Ａｄｖａｎｃｅ ＣＡＤ

プログラミングマニュアル

Advance CAD software version 19

プログラミングマニュアル

Advance CAD software version 19

2009 年　9 月 9 日 第 1 版
2009 年 12 月 1 日 第 2 版 (V19.01)

〒 141-8522 東京都品川区大崎１－２－２ アートヴィレッジ大崎 セントラルタワー

はじめに

本書はユーザが Advance CAD にユーザコマンドを組み込み、カスタマイズする方法を説明しています。

ユーザコマンドを追加するには、コマンド名を登録し、そのコマンドを処理するプログラムを作成する必要があります。このプログラムは関数として作成し、Advance CAD プログラム本体に結合 (バインド) します。このようにして組み込まれたユーザコマンドはシステム供給のコマンドとまったく同じに扱われます。

本書にはユーザコマンドを処理するプログラム作成に必要な Advance CAD モデルデータベースの概要と、図形処理やデータベースアクセスなどの関数の説明があります。

● 技術的なお問い合わせ先

Advance CAD の技術的なご質問は下記で受付けております。
Advance CAD ソフトウェア保守契約に加入されているお客様に限らせていただきます。

--------- Advance CAD ホットラインサービス ----------
電話番号 : 03-5434-0095
FAX 番号 : 03-5434-0054
E-mail : acad_support@ctc-g.co.jp
---------- 受付時間 : 平日 9:00 ～ 17:30 --

●

目次

目次

第 1 章　手続き

1.1　コマンドレベル

プログラミングインターフェイスを利用することで、基本コマンドと割り込みコマンドを作成することができます。割り込みレベルは数値の大きいものが、より上位の割り込みコマンドになります。

割り込みレベル	コマンドレベル	ディスパッチャ番号	標準コマンドの例
1	1	1 ～ 32	作図、製図、編集
2	5	81 ～ 88	トリム、幾何図形の修正
3	4	65 ～ 80	削除、ベリファイ
4	3	49 ～ 64	属性設定
5	6	89 ～ 90	テキストライブラリ
6	7	91 ～ 94	ズーム、再表示
7	8	95 ～ 98	テンポラリポイント
修飾子	(2)	33 ～ 48	

ユーザコマンドを追加できるコマンドレベルは 1、5、4、3 です。修飾子も新たに追加できます。

コマンドレベル	コマンドの種類	ユーザが追加できるディスパッチャ番号
1	基本	32
5	割り込み	88
4	割り込み	80
3	割り込み	64
(2)	修飾子	48

1.2　新コマンドの登録

新しいコマンドと必要ならば修飾子をコマンド名定義ファイルに追加します。

+ [a, b, c]！コマンド名！

[a, b, c] は、コマンド識別番号です。a はディスパッチャ番号、b はドライバ番号、c はフォーム番号です。ユーザ作成コマンド用に以下の番号を確保してあります。

ディスパッチャ番号 : 32, 64, 80, 88, 48
ドライバ番号 : 1 ～ 255
フォーム番号 : 1 ～ 32767

ディスパッチャ番号 32 でフォーム番号が 0 のコマンドはメニューの切り換えをするコマンドになります。それ以外の目的には使えません。

!コマンド名! は、コマンド識別番号に対応するコマンド名を指定します。
名前は大文字の英字で始まり、英数字、スラッシュ、またはアンダースコアの文字で構成し、15 文字以内とします。
システムのコマンド名と重複しないように、ユーザコマンド名は U で始めることをお薦めします。システムのコマンド名は menu ディレクトリにあるシステムコマンド定義ファイル ACADCMD.MEN と ACADCMDOPT.MEN に記述されています。

コマンドの実行手順はつぎのようになります。
まずオンスクリーンメニューをマウスで指示するか、またはキーボードからコマンド名を入力するとコマンドの識別番号 (a,b,c) が得られます。システムはこの識別番号で示されたコマンド処理関数を呼び出します。そしてこのコマンド処理関数で所定の処理を行ないます。

メニューの作成方法は「システム管理者の手引き」メニューの作成をご覧ください。

1.3 プログラムの呼び出し

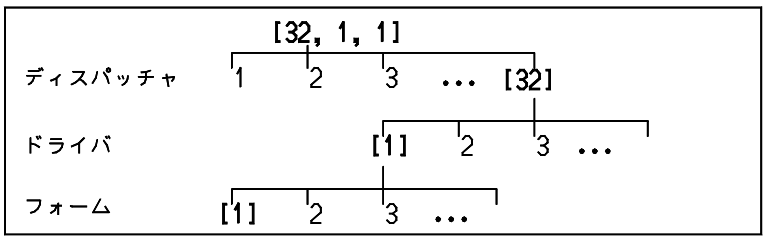
1.3.1 ディスパッチャ

コマンドを登録したら、そのコマンドの機能を実現するプログラムを作成します。
最初にそのコマンドの機能を実現するコマンド処理関数を呼び出すための手順を説明します。

コマンドが選択されると、最初にディスパッチャ番号に対応したディスパッチャ関数が呼び出されます。関数名は dspatchXX です。XX はディスパッチャ番号です。
ディスパッチャ番号により dspatch32、dspatch64、dspatch80、dspatch88 のどれかが呼び出されます。ここまではあらかじめ設定されています。

この関数はドライバ番号、フォーム番号に対応するコマンド処理関数をよびだすために使います。つまり、この関数を修正して所望のコマンド処理関数を呼び出すようにします。
もし多くのコマンドを追加する必要があれば、ドライバ番号で階層化することができます。つまり、dspatch32 ではドライバ番号によりドライバ関数を呼び出し、そこではさらにフォーム番号によりコマンドを処理するコマンド処理関数を呼び出すようにできます。

コマンド識別番号による関数の呼び出し



修飾子は、メニューから選択またはキーボードから修飾子名が入力された場合に、現在処理中のコマンドに、修飾子が選択されたことを示すトークンを渡します。

1.3.2 コマンド処理関数

呼び出しの径路が決まったら、コマンドの機能を実現するコマンド処理関数を作ります。
このコマンド処理関数は、コマンドが選択されたときに呼び出されます。通常、このときはコマンドの処理に必要な初期設定を行ない、関数を抜け出します。
つぎに何らかの入力があると、そのたびに上記径路でこのコマンド処理関数が呼び出されます。こんどは入力に応じて所定の処理を行なうようにします。
つまり、コマンドラインを構文解析し、必要なコマンドパラメータが揃った時点で処理を実行するようにプログラミングします。コマンドの終了が入力されるか別のコマンドが選択されると、このコマンドが終了します。コマンド処理関数の内容はコマンドの仕様によりさまざまですが、この関数は１つの入力があるたびに呼び出されること、次の入力を得るためにはこの関数を抜け出し入力処理関数に制御が渡らなければならないことに注意してプログラミングしてください。

1.3.3 ユーザ関数名

ディスパッチャ関数の名前は決まっていますが、それ以外の関数の名前は特に決まってはいません。しかし、システム関数と重複しないようにユーザ関数名は u で始まる名前をつけることをお薦めします。

```
/*
 * Filename : dspatch32.cpp
 * Category : User dispatcher (level 1 commands)
 */

#include "acaddef.h"
#include "acadusr.h"
#include "acadupi.h"

/*****************************************************************************
   Purpose
        Example of user dispatcher
   Inputs
        token   Token
   Outputs
        None
*/
void dspatch32(TOKEN *token) {
  switch (Idriver(1)) {
  case 1:
    udr01(Iformat(1), token);
    break;
  case 2:
    udr02(Iformat(1), token);
    break;
  case 3:
    udr03(Iformat(1), token);
    break;
  default:
    break;
  }
} /* dspatch32 */
```

```
/*************************************************************************
   Purpose
        Example of user driver #1
   Inputs
        form    Form number
        token   Token
   Outputs
        None
*/
void udr01(int form, TOKEN *token) {
  switch (form) {
  case 1:
    ucmd01(token);
    break;
  case 2:
    ucmd02(token);
    break;
  case 3:
    ucmd03(token);
    break;
  default:
    break;
  }
} /* udr01 */
```

1.4 ユーザ関数の組み込み (UNIX 版)

- **プラットフォーム**

  Sun Solaris 10 SPARC
  Red Hat Enterprise Linux Vesion 4

- **コンパイラ**
  - Sun Solaris 10 SPARC 版

    Sun Studio 10

  - Red Hat Enterprise Linux version 4 版

    g++ (gcc 3.4.3) 以上

- **動的リンク**

  プログラム（実行可能形式）はアプリケーションのオブジェクトとオペレーティングシステムが用意しているオブジェクトライブラリを結合（バインド）することにより作成します。
  たとえば、X ウインドウシステムのもとで動作するプログラムはライブラリ libX11 が必要です。C++ 言語でかかれたプログラムは C++ ライブラリも一緒に組み込まなければなりません。

  バインドの時に必要なオブジェクトをライブラリからコピーしてきて、実行形式ファイルの一部分として取り込んでしまう方法を静的リンクといい、そうしてできた実行形式ファイルを静的リンクの実行形式（Statically linked executable) といいます。
  この実行形式は、実行形式ファイルさえできてしまえば、実行時にはオブジェクトライブラリは不要です。

  もうひとつは、動的リンクの実行形式（Dynamically linked executable）です。バインド時に、ライブラリは参照しますが、オブジェクトを実行形式の一部として取り込むことはしません。そして、そのプログラムの実行時に必要なライブラリをバインドします。したがって、動的リンクの実行形式は必要なオブジェクトライブラリが揃っていないと実行できません。

この形式の実行形式はファイルサイズが小さくなります。そして、同時に実行している複数のプログラムが同じオブジェクトライブラリを参照しているならば、メモリ上のひとつのオブジェクトライブラリを共有しますので、メモリの節約の効果があります。

静的リンクで使用するオブジェクトライブラリの名前は libX11.a のように .a がつきます（Archive Library)。動的リンクで使用するオブジェクトライブラリは共用ライブラリ（Shared Library または Shared Object）とよばれ、libX11.so のように .so がつく名前です。

Advance CAD の実行形式は、動的リンクで作成されており、以下のファイルで構成されています。

```
acad.exe              （実行可能形式）
libacadcore.so        （システム・共有ライブラリ）
libacadbc.so          （基本・共有ライブラリ）
libacadmdl.so         （モデル・共有ライブラリ）
libacadcui.so         （コマンド・共有ライブラリ）
libacadcontrol.so     （コマンド制御・共有ライブライリ）
libacadgr.so          （グラフィックス・共有ライブラリ）
libacadstd.so         （標準機能・共有ライブライリ）
libacadxrf.so         （モデル検索・共有ライブラリ）
libacadcadam.so       (CADAM Option・共有ライブラリ）
libacadcatia.so       (Catia Option・共有ライブラリ）
libacaddxf.so         (DXF Opttion・共有ライブラリ）
libacadnc.so          (NC Option・共有ライブラリ）
libacadpid.so         (P&ID/Sequence Option・共有ライブラリ）
libacadtiff.so        (TIFF・共有ライブラリ）
libacaduser.so        （ユーザ プログラミング インタフェース・共有ライブラリ）
```

Advance CAD が使用する共有ライブラリを調べるには SUN Solaris ではコマンドを使います。例えば次のように表示します。

```
% ldd -r acad.exe
        libacadbc.so => /home/acad16r/sparc/user/libacadbc.so
        libacadcontrol.so => /home/acad16r/sparc/user/libacadcontrol.so
        libacadcore.so.16 => /home/acad16r/sparc/user/libacadcore.so.16
        libacadcui.so => /home/acad16r/sparc/user/libacadcui.so
        libacadgr.so => /home/acad16r/sparc/user/libacadgr.so
        libacadmdl.so => /home/acad16r/sparc/user/libacadmdl.so
        libacadstd.so => /home/acad16r/sparc/user/libacadstd.so
        libacaduser.so => /home/acad16r/sparc/user/libacaduser.so
        libacadtiff.so => /home/acad16r/sparc/user/libacadtiff.so
        libXi.so.5 => /usr/openwin/lib/sparcv9/libXi.so.5
        libX11.so.4 => /usr/openwin/lib/sparcv9/libX11.so.4
        libsocket.so.1 => /usr/lib/sparcv9/libsocket.so.1
        libnsl.so.1 => /usr/lib/sparcv9/libnsl.so.1
        libintl.so.1 => /usr/lib/sparcv9/libintl.so.1
        libw.so.1 => /usr/lib/sparcv9/libw.so.1
        libm.so.1 => /usr/lib/sparcv9/libm.so.1
        libgen.so.1 => /usr/lib/sparcv9/libgen.so.1
        libc.so.1 => /usr/lib/sparcv9/libc.so.1
        libXext.so.0 => /usr/openwin/lib/sparcv9/libXext.so.0
        libdl.so.1 => /usr/lib/sparcv9/libdl.so.1
        libmp.so.2 => /usr/lib/sparcv9/libmp.so.2
        /usr/platform/SUNW,Sun-Blade-100/lib/sparcv9/libc_psr.so.1
%
```

Advance CAD は libacadbc.so、libacadcontrol.so、libacadcore.so、libacaduser.so 等を動的リンクしていることが分かります。

- **ユーザ関数の組み込み**

Advance CAD は libacaduser.so を動的リンクしています。ユーザは、ユーザ関数を含む共有ライブラリ libacaduser.so をバインドしなおすだけです。つづいて Advance CAD を起動するだけで、ユーザ関数をテストできます。

この方法は、Advance CAD 全体をリンクするのではなく、ユーザの共有ライブラリだけをリンクするのでリンク時間が短く、デバッグの効率が向上する利点があります。

● **システムファイル**

以下のファイルは user ディレクトリにあります。

変更してはならないオブジェクトライブラリ

```
libacadbc.so
libacadcore.so
libacadcontrol.so
libacadcui.so
libacadmdl.so
libacadgr.so
libacadstd.so
libacadxrf.so
libacadcadam.so
libacadcatia.so
libacaddxf.so
libacadnc.so
libacadpid.so
libacadtiff.so
```

変更してよいオブジェクトライブラリ

```
libacaduser.a libacaduser.so
```

変更してはならないソースコード

```
acaddef.h acadprm.h acadupi.h acadusr.h
```

以下のファイルは sample/USER ディレクトリにあります。必要なファイルを user ディレクトリにコピーし、コピー後のファイルを修正します。

```
dspatch32.cpp dspatch64.cpp dspatch80.cpp dspatch88.cpp
udbaccess.cpp usrcom.cpp usrmdm.cpp
Makefile
```

● **コンパイル／リンク**

Makefile を修正して、make とタイプすれば、コンパイルをし、ライブラリ libacaduser.a と libacaduser.so をつくり直します。

Makefile の修正は、SRCC に C++ コンパイラでコンパイルするソースコードのファイル名を追加するだけです。
たとえば、udr01.cpp, udr02.cpp, udr03.cpp を追加するなら次のように変更します。

```
SRCC =  dspatch32.cpp dspatch64.cpp dspatch80.cpp dspatch88.cpp ¥
        usrcom.cpp usrmdm.cpp udbaccess.cpp ¥
        udr01.cpp udr02.cpp udr03.cpp
```

● **実行**

- Solaris 版

Advance CAD を実行する前に、環境変数 LD_LIBRARY_PATH_64 を確認します。
これは共用ライブラリのあるディレクトリ名のリストで、libacadbc.so、libacadcore.so、libacaduser.so など Advance CAD のライブラリがあるディレクトリ名を含めておかなければなりません。

設定

```
% setenv LD_LIBRARY_PATH_64 "/acad/user:/usr/lib/sparcv9:/usr/openwin/lib/sparcv9:/usr/dt/
lib/sparcv9"
```

確認

```
% echo $LD_LIBRARY_PATH_64
/acad/user:/usr/lib/sparcv9:/usr/openwin/lib/sparcv9:/usr/dt/lib/sparcv9
```

- Red Hat Enterprise Linux 版

Advance CAD を実行する前に、環境変数 LD_LIBRARY_PATH を確認します。
これは共用ライブラリのあるディレクトリ名のリストで、libacadbc.so、libacadcore.so、libacaduser.so 等 Advance CAD のライブラリがあるディレクトリ名を含めておかなければなりません。

設定

```
% setenv LD_LIBRARY_PATH "/acad/user:/usr/openwin/lib:/usr/lib"
```

確認

```
% echo $LD_LIBRARY_PATH
/acad/user:/usr/openwin/lib:/usr/lib
```

● 参考

Makefile

```
# Advance CAD Version 18 (ITOCHU Techno-Science Corpolation)
#
# Purpose : Make user's object library for Sun Solaris
#
.SUFFIXES: $(.SUFFIXES) .cpp
.cpp.o
        $(COMPILE.cc) $(OUTPUT_OPTION) $<
CC = CC

SHELL  = /bin/sh
SOFILE = libacaduser.so
ARFILE = libacaduser.a

#
# Compiler option for Solaris 2 64bit Kernel
CFLAGS = -KPIC -O -c -misalign -xarch=generic64
LDFLAGS = -G -xarch=generic64
#
# Name of C source codes
SRCC =  dspatch32.cpp dspatch64.cpp dspatch80.cpp dspatch88.cpp ¥
        udbaccess.cpp usrcom.cpp usrmdm.cpp
#
# Name of object files
OBJS = $(SRCC:.cpp=.o)
#
# Define targets
all: $(SOFILE) $(ARFILE)

$(SOFILE): $(OBJS)
        $(CC) $(LDFLAGS) -o $@ `lorder $(OBJS) | tsort`
$(ARFILE): $(OBJS)
        ar r $@ $(OBJS)
$(OBJS): $$(@:.o=.cpp)
        $(CC) $(CFLAGS) $<
```

```
clean:
        -@rm $(SOFILE) $(ARFILE) $(OBJS)
help:
        @echo "make        : Up to date library (so, a)"
        @echo "make clean  : Remove library (so, a) and *.o"
#
# End of make
```

1.5 ユーザ関数の組み込み (Windows 版)

- プラットフォーム
  Windows XP Professional + Service Pack 2 以上

- コンパイラ
  Visual C++ .NET 2003 日本語版以上

- リンク
  Advance CAD Windows XP Professional 版 (以降 Windows 版) は以下のファイルで構成されています。

acad.exe	(実行可能形式)
acadbc.dll	(基本・ダイナミックリンクライブラリ)
acadcore.dll	(システム・ダイナミックリンクライブラリ)
acadcontrol.dll	(コマンド制御・ダイナミックリンクライブラリ)
acadcui.dll	(コマンド・ダイナミックリンクライブラリ)
acadmdl.dll	(モデル・ダイナミックリンクライブラリ)
acadgr.dll	(グラフィック・ダイナミックリンクライブラリ)
acaddlg.dll	(ダイアログ・ダイナミックリンクライブラリ)
acaddlgcv.dll	(ダイアログ・ダイナミックリンクライブラリ)
acaddlgcv2.dll	(ダイアログ・ダイナミックリンクライブラリ)
acaddlglib.dll	(ダイアログ・ダイナミックリンクライブラリ)
acadstd.dll	(標準機能・ダイナミックリンクライブラリ)
acadsxf.dll	(SXF・ダイナミックリンクライブラリ)
acadxrf.dll	(モデル検索・ダイナミックリンクライブラリ)
acaddlgxrf.dll	(ダイアログ・ダイナミックリンクライブラリ)
acadcadam.dll	(CADAM・ダイナミックリンクライブラリ)
acadcatia.dll	(Catia・ダイナミックリンクライブラリ)
acaddxf.dll	(DXF&DWG・ダイナミックリンクライブラリ)
acadnc.dll	(Nc・ダイナミックリンクライブラリ)
acadpid.dll	(P&ID シーケンス図・ダイナミックリンクライブラリ)
acadswi.dll	(SolidWorks I/F・ダイナミックリンクライブラリ)
common_lib.dll	(SXF 仕様 .sfc 用ライブラリ)
common_lib_AP202.dll	(SXF 仕様 .p21 用ライブラリ)
acadtiff.dll	(TIFF ・ダイナミックリンクライブラリ)
acaduser.dll	(ユーザプログラミングインタフェース・ダイナミックリンクライブラリ)

- ユーザ関数の組み込み
  Advance CAD は acaduser.dll をダイナミックリンクしています。ユーザはユーザ関数を含む acaduser.dll を作成し直すことによって関数を組み込むことができます。
  続いて Advance CAD を起動するだけでユーザ関数をテストすることができます。
  古いバージョンの dll は使用できませんので、必ず作成し直してください。

- システムファイル
  以下のファイルは exe ディレクトリにあります。

  変更してはならないファイル
  acad.exe

acadbc.dll
acadcore.dll
acadcontrol.dll
acadcui.dll
acadmdl.dll
acadgr.dll
acaddlg.dl
acaddlglib.dll
acaddlgcv.dll
acaddlgcv2.dll
acaddlgxrf.dll
acadstd.dll
acadsxf.dll
acadxrf.dll
common_lib.dll
common_lib_AP202.dll
acadtiff.dll
acadcadam.dll
acadcatia.dll
acaddxf.dll
acadnc.dll
acadpid.dll
acadswi.dll

変更してよいファイル
acaduser.dll

バージョン 12 から Advance CAD に必要なダイナミックリンクライブラリはすべて acad.exe と同じディレクトリに置きます。
これは Windows の標準的な方法で、以前のように dll ファイルのあるディレクトリを環境変数 PATH に追加する必要がありません。

以下のファイルは user ディレクトリにあります。

変更してはならないファイル
acadbc.lib
acadcontrol.lib
acadcui.lib
acadmdl.lib
acadgr.lib
acadstd.lib
acaddef.h acadprm.h acadupi.h acadusr.h

変更してよいファイル
acaduser.lib, acaduser.exp (nmake で作成される)

以下のフィルは sample/USER ディレクトリにあります。必要なファイルを user ディレクトリにコピーし、コピー後のファイルを修正します。
dspatch32.cpp dspatch64.cpp dspatch80.cpp dspatch88.cpp
usrcom.cpp usrmdm.cpp udbaccess.cpp
Makefile

- **コンパイル／リンク**

Advance CAD を停止してからコンパイル／リンクを実行してください。
Makefile を修正して、nmake とタイプすれば、コンパイルし、ダイナミックリンクライブラリ acaduser.dll をカレントディレクトリに作り直し、かつ dll を exe ディレクトリにコピーします。

Makefile の修正は、CSRCS にコンパイルするソースコードファイル名を追加するだけです。たとえば、udr01.c, udr02.c, udr03.c を追加する場合はつぎのように変更します。

```
CSRCS =¥
dspatch32.cpp dspatch64.cpp dspatch80.cpp dspatch88.cpp ¥
udbaccess.cpp usrcom.cpp usrmdm.cpp ¥
udr01.cpp udr02.cpp udr03.cpp
```

- **実行**

Advance CAD を実行し、ユーザ関数へ制御が渡ることを確認してください。
システムはダイナミックリンクライブラリをつぎの順番で探しますので、ユーザ関数へ制御が渡らない場合は acaduser.dll が検索対象ディレクトリにあることを確認してください。

(1) 実行形式 (acad.exe) があるディレクトリ (デフォルトでは dll をここに置く)
(2) カレントディレクトリ (起動ディレクトリ)
(3) 32 ビット Windows システムディレクトリ (SYSTEM32)
(4) 16 ビット Windows システムディレクトリ (SYSTEM16)
(5) Windows ディレクトリ
(6) PATH 環境変数に設定したディレクトリ

- **参考**

Makefile

```
# AdvanceCAD Ver 18 (ITOCHU TECHNO-SCIENCE Corporation)
#
# Purpose : Make user DLL
#
!include <ntwin32.mak>

CFLAGS   = $(cflags) $(cdefs) $(cvarsdll) -DAcadUserIMPL -nologo
CINCS    = -I.

EXEPATH  = ..¥exe
USERDLL  = acaduser.dll
EXPFILE  = acaduser.exp
IMPLIB   = acaduser.lib
DLLFLAGS = $(dlllflags)
DLLLIBS  = $(guilibsdll)
NETLIBS  =
ACADLIB  = acadbc.lib acadcontrol.lib acadcui.lib acadgr.lib ¥
           acadmdl.lib acadstd.lib
LIBS     = $(DLLLIBS) $(NETLIBS) $(ACADLIB)

CSRCS = ¥
dspatch32.cpp dspatch64.cpp dspatch80.cpp dspatch88.cpp ¥
udbaccess.cpp usrcom.cpp usrmdm.cpp

COBJS = $(CSRCS:.cpp=.obj)
all: dll copy
.cpp.obj:
        $(cc) $(CFLAGS) $(CINCS) $<

dll: $(USERDLL)

$(EXPFILE): $(COBJS)
        $(implib) -machine:$(CPU) -out:$(IMPLIB) -def: @<<
$(COBJS)
<<

$(USERDLL): $(COBJS) $(EXPFILE)
        $(link) $(DLLFLAGS) -out:$@ $** $(LIBS)
```

```
copy:
        copy $(USERDLL) $(EXEPATH)

clean:
        del *.obj $(EXPFILE) $(IMPLIB) $(USERDLL)
# end of file
```

第 2 章　会話型入出力

この章では入出力について説明します。

2.1　入力

2.1.1　トークン

入力情報は入力処理関数がトークンに変換してユーザ関数に渡します。たとえばキーボードから 50,100 と入力された場合は座標トークンに、50 と入力された場合は数値トークンになります。

入力されたトークンはユーザ関数 dspatchXX の入力引数として渡されます。

```
dspatch32(TOKEN *token)
```

トークンタイプ一覧 (token->typ)

```
TknCMD  (==  1) : コマンド
TknMDF  (== 16) : 修飾子
TknCOD  (==  2) : 座標
TknDIG  (==  6) : デジタイズ座標
TknSCL  (==  3) : 数値
TknTXT  (==  4) : 文字列
TknITM  (== 17) : アイテム選択
TknPNT  (== 18) : テンポラリポイント
TknIDP  (== 15) : アイテム識別子
TknBSP  (==  7) : バックスペース
TknSPC  (== 11) : スペースキー
TknVEC  (==  8) : ベクトル
TknMZN  (== 13) : メッセージゾーン
TknGZN  (== 12) : グラフィックゾーン
TknUZN  (== 14) : テンポラリウインドウ
TknEOC  (==  5) : ＣＥ（Enter キー、ボタンに割り付けた <CE>）
TknEXIT (== 31) : コマンド終了
```

構造体 TOKEN は acaddef.h 内に、トークンタイプは acadprm.h 内に定義されている。

- TknCMD : コマンドトークン

コマンドが選択されたときに渡されるトークン。
選択されたコマンドの識別番号は token->cid にセットされている。

```
token->cid[0]　　：　ディスパッチャ番号
token->cid[1]　　：　ドライバ番号
token->cid[2]　　：　フォーム番号
```

- TknMDF : 修飾子トークン

修飾子が指定されたときに渡されるトークン。
指定された修飾子の識別番号は token->cid にセットされている。

```
token->cid[0]          :  ディスパッチャ番号
token->cid[1]          :  ドライバ番号
token->cid[2]          :  フォーム番号
```

● TknCOD : 座標トークン（モデル座標）

座標値が入力されたときに渡されるトークン。
座標値は token->pnt にセットされている。

● TknDIG : デジタイズ座標トークン（モデル座標）

マウス／タブレットで図形領域をピックしたときに渡されるトークン。
ピックされたビューポート番号は token->vp に、座標値は token->pnt にセットされている。
シフトキー／コントロールキーが押されていたときは token->scl に 0.0 以外が、押されていないときは 0.0 がセットされている。

● TknSCL : 数値トークン

数値が入力されたときに渡されるトークン。
数値は token->scl にセットされている。

● TknTXT : 文字列トークン

文字列が入力されたときに渡されるトークン。
文字列は token->txt にセットされている。文字列の最後は '\0' がセットされている。日本語文字コードは EUC。

● TknITM : アイテム選択トークン

「複数アイテムの自動選択コマンド」によってアイテムが選択されたときに渡されるトークン。
選択されたアイテムはハイライトアイテムリストに格納されている。
詳しくは後述の「例．複数アイテムの自動選択」を参照。

