ICOT Technical Report: TM-0025

TM-0025

GP—PRO
Graphic Display Control Library
written in Prolog

清水肇

August, 1983

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ICOT TechnicalMemorandum: TM-0025

8-31-83 TM-0025

GP-PRO

Graphic Display Control Library written in Prolog

第1研究室 清水 肇

ICOT
〒108 東京都港区三田1丁目4－28 三田国際ビル21階
☎ (03) 456－3195
telex ICOT J32964

Institute for New Generation Computer Technology

目次

1　概要

本MEMOでは、PROLOGからGraphic Display (SEILLAC-3) を制御する為に開発した、PROLOGのパッケージ GP-PRO (Graphic Package in PROlog) について述べる。GP-PROはFORTRAN で記述されたSAILと同等の機能をもつ。

1.1　起動と終了

```
@ttyini.exe     ......    SEILLAC の接続
      ?37

@pim:gp-pro.exe
又は
@pim:tgp-pro.exe  ( DEBUG TOOLを含むバージョン )

  | ?-           (in prolog)



  | ?-halt.

@restty.exe     ......    SEILLAC の解放
      ?37
```

注）　ここで　ロジカル名 pim: は us1:<pim>　である

1.2 SEILLAC ファイル構造概要

SEILLAC には、入力されたデータを一度だけ表示するTemporary モードと、必要に応じて再表示を行えるRetainedモードとがあり、後者の場合データは、ディスプレイ・ファイル又はビューイング・ファイルとして、セグメント・バッファと呼ばれるメモリーに格納される。

1.2.1 ディスプレイ・ファイルの概要

ディスプレイ・ファイルは下記の３要素からなる。

(1) ID
(2) SEGMENT
(3) SEGMENT CLASS (以下CLASS と略す)

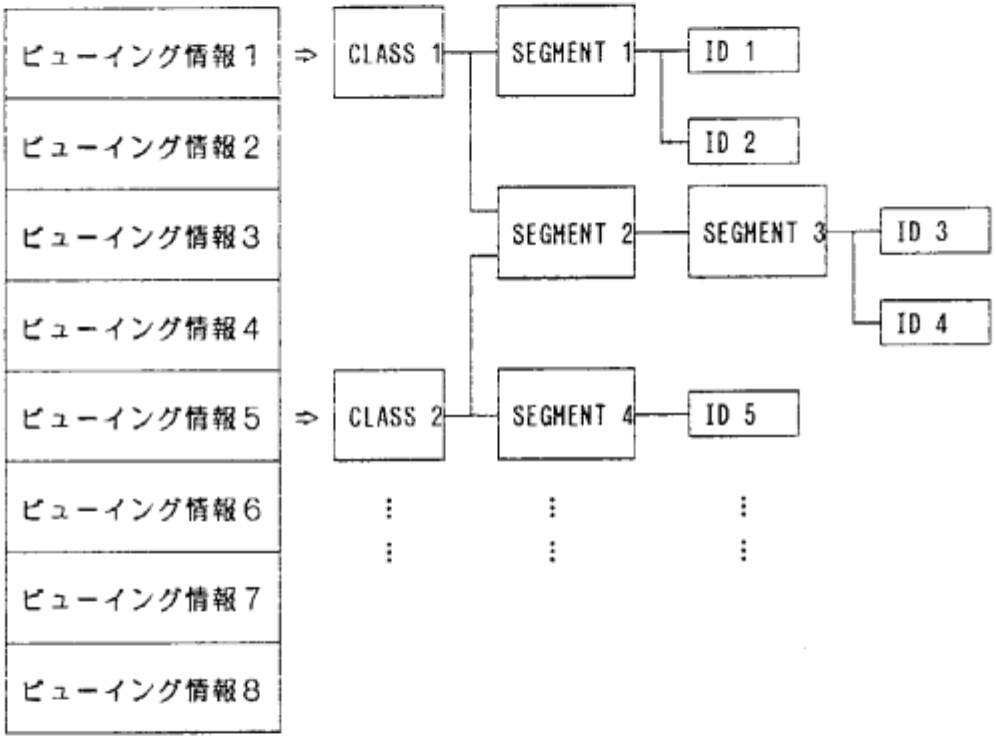
ここでIDとは、１つ以上のプリミティブ（図形、文字及びその属性）を含み、（ライトペン等による）検出や強調が可能であるような最小の図形単位である。このID又はプリミティブの集りで、論理的まとまりのある図形単位をSEGMENT と呼ぶ。SEGMENT は、サブルーチン化できる。これによってSEGMENT をuser定義のsymbolと似た使い方ができるが、前者が修正可能なのに対して後者が修正不可能な点が異なる。さらに SEGMENTをグループ化したものを、CLASS と呼ぶ。これは通常の場合、ディスプレイへの表示単位である。

SEGMENT とCLASS の相違点は、異なるSEGMENT 間で（SEGMENT のサブルーチン化によるSEGMENT の共有によらない）IDの共有はできないが、異なるCLASS 間では、SEGMENT 単位で共有ができる点と、SEGMENT に対しては、モデリング変換（対象図形そのものが変形する）が可能なのに比べ、CLASS に対しては、ビューイング変換（視点、射影面等の変換）が行われる点である。

CLASS,SEGMENT には、夫々、どのような変換を許すかという（CLASS,SEGMENT を定義した時点で静的に定める）静的属性と、表示，強調（プリンキング），（ライトペン等による）検出可能性を動的に制御する動的属性、及び像の変換属性（CLASS に対しては、クリッピング・ウィンドウの平行，回転移動、拡大／縮小及び視点の移動を、SEGUMENTに対しては、モデリング変換）を持つ。

1.2.2 ビューイング・ファイルの概要

ビューイング・ファイルは、図形のビューイング情報、即ち図形をどのように眺めスクリーンに映すかの情報から成っていて、任意の時点で画面全体あるいは複数に分割した画面（マルチ・ビューポート）で１つ又は複数のCLASS を表示可能である。ビューイング情報は、クリッピング・ウインドウ，ビューポート，視点情報からなる。



ビューイング　　ディスプレイ・ファイル
　ファイル

図 1.2　SEILLAC のファイル構造

1.3 SEILLAC の図形変換概要

SEGMENT は、ボデー座標系を持ち、ここで定義された図形の構成要素である、vector群からなっている。これをCRT 上に表示するまで即ちデバイス座標系に変換するまでには、以下の処理を行う。

(1) モデリング変換

ボデー座標系で定義された図形をワールド座標系（CLASS の持つ座標系）へ変換を行う。default は恒等変換（ボデー座標系とワールド座標系が一致するような変換）である。

(2) 視野変換

ワールド座標系の図形を視点座標系に変換し、ウィンドウ（図形の見える範囲で、これを越えた図形に対してクリッピングが行われる）を定める。

３次元図形の場合には、View Planeと呼ぶ平面（図形の投影面）を定義する。これは、View Reference Point (R)と、平面の法線vector (N)、及び、R から平面までの距離View Plane Distance (VD)とによって定まる。

２次元図形の場合にはウィンドウの設定に他ならない。

(3) 透視変換・平行投射

３次元図形の場合のみで、視点座標系に変換された図形をView Planeに、透視変換又は平行投射を行ないウィンドウで正規化する。

(4) ビューポート変換

前段までで得られた座標値を、CRT のデバイス座標系に変換する。即ち、ウィンドウに投影された図形をCRT スクリーンの一部であるビューポートに表示する。１スクリーン上には、複数のビューポートが定義可能である。

2. Data Type

2.1 数値型dataのType

本ライブラリーのArguments の現われる数値は、以下の５個の表現形式がある。

a)16bit のバイナリー表現

サイラックの標準表現形式

b)18bit のバイナリー表現

Dec 10 Prolog の標準表現で負数は２の補数表現である。

c)実数のlist表現 [X.Y]

各data type の整数部、小数部をlistによって表現したもので、

X+(Y/C) を意味する。

ここでC はdata type に依存した定数。

d)32bit のfunctor による実数の表現 real(X,Y)

X は整数部で符号付の16bit 又は18bit 表現(-32768 ～32767)

Y は16bit の自然数 (0～65535)

実数の演算用のライブラリー['number.lib']での実数の表現形式でもある。

e)atomによる実数の表現

'3' , '3.0' , '3.1415'の様な表現

小数部分の桁数は問わないが、d)の形式に内部で変換を行なうため、有効は少数点以下４桁である。

又、数値dataは、以下の６個のdata type を有し、許される表現形式が定まっている。

(1) 自然数type		(N type)	
SEGHENT no.	ID no.	(1～ 32767)	(write_code(16,X)で転送)
CLASS no.		(1～ 255)	(write_code(8,X) で転送)
char just		(1～ 16)	(write_code(4,X) で転送)
on-off用swich etc.			(write_seil(X) で転送)

表現としては、a), b)が許される。

(2) 整数type (Z type)

座標値，半径，文字間サイズ等 (-32768～32767)

表現としては a) (但し符号付16bit) , b)が許される。

サイラックへのdata転送は、write-code (16,X) を用いる。

(3) vector type　　(-1 ～1)

全表現が許される。

但し、

a)の場合、整数部2bit、小数部14bit の符号付16bit

b)の場合、a)の表現を整数と見た時の18bit 表現

c)　C=8192

サイラックへのdata転送は write_vectorを用いる。

(4) degree type　　(-360～360)

全表現が許される。

但し、

a)の場合、整数部10bit 、小数部6bitの符号付16bit

b)の場合、vestor type と同様

c)　C=64

サイラックへのdata転送は write_degreeを用いる。

(5) scale type　　(1/256～255)

全表現が許される。

但し、

a)の場合、整数部8bit、小数部8bitの16bit

b)、a)と同じ

c)　C= 256

サイラックへのdata転送は write_scale を用いる。

(6) (Matrix) element type　　(-32768<X<32768)

整数部16bit 、小数部16bit の実数

c), d), e)の表現が許される。

但し、

c)　C=65536

サイラックへのdata転送は、 write_matrixを用いる。

2.2　文字型dataのType

文字dataは (DEC-10 Prolog でいう) atomic型とstring型のtypeが許されている。

3　GP-PRO　コマンド

3.1　DISPLAY MODE　／　制御コマンド

本章においては，Display 装置のモード設定や、装置の初期化等、制御用のコマンドについての説明と、Prologで書かれたcommand 形式を述べる。

3.1.1 MODE コマンド

=> 取扱説明書 6.3.2

(1) 機能

コンソール又はグラフィック2D/3D のいずれかを指定する。

(2) Prolog command 形式

disp_mode(X)

(3) Arguments の説明

X= 2：グラフィック・モードの2D

X= 3：グラフィック・モードの3D

X= C：コンソール・モード

(4) 注意事項

ディスプレイ装置は、Power on時にコンソール・モードとなる。

3.1.2 DISPLAY ON コマンド

=> 取扱説明書 6.3.3

(1) 機能

可視画面の表示を行なう。

(2) Prolog command 形式

disp_on

(3) Arguments の説明

なし

(4) 注意事項

以下の場合に使用

a) Batchモードでの表示タイミング

b) Erace後の表示タイミング

このコマンドは、可視class に含まれる可視segment を表示する。

3.1.3 ERASE　コマンド

=> 取扱説明書 6.3.4

(1) 機能

画面の全面消去

(2) Prolog command　形式

erase

(3) Arguments の説明

なし

(4) 注意事項

現在表示中のPLANE に対しERASE を行う。

3.1.4 INITIALIZE コマンド

=> 取扱説明書 6.3.84

(1) 機能

display 、セグメント、ユーザー定義symbolの初期化。

(2) Prolog command 形式

initialize(A,B,C)

(3) Arguments の説明

A,B,C ：N Type (0,1)

A=1 ：画面消去をしない。

A=0 ：画面消去をする。

B=1 ：セグメントの初期化をしない。

B=0 ：セグメントの初期化をする。

C=1 ：ユーザー定義シンボルの初期化をしない。

C=0 ：ユーザー定義シンボルの初期化をする。

3.1.5 MACHINE RESET コマンド

=> 取扱説明書 6.3.85

(1) 機能

ソフトウェア・リセット （display を電源投入時に戻す）

(2) Prolog command 形式

mac_reset

(3) Arguments の説明

なし

(4) 注意事項

セグメント等の一切のテーブルは削除される。

（但し、DISP_modeの設定は解除されない）

3.1.6 LINE SMOOTHING コマンド

=> 取扱説明書 6.3.86

(1) 機能

ラインスムージング処理の指定

(2) Prolog command 形式

line_smooth(I)

(3) Arguments の説明

I : N type (0,1)

I=0 : ラインスムージング処理を行わない。

I=1 : ラインスムージング処理を行う。

(4) 注意事項

1)本コマンドは、セグメント内に含めることができる。

2)LINE SMOOTHING状態で、近接した 1ライン毎に線を描くような場合、輝度が重り合って、まだらとなる事がある。

3.1.7 BATCH コマンド

=> 取扱説明書 6.3.90

(1) 機能

セグメントの一括登録又はクラス／セグメント／IDの一括変更を行なう。

(2) Prolog command 形式

batch (I)

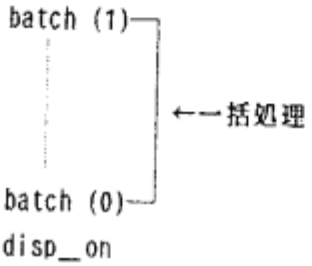
(3) Arguments の説明

I : N type (0,1)

I=0：一括処理の終了

I=1：一括処理の開始

(4) 注意事項



disp_on

1)セグメントに対する作成、変更を行ってもbatch on状態ではスクリーン上の図形に影響は与えない。

2)新しいセグメントの単純な追加登録は本コマンドにより高速化できる。

(seiliac ではセグメントの作成が生ずると、それまでの登録済のセグメントを全て再書込するため)

3.1.8 TEMPORARY SEGMENT コマンド　　=> 取扱説明書 6.3.39

(1) 機能

Retained / Temporary Segument モードの切替え

(2) Prolog command 形式

temp_seg(I)

(3) Arguments の説明

I　：N type (0,1)

I=0 ：Temporary モードへ切替る。

I=1 ：Retainedモードへ切替る。

(4) 注意事項

default はRetainedモードで、クラス／セグメント関連のコマンドはRetained モードの時のみ有効。

3.1.9 SELECTIVE ERASE コマンド

=> 取扱説明書 6.3.5

(1) 機能

Temporary segment の開始及び消去

(2) Prolog command 形式

sel_erase (I)

(3) Arguments の説明

I : N type (0,1)

I=0 : 消去

I=1 : 表示開始

3.1.10 THROUGH コマンド　　=> 取扱説明書 6.3

(1) 機能

スクリーン座標系のデータ群が送られてくることを指定する。

(2) Prolog command　形式

through (I)

(3) Arguments の説明

I　　: N type (0,1)

I=0 : through モードoff

I=1 : through モードon

(4) 注意事項

1)セグメントデータ間に挿入可

2)本モードを指定した時、座標値は、左下を原点としたスクリーン座標系となる。
（座標変換は行わない）

3.2　基本図形コマンド

本章においては、現在位置（CP：カレント・ポジション）の移動や、線分、円等の基本的な2D/3D の図形を描くコマンドについての説明と、Prologで書かれたcommandの形式について述べる。

3.2.1 MOVE ABS コマンド

=> 取扱説明書 6.3.6

(1) 機能

CPを絶対値で指定された位置へ移動する。

(2) Prolog command 形式

2D : move_abs _2 (X,Y)

3D : move_abs _3 (X,Y,Z)

(3) Arguments の説明

X,Y,Z : Z type (-32768 ～32767)

1) 2D :

(X,Y) は位置　　new CP = (X,Y)

2) 3D :

(X,Y,Z) は位置　　new CP = (X,Y,Z)

(4) 注意事項

なし

3.2.2 MOVE REL コマンド

=> 取扱説明書 6.3.7

(1) 機能

CPに相対値を加えた位置へ移動する。

(2) Prolog command 形式

2D : move_rel _2 (Dx,Dy)

3D : move_rel _3 (Dx,Dy,Dz)

(3) Arguments の説明

X,Y,Z : Z type (-32768 ～32767)

1) 2D :

(Dx,Dy) は移動量　　new CP = CP+(Dx,Dy)

2) 3D :

(Dx,Dy,Dz)は移動量　　new CP = CP+(Dx,Dy,Dz)

(4) 注意事項

なし

3.2.3 LINE ABS コマンド

=> 取扱説明書 6.3.8

(1) 機能

CPから絶対値で指定された位置まで線分を引く。

(2) Prolog command 形式

2D : line_abs _2 (X,Y)

3D : line_abs _3 (X,Y,Z)

(3) Arguments の説明

X,Y,Z : Z type (-32768 ～32767)

1) 2D :

CPより(X,Y) まで線分を引く。

new CP = (X,Y)

2) 3D :

CPより(X,Y,Z) まで線分を引く。

new CP = (X,Y,Z)

(4) 注意事項

なし

3.2.4 LINE REL コマンド

=> 取扱説明書 6.3.9

(1) 機能

CPよりCPに相対値を加えた位置まで線分を引く。

(2) Prolog command 形式

2D : line_rel _2 (Dx,Dy)

3D : line_rel _3 (Dx,Dy,Dz)

(3) Arguments の説明

Dx,Dy,Dz : Z type (-32768 ～32767)

1) 2D :

CPよりnew CP まで線分を引く。

new CP = CP+(Dx,Dy)

2) 3D :

CPよりnew CP まで線分を引く。

new CP = CP+(Dx,Dy,Dz)

(4) 注意事項

なし

3.2.5 GRID コマンド

=> 取扱説明書 6.3.10

(1) 機能

格子点を描く（２次元のみ）

(2) Prolog command 形式

grid (X,Y,Lx,Xp,Ly,Yp)

(3) Arguments の説明

X,Y ：Z type (-32768～32767)

(X,Y) は基点座標

Lx,Ly ：N type (0～32767)

X,Y 方向の格子を描く範囲

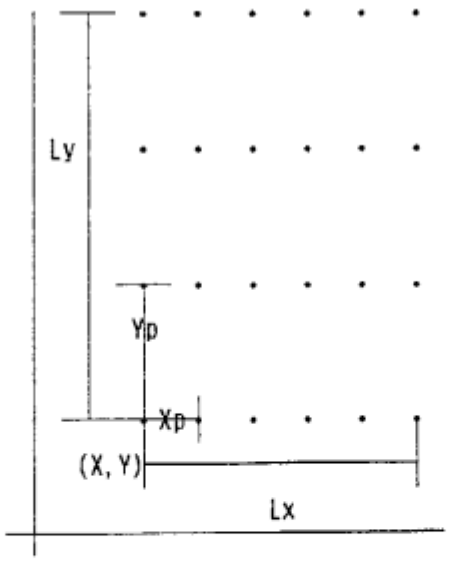
Xp,Yp ：N type (0～32767)

X,Y 方向の格子点間距離

(4) 注意事項

1) ワールド座標系で定義する。

2) gridマークは "・" である。



3.2.6 POLYLINE ABS コマンド

=> 取扱説明書 6.3.11

(1) 機能

CPを始点とし、絶対値で指定された点列を順次線分で結ぶ。

(2) Prolog comand 形式

2D : polyl _abs _2 (LIST2)

3D : polyl _abs _3 (LIST3)

(3) Arguments の説明

LIST2, LIST3 ：各要素は Z type (-32768 ～32767)

1) 2D :

LIST2=[X1, Y1, X2, Y2, ……, Xn, Yn] , length (2n, LIST2)

であり、

P1=(X1, Y2), ……, Pn=(Xn, Yn)

とおくと、折れ線CP-P1-P2- ……-Pn を描く

new CP=Pn

2) 3D :

LIST3=[X1, Y1, Z1, X2, Y2, Z2, ……, Xn, Yn, Zn], length (3n, LIST3)

であり、

P1=(X1, Y1, Z1), ……, Pn=(Xn, Yn, Zn)

とおくと、折れ線CP-P1-P2- ……-Pn を描く

new CP=Pn

(4) 注意事項

なし

3.2.7 POLYLINE REL コマンド

=> 取扱説明書 6.3.12

(1) 機能

CPを始点とし、相対値で指定された点列を順次直線として描く。

(2) Prolog command 形式

2D : poly1 _rel _2 (LIST2)

3D : poly1 _rel _3 (LIST3)

(3) Arguments の説明

LIST2, LIST3 ：各要素は Z type (-32768 ～32767)

2) 2D :

LIST2=[Dx1, Dy1, Dx2, Dy2, ……, Dxn, Dyn] , length (2n, LIST2)

であり、

P1=CP+(Dx1, Dy1)

P2=P1+(Dx2, Dy2)

⋮

⋮

Pn=Pn-1+(Dxn+Dyn)

とすると、折れ線CP-P1-P2- ……-Pn を描く

new CP=Pn

2) 3D :

LIST3=[Dx1, Dy1, Dz1, ……, Dxn, Dyn, Dzn] , length (3n, LIST3)

であり、

P1=CP+(Dx1, Dy1, Dz1)

P2=CP+(Dx2, Dy2, Dz2)

⋮

⋮

Pn=Pn-1+(Dxn+Dyn+Dzn)

とすると、折れ線CP-P1-P2- ……-Pn を描く

new CP=Pn

(4) 注意事項

なし

3.2.8 POLYGON ABS コマンド

=> 取扱説明書 6.3.13

(1) 機能

絶対値で示される座標値を用いて多角形を描く。

(2) Prolog command 形式

2D : polyg _abs _2 (LIST2)

3D : polyg _abs _3 (LIST3)

(3) Arguments の説明

LIST2, LIST3 ：各要素は Z type (-32768 ～32767)

1) 2D :

LIST2=[X1, Y1, X2, Y2, ……, Xn, Yn] , length (2n, LIST2)

であり、

P1=(X1, Y1), P2=(X2, Y2), ……Pn=(Xn, Yn)

とおけば多角形P1-P2-……-Pn を描く

new CP=P1

2) 3D :

LIST3=[X1, Y1, Z1, X2, Y2, Z2, ……, Xn, Yn, Xn], length (3n, LIST3)

であり、

P1=(X1, Y1, Z1) , P2=(X2, Y2, Z2) , ……, Pn=(Xn, Yn, Zn)

とおけば多角形P1-P2-……-Pn を描く

new CP=P1

(4) 注意事項

塗りつぶしを伴うPolygon は凸でなければならない。

（凸でないPOLYGON の塗りつぶしは、保証しない）

3.2.9 POLYGON REL コマンド

=> 取扱説明書 6.3.14

(1) 機能

相対値で示される座標値を用いて多角形を描く。

(2) Prolog command 形式

2D : polyg _rel _2 (LIST2)

3D : polyg _rel _3 (LIST3)

(3) Arguments の説明

LIST2,LIST3 ：各要素は Z type (-32768 ～32767)

2) 2D :

LIST2=[Dx1,Dy1,Dx2,Dy2, ……,Dxn,Dyn] , length (2n,LIST2)

であり、

P1=CP+(Dx1,Dy1)

P2=P1+(Dx2,Dy2)

⋮

⋮

Pn=Pn-1+(Dxn+Dyn)

とおけば多角形P1-P2-……-Pn を描く

new CP=P1

3D =>

LIST3=[Dx1,Dy1,Dz1,Dx2, ……,Dxn,Dyn,Dzn] , length (3n,LIST3)

であり、

P1=CP+(Dx1,Dy1,Dz1)

P2=CP+(Dx2,Dy2,Dz2)

⋮

⋮

Pn=Pn-1+(Dxn+Dyn+Dzn)

とおけば多角形P1-P2-……-Pn を描く

new CP=P1

(4) 注意事項

POLYGON ABS と同じ

3.2.10 RECTANGLE　コマンド　　=> 取扱説明書 6.3.15

(1) 機能

絶対値で示される２対の座標値を用いて長方形を描く。

(2) Prolog command　形式

rectang(X1,Y1,X2,Y2)

(3) Arguments の説明

X1,Y1,X2,Y2 ：Z type (-32768 ～32767)

(X1,Y1),(X2,Y2) は長方形の対角線方向の頂点の座標値で
長方形(X1,Y1)-(X1,Y2)-(X2,Y2)-(X2,Y1) を描く。

(4) 注意事項

new CP=(X1,Y1)

3.2.11 CUBIC　コマンド　　=> 取扱説明書 6.3.16

(1) 機能

絶対値で示される２対の座標値を用いて直方体を描く。

(2) Prolog command　形式

cubic(X1,Y1,Z1,X2,Y2,Z2)

