))*180/Pi;
      D_SetTextStyle(CSFont,Alfa,VZ);
    end;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------------ LT ------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure LT ( RG : string ; Od : Integer ) ;

var   PocPar : Byte;
          C : Integer;
begin
  PocPar:=PocetPar(RG,Od);
  if PocPar>0 then
    begin
      C:=Trunc(Par(1));
      case C of
        1 : D_SetLineStyle(3);   { ..... }
        2 : D_SetLineStyle(1);   { - - - - - - }
        3 : D_SetLineStyle(2);   { -- -- -- -- }
        4 : D_SetLineStyle(4);   { -.-.-.-.-.-. }
        5 : D_SetLineStyle(4);   { -.-.-.-.-.-. }
        6 : D_SetLineStyle(5);   { -..-..-..-.. }
      end;
      if Par(1)<0 then D_SetLineStyle(0);
    end
  else
  D_SetLineStyle(0);
end;

{ ------------------------------------------------------------------------ }
{ ------------------------------ PD ------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure PD ( RG : string ; Od : Integer ) ;

var    PocPar : Byte;
         I,II : Integer;
   dX,dY,X,Y : real;

begin
  PocPar:=PocetPar(RG,Od);
  Pen:=Down;
  I:=1;
  if PocPar>0 then
    begin
      for II:=1 to (PocPar div 2) do
        begin
          if SMod=AbsMod then
            begin
              X:=Abs((X0+Par(I))/KonX);
              Y:=Abs((Y0+Par(I+1))/KonY);
```

```
            D_LineTo(X,Y);
            Xact:=X;
            Yact:=Y;
          end;
        if SMod=RelMod then
          begin
            dX:=Par(I)/KonX;
            dY:=Par(I+1)/KonY;
            D_LineRel(dX,dY);
            Xact:=Xact+dX;
            Yact:=Yact+dY;
          end;
        Inc(I,2);
      end;
  end;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------------ PA ------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure PA ( RG : string ; Od : Integer ) ;

var PocPar  : Byte;
        I   : Integer;
      X,Y   : real;
begin
  PocPar:=PocetPar(RG,Od);
  if PocPar>0 then
    begin
      for I:=1 to (PocPar div 2) do
        begin
          X:=Abs((X0+Par(I))/KonX);
          Y:=Abs((Y0+Par(I+1))/KonY);
          if Pen=Up then D_MoveTo(X,Y);
          if Pen=Down then D_LineTo(X,Y);
          Xact:=X;
          Yact:=Y;
        end;
    end;
                { Nastaveni absolutniho souradnicoveho modu }
  SMod:=AbsMod;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------------ PR ------------------------------------- }
{ ------------------------------------------------------------------------ }

procedure PR ( RG : string ; Od : Integer ) ;

var PocPar  : Byte;
        I   : Integer;
    dX,dY   : real;
begin
  PocPar:=PocetPar(RG,Od);
  if PocPar>0 then
    begin
      for I:=1 to (PocPar div 2) do
        begin
          dX:=Par(I)/KonX;
          dY:=Par(I+1)/KonY;
          if Pen=Up then D_MoveRel(dX,dY);
          if Pen=Down then D_LineRel(dX,dY);
          Xact:=Xact+dX;
          Yact:=Yact+dY;
```

```
      end;
    end;
              { Nastaveni relativniho souradnicoveho modu }
  SMod:=RelMod;
end;

{ ------------------------------------------------------------------------- }
{ ------------------------------ PU ----------------------------------------- }
{ ------------------------------------------------------------------------- }

procedure PU ( RG : string ; Od : Integer ) ;

var    PocPar  : Byte;
          I   : Integer;
    dX,dY,X,Y  : real;
begin
  PocPar:=PocetPar(RG,Od);
  Pen:=Up;
  if PocPar>0 then
    begin
      for I:=1 to (PocPar div 2) do
        begin
          if SMod=AbsMod then
            begin
              X:=Abs((X0+Par(I))/KonX);
              Y:=Abs((Y0+Par(I+1))/KonY);
              D_MoveTo(X,Y);
              Xact:=X;
              Yact:=Y;
            end;
          if SMod=RelMod then
            begin
              dX:=Par(I)/KonX;
              dY:=Par(I+1)/KonY;
              D_MoveRel(dX,dY);
              Xact:=Xact+dX;
              Yact:=Yact+dY;
              end;
        end;
    end;
end;