● TknPNT : テンポラリポイントトークン

「テンポラリポイント作成コマンド」によってテンポラリポイントが作成されたときに渡されるトークン。テンポラリポイント座標は token->pnt にセットされている。
詳しくは後述の「例．テンポラリポイントの作成」を参照。

● TknIDP : アイテム識別子トークン

マクロの idptr 関数によってこのトークンが渡される。通常の操作ではこのトークンは発生しない。アイテム識別子は token->scl にセットされている。負のときもある。

● TknBSP : バックスペーストークン

文字や数値が 1 文字も入力されていないときに BackSpace キーまたは Del キーが入力されたときに渡されるトークン。
文字列や座標値入力中の BackSpace や Del キーの入力は入力処理関数が処理を行うのでこのトークンは発生しない。

● TknSPC : スペースキートークン

スペースキーが入力されたときに渡されるトークン。

● TknVEC : ベクトルトークン

@DX10 などのベクトル入力やベクトル作成割り込みコマンドでベクトルが作成されときに渡されるトークン。
ベクトル値 DX、DY は token->pnt にセットされている。

● TknMZN : メッセージゾーントークン

メッセージゾーンがピックされたときに渡されるトークン。
ピックされたメッセージゾーンの位置（行、列）は token->pnt にセットされている。

```
token->pnt.x        :  列
token->pnt.y        :  行
```

● TknGZN : グラフィックゾーントークン

グラフィックゾーンがピックされたときに渡されるトークン。
ピックされたグラフィックゾーンの位置（行、列）は token->pnt にセットされている。

```
token->pnt.x        :  列
token->pnt.y        :  行
```

このトークンを得るには予めトークン種類設定関数 tknmsk001 で入力可に指定しておくこと。

● TknUZN : テンポラリウインドウトークン

マクロの mopen 関数によって作成されたテンポラリウインドウがピックされたときに渡されるトークン。
ピックされたテンポラリウインドウの位置（行、列）は token->pnt にセットされている。

```
token->pnt.x        :  列
token->pnt.y        :  行
```

このトークンを得るには予めトークン種類設定関数 tknmsk001 で入力可に指定しておくこと。

● TknEOC : ＣＥ トークン

Enter キーまたはボタンに割り付けた <CE> が入力されたときに渡されるトークン。

● TknEXIT : コマンド終了トークン

コマンドが切り替わったときに渡されるトークン。
基本コマンド（コマンドレベル１）の場合は他の基本コマンドに切り替わるときに渡される。割り込みコマンドの場合は自分と同じかまたは下位の割り込みレベルのコマンドおよび基本コマンドに切り替わるときに渡される。
F1 キー入力による強制終了のときもこのトークンが渡される。

2.1.2 トークンの例

「二点間線」0,0 100,0 <CE> dig1 dig2 「連結線」としたときに二点間線コマンドに渡されるトークンは以下のようになります。

```
「二点間線」      :  コマンドトークン（TknCMD）
0,0               :  座標トークン（TknCOD）
100,0             :  座標トークン（TknCOD）
<CE>              :  ＣＥ トークン（TknEOC）
dig1              :  デジタイズ座標トークン（TknDIG）
dig2              :  デジタイズ座標トークン（TknDIG）
「連結線」        :  コマンド終了トークン（TknEXIT）
                     その後「連結線」コマンドにコマンドトークンが渡される。
```

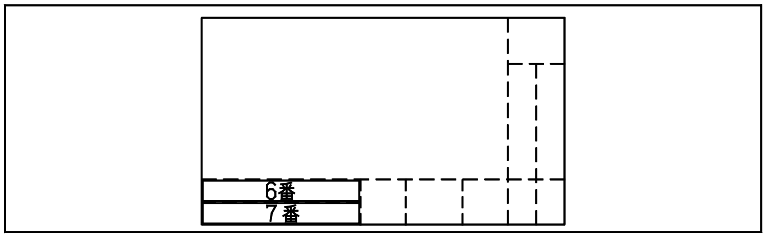
2.2 出力

2.2.1 メッセージ

Advance CAD はメッセージ出力専用の装置をもっていないため、メッセージはすべて
グラフィックスクリーンの一部にメッセージ用ゾーンを確保し、そこに出力します。

メッセージ用ゾーンは、６番と７番の２つです。
６番のゾーンは、ユーザの入力を一時的に表示しておいたり、計算結果を表示するためのゾーンです。７番のゾーンは、オペレータにつぎの操作を示すためのオペレーションメッセージとエラーメッセージ用のゾーンです。

標準メニューを使用時のメッセージゾーン 6番、7番



オペレーションメッセージや、エラーメッセージはすべてメッセージファイルに登録します。ユーザプログラミングインタフェース用のメッセージファイルは、MSG90.TXT (オペレーションメッセージファイル) と ERR90.TXT (エラーメッセージファイル) の２つです。

メッセージは次の書式で１行で記述します。

+（メッセージ番号）″メッセージ″

メッセージおよびエラーメッセージ番号は、それぞれ 9000000 から 9999999 です。
メッセージとエラーメッセージは独立しているので、番号がだぶってもかまいません。

バージョン 12 からテキストファイルを直接読み込むようにしました。従ってテキストファイルをバイナリファイルに作り直す必要はありません。

メッセージ番号を重複して指定すると、最初に現われたものが有効になります。
このため、MSG.INP 内で !MSG90! を ERR.INP 内で !ERR90! を一番最初に記述してあります。

以下のシステムモジュールは、メッセージ番号を指定するとメッセージを所定の位置に表示します。

Opmsgcode	オペレーションメッセージ表示
Errorcode	エラーメッセージ表示
Errorb	ブザーを鳴らす
Mesagdisp	メッセージ表示
Mesageras	メッセージの消去

2.2.2 アイテム

アイテムの出力、いいかえればアイテムの作成および表示ですが、これについては別の章で説明します。

2.3 日本語文字コード

Advance CAD 標準の日本語文字コードは EUC です。プログラム中の文字列はメッセージやエラーメッセージをはじめ、すべて日本語 EUC、マルチバイト文字列です。
Windows の場合、日本語文字コードは MS 漢字コード (シフト JIS とも言う) です。

この場合ソースコードに直接 MS 漢字コードの文字定数を記述してはいけません。またディスクファイルから日本語文字列を読み込んだ場合は、MS 漢字コードを日本語 EUC に変換しなければなりません。ファイルに出力する場合は、逆の変換をしなければなりません。
日本語文字コードの変換のため、２つの関数を用意しています。

```
sjis2euc        :  MS 漢字コードから日本語 EUC への変換
euc2sjis        :  日本語 EUC から MS 漢字コードへの変換
```

ソースコードに MS 漢字コードを直接記述する場合はつぎのようにします。

```
char text[80];
sjis2euc("日本語の文字列", text, sizeof(text));
Mesagdisp(MZONECOLOR1, 1, 1, 0, 10*strlen(text), text);
```

テキストファイルを１行読み込み表示したい場合は、つぎのようにします。

```
FILE *fp
char cbuf[80], text[80];
if ((fp=fopen("myfile.txt", "r")) == (FILE *)NULL)
    return;
if (fgets(cbuf, sizeof(cbuf), fp) == (char *)NULL) {
    fclose(fp);
    return;
}
fclose(fp);
sjis2euc(cbuf, text, sizeof(text));
Mesagdisp(MZONECOLOR1, 1, 1, 0, 10*strlen(text), text);
```

このように Advance CAD のシステム関数、広域変数に設定する文字列は、すべて日本語 EUC でなければなりません。誤って日本語 EUC 以外の日本語文字コードが混入すると、モデルファイル、シンボルファイルなどの重要なデータファイル中の文字列は信用出来ず、正しい結果が得られませんので注意してください。
メニュー、メッセージ、エラーメッセージのソースファイルは MS 漢字コードでかまいません。

2.4 例

2.4.1 座標値、数値および文字列トークンの取り扱い

```
/*
 * 入力された座標値、数値および文字列をメッセージ領域に表示する。
 * <CE> でこのコマンドを終了する。
 */

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

#define CMDLVL 3 /* コマンドレベル３の割り込みコマンド */

void ucmd01(TOKEN *token) {
  switch (token->typ) {

  case TknCMD: /* コマンドトークン */
    break;
```

```
  case TknCOD: /* 座標トークン */
  case TknDIG: /* デジタイズ座標トークン */
    /*
     * 入力された座標値をメッセージ領域に表示する。
     * メッセージ番号 9000001 : "座標値 X"
     * メッセージ番号 9000002 : "座標値 Y"
     */
    Mesageras(MSGZONE, 0, 0);
    Mesagdisp(MZONECOLOR1, 1, 1, 9000001, 3, &token->pnt.x);
    Mesagdisp(MZONECOLOR1, 2, 1, 9000002, 3, &token->pnt.y);
    break;

  case TknSCL: /* 数値トークン */
    /*
     * 入力された数値をメッセージ領域に表示する。
     * メッセージ番号 9000003 : "数値"
     */
    Mesageras(MSGZONE, 0, 0);
    Mesagdisp(MZONECOLOR1, 1, 1, 9000003, 3, &token->scl);
    break;

  case TknTXT: /* 文字列トークン */
    /*
     * 入力された文字列をメッセージ領域に表示する。
     * メッセージ番号 9000004 : "文字列"
     */
    Mesageras(MSGZONE, 0, 0);
    Mesagdisp(MZONECOLOR1, 1, 1,9000004, strlen(token->txt) * 10, token->txt);
    break;

  case TknEOC: /* ＣＥ トークン */
    /*
     * Advance CAD では ＣＥ トークンの場合は概ね以下のように処理している。
     * 基本コマンドの場合は今までの操作で設定された条件に基づき処理を行う。
     * その後このコマンド処理関数を初期状態に戻して次のトークンの入力を待つ。
     * 割り込みコマンドの場合はコマンド処理を終了させる。
     * この例ではコマンドレベル３の割り込みコマンドであるので cmdidcla 関数
     * を呼出してこのコマンドを終了する。
     * これ以後に入力されたトークンは現在動作中のより下位の割り込みコマンド
     * または基本コマンドに渡されるようになる。
     */
    cmdidcla(CMDLVL);
    return;

  case TknEXIT: /* コマンド終了トークン */
    return;

  default: /* その他のトークン */
    Errorcode(9000001); /* 無効な入力です */
    break;
  }

  /* 入力促進メッセージ */
  Opmsgcode(CMDLVL, 9000005); /* 座標値、数値、文字列を入力／終了は<CE> */
}
```

2.4.2 修飾子トークンの取り扱い

```
/*
 * 修飾子により、入力された数値が文字高さか文字角度かを判定する。
```

```
 */

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

#define CMDLVL 1 /* 基本コマンド */

void ucmd01(TOKEN *token) {
  switch (token->typ) {

  case TknCMD: /* コマンドトークン */
    break;

  case TknMDF: /* 修飾子トークン */
    if (token->cid[0] == 34 &&
        token->cid[1] == 1 &&
        token->cid[2] == CSWTSIZE) { /* 修飾子 TSIZE */
      /* EMPTY */
    } else if (token->cid[0] == 34 &&
               token->cid[1] == 1 &&
               token->cid[2] == CSWANG) { /* 修飾子 ANG */
      /* EMPTY */
    } else { /* その他の修飾子 */
      Errorcode(9000001); /* 無効な修飾子です */
      /*
       * 無効な修飾子をクリアする。
       */
      cmdmdfcla(CMDLVL);
    }
    break;

  case TknSCL: /* 数値トークン */
    if (ldispatch(2) == 34 &&
        ldriver(2) == 1 &&
        lformat(2) == CSWTSIZE) {
      /*
       * 修飾子 TSIZE が指定されている。
       * 入力された数値は文字高さ。文字高さをメッセージ領域に表示する。
       * メッセージ番号 9000001 : "文字高さ"
       */
      Mesageras(MSGZONE, 1, 0);
      Mesagdisp(MZONECOLOR1, 1, 1, 9000001, 3, &token->scl);
    } else if (ldispatch(2) == 34 &&
               ldriver(2) == 1 &&
               lformat(2) == CSWANG) {
      /*
       * 修飾子 ANG が指定されている。
       * 入力された数値は文字角度。文字角度をメッセージ領域に表示する。
       * メッセージ番号 9000002 : "文字角度"
       */
      Mesageras(MSGZONE, 2, 0);
      Mesagdisp(MZONECOLOR1, 2, 1, 9000002, 3, &token->scl);
    } else {
      /*
       * 修飾子が指定されていない。
       */
      Errorcode(9000002); /* 修飾子が指定されていない */
    }
    break;
```

```
  case TknEOC: /* ＣＥ トークン */
    break;

  case TknEXIT: /* コマンド終了トークン */
    return;

  default: /* その他のトークン */
    Errorcode(9000003); /* 無効な入力です */
    break;
  }

  /* 入力促進メッセージ */
  if (Idispatch(2) == 34 &&
      Idriver(2) == 1 &&
      Iformat(2) == CSWTSIZE) {
    /*
     * 修飾子 TSIZE が指定されている。
     */
    Opmsgcode(CMDLVL, 9000003); /* 文字高さを入力 */
  } else if (Idispatch(2) == 34 &&
             Idriver(2) == 1 &&
             Iformat(2) == CSWANG) {
    /*
     * 修飾子 ANG が指定されている。
     */
    Opmsgcode(CMDLVL, 9000004); /* 文字角度を入力 */
  } else {
    /*
     * 修飾子が指定されていない。
     */
    Opmsgcode(CMDLVL, 9000005); /* 修飾子「文字高さ」「文字角度」を選択 */
  }
}
```

2.4.3　テンポラリポイントの作成

```
/*
 * 入力された二点間の距離をメッセージ領域に表示する。
 */

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

#define CMDLVL 1 /* 基本コマンド */

void ucmd01(TOKEN *token) {
  static DPOINT pnts[2]; /* テンポラリポイントの保存領域 */
  static int npnt = 0;   /* 入力済みのテンポラリポイントの点数 */

  switch (token->typ) {

  case TknCMD: /* コマンドトークン */
    /*
     * 初期設定。
     */
    npnt = 0;
    break;
```

```
case TknCOD: /* 座標トークン */
case TknDIG: /* デジタイズ座標トークン */
case TknPNT: /* テンポラリポイントトークン */
  /*
   * テンポラリポイントを作成する。
   * IdentPoint 関数の概要
   * TknCOD が渡されたときの戻り値
   *   IDENT_SUCCESS  : 渡された点をテンポラリポイントとする。
   * TknDIG が渡されたときの戻り値
   *   IDENT_SUCCESS  : テンポラリポイントが作成できた。
   *   IDENT_FAILURE  : テンポラリポイントが作成できない。
   *   IDENT_CONTINUE : テンポラリポイントモードが交点や投影点などであり、
   *                    指定された１点ではテンポラリポイントが作成できな
   *                    いのでテンポラリポイント作成コマンド（割り込みコ
   *                    マンド）を起動した。
   *                    テンポラリポイント作成コマンドが終了すると、
   *                    その結果はトークンタイプ TknPNT で通知される。
   * TknPNT が渡されたときの戻り値
   *   IDENT_SUCCESS  : テンポラリポイントが作成できた。
   *   IDENT_FAILURE  : テンポラリポイントが作成できない。
   * 戻り値が IDENT_SUCCESS のときは２番目の引数にテンポラリポイント
   * が設定されている。
   * 戻り値が IDENT_FAILURE のときは IdentPoint 関数内でエラーメッセージ
   * を表示している。
   */
  if (IdentPoint(token, &pnts[npnt]) != IDENT_SUCCESS) {
    return;
  } else {
    /*
     * テンポラリポイントが作成できた。
     */
    npnt++;
    if (npnt == 2) {
      /*
       * 二点間の距離を計算しメッセージ領域に表示する。
       * メッセージ番号 9000001 : ″距離″
       */
      double dist = gmudst(&pnts[0], &pnts[1], 2, 1);
      Mesageras(MSGZONE, 0, 0);
      Mesagdisp(MZONECOLOR1, 1, 1, 9000001, 3, &dist);
      npnt = 0;
    }
  }
  break;

case TknBSP: /* バックスペーストークン */
  if (npnt == 1) {
    /*
     * １点目のテンポラリポイントをキャンセルする。
     */
    npnt = 0;
  } else {
    Errorcode(9000001); /* 取り消すテンポラリポイントがない */
  }
  break;

case TknEOC: /* ＣＥ トークン */
  /*
    初期化する。
  */
  npnt = 0;
  break;
```

```
  case TknEXIT: /* コマンド終了トークン */
    return;

  default: /* その他のトークン */
    Errorcode(9000002); /* 無効な入力です */
    break;
  }

  /* 入力促進メッセージ */
  if (npnt == 0) {
    Opmsgcode(CMDLVL, 9000002); /* 線分の１点目を入力 */
  } else {
    Opmsgcode(CMDLVL, 9000003); /* 線分の２点目を入力 */
  }
}
```

2.4.4　アイテムの選択

※V19 からは次候補アイテムの有無を調べる方法が変更になりました。

```
/*
 * 選択された曲線アイテムの中点をメッセージ領域に表示する。
 */

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

#define CMDLVL 1

void ucmd01(TOKEN *token) {
  int idptr;

  switch (token->typ) {

  case TknCMD: /* コマンドトークン */
    /*
     * 初期設定。
     * IdentItem 関数の初期化を行う。
     * 入力処理関数は、トークンタイプ TknCMD でコマンド処理関数を呼出す前に
     * IdentItem 関数を初期化している。従って下の１行は不要。あってもよい。
     */
    (void)IdentItem(CMDLVL);
    break;

  case TknCOD: /* 座標トークン */
  case TknDIG: /* デジタイズ座標トークン */
  case TknIDP: /* アイテム識別子トークン */
  case TknSPC: /* スペースキートークン */
    /*
     * アイテムを選択する。
     * IdentItem 関数の概要
     * TknCOD、TknDIG が渡されたときの処理
     *   渡された点でアイテムをピックする。
     *   ピックできればそのアイテムのアイテム識別子を返す。
     *   ピックできなければ 0 を返す。
     * TknIDP が渡されたときの処理
     *   渡されたアイテム識別子を調べる。
```

```
     *   アイテム識別子が正しければそのアイテム識別子を返す。
     *   アイテム識別子が正しくなければ 0 を返す。
     * TknSPC が渡されたときの処理
     *   次候補アイテムの有無を調べる。
     *   次候補アイテムがあればそのアイテムのアイテム識別子を返す。
     *   次候補アイテムがなければ 0 を返す。
     */
    if ((idptr = IdentItem(CMDLVL, token, 0)) == 0) {
      /* アイテムが選択できなかった */
      Errorcode(9000001);
    } else {
      /*
       * アイテムが選択できた。
       * 選択された曲線アイテムの中点座標をメッセージ領域に表示する。
       * メッセージ番号 9000001 : "中点 X"
       * メッセージ番号 9000002 : "中点 Y"
       */
      DPOINT pmid;
      Mesageras(MSGZONE, 0, 0);
      if (gmpntpdv(1, idptr, 1, 0.0, (DPOINT *)NULL, &pmid) == 1) {
          Mesagdisp(MZONECOLOR1, 1, 1, 9000001, 3, &pmid.x);
          Mesagdisp(MZONECOLOR1, 2, 1, 9000002, 3, &pmid.y);
      }
    }
    break;

  case TknEOC: /* ＣＥ トークン */
    break;

  case TknEXIT: /* コマンド終了トークン */
    return;

  default: /* その他のトークン */
    Errorcode(9000002); /* 無効な入力です */
    break;
  }

  /* 入力促進メッセージ */
  if (IdentItemCandidate(CMDLVL) >= 2) { /* 次候補アイテムの有無を調べる */
    /*
     * 次候補アイテムあり。
     */
    Opmsgcode(CMDLVL, 9000003); /* 中点を求める曲線アイテムを選択
                                   または<SP>（次候補）を入力 */
  } else {
    /*
     * 次候補アイテムなし。
     */
    Opmsgcode(CMDLVL, 9000004); /* 中点を求める曲線アイテムを選択 */
  }
}
```

2.4.5 複数アイテムの自動選択

※V19 からは次候補アイテムの有無を調べる方法が変更になりました。

```
/*
 * 複数アイテムを選択し、現在選択されているアイテム数と最後に選択された
 * アイテムのアイテム識別子をメッセージ領域に表示する。
 *<CE> で選択状態を初期化する。
```

```
 */

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

#define CMDLVL 1

void ucmd01(TOKEN *token) {

  switch (token->typ) {

  case TknCMD: /* コマンドトークン */
    /*
     * 初期設定。
     * IdentItems 関数の初期化を行う。
     * 入力処理関数は、トークンタイプ TknCMD でコマンド処理関数を呼出す前に
     * IdentItems 関数を初期化している。従って下の１行は不要。あってもよい。
     */
    (void)IdentItems(CMDLVL);
    break;

  case TknCOD: /* 座標トークン */
  case TknDIG: /* デジタイズ座標トークン */
  case TknIDP: /* アイテム識別子トークン */
  case TknSPC: /* スペースキートークン */
  case TknBSP: /* バックスペーストークン */
  case TknITM: /* アイテム選択トークン */
    /*
     * アイテムを選択する。
     * IdentItems 関数の概要
     * TknCOD、TknDIG が渡されたときの戻り値
     *   IDENT_SUCCESS  : アイテムがピックできた。
     *   IDENT_CONTINUE : 渡された点でアイテムがピックできなかったので
     *                    矩形または多角形領域でのアイテム選択とみなし、
     *                    複数アイテムの自動選択コマンド（割り込み
     *                    コマンド）を起動した。
     *                    複数アイテムの自動選択コマンドが終了すると、
     *                    その結果はトークンタイプ TknITM で通知される。
     * TknIDP が渡されたときの戻り値
     *   IDENT_SUCCESS  : アイテムが選択できた。
     *   IDENT_FAILURE  : アイテムが選択できなかった。（渡されたアイテム
     *                    識別子が不正）
     * TknSPC が渡されたときの戻り値
     *   IDENT_SUCCESS  : 次候補アイテムあり。
     *                    以前のアイテムを排除し、次候補アイテムを選択する。
     *   IDENT_FAILURE  : 次候補アイテムはない。
     * TknBSP が渡されたときの戻り値
     *   IDENT_SUCCESS  : 最後の操作を元に戻した。
     *   IDENT_FAILURE  : 元に戻すべきアイテムが存在しない。
     * TknITM が渡されたときの戻り値
     *   IDENT_SUCCESS  : 複数アイテムの自動選択コマンドでアイテムが選択
     *                    できた。
     *   IDENT_FAILURE  : アイテムが選択できなかった。
     * 戻り値が IDENT_SUCCESS のときは、選択されたアイテムはハイライト
     * アイテムリストに追加されている（または排除されている）。
     * 戻り値が IDENT_FAILURE のときは IdentItems 関数内でエラーメッセージ
     * を表示している。
     */
    if (IdentItems(CMDLVL, token, 0) != IDENT_SUCCESS) {
      return;
```

```
    } else {
      /*
       * アイテムが選択／排除できた。
       * 選択されているアイテム数および最後に選択されたアイテムの
       *アイテム識別子をメッセージ領域に表示する。
       * メッセージ番号 9000001 : ″アイテム数″
       * メッセージ番号 9000002 : ″アイテム識別子″
       */
      int nitem = rptitmnum(CMDLVL); /* ハイライトアイテムリスト内のアイテム数 */
      Mesageras(MSGZONE, 0, 0);
      Mesagdisp(MZONECOLOR1, 1, 1, 9000001, 4, &nitem);
      if (nitem > 0) {
        int *idptrs = rptitmptr(CMDLVL); /* ハイライトアイテムリストのポインター */
        int idptr = idptrs[nitem - 1];   /* 最後に選択されたアイテムのアイテム識別子 */
        Mesagdisp(MZONECOLOR1, 2, 1, 9000002, 4, &idptr);
      }
    }
    break;

  case TknEOC: /* ＣＥ トークン */
    /*
     * 選択状態を初期化する（IdentItems 関数の初期化を行う）。
     */
    (void)IdentItems(CMDLVL);
    break;

  case TknEXIT: /* コマンド終了トークン */
    return;

  default: /* その他のトークン */
    Errorcode(9000001); /* 無効な入力です */
    break;
  }

  /* 入力促進メッセージ */
  if (IdentItemsCandidate(CMDLVL) >= 2) { /* 次候補アイテムの有無を調べる */
    /*
     * 次候補アイテムあり。
     */
    Opmsgcode(CMDLVL, 9000003); /* アイテムを選択または<SP>（次候補）を入力 */
  } else {
    /*
     * 次候補アイテムなし。
     */
    Opmsgcode(CMDLVL, 9000004); /* アイテムを選択 */
  }
}
```

第 3 章　アイテムの読みだし・作成・修正

3.1　データベース概要

ユーザ関数を Advance CAD に組み込んで Advance CAD のデータベースのデータを読みだしたり、データを追加するには、Advance CAD データベースに関する知識が必要です。
この節では、データベースのアイテムを読みだしたり、データベースにアイテムを追加あるいは修正するために必要なデータベースの概要を説明します。

● **データベース**

Advance CAD に必要な情報を保持します。大きくわけると「ステータス情報」と「アイテムの情報」という２つの部分からなります。

ステータス情報は、円の半径などのモーダルパラメータやスクリーンレイアウトなど、ユーザが定義した情報を保持しています。
新モデルではこれらはシステムの初期値が設定されています。ファイルしたモデルではファイルしたときの設定値が保存されています。

アイテムの情報は、モデルの状態によります。新モデルではアイテムは１つもありません。ファイルしたモデルではファイルしたとき存在したすべてのアイテムが保存されています。

以下に、アイテムがどのように記述されるかを説明します。

● **アイテム識別子**

データベースに存在するアイテムにはすべて固有の番号がついています。
番号はアイテム作成順に１から自動的に割り付けられます。
この番号は、あるアイテムが作成され消去されるまで変わりません。
データベースにあるアイテムを指示するには、この番号を指定しなければなりません。

● **アイテム属性**

各アイテムごとに以下のような属性を持っています。

- **アイテムの属するピクチャ番号**
- **アイテムの種類**
- **クラス番号**
- **レビジョン番号**
- **線種**
- **線幅　など**

● **アイテムの記述**

アイテムは、いくつかの「サブレコード」とよばれる単位データのあつまりで記述します。サブレコードは 63 種類あり、それぞれ用途が決まっています。

アイテムの種類によってどのサブレコードをどのような順序で記述すべきかが決まっています。
たとえば、線分アイテムではつぎのようになります。

順番	サブレコード	内容
#1	3	**線分始点**
#2	4	**線分終点**
#3	31	**アイテムの終了**

最初に線分の始点をもつサブレコード、つぎに線分の終点を持つサブレコードがきます。最後にアイテムの終了をあらわすサブレコードをおきます。円／円弧アイテムの場合はつぎのようになります。

順番	サブレコード	内容
#1	3	**円弧始点**
#2	5	**円弧データ**
#3	31	**アイテムの終了**

最初に円弧始点をもつサブレコード、つぎに円弧データを持つサブレコードがきます。
円弧始点をもつサブレコードは線分の始点をもつサブレコードと同じもの (番号 3) です。やはり最後にアイテムの終了をあらわすサブレコードをおきます。

● アイテムのサイズ

ひとつのアイテムが持つことができるサブレコードの数は最大 32500 までです。
ひとつのアイテムを構成するサブレコードのデータサイズの総合計は最大 524288 Bytes までです。
この制限はコンフィグレーションファイル ACAD.SET で大きくすることができます。(マニュアル「システム管理者の手引き」の「2.2 コンフィグレーションファイル ACAD.SET」をご覧下さい。)
また現在の最大値は関数 **DBGetSitMax, DBGetBufMax** で得ることができます。

● サブレコード

サブレコードはヘッダとデータ (内容) の２つからなります。

サブレコードヘッダ

サブレコードの種類 (1-63)
データのタイプ (0-5)
- 0 : **文字** (1 BYTE)
- 1 : **整数** (2 BYTES)
- 2 : **単精度浮動小数** (4 BYTES)
- 3 : **倍精度浮動小数** (8 BYTES)
- 4 : **整数** (4 BYTES)
- 5 : **複合 (上記の組み合わせ)**