(3) Arguments の説明

X1,Y1,Z1,X2,Y2,Z2 ：Z type (-32768 ～32767)

(X1,Y1,Z1),(X2,Y2,Z2) は直方体の対角線方向の頂点の座標値で各辺が座標軸に平行な直方体を描く

(4) 注意事項

new CP=(X1,Y1,Z1)

3.2.12 CIRCLE コマンド

=> 取扱説明書 6.3.17

(1) 機能

CPを中心に円を描く。

(2) Prolog command 形式

2D : circle_2(R)

3D : circle_3(R,Ux,Uy,Uz)

(3) Arguments の説明

R : N type (1～32767)

半径を指定する

Ux,Uy,Uz : degree type (-360～360)

円の面の回転角を指定する。(3Dのみ)

(4) 注意事項

1) new CP = CP

2) 3Dの場合

原点を中心としたXY平面上の円に対し、
X軸を軸とする角度Uxの回転、
Y軸を軸とする角度Uyの回転、
Z軸を軸とする角度Uzの回転を順次加え、
さらに、中心をCPへ移す平行移動を加えた図形を描く。

3) 3Dにおいて、特に円の法線 vector が

(sinA cosB , sinA sinB , cosA)

の場合，この円は

circle_3(R,0,A,B)

によって描ける

3.2.13 ELLIPSE コマンド

=> 取扱説明書 6.3.18

(1) 機能

CPを中心に楕円を描く。

(2) Prolog command　形式

2D：ellipse _2(A,B)

3D：ellipse _3(A,B,Ux,Uy,Uz)

(3) Arguments の説明

A,B ：N type (1～32767)

楕円の X軸 Y軸方向の長さを指定する

Ux,Uy,Uz：degree type (-360～360)

楕円の面の回転角を指定する。(3Dのみ)

(4) 注意事項

1)　new CP = CP

2) 2D　の場合

原点が中心で、主軸を X軸 Y軸とし、
X軸方向の長さが A、 Y軸方向の長さが Bである楕円に、
中心がCPとなるような平行移動を加えた図形を描く

3) 3D　の場合

XY平面の楕円にcircle_3 と同回転を加える

3.2.14 RAC コマンド

=> 取扱説明書 6.3.19

(1) 機能

CPを中心に円弧を描く。

(2) Prolog command 形式

2D : arc __2(R,O1,O2)

3D : arc __3(R,O1,O2,Ux,Uy,Uz)

(3) Arguments の説明

R : N type (1～32767)

円弧の半径を指定

O1,O2 : degree type

O1 : 始点角

O2 : 終点角

Ux,Uy,Uz : degree type (-360～360)

円弧の面の回転角を指定する。(3Dのみ)

(4) 注意事項

1) new CP = CP

2) 円弧の方向は反時計まわり

3) 3Dの場合の回転角の指定 Ux,Uy,Uz の意味は、 3D の円と同じ

3.2.15 FAN コマンド

=> 取扱説明書 6.3.19

(1) 機能

CPを中心に扇形を描く。

(2) Prolog command 形式

2D : fan __2(R,O1,O2)

3D : fan __3(R,O1,O2,Ux,Uy,Uz)

(3) Arguments の説明

R : N type (1～32767)

扇形の半径を指定

O1,O2 : degree type

O1 : 始点角

O2 : 終点角

Ux,Uy,Uz : degree type (-360～360)

扇形の面の回転角を指定する。(3Dのみ)

(4) 注意事項

1) new CP = CP

2) 扇形の方向は反時計まわり

3) 3Dの場合の回転角の指定 Ux,Uy,Uz の意味は、 3D の円と同じ

3.3 CHARACTER/CURSOR コマンド

本章では、CHARACTER/CURSOR関連のコマンドについての説明と、Prologで書かれたcommand の形式を述べる。

キャラクタには、グラフィック・キャラクタとコンソール・キャラクタの２種類があり、char_prac_attrにより切替る。

グラフィック・キャラクタは、文字のユニット・サイズは32ドット×32ドットであるが、ワールド座標での座標変換を加えることができる。CPの位置に出力する。

コンソール・キャラクタは10ドット×16ドットであり、 126文字×32行のコンソール・イメージで出力される。出力位置は CONSOLE CHAR POSITION によって指定する。

（開発中）

3.3.1 CHARACTER TYPE コマンド

=> 取扱説明書 6.3.55

(1) 機能

ROMAN / ITALIC文字の指定

(2) Prolog command 形式

char_typ (A)

(3) Arguments の説明

A : N type (0, 2)

A=0 : ROMAN

A=2 : ITALIC

(4) 注意事項

default はROMAN

3.3.2 CHAR-PRECISION ATTRIBUTE コマンド

=> 取扱説明書 6.3.56

(1) 機能

グラフィック／コンソール・キャラクタの設定

(2) Prolog command 形式

char_prec_attr (I)

(3) Arguments の説明

I : N type (0～2)

I=0 : STRING_PRECISION

グラフィック・キャラクタを指定

文字の基準点と大きさのみ変更可能

text_ent _charによってCPより出力を行う

I=1 : char_PRECISION を指定を指定

コンソール・キャラクタ

console _attrが意味を持つ。

consol_char_positionによって位置を指定し出力を行う。

I=2 : storoke _PRECISION

グラフィック・キャラクタを指定

char_attr, char_co_attr, char_typ の全属性を指定できる。

text_ent _charによってCPより出力を行う

(4) 注意事項

3.3.3 CHARACTER ATTRIBUTE　コマンド　　=> 取扱説明書 6.3.57

(1) 機能

文字の大きさや文字間隔等を指定する。

(2) Prolog command　形式

char_attr (P,J,S,X,Y)

(3) Arguments の説明

P　：N type (0～3)

P=0：上方向

P=1：下方向

P=2：左方向

P=3：右方向

J　：N type (0～15)

文字の基準点 (char Just)の設定

S　：Z type (-32768 ～32767)

文字間隔を設定する。

X,Y ：scale type　(1/256～255)

文字の縦、横の大きさを指定する。

(4) 注意事項

なし

3.3.4 CHARACTER COORDINATE ATTRIBUTE 2 コマンド => 取扱説明書 6.3.58

(1) 機能

2Dにおいて文字の方向を設定する。

(2) Prolog command 形式

char_co_attr_2(Vx,Vy)

(3) Arguments の説明

Vx,Vy : vector type

文字の方向(char _up) を示す方向vector (Vx,Vy)

(4) 注意事項

3.3.5 CHARACTER COORDINATE ATTRIBUTE 3 コマンド => 取扱説明書 6.3.59

(1) 機能

3Dにおける文字表示平面と文字の方向を設定

(2) Prolog command 形式

char_co_attr_3(Ux,Vx,Wx,Uy,Vy,Wy,Uz,Vz,Wz)

(3) Arguments の説明

Ux,Uy,…Wz : vector type (-1～1)

文字に対して変換

$$\begin{vmatrix} Ux & Vx & Wx & 0 \\ Uy & Vy & Wy & 0 \\ Uz & Vz & Wz & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

を加える

(4) 注意事項

文字表示面の単位法線vectorを

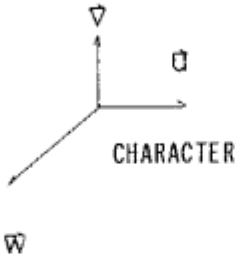
$\vec{W} = (Wx, Wy, Wz)$

文字の文字の方向を表す単位vectorを

$\vec{U} = (Ux, Uy, Uz)$

$\vec{V} = (Vx, Vy, Vz)$

とする。



3.3.6 TEXT ENTER (CHARACTER) コマンド

=> 取扱説明書 6.3.24

(1) 機能

文字列を書く。

(2) Prolog command 形式

text_ent _char (A)

(3) Arguments の説明

A : char_ type

ASCII+カナコードのLIST (string) 又はatomic

(4) 注意事項

- graphic character の表示位置は、MOVEコマンドを使用。
- console character の表示位置は、console _char_pos コマンドを使用

3.3.7 CONSOLE CHARACTER ATTRIBUTE コマンド　　=> 取扱説明書 6.3.74.1

(1) 機能

コンソール文字の強張

(2) Prolog command 形式

console _char_attr (I)

(3) Arguments の説明

b　：N type (0,1)

b=0 ：ブリンクしない

b=1 ：ブリンクする

(4) 注意事項

本属性は、グラフィック・モードにおいて console character imageを表示する時（char_prec_attr (1)の時）有効。

3.3.8 CONSOLE CHARACTER POSITION コマンド

=> 取扱説明書 6.3.24.1

(1) 機能

console character の表示位置

(2) Prolog command 形式

console _char_pos (X,Y)

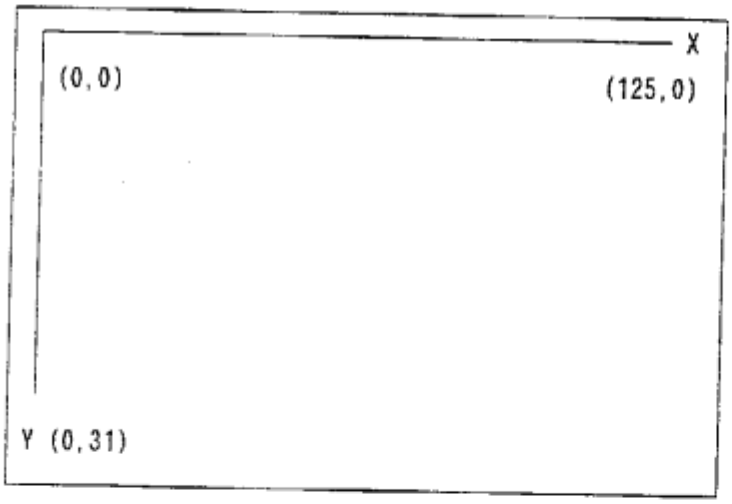
(3) Arguments の説明

X : N type (0～125)

Y : N type (0～31)

(4) 注意事項

原点は左上



3.3.9 CONSOLE CLEAR コマンド

=> 取扱説明書 6.3.5.2

(1) 機能
コンソール文字の消去

(2) Prolog command 形式
console _clear

(3) Arguments の説明
なし

(4) 注意事項
なし

3.3.10 CURSOR ON/OF コマンド

=> 取扱説明書 6.3.5.1

(1) 機能

cursorの表示を設定

(2) Prolog command 形式

cursor_on (I)

(3) Arguments の説明

I : N type

I=0 : cursorを表示しない

I=1 : cursorを表示する

(4) 注意事項

3.3.11 SET CURSOR TYPE コマンド　　=> 取扱説明書

(1) 機能

グラフィックモードでのCURSORのタイプを指定する

(2) Prolog command 形式

set _cursor_typ(I)

(3) Arguments の説明

I　：N type

シンボル番号を指定する

(4) 注意事項

1) CURSOR POSITION を発行する直前で指定する

2) ローカルインタラクション時のCURSORのタイプにも有効である

3.3.12 CURSOR POSITION コマンド　　=> 取扱説明書

(1) 機能

グラフィックモードでのCURSORの（スクリーン座標における）位置を指定する

(2) Prolog command　形式

cursor_pos(X,Y)

(3) Arguments の説明

X　：N type (0～5468)

Y　：N type (0～4095)

(4) 注意事項

3.4 MARKER コマンド

本章においては、systemに定義されているmarker symbol 関連のコマンドについての説明と、Prologで書かれたcommand の形式について述べる。

marker symbol は、モデリング変換や視野変換の影響を受けるが、大きさは固定である。

3.4.1 MARKER TYPE　コマンド　　=> 取扱説明書 6.3.74

(1) 機能

表示するmarker symbol を選択する。

(2) Prolog command　形式

mark_typ (I)

(3) Arguments の説明

I ：N type (1～8)

marker type 番号を指定する。

(4) 注意事項

3.4.2 MARKER ABS コマンド

=> 取扱説明書 6.3.20

(1) 機能

絶対値で示される位置にmarker symbol を描く。

(2) Prolog command 形式

2D : mark_abs _2 (X,Y)

3D : mark_abs _3 (X,Y,Z)

(3) Arguments の説明

X,Y,Z : Z type (-32768 ～32767)

1) 2D : new CPにmarkerを描く。

new CP=(X,Y)

2) 3D : new CPにmarkerを描く。

new CP=(X,Y,Z)

(4) 注意事項

なし

3.4.3 MARKER REL コマンド

=> 取扱説明書 6.3.21

(1) 機能

相対値で示される位置にmarker symbol を描く。

(2) Prolog command 形式

2D : mark_rel _2 (Dx, Dy)

3D : mark_rel _3 (Dx, Dy, Dz)

(3) Arguments の説明

Dx, Dy, Dz : Z type (-32768 ～32767)

1) 2D : new CPにmarker symbol を描く

new CP = CP+(Dx, Dy)

2) 3D : new CPにmarker symbol を描く

new CP = CP+(Dx, Dy, Dz)

(4) 注意事項

なし

3.4.4 POLYMARKER ABS コマンド

=> 取扱説明書 6.3.22

(1) 機能

絶対値の座標値で指定される点列にmarke symbolを描く。

(2) Prolog command 形式

2D : polym _abs _2 (LIST2)

3D : polym _abs _ 3(LIST3)

(3) Arguments の説明

LIST2 , LIST3 ：各要素はZ type (-32768 ～32767)

1) 2D : LIST2=[X1,Y1,X2,Y2, ……,Xn,Yn]

length (2n,LIST2)

であり

P1=(X1,Y1), P2=(X2,Y2), ……, Pn(Xn,Yn)

とおくとき、P1,P2,……,Pn にmarker symbol を描く。

new CP = CP

2) 3D : LIST3=[X1,Y1,Z1,X2,Y2,Z2, ……,Xn,Yn,Zn]

length (3n,LIST3)

であり

P1=(X1,Y1,Z1) , P2=(X2,Y,Z2), ……, Pn(Xn,Yn,Zn)

とおくとき、P1,P2,……,Pn にmarker symbol を描く。

new CP = CP

(4) 注意事項

なし

3.4.5 POLYMARKER REL コマンド

=> 取扱説明書 6.3.23

(1) 機能

相対値の座標で指定された点列にmarker symbol を描く。

(2) Prolog command 形式

2D : polym _rel _2 (LIST2)

3D : polym _rel _3 (LIST3)

(3) Arguments の説明

LIST2 , LIST3 : 各要素はZ type (-32768 ～32767)

1) 2D :

LIST2=[Dx1,Dy1,Dx2,Dy2, ……,Dxn,Dyn] , length (2n,LIST2)

であり

P1=CP+(Dx1,Dy1)

P2=P1+(Dx2,Dy2)

⋮

⋮

Pn=Pn-1+(Dxn,Dyn)

とおくとき、P1,P2,……,Pn にmarker symbol を描く。

new CP = CP

2) 3D :

LIST3=[Dx1,Dy1,Dz1,Dx2, ……,Dxn,Dyn,Dzn] , length (3n,LIST3)

であり

P1=CP+(Dx1,Dy1,Dz1)

P2=P1+(Dx2,Dy2,Dz2)

⋮

⋮

Pn=Pn-1+(Dxn,Dyn,Dzn)

とおくとき、P1,P2,……,Pn にmarker symbol を描く。

new CP = CP

(4) 注意事項

なし

3.5 SYMBOL コマンド

本章においては、ユーザーが定義することのできるSYMBOL関連のコマンドについての説明と、Prologで書かれたcommand の形式について述べる。

3.5.1 USER SYMBOL DEFINITION コマンド

=> 取扱説明書 6.3.88

(1) 機能

ユーザー定義のSYMBOLの登録を開始する。

(2) Prolog command 形式

user_sym _def (S)

(3) Arguments の説明

S : N type (1～32767)

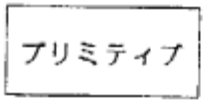
ユザー定義のSYMBOLコードを指定する。

(4) 注意事項

1)TEMPORALY/RETAINEDモードのどちらでも可。

2)SEGMENT がオープン中の場合はエラーとなる。

user_sym _def(S)



user_sym _term

3.5.2 USER SYMBOL TERMINATE コマンド

=> 取扱説明書 6.3.89

(1) 機能

ユーザ定義のSYMBOLの登録を終了する。

(2) Prolog command 形式

user_sym _term

(3) Arguments の説明

なし

(4) 注意事項

なし

3.5.3 TEXT ENTER (SYMBOL) コマンド

=> 取扱説明書 6.3.24

(1) 機能

CPにSYMBOLを書く。

(2) Prolog command 形式

text_ent _sym (S)

(3) Arguments の説明

S : N type (1～32767)

SYMBOLコードを指定する。

(4) 注意事項

なし

3.6 COLOR/FILLING コマンド

本章においては、COLOR 及び中塗り用のcommand についての説明と、Prologで書かれ command の形式を述べる

3.6.1 COLOR TYPE 1 コマンド

=> 取扱説明書 6.3.46

(1) 機能

基本色の設定を行う

(2) Prolog command 形式

col _typ _1(R,G,B)

(3) Arguments の説明

R : N type (0～7)

赤の濃度を指定する

G : N type (0～7)

緑の濃度を指定する

B : N type (0～4)

青の濃度を指定する

(4) 注意事項

col _typ _1(7,7,3) は白

col _typ _1(0,0,0) は無色(BACKGROUND COLOR)

default は緑

3.6.2 COLOR TYPE 2 コマンド

=> 取扱説明書 6.3.47

(1) 機能

混合色の設定を行う

(2) Prolog command 形式

col _typ _2(R1,G1,B1,R2,G2,B2)

(3) Arguments の説明

R1,R2 : N type (0～7)

赤の濃度を指定する

G1,G2 : N type (0～7)

緑の濃度を指定する

B1,B2 : N type (0～4)

青の濃度を指定する

(4) 注意事項

(R1,G1,B1) と (R2,G2,B2) の混合色

col _typ _2(R,G,B,R,G,B)は col _typ _1(R,G,B)に同じ

default は緑

3.6.3 BACKGRAUND COLOR コマンド

=> 取扱説明書 6.3.54

(1) 機能

背景色の設定を行う

(2) Prolog command 形式

back_col(X)

(3) Arguments の説明

X ：N type (0～7)

背景色を指定する

X=0：無色

X=1：青

X=2：緑

X=3：空色

X=4：赤

X=5：紫

X=6：黄色

X=7：白

(4) 注意事項

seg _def とseg _termの間でのみ使用可能

3.6.4 FILLING MODE コマンド

=> 取扱説明書 6.3.48.1

(1) 機能

中塗りの開始と終了を指定する

(2) Prolog command 形式

fill_mode(I)

(3) Arguments の説明

I : N type (0,1)

I=1 : 中塗りの開始

I=0 : 中塗りの終了

(4) 注意事項

3.6.5 FILLING ATTRIBUTE コマンド

=> 取扱説明書 6.3.49

(1) 機能

中塗りの型と外周表示の型を指定する

(2) Prolog command 形式

fill_attr(E,P)

(3) Arguments の説明

E : N type (0,1)

E=0：外周表示をしない

E=1：外周表示をする

P : N type (0,1)

P=0：無地

P=1：柄地

(4) 注意事項

3.6.6 FILLING COLOR TYPE 1 コマンド　　=> 取扱説明書 6.3.50

(1) 機能

中塗りの基本色の設定を行う

(2) Prolog command 形式

fill_col _typ _1(R,G,B)

(3) Arguments の説明

R : N type (0～7)

赤の濃度を指定する

G : N type (0～7)

緑の濃度を指定する

B : N type (0～4)

青の濃度を指定する

(4) 注意事項

fill_col _typ _1(7,7,3) は白

fill_col _typ _1(0,0,0) は無色(BACKGROUND COLOR)

default は緑

3.6.7 FILLING COLOR TYPE 2 コマンド

=> 取扱説明書 6.3.51

(1) 機能

中塗りの混合色の設定を行う

(2) Prolog command 形式

fill_col _typ _2(R1,G1,B1,R2,G2,B2)

(3) Arguments の説明

R1,R2 : N type (0～7)

赤の濃度を指定する

G1,G2 : N type (0～7)

緑の濃度を指定する

B1,B2 : N type (0～4)

青の濃度を指定する

(4) 注意事項

(R1,G1,B1) と (R2,G2,B2) の混合色

fill_col _typ _2(R,G,B,R,G,B)は fill_col _typ _1(R,G,B)に同じ

default は緑

3.6.8 FILLING PATTERN　コマンド

=> 取扱説明書 6.3.52

(1) 機能

中塗りの柄の設定を行う

(2) Prolog command　形式

fill_pat(I)

(3) Arguments の説明

I　：N type (1～63)

中塗りの柄の番号を指定する（付録1 参照）

(4) 注意事項

fill_attr(X,1) の時、意味をもつ

3.6.9 LINE TYPE　コマンド

=> 取扱説明書 6.3.48

(1) 機能

線分の種類の設定を行う

(2) Prolog command　形式

line_typ(I)

(3) Arguments の説明

I　：N type (1～8)

線分の種類を指定する (付録1 参照)

(4) 注意事項

なし

3.7 CLASSコマンド

本章においては、（VIEWING を除く） CLASSに関連したコマンドについての説明と、Prologで書かれたcommand 形式について述べる。

SEGMENT をグループ化したものをCLASS と呼び、これがディスプレイへの表示単位となる。CLASS に対してはビューイング変換が行われるが、これ関するコマンドは、後章で述べる。

CLASS に対する属性として、どのような像の変換属性を、CLASS に対して許すか、CLASS を定義した時点で静的に定める静的属性と、CLASS の表示，強調（ブリンキング）及び（ライトペン等による）検出可能性を動的に制御する動的属性、及び、クリッピング・ウインドウの平行移動，回転移動，拡大／縮小，視点の移動といった、像の変換属性を持つ。

3.7.1 SEGMENT CLASS DIFINITION コマンド

=> 取扱説明書 6.3.25

(1) 機能

CLASS の定義を行う。

(2) Prolog command 形式

seg_class _def (C,LIST)

(3) Arguments の説明

C : N type (1～255)

定義するCLASS 名

LIST : 各要素はZ type (1～32767)

本CLASS を構成するSEGMENT の名前のLIST

(4) 注意事項

SEGMENT は登録されていなくてもよい。

LIST中のSEGMENT はuniqueでなければいけない。

3.7.2 DELETE CLASS コマンド

=> 取扱説明書 6.3.26

(1) 機能

指定したCLASS を削除する。

(2) Prolog command 形式

del_class (C)

(3) Arguments の説明

C : N type (1～255)

削除するCLASS 名

(4) 注意事項

SEGMENT のCLASS 化に関する情報が削除されるだけであり、SEGMENT 自体には影響を与えない。

3.7.3 DELETE CLASS ALL コマンド

=> 取扱説明書 6.3.27

(1) 機能

全CLASS の削除をする。

(2) Prolog command 形式

del _class _all

(3) Arguments の説明

なし

(4) 注意事項

DELETE CLASSコマンドと同様。

3.7.4 RENAME CLASS コマンド

=> 取扱説明書 6.3.28

(1) 機能

CLASS 名を変更する。

(2) Prolog command 形式

ren _class (Old, New)

(3) Arguments の説明

Old , New : N Type (1～255)

Old : 旧CLASS 名

New : 新CLASS 名

(4) 注意事項

CLASS に対する全ての属性は引き継がれる。

3.7.5 APPEND CLASS MEMBER コマンド

=> 取扱説明書 6.3.40

(1) 機能

CLASS の構成メンバーに指定したセグメントを追加する。

(2) Prolog command 形式

app_class _mem (C, S)

(3) Arguments の説明

C : N type (1～255)

SEGMENT を追加するCLASS 名

S : N type (1～32767)

追加するSEGMENT 名

(4) 注意事項

CLASS に既にSEGMENT が含まれている場合、エラーとなる。

追加するSEGMENT がオープン中の場合*)はエラーとなる。

*) SEGMENT DEFINITIONまたはEDIT SEGMENTコマンド実行後、SEGMENT TERMINATOR コマンド実行までの間

3.7.6 DELETE CLASS MEMBER コマンド　　=> 取扱説明書 6.3.41

(1) 機能

CLASS の構成メンバーから指定したSEGMENT を削除する。

(2) Prolog command 形式

del_class _mem (C, S)

(3) Arguments の説明

C : N type (1～255)

SEGMENT を削除するクラス名

S : N type (1～32767)