{ ------------------------------------------------------------------------- }
{ ------------------------------ SC ----------------------------------------- }
{ ------------------------------------------------------------------------- }

procedure SC ( RG : string ; Od : Integer ) ;

var PocPar  : Byte;
       I   : Integer;
    Xmin,Xmax,Ymin,Ymax  : real;
begin
  PocPar:=PocetPar(RG,Od);
  Form(Format);              { Zjisteni rozmeru papiru ze zadaneho formatu }
  if PocPar>0 then
    begin
      Xmin:=Par(1);
      Xmax:=Par(2);
      Ymin:=Par(3);
      Ymax:=Par(4);
      if Xmin<0 then X0:=(Abs(Xmin)+Abs(Xmax))/2;
      if Ymin<0 then Y0:=(Abs(Ymin)+Abs(Ymax))/2;
      if Xmin>=0 then X0:=Xmin;
      if Ymin>=0 then Y0:=Ymin;
```

```
      KonX:=(Abs(Xmin)+Abs(Xmax))/FXMax;
      KonY:=(Abs(Ymin)+Abs(Ymax))/FYMax;
    end
  else
    begin
      X0:=FXMax*20; { Nastaveni uzivatelskeho meritka pro SC bez parametru }
      Y0:=FYMax*20; {       Xmin=-Max.FormatX*20,Xmax=Max.FormatX*20       }
      KonX:=40;     {       Ymin=-Max.FormatY*20,Ymax=Max.FormatY*20       }
      KonY:=40;     {                Konstanta meritka = 40                }
    end;
  if Orcad then
    begin
      X0:=0;
      Y0:=0;
      KonX:=40;
      KonX:=40;
    end;
end;

{ ------------------------------------------------------------------------------
{ ------------------------------ SI ------------------------------------------
{ ------------------------------------------------------------------------------

procedure SI ( RG : string ; Od : Integer ) ;

var PocPar  : Byte;
        VP  : Real;
begin
  PocPar:=PocetPar(RG,Od);
  if Par(2)<>VZOld then
    begin
      if PocPar>0 then
        begin
          VP:=Par(2)*10;
          if VP>2 then VP:=2;    { Minimalni vyska znaku je 2 mm !!! }
          D_SetTextStyle(CSfont,0,VP);
          VZOld:=VP;
        end
        else
        begin
          D_SetTextStyle(CSfont,0,VZ);
          VZOld:=VZ;
        end;
    end;
end;

{ ------------------------------------------------------------------------------
{ ------------------------------ SP ------------------------------------------
{ ------------------------------------------------------------------------------

procedure SP ( RG : string ; Od : Integer ) ;

var PocPar  : Byte;

begin
  PocPar:=PocetPar(RG,Od);
  if PocPar>0 then
    D_SetPen(Trunc(Par(1)))
  else
    D_SetPen(1);
end;

{ ------------------------------------------------------------------------------
{ ------------------------------ VS ------------------------------------------
{ ------------------------------------------------------------------------------
```

```
procedure VS ( RG : string ; Od : Integer ) ;

var PocPar  : Byte;

begin
  PocPar:=PocetPar(RG,Od);
  if PocPar>0 then
    D_SetSpeed(400,Trunc(Par(1)*10))
  else
    D_SetSpeed(400,200);
end;

{ ------------------------------------------------------------------------
{ ------------------------------ LB ---------------------------------------
{ ------------------------------------------------------------------------

procedure LB ( RG : string ; Od : Integer ) ;

begin
  ParGI:=Copy(RG,Od+2,Length(RG)-Od-1);
  D_OutText(ParGI);
end;