サブレコードの線種番号 (0-64)
- 0 : **アイテムと同じ**
- 1 : **非表示**
- 2-64 : **線種番号** (1-63)

サブレコードの線幅番号 (0-16)
- 0 : **アイテムと同じ**
- 1-16 : **線幅番号** (1-16)

データ数 (1-2048)

サブレコードデータ　　サブレコードの種類により決まります。

● サブレコードのデータサイズ
ひとつのサブレコードのデータサイズは 2 KBytes 以内です。

3.2 データベースアイテムの読み出し

3.2.1 アイテム属性を得る

アイテム属性 (アイテムタイプ、クラスなど) を得ます。

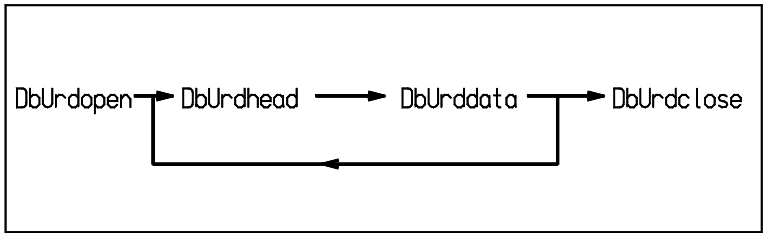
Dbvriatr　　：　アイテムの属性を得る。

3.2.2 アイテム読み出し

アイテムをオープンし先頭から順次レコードを読み出します。

DbUrdopen　　：　アイテムのサブレコードの参照のためデータベースをオープンする。
DbUrdhead　　：　サブレコードのヘッダを読み出す。
DbUrddata　　：　サブレコードのデータを読み出す。
DbUrdclose　　：　アイテムのサブレコードの参照のためオープンしたデータベースをクローズする。
DbUrdback　　：　一度参照したサブレコードを再度参照させる。

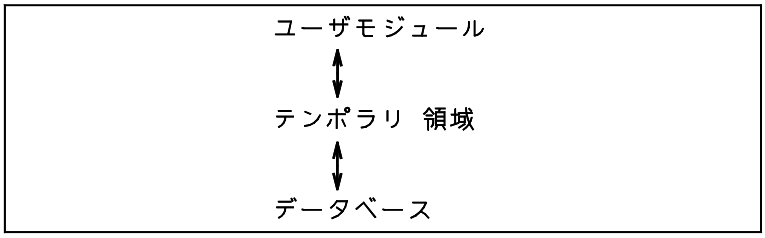
一般的な呼出し方法は、下図のようになります。



3.3 データベースアイテムの作成と修正

● テンポラリアイテム
新アイテムを作成したり既存のアイテムを修正するとき、データベースを直接更新するのではなく、「テンポラリデータ領域」を使用します。
新アイテムは一度テンポラリデータ領域に作成し、確認後データベースに追加します。
既存アイテムを修正するときは、テンポラリデータ領域にアイテムのコピーをおき、それを修正します。変更が完了したらデータベースを更新します。
この方法はいくつか有利な点があります。アイテムの作成や修正を止めたいときは、テンポラリデータを放棄するだけでよいのです。データベースを直接更新する方式ではこのようなことはやっ

かいです。またデータベースの更新を 1 か所にまとめることで、プログラミングの不注意による
データベースの破壊を防止できます。



3.3.1 テンポラリデータの構造

大きく３つの部分からなります。

● テンポラリアイテム エントリテーブル

テンポラリアイテム１つに１つのエントリがとられます。各アイテムの先頭と最後のサブレコードの番号を持ちます。この番号はサブレコードエントリテーブルを指します。したがって、１つのアイテムのサブレコードは連続していなければなりません。
またアイテムの属性 (クラス、アイテムの種類、線種など) も持ちます。
テンポラリアイテムの最大数は 256 アイテムです。これを越えてアイテムを作ろうとしたときには、自動的にデータベースに書き出されます。

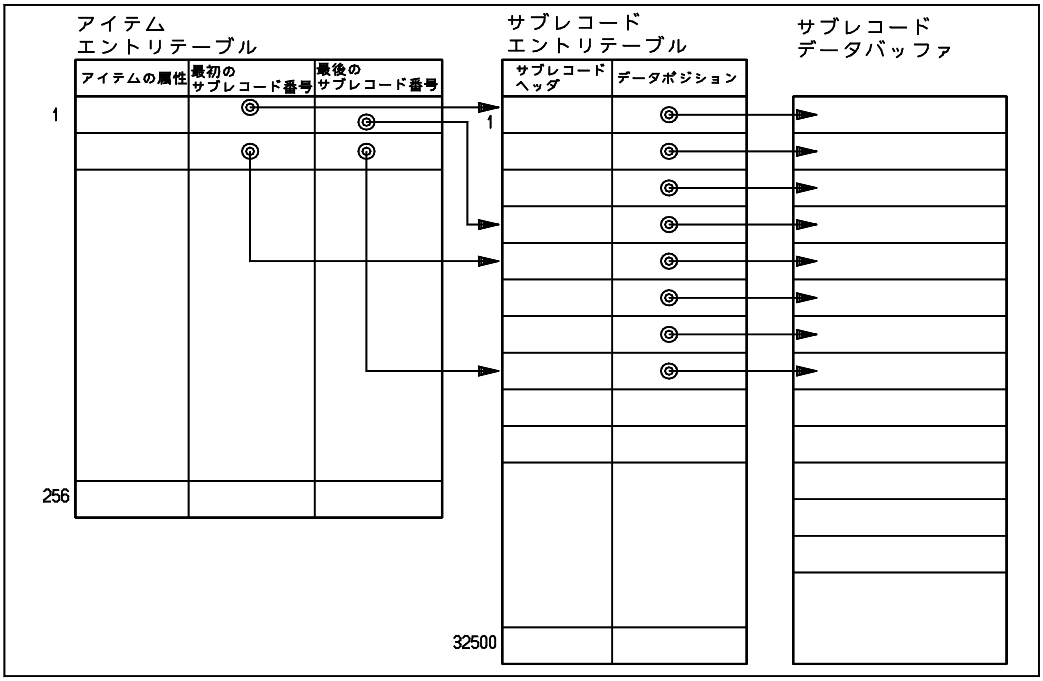
● サブレコードエントリテーブル

サブレコードのヘッダだけを持ちます。サブレコードの内容はサブレコードデータバッファにあり、サブレコードデータバッファへのポインタを持ちます。
ブレコードの最大数は 32500 レコードです。(ACAD.SET の設定で大きくできます)

● サブレコードデータバッファ

サブレコードの内容を持ちます。
容量は 524288 Bytes、およそ 512 KBytes です。(ACAD.SET の設定で大きくできます)

● 概念図



3.3.2 関数

初期化

TmpgInit : テンポラリデータ領域を初期化する。テンポラリアイテム数が 0 になる。

アイテムのオープン

TmpgOpen1 : 新しくテンポラリアイテムをオープンする。
この関数は、テンポラリアイテムエントリを確保するだけであり、アイテムのデータはなにもない。
まず全く新しいアイテムを作る。アイテム属性はコモンから取られる。つぎに既存のアイテムと同じアイテム属性でオープンする。
テンポラリアイテムをデータベースに格納するとき、データベースの既存のアイテムを更新するか、新しいアイテムとしてデータベースに格納するかを選択できる。

TmpgCopy : TmpgOpen1 が単に新しくテンポラリアイテムエントリを確保するだけなのに対して、この関数は新しいテンポラリアイテムをオープンし、データベースから指定したアイテムのデータをコピーしてくる。
APPEND モードを指示すると、カレントテンポラリアイテムの最後に、指定したアイテムのデータをデータベースからコピーし追加する。

サブレコードの追加

TmpgAddSr : カレント テンポラリアイテムの最後にサブレコードを１つ追加する。

TmpgALast : 最後のテンポラリアイテムの最後にサブレコードを１つ追加する。
もし最後のテンポラリアイテムが既に、アイテムの終わりを示すサブレコードを持っていればその直前に追加する。

アイテムのクローズ

TmpgClose : 最後のテンポラリアイテムの最後にアイテムの終わりを示すサブレコードを１つ追加しアイテムをクローズする。
このあとで、TmpgAddSr を使用してはいけない。

アイテムのデータベースへの格納

TmpgWrtoDB : テンポラリアイテムをデータベースに書き込む。

サブレコードの読出し
TmpgsityP : サブレコードのヘッダを得る。
TmpgGetSr : サブレコードの内容を取り出す
Tmpgcanon : セグメントを得る。
Tmpgfont : サブレコードのフォント番号を得る。またはフォント番号を変更する。

サブレコードの更新
TmpgWrt : サブレコードの内容を更新する。

サブレコードの移動
Tmpgdel : サブレコードを削除する。削除したサブレコードの位置はつめられる。
Tmpgmove : サブレコードを移動する。サブレコードを間に追加するために使用する。また逆に不要なサブレコードを削除するのに使用する。

アイテムの削除
Tmpgback : 最後のテンポラリアイテムを削除する。

アイテムの表示
Tmpgrptn : テンポラリアイテムをスクリーンに表示する。

ItmSR 構造体
サブレコードを関数に渡すときに使用する構造体。この構造体に値を設定する場合はマクロ SETITMSR を使用できる。

3.3.3 アイテムの作成手順

一般的な呼出し方法は、つぎのようになります。

(1) TmpgInit
新規アイテムなら (2) へ 既存アイテムなら (3) へ
(2) TmpgOpen1 → TmpgAddSr → TmpgClose 新規アイテムの作成
(3) TmpgCopy → 修正 → TmpgClose 既存アイテムの修正
(4) TmpgWrtoDB

(1) アイテム属性を設定する。
アイテムタイプは SetItemtype() で必ず設定しなければなりません。
アイテムタイプなどのシステムで使用する定数は acadprm.h で定義しています。
できるだけそれを使用すべきです。

(2) アイテムによっては、アイテム作成条件を設定する必要がある。
たとえば、ジェネラルノートをつくるときは、あらかじめテキストの文字高さやテキストフォント番号などを必要に応じて設定しなければなりません。

(3) アイテムを作業領域に作る。このアイテムはテンポラリアイテムと呼ぶ。
アイテムをオープンします。

```
TmpgOpen1(0, 0);
```

アイテムを構成するサブレコードを追加します。アイテムタイプ別のサブレコードの構成は別途記述します。

```
TmpgAddSr(&srh);
```

アイテムをターミネイトします。

```
TmpgClose(1);
```

(4)　アイテムを格納する

テンポラリアイテムをデータベースに格納します。

```
TmpgWrtoDB(0);
```

作業領域を開放します。

```
TmpgInit();
```

● 例題

以上を簡単な例題で示します。

```
#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

void ucmd02(TOKEN *token) {

    DPOINT dpnts[5] = { { 100.0, 0.0}, {0.0,  100.0},
                        {-100.0, 0.0}, {0.0, -100.0}, {100.0, 0.0} };

    if (token->typ != TknCMD)
        return;

    /* ２つの線分を作る。*/
    SetItemtype(ITMLINE);
    SetItemlwt(1);
    for (int j = 0; j < 2; ++j) {
        ItmSR srh;
        TmpgOpen1(0, 0);
        SETITMSR(srh, SITSTART, 3, 2, 0, 0, &dpnts[j]);
        TmpgAddSr(&srh);
        SETITMSR(srh, SITLINE,  3, 2, 0, 0, &dpnts[j+1]);
        TmpgAddSr(&srh);
        TmpgClose(1);
    }

    /* ストリングアイテムを作る。*/
    SetItemtype(ITMSTRING);
    SetItemlwt(2);
    TmpgOpen1(0, 0);
    SETITMSR(srh, SITSTART, 3, 2, 0, 0, &dpnts[2]);

    TmpgAddSr(&srh);
    for (int j = 3; j < 5; ++j) {
        ItmSR srh;
        SETITMSR(srh, SITLINE, 3, 2, 0, 0, &dpnts[j]);
        TmpgAddSr(&srh);
    }
    TmpgClose(1);

    /* アイテムをデータベースに格納する。*/
    TmpgWrtoDB(0);
    TmpgInit();
}
```

第 4 章　アイテムタイプ一覧

以下の表はアイテムタイプの一覧です。
パラメータ名はヘッダーファイル acadprm.h 内で定義されています。本書中の関数でアイテムタイプを指示するものがいくつかあります。その場合は、直接アイテムタイプ番号を記入するよりは、パラメータ名を記入した方が分かりやすいでしょう。

(1)　図形アイテム

タイプ番号	アイテム	パラメータ名
1	点 (Point)	ITMPOINT
2	直線 (Line)	ITMLINE
3	円／円弧 (Circular Arc)	ITMARC
4	自由曲線 (Free curve)	ITMFREE
5	ストリングアイテム (String item)	ITMSTRING
6	円錐曲線 (Conic curve - 予約)	ITMCONIC
7-8	未使用	
9	複合アイテム (Composit item)	ITMCOMP
10	未使用	

(2)　製図アイテム

タイプ番号	アイテム	パラメータ名
11	ジェネラルテキスト (General note, General label, Reference note, Reference label, cutting plane line)	ITMTEXT
12	マーク (General mark)	ITMMARK
13	寸法 (Dimension)	ITMDIM
14	幾何公差 (Geometrical Tolerance)	ITMFCS
15	ハッチングアイテム (hatching item)	ITMHATCH
16	塗り潰しアイテム (area fill item)	ITMAFL
17-20	未使用	

(3) 構造化アイテム

タイプ番号	アイテム	パラメータ名
21-26	未使用	
27	メンバアイテム	ITMMEMBER
28	APG アイテム	ITMAPG
29	アソシエイトアイテム	ITMASSC
30	シンボル (Symbol)	ITMSYMBOL
31	サブモデル (Submodel)	ITMSUBMDL

(4) SXF アイテム

タイプ番号	アイテム	パラメータ名
32	イメージ アイテム (V19-)	ITMIMAGE
33	SXF 作図部品定義 (V18-)	ITMSFIGORG
34	SXF 作図部品配置 (V18-)	ITMSFIGLOC
35	SXF 元図定義 (ハッチング領域) (V18-)	ITMCRVDEF
36	SXF 元図定義 (楕円、文字) (V18-)	ITMORGDEF
37-63	未使用	

4.1 図形アイテム

4.1.1 アイテム１ 点

サブレコード構造
1. Point (X,Y) (SR= 2)
2. End of item (SR= 31)

アイテムの最小／最大座標値
Xmin = X
Ymin = Y
Xmax = X
Ymax = Y

4.1.2 アイテム２ 線分

サブレコード構造
1. Start point(Xs,Ys) (SR= 3)
2. Line(Xe,Ye) (SR= 4)
3. End of item (SR= 31)

アイテムの最小／最大座標値
Xmin = AMIN(Xs,Xe)
Ymin = AMIN(Ys,Ye)
Xmax = AMIN(Xs,Xe)
Ymax = AMIN(Ys,Ye)

注意
部分線種／線幅の変更をほどこした線分アイテムは、複数の線分に分割される。
このとき分割された線分すべてが一直線上になければならない。
このときのサブレコードの並びは、つぎのようになる。
SR3, SR4, …, SR4, SR31

4.1.3 アイテム３ 円／円弧

サブレコード構造

1.	Start point(Xs,Ys)	(SR= 3)
2.	Arc(m,Ym,Xe,Ye,Xc,Yc,radius,angle)	(SR= 5)
3.	End of item	(SR= 31)

注意
(1) 円は円弧の特殊解（Ps==Pe）として処理する。
(2) 部分線種／線幅の変更をほどこした円弧アイテムは、複数の円弧に分割される。
このとき分割された円弧すべてがひとつの円上になければならない。
このときのサブレコードの並びは、つぎのようになる。
SR3, SR5, SR5, …, SR5, SR31

4.1.4 アイテム４ 自由曲線

サブレコード構造

1.	Start point	(SR= 3)
2.	Bezier curve #1	(SR= 6)
3.	Bezier curve #2	(SR= 6)
	:	
m+1.	Bezier curve #m	(SR= 6)
m+2.	end of item	(SR= 31)

ここで、m は３次 Bezier カーブセグメントの数。

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X)
Ymin=AMIN（各サブレコードの最小 Y)
Xmax=AMAX（各サブレコードの最大 X)
Ymax=AMAX（各サブレコードの最大 Y)

注意
(1) 曲線の種類
Composite Cubic Bezier Curve

4.1.5 アイテム５ ストリングアイテム

アイテムの定義

ライン、円弧、３次 Bezier カーブセグメントの連なる曲線。
各セグメントは連続しており、一筆書きできること。

サブレコード構造

1.	始点 (X1,Y1)	(SR= 3)
2.	線分、円弧 または ３次 Bezier カーブセグメント	(SR= 4,5 または 6)
	:	
n-1.	線分、円弧 または ３次 Bezier カーブセグメント	(SR= 4, 5 または 6)
n.	End of item	(SR=31)

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X)
Ymin=AMIN（各サブレコードの最小 Y)
Xmax=AMAX（各サブレコードの最大 X)
Ymax=AMAX（各サブレコードの最大 Y)

4.1.6 アイテム９ 複合アイテム

アイテムの定義
いくつかのアイテムを集めて１つのアイテムとしたもの。

サブレコード構造

1.	もとのアイテムの属性	(SR= 32)
2.	もとのアイテムのサブレコード	
	:	
m.	もとのアイテムの属性	(SR= 32)
m+1.	もとのアイテムのサブレコード	
	:	
n.	End of item	(SR= 31)

注意
(1) もとのアイテムのサブレコード群がそのままの順序で取り込まれる。
ただし、アイテムの終りを示す SR= 31 だけは除かれる。
(2) 複合アイテムを複合アイテムに含めてもよい。複合アイテムにはアイテムの階層はない。

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X)
Ymin=AMIN（各サブレコードの最小 Y)
Xmax=AMAX（各サブレコードの最大 X)
Ymax=AMAX（各サブレコードの最大 Y)

4.2 製図アイテム

4.2.1 アイテム１１ グラフィックステキスト

アイテムの定義
ジェネラルノート／ジェネラルラベル
リファレンスノート／リファレンスラベル
切断線

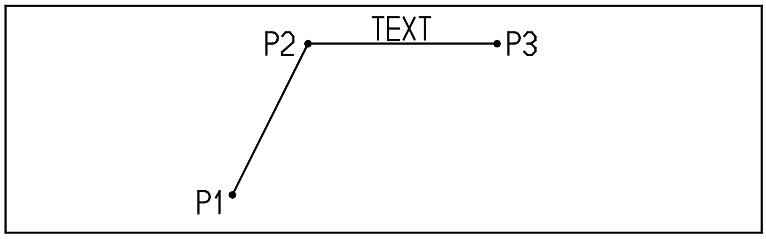
レコード構造
ジェネラルノート、ジェネラルラベルはテキストと引出線の２種類のレコードで構成される。
ジェネラルノート
テキストレコード

ジェネラルラベル
テキストレコード
引出線レコード

リファレンスノート・リファレンスラベルは、テキスト・風船・引出線の３種類のレコードで構成される。
リファレンスノート
テキストレコード
風船レコード
リファレンスラベル
テキストレコード
風船レコード
引出線レコード

切断線はテキスト・矢印・切断線の３種類のレコードで構成される。
テキスト、矢印はそれぞれ２つまで。
テキストレコード (× 2)
矢印レコード (× 2)
切断線レコード

注意
(1) 引出線の点列はマークからテキストストリングへむかって並ぶ。



(2) テキスト枠レコード、テキスト下線レコードが付くときは、テキストレコードの直後になければならない。

ジェネラルノート、ジェネラルラベル
(1) テキストレコード
1. カテゴリ (SR= 1)
メインカテゴリ #3
サブカテゴリ #0（未使用）
2. テキスト (SR=33)

テキスト枠レコード（オプショナル）
1. カテゴリ (SR= 1)
メインカテゴリ #21
サブカテゴリ #0（未使用）
2. 枠の始点 (SR=3)
3. 最初の線分 (SR=4)
4. 2 番目の線分 (SR=4)
5. 3 番目の線分 (SR=4)
6. 最後の線分 (SR=4)
テキスト下線レコード（オプショナル）

1. カテゴリ (SR= 1)
   メインカテゴリ #22
   サブカテゴリ #m（m は下線の数）
2. 下線の始点 (SR=3)
3. 下線の終点 (SR=4)
4. 下線の始点 (SR=3)
   :
2m+1. 下線の終点 (SR=4)

(2) 引出線レコード
1. カテゴリ (SR= 1)
   メインカテゴリ #2
   サブカテゴリ #1
2. 引出線の始点 (SR= 3)
3. 引出線の最初の線分 (SR= 4)
   :
m+1. 引出線の最後の線分 (SR= 4)
m+2. 矢のマーク (SR= 34)
m は引出線の点数。32 点以下。

リファレンスノート、リファレンスラベル
(1) テキストレコード
1. カテゴリ SR= 1)
   メインカテゴリ #6
   サブカテゴリ #0（文字列）
2. テキスト (SR=33)

(2) 風船レコード
1. カテゴリ (SR= 1)
   メインカテゴリ #6
   サブカテゴリ #1（マーク）
2. マーク (SR= 34)

(3) 引出線レコード
1. カテゴリ (SR= 1)
   メインカテゴリ #2
   サブカテゴリ #1
2. 引出線の始点 (SR= 3)
3. 引出線の最初の線分 (SR= 4)
   :
m+1. 引出線の最後の線分 (SR= 4)
m+2. 矢のマークの原点 (SR= 34)
m はリーダの点数。32 点以下。

切断線
(1) テキストレコード
1. カテゴリ (SR= 1)
   メインカテゴリ #18
   サブカテゴリ #1, #2（１番目か２番目かを示す）
2. テキスト (SR=33)

(2) 矢印レコード
1. カテゴリ (SR= 1)
   メインカテゴリ #19
   サブカテゴリ #1, #2（１番目か２番目かを示す）
2. 矢印マーク (SR= 34)

(3) 切断線レコード
1. カテゴリ (SR= 1)
   メインカテゴリ #2
   サブカテゴリ #1

2. 切断線の始点 (SR= 3)
3. 切断線の最初の線分 (SR= 4)
:
m+1. 切断線の最後の線分 (SR= 4)
m は切断線の点数。32 点以下。

4.2.2 アイテム１２ マークアイテム

アイテムの定義
単純なマークだけのアイテム
溶接記号
面の肌の記号

単純なマークだけのアイテム サブレコード構造
1. マーク (SR= 34)
2. End of item (SR= 31)

溶接記号 レコード構造
(1) 溶接記号レコード（矢の手前側）
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #1
2. マーク (SR= 34)

(2) 溶接の仕様レコード（矢の手前側）
このレコードは必要に応じて付ける。サブカテゴリの番号によって溶接部断面寸法やルート開先などの区別がなされる。したがって同じサブカテゴリのレコードが２つ以上あってはならない。
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #11 ～ #13
2. テキスト (SR= 33)

(3) 溶接部の表面形状マークと仕上げ方法（矢の手前側）
このレコードは、必要に応じてひとつだけ付けることができる。
表面形状マーク
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #4
2. マーク (SR= 34)
仕上げ方法の文字
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #14
2. テキスト (SR= 33)

(4) 溶接記号レコード（矢の向う側）
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #2
2. マーク (SR= 34)

(5) 溶接の仕様レコード（矢の向う側）
このレコードは必要に応じて付ける。サブカテゴリの番号によって溶接部断面寸法やルート開先などの区別がなされる。したがって同じサブカテゴリのレコードが２つ以上あってはならない。
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #21 ～ #23
2. テキスト (SR= 33)

(6) 溶接部の表面形状マークと仕上げ方法（矢の向う側）
このレコードは、必要に応じてひとつだけ付けることができる。
表面形状マーク
1. カテゴリ (SR= 1)
メインカテゴリ #19
サブカテゴリ #5
2. マーク (SR= 34)
仕上げ方法の文字
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #24
2. テキスト (SR= 33)

(7) 現場溶接記号レコード（オプショナル）
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #3
2. マーク (SR= 33)

(8) 特記事項レコード（オプショナル）
1. カテゴリ (SR= 1)
メインカテゴリ #9
サブカテゴリ #31
2. テキスト (SR= 33)

(9) 引出線レコード
1. カテゴリ (SR= 1)
メインカテゴリ #2
サブカテゴリ #1
2. 引出線の始点 (SR= 3)
3. 引出線の最初の線分 (SR= 4)
:
m+1. 引出線の最後の線分 (SR= 4)
m+2. 矢のマーク (SR= 34)
m は引出線の点数。32 点以下。

(10) 矢の向こう側を示す基線 (オプショナル)
1. カテゴリ (SR= 1)
メインカテゴリ #2
サブカテゴリ #2
2. 基線のパラメータ (SR= 25)
基線の線種
基線間の距離
3. 基線の始点 (SR= 3)
4. 基線の終点 (SR= 4)

面の肌の記号 レコード構造
(1) 面の肌の記号レコード
1. カテゴリ (SR= 1)
メインカテゴリ #10
サブカテゴリ #1
2. マーク (SR= 34)
面の肌に加工方法が付くとき、面の肌の記号に水平線が付く。
3. マークに付く線の始点 (SR= 3)
4. マークに付く線の終点 (SR= 4)

(2) 面の肌の仕様レコード
このレコードは必要に応じて付ける。サブカテゴリの番号によって平均粗さ、カットオフ値などの区別がなされる。したがって同じサブカテゴリのレコードが２つ以上あってはならない。

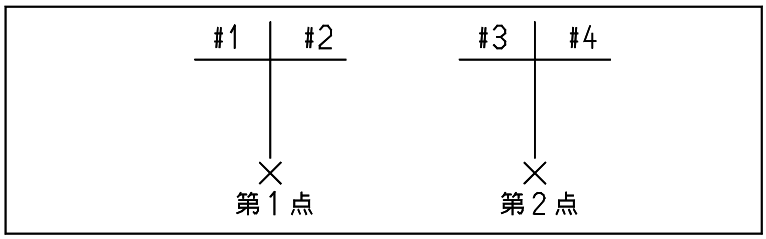
1. カテゴリ (SR= 1)
   メインカテゴリ #10
   サブカテゴリ #11 ～ #21
2. テキスト (SR= 33)

(3) 引出線レコード（オプショナル）
1. カテゴリ (SR= 1)
   メインカテゴリ #2
   サブカテゴリ #1
2. 引出線の始点 (SR= 3)
3. 引出線の最初の線分 (SR= 4)
   :
m+1. 引出線の最後の線分 (SR= 4)
m+2. 矢のマーク (SR= 34)
m は引出線の点数。32 点以下。

4.2.3 アイテム１３ 寸法

レコードの説明
ひとつのレコードは、先頭がカテゴリサブレコード (SR=1) で始まり、次のカテゴリ サブレコードの直前まで、または End of Item (SR=31) までである。

寸法線レコード
1. カテゴリ サブレコード (SR= 1)
   メインカテゴリ #1
   サブカテゴリ #1 ～ #4
   サブカテゴリは寸法線の番号を表わす。



2. 寸法線始点 (SR= 3)
3. 寸法線終点または円弧 (SR= 4 または 5)
4. 寸法線の矢 (SR= 34)

寸法線の矢は、なくてもよい。
各寸法線は一直線上にあること。
角度寸法，円弧長寸法のときは #2, #3 の寸法線は円弧になる。

寸法補助線レコード
1. カテゴリ サブレコード (SR= 1)
   メインカテゴリ #5
   サブカテゴリ #1, #2 １番目か２番目かを示す。
2. 寸法補助線 始点 (SR= 3)
3. 寸法補助線 終点 (SR= 4)

寸法値 レコード
1. カテゴリ サブレコード (SR= 1)
   メインカテゴリ #11

サブカテゴリ　#0
2. 寸法値　(SR= 33)
64 文字 (バイト) 以内

公差値レコード
1. カテゴリサブレコード　(SR=1)
メインカテゴリ　#12
サブレコード　#1　上下の寸法許容差
#2　上の寸法許容差
#3　下の寸法許容差
2. 公差値　(SR= 33)
32 文字 (バイト) 以内
上下の寸法許容差を持つときは、上の寸法許容差・下の寸法許容差は持てない。
逆に、上の寸法許容差または下の寸法許容差を持つとき、上下の寸法許容差は持てない。