削除するSEGMENT

(4) 注意事項

削除するSEGMENT がオープン中はエラー

3.7.7 CLASS STATIC ATTRIBUTE コマンド

=> 取扱説明書 6.3.60

(1) 機能

CLASS に許される像変換の設定

(2) Prolog command 形式

class _stat_attr (I, C)

(3) Arguments の説明

I : N type (0～3)

I=0 : NO CLASS TRANSFORMATION

I=1 : 2D WINDOW TRANSFORMATION

I=2 : 2D WINDOW TRANSFORMATION ZOOMING

I=3 : 3D LOOKPOINT TRANSFORMATION ZOOMING

C : N type (1～255)

クラス名

(4) 注意事項

動的に変更はできない。

3.7.8 CLASS DYNAMIC ATTRIBUTE TYPE 1 コマンド => 取扱説明書 6.3.61

(1) 機能

CLASS の可視性の設定

(2) Prolog command 形式

class _dyn _attr_typ _1

(3) Arguments の説明

I : N type (0,1)

可視性の設定

I=1：可視CLASS である。 (VISIBILITY ON)

I=0：可視CLASS でない。 (VISIBILITY OFF)

C : N type (1～255)

CLASS 名

(4) 注意事項

表示する順序はユーザーが本コマンドで該当クラスの可視性を設定した順に従う。

3.7.9 CLASS DYNAMIC ATTRIBUTE TYPE 2 コマンド => 取扱説明書 6.3.62

(1) 機能

CLASS の強張性の設定

(2) Prolog command 形式

class_dyn _attr_typ _2 (I, C)

(3) Arguments の説明

I : N type (0,1)

強張性の設定

I=1：プリンキングの開始 (HIGHLIGHTING ON)

I=0：プリンキングの終了 (HIGHLIGHTING OFF)

C : N type (1～255)

CLASS 名

(4) 注意事項

なし

3.7.10 CLASS DYNAMIC ATTRIBUTE TYPE 3　コマンド　　=> 取扱説明書 6.3.63

(1) 機能

クラスの検出性の設定

(2) Prolog command　形式

class _dyn _attr_typ _3(I,C)

(3) Arguments の説明

I　：N type (0,1)

クラスをライトペン等で検出可能とするか否かの設定

I=0：検出不可能とする。(DETECTABILITY OFF)

I=1：検出可能とする。　(DETECTABILITY ON　)

C　：N type (1～255)

CLASS 名

(4) 注意事項

なし

3.8 SEGMENT コマンド

本章においては、(Retained)SEGMENT 関連の、コマンドについての説明と、Prologで書かれたcommand の形式について述べる

SEGMENT とは、ボディー座標で定義された、ID又はプリミティブの集まりで、論理的まとまりのある図形単位のことである。

SEGMENT に対する属性には、どのような MODELLING属性を、SEGUMENTに対して許すかを、SEGMENT を定義した時点で静的に定める静的属性と、SEGMENT の表示，強調（ブリンキング）及び（ライトペン等による）検出を動的に制御する動的属性、及び、SEGMENT の回転、平行移動や拡大／縮小といったMODELLING 変換の属性（MODELLING 属性）を持つ。

SEGMENT は、CALL SEGMENTコマンドによって、サブルーチン化できる。この時のネスティングは 32767レベルまで可能で、呼ばれる側のSEGMENT のMODELLING 属性やCLASS に対する属性は無視される。即ちSEGUMENTの持つプリミティブに関する情報だけがサブルーチン化される。

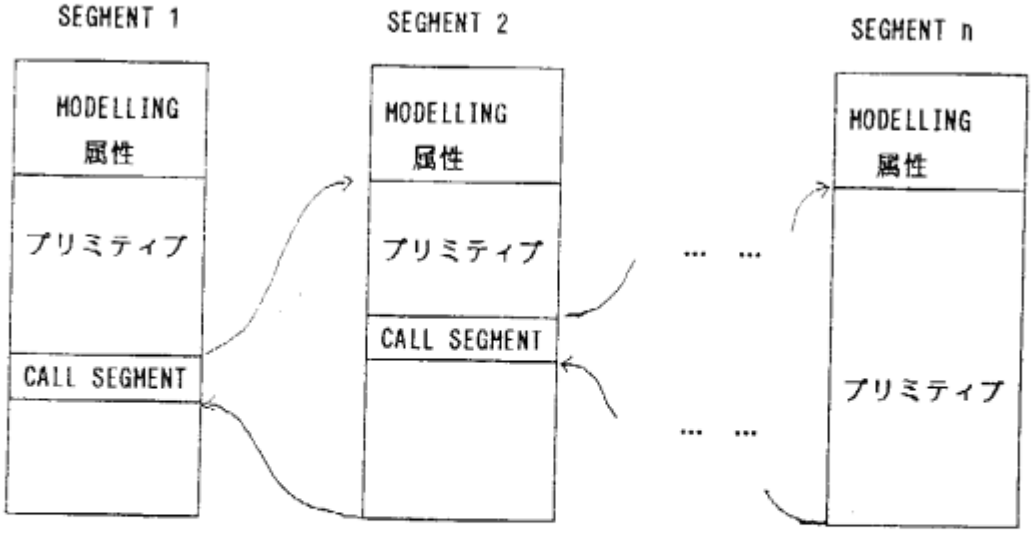


図 1.8 SEGMENT のサブルーチン化

3.8.1 SEGMENT DIFINITION コマンド

=> 取扱説明書 6.3.29

(1) 機能

SEGMENT の定義を開始する

(2) Prolog command 形式

seg_def(S)

(3) Arguments の説明

S : N type (1～255)

定義するSEGMENT 名

(4) 注意事項

1) DEFINITION中は、CLASS 及び他のSEGMENT に対する操作は行えない

2) SEGMENT の修正はID単位で行う。

(SEGMENT に直接定義されたPRIMITIVESの削除や、
text data の修正は行えない)

3.8.2 EDIT SEGMENT コマンド

=> 取扱説明書 6.3.30

(1) 機能

SEGMENT の修正を開始する。

（すでにあるSEGMENT をopenする）

(2) Prolog command 形式

edit_class (S)

(3) Arguments の説明

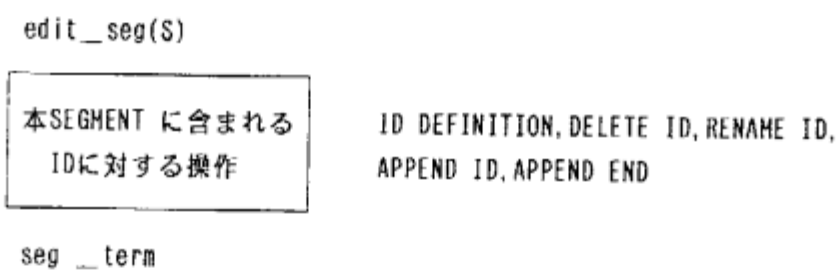
S : N type (1～32767)

修正するSEGMENT 名

(4) 注意事項

1) EDIT 中は、CLASS 及び他のSEGMENT に対する操作は、行えない

2) SEGMENTに対する修正は、SEGMENT 単位で行う。



3.8.3 SEGMENT TERMINATOR コマンド

=> 取扱説明書 6.3.35

(1) 機能

open中のSEGMENT を閉じる

(2) Prolog command 形式

seg _term

(3) Arguments の説明

なし

(4) 注意事項

なし

3.8.4 CALL SEGMENT コマンド

=> 取扱説明書 6.3.31

(1) 機能

SEGMENT の階層構造を定義する

(2) Prolog command 形式

call_seg(S)

(3) Arguments の説明

S : N Type (1～32767)

子となるSEGMENT 名

(4) 注意事項

1) SEGMENTの定義中にのみ有効。

2) call される側のSEGMENT が未登録の場合、本処理は無視される

3.8.5 DELETE SEGMENT コマンド

=> 取扱説明書 6.3.32

(1) 機能

指定されたSEGUMENTを削除する。

(2) Prolog command 形式

del _seg(S)

(3) Arguments の説明

S : N Type (1～32767)

削除するSEGMENT 名を指定する

(4) 注意事項

1) CLASS 化の情報は削除されない
2) RETAINEDモードでのみ発行可能
3) 削除するSEGMENT がopen中はエラーとなる

3.8.6 DELETE SEGMENT ALL コマンド

=> 取扱説明書 6.3.33

(1) 機能

全SEGMENT の削除をする。

(2) Prolog command 形式

del _seg _all

(3) Arguments の説明

なし

(4) 注意事項

1) CLASS 化の情報は削除されない
2) RETAINEDモードでのみ発行可能
3) 削除するSLGMENT がopen中はエラーとなる

3.8.7 RENAME SEGMENT コマンド

=> 取扱説明書 6.3.34

(1) 機能

SEGMENT 名を変更する。

(2) Prolog command 形式

ren _seg (Old, New)

(3) Arguments の説明

Old , New : N Type (1～32767)

Old : 旧SEGMENT 名

New : 新SEGMENT 名

(4) 注意事項

SEGMENT に対する全ての属性は引き継がれる。

3.8.8 SEGMENT STATIC ATTRIBUTE コマンド

=> 取扱説明書 6.3.67

(1) 機能

CLASS に許される像変換の設定

(2) Prolog command 形式

stat_attr (I, S)

(3) Arguments の説明

I : N type (0～3)

I=0 : NO MODELLING TRANSFORMATION

I=1 :

I=2 : 2D MODELLING TRANSFORMATION (拡大／縮小、回転、移動をゆるす)

I=3 : 3D MODELLING TRANSFORMATION (拡大／縮小、回転、移動をゆるす)

S : N type (1～32767)

SEGMENT 名

(4) 注意事項

動的に変更はできない。

3.8.9 DYNAMIC ATTRIBUTE TYPE 1 コマンド => 取扱説明書 6.3.68

(1) 機能

SEGMENT の可視性の設定

(2) Prolog command 形式

dyn _attr_typ _1(I,S)

(3) Arguments の説明

I : N type (0,1)

可視性の設定

I=1：可視SEGMENT である。 (VISIBILITY ON)

I=0：可視SEGMENT でない。 (VISIBILITY OFF)

S : N type (1～32767)

SEGMENT 名

(4) 注意事項

表示する順序はユーザーが本コマンドで該当SEGMENT の可視性を設定した順に従う。

3.8.10 DYNAMIC ATTRIBUTE TYPE 2　コマンド　　=> 取扱説明書 6.3.69

(1) 機能

SEGMENT の強張性の設定

(2) Prolog command　形式

dyn _attr_typ _2 (I, S)

(3) Arguments の説明

I　：N type (0,1)

強張性の設定

I=1：ブリンキングの開始　(HIGHLIGHTING ON　)

I=0：ブリンキングの終了　(HIGHLIGHTING OFF)

S　：N type (1～32767)

SEGMENT 名

(4) 注意事項

なし

3.8.11 DYNAMIC ATTRIBUTE TYPE 3　コマンド　　=> 取扱説明書 6.3.70

(1) 機能

SEGMENT の検出性の設定

(2) Prolog command　形式

dyn _attr_typ _3(I,S)

(3) Arguments の説明

I　：N type (0,1)

SEGMENT をライトペン等で検出可能とするか否かの設定

I=0：検出不可能とする。(DETECTABILITY OFF)

I=1：検出可能とする。　(DETECTABILITY ON　)

S　：N type (1～32767)

SEGMENT 名

(4) 注意事項

なし

3.8.12 MODELLING TRANSFORMATION 2　コマンド　　=> 取扱説明書 6.3.67

(1) 機能

2DにおけるMODELLING 変換を定める

(2) Prolog command　形式

model _trans _2(SEG,Sx,Sy,R,Tx,Ty)

(3) Arguments の説明

SEG　: N type (1～32767)

SEGMENT 名

Sx,Sy : scale type (1/256 ～255)

X軸 Y軸方向の拡大／縮小

R　: degree type (-360 ～360)

回転量を指定する

Tx,Ty : Z type (-32767 ～32767)

X軸 Y軸方向の移動量

(4) 注意事項

最初に定義されたSEGMENT に対して変換を行う

3.8.13 MODELLING TRANSFORMATION 3　コマンド　　=> 取扱説明書 6.3.68

(1) 機能

3DにおけるMODELLING 変換を定める

(2) Prolog command　形式

model _trans _3(SEG,Sx,Sy,Sz,Rx,Ry,Rz,Tx,Ty,Tz)

(3) Arguments の説明

SEG　：N type (1～32767)

SEGMENT 名

Sx,Sy,Sz：scale type (1/256 ～255)

X軸 Y軸 Z軸方向の拡大／縮小

Rx,Ry,Rz：degree type (-360 ～360)

回転量を指定する (circle_ 3のそれに同じ)

Tx,Ty,Tz：Z type (-32767 ～32767)

X軸 Y軸 Z軸方向の移動量

(4) 注意事項

最初に定義されたSEGMENT に対して変換を行う

3.9 ID コマンド

本章においては、ID関連のコマンドについての説明と、Prologで書かれたcommand形式について述べる。

IDは、１つ以上のプリミティブを含み、ライトペン等による検出や強張を行うことのできる図形の単位である。

IDは、１つのSEGMENT 又はIDに含まれていなければならない。
（図 3.9.1参照）

ID関連のコマンドはmult_repl_text_dataを除きIDを含む（一番下の階層にある）SEGMENT がオープン中にのみ有効である。

すでに定義されているIDに新しくプリミティブを追加する時は、APPEND ID コマンドにより、そのIDの下に新しいIDを追加しその下に新しいプリミティブを追加する。
（図 3.9.2参照）

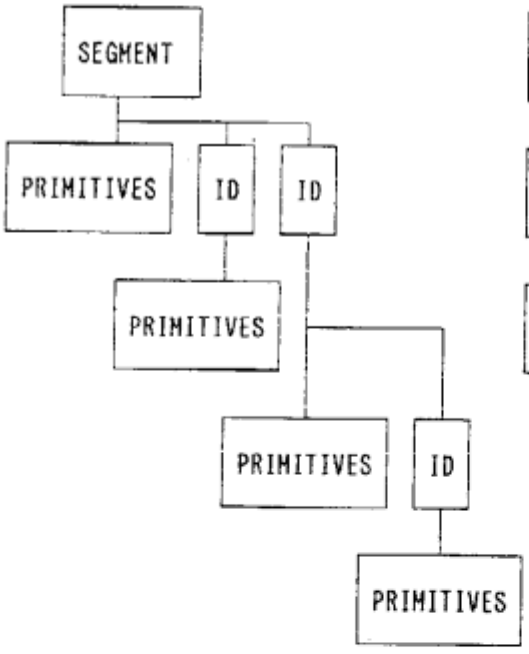


図 3.9.1 IDの階層

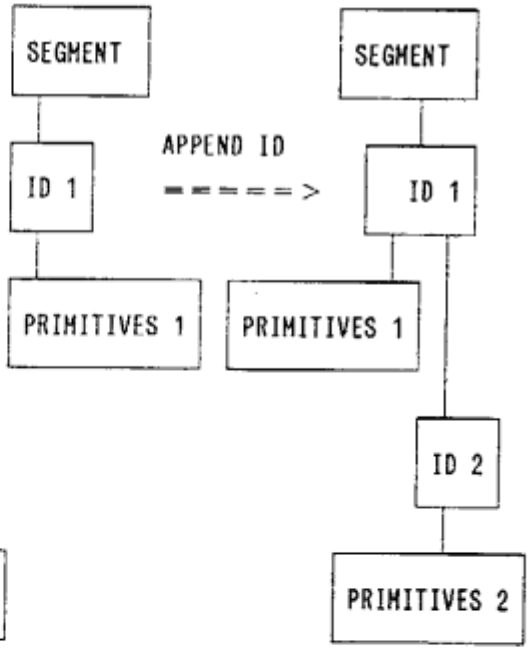


図 3.9.2 PRIMITIVESの追加

3.9.1 ID DEFINITION　コマンド　　=> 取扱説明書 6.3.36

(1) 機能

IDの定義を開始する。

(2) Prolog command　形式

id_def (H,ID)

(3) Arguments の説明

H　：N type (0,1)

H=0：IDを強張しない　(HILENGTH OFF)

H=1：IDを強張する　(HILENGTH ON)

ID　：N type (1～32767)

ID名称を指定する。

(4) 注意事項

- セグメントがオープン中でなければエラー。
- 定義の終了は次のid_def 又はseg _termである。
- EDIT SEGMENTコマンド発行後の場合は追加モードとなる。

3.9.2 RENAME ID コマンド

=> 取扱説明書 6.3.37

(1) 機能

ID名の変更

(2) Prolog command 形式

ren _id (Old,New)

(3) Arguments の説明

Old,New : N type (1～32767)

Old : 旧ID名

New : 新ID名

(4) 注意事項

IDに対する属性も全て引き継がれる。

階層構造には影響を与えない。

該当セグメントがEDIT中でなければならない。

3.9.3 DELETE ID　コマンド　　=> 取扱説明書 6.3.38

(1) 機能

IDを削除する。

(2) Prolog command　形式

del _id (ID)

(3) Arguments の説明

ID：N type （1～32767）

削除するIDを指定する。

(4) 注意事項

階層構造に影響しない。

該当セグメントがEDIT中のみ有効。

3.9.4 APPEND ID コマンド

=> 取扱説明書 6.3.42

(1) 機能

指定したIDにIDを追加する（PRIMITIVESをIDの下に追加する）。

(2) Prolog command 形式

app _id (ID)

(3) Arguments の説明

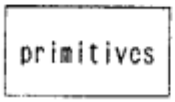
ID : N type (1～32767)

primitivesを追加するIDを指定する。

(4) 注意事項

app _id (ID 1)

id_def (ID 2)



（図 3.9.2参照）

app _end

3.9.5 APPEND END コマンド

=> 取扱説明書 6.3.43

(1) 機能

primitivesの追加終了

(2) Prolog command 形式

app _end

(3) Arguments の説明

なし

(4) 注意事項

なし

3.9.6 REPLACE TEXT DATA コマンド

=> 取扱説明書 6.3.44

(1) 機能

指定されたIDに含まれるtext data 群を新しいdataに置き換えを開始する。

(2) Prolog command 形式

repl_text_data (ID)

(3) Arguments の説明

ID : N type (1～32767)

置き換えを行うIDS 名

(4) 注意事項

rep _text_data

text_ent (text _ent _char又はtext_ent _sym)

3.9.7 MULT REPLACE FOR TEXT DATA コマンド　　=> 取扱説明書 6.3.45

(1) 機能

text data の置き換えを一括して行う。

(2) Prolog command 形式

mult_repl_data (LIST)

(3) Arguments の説明

LIST=[ENT1,ENT2,……,ENTn]

ENT=[S,ID,T,Data]

S　：N type (1～32767)

修正するIDを含むセグメントを指定

ID：N type (1～32767)

修正するIDを指定

T ：atomic (a/s)

T=a ：character を指定

この場合Dataは文字列

T=s ：symbolを指定

この場合Dataはsymbolコード

(4) 注意事項

セグメントがオープン中はエラー

3.9.8 ID DYNAMIC ATTRIBUTE TYPE　コマンド　　=> 取扱説明書 6.3.73

(1) 機能

IDの強調性の設定

(2) Prolog command　形式

id_dyn _attr_typ (H,S,I)

(3) Arguments の説明

H　：N type (0,1)

H=1：ブリンキング開始(HIGHLIGHTING ON)

H=0：ブリンキング終了(HIGHLIGHTING OFF)

S,I ：N type (1～32767)

S　　：SEGMENT を指定

I　：IDを指定

(4) 注意事項

3.10 VIEWING コマンド

本章においては、VIEWING の設定に関するcommand についての説明と、PROLOGで書かれたcommand の形式について述べる

3.10.1 INIT VIEWING コマンド

=> 取扱説明書 6.3.81

(1) 機能

viewing 情報の初期化を行う

(2) Prolog command 形式

init_wiew (I)

(3) Arguments の説明

I : N type (1～8)

viewing 番号を指定する

(4) 注意事項

なし

3.10.2 SELECT VIEWING コマンド

=> 取扱説明書 6.3.82

(1) 機能

viewing 番号の選択を行う

(2) Prolog command 形式

sel _view(I)

(3) Arguments の説明

I : N type (1～8)

viewing 番号を指定する

(4) 注意事項

なし

3.10.3 WINDOW コマンド　　=> 取扱説明書 6.3.77

(1) 機能

ワールド座標のクリッピングウインドウを定義する

(2) Prolog command 形式

wind(X1,Y1,X2,Y2)

(3) Arguments の説明

X1,Y1,X2,Y2 ：Z type (-32768 ～ 32767)

クリッピングウインドウ NO 対角線方向の頂点

(X1,Y1) , (X2,Y2)

を指定する

(4) 注意事項

1) wind_term ,look_point _termによりクリッピングウインドウの移動が行える

2) 本コマンドでwindowの大さを変更する場合viewing matrixを再定義する

3.10.4 VIEWPORT 2　コマンド　　=> 取扱説明書 6.3.78

(1) 機能

2D図形をdisplay のどの位置に表示するか(DEVICE 座標のVIEWPORT) の指定

(2) Prolog command　形式

view_ 2(X1,Y1,X2,Y2)

(3) Arguments の説明

X1,X2 : N type (0～5471)

Y1,Y2 : N type (0～4093)

(X1,Y1) , (X2,Y2)

で定まるdisplay 上の長方形の部分に表示する

(4) 注意事項

本コマンドでVIEWPORTを変更する場合viewing matrixを再定義する

3.10.5 VIEWPORT 3　コマンド　　=> 取扱説明書 6.3.79

(1) 機能

3D図形をdisplay のどの位置に表示するか(DEVICE 座標のVIEWPORT) の指定

(2) Prolog command　形式

view_ 2(X1,Y1,Z1,X2,Y2,Z2)

(3) Arguments の説明

X1,X2 : N type (0～5471)

Y1,Y2 : N type (0～4093)

Z1,Z2 : N type (0～5471)

(X1,Y1,Z1), (X2,Y2,Z2)

で定まるdisplay 上の長方形の部分に表示する

(4) 注意事項

本コマンドでVIEWPORTを変更する場合viewing matrixを再定義する

3.10.6 FRONT/BACK DISTANCE コマンド　　=> 取扱説明書 6.3.79

(1) 機能

FRONT DISTANCEとBACK DISTANCE の指定

(2) Prolog command　形式

front _back_dist(F,B)

(3) Arguments の説明

F,B ：Z type (-32768 ～ 32767)

F　：FRONT DISTANCE(FRONTクリッピングPLANと視点との距離) を指定する

B　：BACK　DISTANCE(BACK クリッピングPLANと視点との距離) を指定する

(4) 注意事項

本コマンドでVIEWPORTを変更する場合viewing matrixを再定義する

3.10.7 VIEWING MATRIX 2　コマンド　　=> 取扱説明書 6.3.79

(1) 機能

2DにおけるVIEWING MATRIXの指定

(2) Prolog command　形式

view_mat __ 2(Typ,Mat)

(3) Arguments の説明

Typ ：

VIEWING MATRIXのタイプでp,n,v の３タイプがある

Mat ：各要素はmatrix element type (-32768 ～ 32767)

VIEWING MATRIXを指定する

(4) 注意事項

なし

3.10.8 VIEWING MATRIX 3　コマンド　　=> 取扱説明書 6.3.79

(1) 機能

3DにおけるVIEWING MATRIXの指定

(2) Prolog command　形式

view_mat __ 3(Typ,Mat)

(3) Arguments の説明

Typ ：

VIEWING MATRIXのタイプでc,e,r,n,p の５タイプがある

Mat ：各要素はmatrix element type (-32768 ～ 32767)

VIEWING MATRIXを指定する

(4) 注意事項

なし

3.11 DEVICE　コマンド

本章においては、deviceの初期化等のcommand についての説明と、Prologで書かれたcommand の形式について述べる。

SEILLAC では、ジョイスティクやライトペン等の物理deviceを論理分類し、data形式の標準化を行って、物理deviceからの独立を図っている。１つの論理deviceに対して、複数の物理deviceが存在する場合があるため、論理入力deviceは、（論理入力）device CLASSとdevice NO の組で現す。　（表 3.11.1,表 3.11.2 参照）