{ ------------------------------------------------------------------------
{ ------------------------------ DF ---------------------------------------
{ ------------------------------------------------------------------------

procedure DF ;

begin
  SSC:=False;  { Indikator provedeni intrukce SC pro nastav. uziv.meritka }
  VZ:=0.5;
  VZOld:=0.5;  { Promenna pro ulozeni vysky znaku                          }
  SMod:=AbsMod;{ Nastaveni absolutniho souradnicoveho modu                 }
  Pen:=Up;
  SC('SC;',1); { Nastaveni Meritka pri eventualni absenci                  }
               { instrukce SC pro nastaveni uzivatelskeho meritka          }
  D_SetLineStyle(0); { Typ cary : Plna                                     }
  Term:=3;     { Textovy exterminator : ETX                                }
end;

{ ------------------------------------------------------------------------
{ ------------------------------ DT ---------------------------------------
{ ------------------------------------------------------------------------

procedure DT ( RG : string ; Od : Integer ) ;

begin
  Term:=Ord(RG[Od+2]);
end;

{ ------------------------------------------------------------------------
{ ------------------------- DekoderGrIn -----------------------------------
{ ------------------------------------------------------------------------

procedure DekoderGrIn ( RGI : string );


var DekOk          : Boolean;
    I,L            : Byte;
begin
  KodGI:='';
  L:=Length(RGI);
  if L > 1 then
```

```
begin
  I:=0;
  DekOk:=False;
  while (I < L-1)  and (DekOk=False) do
    begin
      Inc(I);
      KodGI:=Copy(RGI,I,2);
      ParGI:='';
      if KodGI='AA' then
        begin
          DekOk:=True;
          AA(RGI,I);
        end;
      if KodGI='AR' then
        begin
          DekOk:=True;
          AR(RGI,I);
        end;
      if KodGI='CA' then    { Ignorovana }
        begin
          DekOk:=True;
        end;
      if KodGI='CI' then
        begin
          CI(RGI,I);
          DekOk:=True;
        end;
      if KodGI='CP' then
        begin
          DekOk:=True;
        end;
      if KodGI='CS' then    { Ignorovana }
        begin
          DekOk:=True;
        end;
      if KodGI='DC' then    { Ignorovana }
        begin
          DekOk:=True;
        end;
      if KodGI='DF' then
        begin
          DekOk:=True;
          DF;
        end;
      if KodGI='DI' then
        begin
          DekOk:=True;
          DI(RGI,I);
        end;
      if KodGI='DP' then    { Ignorovana }
        begin
          DekOk:=True;
        end;
      if KodGI='DR' then
        begin
          DekOk:=True;
          DI(RGI,I);
        end;
      if KodGI='DT' then
        begin
          DekOk:=True;
        end;
      if KodGI='IM' then    { Ignorovana }
        begin
          DekOk:=True;
```

```
  end;
if KodGI='IN' then
  begin
    DekOk:=True;
    DF;
  end;
if KodGI='IP' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='IW' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='LB' then
  begin
    DekOk:=True;
    LB(RGI,I);
  end;
if KodGI='LT' then
  begin
    DekOk:=True;
    LT(RGI,I);
  end;
if KodGI='OA' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OC' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OD' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OE' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OF' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OH' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OI' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OO' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OP' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OS' then    { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='OW' then    { Ignorovana }
```

```
  begin
    DekOk:=True;
  end;
if KodGI='PA' then
  begin
    DekOk:=True;
    PA(RGI,I);
  end;
if KodGI='PD' then
  begin
    DekOk:=True;
    PD(RGI,I);
  end;
if KodGI='PR' then
  begin
    DekOk:=True;
    PR(RGI,I);
  end;
if KodGI='PS' then   { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='PU' then
  begin
    DekOk:=True;
    PU(RGI,I);
  end;
if KodGI='RO' then   { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='SA' then   { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='SC' then
  begin
    DekOk:=True;
    SSC:=True;
    SC(RGI,I);
  end;
if KodGI='SI' then
  begin
    DekOk:=True;
    SI(RGI,I);
  end;
if KodGI='SL' then   { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='SM' then   { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='SP' then
  begin
    DekOk:=True;
    SP(RGI,I);
  end;
if KodGI='SR' then   { Ignorovana }
  begin
    DekOk:=True;
  end;
if KodGI='SS' then   { Ignorovana }
  begin
```