付加文字列レコード
1. カテゴリサブレコード　(SR=1)
メインカテゴリ #11
サブカテゴリ　#1
2. 付加文字列　(SR= 33)
64 文字 (バイト) 以内

引出線レコード
1. カテゴリ サブレコード　(SR= 1)
メインカテゴリ　#2
サブカテゴリ　#1
2. 引出線始点　(SR= 3)
3. 引出線の点　(SR= 4)
:
m+1. 引出線の点　(SR= 4)
m+2. 矢印マーク　(SR=34)
m は引出線の点数。32 点以下。

長さ寸法 (Linear Dimension)

レコード構造
1. 寸法パラメータ　(SR= 37)
2. 第１寸法線レコード（オプショナル）
3. 第２寸法線レコード（オプショナル）
4. 第３寸法線レコード（オプショナル）
5. 第４寸法線レコード（オプショナル）
6. 第１寸法補助線レコード（オプショナル）
7. 第２寸法補助線レコード（オプショナル）
8. 寸法値 レコード
9. 公差値 レコード（オプショナル）
10. 付加文字列レコード（オプショナル）
11. End of Item　(SR= 31)

オプショナルのレコードはなくてもよい。

注意
レコードについては前記を参照のこと。
寸法補助線は互いに平行であること。

片寄せ寸法
第２寸法線レコードと第１寸法補助線レコードまたは

第１寸法線レコードと第１寸法補助線レコードを使用。

オーディネイト寸法
第２寸法補助線レコードだけを使用。

累進寸法
第３寸法線レコードと第２寸法線レコードだけを使用。

角度寸法 (Angular dimension)

レコード構造
1. 寸法パラメータ (SR= 37)
2. 第１寸法線レコード（オプショナル）
3. 第２寸法線レコード（オプショナル）
4. 第３寸法線レコード（オプショナル）
5. 第４寸法線レコード（オプショナル）
6. 第１寸法補助線 レコード（オプショナル）
7. 第２寸法補助線 レコード（オプショナル）
8. 寸法値 レコード
9. 公差値 レコード（オプショナル）
10. 付加文字列レコード（オプショナル）
11. End of Item (SR= 31)

オプショナルのレコードはなくてもよい。

注意
レコードについては前記を参照のこと。

半径寸法 (Radius dimension)

レコード構造
1. 寸法パラメータ (SR= 37)
2. 引出線レコード
3. 反対側の引出線レコード（オプショナル）
反対側の引出線は、寸法値テキストが付く引出線の矢印の反対側に付ける線。
    1. カテゴリサブレコード (SR= 1)
    メインカテゴリ #7
    サブカテゴリ #0
    2. 引出線の始点 (SR= 3)
    3. 引出線の終点 (SR= 4)
4. 寸法値レコード
5. 公差値レコード（オプショナル）
6. 付加文字列レコード（オプショナル）
7. End of item (SR= 31)

注意
レコードについては前記を参照のこと。

直径寸法 (diameter dimension)

レコード構造
1. 寸法パラメータ (SR= 37)
2. 第１寸法線レコード（オプショナル）
3. 第２寸法線レコード（オプショナル）
4. 第３寸法線レコード（オプショナル）
5. 寸法値 レコード

6. 公差値 レコード（オプショナル）
7. 付加文字列レコード（オプショナル）
8. End of Item (SR= 31)

オプショナルのレコードはなくてもよい。

注意
レコードについては前記を参照のこと。
寸法テキストが円の外側に出るとき、外側に出る寸法線は常に第１寸法線レコードとする。

円弧長寸法 (Arclength dimension)

レコード構造
1. 寸法パラメータ (SR= 37)
2. 第１寸法線レコード（オプショナル）
3. 第２寸法線レコード（オプショナル）
4. 第３寸法線レコード（オプショナル）
5. 第４寸法線レコード（オプショナル）
6. 第１寸法補助線レコード（オプショナル）
7. 第２寸法補助線レコード（オプショナル）
8. 寸法値 レコード
9. 公差値 レコード（オプショナル）
10. 付加文字列レコード（オプショナル）
11. End of Item (SR= 31)

オプショナルのレコードはなくてもよい。

チャンファ寸法 (Chamfer dimension)

レコード構造
1. 寸法パラメータ (SR= 37)
2. 第１寸法補助線（オプショナル）
3. 引出線レコード
4. 寸法値レコード
5. 公差値レコード（オプショナル）
6. 付加文字列レコード（オプショナル）
7. End of Item (SR= 31)

オプショナルのレコードはなくてもよい。

注意
第１寸法補助線は、引出線の矢が面取り線の外に出るときに付ける。

4.2.4 アイテム１４ 幾何公差 (Geometrical tolerance)

レコード構造
1. カテゴリ サブレコード (SR=1)
   メイン カテゴリ #8
   サブ カテゴリ 幾何公差データ行数 (m)
2. 幾何公差パラメータ (SR=17)
3. 幾何公差データレコード
   幾何公差データ１行につき、以下のレコードがある。
   幾何公差記号レコード
   精度テキストレコード

データム文字レコード
幾何公差記入枠線レコード（複数）
4. 引出線レコード（オプショナル、2 つまで）

レコードの説明
幾何公差記号レコード
1. カテゴリレコード (SR=1)
メインカテゴリ #14
サブカテゴリ 行番号 (1 - m)
2. 幾何公差マーク (SR=34)

精度テキストレコード
1. カテゴリレコード (SR=1)
メインカテゴリ #15
サブカテゴリ 行番号 (1 - m)
2. 精度テキスト (SR=33)

データム文字レコード
1. カテゴリレコード (SR=1)
メインカテゴリ #16
サブカテゴリ 行番号 (1 - m)
2. データム文字 (SR=33)

幾何公差記入枠線レコード
1. カテゴリレコード (SR=1)
メインカテゴリ #17
サブカテゴリ 行番号 (1 - m)
2. サブレコード #3, #4 で構成する線分群

引出線レコード
1. カテゴリ サブレコード (SR= 1)
メインカテゴリ #2
サブカテゴリ #1
2. 引出線始点 (SR= 3)
3. 引出線の点 (SR= 4)
:
m+1. 引出線の点 (SR= 4)
m+2. 矢印マーク (SR=34)
m は引出線の点数。32 点以下。

4.2.5 アイテム１５ ハッチング

アイテムの定義
閉領域内を一定間隔の平行線分でおおう。

サブレコード構造 (バージョン 9 以降)
1. ハッチパラメータ (SR= 36)
2. 境界 #1
:
n+1. 境界 #n
n+2. End of item (SR31)
境界はストリングアイテムと同じで SR3,[SR4 or SR5 or SR6]+ の並び。

サブレコード構造 (バージョン 8 以下。使用してはなりません。参考のみ)
1. ハッチングラインの始点 (SR= 3)

2. ハッチングラインの終点 (SR= 4)
:
2m-1. ハッチングラインの始点 (SR= 3)
2m. ハッチングラインの終点 (SR= 4)
2m+1. End of item (SR=31)
m はハッチングライン数

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X)
Ymin=AMIN（各サブレコードの最小 Y)
Xmax=AMAX（各サブレコードの最大 X)
Ymax=AMAX（各サブレコードの最大 Y)

4.2.6 アイテム１６ 塗り潰しアイテム

アイテムの定義
閉領域内を指定パターンでおおう。

サブレコード構造
1. 塗り潰しパラメータ (SR=35)
2. 境界 #1
:
n+1. 境界 #n
n+2. End of Item (SR=31)

注意
(1) 境界は次のサブレコードの並びである。
1. 始点 (SR3)
2. 線分、円弧または３次 Bezier カーブセグメント (SR4, SR5 または SR6)
:
n+1. 線分、円弧または３次 Bezier カーブセグメント (SR4, SR5 または SR6)
(2) 境界は閉じていること（始点と終点が一致すること)。
(3) 境界は複数持つことができる。

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X)
Ymin=AMIN（各サブレコードの最小 Y)
Xmax=AMAX（各サブレコードの最大 X)
Ymax=AMAX（各サブレコードの最大 Y)

4.3 その他

4.3.1 アイテム２７ メンバーアイテム

アイテムの定義
同時設計で使用するサブモデルのこと。本来のサブモデルと区別するために、アイテムタイプ番号が異なる。
このアイテムは Advance CAD 実行時のみ存在し、保存されることはない。

4.3.2 アイテム２８ ＡＰＧアイテム

アイテムの定義
APG 配置時に作成される図形アイテムの集合である。APG 配置時の APG 名称、APG パラメータ、配置条件などを持っている。APGREGEN で再作成できる。

サブレコード構造

1. ヘッダーブロック
   - カテゴリレコード 250/1 (SR=1)
   - APG 名称 (SR=26)
   - APG 配置条件 (SR=28)
     - 原点、ボックス、ミラー条件、配置角度

2. APG パラメータブロック
   - カテゴリレコード 250/2 (SR=1)
   - APG パラメータ値 A=105 (SR=26)
   - :

3. 図形データブロック
   - カテゴリレコード 251 (SR=1)
   - [もとのアイテムの属性 (SR=32)
     もとのアイテムのサブレコード
   - :
   - [もとのアイテムの属性 (SR=32)
     もとのアイテムのサブレコード
   - カテゴリレコード 252 (SR=1)

4. End of item (SR=31)

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X)
Ymin=AMIN（各サブレコードの最小 Y)
Xmax=AMAX（各サブレコードの最大 X)
Ymax=AMAX（各サブレコードの最大 Y)

4.3.3 アイテム２９ アソシエイト アイテム

アイテムの定義
アソシエイトアイテム名と、関係するアイテムの ID ポインタ番号の集合体からなるアイテム。

サブレコード構造

1. ヘッダーブロック
   - カテゴリレコード 61/n (SR=1)
     - n: アソシエイトカテゴリ番号
   - アソシエイト アイテム名 (SR=26)
   - カテゴリ 62/1 (SR=1) （オプショナル）
   - シンボル名 (SR=26)（オプショナル）
   - カテゴリ 62/2 (SR=1) （オプショナル）

   - X,Y,Z,ANGLE,PIC (SR=25)（オプショナル）
     - X,Y,Z : 原点
     - ANGLE : 配置角度 (0,90,180,270)
     - PIC : 配置ピクチャ

2. アソシエイト ブロック
   アソシエイト type , IDPTR1 (SR=29)
   アソシエイト type , IDPTR2 (SR=29)
   :
   アソシエイト type , IDPTRn (SR=29)

3. End of Item (SR=31)

オプショナルのレコードはなくてもよい。

4.3.4 アイテム３０ シンボル アイテム

アイテムの定義
属性データ (シンボル名・ベーステキスト・ノードポイント・ノードテキストなど) と任意の図形データの集合体からなるアイテム。

サブレコード構造
1. ヘッダー ブロック
   Category ンボルヘッダー カテゴリ
   Non-graphic text シンボル名
   System date/revision 入力日時、シンボルバージョン
   Symbol parameter シンボルパラメータ (原点、ボックス、縮尺値、回転角度)
2. ベーステキスト ブロック
   Category ベーステキスト カテゴリ
   Text ベーステキスト
3. ノードポイント ブロック
   Category ノードポイント カテゴリ／ノード数 (n)
   Point ノードポイント #1
   Point ノードポイント #2
   :
   Point ノードポイント #n
4. ノードテキスト ブロック
   Category ノードテキスト カテゴリ／ノード番号
   Text ノードテキスト
   :
   Category ノードテキスト カテゴリ／ノード番号
   Text ノードテキスト

5. 図形データ ブロック
   このブロックは、表示図形を構成するサブアイテムを含む。
   Category シンボルアイテム図形データ開始カテゴリ
   Item attribute サブアイテムの属性 (SR= 32)
   Sub-record サブアイテムのサブレコード
   :
   Sub-record サブアイテムのサブレコード
   :
   Item attribute サブアイテムの属性 (SR= 32)
   Sub-record サブアイテムのサブレコード
   :
   Sub-record サブアイテムのサブレコード
   :
   Item attribute サブアイテムの属性 (SR= 32)
   Sub-record サブアイテムのサブレコード
   :
   Sub-record サブアイテムのサブレコード

```
    Category                シンボルアイテム図形データ終了カテゴリ
6.  End of item  (SR= 31)
```

注意

(1) ベーステキスト ブロック、ノードポイント ブロックまたはノードテキストブロックのないシンボルもある。
(2) 通常表示されるのは、図形データブロックだけである。ただし、アイデントされたときは、ノードポイントも表示される。
(3) シンボルはネスティングできない。

アイテムの最小／最大座標値

Xmin=AMIN（各サブレコードの最小 X）
Ymin=AMIN（各サブレコードの最小 Y）
Xmax=AMAX（各サブレコードの最大 X）
Ymax=AMAX（各サブレコードの最大 Y）

4.3.5 アイテム３１ サブモデル

アイテムの定義

既存モデルの１つのピクチャ上のアイテムで構成されたアイテム。

サブレコード構造

1. ヘッダー ブロック

```
    Category                サブモデルヘッダー カテゴリ
    Scalar                  ピクチャ番号
    Non-graphic text        サブモデル名
    System date/revision    入力日時、サブモデル バージョン
    Sub-model parameter     サブモデルパラメータ（原点、ボックス、縮尺値、回転角度）
    Sub-model mask          サブモデルのマスクデータ

┌   Category                サブモデルマスクデータ／マスクタイプ
└   Sub-record              マスクデータ (SR= 25)
```

2. アイテムデータ ブロック

このブロックは、サブモデルを構成するサブアイテムを含む。

```
    Category                アイテムデータブロック開始 カテゴリ
┌   Item attribute          サブアイテムの属性      (SR= 32)
│   Sub-record              サブアイテムのサブレコード
│     :                       :
└   Sub-record              サブアイテムのサブレコード


┌   Item attribute          サブアイテムの属性      (SR= 32)
│   Sub-record              サブアイテムのサブレコード
│     :                       :
└   Sub-record              サブアイテムのサブレコード
      :                       :
┌   Item attribute          サブアイテムの属性      (SR= 32)
│   Sub-record              サブアイテムのサブレコード
│     :                       :
└   Sub-record              サブアイテムのサブレコード

    Category                アイテムデータブロック終了 カテゴリ
```

3. End of item (SR= 31)

アイテムの最小／最大座標値
Xmin=AMIN（各サブレコードの最小 X）
Ymin=AMIN（各サブレコードの最小 Y）
Xmax=AMAX（各サブレコードの最大 X）
Ymax=AMAX（各サブレコードの最大 Y）

4.3.6 アイテム３２ イメージアイテム

アイテムの定義
イメージを表現するアイテム。

サブレコード構造
1. Image (SR= 38)
2. End of item (SR= 31)

アイテムの最小／最大座標値
Xmin = Image を含む最小矩形の最小 X
Ymin = Image を含む最小矩形の最小 Y
Xmax = Image を含む最小矩形の最大 X
Ymax = Image を含む最小矩形の最大 Y

注意
このアイテムの先頭は Image サブレコードをおきます。
このアイテムは、ピクチャの背景として配置したイメージとして使用します。背景であるため、通常のコマンドではピックできません。イメージアイテムの配置、修正、削除のコマンドがあります。
ユーザプログラミングでは、このアイテムを修正しないで下さい。

第 5 章　サブレコード一覧

下記の表はサブレコードタイプの一覧です。Version18 では、SXF の仕様を取り込み、サブレコードタイプの見直しが行われました。その結果、いくつかのサブレコードタイプが廃止となり、新しいサブレコードタイプが追加されました。**Version 17 以下で作成されたモデルを読み込むと、自動的に適切な新しいサブレコードに変換されます。**(-V17) と記したサブレコードは廃止されたサブレコードタイプで、アイテムに追加できません。
パラメータ名はヘッダーファイル acadprm.h 内で定義されています。本書中の関数でサブレコードタイプを指示するものがあります。この場合は直接サブレコード番号を記入するよりはパラメータ名を記入した方がわかりやすいでしょう。

サブレコード一覧表

タイプ番号	サブレコード	パラメータ名	区分
1	分類	SITCATEG	汎用
2	点	SITPOINT	幾何
3	始点	SITSTART	幾何
4	線分	SITLINE	幾何
5	円、円弧	SITARC	幾何
6	3 次 Bezier 曲線	SITBZCRV	幾何
7 － 10	（未使用）		
11	テキストパラメータ (-V17)		廃止
12	テキスト／マーク 原点枠 (-V17)		廃止
13	文字列 (-V17)		廃止
14	マークパラメータ (-V17)		廃止
15	寸法アイテム種類 (-V17)		廃止
16	寸法アイテム基準データ (-V17)		廃止
17	幾何公差パラメータ	SITFCSPRM	製図
18	塗り潰しパラメータ (-V17)		廃止
19	ハッチングパラメータ (-V17)		廃止
20	特性データ レコード番号	SITSPEC	汎用
21	NC マシニングレコード	SITPPW	NC
22	3D Position (x,y,z)	SIT3DPOS	NC

タイプ番号	サブレコード	パラメータ名	区分
23	(system use -V18)		廃止
24	(system use -V18)		廃止
25	スカラ列	SITSCALAR	汎用
26	非図形文字列	SITNGTEXT	汎用
27	時刻、レビジョン番号	SITDATERV	構造化
28	シンボル / サブモデル /APG パラメータ	SITSSMPRM	構造化
29	アソシエイティビティ	SITASSC	構造化
30	元のアイテム属性 (-V17)		廃止
31	End of item	SITEOI	汎用
32	元のアイテム属性 (V18-)	SITITMATR2	構造化
33	文字列 (V18-)	SITTXTPRM2	製図
34	マーク (V18-)	SITMRKPRM2	製図
35	塗り潰しパラメータ (V18-)	SITAFLPRM2	製図
36	ハッチングパラメータ (V18-)	SITXHTPRM2	製図
37	寸法パラメータ (V18-)	SITDIMPRM2	製図
38	イメージ (V19-)	SITIMAGE	SXF
39	作図部品定義 (V18-)	SITSFIGORG	SXF
40	作図部品配置 (V18-)	SITSFIGLOC	SXF
41	作図部品配置終了 (V18-)	SITSFIGEND	SXF
42	AFL/XHT 境界表示制御 (V18-)	SITBNDATTR	SXF
43	元図定義 (V18-)	SITORGDEF	SXF
44	元図参照 (V18-)	SITORGREF	SXF

5.1 汎用

5.1.1 サブレコード 1 分類

【 目的 】

このサブレコードに続くサブレコード群の意味を示す。
通常サブレコード #1 からつぎのサブレコード #1 の直前までがひとくくりとなり、1 つのレコードを構成する。

SR 1

```
MODE    1
COUNT   1
```

【 内容 】

1.1　　主分類コード　(1 - 255)　8 bits (1 - 8)
1.2　　副分類コード　(0 - 255)　8 bits (9 - 16)

以下に、使用している主分類コードと副分類コードの一覧表を記述する。

```
1 = 寸法線 (Dimension)
      1 = 第１寸法線
      2 = 第２寸法線
      3 = 第３寸法線
      4 = 第４寸法線
2 = 引出線 (Drafting)
      切断線のライン (Cutting plane line)
3 = グラフィックステキスト (Drafting)
4 = マーク (Gen-mark)
5 = 寸法補助線 (Dimension)
      1 = 第１寸法補助線
      2 = 第２寸法補助線
6 = 風船 (Gen-text)
      0 = 風船用テキスト
      1 = 風船用マーク
7 = 半径寸法の円中心側引出線 (DMR dimension)
8 = 幾何公差 (Fcs)
      許容値の行数
9 = 溶接記号 (Weldmark)
      1 = 矢の手前側溶接記号のマーク
      2 = 矢の向う側溶接記号のマーク
      3 = 全周／現場溶接記号のマーク
      4 = 溶接部の表面形状マーク（手前側）
      5 = 溶接部の表面形状マーク（向う側）
      11 = 溶接部断面寸法または強さ（手前側）
      12 = ルート開先，開先角度（手前側）
      13 = 溶接長さ，数，ピッチ（手前側）
      14 = 溶接部の仕上方法（手前側）
      21 = 溶接部断面寸法または強さ（向う側）
      22 = ルート開先，開先角度（向う側）
      23 = 溶接長さ，数，ピッチ（向う側）
      24 = 溶接部の仕上方法（向う側）
      31 = 特記事項
10 = 面の肌記号
      1 = 面の指示記号のマーク
      11 = 中心線平均粗さの値 (Ra)
      12 = カットオフ値
      13 = 最大長さ (Rmax)＋点平均粗さ (Rz)
      14 = 基準長さ
      15 = 加工方法
      16 = 筋目方向
      17 = 仕上げ代／削り代
      18 = 表面うねり仕様
      19 = 表面性状パラメータ
      20 = ２番目のパラメータ
      21 = ３番目のパラメータ
11 = 寸法値 (Dimension)
      0 = 寸法値テキスト
      1 = 寸法付加テキスト
12 = 寸法公差値 (Dimension)
```

1 = 上下の寸法許容差
2 = 上の寸法許容差
3 = 下の寸法許容差
14 = 幾何公差の記号 (Fcs)
15 = 幾何公差の精度 (Fcs)
16 = 幾何公差のデータム (Fcs)
17 = 幾何公差記入枠 (Fcs)
18 = 切断線のテキスト (Cutting plane line)
1 = 始点側
2 = 終点側
19 = 切断線のマーク (Cutting plane line)
1 = 始点側
2 = 終点側
20 = 切断線 (Cutting plane line)
21 = テキスト枠の線 (Drafting)
22 = テキスト下線 (Drafting)
n = テキスト下線の数
31 = NC Drill
40 = NC EDM
51 = 特性データカテゴリ番号
n = カテゴリ番号
52 = 特性データタイプ
1 = GNT
2 = 風船
3 = TAG
61 = アソシエイトアイテムの名前とアソシエイト分類番号
n = アソシエイト分類番号
62 = アソシエイト属性
1 = サブモデル名
2 = 配置原点
63 = CUT_MOVE, CUT_REGEN 用オリジナルデータ。
直後の サブレコード２５ に元の点座標 (x,y)、移動先点座標 (x,y), スケール値、出力先ピクチャ番号および境界上のアイテムの処理方法の 7 word を持つ。
201 = Mes 面積データ
230 = シンボルヘッダ (Symbol)
231 = シンボルベーステキスト (Symbol)
232 = シンボルノード点 (Symbol)
233 = シンボル ノードテキスト #1（ノード Name テキスト）
234 = シンボル ノードテキスト #2（コネクトノード プレイスメント）
235 = シンボル ノードテキスト #3（コネクトノード シーケンス番号）
238 = シンボルデータの開始 (Symbol)
239 = シンボルデータの終了 (Symbol)
240 = サブモデルヘッダ (Sub-model)
bit 9 : 寸法要素再作成フラグ
bit 10 : 製図要素縮尺フラグ
bit 11 : ピクチャ参照フラグ
247 = サブモデルのマスクデータ (Sub-model)
1 = アイテムマスク
2 = クラスマスク
3 = レビジョンマスク
4 = 線種マスク
5 = 線幅マスク
248 = サブモデルデータの開始 (Sub-model)
249 = サブモデルデータの終了 (Sub-model)
250 = APG ヘッダ
1 = APG ファイル名および配置パラメータ
2 = APG 変数の値
251 = APG データの開始

252 = APG データの終了

5.1.2 サブレコード２ 点

【 目的 】

点

```
SR      2
MODE    3
COUNT   2
```

【 内容 】

1. X 座標
2. Y 座標

【補則 】

Version 18 よりデータタイプは倍精度実数 (MODE=3) のみを使用し、単精度実数 (MODE=2) は使用しない。

5.1.3 サブレコード３ 始点

【 目的 】

線分、円／円弧、３次 Bezier 曲線の始点。

```
SR      3
MODE    3
COUNT   2
```

【 内容 】

1. カーブ始点 X 座標
2. カーブ始点 Y 座標

【 補則 】

Version 18 よりデータタイプは倍精度実数 (MODE=3) のみを使用し、単精度実数 (MODE=2) は使用しない。

5.1.4　サブレコード４　線分

【 目的 】

線分の定義。

```
SR      4
MODE    3
COUNT   2
```

【 内容 】

1.　　線分終点 X 座標
2.　　線分終点 Y 座標

【 補則 】

(1)　線分の終点座標のみを持つ。
(2)　線分の始点は直前の**幾何図形サブレコード** (SR=3, 4, 5 または 6) の終点。
(3)　Version 18 よりデータタイプは倍精度実数 (MODE=3) のみを使用し、単精度実数 (MODE=2) は使用しない。

5.1.5　サブレコード５　円／円弧

【 目的 】

円弧セグメント

```
SR      5
MODE    3
COUNT   8
```

【 内容 】

1.　　中間点 X 座標
2.　　中間点 Y 座標
3.　　終点 X 座標
4.　　終点 Y 座標
5.　　中心点 X 座標
6.　　中心点 Y 座標
7.　　半径
8.　　中心角 (Radian, -2π から 2π)
　　正　反時計廻り (CCW)
　　負　時計廻り (CW)

【 補則 】

(1) 円弧の始点は 直前の幾何図形サブレコード (SR=3,4,5 または 6) の終点。
(2) 円は円弧の特殊解として処理する。
中心角 = 2π (CCW), -2π (CW)
円始点 = 円終点
(3) 中間点は必ず弧の２分点になければならない。
(4) Version 18 よりデータタイプは倍精度実数 (MODE=3) のみを使用し、単精度実数 (MODE=2) は使用しない。

5.1.6 サブレコード６ 3 次 Bezier 曲線

【 目的 】

3 次 Bezier 曲線

```
SR      6
MODE    3
COUNT   7
```

【 内容 】

1. 制御点 (Q2) の X 座標
2. 制御点 (Q2) の Y 座標
3. 制御点 (Q3) の X 座標
4. 制御点 (Q3) の Y 座標
5. 終点 (Q4) の X 座標
6. 終点 (Q4) の Y 座標
7. 曲線長さ

【 補則 】

(1) 始点 (Q1) は直前の幾何図形サブレコード (SR=3,4,5 または 6) の終点を使用する。
(2) 制御点列 Q1,Q2, Q3, Q4 として、３次 Bezier カーブ点 P(t) は次式のとおり。
P(t) = (1-t)^3*Q1 + 3t(1-t)^2*Q2 + 3t^2(1-t)*Q3 + t^3*Q4
0 <= t <= 1
(3) Version 18 よりデータタイプは倍精度実数 (MODE=3) のみを使用し、単精度実数 (MODE=2) は使用しない。

5.2 製図用

5.2.1 サブレコード 33 文字列

【 目的 】

グラフィックステキストのパラメータ、位置、文字列を持ちます。Version 17 までのサブレコードの順序列（SR=11, SR=12, SR=13) に代わる新しいサブレコードです。サブレコードのデータは整数、倍精度実数、文字列の三つのプリミティーフデータで表現します。データタイプは５（**複合型**）**とし、データ数は 2BYTES 整数を単位として数えます。**倍精度実数は 4、文字は 2 文字で 1 と数えます。

```
SR      33
MODE     5
COUNT   60 - 1083
```

【 内容 】

1.

1.1 テキストの表示モード　1 bit (1)
0= 横書き、1= 縦書き

1.2 X 座標反転 (X-mirror)　1 bit (2)
0= しない、1= する（テキストの Y 軸に対して反転する）

1.3 Y 座標反転 (Y-mirror)　1 bit (3)
0= しない、1= する（テキストの X 軸に対して反転する）

1.4 枠 / 下線表示　3 bits(4- 6)
0= なし、1= 枠、2= 下線、3= 枠と下線、4= 可変長下線

1.5 文字列表示水平基準　2 bits(7- 8)
0= 左詰め、1= 中央、2= 右詰め

1.6 水平方向文字基準　2 bits(9-10)
0= 左、1= 中央、2= 右

1.7 垂直方向文字基準　2 bits(11-12)
0= 下、1= 中央、2= 上

1.8 文字フォントタイプ　2 bits(13-14)
0= 横書きフォント、1= 縦書きフォント

1.9 未使用　2 bits(15-16)

2.