以下に、SEILLAC の各論理deviceについて述べる。

(1) イベントdevice

オペレータ操作によって発生する１つの事象を、メニューやシンボル情報として取り扱うdeviceで、以下の４つのdevice CLASSに細分される。

1) PICK論理device

画面に表示中の図形群の中から、ある特定の図形を選択する（ピッキング）機能をもつdeviceであり、物理deviceには、ジョイスティクとライトペンがある。

ピッキングの機能によって、図形の変更（拡大／縮小、回転、移動）の際の対象図形の選択を行える。又、図形の名称は、アプリケーションprogramに、報告されるので、他の目的にも使用できる。

2) KEYBOARD論理device

文字列入力機能をもつdeviceであり、物理deviceには、（SEILLAC に標準装備されている）キーボードがある。

キー入力した文字は、キーボード上の（80桁）液晶にエコーバックされ、復帰キーを押下した時点で eventとしてdataが発生する。

3) BUTTON論理device

アプリケーションprogram で規定した機能を、キーのワンタッチ押下で、アプリケーションprogram に報告する機能をもつdeviceであり、物理deviceには、ファンクションキーがある。

4) STROKE論理device

オペレータ操作によって、点列として座標値列を、発生する機能を有するdeviceであるが、現在ICOTには、これに相当する物理deviceはない。

(2) サンプルdevice

オペレータ操作によって、変化量等の数値を読み取るdeviceで、以下の３つのdevice CLASSに細分される。

1) LOCATOR 論理device

2D/3D の座標値の入力の機能をもつdeviceであり、物理deviceは、ジョイスティクである。

2) VALUATOR論理device

scaler量の入力の機能をもつdeviceであり、物理deviceは、ジョイスティクである。

SEGMENT の拡大／縮小（スケーリング），回転（ローテーション），平行移動（トランジション）といった MODELLING変換をともなうインタラクションや、CLASS の視点の遠近移動（ズーミング），回転移動，平行移動（パンニング）といった、画面上への表示を、変化させるインタラクションに用いる。

3) SUB VALUATOR論理device

VALUATOR論理deviceと同じ機能をもつdeviceであるが、離散型の変移量を発生する点が異なる、物理deviceは、ジョイスティクである。

この論理deviceは、ディスプレイプロセッサのサポートするインタラクションでは、使用しないが、アプリケーションprogram で、他の目的に使用できる論理deviceである。

論理 device CLASS 名	指定値（10進）
PICK 論理 device	10
KEYBOARD 論理 device	20
BUTTON 論理 device	30
STROKE 論理 device	40
LOCATOR 論理 device	50
VALUATOR 論理 device	60
SUB VALUATOR論理 device	70

表 3.11.1

物理 device			論理 device	device NO
キーボード	キー入力		KEYBOARD (20)	10
ファンクションキー	ファンクションキー入力		BUTTON (30)	1～16
ライトペン	ピッキング		PICK (10)	1
ジョイスティック	ピッキング		PICK (10)	2
	ドラッキング ポジショニング		LOCATOR (50)	2
	スケーリング ローテーション トランジション	MODELLING	VALUATOR (60)	2
	ズーミング 視点の回転 パンニング	視点移動		
	値の読み取り		SUB VALUATOR(70)	1

表 3.11.2

3.12.1 INITIALIZE DEVICE　コマンド　　=> 取扱説明書 6.3.91

(1) 機能

入力 device の初期化を行う

(2) Prolog command　形式

init_dev(I,J)

(3) Arguments の説明

I,J : N type (1～255)

I　: device CLASSを指定する

J　: device NO.　を指定する

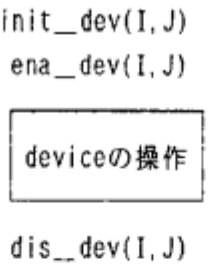
(4) 注意事項

1) 初期化されるのは、次のものである

イベント　キュー

2) BUTTON device に対しては、CLASS 毎のinitializeが必要である

device NO.　= 0



3.11.2 ENABLE DEVICE　コマンド　　=> 取扱説明書 6.3.92

(1) 機能

入力 device を有効にする

(2) Prolog command　形式

ena _dev(I,J)

(3) Arguments の説明

I,J ：N type (1～255)

I　：device CLASSを指定する

J　：device NO.　を指定する

(4) 注意事項

BUTTON device に対しては、CLASS 毎に必要である

device NO.　= 0

3.11.3 DISEBLE DEVICE　コマンド　　=> 取扱説明書 6.3.92

(1) 機能

入力 device を無効にする

(2) Prolog command　形式

dis _dev(I,J)

(3) Arguments の説明

I,J ：N type (1～255)

I　：device CLASSを指定する

J　：device NO.　を指定する

(4) 注意事項

BUTTON device に対しては、CLASS 毎に必要である

device NO.　= 0

3.11.4 TERMINATE DEVICE コマンド

=> 取扱説明書 6.3.92

(1) 機能

入力 device を終了する

(2) Prolog command 形式

term_dev(I,J)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを指定する

J : device NO. を指定する

(4) 注意事項

BUTTON device に対しては、CLASS 毎に必要である

device NO. = 0

3.12 ECHO　コマンド

本章においては、ECHOの設定に関するcommand についての説明と、PROLOGで書かれたcommand の形式について述べる

3.12.1 SET ECHO TYPE コマンド　　=> 取扱説明書 6.3.97

(1) 機能

指定した入力 device に対するエコータイプを設定する

(2) Prolog command　形式

set _echo_typ(T,I,J)

(3) Arguments の説明

T　：N type (1～255)

エコータイプの指定を行う

I,J ：N type (1～255)

I　：device CLASSを指定する

J　：device NO.　を指定する

(4) 注意事項

なし

3.12.2 SET ECHO CLASS コマンド

=> 取扱説明書 6.3.98

(1) 機能

指定した入力 device のエコーをCLASS に対して行う

(2) Prolog command 形式

set _echo_class(I,J,C)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを指定する

J : device NO. を指定する

C : N type (1～255)

CLASS 名を指定する

(4) 注意事項

なし

3.12.3 SET ECHO SEGMENT コマンド　　=> 取扱説明書 6.3.99

(1) 機能

指定した入力 device のエコーをSEGMENT に対して行う

(2) Prolog command 形式

set _echo_seg(I,J,S)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを指定する

J : device NO. を指定する

S : N type (1～255)

SEGMENT 名を指定する

(4) 注意事項

なし

3.12.4 SET ECHO ID コマンド　　=> 取扱説明書 6.3.100

(1) 機能

指定した入力 device のエコーを(SEGMENT内の) IDに対して行う

(2) Prolog command　形式

set __echo__id(I,J,ID)

(3) Arguments の説明

I,J ：N type (1～255)

I　：device CLASSを指定する

J　：device NO.　を指定する

ID　：N type (1～255)

ID名を指定する

(4) 注意事項

なし

3.13 同期／非同期　コマンド

本章においては、同期／非同期の問合せに関するcommand についての説明と、PROLOG で書かれたcommand の形式について述べる

3.13.1 FLUSH ALL EVENT コマンド　　=> 取扱説明書 6.3.95

(1) 機能

EVENT キューをすべて削除する

(2) Prolog command　形式

flush _all _eve

(3) Arguments の説明

なし

(4) 注意事項

ローカル　インタラクションの先頭で発行する

3.13.2 FLUSH DEVICE EVENT コマンド　　=> 取扱説明書 6.3.98

(1) 機能

指定した入力 device のEVENT キューを除去する

(2) Prolog command 形式

flush _dev _eve(I,J)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを指定する

J : device NO. を指定する

(4) 注意事項

ローカル インタラクションの先頭で発行する

3.13.3 AWAIT EVENT コマンド

=> 取扱説明書 6.3.115

(1) 機能

EVENT の発生を待つ

(2) Prolog command 形式

await _eve(T)

(3) Arguments の説明

T　：N type (1～32767)

timeを指定する（単位は秒）

(4) 注意事項

なし

3.13.4 REPORT DATA EVENT コマンド

=> 取扱説明書 6.3.124.9

(1) 機能

EVENT の発生したdeviceを報告する

(AWAIT EVENT コマンドに対するREPORT)

(2) Prolog command 形式

rep _data_eve(I,J)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを返す

J : device NO. を返す

(4) 注意事項

非同期方式の場合にのみ有効

3.13.5 AWAIT PICK　コマンド　　=> 取扱説明書 6.3.117

(1) 機能

PICK論理deviceのEVENT の発生を待つ

(2) Prolog command　形式

await _pick(J,T)

(3) Arguments の説明

J　：N type (1～255)

EVENT の発生を待つPICK論理deviceの、device NO 指定する

T　：N type (1～32767)

timeを指定する（単位は秒）

(4) 注意事項

本コマンドで指定されたdeviceに、ECHOが設定されていれば、処理は、その定義に従う

3.13.6 REPORT DATA PICK　コマンド　　=> 取扱説明書 6.3.124.1

(1) 機能

EVENT の発生したdeviceとPICKの情報（CLASS/SEGMENT/ID）を報告する

（AWAIT PICK　コマンドに対するREPORT）

(2) Prolog command　形式

rep _data_pick(I,J,C,S,ID)

(3) Arguments の説明

I,J ：N type (1～255)

I　：device CLASSを返す

J　：device NO.　を返す

C,S,ID：N type (1～32767)

C　：CLASS 名を返す

S　：SEGMENT 名を返す

ID ：ID名を返す

(4) 注意事項

1) IDがPICKされた場合CLASS/SEGMENT/ID名を返す

2) SEGMENT がPICKされた場合CLASS/SEGMENT 名を返す

3.13.7 SWAIT KEYBOARD　コマンド　　=> 取扱説明書 6.3.118

(1) 機能

KEYBOARD論理deviceのEVENT の発生を待つ

(2) Prolog command　形式

await _key (J,T)

(3) Arguments の説明

J　：N type (1～255)

EVENT の発生を待つKEYBOARD論理deviceの、device NO 指定する

T　：N type (1～32767)

timeを指定する（単位は秒）

(4) 注意事項

本コマンドで指定されたdeviceに、FCHOが設定されていれば、処理は、その定義に従う

3.13.8 REPORT DATA KEYBOARD コマンド

=> 取扱説明書 6.3.124.2

(1) 機能

EVENT の発生したdeviceと入力stringを報告する
（AWAIT KEYBOARD コマンドに対するREPORT）

(2) Prolog command 形式

rep _data_key (I,J,STRING)

(3) Arguments の説明

I,J ：N type (1～255)
I ：device CLASSを返す
J ：device NO. を返す
STRING：string型の文字列
入力stringを返す

(4) 注意事項

なし

3.13.9 AWAIT BUTTON コマンド

=> 取扱説明書 6.3.119

(1) 機能

BUTTON論理deviceのEVENT の発生を待つ

(2) Prolog command 形式

await _but (J,T)

(3) Arguments の説明

J ：N type (1～255)

EVENT の発生を待つBUTTON論理deviceの、device NO 指定する

T ：N type (1～32767)

timeを指定する（単位は秒）

(4) 注意事項

本コマンドで指定されたdeviceに、ECHOが設定されていれば、処理は、その定義に従う

3.13.10 REPORT DATA BUTTON コマンド => 取扱説明書 6.3.124.3

(1) 機能

EVENT の発生したdeviceを報告する

(AWAIT BUTTON コマンドに対するREPORT)

(2) Prolog command 形式

rep _data_but (I,J)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを返す

J : device NO. を返す

(4) 注意事項

なし

3.13.11 AWAIT BUTTON&LOCATOR コマンド　　=> 取扱説明書 6.3.119

(1) 機能

LOCATOR 論理deviceのEVENT の発生を待つ

(2) Prolog command 形式

await _but _loc (J,T)

(3) Arguments の説明

J　：N type (1～255)

EVENT の発生を待つLOCATOR 論理deviceの、device NO 指定する

T　：N type (1～32767)

timeを指定する（単位は秒）

(4) 注意事項

本コマンドで指定されたdeviceに、ECHOが設定されていれば、処理は、その定義に従う

3.13.12 REPORT DATA BUTTON&LOCATOR コマンド => 取扱説明書 6.3.124.5

(1) 機能

EVENT の発生したBUTTON device NOとLOCATOR deviceの値を報告する
(AWAIT BUTTON&LOCATOR コマンドに対するREPORT)

(2) Prolog command 形式

rep _data_but _loc (I,J,B,X,Y,Z)

(3) Arguments の説明

I,J,B : N type (1～255)
I : device CLASSを返す
J : device NO. を返す
B : BUTTON論理device NO. を返す
X,Y,Z : Z type (-32768 ～ 32767)
LOCATOR deviceから読み取った値を返す

(4) 注意事項

なし

3.13.13 GET DATA コマンド

=> 取扱説明書 6.3.116

(1) 機能

入力の要求を行う

(2) Prolog command 形式

get __data

(3) Arguments の説明

なし

(4) 注意事項

なし

3.13.14 REPORT DATA LOCATOR コマンド　　=> 取扱説明書 6.3.124.7

(1) 機能

EVENT の発生したLOCATOR deviceの値を報告する

（GET DATAコマンドに対するREPORT）

(2) Prolog command 形式

rep _data_but _loc (I,J,X,Y,Z)

(3) Arguments の説明

I,J ：N type (1～255)

I ：device CLASSを返す

J ：device NO. を返す

X,Y,Z ：Z type (-32768 ～ 32767)

LOCATOR deviceから読み取った値を返す

(4) 注意事項

3.14 応答　コマンド

本章においては、SEILLAC に対する応答command についての説明と、PROLOGで書かれたcommand の形式について述べる

3.14.1 REPORT CURRENT POSITION 2 コマンド　　=> 取扱説明書 6.3.126.1

(1) 機能

現在のビーム位置を応答する

(2) Prolog command　形式

rep __cur __pos __2(X,Y)

(3) Arguments の説明

X,Y ：Z　type (-32768 ～32767)

（ワールド座標の）現在のビーム位置を返す

(4) 注意事項

なし

（開発中）

3.14.2 REPORT CURRENT POSITION 3 コマンド　　=> 取扱説明書 6.3.126.2

(1) 機能

現在のビーム位置を応答する

(2) Prolog command　形式

rep _cur _pos _3(X,Y,Z)

(3) Arguments の説明

X,Y,Z　: Z　type (-32768 ～32767)

（ワールド座標の）現在のビーム位置を返す

(4) 注意事項

なし

3.14.3 REPORT CURRENT ERROR　コマンド　　=> 取扱説明書 6.3.126.22

(1) 機能

１番最近に発生したエラー情報を応答する

(2) Prolog command　形式

rep _cur _err(L,N)

(3) Arguments の説明

L　：N　type (1～255)

エラーレベルを返す

N　：N　type (1～32767)

エラー番号を返す

(4) 注意事項

なし

3.14.4 REPORT CLASS VISIBILITY コマンド　　=> 取扱説明書 6.3.126.32

(1) 機能

指定したCLASS の可視属性の状態を応答する

(2) Prolog command　形式

rep _class _visi(C,B)

(3) Arguments の説明

C　　: N　type (1～255)

CLASS 名を指定する

B　　: N　type (0,1)

B=0 : VISIBILITY OFF

B=1 : VISIBILITY ON

(4) 注意事項

class _dyn _attr_typ _ 1で設定した属性に対する応答を得る

3.14.5 REPORT CLASS HIGHLIGHTING コマンド　　=> 取扱説明書 6.3.126.33

(1) 機能

指定したCLASS の強調属性の状態を応答する

(2) Prolog command　形式

rep _class _high(C,B)

(3) Arguments の説明

C　　: N　type (1～255)

CLASS 名を指定する

B　　: N　type (0,1)

B=0 : HIGHLIGHTING OFF

B=1 : HIGHLIGHTING ON

(4) 注意事項

class _dyn _attr_typ _2 で設定した属性に対する応答を得る

3.14.6 REPORT CLASS DETECTABILITY コマンド　　=> 取扱説明書 6.3.126.23

(1) 機能

指定したCLASS の検出属性の状態を応答する

(2) Prolog command 形式

rep _class _detect(C,B)

(3) Arguments の説明

C　：N type (1～255)

CLASS 名を指定する

B　：N type (0,1)

B=0：DETECTABILITY OFF

B=1：DETECTABILITY ON

(4) 注意事項

class _dyn _attr_typ _3 で設定した属性に対する応答を得る

3.14.7 REPORT CLASS TRANSFORMATION TYPE コマンド => 取扱説明書 6.3.126.25

(1) 機能

指定したCLASS の静的属性の状態を応答する

(2) Prolog command 形式

rep _class _trans _typ (C,B)

(3) Arguments の説明

C : N type (1～255)

CLASS 名を指定する

B : N type (0,3)

CLASS の静的属性を返す

(4) 注意事項

class _stat_attr_typ で設定した属性に対する応答を得る

3.14.8 REPORT CLASS TRANSFORMATION 2 コマンド　　=> 取扱説明書 6.3.126.5

(1) 機能

指定したCLASS の2D視野変換の値を応答する

(2) Prolog command　形式

rep _class _trans _2(C,S,R,Tx,Ty)

(3) Arguments の説明

C　：N　type (1～255)

CLASS 名を指定する

S　：scale type (1/255 ～255)

scale 量を返す

R　：vector type (-360 ～360)

回転量を返す

Tx,Ty　：Z type (-32768～32767)

移動量を返す

(4) 注意事項

zoom_trans,wind_trans で設定した値に対する応答を得る

3.14.9 REPORT CLASS TRANSFORMATION 3 コマンド　　=> 取扱説明書 6.3.126.27

(1) 機能

指定したCLASS の3D視野変換の値を応答する

(2) Prolog command　形式

rep _class _trans _3(C,S,Rx,Ry,Rz,Tx,Ty,Tz)

(3) Arguments の説明

C　：N　type (1～255)

CLASS 名を指定する

S　：scale type (1/255 ～255)

scale 量を返す

Rx,Ry,Rz　：vector type (-360 ～360)

回転量を返す

Tx,Ty,Tz　：Z type (-32768～32767)

移動量を返す

(4) 注意事項

zoom_trans,look_point _trans で設定した値に対する応答を得る

3.14.10 REPORT SEGMENT IN THE CLASS コマンド　　=> 取扱説明書 6.3.126.37

(1) 機能

指定したCLASS に登録されているSEGMENT を応答する

(2) Prolog command 形式

rep _seg _in_class(C,LIST)

(3) Arguments の説明

C　: N type (1～255)

CLASS 名を指定する

LIST : 各要素は、N type (1 ～32767)

SEGMENT 名のlistを返す

(4) 注意事項

1) SEGMENT が存在しない場合は、CURRENT ERROR を返す

2) seg _class _def,app _class _mem,del _class _mem によって定義、追加、削除した結果、CLASS に含まれるSEGMENT 名のlistを返す

3.14.11 REPORT RETAINED SEGMENT ALL コマンド => 取扱説明書 6.3.126.34

(1) 機能

登録されているSEGMENT 名を全て応答する

(2) Prolog command 形式

rep _retain_seg _all(LIST)

(3) Arguments の説明

LIST：各要素は、N type (1～32767)

SEGMENT 名のlistを返す

(4) 注意事項

SEGMENT が存在しない場合は、CURRENT ERROR を返す

3.14.12 REPORT OPEN RETAINED NAME コマンド　　=> 取扱説明書 6.3.126.3

(1) 機能

現在 open 中のSEGMENT 名を全て応答する

(2) Prolog command 形式

rep __open__seg __name(S)

(3) Arguments の説明

S　：N type (0～32767)

SEGMENT 名を返す

(4) 注意事項

open中のSEGMENT がない場合は、0 を返す

3.14.13 REPORT SEGMENT VISIBILITY コマンド　　=> 取扱説明書 6.3.126.30

(1) 機能

指定したSEGMENT の可視属性の状態を応答する

(2) Prolog command 形式

rep _seg _visi(S,B)

(3) Arguments の説明

S　: N type (1～32767)

CLASS 名を指定する

B　: N type (0,1)

B=0 : VISIBILITY OFF

B=1 : VISIBILITY ON

(4) 注意事項

dyn _attr_typ _ 1で設定した属性に対する応答を得る

3.14.14 REPORT SEGMENT HIGHLIGHTING コマンド　　=> 取扱説明書 6.3.126.31

(1) 機能

指定したSEGMENT の強調属性の状態を応答する

(2) Prolog command 形式

rep _seg _high(S,B)

(3) Arguments の説明

S　：N type (1～32767)

SEGMENT 名を指定する

B　：N type (0,1)

B=0：HIGHLIGHTING OFF

B=1：HIGHLIGHTING ON

(4) 注意事項

dyn _attr_typ _2 で設定した属性に対する応答を得る

3.14.15 REPORT SEGMENT DETECTABILITY コマンド => 取扱説明書 6.3.126.4

(1) 機能

指定したSEGMENT の検出属性の状態を応答する

(2) Prolog command 形式

rep _seg _detect(S,B)

(3) Arguments の説明

S : N type (1～32767)

SEGMENT 名を指定する

B : N type (0,1)

B=0 : DETECTABILITY OFF

B=1 : DETECTABILITY ON

(4) 注意事項

dyn _attr_typ _3 で設定した属性に対する応答を得る

3.14.16 REPORT SEGMENT TRANSFORMATION TYPE コマンド => 取扱説明書 6.3.126.36

(1) 機能

指定したSEGMENT の静的属性の状態を応答する

(2) Prolog command 形式

rep _seg _trans _typ (S,B)

(3) Arguments の説明

S : N type (1～32767)

SEGMENT 名を指定する

B : N type (0,3)

SEGMENT の静的属性を返す

(4) 注意事項

stat_attr_typ で設定した属性に対する応答を得る

3.14.17 REPORT SEGMENT TRANSFORMATION 2 コマンド　=> 取扱説明書 6.3.126.6

(1) 機能

指定したSEGMENT の2D MODELLING変換の値を応答する

(2) Prolog command 形式

rep _seg _trans _2(C,Sx,Sy,R,Tx,Ty)

(3) Arguments の説明

S　：N type (1～32767)

SEGMENT 名を指定する

Sx,Sy ：scale type (1/255 ～255)

scale 量を返す

R　：vector type (-360 ～360)

回転量を返す

Tx,Ty ：Z type (-32768～32767)

移動量を返す

(4) 注意事項

model _trans _2 で設定した値に対する応答を得る

3.14.18 REPORT SEGMENT TRANSFORMATION 3 コマンド => 取扱説明書 6.3.126.28

(1) 機能

指定したSEGMENT の3D MODELLING変換の値を応答する

(2) Prolog command 形式

rep _seg _trans _3(S,Sx,Sy,Sz,Rx,Ry,Rz,Tx,Ty,Tz)

(3) Arguments の説明

S : N type (1～32767)

SEGMENT 名を指定する

Sx,Sy,Sz : scale type (1/255 ～255)

scale 量を返す

Rx,Ry,Rz : vector type (-360 ～360)

回転量を返す

Tx,Ty,Tz : Z type (-32768～32767)

移動量を返す

(4) 注意事項

model _trans _3 で設定した値に対する応答を得る

3.14.20 REPORT DEVICE STATUS コマンド　　=> 取扱説明書 6.3.126.19

(1) 機能

指定した論理deviceの初期化の状態及びENABLE/DISABLEの状態を応答する

(2) Prolog command　形式

rep _dev _stat(I,J,B,E)

(3) Arguments の説明

I,J ：N　type (1～255)

I　：device CLASSを指定する

J　：device NO を指定する

B,E ：N type (0,1)

論理deviceの初期化の状態及びENABLE/DISABLEの状態を返す

B=0：初期化されていない

B=1：初期化されている

E=0：DISABLE

E=1：ENABLE

(4) 注意事項

3.14.21 REPORT STROKE DIMENSION コマンド

=> 取扱説明書 6.3.126.17

(1) 機能

指定したSTROKE論理deviceの2D/3D を応答する

(2) Prolog command 形式

rep _str _dim (I,J,B)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを指定する

J : device NO を指定する

B : N type (0,1)

STROKE論理deviceの2D/3D を返す

B=0 : 2D

B=1 : 3D

(4) 注意事項

3.14.22 REPORT LOCATOR DIMENSION コマンド　　=> 取扱説明書 6.3.126.18

(1) 機能

指定したLOCATOR 論理deviceの2D/3D を応答する

(2) Prolog command　形式

rep __loc __dim (I,J,B)

(3) Arguments の説明

I,J : N type (1～255)

I : device CLASSを指定する

J : device NO を指定する

B : N type (0,1)