```
            DekOk:=True;
          end;
        if KodGI='TL' then    { Ignorovana }
          begin
            DekOk:=True;
          end;
        if KodGI='UC' then    { Ignorovana }
          begin
            DekOk:=True;
          end;
        if KodGI='VS' then
          begin
            DekOk:=True;
          end;
        if KodGI='XT' then    { Ignorovana }
          begin
            DekOk:=True;
          end;
        if KodGI='YT' then    { Ignorovana }
          begin
            DekOk:=True;
          end;
      end;
  end;
end;

{ ------------------------------------------------------------------------ }
{ ------------------------- CTIHPGLfile ---------------------------------- }
{ ------------------------------------------------------------------------ }

procedure CtiHPGLfile( var Soub : Text );

var X          : char;
    I          : Integer;
    RsGI       : string;
begin
  repeat
    RsGI:='';
     I:=0;
    repeat
      read(HPGLfile,X);
      if Ord(X)>31 then      { Vyrad ridici znaky z retezce }
        begin
          Inc(I);
          insert(X,RsGI,I);
        end;
    until ((Ord(X)=59) or (Ord(X)=Term));
    DekoderGrIn(RsGI);
  until EOF(HPGLfile) ;
end;

{ ------------------------------------------------------------------------ }
{ ------------------ H L A V N I   P R O G R A M ------------------------- }
{ ------------------------------------------------------------------------ }

begin

  ClrScr;
  WriteLn('------------------------------------------------------------------
  WriteLn;
  WriteLn('                        INTERPRETR HPGL  ->  DGF  ');
  WriteLn;
  WriteLn('                              (c) 1994 M.S.M.  ');
  WriteLn;
  WriteLn('------------------------------------------------------------------
```

```
  WriteLn;

  WriteLn('Zadej cestu a jmeno interpretovaneho souboru (*.PLT,*.000)');
  Write(':');ReadLn(Path);
  WriteLn;
  Write('Zadej format papiru (A0=0 .. A5=5):');
  ReadLn(Format);
  WriteLn;
  Write('Zadej orientaci papiru (Vertical=1 , Horizontal=2) :');
  ReadLn(Orientace);
  FSplit(Path,Adresar,Jmeno,Pripona);
  Orcad:=False;
  if Pripona='.000' then Orcad:=True;
  if Jmeno = '' then Jmeno:='NONAME';
  if (Pripona = '.plt') or (Pripona = '.PLT') or (Pripona = '.000') then
    Pripona:='.DGF'
  else
    begin
      WriteLn('   Kompilovany soubor musi mit priponu *.PLT nebo *.000 !!! ')
      WriteLn;WriteLn('Stiskni klavesu <ENTER> ....');
      ReadLn;
      Halt(1);
    end;
  Cesta := Adresar + Jmeno + Pripona;
  D_SetPath(Cesta);
  Assign(HPGLfile,Path);
  Reset(HPGLfile);
  WriteLn('-------------------------------------------------------------------
  DF;
  D_SetPaperFormat(Format);
  D_SetPaperOrient(Orientace);
  InitDGF;       {      Inicializace DIGIGRAFU  ( Tisk do souboru *.DGF)     }
  InitMonitor;   {    Inicializace MONITORU   ( Kontrolni tisk na monitoru)  }
  CtiHPGLfile(HPGLfile);   {       Dekodovani souboru *.PLT                  }
  for Music:=1 to 5 do     {           Dokonceni tisku                       }
    begin
      Sound(600);
      Delay(90);
      Sound(1000);
      Delay(90);
      Sound(1400);
      Delay(90);
    end;
  NoSound;
  readln;
  Close(HPGLfile);

  CloseMonitor;
  CloseDGF;

end.
```