2.1 日本語テキストフォント番号　8 bits(1- 8)
101 = 日本語ストロークフォント
102-109 = アウトラインフォント
111-130 = トゥルータイプフォント (OS 依存、OS 間の互換性なし)

2.2 ASCII テキストフォント番号　8 bits(9-16)
1-99 = 英数字ストロークフォント
102-109 = アウトラインフォントの英数字部分を使う
111-130 = トゥルータイプフォントの英数字部分を使う (OS 依存、OS 間の互換性なし)

3.

3.1 行幅整列係数　6 bits(1- 6)
内部表現 : 0 ～ 60（外部表現 : 0 ～ 100)。
行幅整列係数は、0% ～ 100% の設定値を 6bits で表現するために、下表のように 0 ～ 50 までが 1 刻み、55 ～ 100 までを 5 刻みとする

設定値 (%)	内部表現
0 ～ 50	0 ～ 50
55	51
60	52
65	53
70	54

設定値(%)	内部表現
75	55
80	56
85	57
90	58
95	59
100	60

3.2 未使用 10 bits(7-16)

4. 水平方向ゆとり幅 (0 - 32767 の整数)
5. 垂直方向ゆとり幅 (0 - 32767 の整数)
ドローイングレイアウトスペースでのゆとり幅 (0.0 ～ 327.67) X 100 で整数化した値。
例 0.1mm → 10
10mm → 1000
0.15 インチ → 15
6. 文字傾き角 (Slant_Angle)(-8500 - 8500 の整数)
実際の角度 (-85.00 ～ 85.00) X 100 で符号付整数化した値。

7. 文字列長さ (1 - 2048 未満)
文字列長さはバイト数で数える。文字数ではないので注意。

8-11. 文字高さ H (倍精度実数) (0.01 - 327.67)
ドローイングレイアウトスペースでの文字高さ。
モデルスペースでの文字高さはピクチャスケールとドローイングスケールを反映させて決まる。ピクチャスケール (部分拡大図倍率) PSF=2.0、ドローイングスケール (図面縮尺) DSF=1/5、文字高さ h=4.0 とすれば、モデルスペースでの文字高さは 10.0 になる。
h / (DSF*PSF) = 4.0 / (0.2*2.0) = 10.0

12-15. 文字列角度 (倍精度実数) (0.0-360.0 度)
16-19. 文字縦横比 (倍精度実数) (0.1-10.0)
文字高さを基準とした比率 (= 幅 / 高さ)。
20-23. 文字間隔 (倍精度実数) (0.0-10.0)
文字高さを基準とした比率 (= 文字の間隔 / 高さ)。
24-27. 行間隔 (倍精度実数) (0.0-10.0)
文字高さを基準とした比率 (= 行の間隔 / 高さ)。

28-31. 最初の文字の位置 P0 の X 座標 (倍精度実数)
32-35. 最初の文字の位置 P0 の Y 座標 (倍精度実数)
36-39. 枠の左下隅 P1 の X 座標 (倍精度実数)
40-43. 枠の左下隅 P1 の Y 座標 (倍精度実数)
44-47. 枠の右下隅 P2 の X 座標 (倍精度実数)
48-51. 枠の右下隅 P2 の Y 座標 (倍精度実数)
52-55. 枠の左上隅 P3 の X 座標 (倍精度実数)
56-59. 枠の左上隅 P3 の Y 座標 (倍精度実数)
(1) すべてモデルスペース座標 (文字高さの項を参照のこと)。
(2) 最初に入力された文字列位置 Porg は文字原点基準に依存する。
水平方向文字基準 JH = 0 左 1 中央 2 右
垂直方向文字基準 JV = 0 下 1 中央 2 上
としたとき、次の式で逆算して求める。
文字列位置は矩形の四隅、矩形の四辺の中点、矩形の中心点のどれかひとつである。
Porg = P1 + 0.5 * (JH*(P2-P1) + JV*(P3-P1))

60-. 文字列

(1) 文字コードは EUC-JIS ("EUC-JP")。文字列長さはバイト数 2048 未満

(2) 改行文字

ASCII コード (0x0D) が行の区切り。

(3) 文字の種類　　ビットパターン

ASCII 文字　1 バイト

バイト　0x20 ～ 0x7E　　(#0xxxxxxx)

空白および"!" から " ～ " までの印字可能文字のみ。

メタ文字　2 バイト

1 バイト目 (B1) 0x8D　　(#10001101)

2 バイト目 (B2) 0xA1 ～ 0xFE　　(#1xxxxxxx)

1 バイト目は、つぎの 1 バイトがメタ文字であることを示すシングルシフトである。2 バイト目の第１ビットを除いた下 7 ビットは ASCII 文字の "!" から " ～ " に対応する。ASCII は 7 ビットであるがこれを 8 ビットまで拡張し、拡張した部分をメタ文字と呼んでいる。通常の ASCII 文字と区別するため､"¥M" をつけて ¥MA, ¥MB などと記述している｡¥MA は ASCII 文字 "A" に対応するメタ文字である。

日本語文字 (JIS X0208)　2 バイト

1 バイト目 (B1) 0xA1 ～ 0xFE　　(#1xxxxxxx)

2 バイト目 (B2) 0xA1 ～ 0xFE　　(#1xxxxxxx)

JIS 区点番号との対応はつぎの通り。

JIS 区 = B1 - 160 = (1 ～ 94)

JIS 点 = B2 - 160 = (1 ～ 94)

日本語文字 (JIS X0201)　2 バイト

1 バイト目 (B1) 0x8E　　(#10001111)

2 バイト目 (B2) 0xA1 ～ 0xFE　　(#1xxxxxxx)

1 バイト目は、JIS X0201 を示すシングルシフトである。

俗にいう半角カタカナです。SXF 互換のため追加しましたが、それ以外での使用はお勧めしません。

(4) メタ文字の用途 - ASCII テキストフォントの製図用文字

度記号	¥M0
径記号	¥M1
プラスマイナス	¥M2
角記号	¥M3
弧記号	¥M4

(5) メタ文字の用途 - 特殊な表記

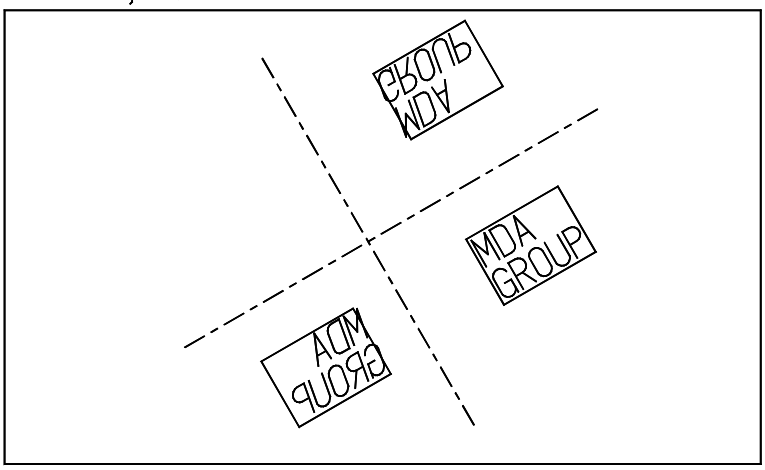
寸法値	¥MD 文字列 ¥MZ
分数	¥MF 分子 ¥MY 分母 ¥MZ
２段文字	¥MS 上段文字列 ¥MY 下段文字列 ¥MZ ¥MS 上段文字列 ¥MZ ¥MS ¥MY 下段文字列 ¥MZ 区切り文字として、¥MY（左寄せ）以外に ¥MX（中央寄せ）、¥MW（右寄せ）がある。
マーク挿入	¥MM マーク番号, マーク番号,……, マーク番号 ¥MZ

(6) 上記以外のメタ文字は使用できない。

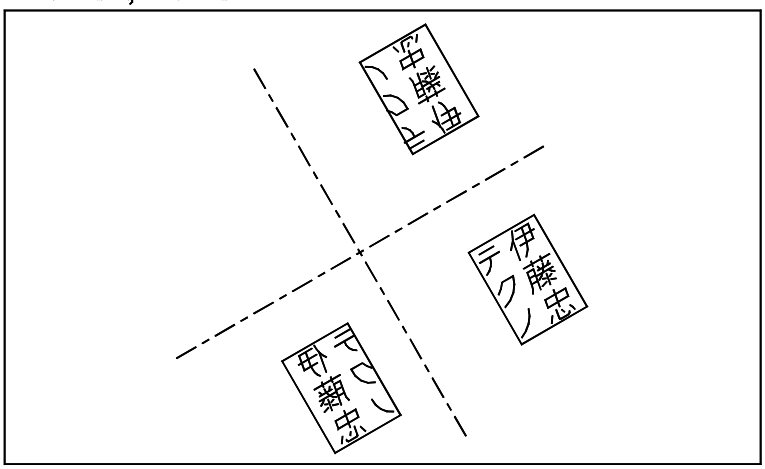
【補則】

(1) テキストの mirror は、先に text angle をかけた状態に対して行われる。
X-mirror はテキストの Y 軸に対して反転する。つまり X 座標の符号が反対になる。
IGES などでいう Y-mirror のこと。Y-mirror も同様に IGES などでいう X-mirror のこと。

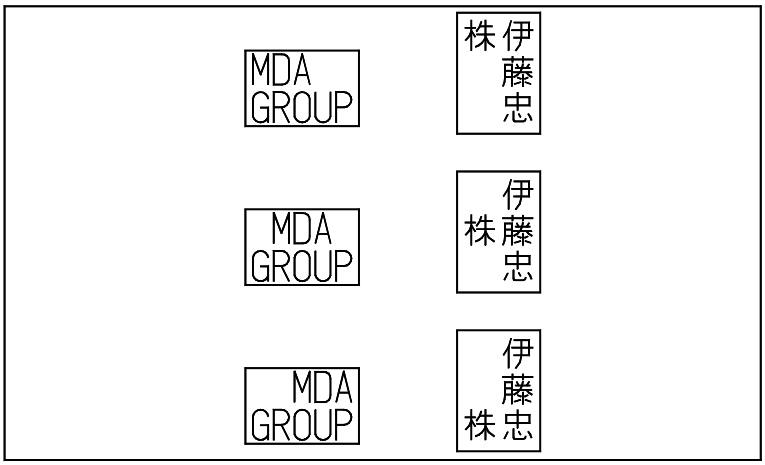
Xmirror,Ymirror



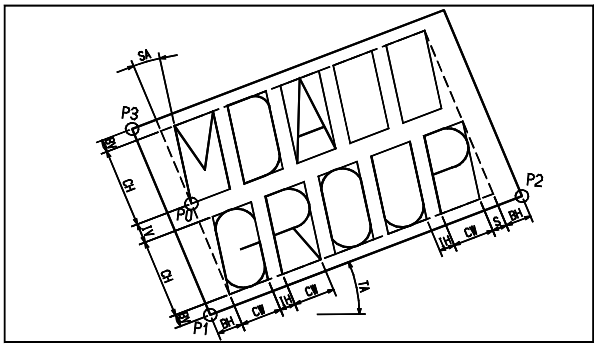
Xmirror,Ymirror



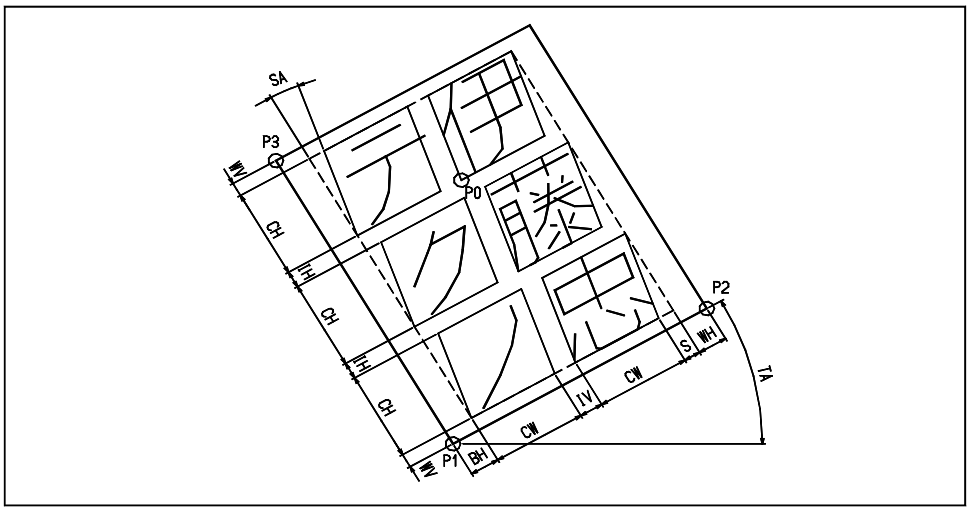
文字列表示基準　左づめ・中央・右づめ



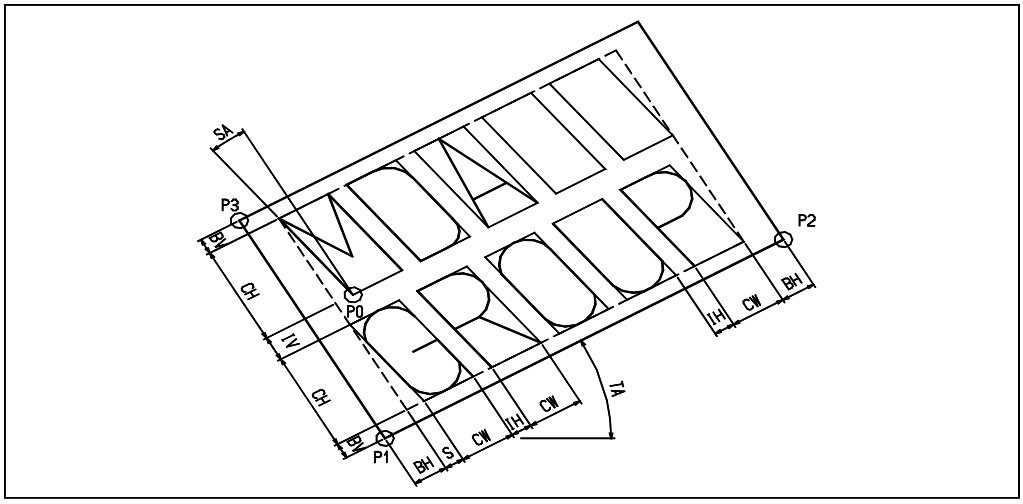
横書き〔文字傾斜角が正〕



縦書き〔文字傾斜角が正〕



横書き〔文字傾斜角が負〕



5.2.2　サブレコード 34　マーク

【 目的 】

マークのパラメータ、位置を持ちます。Version 17 までのサブレコードの順序列（SR=14, SR=12) に代わる新しいサブレコードです。サブレコードのデータは整数、倍精度実数の二つのプリミティーフデータで表現します。データタイプは５ (複合型 ）とし、データ数は 2BYTES 整数を単位として数えます。倍精度実数は 4 と数えます。

```
SR      34
MODE     5
COUNT   41
```

【 内容 】

```
1.
1.1  未使用                                          1 bit ( 1   )
1.2  X 座標反転 (X-mirror)                           1 bit ( 2   )
     0= しない、1= する（マークの Y 軸に対して反転する）
1.3  Y 座標反転 (Y-mirror)                           1 bit ( 3   )
     0= しない、1= する（マークの X 軸に対して反転する）
1.4  未使用                                          1 bit ( 4   )
1.5  マーク番号 (0-4095)                             12bits( 5-16)

2-5. マークサイズ（倍精度実数）
     ドローイングレイアウトでのマークサイズ。
     モデルスペースでのマークサイズはピクチャスケールとドローイングスケールを反映させて決まる。
     ピクチャスケール（部分拡大図倍率）PSF=2.0、ドローイングスケール（図面縮尺）DSF=1/5、サイズ
     h=4.0 とすれば、モデルスペースでのサイズは 10.0 になる。
         h / (DSF*PSF) = 4.0 / (0.2*2.0) = 10.0
6-9. マーク配置角（倍精度実数）(0.0-360.0 度）

10-13.  原点 P0 の X 座標（倍精度実数）
14-17.  原点 P0 の Y 座標（倍精度実数）
18-21.  枠の左下隅 P1 の X 座標（倍精度実数）
22-25.  枠の左下隅 P1 の Y 座標（倍精度実数）
26-29.  枠の右下隅 P2 の X 座標（倍精度実数）
30-33.  枠の右下隅 P2 の Y 座標（倍精度実数）
34-37.  枠の左上隅 P3 の X 座標（倍精度実数）
38-41.  枠の左上隅 P3 の Y 座標（倍精度実数）
```

【 補則 】

(1)　座標はすべてモデルスペース（マークサイズの項を参照のこと）。

5.2.3　サブレコード 37　寸法パラメータ

【 目的 】

寸法のパラメータ、寸法参照点、記入位置を持ちます。Version 17 までのサブレコードの順序列 (SR=15, SR=16) に代わる新しいサブレコードです。サブレコードのデータは整数、倍精度実数の二つのプリミティーフデータで表現します。データタイプは５ (複合型) とし、データ数は 2BYTES 整数を単位として数えます。倍精度実数は 4 と数えます。

SR　　37
MODE　　5
COUNT　　17, 25, 41, 45 または 49

【 内容 】

1.
1.1　寸法種類　　4 bits(1 - 4)
　　1 = 長さ寸法 (Linear Dimension)
　　2 = 角度寸法 (Angular Dimension)
　　3 = 半径寸法 (Radius Dimension)
　　4 = 直径寸法 (Diameter Dimension)
　　5 = 座標寸法 (Coordinate Dimension)
　　6 = 円弧長寸法 (Arclength Dimension)
　　7 = 面取り寸法 (Chamfer Dimension)
1.2　副分類コード　　3 bits(5 - 7)
　　0 = 単一寸法 (Single mode)
　　1 = 直列寸法 (Chain Dimension)
　　2 = 並列寸法 (Parallel Dimension)
　　3 = Ordinate Dimension
　　4 = 片寄せ寸法
　　5 = 累進寸法 (Running Dimension)
　　6 = 寸法補助線が放射状の円弧長寸法

分類	副分類
1. 長さ寸法	0,1,2,3,4,5
2. 角度寸法	0,1,2,5
3. 半径寸法	0
4. 直径寸法	0
5. 座標寸法	0
6. 円弧長寸法	0,6
7. 面取り寸法	0

1.3　未使用　　2 bits(8-10)
1.4　製図基準　　2 bits(11-12)
　　0 = 水平／寸法線分断 (ANSI)
　　1 = 平行／寸法線閉 (JIS)
1.5　十進整数部３桁区切り記号　　2 bits(13-14)
　　0 = なし
　　1 = カンマ
　　2 = 空白
　　3 = 点
1.6　十進小数点　　1 bit (15)
　　0 = なし
　　1 = カンマ
1.7　十進小数部うしろのゼロ除去　　1 bit (16)
　　0 = 除去する

1 = 除去しない

2.

2.1 長さ寸法 単寸法／両寸法モード 2 bits (1- 2)

0 = 単一

1 = mm/inch 併記

2 = inch/mm 併記

2.2 長さ寸法 表示形式 2 bits (3- 4)

0 = 十進

1 = feet & inch

2 = feet

3 = inch

2.3 長さ寸法 単位 3 bits (5- 7)

1 = milimeter

2 = centimeter

3 = meter

4 = inch

5 = feet

6 = mils

2.4 長さ寸法 単位記号表示 2 bits (8- 9)

0 = 表示なし

1 = シンボル

2 = 省略文字

2.5 予約 2 bits (10-11)

2.6 未使用 5 bits (12-16)

3.

3.1 長さ寸法 インチ分数の単位 3 bits (1- 3)

長さ寸法の表示形式が十進でないときだけ参照

0 = 1/1

1 = 1/2

2 = 1/4

3 = 1/8

4 = 1/16

5 = 1/32

6 = 1/64

3.2 長さ寸法 十進小数桁数 (1-6) 4 bits (4- 7)

長さ寸法の表示形式が十進のときだけ使用

3.3 半径寸法 寸法補助記号 2 bits (8- 9)

0 = R 前

1 = 後 R

2 = なし

3.4 直径寸法 寸法補助記号 3 bits (10-12)

0 = φ 前

1 = 後 φ

2 = DIA 前

3 = 後 DIA

4 = φ 前

5 = 後 φ

6 = DIA 前

7 = 後 DIA

0 ～ 3 は直径寸法と長さ寸法で φ 追加の両方に摘要。

4 ～ 7 は直径寸法では寸法補助記号は付けないが、長さ寸法の φ 追加には有効。

3.5 未使用 4 bits (13 -16)

4.

4.1 角度寸法 表示形式 2 bits (1- 2)

0 = 度分秒 (60 進)

1 = 単位表示 (Deg)

2 = 単位表示なし

3 = 単位表示 (° シンボル)

4.2 角度寸法 度分秒 (60 進) 表示の最小単位 2 bits (3- 4)

0 = 度まで
1 = 分まで
2 = 秒まで
4.3 角度寸法　十進表示の小数桁数 (1-6)　4 bits (5- 8)
4.4 角度寸法　外寸法線の形状　1 bit (9)
0 = 線分
1 = 円弧
4.5 未使用　7 bits (10-16)
5.
5.1 未使用　1 bit (1)
5.2 公差　十進表示の小数桁数 (1-6)　4 bits (2- 5)
5.3 十進小数部うしろのゼロ除去　1 bit (6)
0 = 除去
1 = 除去しない
5.4 未使用　9 bits (7-16)

6-9. 寸法参照点 #1 の X 座標 (倍精度実数)
10-13. 寸法参照点 #1 の Y 座標 (倍精度実数)
以下省略。

寸法参照点、記入位置は寸法種類により異なります。詳細は以下の説明および図を参照のこと。
以下では P1, P2, …, P5 は点 (x, y) を、scf は寸法値倍率を表わします。

1) 長さ寸法 (P1, P2, P3, P4, scf)
P1 → P2, P3 → P4 は寸法補助線で平行であること。
P2 → P4 は寸法線の方向で、寸法値測定方向

2) 角度寸法 (P1, P2, P3, P4, P5)
P1 → P2, P3 → P4 は寸法補助線、P5 は中心点。
P2, P4 は寸法線 (円弧) の上の点
角度は P2 から P4 に向かって反時計回りとする。

3) 半径寸法 (P1, P2, scf)
P1 円周上の点
P2 円中心点

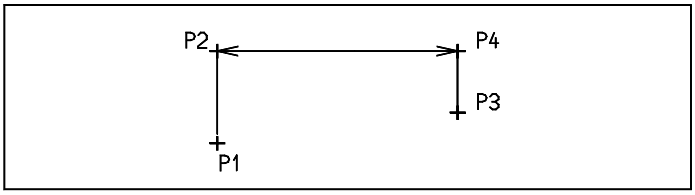
4) 直径寸法 (P1, P2, scf)
P1, P2 は円周上の対向点。
中心点は (P1+P2)/2 で求まる。

5) 座標寸法 (P1, scf)
P1 寸法参照点。

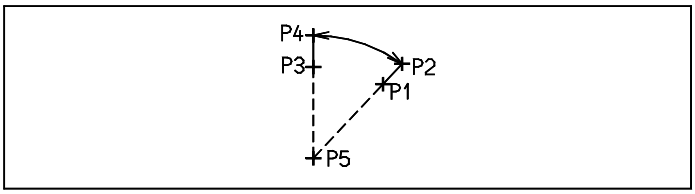
6) 円弧長寸法 (P1, P2, P3, P4, P5, scf)
P1 → P2, P3 → P4 は寸法補助線の方向。
P1, P3 は寸法記入円弧上の点、P2, P4 は寸法線 (円弧) 上の点、P5 は寸法記入円弧の中心点である。寸法記入円弧は P5 を中心として P1 から P3 へ反時計回りとする。

7) 面取り寸法 (P1, P2, scf)
P1 → P2 は 45° 面取り線の端点。

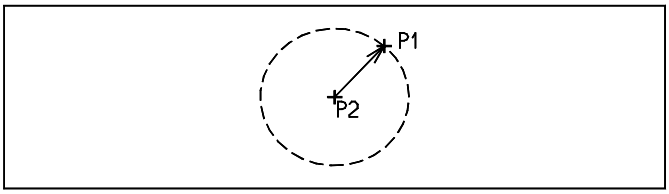
長さ寸法　分類コード〔1〕



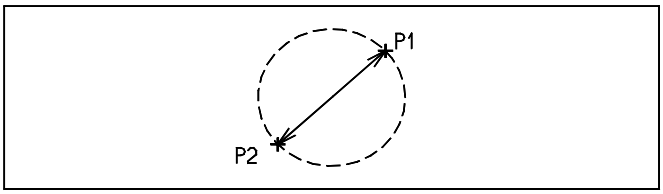
角度寸法　分類コード〔2〕



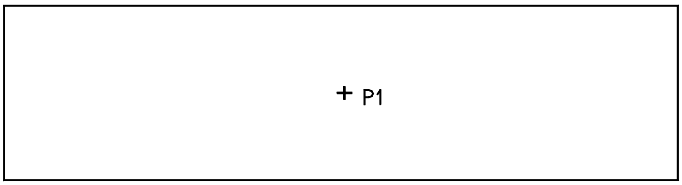
半径寸法　分類コード〔3〕



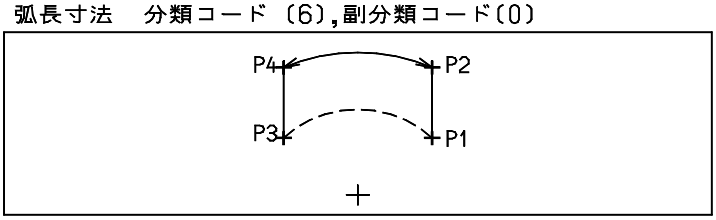
直径寸法　分類コード〔4〕



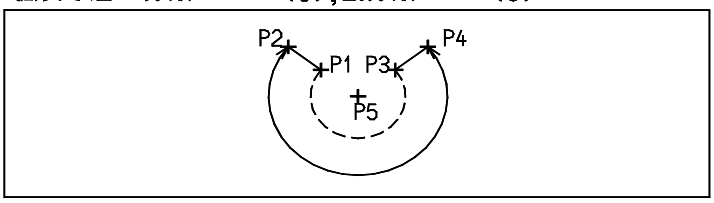
点寸法　分類コード〔5〕



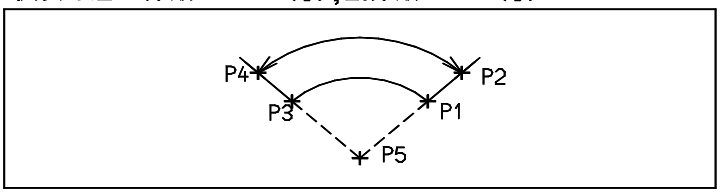
弧長寸法　分類コード〔6〕,副分類コード〔0〕



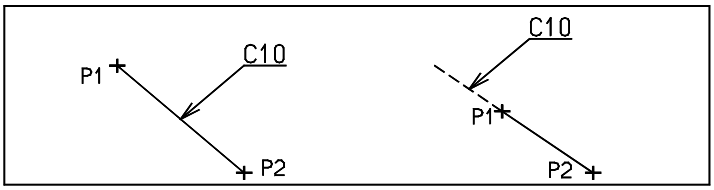
弧長寸法　分類コード〔6〕,副分類コード〔0〕



弧長寸法　分類コード〔6〕,副分類コード〔6〕



面取り寸法　分類コード〔7〕



5.2.4　サブレコード 17　幾何公差

【 目的 】

幾何公差パラメータ (Geometrical Tolerance)

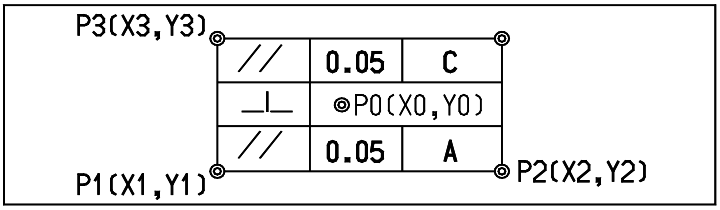
```
SR      17
MODE     2
COUNT    8
```