LOCATOR 論理deviceの2D/3D を返す

B=0 : 2D

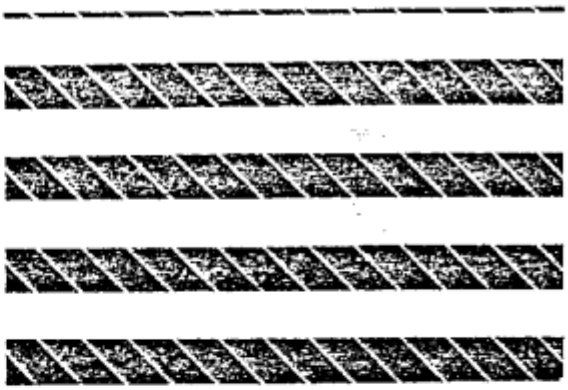
B=1 : 3D

(4) 注意事項

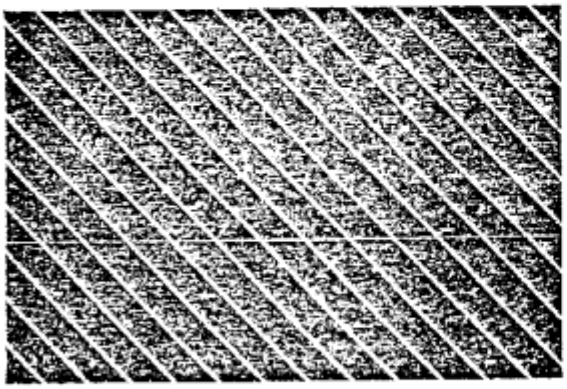
付録１　　中塗りの柄と線分の種類

1

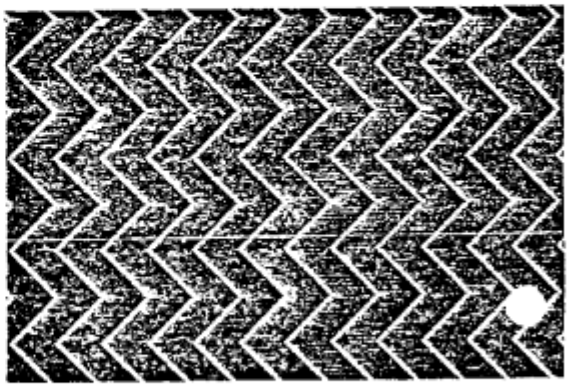
2



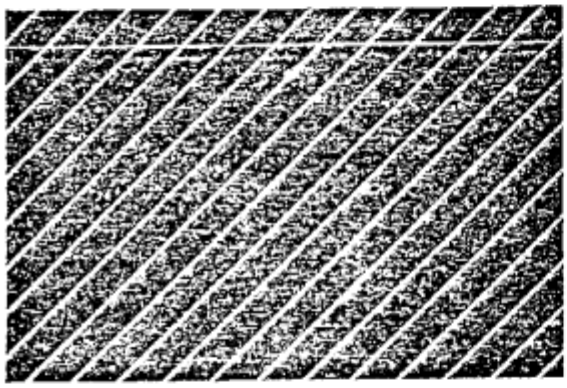
3



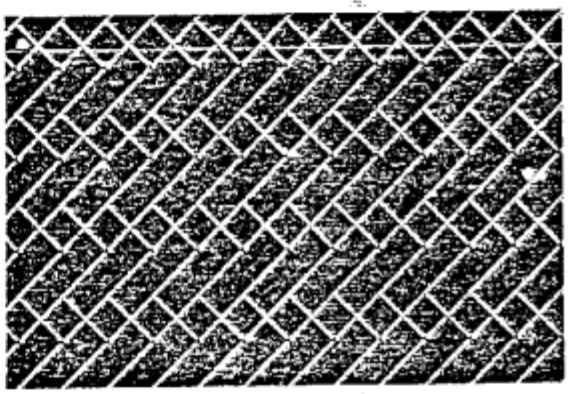
4



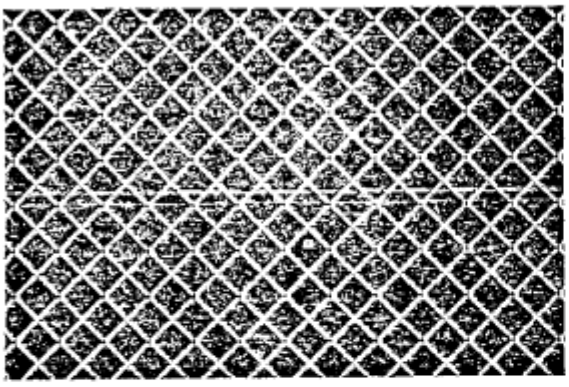
5



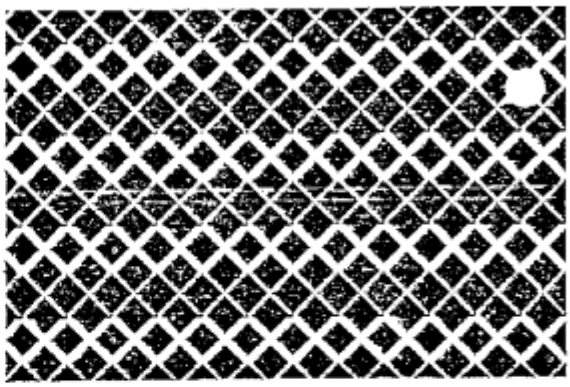
6



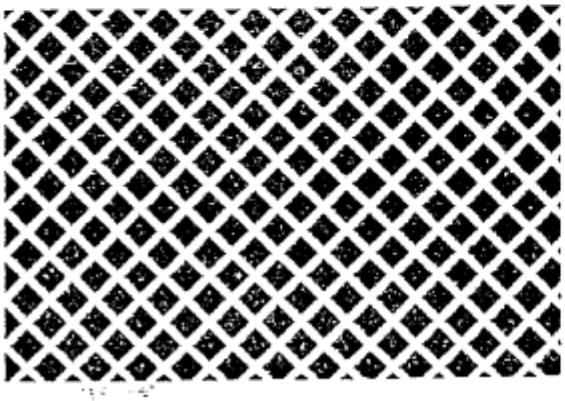
7



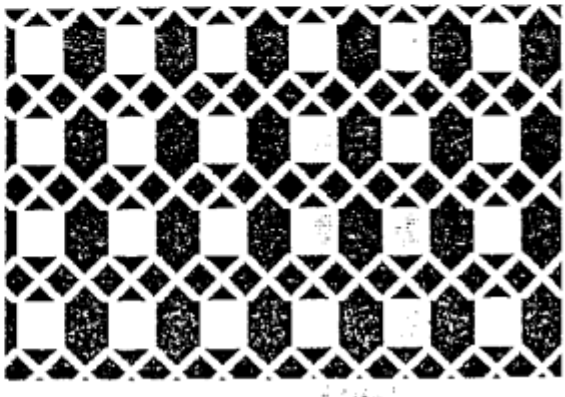
8



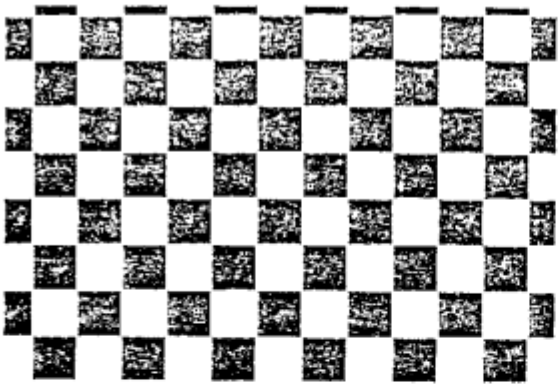
9



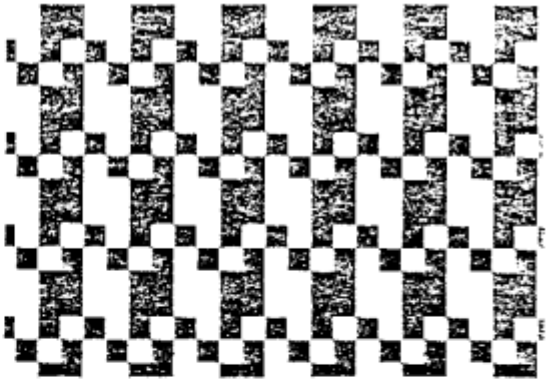
10



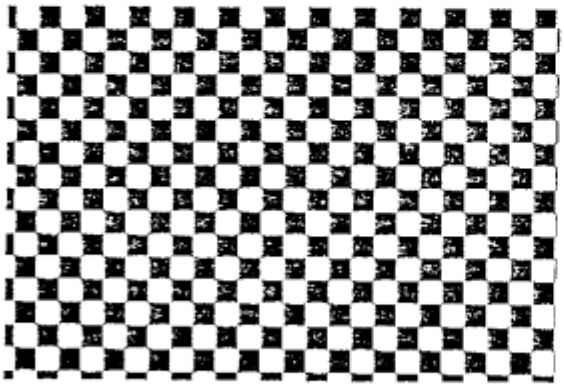
11



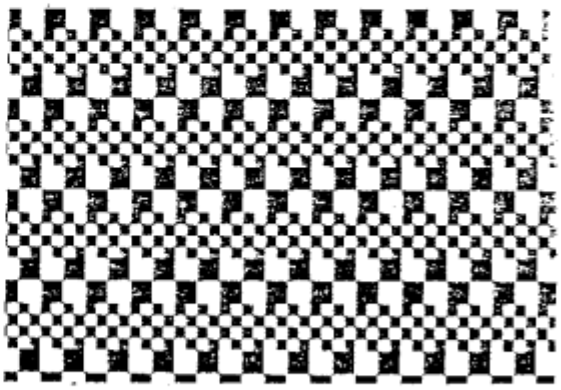
12



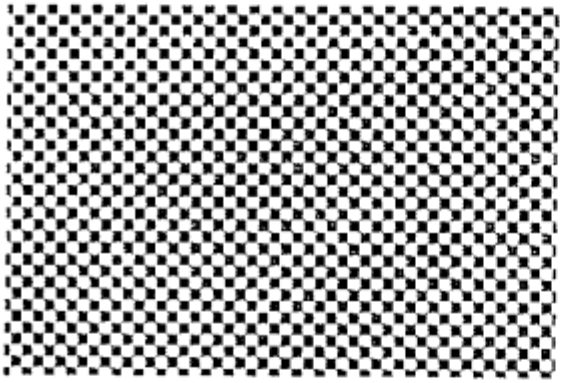
13



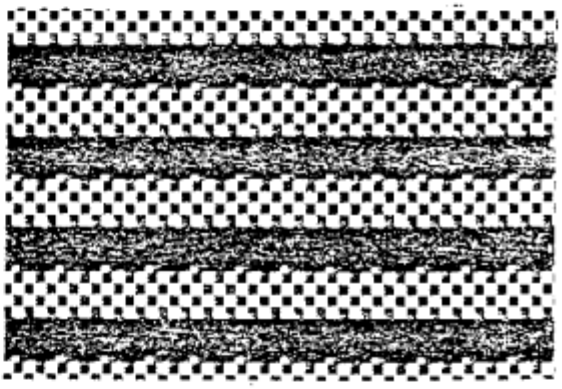
14



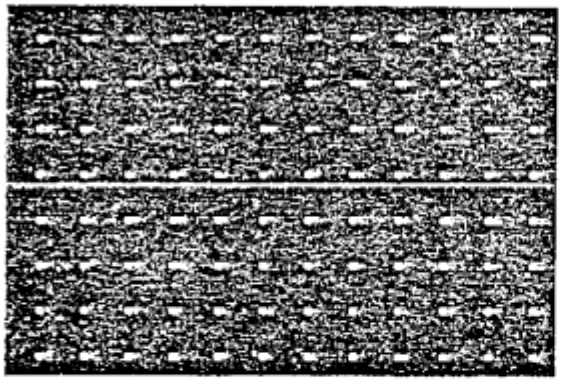
15



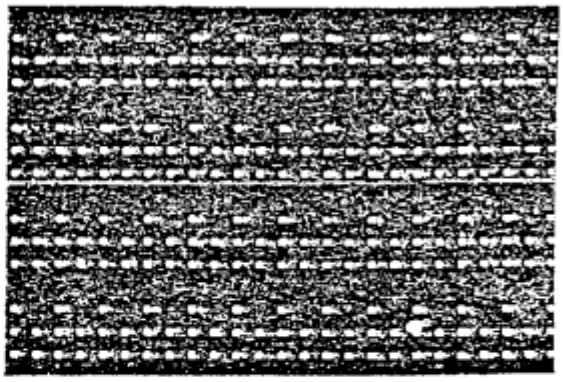
16



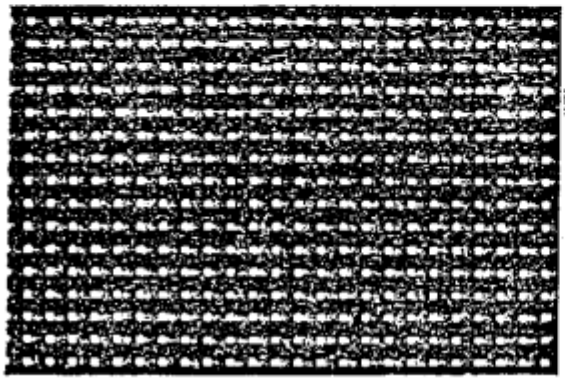
17



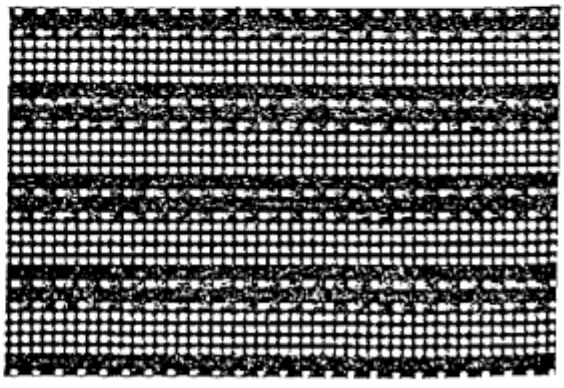
18



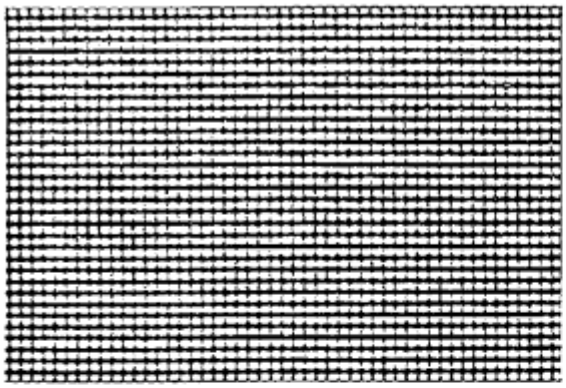
19



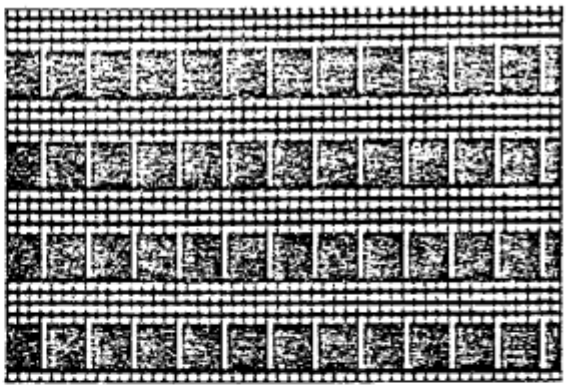
20



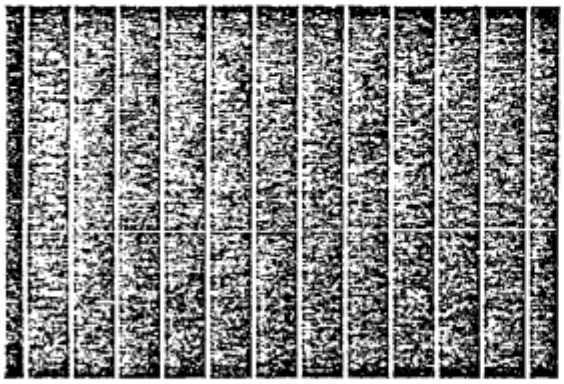
21



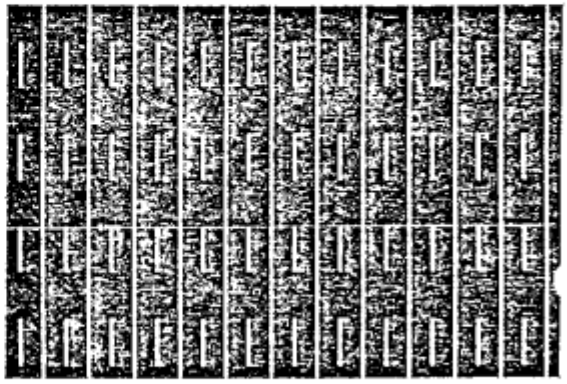
22



23



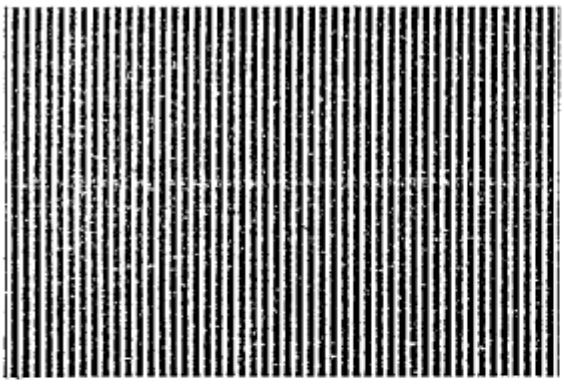
24



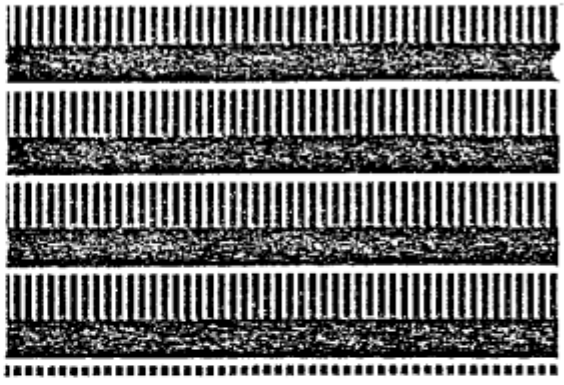
25

26

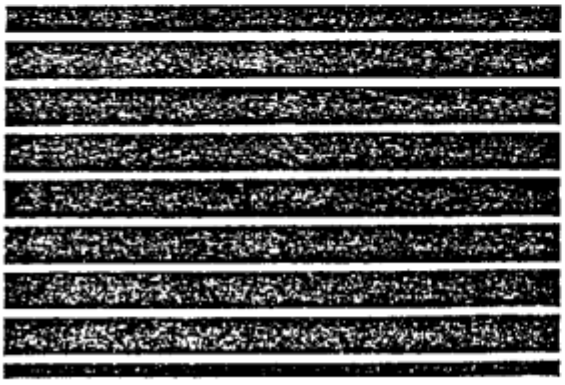
27



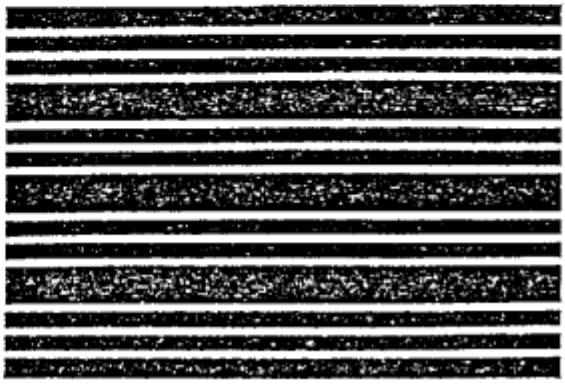
28



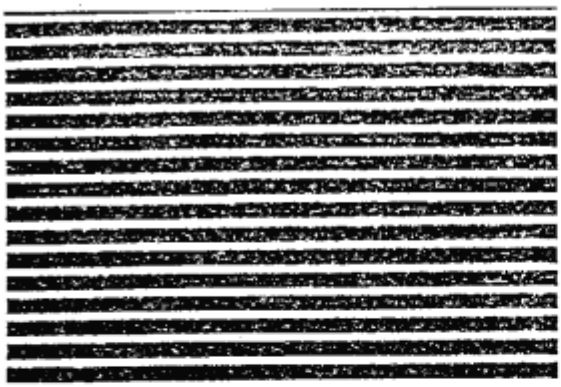
29



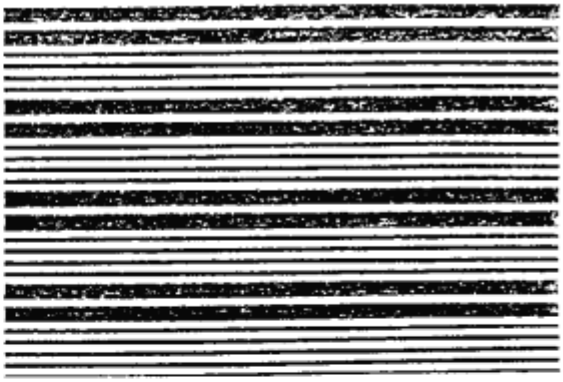
30



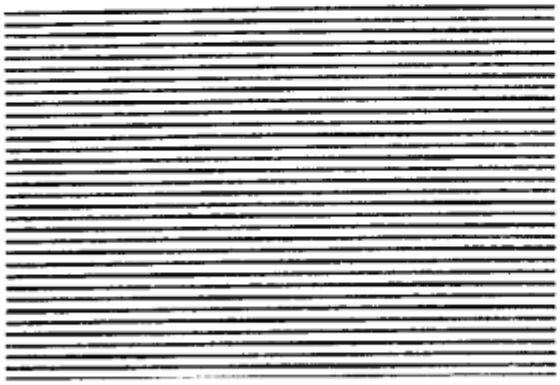
31



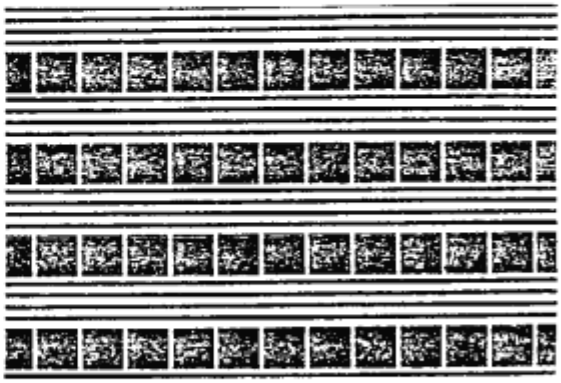
32



33

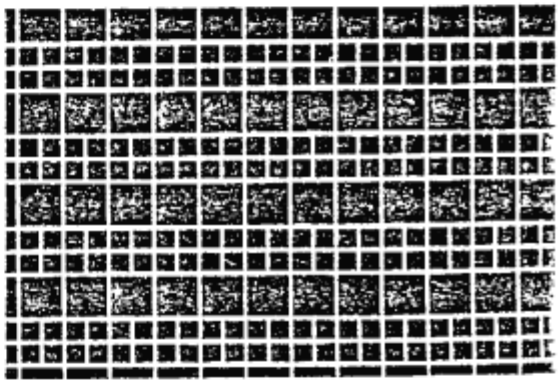


34

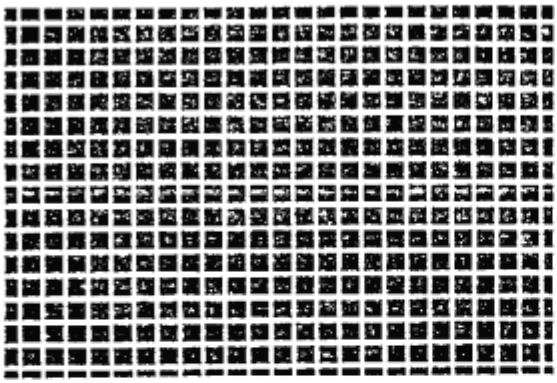


35

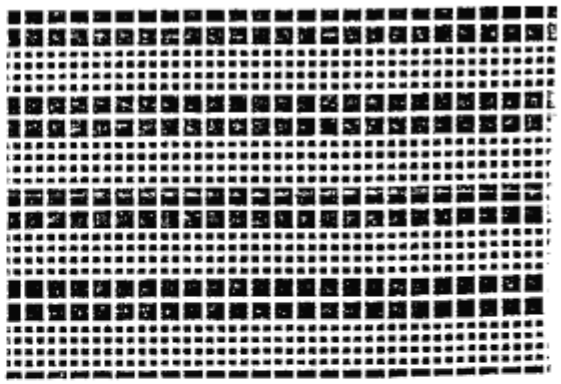
36



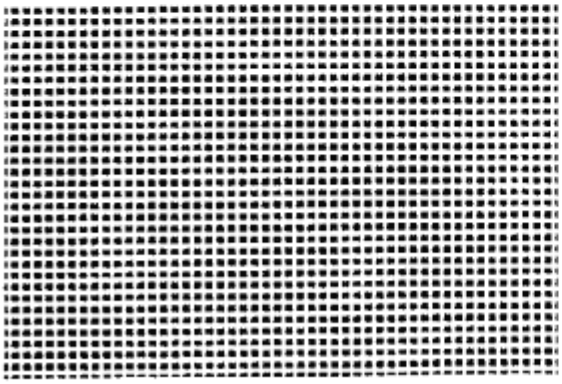
37



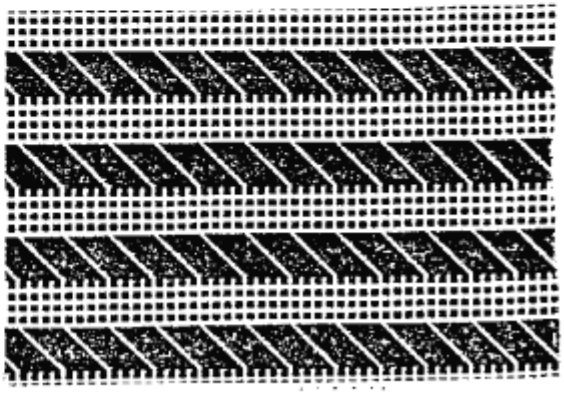
38



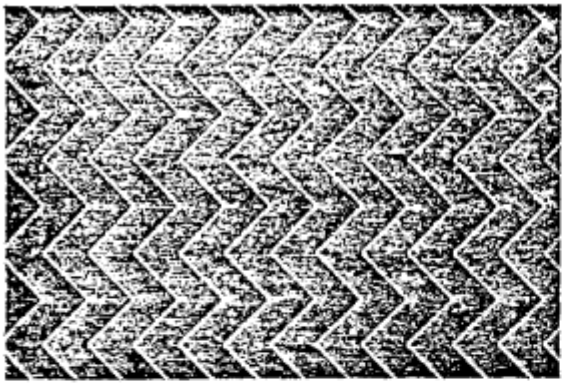
39



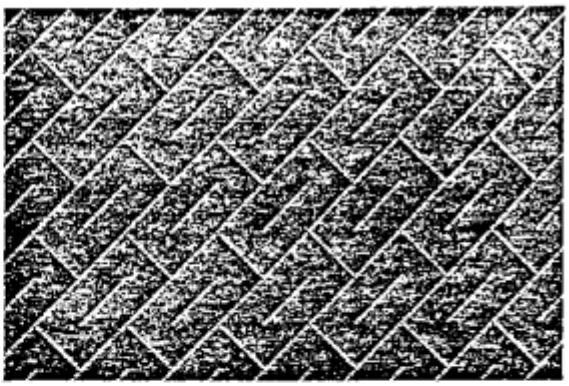
40



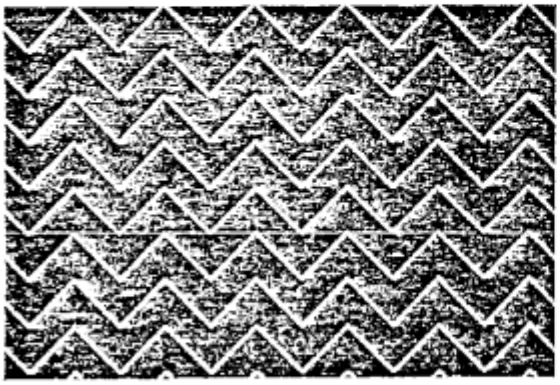
41



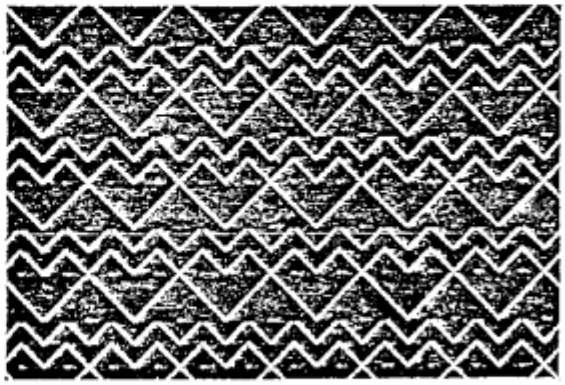
42



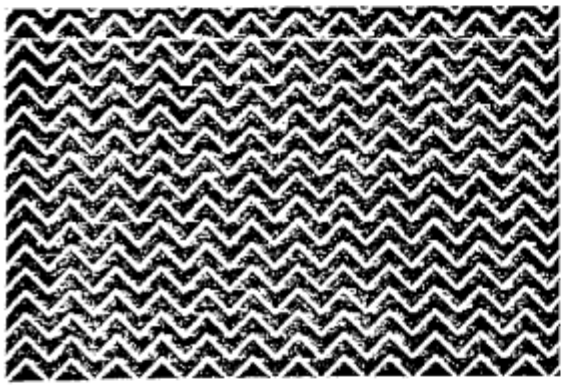
43



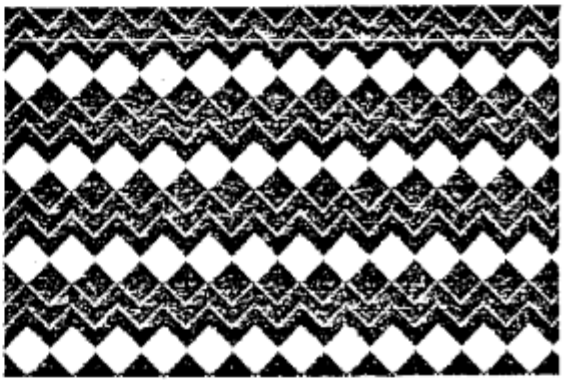
44



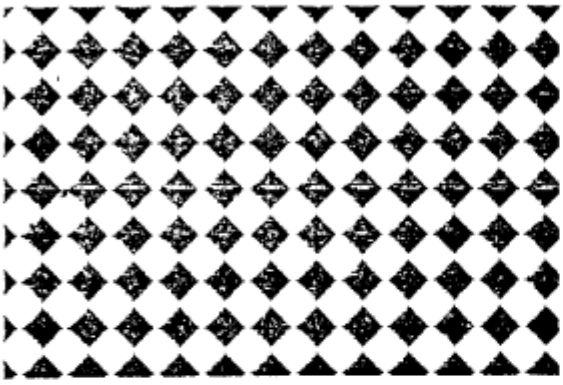
45



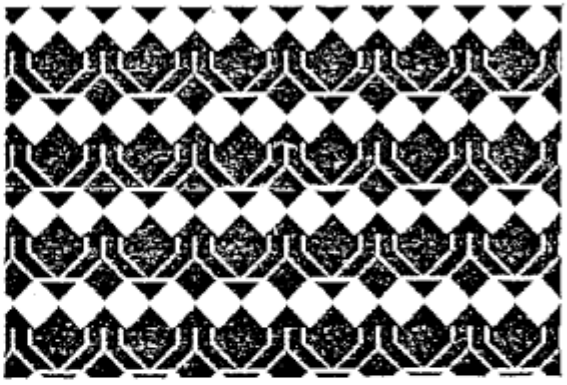
46



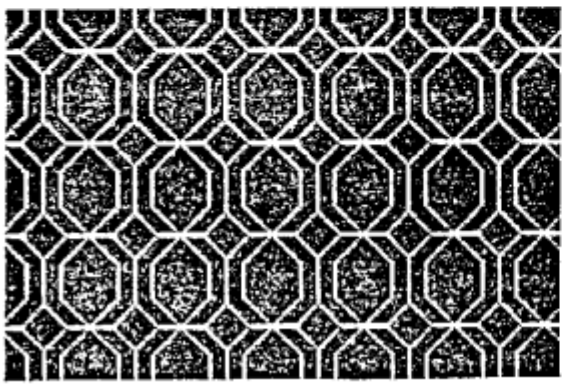
47



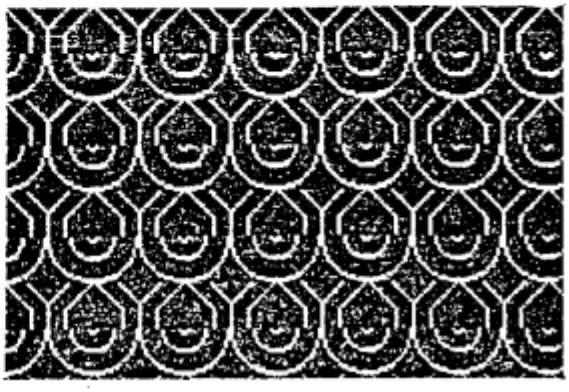
48



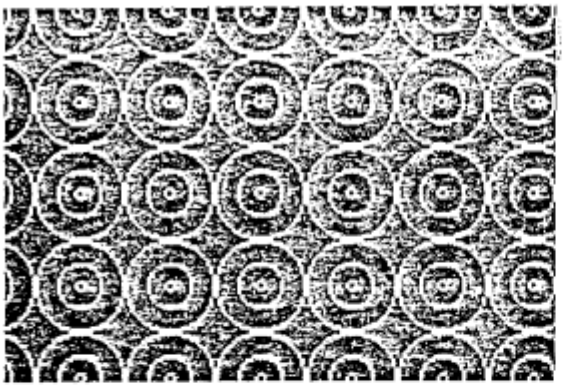
49



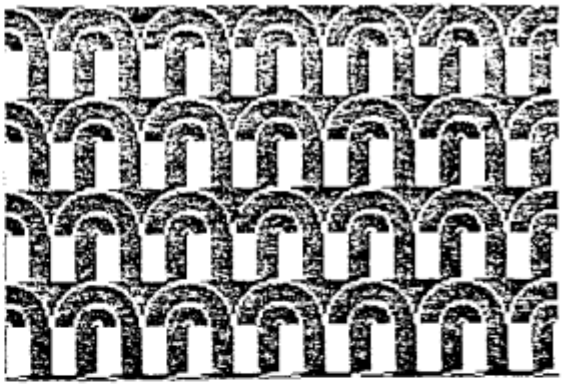
50



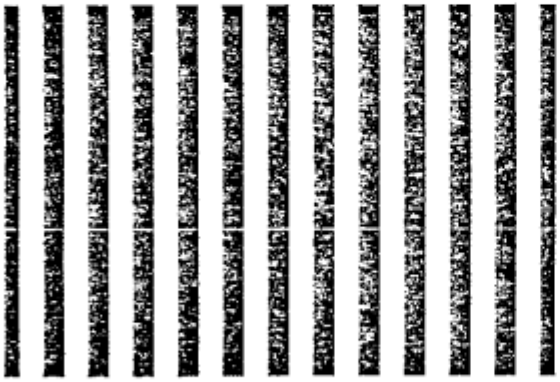
51



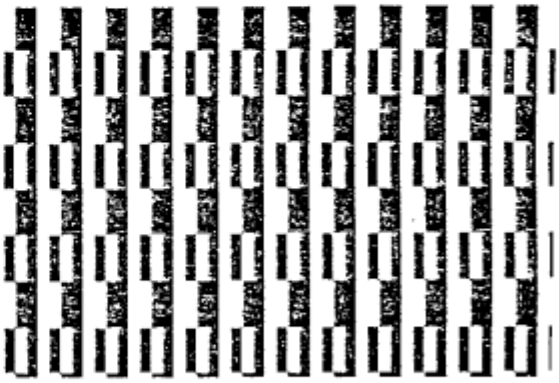
52



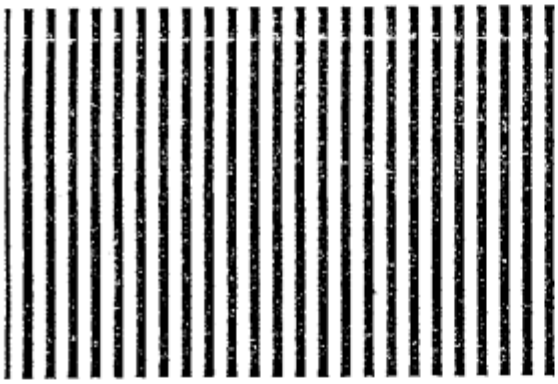
53



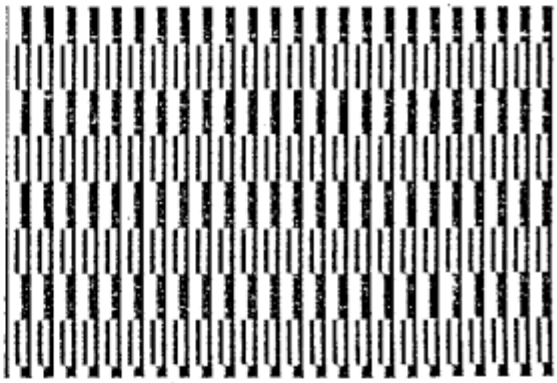
54



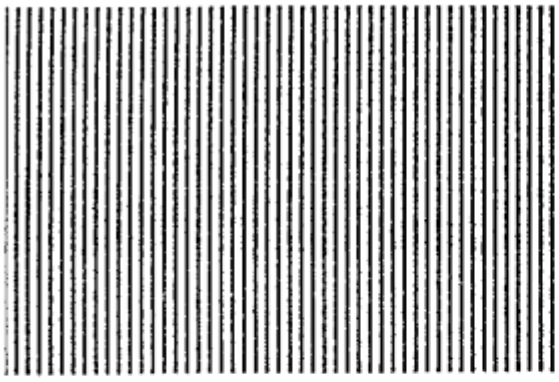
55



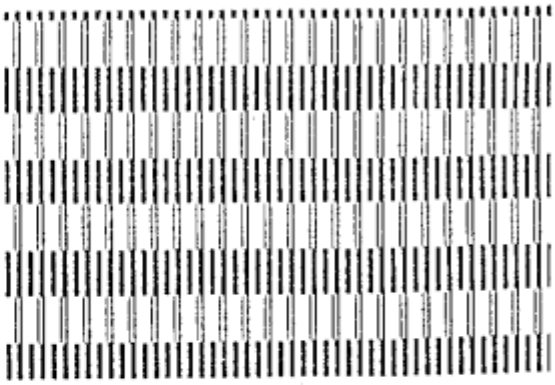
56



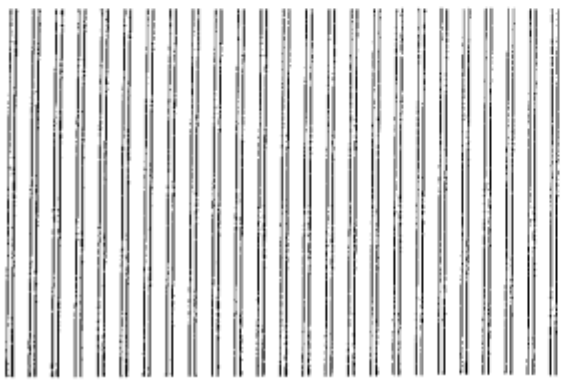
57



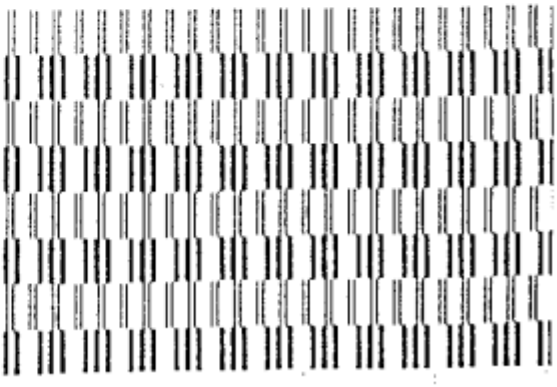
58



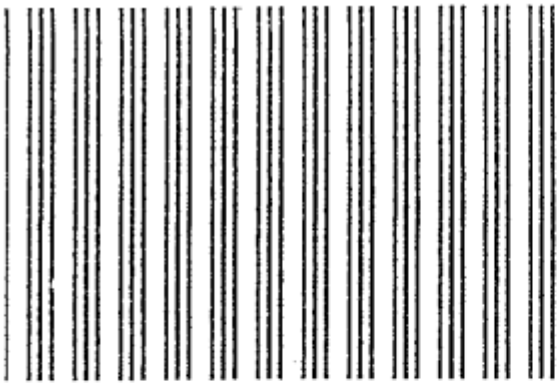
59



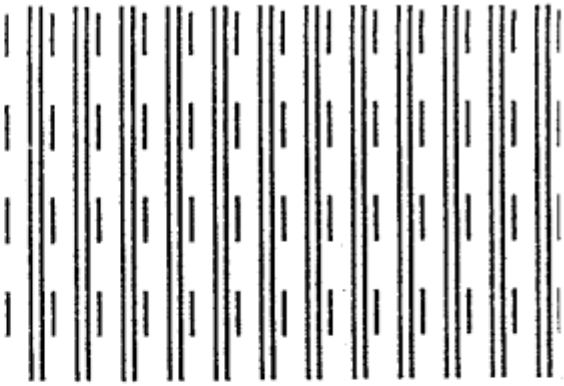
60



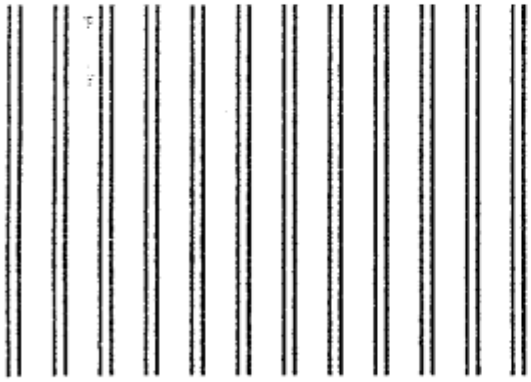
61



62



63



線　種　別

線種は下記の８種類あります。

項番	線　種　名	線　種
1	実　　線	
2	破線(SHORT)	
3	一　点　鎖　線	
4	二　点　鎖　線	
5	破線(MIDLE)	
6	破線(L O N G)	
7	LONG一点鎖線	
8	LONG二点鎖線	

付録２　　'NUMBER.LIB'

本ライブラリーは、 2.1章 d),e)で述べた実数の表現形式のサポート、及び、実数演算のライブラリーである。

1) change_num _char(FUNC,CHAR)　　mode (+,-)又は(-,+)
実数のfunctor 表現(2.1章d))と、実数のatom表現(2.1章c))との相互変換を行う

2) add _real(X,Y,Z)　　mode (+,+,-)
Z=X+Y　: X,Y,Z 実数のfunctor 表現

3) mult_real(X,Y,Z)　　mode (+,+,-)
Z=X*Y　: X,Y,Z 実数のfunctor 表現

4) sub _real(X,Y,Z)　　mode (+,+,-)
Z=X-Y　: X,Y,Z 実数のfunctor 表現

5) dev _real(X,Y,Z)　　mode (+,+,-)
Z=X÷Y : X,Y,Z 実数のfunctor 表現

6) sqrt(X,Y)　　mode (+,-)
Y=√X　: X,Y 実数のfunctor 表現

7) sin _deg (X,Y)　　mode (+,-)
Y=sin(X) : X,Y 実数のfunctor 表現 (但しX の単位は度で与える)

8) cos _deg (X,Y)　　mode (+,-)
Y=cos(X) : X,Y 実数のfunctor 表現 (但しX の単位は度で与える)