【 内容 】

1. 中心点 P0 の X 座標
2. 中心点 P0 の Y 座標
3. 枠の左下隅 P1 の X 座標
4. 枠の左下隅 P1 の Y 座標
5. 枠の右下隅 P2 の X 座標
6. 枠の右下隅 P2 の Y 座標
7. 枠の左上隅 P3 の X 座標
8. 枠の左上隅 P3 の Y 座標

【 補則 】

(1) 座標はすべてモデルスペース。



5.2.5 サブレコード 35 塗り潰しパラメータ

【 目的 】

塗り潰しのパターンを持ちます。SXF の仕様を取り込み拡張しています。Version 17 までのサブレコード (SR=18) に代わる新しいサブレコードです。サブレコードのデータは整数、倍精度実数の二つのプリミティーフデータで表現します。データタイプは５(複合型)とし、データ数は 2BYTES 整数を単位として数えます。倍精度実数は 4 と数えます。

```
SR          35
MODE         5
COUNT       38
```

【 内容 】

1 パターン
1.1 パターンの種類 2 bits(1- 2)
0 = ハードウェアのパターン
1 = マーク
2 = 文字
1.2 未使用 6 bits(3- 8)
1.3 テキストフォント番号 8 bits(9-16)
パターンの種類が 2 = 文字 のときのみ。それ以外は 0。
2. パターン番号
パターン種類＝0 : ハードウェアパターン番号
パターン種類＝1 : マーク番号
パターン種類＝2 : 文字コード ("EUC-JP")

以下のデータはパターン種類＝0 では無視されます。
3-6.　パターン配置角度（倍精度実数）(0.0-360.0 度）
7-10.　パターン配置基準点 X 座標（倍精度実数）
11-14.　パターン配置基準点 Y 座標（倍精度実数）
15-18.　パターン大きさ #1(倍精度実数)。負の値はパターンを反転する。
19-22.　パターン大きさ #2(倍精度実数)。負の値はパターンを反転する。
23-26.　パターン間隔 #1(倍精度実数）
27-30.　パターン間隔 #2(倍精度実数）
31-34.　パターン間隔 #1 方向の角度（倍精度実数）(0.0-360.0 度）
35-38.　パターン間隔 #2 方向の角度（倍精度実数）(0.0-360.0 度）

【 補則 】

(1)　座標はすべてモデルスペース。
(2)　パターン間隔は、隣り合うパターンの配置位置の距離で、パターン間の隙間の距離ではない。
(3)　SXF 仕様でないときは、以下の拘束を課します。
パターン配置角度＝パターン間隔 #1 方向の角度。
パターン間隔 #1 方向と間隔 #2 方向は直交。(+X 軸と +Y 軸の関係になる）
大きさ #1 <= パターン間隔 #1。
大きさ #2 <= パターン間隔 #2。

5.2.6　サブレコード 36　ハッチングパラメータ

【 目的 】

ハチングパターンを持ちます。SXF の仕様を取り込み拡張しています。Version 17 までのサブレコード（SR=19) に代わる新しいサブレコードです。サブレコードのデータは整数、倍精度実数の二つのプリミティーフデータで表現します。データタイプは５(複合型）とし、データ数は 2BYTES 整数を単位として数えます。倍精度実数は 4 と数えます。

```
SR          36
MODE         5
COUNT        3, 21, 39, 57, 75
```

【 内容 】

1.
1.1　ハッチング線定義数　　4 bits(1- 4)
SXF 仕様では 0 － 4、それ以外では 1。
SXF 仕様で線定義なし（=0）はハッチング領域保持に使用する。
1.2　仕様　　1 bit (5　)
0 =AdvanceCAD 仕様
1 =SXF 仕様
1.3　スタイル　　1 bit (6　)
0= 平行、1= クロス
1.4　未使用　　2 bits(7- 8)
1.5　ハッチングパターン番号　　8 bit (9-16)
AdvanceCAD 仕様
0 = パターンを使用しないで表示 / 非表示線の数を指示
1 - 32 = パターン番号

SXF 仕様
0 = ユーザ定義 (0)
1 - 6 = 既定定義ハッチング番号

2. 表示線の数 (1 - 1000)
AdvanceCAD 仕様でパターン番号 =0 の場合に有効。それ以外では 1 とみなす。

3. 非表示線の数 (0 - 1000)
AdvanceCAD 仕様でパターン番号 =0 の場合に有効。それ以外では 0 とみなす。

ハッチング線定義

4.
4.1 未使用 7 bits(1- 7)
4.2 線の色 (1-256) 9 bits(8-16)
SXF 仕様で有効。

5.
5.1 線種 (1-63) 8 bits(1- 8)
SXF 仕様で有効。
5.2 線幅 (1-16) 8 bits(9-16)
SXF 仕様で有効。

6-9. 通過点 X 座標 (倍精度実数)(モデルスペース)
10-13. 通過点 Y 座標 (倍精度実数)(モデルスペース)
14-17. 間隔 (倍精度実数)(ドローイングレイアウトスペース、正の値)
18-21. 角度 (倍精度実数)(0.0 - 360.0)

【 補則 】

(1) ハッチング線定義はハッチング線定義数分必要。
(2) SXF 仕様で既定定義ハッチングのときはハッチング線定義を変更してはならない。

5.3 その他

5.3.1 サブレコード 20 スペックデータ レコード番号

【 目的 】

スペックデータ レコード番号

```
SR      20
MODE     1
COUNT    1
```

【 内容 】

アイテムに付けたスペック (特性) データのレコード番号

5.3.2 サブレコード 21 NC マシニング レコード

【 目的 】

NC マシニング レコードを定義する。
(このサブレコードは NC のみに使用し、モデルファイルには保存できない)

SR　　21
MODE　　3
COUNT　　n (1 - 256)

【 内容 】

1.　実数値 #1
2.　実数値 #2
:　　:
n.　実数値 #n

NC マシニング レコードは、整数値・文字列・実数値の 3 種類のデータタイプがあり、これらはいずれも実数配列に格納される。
各実数配列要素がどのデータタイプを含んでいるかは以下のようにして判定する。
実数配列要素は 64 ビットで、これを 4 つの 16 ビット整数の並びとみなすと、

実数値 = { I1, I2, I3, I4 }

I1, I2, I3 = 0 で、I4 ≠ 0 のときは、I4 が整数値。
I1 = 0, I2 ≠ 0 のときは、I2, I3, I4 に 1 ～ 6 文字含む。 5 文字以下のときは残りの文字に NULL 文字 (=0) が入る。ASCII コード。
上記以外は全体で 64 ビットの実数値を表現する。0.0 のとき I1, I2, I3, I4= 0 である。

5.3.3　サブレコード 22　3D ポジションデータ

【 目的 】

NC 用位置データを 3 次元で定義する。
必要に応じて 3 次元点座標と 3 次元ベクトルを持つことができる。
(このサブレコードは NC のみに使用し、モデルファイルには保存できない)

SR　　22
MODE　　3
COUNT　　3 または 6

【 内容 】

COUNT = 3 のとき
1.　X
2.　Y
3.　Z

COUNT = 6 のとき
1.　X
2.　Y
3.　Z
4.　I
5.　J

6. K

5.3.4 サブレコード 25 スカラ列

【 目的 】

情報としての数値列

```
SR          25
MODE         1, 2, または 3
COUNT        n (MODE が 1 のとき 1 - 1024)
               (MODE が 2 のとき 1 - 512)
               (MODE が 3 のとき 1 - 256)
```

【 内容 】

1. 数値 #1
2. 数値 #2
: :
n. 数値 #n

5.3.5 サブレコード 26 非図形文字列

【 目的 】

情報としての文字列

```
SR          26
MODE         0
COUNT        文字列長さ（バイト数, 1 - 2048)
```

【 内容 】

1. byte#1
2. byte#2
: :
n. byte#n

【 補則 】

文字コードは EUC-JIS ("EUC-JP")。詳細はサブレコード 33 文字列の項を参照のこと。

5.3.6 サブレコード 27 時刻、レビジョン番号

【 目的 】

シンボル／サブモデルを配置した日時

SR 27
MODE 1
COUNT 3

【 内容 】

1. システム日付
   1986 年 1 月 1 日を起点とする日数。1986 年 1 月 1 日 == 0。
2. システム時間
2.1 時 (0-23) 5 bits (1 - 5)
2.2 分 (0-59) 6 bits (6 - 11)
2.3 秒 (0-29) 5 bits (12 - 16)
   実際の秒は、これを２倍したものである。たとえば 0 は、0 秒または 1 秒を意味する。
3. レビジョン番号
   シンボル／サブモデルのデータフォーマットのレビジョン番号

5.3.7 サブレコード 28 シンボル／サブモデル パラメータ

【 目的 】

シンボル／サブモデルを配置情報

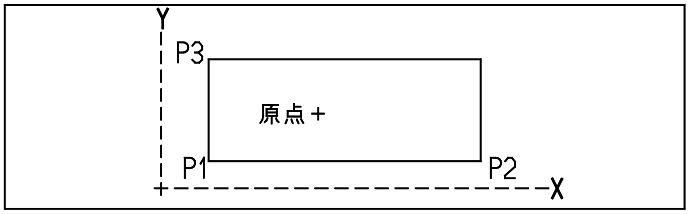
SR 28
MODE 3
COUNT 11

【 内容 】

シンボル／サブモデル パラメータ

1. 原点 X
2. 原点 Y
3. P1 X
4. P1 Y
5. P2 X
6. P2 Y
7. P3 X
8. P3 Y
9. X 軸縮尺値
   シンボル／サブモデルの配置の際の X 方向縮尺値。
   縮尺値 ＞ 0 X 軸反転なし
   縮尺値 ＜ 0 X 軸反転あり
10. Y 軸縮尺値
    シンボル／サブモデルの配置の際の Y 方向縮尺値。
    縮尺値 ＞ 0 Y 軸反転なし
    縮尺値 ＜ 0 Y 軸反転あり
11. 回転角度
    シンボル／サブモデルの配置の際の絶対座標系の X 軸に対する回転角度。X 軸と、P1 から P2 へのベクトルとのなす角度と一致する。

単位は度。(-360 ＜回転角度＜ 360)。



5.3.8 サブレコード 29 アソシエイト

【 目的 】

関係するアイテムの ID ポインタ番号を持つ。

```
SR          29
MODE         4
COUNT        2
```

【 内容 】

1. アソシエイティビティ カテゴリ番号 byte 1 / byte 2 / byte 3 / byte 4
   byte 3 : カテゴリ番号 1
   byte 4 : カテゴリ番号 2
2. 関係するアイテムの ID ポインタ番号

5.3.9 サブレコード 32 元のアイテム属性

【 目的 】

Version 17 までのサブレコード（SR=30) に代わる新しいサブレコードです。複合アイテム、サブモデル、シンボルなどの構造化アイテムで、元になったアイテムの属性を保持するのに用います。
SR=32 の次からつぎの SR=32 または SR=31 の直前までのサブレコードの並びが、元になったアイテムのデータであることを示します。

```
SR          32
MODE        1
COUNT       3
```

【 内容 】

1.
1.1 ブランクフラグ　　1 bit (1)
　　0 = 表示、1 = 非表示（ブランク）
1.2 アイテムタイプ　　6 bits(2- 7)
　　1 - 63

1.3	色番号 1 - 256	9 bits(8-16)
2.		
2.1	クラス番号 1 - 256	9 bits(1- 9)
2.2	未使用	1 bit (10)
2.3	線種番号 1 - 63	6 bits(11-16)
3.		
3.1	レビジョン番号 1 - 256	9 bits(1- 9)
3.2	未使用	2 bits(10-11)
3.3	線幅番号 1 - 16	6 bits(12-16)

5.3.10 サブレコード 31　End of item

【 目的 】

アイテムの終了を示す。これ以後にあらわれるサブレコードは無視される。あるいは消去される。

```
SR          31
MODE         1
COUNT        0
```

【 内容 】

なし。サブレコードヘッダのみ。

第 6 章　メッセージ・図形入出力モジュール

6.1　コマンド識別番号

● 関数一覧

関数名	機能
cmdidget	コマンド識別番号を得る。
cmdidset	コマンド識別番号を設定する。
cmdidcla	コマンド識別番号をクリアする。
cmdmdfget	修飾子識別番号を得る。
cmdmdfset	修飾子識別番号を設定する。
cmdmdfcla	修飾子識別番号をクリアする。
ldispatch	ディスパッチャ番号を得る。
ldriver	ドライバ番号を得る。
lformat	フォーム番号を得る。

6.1.1　コマンド識別番号の取得

【関数名】

cmdidget

【機能】

現在動作中のコマンドのコマンド識別番号を得る。

【呼出し形式】

```
void cmdidget(int cmdlvl, int *disp, int *driv, int *form)
```

【入力引数】

cmdlvl	コマンドレベル。(1, 3, 4, 5)

【出力引数】

disp	ディスパッチャ番号
driv	ドライバ番号
form	フォーム番号

6.1.2　コマンド識別番号の設定

【関数名】

cmdidset

【機能】

コマンド識別番号を設定する。

【呼出し形式】

```
void cmdidset(int cmdlvl, int disp, int driv, int form)
```

【入力引数】

cmdlvl	コマンドレベル。(1, 3, 4, 5)
disp	ディスパッチャ番号
driv	ドライバ番号
form	フォーム番号

6.1.3　コマンド識別番号のクリア

【関数名】

cmdidcla

【機能】

コマンド識別番号をクリアする。

【呼出し形式】

```
void cmdidcla(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

6.1.4　修飾子識別番号の取得

【関数名】

cmdmdfget

【機能】

修飾子識別番号を得る。

【呼出し形式】

```
void cmdmdfget(int cmdlvl, int *disp, int *driv, int *form)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

【出力引数】

disp　　ディスパッチャ番号
driv　　ドライバ番号
form　　フォーム番号

6.1.5　修飾子識別番号の設定

【関数名】

cmdmdfset

【機能】

修飾子識別番号を設定する。

【呼出し形式】

```
void cmdmdfset(int cmdlvl, int disp, int driv, int form)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)
disp　　ディスパッチャ番号
driv　　ドライバ番号
form　　フォーム番号

6.1.6　修飾子識別番号のクリア

【関数名】

cmdmdfcla

【機能】

修飾子識別番号をクリアする。

【呼出し形式】

```
void cmdmdfcla(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

6.1.7　ディスパッチャ番号の取得

【関数名】

ldispatch

【機能】

コマンドまたは修飾子のディスパッチャ番号を得る。

【呼出し形式】

```
int ldispatch(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドか修飾子かの指定。(1, 2, 3, 4, 5)
1, 3, 4, 5　　:　このコマンドレベルのコマンドのディスパッチャ番号を得る
2　　:　修飾子のディスパッチャ番号を得る

【返り値】

ディスパッチャ番号

6.1.8　ドライバ番号の取得

【関数名】

ldriver

【機能】

コマンドまたは修飾子のドライバ番号を得る。

【呼出し形式】

int ldriver(int cmdlvl)

【入力引数】

cmdlvl　コマンドか修飾子かの指定。(1, 2, 3, 4, 5)
1, 3, 4, 5　：　このコマンドレベルのコマンドのドライバ番号を得る
2　：　修飾子のドライバ番号を得る

【返り値】

ドライバ番号

6.1.9　フォーム番号の取得

【関数名】

lformat

【機能】

コマンドまたは修飾子のフォーム番号を得る。

【呼出し形式】

int lformat(int cmdlvl)

【入力引数】

cmdlvl　コマンドか修飾子かの指定。(1, 2, 3, 4, 5)
1, 3, 4, 5　：　このコマンドレベルのコマンドのフォーム番号を得る
2　：　修飾子のフォーム番号を得る

【返り値】

フォーム番号

6.2 メッセージ

(1) ゾーンはメニューのゾーン定義ファイル (ACADZON.MEN) で指定している。

(2) ゾーンは表形式 (行列) に区分けしてあり、メッセージ開始位置を指定できる。

ゾーン	行	列
#6 メッセージゾーン	1-3	1-3
#7 プロンプトゾーン	1-3	1-3

(3) メッセージゾーンは、コマンドの状態表示に使用する。

(4) プロンプトゾーンは、操作促進メッセージ行 (#1 と #2) とエラーメッセージ行 (#3) の２つから成る。
行 #1 と #2 は、一般のメッセージ表示には使用できない。
エラーメッセージと同様に、すぐに消えてもよいメッセージには行 #3 を使用できる。

(5) メッセージ番号に対応するメッセージがなければ表示しない。

● 関数一覧

関数名	機能
Errorb	ベルを鳴らす。
Errorcode	エラーメッセージを表示する。
Mesagdisp	メッセージを表示する。
Mesageras	メッセージを消す。
Opmsgcode	操作促進メッセージを表示する。
Menuzone	ゾーンの情報を得る。

6.2.1 エラーベル

【関数名】

Errorb

【機能】

ベルを鳴らす。

【呼出し形式】

```
void Errorb(void)
```

6.2.2　エラーメッセージの表示

【関数名】

Errorcode

【機能】

エラーメッセージを表示する。

【呼出し形式】

```
void Errorcode(int msgid)
```

【入力引数】

```
msgid   メッセージ番号
        msgid ＞ 0   ベルも鳴らす
        msgid ＜ 0   ベルは鳴らさない
```

【 注意 】

プロンプトゾーンの 3 行目に表示する。

6.2.3　メッセージの表示

【関数名】

Mesagdisp

【機能】

メッセージを表示する。

【呼出し形式】

```
void Mesagdisp(int iz, int row, int col, int msgid, int type, const void* data)
```

【入力引数】

```
iz      メッセージを表示するゾーンとカラー指定
        MZONECOLOR1     :  メッセージゾーンにカラー１で表示する
        MZONECOLOR2     :  メッセージゾーンにカラー２で表示する
        INPZONE         :  プロンプトゾーンに表示する
row     行番号
col     列番号
msgid   メッセージ番号。0 : メッセージなし
type    メッセージの後に表示するデータのタイプ。
           0        :なし, 1 : short (16 bits), 2 : float, 3 : double
           4        :int (32 bits), 10 * 文字数 : 文字列
data    メッセージの後に表示するデータ。
```

注意

標準では MZONECOLOR1 は緑色、MZONECOLOR2 は水色に設定されている。(詳しくは「システム管理者の手引き ８章メニューの作成」を参照)
コマンドオプションとして選択可能なメッセージは MZONECOLOR2 を、単なるメッセージは MZONECOLOR1 を使用している。

6.2.4 メッセージの消去

【関数名】

Mesageras

【機能】

メッセージを消す。

【呼出し形式】

```
void Mesageras(int iz, int row, int col)
```

【入力引数】

iz	ゾーン番号 MSGZONE : メッセージゾーン INPZONE : プロンプトゾーン
row	行番号。0 : すべての行
col	列番号。0 : すべての列

6.2.5 操作促進メッセージ表示

【関数】

Opmsgcode

【機能】

操作促進メッセージを表示する。

【呼出し形式】

```
void Opmsgcode(int ictg, int msgid)
```

【入力引数】

ictg	メッセージタイプ * 100 + コマンドレベル メッセージタイプ 0 または 1 : プライマリメッセージ プロンプト領域の１行目に表示する 2 : セカンダリメッセージ プロンプト領域の２行目に表示する コマンドレベル (1, 3, 4, 5)
msgid	メッセージ番号

注意

プライマリメッセージが指定されるとセカンダリメッセージはクリアされる。

従って操作促進メッセージを２行表示したい場合は、プライマリメッセージを指定してからセカンダリメッセージを指定すること。

6.2.6　指定ゾーン情報を抽出・設定

【関数名】

Menuzone

【機能】

指定ゾーンの情報を抽出、または設定する。

【呼出し形式】

```
int Menuzone(int iswt, int iz, short *izon)
```

【入力引数】

iswt　　抽出か設定かのスイッチ
　　1　　：設定する，2 ： 抽出する
iz　　ゾーン番号
　　1　　：グラフィックゾーン
　　2　　:MSC ゾーン
　　6　　：メッセージゾーン
　　7　　：プロンプトゾーン

【入出力引数】

izon　　指定ゾーンの情報
　iswt == 1 のときは入力引数
　iswt == 2 のときは出力引数
　izon は必ず 16 以上の配列とすること。(short izon[16])
　izon[0] ゾーンの左下座標 X （ラスター）
　izon[1] ゾーンの左下座標 Y （ラスター）
　izon[2] ゾーンの右上座標 X （ラスター）
　izon[3] ゾーンの右上座標 Y （ラスター）
　izon[4] 文字高さ（ラスター）
　izon[5] １行の高さ（ラスター）
　izon[6] １列の幅（ラスター）
　izon[7] 行数
　izon[8] 列数
　izon[9] ゾーンが動作中かどうか
　　1 ： 動作中、-1 ： 動作していない
　　ゾーン番号が 32 以下のときは必ず１となる。
　　ゾーン番号が 33 以上（マクロ テンポラリウインドウ）のときは，1 または -1 となる。
　zon[10-15] 未使用

【返り値】

0　　： 正常
1　　： ゾーン番号が範囲外
2　　： 指定ゾーンは定義されていない

6.3 アイテムピック

● 関数一覧

関数名	機能
IdentItem	アイテムをピックする。
IdentItemCandidate	次候補アイテムの有無を調べる。
IdentInfoCount	ピックされたアイテムの詳細情報の数を得る。
IdentInfo	ピックされたアイテムの詳細情報を得る。
IdentItems	複数アイテムの自動選択。
IdentItemsCandidate	次候補アイテムの有無を調べる。
IdentPoint	テンポラリポイント作成する。
identdb	アクティブモデル内のアイテムをピックする。
identsetbox	矩形のピック領域を設定する。
identsetply	多角形のピック領域を設定する。
IdentCount	ピックされたアイテム数を得る。
Identldptrs	ピックされたアイテムのアイテム識別子を得る。

テンポラリポイントコマンドの設定

関数名	機能
Tpntset	テンポラリポイントコマンドを設定する。
Tpntreset	Tpntset での設定を解除する。
Tpntstore	Tpntset での設定を現在のテンポラリポイントコマンドとする。

Tpntset を呼ぶと、ユーザがコマンドライン内で設定したテンポラリポイントコマンドを使わず、ここで指定したテンポラリポイントコマンドが有効になる。
Tpntset での設定を解除するには、Tpntreset を使用する。
また設定したテンポラリポイントコマンドをそのまま生かしたいときは Tpntstore を使用する。こうするとユーザがコマンドラインから入力したのと同じになる。

一時的な選択マスクの設定

関数名	機能
Tmskset	一時的な選択マスクを設定する。
Tmskreset	一時的な選択マスクを解除する。
Tmskstore	一時的な選択マスクを恒久的な選択マスクにする。

Tmskset を呼ぶと、一時的な選択マスクを設定できる。
Tmskset で設定した一時的な選択マスクを解除するには、Tmskreset を使用する。
また設定した一時的な選択マスクを恒久的な選択マスクにしたいときは Tmskstore を使用する。

選択 / 表示マスクの設定

関数名	機能
Msk001	選択マスクを設定する。
Msk002	表示マスクを設定する。
Msk101	選択マスクを得る。
Msk102	表示マスクを得る。
Mskcls	クラス選択マスクまたはクラス表示マスクを設定する。
Mskfnt	線種選択マスクまたは線種表示マスクを設定する。
Mskitm	アイテム選択マスクまたはアイテム表示マスクを設定する。
Mskrev	レビジョン選択マスクまたはレビジョン表示マスクを設定する。
Mskwet	線幅選択マスクまたは線幅表示マスクを設定する。

入力トークンの種類を設定する。

関数名	機能
tknmsk001	入力可能なトークンの種類を設定する。

構造体 TOKEN のメンバーを初期化する。

関数名	機能
tknclear	構造体 TOKEN のメンバー変数を初期化する。

6.3.1 アイテムのピック

【関数名】

IdentItem

【機能】

アイテムをピックする。

【呼出し形式】

```
int IdentItem(int cmdlvl, const TOKEN* token = NULL, int iswt = 0)
```

【入力引数】

cmdlvl　コマンドレベル。(1, 3, 4, 5)
token　トークン
token.typ
TknCMD または token が NULL:
初期化する。
・次候補アイテム情報のクリア
TknCOD、TknDIG　：　指定された座標でアイテムを選択する。
TknSPC　：　次候補アイテムを選択する。
TknIDP、TknSCL　：　指定されたアイテム識別子でアイテムを選択する。
token.pnt
token.typ が TknCOD、TknDIG のとき ： 座標値。
token.scl
token.typ が TknIDP、TknSCL のとき ： アイテム識別子。
iswt　アテンションボックスを表示するかどうか。
0　：表示しない。
1　：表示する。

【返り値】

token.typ が TknCOD、TknDIG、TknSPC、TknIDP、TknSCL のとき、選択されたアイテムのアイテム識別子。
0　：アイテムが選択できない または 引数が不正。
token.typ が TknCMD のとき
常に 0。

【補足】

次候補アイテムの有無を調べるには IdentItemCandidate() の返り値をみる。返り値が 2 以上ならば次候補アイテムが存在する。

注意

詳細な情報が必要なときは IdentInfo 関数を参照。

例

「2.4.4 アイテムの選択」を参照。

6.3.2　次候補アイテムの有無を調べる

【関数名】

IdentItemCandidate

【機能】

IdentItem 関数でピックされたアイテムに、次候補アイテムが存在するかどうかを調べる。

【呼出し形式】

```
int IdentItemCandidate(int cmdlvl)
```

【入力引数】

cmdlvl　コマンドレベル。(1, 3, 4, 5)

【返り値】

次候補アイテムの有無。(0, 1-n)
0　：アイテムがピックできなかった。次候補アイテムは存在しない。
1　：アイテムがピックできた。近傍に、次候補アイテムは存在しない。
2-　：アイテムがピックできた。近傍に、次候補アイテムが存在する。

6.3.3　ピックされたアイテムの詳細情報の数を得る

【関数名】

IdentInfoCount

【機能】

IdentItem 関数でピックされたアイテムの詳細情報の数を得る。

【呼出し形式】

```
int IdentInfoCount(void)
```

【返り値】

IdentItem 関数でピックされたアイテムの詳細情報の数。(0, 1-n)
0　：詳細情報がない。(アイテムがピックできなかった)
1　：詳細情報は一つだけ存在する。(アイテムが一つだけピックできた)
2-　：詳細情報は複数存在する。(近傍にアイテムが複数あった)
これはピックされた位置とアイテムとの距離の近い順に並んでおり
IdentItem、IdentItems の次候補要求で順次選択される。

【補足】

一度のピックで１アイテムだけを選択する場合はこの関数で数を調べる必要はなく、詳細情報の最初の一つだけを参照すればよい。
たとえば一点トリムコマンドのように一度のピックで二つのアイテムが必要な場合にこの関数で数を調べればよい。

6.3.4　ピックされたアイテムの詳細情報を得る

【関数名】

IdentInfo

【機能】

IdentItem 関数でピックされたアイテムの詳細情報配列のポインタを得る。

【呼出し形式】

```
IDENTINFO *IdentInfo(void)
```

【返り値】

IdentItem 関数でピックされたアイテムの詳細情報配列のポインタ。
NULL　：詳細情報がない。(IdentItem でアイテムがピックできなかった)

IDENTINFO 構造体のメンバ

int idptr	アイテム識別子。
int snum	ピックしたサブレコードの相対番号。 アイテムの最初のサブレコードを 1 として数える。
int styp	ピックしたサブレコードの種類。
int wend	幾何図形要素のどちらの端点に近い方を指示したか。 1 = 始点、-1 = 終点。ピックしたサブレコードの種類が線分・円・Bezier セグメントのときのみ有効。 その他のときは 1。
double dist	指示した位置と、カーブ上に投影した点との距離。
DPOINT ploc	指示した位置を、ピックしたカーブ上に投影した点。
DGEOM geom	サブレコードの内容。

サブレコードの種類	データ
SITPOINT : 点	P
SITLINE : 線分	Ps, Pe
SITARC : 円	Ps, Pm, Pe, Pc, radius, angle
SITBZCRV : 曲線	Q1, Q2, Q3, Q4, length
SITTXTPRM2 : 文字列	P0, P1, P2, P3, Px Px は文字列位置
SITMRKPRM2 : マーク	P0, P1, P2, P3, Px Px はスナップノード位置

int flag01	ピックしたサブレコードの前に現れた分類サブレコード (SITCATEG) の相対番号。
short sr01	分類サブレコードの内容。(flag01 >= 1 のとき有効)
int pid	図面配置状態のとき、ピックされたアイテムの属するピクチャ番号。(1 - 512) 0 = 図面配置状態ではない。
int wid	図面配置状態のとき、ピックされたアイテムの属するウインドウ番号。

IDENTINFO 構造体は acaddef.h 内で定義している。

例