付録３　GP-PRO　リスト

```
%-------------------------------------------------------------
%
%               seillac write routine
%
%-------------------------------------------------------------

:-public          abs_bound/2    , write_degree/1, write_matrix/1,
write_vector/1  , write_code/2   , write_cod1/2   , write_scale/1 ,
write_seil/1    , write_code_list/1      , write_ascii_list/1    .

:-mode            write_matrix(+)        ,
abs_bound(+,+)  , write_degree(+)        , write_scale(+)        ,
write_vector(+) , write_code(+,+)        , write_cod1(+,+)       ,
write_seil(+)   , write_code_list(+)     , write_ascii_list(+)   .

write_seil(X) :- tell('tty37:'),write(X),told.

write_code(N,X):-
        tell('tty37:'),
        ( ( X < 0 , ! , Y is 65536+X ,write_cod1(N,Y) ) ;
          ( write_cod1(N,X)                           ) ) ,
        told.

write_cod1(4,X) :- Y is X+48 ,name(Y1,[Y]) ,!,write(Y1).
write_cod1(N,X) :- N1 is N - 4 ,Y is 1 << N1 ,
                P is ( X / Y )+48 ,X1 is X mod Y ,
                name(P1,[P]),write(P1) , ! , write_cod1(N1,X1).

write_code_list([]):-!.
write_code_list([X|Y]):-write_code(16,X),!,write_code_list(Y).

write_ascii_list([]):-!.
write_ascii_list([X|Y]):-name(Z,[X]),write_seil(Z),!,write_ascii_list(Y).

%write_vector-------------------------------------------------
%       write vector data
%               V[X,Y]  is vector then
%                 V = X + ( Y / (1 << 15 ) )
%               where -1 =< V >= 1
%-------------------------------------------------------------
write_vector([0,Y]):-!,
        Z is 1 << 15,abs_bound(Y,Z),write_code(16,Y).
write_vector([X,0]):-!,
        abs_bound(X,2),Y is X << 14,write_code(16,Y).
write_vector(X):-
        integer(X),!,abs_bound(X,2),
        Y is X << 14,write_code(16,Y).
write_vector(X):-
        atom(X),!,change_num_char(real(H,L),X),
        abs_bound(H,2), H1 is H << 14 ,
        L1 is L >> 2 , Z is H1+L1 , write_code(16,Z).
write_vector(real(H,L)):-
        abs_bound(H,2), H1 is H << 14 ,
        L1 is L >> 2 , Z is H1+L1 , write_code(16,Z).
%
```

```
%write_degree-------------------------------------------------------------
%       write write_degreeeree data
%               D[X,Y]  is write_degreeeree then
%                 D = X + ( Y / 16 )
%               where -360 =< D >= 360
%-------------------------------------------------------------------
write_degree([X,Y]):-
        abs_bound(X,361),abs_bound(Y,16),
        Z is ( X << 6 ) + Y ,write_code(16,Z).
write_degree(X):-
        integer(X),!,abs_bound(X,361),
        Z is X * 64 ,write_code(16,Z).
write_degree(X):-
        atom(X),!,change_num_char(real(H,L),X),
        H1 is H << 6,
        L1 is L >> 10 ,Z is H1 + L1 ,write_code(16,Z).
write_degree(real(H,L))-
        abs_bound(H,361),H1 is H << 6,
        L1 is L >> 10 ,Z is H1 + L1 ,write_code(16,Z).

%write_scale--------------------------------------------------------------
%       write write_scale data
%               S[X,Y]  is write_scale then
%                 S = X + ( Y / 256 )
%               where 0 < D < 256
%-------------------------------------------------------------------
write_scale([X,Y]):-
        abs_bound(X,256),abs_bound(Y,256),
        Z is X * 256 + Y,write_code(16,Z).
write_scale(X):-
        integer(X),abs_bound(X,256),
        Z is X * 256,write_code(16,Z).
write_scale(X):-
        atom(X),!,change_num_char(real(H,L),X),
        abs_bound(H,256),H1 is H << 8,
        L1 is L >> 8 ,Z is H1 + L1 ,write_code(16,Z).
write_scale(real(H,L)):-
        abs_bound(H,256),H1 is H << 8,
        L1 is L >> 8 ,Z is H1 + L1 ,write_code(16,Z).
%write_matrix-------------------------------------------------------------
%       write matrix  element data
%
%-------------------------------------------------------------------
write_matrix([H,L]):-
        write_code(16,H),write_code(16,L).
write_matrix(X):-
        atom(X),!,change_num_char(real(H,L),X),
        write_code(16,H),write_code(16,L).
write_matrix(X):-
        integer(X),!,write_code(16,X),write_code(16,0).
write_matrix(real(H,L)):-
        write_code(16,H),write_code(16,L).

abs_bound(X,Y):-Z is 0-Y, X>Z,X<Y.
%
```

```
%-----------------------------------------------------------------
%
%               seillac read routine
%
%-----------------------------------------------------------------

:- public
read_seil/1     , read_code/2    , read_cod1/3    , read_CR/0     ,
read_string1/2  , read_co_list_2/2       , read_co_list_3/2      .

:-mode
read_seil(-)    , read_string1(+,-)      , read_co_list_2(+,-)   ,
read_code(+,-)  , read_cod1(+,+,-)       , read_co_list_3(+,-)   .

read_seil(X):-see('tty37:'),get0(X),seen.

read_code(N,X):-see('tty37:'),read_cod1(N,0,X),seen.

read_cod1(1,X,Y):-get0(U),Y is (X << 4)+U-48.
read_cod1(N,X,Y):-
        get0(U),X1 is (X << 4)+U-48,
        N1 is N-1,!,read_cod1(N1,X1,Y).
read_CR:-see('tty37:'),get0(U),U = 31,seen.

read_string1(0,[]):-!.
read_string1(N,[X|Y]):-get0(X),N1 is N-1,!,read_string(N1,Y).
read_string(N,X):-see('tty37:'),read_string1(N,Y),seen.

read_co_list_2(N,[X,Y|U]):-
        read_code(4,X),read_code(4,Y),!,
        N1 is N-1, read_co_list(N1,U).
read_co_list_3(N,[X,Y,Z|U]):-
        read_code(4,X),read_code(4,Y),read_code(4,Z),!,
        N1 is N-1, read_co_list(N1,U).

%
```

```
/*      1. DISPLAY COMMAND      */

:-public  disp_mode/1, disp_on/0, erase/0, through/1,
          cursor_on/1, console_clear/0.

:-mode    disp_mode(+), through(+),cursor_on(+).

/*   2  */
disp_mode(X):-
        ( ( X=2 , ! , write_seil('$MG2') ) ;
          ( X=3 , ! , write_seil('$MG3') ) ;
          ( X=c , ! , write_seil('$MC')  ) ) .
/*   3  */
disp_on:-write_seil('$N').
/*   4  */
erase :- write_seil('$E').
/*   5  */
through(X):-write_seil('$T'),write_seil(X).
/*   5.1  */
cursor_on(I):-write_seil('$CU'),write_seil(I).
/*   5.2  */
consol_clear(I):-write_seil('$CL').
%
```

```
/*      2. BASIC OUTPUT COMMAND          */
:-public            rectang/4      , cubic/6        , grid/6         ,
move_abs_2/2     , move_abs_3/3  , move_rel_2/2  , move_rel_3/3  ,
line_abs_2/2     , line_abs_3/3  , line_rel_2/2  , line_rel_3/3  ,
circle_2/1       , circle_3/4    , ellipse_2/2   , ellipse_3/5   ,
arc_2/3          , arc_3/6       , fan_2/3       , fan_3/6       ,
polyl_abs_2/1    , polyl_abs_3/1 , polyl_rel_2/1 , polyl_rel_3/1 ,
polyg_abs_2/1    , polyg_abs_3/1 , polyg_rel_2/1 , polyg_rel_3/1 ,
mark_abs_2/2     , mark_abs_3/3  , mark_rel_2/2  , mark_rel_3/3  ,
polym_abs_2/1    , polym_abs_3/1 , polym_rel_2/1 , polym_rel_3/1 ,
text_ent_char/1                  , text_ent_sym/1                .

:- mode                    grid(+,+,+,+,+,+)     ,
move_abs_2(+,+)          , move_abs_3(+,+,+)     , move_rel_2(+,+)        ,
move_rel_3(+,+,+)        , line_abs_2(+,+)       , line_abs_3(+,+,+)      ,
line_rel_2(+,+)          , line_rel_3(+,+,+)     , circle_2(+)            ,
circle_3(+,+,+,+)        , ellipse_2(+,+)        , ellipse_3(+,+,+,+,+)   ,
arc_2(+,+,+)             , arc_3(+,+,+,+,+,+)    , fan_2(+,+,+)           ,
fan_3(+,+,+,+,+,+)       , polyl_abs_2(+)        , polyl_abs_3(+)         ,
polyl_rel_2(+)           , polyl_rel_3(+)        , polyg_abs_2(+)         ,
polyg_abs_3(+)           , polyg_rel_2(+)        , polyg_rel_3(+)         ,
mark_abs_2(+,+)          , mark_abs_3(+,+,+)     , mark_rel_2(+,+)        ,
mark_rel_3(+,+,+)        , polym_abs_2(+)        , polym_abs_3(+)         ,
polym_rel_2(+)           , polym_rel_3(+)        , text_ent_char(+)       ,
rectang(+,+,+,+)         , cubic(+,+,+,+,+,+)    , text_ent_sym(+)        .

/*   6  */
move_abs_2(X,Y):-
        write_seil('$AA2'),write_code(16,X),write_code(16,Y).
move_abs_3(X,Y,Z):-
        write_seil('$AA3'),write_code(16,X),
                write_code(16,Y),write_code(16,Z).
/*   7  */
move_rel_2(X,Y):-
        write_seil('$AR2'),write_code(16,X),write_code(16,Y).
move_rel_3(X,Y,Z):-
        write_seil('$AR3'),write_code(16,X),
                write_code(16,Y),write_code(16,Z).
/*   8  */
line_abs_2(X,Y):-
        write_seil('$LA2'),write_code(16,X),write_code(16,Y).
line_abs_3(X,Y,Z):-
        write_seil('$LA3'),write_code(16,X),
                write_code(16,Y),write_code(16,Z).
/*   9  */
line_rel_2(X,Y):-
        write_seil('$LR2'),write_code(16,X),write_code(16,Y).
line_rel_3(X,Y,Z):-
        write_seil('$LR3'),write_code(16,X),
                write_code(16,Y),write_code(16,Z).
/*  10  */
grid(X,Y,Xpc,Xp,Ypc,Yp):-
        write_seil('$G2'),
        write_code(16,X),write_code(16,Y),write_code(16,Xpc),
        write_code(16,Xp),write_code(16,Ypc),write_code(16,Yp).
%
```

```
/*  11  */
polyl_abs_2(List):-
        length(List,N1),N is N1 / 2,
        write_seil('$PA2'),write_code(16,N),write_code_list(List).
polyl_abs_3(List):-
        length(List,N1),N is N1 / 3,
        write_seil('$PA3'),write_code(16,N),write_code_list(List).
/*  12  */
polyl_rel_2(X):-
        length(X,N1),N is N1 / 2 ,
        write_seil('$PR2'),write_code(16,N),write_code_list(X).
polyl_rel_3(X):-
        length(X,N1),N is N1 / 3 ,
        write_seil('$PR3'),write_code(16,N),write_code_list(X).
/*  13  */
polyg_abs_2(X):-
        length(X,N1),N is N1 / 2 ,
        write_seil('$OA2'),write_code(16,N),write_code_list(X).
polyg_abs_3(X):-
        length(X,N1),N is N1 / 3 ,
        write_seil('$OA3'),write_code(16,N),write_code_list(X).
/*  14  */
polyg_rel_2(X):-
        length(X,N1),N is N1 / 2 ,
        write_seil('$OR2'),write_code(16,N),write_code_list(X).
polyg_rel_3(X):-
        length(X,N1),N is N1 / 3 ,
        write_seil('$OR3'),write_code(16,N),write_code_list(X).
/*  15  */
rectang(X1,Y1,X2,Y2):-
        write_seil('$RT'),write_code(16,X1),write_code(16,Y1),
                          write_code(16,X2),write_code(16,Y2).
/*  16  */
cubic(X1,Y1,Z1,X2,Y2,Z2):-
        write_seil('$CB'),
        write_code(16,X1),write_code(16,Y1),write_code(16,Z1),
        write_code(16,X2),write_code(16,Y2),write_code(16,Z2).
/*  17  */
circle_2(X) :- write_seil('$C2'),write_code(16,X).
circle_3(R,X,Y,Z):-
        write_seil('$C3'),write_code(16,R),
        write_degree(X),write_degree(Y),write_degree(Z).
/*  18  */
ellipse_2(A,B):-
        write_seil('$L2'),write_code(16,A),write_code(16,B).
ellipse_3(A,B,X,Y,Z):-
        write_seil('$L3'),write_code(16,A),write_code(16,B),
        write_degree(X),write_degree(Y),write_degree(Z).
/*  19.1  */
arc_2(R,01,02):-
        write_seil('$F2A'),write_code(16,R),
        write_degree(01),write_degree(02).
arc_3(R,01,02,X,Y,Z):-
        write_seil('$F3A'),write_code(16,R),write_degree(01),
        write_degree(02),write_degree(X),write_degree(Y),write_degree(Z).
%
```

```
/*  19.2  */
fan_2(R,01,02):-
        write_seil('$F2F'),write_code(16,R),
                write_degree(01),write_degree(02).
fan_3(R,01,02,X,Y,Z):-
        write_seil('$F3F'),write_code(16,R),write_degree(01),
        write_degree(02),write_degree(X),write_degree(Y),write_degree(Z).
/*  20  */
mark_abs_2(X,Y):-
        write_seil('$KA2'),write_code(16,X),write_code(16,Y).
mark_abs_3(X,Y,Z):-
        write_seil('$KA3'),write_code(16,X),write_code(16,Y),write_code(16,Z).
/*  21  */
mark_rel_2(X,Y):-
        write_seil('$KR2'),write_code(16,X),write_code(16,Y).
mark_rel_3(X,Y,Z):-
        write_seil('$KR3'),write_code(16,X),write_code(16,Y),write_code(16,Z).
/*  22  */
polym_abs_2(List):-length(List,N1),N is N1/2,
        write_seil('$YA2'),write_code(16,N),write_code_list(List).
polym_abs_3(List):-length(List,N1),N is N1/3,
        write_seil('$YA3'),write_code(16,N),write_code_list(List).
/*  23  */
polym_rel_2(List):-length(List,N1),N is N1/2,
        write_seil('$YR2'),write_code(16,N),write_code_list(List).
polym_rel_3(List):-length(List,N1),N is N1/3,
        write_seil('$YR3'),write_code(16,N),write_code_list(List).
/*  24  */
text_ent_char(Asc):-
        write_seil('$TA'),
        ( ( atomic(Asc),!,name(Asc,X) ) ;
          X = Asc                        ),
        length(X,N),write_code(16,N),write_ascii_list(X).
text_ent_sym(Symbol):-
        write_seil('$TS1'),write_code(16,S).
/*  24.1  */
consol_char_pos(X,Y):-
        write_seile('$CP'),write_code(16,X),weite_code(16,Y).
/*  24.2  */
cursol_pos(X,Y):-
        write_seil('$CP'),write_code(16,X),weite_code(16,Y).
%
```

```
/*      3. CLASS/SEGMENT COMMAND        */
:-public
seg_class_def/2 , del_class/1    , ren_class/2    , del_seg/1      ,
del_class_all/0 , seg_def/1      , edit_seg/1     , call_seg/1     ,
del_seg_all/0   , ren_seg/2      , seg_term/0     , id_def/2       ,
del_class_mem/2 , del_id/1       , temp_seg/1     , app_end/0      ,
app_class_mem/2 , app_id/1       , ren_id/2       ,
repl_text_data/2                 , mult_repl_text_data/1          .

:-mode
seg_class_def(+,+)       , del_class(+)   , ren_class(+,+)         ,
del_seg(+)               , seg_def(+)     , edit_seg(+)            ,
call_seg(+)              , ren_seg(+,+)   , id_def(+,+)            ,
del_class_mem(+,+)       , del_id(+)      , temp_seg(+)            ,
app_class_mem(+,+)       , app_id(+)      , ren_id(+,+)            ,
repl_text_data(+)        , mult_repl_text_data(+)                  ,
mult_repl_text_data1(+) .
/*  25  */
seg_class_def(Class,List):-
        length(List,N),write_seil('$CF'),
        write_code(8,Class),write_code(16,N),write_code_list(List).
/*  26  */
del_class(Class):-
        write_seil('$CDN'),write_code(8,Class).
/*  27  */
del_class_all:-eac,write('$CDA').
/*  28  */
ren_class(Old,New):-
        write_seil('$CR'),write_code(8,Old),write_code(8,New).
/*  29  */
seg_def(X) :-
        write_seil('$SF'),write_code(16,X).
/*  30  */
edit_seg(X):-
        write_seil('$SE'),write_code(16,X).
/*  31  */
call_seg(X):-
        write_seil('$SC'),write_code(16,X).
/*  32  */
del_seg(Int):-
        write_seil('$SDN'),write_code(16,Int).
/*  33  */
del_seg_all:-write_seil('$SDA').
/*  34  */
ren_seg(Old,New):-
        write_seil('$SR'),write_code(16,Old),write_code(16,New).
/*  35  */
seg_term :- write_seil('$ST').
/*  36  */
id_def(High,Id_num) :-
        write_seil('$IF'),write_seil(High),write_code(16,Id_num).
id_def0 :- write_seil('$IF00001').
%
```

```
/*  37  */
ren_id(Old,New):-
        write_seil('$IR'),write_code(16,Old),write_code(16,New).
/*  38  */
del_id(Int):-
        write_seil('$ID'),write_code(16,Int).
/*  39  */
temp_seg(X):-
        write_seil('$SP'),write_seil(X).
/*  40  */
app_class_mem(Class,Seg):-
        write_seil('$CMA'),write_code(8,Class),write_code(16,Seg).
/*  41  */
del_class_mem(Class,Seg):-
        write_seil('$CMD'),write_code(8,Class),write_code(16,Seg).
/*  42  */
app_id(Id_number):-
        write_seil('$IA'),write_code(16,Id_number).
/*  43  */
app_end :- write_seil('$IE').
/*  44  */
repl_text_data(Id_num):-write_seil('$IT'),write_code(16,Id_num).
/*  45  */
mult_repl_text_data(List):-
        write_seil('$SM'),length(N,List),
        mult_repl_text_data1(List).
mult_repl_text_data1([[Seg,Id,Data_typ,Data]|Y]):-
        write_code(16,Seg),write_code(16,ID),
        ( ( Data_typ = s ,!, write_seil('S1'),write_code(16,Data) );
          ( data_typ = a ,!,
            ( ( atomic(Data),!, name(X,Data) ) ;
              X = Data                          ),
            length(N,Data),write_code(16,N),write_ascii_list(X) ) ).
%
```

```
/*      4.attributed command    */

:- public
set_cursor_typ/1, col_typ_1/3   , col_typ_2/6   , line_typ/1    ,
fill_attr/2     , fill_pat/1    , back_col/1    , fill_mode/1   ,
dyn_attr_typ_1/2        , dyn_attr_typ_2/2      , char_typ/1    ,
dyn_attr_typ_3/2        , char_prec_attr/1      , char_attr/5   ,
fill_col_typ_1/3        , fill_col_typ_2/6      , zoom_trans/2  ,
char_co_attr_2/2        , char_co_attr_3/9      , stat_attr/1   ,
model_trans_3/10        , model_trans_2/6       , seg_man/2     ,
class_dyn_attr_typ_1/2  , id_dyn_attr_typ/3     , set_prio/2    ,
class_dyn_attr_typ_2/2  , class_stat_attr/2     , wind_trans/4  ,
class_dyn_attr_typ_3/2  , look_point_trans/7    , mark_typ/1    .

:-mode                  set_cursor_typ(+)       ,
col_typ_1(+,+,+)        , line_typ(+)           , fill_mode(+)          ,
col_typ_2(+,+,+,+,+,+,+)  , fill_attr(+,+)      , char_co_attr_2(+,+)   ,
char_attr(+,+,+,+,+)    , zoom_trans(+,+)       , stat_attr(+)          ,
fill_col_typ_1(+,+,+)   , set_prio(+,+)         , dyn_attr_typ_1(+,+)   ,
id_dyn_attr_typ(+,+,+)  , dyn_attr_typ_2(+,+)   , char_prec_attr(+)     ,
fill_col_typ_2(+,+,+,+,+,+)     , back_col(+)   , wind_trans(+,+,+,+)   ,
look_point_trans(+,+,+,+,+,+,+) , fill_pat(+)   , dyn_attr_typ_3(+,+)   ,
model_trans_2(+,+,+,+,+,+)      , seg_man(+,+)  , mark_typ(+)           ,
class_dyn_attr_typ_1(+,+)       , char_typ(+)   , class_stat_attr(+,+)  ,
class_dyn_attr_typ_2(+,+)       , char_co_attr_3(+,+,+,+,+,+,+,+,+)     ,
class_dyn_attr_typ_3(+,+)       , model_trans_3(+,+,+,+,+,+,+,+,+,+)    .
/*  ??  */
set_cursor_typ(I):-
        write_seil('$CT') ,write_seil(I).
/* 46  */
col_typ_1(A,B,C):-
        write_seil('$cc') , write_seil(A),
        write_seil(B) ,     write_seil(C).
/* 47 */
col_typ_2(A1,B1,C1,A2,B2,C2):-
        write_seil('$cx') , write_seil(A1),
        write_seil(B1) ,    write_seil(C1),
        write_seil(A2),     write_seil(B2) ,    write_seil(C2).
/* 48 */
line_typ(A):-
        write_seil('$l'),write_seil(A).
/* 48.1 */
fill_mode(I):-
        write_seil('$RM'),write_seil(I).
/* 49 */
fill_attr(I,J):-
        X is (I mod 2) * 8 + (J mod 2 ),
        write_seil('$RF'),write_seil(X).
/* 50 */
fill_col_typ_1(A,B,C):-
        write_seil('$RC') , write_seil(A),
        write_seil(B) ,     write_seil(C).
/* 51 */
fill_col_typ_2(A1,B1,C1,A2,B2,C2):-
        write_seil('$RD') , write_seil(A1),
        write_seil(B1) ,    write_seil(C1),
        write_seil(A2),     write_seil(B2) ,    write_seil(C2).
/* 52 */
fill_pat(X):-
        write_seil('$RP'),write_code(16,X).
/* 54 */
back_col(A):-
         write_seil('$b'), write_seil(A).
```

```
/* 55 */                    /* ----------  Developing  ----------*/
char_typ(A):-
        write_seil('$ht'), write_seil(A).
/* 56 */
char_prec_attr(A):-
        write_seil('$hp'), write_seil(A).
/* 57 */
char_attr(P,J,S,X,Y):-
        write_seil('$ht'), write_seil(P),
        write_code(4,J),write_code(16,S),
        write_scale(X),write_scale(Y).
/* 58 */
char_co_attr_2(X,Y):-
        write_seil('$h2'),write_vector(X),write_vector(Y).
/* 59 */
char_co_attr_3(Px,Py,Pz,Ux,Uy,Uz,Vx,Vy,Vz):-
        write_seil('$h3'),
        write_vector(Px), write_vector(Py), write_vector(Pz),
        write_vector(Ux), write_vector(Uy), write_vector(Uz),
        write_vector(Vx), write_vector(Vy), write_vector(Vz).
/* 60 */
class_stat_attr(B,C):-
        write_seil('$as'),write_seil(B),write_code(8,C).
/* 61 */
class_dyn_attr_typ_1(I,C):-
        write_seil('$av'),write_seil(I),write_code(8,C).
/* 62 */
class_dyn_attr_typ_2(I,C):-
        write_seil('$ah'),write_seil(I), write_code(8,C).
/* 63 */
class_dyn_attr_typ_3(I,C):-
        write_seil('$ad'),write_seil(I), write_code(8,C).

/* 64 */
wind_trans(C,B,Dx,Dy):-
        write_seil('$w'), write_code(8,C),
        write_degree(B), write_code(16,Dx), write_code(16,Dy).
/* 65 */
look_point_trans(C,Rx,Ry,Rz,Tx,Ty,Tz):-
        write_seil('$o'), write_code(8,C),
        write_degree(Rx), write_degree(Ry), write_degree(Rz),
        write_code(16,Tx), write_code(16,Ty), write_code(16,Tz).
/* 66 */
zoom_trans(C,S):-
        write_seil('$z'), write_code(8,C), write_scale(S).
/* 67 */
model_trans_2(A,Sx,Sy,R,Tx,Ty):-
        write_seil('$m2'),
        write_code(16,A), write_scale(Sx), write_scale(Sy),
        write_degree(R), write_code(16,Tx), write_code(16,Ty).
/* 67.1 */
stat_attr(I):-write_seil('$s'),write_seil(I).
%
```

```
/* 68 */
model_trans_3(A,Sx,Sy,Sz,Rx,Ry,Rz,Tx,Ty,Tz):-
        write_seil('$m3'), write_code(16,A),
        write_scale(Sx), write_scale(Sy), write_scale(Sz),
        write_degree(Rx), write_degree(Ry), write_degree(Rz),
        write_code(16,Tx), write_code(16,Ty), write_code(16,Tz).
/* 68.1 */
dyn_attr_typ_1(I,S):-
        write_seil('$dv'), write_seil(I), write_code(16,S).
/* 69 */
dyn_attr_typ_2(I,S):-
        write_seil('$dh'), write_seil(I), write_code(16,S).
/* 70 */
dyn_attr_typ_3(I,S):-
        write_seil('$dd'), write_seil(I), write_code(16,S).
/* 71 */
set_prio(S,P):-
        write_seil('$p'), write_code(16,S) ,write_code(16,R).
/* 72 */                /* ----------  Developing  ----------*/
seg_man(N,List):-
        write_seil('$g'), write_code(16,N) ,seg_man1(List).
seg_man1([[S,V,H,A,B,C,U]|List]):-
        write_code(16,S) , write_seil(V), write_seil(H),
        write_seil(A), write_seil(B), write_seil(C),
        write_code(16,U), ! ,seg_man1(List).
seg_man1([]):-!.
/* 73 */
id_dyn_attr_typ(C,A,B):-
        write_seil('$i'), write_seil(C), write_code(16,A) ,write_code(16,B).
mark_typ(I):- write_seil('$k'), write_seil(I).

%
```

```
/*      5. viwing command                */

:-public          front_back_dist/2              ,
view_mat_2/2    , view_mat_3/2   , window/4      , view_2/4      ,
view_3/6        , init_view/1    , sel_view/1    , dev_vew/6     .

:-mode
view_mat_2(+,+) , view_mat_3(+,+)       , window(+,+,+,+)       ,
view_2(+,+,+,+) , view_3(+,+,+,+,+,+)   , init_view(+)          ,
sel_view(+)     , dev_vew(+,+,+,+,+,+)  , front_back_dist(+,+)  .

/* 75 */
view_mat_2(p,List):-
          write_seil('$VM2P'),view_mat_1(List).
view_mat_2(v,List):-
          write_seil('$VM2V'),view_mat_1(List).
view_mat_2(n,List):-
          write_seil('$VM2N'),view_mat_1(List).
view_mat_1([X|List]):-write_matrix(X),!,view_mat_1(List).
view_mat_1([]):-!.
/* 76 */
view_mat_3(c,List):-
          write_seil('$VM3C'),view_mat_1(List).
view_mat_3(s,List):-
          write_seil('$VM3S'),view_mat_1(List).
view_mat_3(r,List):-
          write_seil('$VM3R'),view_mat_1(List).
view_mat_3(n,List):-
          write_seil('$VM3N'),view_mat_1(List).
view_mat_3(p,List):-
          write_seil('$VM3P'),view_mat_1(List).
/* 77 */
window(Xmin,Xmax,Ymin,Ymax):-
         write_seil('$VW'),
         write_code(16,Xmin), write_code(16,Xmax),
         write_code(16,Ymin), write_code(16,Ymax).
/* 78 */
view_2(Xmin,Xmax,Ymin,Ymax):-
         write_seil('$VP2'),
         write_code(16,Xmin), write_code(16,Xmax),
         write_code(16,Ymin), write_code(16,Ymax).
/* 79 */
view_3(Xmin,Xmax,Ymin,Ymax,Zmin,Zmax):-
         write_seil('$VP3'),
         write_code(16,Xmin), write_code(16,Xmax),
         write_code(16,Ymin), write_code(16,Ymax),
         write_code(16,Zmin), write_code(16,Zmax).
/* 80 */
front_back_dist(F,B):-
         write_seil('$VD'), write_code(16,F), write_code(16,B).
/* 81 */
init_view(I):- write_seil('$VI'), write_seil(I).
/* 82 */
sel_view(I):- write_seil('$VS'), write_seil(I).
/* 83 */
dev_vew(I,J,Xmin,Xmax,Ymin,Ymax):-
         write_seil('$VT'),
         write_code(8,I),write_code(8,J),
         write_code(16,Xmin), write_code(16,Xmax),
         write_code(16,Ymin), write_code(16,Ymax).
%
```

```
/*      6. machine control command      */
:-public          batch/1       , line_smooth/1 , initialize/3  ,
user_sym_def/1  , mac_reset/0   , user_sym_term/0               .

:-mode            initialize(+,+,+)             ,
user_sym_def(+) , batch(+)      , line_smooth(+)

/*  84  */
initialize(A,B,C):-
        write_seil('$MI'),write_seil(A),
        write_seil(B),write_seil(C).
/*  85  */
mac_reset:-
        write_seil('$MR').
/*  86  */
line_smooth(I):-
        write_seil('$ML'),write_seil(I).
/*  87  */
sel_erase(I):-
        write_seil('$ME'),write_seil(I).
/*  88  */
user_sym_def(A):-
        write_seil('$ML'),write_code(16,A).
/*  89  */
user_sym_term:-
        write_seil('$MT').
/*  90  */
batch(I):-
        write_seil('$MB'),write_seil(I).

%
```

```
/*      7. I/O primitive command        */

:- public
flush_all_eve/0 , term_dev/2     , assoc_dev/5    , init_dev/2     ,
flush_dev_eve/2 , set_echo_id/4  , bell/3         , ena_dev/2      ,
set_echo_surf/3 , hard_cop/1     , buffer/3       , dis_dev/2      ,
set_hit_aper/3  , emerg_key/3    , sense_dist/3   , lamp/3         ,
set_echo_typ/3  , samp_time/3    , set_val/4      , await_eve/1    ,
await_any_but/1 , get_data/0     , await_pic/2    ,
set_echo_class/3         , set_echo_seg/3         , await_key/2     ,
set_init_val_loc_2/4     , set_init_val_loc_3/5   , await_stroke/2  ,
await_any_bot_get_val/2 , set_init_val_val/3      , set_echo_pos/4  ,
await_any_bot_get_loc/2 , set_init_mess/3         , read_samp_dev/2 .

:- mode                    term_dev(+,+) , init_dev(+,+)          ,
flush_dev_eve(+,+)       , set_echo_id(+,+,+,+)  , bell(+,+,+)           ,
set_echo_surf(+,+,+)     , hard_cop(+)           , buffer(+,+,+)         ,
set_hit_aper(+,+,+)      , emerg_key(+,+,+)      , sense_dist(+,+,+)     ,
set_echo_typ(+,+,+)      , samp_time(+,+,+)      , set_val(+,+,+,+)      ,
await_any_but(+)         ,
set_echo_class(+,+,+)    , set_echo_seg(+,+,+)   , await_key(+,+)        ,
set_init_val_val(+,+,+) , assoc_dev(+,+,+,+,+,+) , read_samp_dev(+,+)    ,
set_init_val_loc_2(+,+,+,+)      , await_eve(+)  , await_pic(+,+)        ,
set_init_val_loc_3(+,+,+,+,+)    , ena_dev(+,+)  , await_stroke(+,+)     ,
await_any_bot_get_val(+,+)       , dis_dev(+,+)  , set_echo_pos(+,+,+,+) ,
await_any_bot_get_loc(+,+)       , lamp(+,+,+)   , set_init_mess(+,+,+)  .

/*  91  */
init_dev(I,J):-
        write_seil('$DZ'),write_code(8,I),write_code(8,J).
/*  92.0  */
ena_dev(I,J):-
        write_seil('$DE1'),write_code(8,I),write_code(8,J).
/*  92.1  */
dis_dev(I,J):-
        write_seil('$DE0'),write_code(8,I),write_code(8,J).
/*  93  */
term_dev(I,J):-
        write_seil('$DT'),write_code(8,I),write_code(8,J).
/*  94  */                /* ----------  Developing  ---------- */
%assoc_dev(B,I,J,K,L):-
%       write_seil('$DO'),write_seil(B),write_code(8,I),
%       write_code(8,J),write_code(8,K),write_code(8,L).
/*  95  */
flush_all_eve:-
        write_seil('$DA').
/*  96  */
flush_dev_eve(I,J):-
        write_seil('$DV'),write_code(8,I),write_code(8,J).
/*  97  */
set_echo_typ(T,I,J):-
        write_seil('$DY'),write_code(8,T),
        write_code(8,I),write_code(8,J).
%
```

```
/*  98  */
set_echo_class(I,J,C):-
        write_seil('$DY'),write_code(8,C),
        write_code(8,I),write_code(8,J).
/*  99  */
set_echo_seg(I,J,A):-
        write_seil('$DS'),write_code(8,I),
        write_code(8,J),write_code(16,A).
/* 100 */
set_echo_id(I,J,A,B):-
        write_seil('$DI'),write_code(8,I),
        write_code(8,J),write_code(16,A),write_code(16,B).
/* 101  */
%set_echo_surf(I,J,A):-
%       write_seil('$DR'),write_code(8,I),
%       write_code(8,J),write_code(16,A).
/* 102  */
bell(B,I,J):-
        write_seil('$DB'),write_seil(B),
        write_code(8,I),write_code(8,J).
/* 103  */              /* ----------  Developing  ---------- */
hard_cop(N):-
        write_seil('$DH'),write_code(8,N).
/* 104  */              /* ----------  Developing  ---------- */
emerg_key(B,I,J):-
        write_seil('$DK'),write_seil(B),
        write_code(8,I),write_code(8,J).
/* 105  */
lamp(B,I,J):-
        write_seil('$DL'),write_seil(B),
        write_code(8,I),write_code(8,J).
/* 106  */              /* ----------  Developing  ---------- */
%set_hit_aper(B,I,J):-
%       write_seil('$DQ'),write_seil(B),
%       write_code(8,I),write_code(8,J).
/* 107  */              /* ----------  Developing  ---------- */
buffer(I,J,B):-
        write_seil('$DF'),write_code(8,I),
        write_code(8,J),write_code(16,B).
/* 108  */              /* ----------  Developing  ---------- */
%samp_time(T,I,J):-
%       write_seil('$DM'),write_code(8,T),
%       write_code(8,I),write_code(8,J).
/* 109  */
sense_dist(I,J,D):-
        write_seil('$DN'),write_code(8,I),
        write_code(8,J),write_code(16,D).
/* 110  */              /* ----------  Developing  ---------- */
%set_init_val_loc_2(I,J,X,Y):-
%       write_seil('$DJ2'),
%       write_code(8,I),write_code(8,J),
%       write_code(16,X),write_code(16,Y).
%set_init_val_loc_3(I,J,X,Y,Z):-
%       write_seil('$DJ3'),
%       write_code(8,I),write_code(8,J),
%       write_code(16,X),write_code(16,Y),write_code(16,Z).
```

```
/* 111  */              /* ----------  Developing  ---------- */
%set_init_val_val(I,J,V):-
%       write_seil('$DU'),write_code(8,I),
%       write_code(8,J),write_code(16,V).
/* 112  */              /* ----------  Developing  ---------- */
%set_val(I,J,Vmin,Vmax):-
%       write_seil('$DO'),
%       write_code(8,I),write_code(8,J),
%       write_code(16,Vmin),write_code(16,Vmax).
/* 113  */              /* ----------  Developing  ---------- */
%set_init_mess(I,N,LIST):-
%       X is N*I ,length(X,LIST),write_seil('$DX'),
%       wite_seil(I),write_code(16,N),write_ascii_listt(LIST).
/* 114  */
set_echo_pos(I,J,X,Y):-
        write_seil('$DP'),
        write_code(8,I),write_code(8,J),
        write_code(16,X),write_code(16,Y).
/* 115  */
await_eve(T):-
        write_seil('$DW'),write_code(16,T).
/* 116  */
get_data:-
        write_seil('$DG').
/* 117  */
await_pic(N,T):-
        write_seil('$D1'),
        write_code(8,N),write_code(16,T).
/* 118  */
await_key(N,T):-
        write_seil('$D2'),
        write_code(8,N),write_code(16,T).
/* 119  */
await_any_but(T):-
        write_seil('$D3'),write_code(16,T).
/* 120  */
await_stroke(N,T):-
        write_seil('$D4'),
        write_code(8,N),write_code(16,T).
/* 121  */
await_any_bot_get_loc(N,T):-
        write_seil('$D5'),
        write_code(8,N),write_code(16,T).
/* 122  */
await_any_bot_get_val(N,T):-
        write_seil('$D6'),
        write_code(8,N),write_code(16,T).
/* 123  */
%read_samp_dev(I,J):-
%       write_seil('$D7'),
%       write_code(8,I),write_code(8,J).
%
```

```
:-public
rep_data/3       , rep_dada_keyb/4        , rep_data_str_3/4       ,
rep_data_but/4   , rep_data_str_2/4       , rep_data_but_val/7     ,
rep_data_pic/8   , rep_data_but_loc_2/6   , rep_data_but_loc_3/7   ,
rep_data_val/6   , rep_data_loc_2/5       , rep_data_loc_3/6       ,
set_level/2      , rep_data_event/4       , disp_lcd_mess/3        .

:-mode
rep_data(-,-,-)                    , rep_data_pic(-,-,-,-,-,-,-,-) ,
rep_dada_keyb(-,-,-,-)             , rep_data_but(-,-,-,-)         ,
rep_data_str_2(-,-,-,-)            , rep_data_str_3(-,-,-,-)       ,
rep_data_but_val(-,-,-,-,-,-,-)    , rep_data_loc_2(-,-,-,-,-)     ,
rep_data_loc_3(-,-,-,-,-,-)        , rep_data_val(-,-,-,-,-,-)     ,
rep_data_event(-,-,-,-)            , set_level(-,-)                ,
rep_data_but_loc_2(-,-,-,-,-,-)    , disp_lcd_mess(-,-,-)          ,
rep_data_but_loc_3(-,-,-,-,-,-,-)                                  .

/* 124  */
rep_data(I,J,E):-
        write_seil('$D'),read_code(2,I),
        read_code(2,J)  ,read_code(4,E).
/* 124.1  */
rep_data_pic(Dev_c,Dev_no,Echo,Class,Segment,ID,X,Y):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,Class),read_code(4,Segment),
        read_code(4,ID)   ,read_code(4,X)   ,
        read_code(4,Y)    ,read_CR.
/* 124.2  */
rep_dada_keyb(Dev_c,Dev_no,Echo,String):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,X),read_string(X,String).
/* 124.3  */
rep_data_but(But_dev_c,But_dev_no,Echo):-
        rep_dada(But_dev_c,But_dev_no,Echo),read_CR.
/* 124.4  */
rep_data_str_2(Dev_c,Dev_no,Echo,List):-
        rep_dada(Dev_c,Dev_no,Echo),read_code(4,N),
        read_co_list_2(N,List),read_CR.
rep_data_str_3(Dev_c,Dev_no,Echo,List).
        rep_dada(Dev_c,Dev_no,Echo),read_code(4,N),
        read_co_list_3(N,List),read_CR.
%
```

```
/* 124.5  */
rep_data_but_loc_2(Dev_c,Dev_no,Echo,B,X,Y):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,B),read_code(4,X),
        read_code(4,Y),read_CR.
rep_data_but_loc_3(Dev_c,Dev_no,Echo,B,X,Y,Z):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,B),read_code(4,X),
        read_code(4,Y),read_code(4,Z),read_CR.
/* 124.6  */            /* ----------  Developing  ---------- */
rep_data_but_val(Dev_c,Dev_no,Echo,B,X,Y,Z):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,B),read_code(4,X),
        read_code(4,Y),read_code(4,Z),read_CR.
/* 124.7  */
rep_data_loc_2(Dev_c,Dev_no,Echo,X,Y):-
        rep_dada(Dev_c,Dev_no,Echo),read_code(4,X),
        read_code(4,Y),read_CR.
rep_data_loc_3(Dev_c,Dev_no,Echo,X,Y,Z):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,X),read_code(4,Y),
        read_code(4,Z),read_CR.
/* 124.8  */            /* ----------  Developing  ---------- */
rep_data_val(Dev_c,Dev_no,Echo,X,Y,Z):-
        rep_dada(Dev_c,Dev_no,Echo),
        read_code(4,X),read_code(4,Y),
        read_code(4,Z),read_CR.
/* 124.9  */
rep_data_event(Event_dev_c,Event_dev_no,Echo):-
        rep_dada(Event_dev_c,Event_dev_no,Echo),read_CR.
/* 124.10  */
set_level(Dev_end_on,View_event):-
        X is Dev_end_on + View_event,
        write_seil('$D8'),write_code(8,X).
/* 124.11  */
disp_lcd_mess(I,J,Text):-
        ( ( atomic(Text) , ! , name(Text,X) ) ;
          X = Text                                ),
        length(N,X),write_seil('$D9'),
        write_code(8,I),write_code(8,J),
        write_code(16,N),write_ascii_list(N,Text),read_CR.
%
```

```
:-public          rep_rem_mem/1          ,
rep_echo_pos/2  , rep_echo_sur/2         , rep_dev_stat/4         ,
rep_cur_pos_2/2 , rep_cur_pos_3/3        , rep_open_seg_name/1    ,
rep_echo/2      , rep_seg_detect/2       , rep_class_trans_2/4    ,
rep_keyb/2      , rep_but/1              , rep_str/2              ,
rep_echo_seg/2  , rep_locat_2/2          , rep_locport/2          ,
rep_val/5       , rep_str_dim_2/3        , rep_str_dim_3/3        ,
rep_dev_ass/2   , rep_pick/5             , rep_cur_err/2          ,
rep_echo_id/3   , rep_class_detect/2     , rep_echo_id1/2         ,
rep_seg_prio/1  , rep_echo_class/3       , rep_class_trans_3/7    ,
rep_locat_3/2   , rep_class_visi/2       , rep_class_high/2       ,
rep_seg_high/2  , rep_retain_seg_all/1   , rep_seg_in_class/2     ,
rep_seg_visi/2  , rep_class_trans_typ/2  , rep_seg_trans_typ/2    ,
rep_seg_model_trans_3/10        , rep_seg_model_trans_2/6        .

:-mode  rep_rem_mem(-)   , rep_class_trans_3(+,-,-,-,-,-,-)      ,
rep_cur_pos_2(-,-)               , rep_cur_pos_3(-,-,-)          ,
rep_open_seg_name(-)             , rep_seg_detect(+,-)           ,
rep_class_trans_2(+,-,-,-)       , rep_echo(+,+)                 ,
rep_echo_sur(+,+)                , rep_echo_pos(+,+)             ,
rep_keyb(+,+)                    , rep_but(+)                    ,
rep_str(+,+)                     , rep_echo_seg(+,+)             ,
rep_locat_2(+,+)                 , rep_locport(+,+)              ,
rep_val(+,+,-,-,-)               , rep_str_dim(+,+,-)            ,
rep_loc_dim(+,+,-)               , rep_dev_stat(+,+,-,-)         ,
rep_dev_ass(+,+)                 , rep_pick(+,+,-,-,-)           ,
rep_cur_err(-,-)                 , rep_class_detect(+,-)         ,
rep_echo_id(+,+,-)               , rep_echo_id1(+,-)             ,
rep_echo_class(+,+,-)            ,
rep_seg_prio(-)                  , rep_seg_in_class1(+,-)        ,
rep_locat_3(+,+)                 , rep_seg_visi(+,-)             ,
rep_seg_high(+,-)                , rep_class_visi(+,-)           ,
rep_class_high(+,-)              , rep_retain_seg_all(-)         ,
rep_retain_seg_all1(+,-)         , rep_class_trans_typ(+,-)      ,
rep_seg_trans_typ(+,-)           , rep_seg_in_class(+,-)         ,
rep_seg_model_trans_2(+,-,-,-,-,-)                 ,
rep_seg_model_trans_3(+,-,-,-,-,-,-,-,-,-)         .
%
```

```
/*  ??  */
rep_rem_mem(N):-
        write_seil('$Q64$R64'),resd_code(4,X),read_CR.
/*  126.1  */
rep_cur_pos_2(X,Y):-
        write_seil('$Q00$R00'),
        read_code(4,X),read_code(4,Y),read_CR.
/*  126.2  */           /* ----------  Developing  ---------- */
rep_cur_pos_3(X,Y,Z):-
        write_seil('$Q02$R02'),
        read_code(4,X),read_code(4,Y),
        read_code(4,Z),read_CR.
/*  126.3  */
rep_open_seg_name(Seg),
        write_seil('$Q07$R07'),
        read_code(4,Seg),read_CR.
/*  126.4  */
rep_seg_detect(Seg,B):-
        write_seil('$Q1?'),write_code(16,Seg),
        write_seil('$R1?'),read_seil(B),read_CR.
/*  126.5  */
rep_class_trans_2(Class,R,Tx,Ty):-
        write_seil('$Q22'),write_code(16,Class),
        write_seil('$R22'),
        read_code(4,R),read_code(4,Tx),
        read_code(4,Ty),read_CR.
/*  126.6  */
rep_seg_model_trans_2(Seg,Sx,Sy,R,Tx,Ty):-
        write_seil('$Q23'),write_code(16,Seg),
        write_seil('$R23'),
        read_code(4,Sx),read_code(4,Sy),
        read_code(4,R),read_code(4,Tx),
        read_code(4,Ty),read_CR.
/*  126.7  */           /* ----------  Developing  ---------- */
rep_echo(I,J):-
        write_seil('$Q3:'),write_code(8,I),
        write_code(8,J),write_seil('$R3:').
/*  126.8  */           /* ----------  Developing  ---------- */
rep_echo_sur(I,J):-
        write_seil('$Q3;'),write_code(8,I),
        write_code(8,J),write_seil('$R3;').
/*  126.9  */           /* ----------  Developing  ---------- */
rep_echo_pos(I,J):-
        write_seil('$Q3<'),write_code(8,I),
        write_code(8,J),write_seil('$R3<').
/*  126.10 */           /* ----------  Developing  ---------- */
rep_keyb(I,J):-
        write_seil('$Q3='),write_code(8,I),
        write_code(8,J),write_seil('$R3=').
/*  126.11 */           /* ----------  Developing  ---------- */
rep_but(I):-
        write_seil('$Q3>'),write_code(8,I),
        write_seil('$R3>').
%
```

```
/*  126.12 */           /* ----------  Developing  ---------- */
rep_str(I,J):-
        write_seil('$Q3?'),write_code(8,I),
        write_code(8,J),write_seil('$R3?').
/*  126.13 */           /* ----------  Developing  ---------- */
rep_echo_seg(I,J):-
        write_seil('$Q40'),write_code(8,I),
        write_code(8,J),write_seil('$R40').
/*  126.14 */           /* ----------  Developing  ---------- */
rep_locat_2(I,J):-
        write_seil('$Q41'),write_code(8,I),
        write_code(8,J),write_seil('$R41').
/*  126.15 */           /* ----------  Developing  ---------- */
rep_locport(I,J):-
        write_seil('$Q42'),write_code(8,I),
        write_code(8,J),write_seil('$R42').
/*  126.16 */
rep_val(I,J,X,Y,Z):-
        write_seil('$Q43'),write_code(8,I),
        write_code(8,J),write_seil('$R43'),
        read_code(4,X),read_code(4,Y),
        read_code(4,Z),read_CR.
/*  126.17 */
rep_str_dim(I,J,B).
        write_seil('$Q44'),write_code(8,I),
        write_code(8,J),write_seil('$R44'),
        read_seil(4,B),read_CR.
/*  126.18 */
rep_loc_dim(I,J,B):-
        write_seil('$Q45'),write_code(8,I),
        write_code(8,J),write_seil('$R45'),
        read_seil(B),read_CR.
/*  126.19 */
rep_dev_stat(I,J,A,B):-
        write_seil('$Q48'),write_code(8,I),
        write_code(8,J),write_seil('$R48'),
        read_seil(A),read_seil(B),read_CR.
/*  126.20 */           /* ----------  Developing  ---------- */
rep_dev_ass(I,J):-
        write_seil('$Q49'),write_code(8,I),
        write_code(8,J),write_seil('$R49').
%
```

```
/*  126.21 */
rep_pick(I,J,Class,Seg,Id):-
        write_seil('$Q4:'),write_code(8,I),
        write_code(8,J),write_seil('$R4:'),
        read_code(4,Class),read_code(4,Seg),
        read_code(4,Id)   ,read_CR.
/*  126.22 */
rep_cur_err(C,N):-
        write_seil('$Q4;$R4;'),
        read_code(2,C),read_code(4,N),read_CR.
/*  126.23 */
rep_class_detect(Class,B):-
        write_seil('$Q4;'),write_code(16,Class),
        write_seil('$R4;'),
        read_seil(B),read_CR.
/*  126.24 */           /* ----------  Developing  ---------- */
rep_echo_id(I,J,List):-
        write_seil('$Q53'),write_code(8,I),
        write_code(8,J),write_seil('$R53'),
        read_code(N),rep_echo_id1(N,List).
rep_echo_id1(0,[]):-!,read_CR.
rep_echo_id1(N,[Seg,Id|list]):-
        read_code(4,Seg),read_code(4,Id),
        N1 is N - 1 , ! ,rep_echo_id1(N1,List).
/*  126.25 */
rep_echo_class(I,J,Class):-
        write_seil('$Q54'),write_code(8,I),
        write_code(8,J),write_seil('$R54'),
        read_seil(2,Class),read_CR.
/*  126.26 */           /* ----------  Developing  ---------- */
rep_seg_prio(X):-
        write_seil('$Q55'),
        write_seil('$R55').
/*  126.27 */
rep_class_trans_3(Class,Rx,Ry,Rz,Tx,Ty,Tz):-
        write_seil('$Q24'),write_code(16,Class),
        write_seil('$R24'),
        read_code(4,Rx),read_code(4,Ry),
        read_code(4,Rz),read_code(4,Tx),
        read_code(4,Ty),read_code(4,Tz),read_CR.
/*  126.28 */
rep_seg_model_trans_3(Seg,Sx,Sy,Sz,Rx,Ry,Rz,Tx,Ty,Tz):-
        write_seil('$Q25'),write_code(16,Seg),
        write_seil('$R25'),
        read_code(4,Sx),read_code(4,Sy),read_code(4,Sz),
        read_code(4,Rx),read_code(4,Ry),read_code(4,Rz),
        read_code(4,Rx),read_code(4,Ry),read_code(4,Rz),read_CR.
/*  126.29 */           /* ----------  Developing  ---------- */
rep_locat_3(I,J):-
        write_seil('$Q41'),write_code(8,I),
        write_code(8,J),write_seil('$R41').
%
```

```
/*  126.30 */
rep_seg_visi(Seg,B):-
        write_seil('$Q1<'),write_code(16,Seg),
        write_seil('$R1<'),
        read_seil(B),read_CR.
/*  126.31 */
rep_seg_high(Seg,B):-
        write_seil('$Q1='),write_code(16,Seg),
        write_seil('$R1='),
        read_seil(B),read_CR.
/*  126.32 */
rep_class_visi(Class,B):-
        write_seil('$Q4<'),write_code(16,Class),
        write_seil('$R4<'),
        read_seil(B),read_CR.
/*  126.33 */
rep_class_high(Class,B):-
        write_seil('$Q4='),write_code(16,Class),
        write_seil('$R4='),
        read_seil(B),read_CR.
/*  126.34 */
rep_retain_seg_all(List):-
        write_seil('$Q06$R06'),
        read_code(4,N),rep_retain_seg_all1(N,List).
rep_retain_seg_all1(0,[]):-!,read_CR.
rep_retain_seg_all1(N,[X|List]):-
        read_code(4,X),N1 is N -1 ,!,
        rep_retain_seg_all1(N1,List).
/*  126.35 */
rep_class_trans_typ(Class,B):-
        write_seil('$Q20'),write_code(16,Class),
        write_seil('$R20'),
        read_seil(B),read_CR.
/*  126.36 */
rep_seg_trans_typ(Seg,B):-
        write_seil('$Q21'),write_code(16,Seg),
        write_seil('$R21'),
        read_seil(B),read_CR.
/*  126.37 */
rep_seg_in_class(Class,List):-
        write_seil('$Q52'),write_code(16,Class),
        write_seil('$R52'),
        read_code(4,N),rep_seg_in_class1(N,List).
rep_seg_in_class1(0,[]):-!,read_CR.
rep_seg_in_class1(N,[X|List]):-
        read_code(4,X),N1 is N-1,!,
        rep_seg_in_class1(N1,List).
```