```
int idptr1, idptr2;
const IDENTINFO* info = IdentInfo();
idptr1 = info->idptr;    /* 一番近いアイテムのアイテム識別子 */
idptr1 = info[0].idptr;  /* 上の行と同じ。どちらでもよい。*/
idptr2 = info[1].idptr;  /* 二番目に近いアイテムのアイテム識別子 */
```

6.3.5 複数アイテムの自動選択

【関数名】

IdentItems

【機能】

まず、与えられた点でアイテムをピックする。
アイテムがピックできないときは矩形または多角形領域でのアイテムの選択であるものとし「複数アイテムの自動選択コマンド（割り込みコマンド)」を起動して呼び出し元に戻る。
選択されたアイテムはハイライトアイテムリストに追加される。

【呼出し形式】

```
int IdentItems(int cmdlvl, TOKEN *token = NULL, int flag = 0)
```

【入力引数】

cmdlvl　コマンドレベル。(1, 3, 4, 5)

token　トークン

token.typ

TknCMD または token が NULL:
初期化する。
・次候補アイテム情報のクリア
・最後の操作情報のクリア

TknCOD、TknDIG : 指定された座標でアイテムをピックする。
ピックできないときは矩形または多角形での領域指定とみなし「複数アイテムの自動選択コマンド（割り込みコマンド)」を起動する。
選択されたアイテムはハイライトアイテムリストに追加される。
TknDIG のときに token.scl が 1 のとき（コントロール／シフトキーが押されていたとき）は選択されたアイテムをハイライトアイテムリストから排除する。

TknSPC : 次候補アイテムがあれば、前に追加されたアイテムをハイライトアイテムリストから排除し、次候補アイテムをハイライトアイテムリストに追加する。

TknBSP : 前回の操作を元に戻す。さらに続けて TknBSP が渡されるとハイライトアイテムリストに追加されているアイテムを一つづつ排除する。

TknIDP、TknSCL : 指定されたアイテム識別子をハイライトアイテムリストに追加する。

TknITM : 複数アイテムの自動選択コマンドからの完了通知。

TknEOC : 今回処理したアイテムのアイテム識別子を次回以降に参照するために保存しておく。保存対象は３番目の引数 flag で指定する。

TknMDF : TknEOC で保存されていたアイテム識別子（PRV）またはアクティブリスト内のアイテム識別子（USEACT）をハイライトアイテムリストに追加する。
どちら（PRV か USEACT）を追加するかは token.cid で指定する。

token.cid

token.typ が TknMDF のとき : コマンド修飾子のコマンド番号。
PRV　[34, 1, 207]
USEACT [34, 1,　1]

token.pnt

token.typ が TknCOD、TknDIG のとき:
座標値。

token.scl

token.typ が TknDIG のとき : コントロールキー／シフトキーが押されていたかどうか。
0 : 押されていない（アイテムの追加）
1 : 押されていた（アイテムの排除）

token.typ が TknIDP、TknSCL のとき:
アイテム識別子。

token.typ が TknITM のとき : 複数アイテムの自動選択コマンドからの完了通知。
IDENT_SUCCESS : 正常。
IDENT_FAILURE : 選択できなかった。

flag　token.typ が TknEOC のとき、保存対象を指定する。

SAVE_LST_ITM　　：ハイライトアイテムリストの内容。
SAVE_NEW_ITM　　：新たに作成されたアイテムのアイテム識別子。
SAVE_ACT_ITM　　：アクティブリスト内のアイテム識別子。
たとえば移動コマンド（MOVE）で「複製しない」ときは SAVE_LST_ITM または SAVE_ACT_ITM とし、「複製する」ときは SAVE_NEW_ITM としている。

【返り値】

token.typ が TknCOD、TknDIG のとき
IDENT_SUCCESS　　：アイテムが選択された。
IDENT_FAILURE　　：引数が不正。
IDENT_CONTINUE　　：複数アイテムの自動選択コマンドが起動された。
token.typ が TknSPC、TknIDP、TknSCL、TknBSP、TknMDF、TknITM のとき
IDENT_SUCCESS　　：アイテムが選択された。
IDENT_FAILURE　　：アイテムが選択できない または 引数が不正。
token.typ が TknEOC のとき
IDENT_SUCCESS　　：アイテム識別子が保存された。
IDENT_FAILURE　　：アイテム識別子が保存できない または 引数が不正。
token.typ が TknCMD のとき
IDENT_SUCCESS　　：正常。
IDENT_FAILURE　　：引数が不正。

【補足】

次候補アイテムの有無を調べるには IdentItemsCandidate() の返り値をみる。返り値が 2 以上ならば次候補アイテムが存在する。

例

「2.4.5 複数アイテムの自動選択」を参照。

6.3.6　次候補アイテムの有無を調べる

【関数名】

IdentItemsCandidate

【機能】

IdentItems 関数で、与えられた点でアイテムがピックできた場合、ピックされたアイテムの次候補アイテムが存在するかどうかを調べる。
矩形および多角形領域での選択の場合は返り値が 0 になる。

【呼出し形式】

```
int IdentItemsCandidate(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

【返り値】

次候補アイテムの有無。(0, 1-n)
0　　：与えられた点でアイテムがピックできなかった。または、矩形か多角形領域での選択。
1　　：アイテムがピックできた。近傍に、次候補アイテムは存在しない。
2-　　：アイテムがピックできた。近傍に、次候補アイテムが存在する。

6.3.7　テンポラリポイントを作成する

【関数名】

IdentPoint

【機能】

与えられた点と現在のテンポラリポイントモードでテンポラリポイントを作成する。
テンポラリポイントモードが交点や投影点などで１点ではテンポラリポイントが作成できないときは「テンポラリポイント作成コマンド（割り込みコマンド)」を起動して呼び出し元に戻る。

【呼出し形式】

```
int IdentPoint(TOKEN *token, DPOINT *pans)
```

【入力引数】

token　　トークン
　token.typ
　　TknCOD　　：　指定された点をテンポラリポイントにする。
　　TknDIG　　：　テンポラリポイントモードが自動点や端点などで１点でテンポラリポイントが作成できるときはテンポラリポイントを作成する。テンポラリポイントモードが交点や投影点などで１点ではテンポラリポイントが作成できないときは「テンポラリポイント作成コマンド（割り込みコマンド)」を起動する。
　　TknPNT　　：　テンポラリポイント作成コマンドからの完了通知。
　token.pnt
　　token.typ が TknCOD,TknDIG のとき：　座標値。
　token.scl
　　token.typ が TknPNT のとき　　：　テンポラリポイント作成コマンドからの完了通知。
　　IDENT_SUCCESS : 正常。
　　IDENT_FAILURE : 作成できなかった。

【出力引数】

pans　　テンポラリポイント。

【返り値】

```
token.typ が TknCOD のとき
  IDENT_SUCCESS     :  テンポラリポイントが作成できた。
  IDENT_FAILURE     :  引数が不正。
token.typ が TknDIG のとき
  IDENT_SUCCESS     :  テンポラリポイントが作成できた。
  IDENT_FAILURE     :  テンポラリポイントが作成できない または 引数が不正。
  IDENT_CONTINUE    :  テンポラリポイント作成コマンドが起動された。
token.typ が TknPNT のとき
  IDENT_SUCCESS     :  テンポラリポイントが作成できた。
  IDENT_FAILURE     :  テンポラリポイントが作成できない または 引数が不正。
```

例.「2.4.3 テンポラリポイントの作成」を参照。

6.3.8 アクティブモデル内のアイテムをピックする

【関数名】

identdb

【機能】

アクティブモデル内のアイテムをピックする。

【呼出し形式】

```
int identdb(int type, int flag, int vpid)
```

【入力引数】

type　　ピックの方法
　　5　　:identsetbox() で設定した矩形内。
　　3　　:identsetply() で設定した多角形内。
flag　　3 とする。
vpid　　ビューポート番号 (1-N)。0 はアクティブビューポート。

【返り値】

ピックしたアイテム数 (0, 1-N)
　0　　:　アイテムがピックできない

注意

予めピック領域を identsetbox または identsetply 関数で指定しておく。
選択マスク、表示マスクを参照し選択可能、表示可能なアイテムのみを対象とする。
ピックしたアイテムの識別子は IdentIdptrs 関数を参照のこと。

6.3.9 矩形のピック領域設定

【関数名】

identsetbox

【機能】

矩形のピック領域を設定する。
この関数で設定した領域は identdb 関数で使用される。

【呼出し形式】

```
void identsetbox(int flag, const DPOINT* p1, const DPOINT* p2)
```

【入力引数】

flag　　フラッグ。0 とする。
p1, p2　　矩形の対角点

6.3.10 多角形のピック領域設定

【関数名】

identsetply

【機能】

多角形のピック領域を設定する。
この関数で設定した領域は identdb 関数で使用される。

【呼出し形式】

```
void identsetply(int npnt, const DPOINT pbuf[])
```

【入力引数】

npnt　　多角形の点数 (3― 128)
pbuf　　多角形の点列（始点と終点は重複させない）

6.3.11 ピックされたアイテムの個数を得る

【関数名】

IdentCount

【機能】

identdb 関数でピックされたアイテムの個数を得る。

【呼出し形式】

```
int IdentCount(void)
```

【返り値】

identdb 関数でピックされたアイテムの個数（identdb の返り値と同じ)。(0, 1-N)
0　　：アイテムがピックできなかった

6.3.12 ピックされたアイテムのアイテム識別子を得る

【関数名】

Identldptrs

【機能】

identdb 関数でピックされたアイテムのアイテム識別子のポインタを得る。

【呼出し形式】

```
int *Identldptrs(void)
```

【返り値】

ピックされたアイテムのアイテム識別子配列のポインタ。
NULL　　：　アイテムがピックできなかった。

例.

```
DRECT rect;
int nitem;
rect.pmin.x = 0.0;
rect.pmin.y = 0.0;
rect.pmax.x = 200.0;
rect.pmax.y = 100.0;
identsetbox(0, &rect.pmin, &rect.pmax);
nitem = identdb(5, 3, 0);
if (nitem > 0) {
  int *idptrs = IdentIdptrs();
  int idptrFirst = idptrs[0];
  int idptrLast  = idptrs[nitem - 1];
}
```

6.3.13 テンポラリポイントの設定

【関数名】

Tpntset

【機能】

テンポラリポイントを設定する。

【呼出し形式】

```
void Tpntset(int key, int type)
```

【入力引数】

key　　投影点およびベクトル点の ON/OFF 状態の継続スイッチ
　　1　　：以前の状態を継続する
　　0　　：投影点及びベクトル点の選択状態を OFF にする
type　　設定するテンポラリポイント
　　1　　：端点
　　2　　：ディジタイズ点
　　3　　：中間点
　　4　　：中心点
　　5　　：交点
　　6　　：投影点
　　7　　：ベクトル点
　　8　　：自動点
　　9　　：ノード点
　　10　　：端点＋距離
　　11　　：配置点
　　12　　：自動点 #2

6.3.14 Tpntset での設定を解除

【関数名】

Tpntreset

【機能】

Tpntset での設定を解除する。

【呼出し形式】

```
void Tpntreset(void)
```

6.3.15 Tpntset での設定を現在のテンポラリポイントに設定

【関数名】

Tpntstore

【機能】

Tpntset での設定を現在のテンポラリポイントとする。

【呼出し形式】

```
void Tpntstore(void)
```

6.3.16 一時的な選択マスクを設定

【関数名】

Tmskset

【機能】

一時的な選択マスクを設定する。

【呼出し形式】

```
void Tmskset(int iform, int iopt, int num)
```

【入力引数】

iform	選択マスクの種類	
	1	：アイテム選択マスク
	2	：クラス選択マスク
	3	：レビジョン選択マスク
	4	：線種選択マスク
	5	：線幅選択マスク
iopt	モード	
	0	：初期設定モード（現在の状態をクリア後追加する）
	+	：追加モード　（現在の状態に追加する）
	-	：削除モード　（現在の状態から削除する）
num	設定するマスクを示す番号	
	アイテム選択マスクのとき	
	-1	： 全図形アイテム
	-2	： 幾何アイテム
	-3	： 製図アイテム
	ITMPOINT	： 点

:
ITMSUBMDL : サブモデル
クラス選択マスクのとき
0 : 全クラス
1 から MAXITMCLS : クラス番号
レビジョン選択マスクのとき
0 : 全レビジョン
1 から MAXITMREV : レビジョン番号
線種選択マスクのとき
0 : 全線種
1 から MAXITMLFT : 線種番号
線幅選択マスクのとき
0 : 全線幅
1 から MAXITMLWT : 線幅番号

6.3.17 一時的な選択マスクの解除

【関数名】

Tmskreset

【機能】

一時的な選択マスクを解除する。

【呼出し形式】

```
void Tmskreset(int iform)
```

【入力引数】

iform 選択マスクの種類
- 0 : 以下の 1 ～ 5 のマスク全部
- 1 : アイテム選択マスク
- 2 : クラス選択マスク
- 3 : レビジョン選択マスク
- 4 : 線種選択マスク
- 5 : 線幅選択マスク

6.3.18 一時的な選択マスクを恒久的に変更

【関数名】

Tmskstore

【機能】

一時的な選択マスクを恒久的な選択マスクにする。

【呼出し形式】

```
void Tmskstore(int iform)
```

【入力引数】

iform 選択マスクの種類

0　　：以下の 1 ～ 5 のマスク全部
1　　：アイテム選択マスク
2　　：クラス選択マスク
3　　：レビジョン選択マスク
4　　：線種選択マスク
5　　：線幅選択マスク

6.3.19 選択マスクの設定

【関数名】

Msk001

【機能】

選択マスクを設定する。

【呼出し形式】

```
int Msk001(int iflg, const short mask[])
```

【入力引数】

iflg　　設定する選択マスクの種類
- 1　　：アイテムタイプ選択マスクおよび一時的なアイテムタイプ選択マスク
- 2　　：クラス選択マスクおよび一時的なクラス選択マスク
- 3　　：レビジョン選択マスクおよび一時的なレビジョン選択マスク
- 4　　：線種選択マスクおよび一時的な線種選択マスク
- 5　　：線幅選択マスクおよび一時的な線幅選択マスク
- -1　　：一時的なアイテムタイプ選択マスク
- -2　　：一時的なクラス選択マスク
- -3　　：一時的なレビジョン選択マスク
- -4　　：一時的な線種選択マスク
- -5　　：一時的な線幅選択マスク

mask　　選択マスク
mask(ビット i) == 1 ： 選択可
mask(ビット i) == 0 ： 選択不可

【返り値】

0　　：正常
1　　：エラー（引数の値が不正）

注意

(1)　short mask[N];
アイテムタイプ表示マスクの時、N は 4 (MAXITMTYP+1 ビット)
クラス表示マスクの時、N は 16 (MAXITMCLS+1 ビット)
レビジョン表示マスクの時、N は 16 (MAXITMREV+1 ビット)
線種表示マスクの時、N は 4 (MAXITMLFT+1 ビット)
線幅表示マスクの時、N は 2 (MAXITMLWT+1 ビット)

(2)　ビット操作については ifld および insfld を参照

例 .

```
short mask[16];
if (Msk101(2, mask) == 0) {
  insfld(mask, 5, 5, 1); /* クラス５も選択可にする */
  Msk001(2, mask);
```

6.3.20 表示マスクの設定

【関数名】

Msk002

【機能】

表示マスクを設定する。

【呼出し形式】

```
int Msk002(int iflg, int ipic, const short mask[])
```

【入力引数】

iflg	設定する表示マスクの種類	
	1	：アイテムタイプ表示マスク
	2	：クラス表示マスク
	3	：レビジョン表示マスク
	4	：線種表示マスク
	5	：線幅表示マスク
ipic	設定するピクチャのピクチャ番号	
	-1	：全ピクチャに設定する
	0	：アクティブピクチャに設定する
	1 - MAXITMPIC	： ピクチャ #ipic に設定する
mask	表示マスク	
	mask(ビット i) == 1 ： 表示	
	mask(ビット i) == 0 ： 非表示	

【返り値】

0 ： 正常
1 ： エラー（ 引数の値が不正 ）

注意

(1) short mask[N];
アイテムタイプ表示マスクの時、N は 4 (MAXITMTYP+1 ビット)
クラス表示マスクの時、N は 16 (MAXITMCLS+1 ビット)
レビジョン表示マスクの時、N は 16 (MAXITMREV+1 ビット)
線種表示マスクの時、N は 4 (MAXITMLFT+1 ビット)
線幅表示マスクの時、N は 2 (MAXITMLWT+1 ビット)

(2) ビット操作については ifld および insfld を参照

例 . ピクチャ１のクラス５も表示可にする

```
short mask[16];
Msk102(2, 1, mask);
insfld(mask, 5, 5, 1);
Msk002(2, 1, mask);
```

6.3.21 選択マスクの取得

【関数名】

Msk101

【機能】

選択マスクを得る。

【呼出し形式】

```
int Msk101(int iflg, short *mask)
```

【入力引数】

iflg　　抽出する選択マスクの種類
- 1　　：アイテムタイプ選択マスク
- 2　　：クラス選択マスク
- 3　　：レビジョン選択マスク
- 4　　：線種選択マスク
- 5　　：線幅選択マスク
- -1　　：一時的なアイテムタイプ選択マスク
- -2　　：一時的なクラス選択マスク
- -3　　：一時的なレビジョン選択マスク
- -4　　：一時的な線種選択マスク
- -5　　：一時的な線幅選択マスク

【出力引数】

mask　　選択マスク
mask(ビット i) == 1 :　選択可
mask(ビット i) == 0 :　選択不可

【返り値】

0　　: 正常
1　　: エラー（引数の値が不正）

【 注意 】

(1)　short mask[N];
アイテムタイプ表示マスクの時、N は 4 (MAXITMTYP+1 ビット)
クラス表示マスクの時、N は 16 (MAXITMCLS+1 ビット)
レビジョン表示マスクの時、N は 16 (MAXITMREV+1 ビット)
線種表示マスクの時、N は 4 (MAXITMLFT+1 ビット)
線幅表示マスクの時、N は 2 (MAXITMLWT+1 ビット)
(2)　ビット操作については ifld および insfld を参照

例 . クラス選択マスクを得る

```
short mask[16];
Msk101(2, mask);
```

6.3.22 表示マスクの取得

【関数名】

Msk102

【機能】

表示マスクを得る。

【呼出し形式】

```
int Msk102(int iflg, int ipic, short *mask)
```

【入力引数】

iflg	抽出する表示マスクの種類
	1 ：アイテムタイプ表示マスク
	2 ：クラス表示マスク
	3 ：レビジョン表示マスク
	4 ：線種表示マスク
	5 ：線幅表示マスク
ipic	抽出するピクチャのピクチャ番号
	0 ：アクティブピクチャから抽出する
	1 - MAXITMPIC ： ピクチャ #ipic から抽出する

【出力引数】

mask	表示マスク
	mask(ビット i) == 1 ： 表示
	mask(ビット i) == 0 ： 非表示

【返り値】

0 ： 正常
1 ： エラー（引数の値が不正）

注意

(1) short mask[N];
アイテムタイプ表示マスクの時、N は 4 (MAXITMTYP+1 ビット)
クラス表示マスクの時、N は 16 (MAXITMCLS+1 ビット)
レビジョン表示マスクの時、N は 16 (MAXITMREV+1 ビット)
線種表示マスクの時、N は 4 (MAXITMLFT+1 ビット)
線幅表示マスクの時、N は 2 (MAXITMLWT+1 ビット)

(2) ビット操作については ifld および insfld を参照

例 . ピクチャ 1 のクラス表示マスクを得る

```
short mask[16];
Msk102(2, 1, mask);
```

6.3.23 クラス選択マスク・クラス表示マスクの設定

【関数名】

Mskcls

【機能】

クラス選択マスクまたはクラス表示マスクを設定する。

【呼出し形式】

```
int Mskcls(int iflg, const short mask[], int nmask, int ipic)
```

【入力引数】

```
iflg      クラス選択マスクかクラス表示マスクかの選択
             1        :クラス選択マスクを設定する
             2        :クラス表示マスクを設定する

mask      設定するクラス番号の並び（short mask[nmask]）
          1 - MAXITMCLS   :  設定するクラス番号
                             クラス 10 から 20 のような範囲指定のときは 10、-20 のように負のクラ
                             ス番号を使用できる。
          991             :  追加モード（以下に続くクラスを追加する）
          992             :  解除モード（以下に続くクラスを解除する）
          999             :  全クラス
nmask     mask に定義されている数
ipic      クラス選択マスクの設定のときは参照しない。
          クラス表示マスクの設定のときは設定するピクチャ
             -1       :全ピクチャに設定する
             0        :アクティブピクチャに設定する
          1 - MAXITMPIC   :  ピクチャ #ipic に設定する
```

【返り値】

```
             0        :正常
             1        :入力引数の誤り
```

例 . アクティブピクチャの表示クラスを 1 から 100 と 201 から 253 と 255 にする。

```
#define ADD 991
#define REL 992
#define ALL 999

short masks[8];
/* 例１  CLS/DSP REL ALL ADD 1 -100 201 -253 255 */
masks[0] = REL;          /* 解除モード REL */
masks[1] = ALL;          /* 全クラス ALL */
masks[2] = ADD;          /* 追加モード ADD */
masks[3] = 1;            /* クラス１から */
masks[4] = -100;         /* クラス１００まで -100 */
masks[5] = 201;          /* クラス２０１から 201 */
masks[6] = -253;         /* クラス２５３まで -253 */
masks[7] = 255;          /* クラス２５５ 255 */
Mskcls(2, masks, 8, 0);


/* 例２  CLS/DSP 1 -100 201 -255 REL 254 */
masks[0] = 1;            /* クラス１から */
masks[1] = -100;         /* クラス１００まで -100 */
masks[2] = 201;          /* クラス２０１から */
masks[3] = -255;         /* クラス２５５まで */
masks[4] = REL;          /* 解除モード */
masks[5] = 254;          /* クラス２５４ */
Mskcls(2, masks, 6, 0);
```

6.3.24 線種選択マスク・線種表示マスクの設定

【関数名】

Mskfnt

【機能】

線種選択マスクまたは線種表示マスクを設定する。

【呼出し形式】

```
int Mskfnt(int iflg, const short mask[], int nmask, int ipic)
```

【入力引数】

iflg　線種選択マスクか線種表示マスクかの選択
- 1　：線種選択マスクを設定する
- 2　：線種表示マスクを設定する

mask　設定する線種番号の並び (short mask[nmask])
- 1 - MAXITMLFT　：　設定する線種番号
  線種番号 10 から 20 のような範囲指定のときは 10、-20 のように負の番号を使用できる。
- 991　：　追加モード（以下に続く線種を追加する）
- 992　：　解除モード（以下に続く線種を解除する）
- 999　：　全線種

nmask　mask に定義されている数

ipic　線種選択マスクの設定のときは参照しない。
線種表示マスクの設定のとき設定するピクチャの指定
- -1　：全ピクチャに設定する
- 0　：現在のピクチャに設定する
- 1 - MAXITMPIC　：　ピクチャ #ipic に設定する

【返り値】

- 0　：正常
- 1　：入力引数の誤り

例

Mskcls を参照のこと。

6.3.25 アイテム選択マスク・アイテム表示マスクの設定

【関数名】

Mskitm

【機能】

アイテム選択マスクまたはアイテム表示マスクを設定する。

【呼出し形式】

```
int Mskitm(int iflg, const short mask[], int nmask, int ipic)
```

【入力引数】

iflg　アイテム選択マスクかアイテム表示マスクかの選択
- 1　：アイテム選択マスクを設定する
- 2　：アイテム表示マスクを設定する

mask　アイテムタイプの並び (short mask[nmask])
- -1　：全図形アイテム
- -2　：幾何アイテム
- -3　：製図アイテム

```
            ITMPOINT        :  点 , ..... ITMSUBMDL : サブモデル
            991             :  追加モード（以下に続くアイテムを追加する）
            992             :  解除モード（以下に続くアイテムを解除する）
    nmask   mask の長さ。
    ipic    アイテム選択マスクの設定のときは参照しない。
            アイテム表示マスクの設定のときピクチャ番号。
               -1       :全ピクチャに設定する
               0        :アクティブピクチャに設定する
            1 - MAXITMPIC   :  ピクチャ #ipic に設定する
```

【返り値】

```
0       : 正常
1       : 入力引数の誤り
```

例 .現在のピクチャの製図アイテムと点を表示しないようにする。

```
#define ADD 991
#define REL 992
short masks[5];
masks[0] = ADD;         /* 追加モード ADD */
masks[1] = -1;          /* 全図形 MANY */
masks[2] = REL;         /* 解除モード REL */
masks[3] = -3;          /* 製図アイテム MDRF */
masks[4] = ITMPOINT;    /* 点 MPNT */
/* ITM/SEL ADD MANY REL MDRF MPNT */
Mskitm(2, masks, 5, 0);
```

6.3.26 レビジョン選択マスク・レビジョン表示マスクの設定

【関数名】

Mskrev

【機能】

レビジョン選択マスクまたはレビジョン表示マスクを設定する。

【呼出し形式】

```
int Mskrev(int iflg, const short mask[], int nmask, int ipic)
```

【入力引数】

```
iflg    レビジョン選択マスクかレビジョン表示マスクかの選択
           1        :レビジョン選択マスクを設定する
           2        :レビジョン表示マスクを設定する

mask    設定するレビジョンのレビジョン番号 (short mask[nmask])
        1 - MAXITMREV   :  設定するレビジョン番号
                           レビジョン 10 から 20 のような範囲指定のときは 10､-20 のように負の番
                           号を使用できる。
        991             :  追加モード ( 以下に続くレビジョンを追加する )
        992             :  解除モード ( 以下に続くレビジョンを解除する )
        999             : 全レビジョン
nmask   mask に定義されている数
ipic    レビジョン選択マスクの設定のときは参照しない。
        レビジョン表示マスクの設定のときはピクチャ番号。
```

```
    -1          :全ピクチャに設定する
    0           :アクティブピクチャに設定する
  1 - MAXITMPIC   :  ピクチャ #ipic に設定する
```

【返り値】

```
0       : 正常
1       : 入力引数の誤り
```

例

Mskcls を参照のこと。

6.3.27 線幅選択マスク又は線幅表示マスクを設定

【関数名】

Mskwet

【機能】

線幅選択マスクまたは線幅表示マスクを設定する。

【呼出し形式】

```
int Mskwet(int iflg, const short mask[], int nmask, int ipic)
```

【入力引数】

```
iflg      線幅選択マスクか線幅表示マスクかの選択
            1         :線幅選択マスクを設定する
            2         :線幅表示マスクを設定する

mask      設定する線幅番号の並び (short mask[nmask])
          1 - MAXITMLWT   :  設定する線幅番号
                             線幅番号 1 から 10 のような範囲指定のときは 1,-10 のように負の番号を
                             使用できる。
          991             :  追加モード ( 以下に続く線幅を追加する )
          992             :  解除モード ( 以下に続く線幅を解除する )
          999             :  全線幅
nmask     mask に定義されている数
ipic      線幅選択マスクの設定のときは参照しない。
          線幅表示マスクの設定のときは設定するピクチャの指定
            -1        :全ピクチャに設定する
            0         :現在のピクチャに設定する
          1 - MAXITMPIC   :  ピクチャ #ipic に設定する
```

【返り値】

```
0       : 正常
1       : 入力引数の誤り
```

例

Mskcls を参照のこと。

6.3.28 入力可能なトークンの設定

【関数名】

tknmsk001

【機能】

入力可能なトークンの種類を設定する。

【呼出し形式】

```
void tknmsk001(int ictg, int mode, int type)
```

【入力引数】

ictg	コマンドレベル。(1, 3, 4, 5)
mode	初期設定するか追加するかのスイッチ
	-1 : 標準設定に戻す
	0 : 初期設定する
	1 : 追加する
type	トークンタイプ。mode が 0 または 1 の時に参照する
	TknCMD : コマンド
	TknMDF : 修飾子
	TknCOD : 座標
	TknDIG : デジタイズ座標
	TknSCL : 数値
	TknTXT : 文字列
	TknIDP : アイテム識別子
	TknBSP : バックスペース
	TknSPC : スペースキー
	TknVEC : ベクトル
	TknMZN : メッセージゾーン
	TknGZN : グラフィックゾーン
	TknUZN : テンポラリウィンドウ
	TknEOC : CE

注意

(1) トークンタイプ TknCMD, TknMDF, TknCOD, TknDIG, TknSCL, TknTXT, TknIDP, TknBSP, TknSPC, TknVEC, TknMZN, TknEOC はこの関数が呼ばれない時はいつでも有効になっている。(標準状態)

(2) この関数での設定は次の一回のトークンの入力だけに有効である。トークンが入力されるとこの設定はクリアされる。(標準状態に戻る)

(3) トークンタイプ TknCOD, TknDIG, TknIDP, TknVEC はグループ化されており、このグループ内のどれかを有効にするとこのグループ全てが有効になる。例えばトークンタイプ TknCOD を有効にすると、トークンタイプ TknDIG, TknIDP, TknVEC も有効になる。

(4) トークンタイプ TknBSP, TknEOC はいかなる場合も有効になっている。たとえばトークンタイプ TknSCL だけを有効と指定した場合は、トークンタイプ TknSCL, TknBSP, TknEOC が有効になる。

(5) トークンタイプ TknCOD, TknDIG, TknIDP, TknVEC と TknGZN は排他的な関係にあるので、同時に有効にはできない。例えばトークンタイプ TknGZN を有効にすると、TknCOD, TknDIG, TknIDP, TknVEC は入力不可になる。

(6) トークンタイプ TknCMD を有効にすると、トークンタイプ TknMDF, TknMZN も有効になる。

例.

1. トークンタイプ TknSPC と TknGZN だけを有効にする。

```
tknmsk001(1, 0, TknSPC); /* TknSPC だけを有効にする */
tknmsk001(1, 1, TknGZN); /* さらに TknGZN も有効にする */
```

2. 標準状態にトークンタイプ TknUZN も有効にする。

```
tknmsk001(1, 1 , TknUZN);
```

6.3.29 構造体 TOKEN メンバの初期化

【関数名】

tknclear

【機能】

構造体 TOKEN のメンバ変数を初期化する。

【呼出し形式】

```
void tknclear(TOKEN *token)
```

【出力引数】

token　初期化する TOKEN 構造体。
各メンバ変数は以下の値になる。

typ	:	TknVOID
cid	:	cid[0] ～ [4] まですべて 0
pnt	:	<0.0, 0.0>
xflg	:	1
yflg	:	1
input	:	<0.0, 0.0>
scl	:	0.0
sclflg	:	0
txt	:	'¥0'
vp	:	アクティブビューポート番号

6.4 画面制御

● 関数一覧

Clr002	**カラー割付の種類を設定する。**
Clr003	**カラー割付を設定する。**
Clr004	**カラーテーブルの値を設定する。**
Clr102	**カラー割付の種類を得る。**
Clr103	**カラー割付状態を得る。**
Clr104	**カラーテーブルの値を得る。**
Pan101	**画面表示部分の移動を行う。**
Pic001	**アクティブピクチャを切り換える。**
Pic101	**アクティブピクチャの番号を得る。**
Pic102	**指定ビューポートのピクチャ番号を得る。**
Rpt101	**画面の再表示を行う。**
Rptitems	**指定されたアイテムを画面上に表示または消去する。**
Slo001	**アクティブスクリーンレイアウトを切り換える。**
Slo101	**アクティブスクリーンレイアウトの番号を得る。**
Vie001	**アクティブビューポートを切り換える。**
Vie101	**アクティブビューポートの番号を得る。**
Viechang	**アクティブビューポートを一時的に切り換える。**
Vieerase	**指定ビューポートの指定プレーンを消去する。**
Zom101	**画面表示のズーミングを行なう。**

6.4.1 カラー割付種類を設定

【関数名】

Clr002

【機能】

カラー割付の種類を設定する。

【呼出し形式】

```
int Clr002(int type)
```

【入力引数】

type　　カラー割付の種類
- 1　：アイテムタイプ
- 2　：クラス
- 3　：レビジョン
- 4　：線種
- 5　：線幅

【返値】

0　　：正常
1　　：エラー（引数の値が不正）

例

```
/* カラー割付の種類をクラスにする */
Clr002(2);
```

6.4.2　カラー割付を設定

【関数名】

Clr003

【機能】

カラー割付を設定する。

【呼出し形式】

```
int Clr003(int type, short iclr[])
```

【入力引数】

type　　設定するカラー割付の種類
　　1　　：アイテムタイプ
　　2　　：クラス
　　3　　：レビジョン
　　4　　：線種
　　5　　：線幅

iclr　　属性それぞれに対するカラー番号 (1 - MAXITMCLR) の並び
　　short iclr[N];
　　アイテムタイプの時、N は MAXITMTYP
　　クラスの時、N は MAXITMCLS
　　レビジョンの時、N は MAXITMREV
　　線種の時、N は MAXITMLFT
　　線幅の時、N は MAXITMLWT

【返り値】

0　　：正常
1　　：エラー（引数の値が不正）

注意

この関数でカラー割付の設定を行なっても現在のカラー割付の種類は変更されない。カラー割付の種類を変更する時は Clr002 を呼出す。

例 .

```
short iclr[MAXITMCLS];
if (Clr103(2, iclr) == 0) {
  iclr[4] = 1; /* クラス５をカラー # １にする */
  Clr003(2, iclr);
}
```

6.4.3　カラーテーブルの設定

【関数名】

Clr004

【機能】

カラーテーブルの値を設定する。

【呼出し形式】

```
int Clr004(int iclr, short igrb[], int iswt)
```

【入力引数】

iclr	カラーの種類		
	-3	:	バックグラウンドカラー
	-2	:	グリッドのカラー
	-1	:	テンポラリ図形のカラー
	0	:	ラバーバンドカラー
	1 - MAXITMCLR	:	実図形のカラー
igrb	カラーテーブルの値		
	igrb[0]	:	緑の輝度 (0 - 255)
	igrb[1]	:	赤の輝度 (0 - 255)
	igrb[2]	:	青の輝度 (0 - 255)
iswt	0。		

【返り値】

0　　: 正常
1　　: エラー（引数の値が不正）

例

```
/* 実図形のカラー＃１を赤にする */
short igrb[3] = { 0, 255, 0};
Clr004(1, igrb, 0);
```

6.4.4　カラー割付種類の取得

【関数名】

Clr102

【機能】

カラー割付の種類を得る。

【呼出し形式】

```
int Clr102(void)
```

【返り値】

カラー割付の種類

1　　: アイテムタイプ
2　　: クラス
3　　: レビジョン
4　　: 線種
5　　: 線幅

例

```
/* カラー割付の種類を得る */
int clrmod;
clrmod = Clr102();
```

6.4.5　カラー割付状態の取得

【関数名】

Clr103

【機能】

カラー割付状態を得る。

【呼出し形式】

```
int Clr103(int type, short iclr[])
```

【入力引数】

type　　抽出するカラー割付の種類
　　1　　:アイテムタイプ
　　2　　:クラス
　　3　　:レビジョン
　　4　　:線種
　　5　　:線幅

【出力引数】

iclr　　属性それぞれに対するカラー番号 (1 - MAXITMCLR) の並び
　　short iclr[N];
　　アイテムタイプの時、N は MAXITMTYP
　　クラスの時、N は MAXITMCLS
　　レビジョンの時、N は MAXITMREV
　　線種の時、N は MAXITMLFT
　　線幅の時、N は MAXITMLWT

【返り値】

0　　: 正常
1　　: エラー（引数の値が不正）

例

```
/* クラスカラーの割付状態を得る */
short iclr[MAXITMCLS];
Clr103(2, iclr);
```

6.4.6　カラーテーブル値の取得

【関数名】

Clr104

【機能】

カラーテーブルの値を得る。

【呼出し形式】

```
int Clr104(int iclr, short igrb[])
```

【入力引数】

iclr　　カラーの種類
　　-3　　：バックグラウンドカラー
　　-2　　：グリッドのカラー
　　-1　　：テンポラリ図形のカラー
　　0　　：ラバーバンドカラー
　1 - MAXITMCLR　：　実図形のカラー

【出力引数】

igrb　　カラーテーブルの値
　　igrb[0]　　：　緑の輝度 (0 - 255)
　　igrb[1]　　：　赤の輝度 (0 - 255)
　　igrb[2]　　：　青の輝度 (0 - 255)

【返り値】

0　　：正常
1　　：エラー（引数の値が不正）

例

```
/* 実図形カラー #1 のカラーテーブルの値を得る */
short igrb[3];
Clr104(1, igrb);
```

6.4.7　アクティブピクチャの切り換え

【関数名】

Pic001

【機能】

アクティブピクチャを切り換える。

【呼出し形式】

```
int Pic001(int ipic)
```

【入力引数】

ipic　アクティブにするピクチャの番号 (1 - MAXITMPIC)

【返り値】

0　　：正常

1　　：エラー（引数の値が不正）

例

```
/* アクティブピクチャをピクチャ５にする */
Pic001(5);
```

6.4.8　アクティブピクチャ番号の取得

【関数名】

Pic101

【機能】

アクティブピクチャの番号を得る。

【呼出し形式】

```
int Pic101(void)
```

【返り値】

アクティブピクチャの番号 (1 - MAXITMPIC)

例 .

```
/* アクティブピクチャのピクチャ番号を得る */
int ipic;
ipic = Pic101();
```

6.4.9　指定ビューポートのピクチャ番号を得る

【関数名】

Pic102

【機能】

指定されたビューポートに表示されているピクチャの番号を得る。

【呼出し形式】

```
int Pic102(int ivie)
```

【入力引数】

ivie　　ビューポート番号。(1-N)

【返り値】

ピクチャ番号。(0、1-N)

0　　：指定されたビューポートが存在しない。

6.4.10　画面表示部分の移動

【関数名】

Pan101

【機能】

画面表示部分の移動を行なう。

【呼出し形式】

```
int Pan101(int form, DPOINT *pnt)
```

【入力引数】

form　　移動の分類
　　1　　：移動量を指定 (PAN P1-P2)
　　2　　：表示中心位置を指定 (PAN/CTR)
pnt　　移動量または表示中心位置
　　form == 1 のときの移動量（モデル座標系）
　　pnt->x　　：　X 方向の移動量
　　図形を向かって右側に移動（カメラを左側に移動）するとき正の値、反対方向のとき負の値とする。
　　pnt->y　　：　Y 方向の移動量
　　図形を上側に移動（カメラを下側に移動）するとき正の値、反対方向のとき負の値とする。
　　form == 2 のとき表示中心位置（モデル座標）
　　pnt->x　　：　表示中心位置 X
　　pnt->y　　：　表示中心位置 Y

【返り値】

0　　：正常
1　　：エラー（引数の値が不正）

6.4.11 画面の再表示

【関数名】

Rpt101

【機能】

画面の再表示を行う。

【呼出し形式】

```
void Rpt101(int iopt)
```

【入力引数】

iopt　　なにを再表示するかを指定する。
　　1　　：アクティブピクチャの再表示。
　　2　　：全てのビューポートの再表示。
　　3　　：メニューを含めた画面全体の再表示。
　　4　　：アクティブビューポートの再表示。

6.4.12 アイテムの表示・消去

【関数名】

Rptitems

【機能】

アイテムを画面上に表示 / 消去する。

【呼出し形式】

```
void Rptitems(int key, int mode, int *idptrs, int nidptr)
```

【入力引数】

key	スクリーンの選択及び表示データの選択
	0 ：テンポラリ図形プレーンに表示
	1 ：実図形プレーンに表示
	-1 ：テンポラリ図形プレーンにアイテムの構成点を表示
mode	表示か消去かの選択
	0 ：表示
	1 ：消去
idptrs	アイテム識別子の並び
nidptr	アイテム識別子の数

例 . テンポラリ図形プレーンにアイテム識別子 101 と 102 のアイテムを表示する。

```
int idptrs[2] = { 101, 102};
Rptitems(0, 0, idptrs, 2);
```

6.4.13 アクティブスクリーンレイアウトの切り換え

【関数名】

Slo001

【機能】

アクティブスクリーンレイアウトを切り換える。

【呼出し形式】

```
int Slo001(int islo)
```

【入力引数】

islo　スクリーンレイアウトの番号 (1 - 31)

【返り値】

0　：正常
1　：エラー（引数の値が不正）

例 .

```
/* スクリーンレイアウトを 5 にする */
Slo001(5);
```

6.4.14 アクティブスクリーンレイアウト番号の取得

【関数名】

Slo101

【機能】

アクティブスクリーンレイアウト番号を得る。

【呼出し形式】

```
int Slo101(void)
```

【返り値】

アクティブスクリーンレイアウト番号 (1 - 31)

例 .

```
/* アクティブなスクリーンレイアウト番号を得る */
int islo;
islo = Slo101();
```

6.4.15 アクティブビューポートの切り換え

【関数名】

Vie001

【機能】

アクティブビューポートを切り換える。

【呼出し形式】

```
int Vie001(int ivie)
```

【入力引数】

ivie　　アクティブにするビューポートの番号 (1 - 128)

【返り値】

0　　: 正常
1　　: エラー (引数の値が不正)

例 .

```
/* アクティブビューポートをビューポート５にする */
Vie001(5);
```

6.4.16 アクティブビューポート番号の取得

【関数名】

Vie101

【機能】

アクティブビューポートの番号を得る。

【呼出し形式】

```
int Vie101(void)
```

【返り値】

アクティブビューポートの番号 (1 - 128)

例 .

```
/* アクティブビューポートの番号を得る */
int ivie;
ivie = Vie101();
```

6.4.17 アクティブビューポートの一時切り換え

【関数名】

Viechang

【機能】

アクティブビューポートを一時的に切り換える。

【呼出し形式】

```
void Viechang(int ivie, int ista)
```

【入力引数】

ivie	アクティブにするビューポートの番号 (1 - 128)
ista	必ず 0 をセットすること。

注意

アクティブビューポート以外がピックされたときに、ピックされたビューポートを一時的にアクティブビューポートにする場合に使用する。処理が終了したらアクティブビューポートを元に戻しておくこと。

例 .

```
switch (token->typ) {

case X:
    .
    .
case TknDIG: /* ディジタイズ座標 */
```

```
        {
          /* アクティブビューポート番号、アクティブピクチャ番号を得る */
          int actvie = Vie101();
          int actpic = Pic101();

          /* ディジタイズされたビューポート番号、ピクチャ番号を得る */
          int tknvie = token->vp;
          int tknpic = Pic102(token->vp);

          /* ピックされたビューポートがアクティブビューポートでは */
          /* ないとき、アクティブビューポートを一時的にピックされた */
          /* ビューポートにする */
          if (tknvie != actvie)
             Viechang(tknvie, 0);

          /* ここで必要な処理を行う */
           .
           .
          /* アクティブビューポートを元に戻す */
          if (tknvie != actvie)
              Viechang(actvie);
        }
        break;

    case X:
        .
        .

    }
```

6.4.18 指定ビューポートの指定プレーン消去

【関数名】

Vieerase

【機能】

指定ビューポートの、指定プレーンを消去する。

【呼出し形式】

```
void Vieerase(int ivie, int plane)
```

【入力引数】

ivie	ビューポート番号	
	0	：全部のビューポート
plane	プレーン指定	
	GRIDSCREEN	： グリッドプレーン
	ACTSCREEN	： 実図形プレーン
	TEMPSCREEN	： テンポラリ図形プレーン
	RUBBSCREEN	： ラバーバンドプレーン

例 . 全てのビューポートのテンポラリ図形プレーンを消去する。

```
Vieerase(0, TEMPSCREEN);
```

6.4.19 表示画面のズーミング

【関数名】

Zom101

【機能】

画面表示のズーミングを行なう。

【呼出し形式】

```
int Zom101(int form, double *dprm)
```

【入力引数】

form　ズームの分類

1	:ZOOM/ALL
2	:ZOOM P1-P2
3	:ZOOM/UP
4	:ZOOM/DOWN
5	:ZOOM/BACK
11	:ZOOM/ALLVIE

dprm　ズームのためのパラメータ。下記以外のときは参照しない。

ZOOM P1-P2 のとき

dprm[0]	:	画面に表示する領域 左下 x
dprm[1]	:	画面に表示する領域 左下 y
dprm[2]	:	画面に表示する領域 右上 x
dprm[3]	:	画面に表示する領域 右上 y

ZOOM SCALE のとき

dprm[0]	:	今の表示状態に対する拡大／縮小率。たとえば倍にするときは 2.0 とする。

【出力引数】

0　: 正常
1　: エラー（引数の値が不正）

6.5 図面配置

● 関数一覧

関数名	機能
Drwmode	ドローイングモード（図面配置中）かどうかを調べる。
Drw001	ドローイングページの図面枠を設定／変更する。
Drw002	ドローイングページにピクチャを追加する。
Drw003	ドローイングページからピクチャを除去する。
Drw004	頁タイトルを設定する。
Drw005	頁タイトルを得る。
Drw006	ドローイングページの情報を得る。
Drw011	プロットファイルを作成する。(オフラインプロット)
Drw012	プロット出力を行う。(オンラインプロット)
Pen001	ペン番号の最大値を設定する。
Pen002	ペン割付の種類を設定する。
Pen003	ペン割付を設定する。
Pen101	ペン番号の最大値を得る。
Pen102	ペン割付の種類を得る。
Pen103	ペン割付状態を得る。
Scfval001	ピクチャ縮尺値またはドローイング縮尺値を設定する。
Scfval101	ピクチャ縮尺値またはドローイング縮尺値を取り出す。
Winorg001	ウインドウ原点を設定する。
Winorg101	ウインドウ原点を得る。
Winzon001	ウインドウゾーンを設定する。
Winzon101	ウインドウゾーンを得る。

6.5.1 ドローイングモードの判定

【関数名】

Drwmode

【機能】

ドローイングモード（図面配置中）かどうかを調べる。

【呼出し形式】

```
int Drwmode(void)
```

【返り値】

1　　：ドローイングモード
0　　：ドローイングモードではない（ドラフティングモード）

例 .

Advance CAD Version 13 からユーザコマンドとして割り込みコマンドを組み込むことができます。
従来のユーザコマンドは基本コマンドでありドローイングモード中に呼び出された場合はドローイングモードを終了させてからユーザコマンドを呼出します。従って基本コマンドのユーザコマンドが呼び出されたときはドローイングモードではありません。
これに対して割り込みコマンドはドローイングモードのままで割り込みコマンドを呼び出します。
割り込みコマンド内でドローイングモードかどうかを調べる場合にこのモジュールを使用します。

```
void dspatch88(TOKEN *token)
{
  switch (token->typ) {
  case TknCMD:
    if (Drwmode() == 1) {  /* ドローイングモード中 */
      cmdidcla(5);         /* コマンドを終了 */
      /* 必要ならばエラーメッセージを表示 */
      return;
    }
```

注意 .

ドローイングモード中にはスクリーンレイアウト、ビューポート、ピクチャ、アイテムに関連する操作はできません。

6.5.2　ドローイングページの図面枠を設定／変更

【関数名】

DrwO01

【機能】

ドローイングページの図面枠を設定／変更する。

【呼出し形式】

```
int DrwO01(int idrw, int keytmp, const char* sym, const int size[])
```

【入力引数】

idrw　　ドローイングページ番号
keytmp　図面枠の設定スイッチ
　　0　　：図面枠シンボルにより設定する。
　　1　　：図面枠の横幅／縦幅により設定する。
sym　　図面枠シンボル名。32 バイト以内。(NULL-terminated)
　　keytmp == 0 のとき参照する。
size　　図面枠の横幅／縦幅
　　keytmp == 1 のとき参照する。
　　size[0]　　：　図面枠の横幅 (1 - 32767)

size[1]　　：　図面枠の縦幅 (1 - 32767)

【返り値】

0　：正常終了
!0　：異常終了

6.5.3　ドローイングページにピクチャの配置

【関数名】

Drw002

【機能】

ドローイングページにピクチャを配置する。

【呼出し形式】

```
int Drw002(int idrw, int ipic, int iwin, const DPOINT* dorg, double dang)
```

【入力引数】

idrw　　ドローイングページ番号
ipic　　ピクチャ番号
iwin　　ウインドウ番号
dorg　　ピクチャ配置点（図面枠上の位置）
dang　　配置角度（度）

【返り値】

0　　：正常終了
!0　　：異常終了

6.5.4　ドローイングページからピクチャを除去

【関数名】

Drw003

【機能】

ドローイングページからピクチャを除去する。

【呼出し形式】

```
int Drw003(int idrw, int ipic, int iwin)
```

【入力引数】

idrw　　ドローイングページ番号
ipic　　ピクチャ番号
iwin　　ウインドウ番号
　　-1　　：全ウインドウ

【返り値】

0　　: 正常終了
!0　　: 異常終了

6.5.5　頁タイトルの設定

【関数名】

Drw004

【機能】

頁タイトルを設定する。

【呼出し形式】

```
int Drw004(int idrw, int ttlno, const char* text, int hmod, int hight,
           int omod, const DPOINT* dorg, int flag)
```

【入力引数】

idrw　　ドローイングページ番号
ttlno　　頁タイトル番号
text　　頁タイトル (NULL-terminated)
　　文字数 0 は 頁タイトル削除となり、自動表示のものは設定値を削除し、自動表示になる。
hmod　　文字高さ変更フラグ
　　0　　: 図面枠定義の文字高さで表示する。
　　1　　:hight で指定した文字高さで表示する。
hight　　文字高さ
　　hmod == 1 のとき、hight で指定された文字高さで表示する。
　　hight は mm または inch の 100 倍の整数で指定する。
　　たとえば 3.5 mm のときは 350 と指定する。
omod　　表示位置変更フラグ
　　0　　: 図面枠定義の位置に表示する。
　　1　　:TORG で指定した位置に表示する。
dorg　　表示位置
　　omod == 1 のとき、dorg で指定された位置に表示する。
　　表示位置は図面枠シンボルの原点からの距離で定義する。
flag　　頁タイトルを表示するかどうか。
　　-1　　: 表示しない。
　　1　　: 表示する。

【返り値】

0　　: 正常終了
!0　　: 異常終了

注意 .

この関数では、図面枠定義のキーワードが [MTLxxx] のものに対しては設定できない。

6.5.6　頁タイトルの取得

【関数名】

DrwO05

【機能】

頁タイトルを得る。

【呼出し形式】

```
int DrwO05(int idrw, int ttlno, char *text, short *hmod, short *hight,short *omod,
           DPOINT *dorg, short *flag)
```

【入力引数】

idrw	ドローイングページ番号
ttlno	頁タイトル番号

【出力引数】

text	頁タイトル (NULL-Terminated)
hmod	文字高さ変更フラグ 0 ：変更されていない。(図面枠定義の文字高さで表示) 1 ：変更されている。
hight	文字高さ (hmod == 1 のときのみ有効。) mm または inch の 100 倍の整数で表わされている。 たとえば 350 は 3.5 mm または 3.5 inch を意味している。
omod	表示位置変更フラグ 0 ：変更されていない。(図面枠定義の位置に表示) 1 ：変更されている。
dorg	表示位置 (omod == 1 のときのみ有効)。
flag	表示／非表示／入力状況 1 ：入力されているまたは入力されるべきタイトル。 2 ：自動表示されるテキスト。 各々負のときは、非表示とされている。

【返り値】

0	: 正常終了
!0	: 異常終了

6.5.7 ドローイングページの情報取得

【関数名】

DrwO06

【機能】

ドローイングページの情報を得る。

【呼出し形式】

```
int DrwO06(int idrw, int *keytmp, char *sym, int *size, int *npic, size_t l_sym)
```

【入力引数】

idrw	ドローイングページ番号
l_sym	配列 sym の大きさ

【出力引数】

keytmp	図面枠情報 -1 ：図面枠が設定されていない。 0 ：図面枠シンボルによって設定されている。 1 ：図面枠の横幅／縦幅によって設定されている。
sym	図面枠シンボル名。(NULL-terminated) keytmp == 0 のときだけ有効。 配列の大きさは 33 バイト必要。
size	図面枠の横幅／縦幅 keytmp == 1 のときだけ有効。 size[0] ： 図面枠の横幅 size[1] ： 図面枠の縦幅
npic	配置されたピクチャの数

【返り値】

0	： 正常終了
!0	： 異常終了

6.5.8 プロットファイルの作成

【関数名】

DrwO11

【機能】

プロットファイルを作成する。(オフラインプロット)

【呼出し形式】

```
int DrwO11(const char* name, int idrw)
```

【入力引数】

name	プロットファイル名 (NULL-terminated) プロットファイル名にファイル拡張子 ".PLT" を含める必要はない。
idrw	ドローイングページ番号 0 ：全頁

【返り値】

0	： 正常終了
!0	： 異常終了

6.5.9 プロット出力をする

【関数名】

DrwO12

【機能】

プロット出力を行う。(オンラインプロット)

【呼出し形式】

```
int Drw012(int idrw, const char* param, const char* log)
```

【入力引数】

idrw	ドローイングページ番号 0	:全頁
param	UNIX 版のときはシェルスクリプト oplot、Windows 版のときはバッチファイル oplot.bat に渡すパラメータ。(NULL-terminated) 複数のパラメータを指定するときはパラメータ間にスペースをいれる。 パラメータがないときはヌルポインター（(char *)NULL）またはヌル文字（""）を指定する。	
log	"ERRLOG"	: エラーログを、ファイル oplot.log に記録する。
	"ERRCHK"	: プロット出力の実行状態をウインドウに表示する。
	その他	: エラーログもウインドウも必要ない場合はヌルポインターまたはヌル文字を指定する。

【返り値】

0 : 正常終了
!0 : 異常終了

6.5.10 ペン番号の最大値設定

【関数名】

Pen001

【機能】

ペン番号の最大値を設定する。

【呼出し形式】

```
int Pen001(int nmax)
```

【入力引数】

nmax ペン番号の最大値 (1 - 255)

【返り値】

0 : 正常
1 : エラー（引数の値が不正）

例 .

```
ペン番号の最大値を 8 にする
Pen001(8);
```

6.5.11 ペン割付の種類を設定

【関数名】

Pen002

【機能】

ペン割付の種類を設定する。

【呼出し形式】

```
int Pen002(int iflg)
```

【入力引数】

iflg　　ペン割付の種類
- 0　　：カラー番号
- 1　　：アイテムタイプ
- 2　　：クラス
- 3　　：レビジョン
- 4　　：線種
- 5　　：線幅

【返り値】

0　　：正常
1　　：エラー（引数の値が不正）

例 . ペン割付の種類をクラスにする

```
Pen002(2);
```

6.5.12 ペン割付を設定

【関数名】

Pen003

【機能】

ペン割付を設定する。

【呼出し形式】

```
int Pen003(int iflg, short *ipen)
```

【入力引数】

iflg　　設定するペン割付の種類
- 1　　：アイテムタイプ
- 2　　：クラス
- 3　　：レビジョン
- 4　　：線種
- 5　　：線幅

ipen　　属性それぞれに対するペン番号（1-255）の並び。
short ipen[N]
iflg == 1 の時、N は MAXITMTYP
iflg == 2 の時、N は MAXITMCLS
iflg == 3 の時、N は MAXITMREV
iflg == 4 の時、N は MAXITMLFT
iflg == 5 の時、N は MAXITMLWT

【返り値】

0　　：正常

1　　：エラー（引数の値が不正）

注意

この関数でペン割付の設定を行なっても現在のペン割付の種類は変更されない。
ペン割付の種類を変更する時は Pen002 を使う。

例 . クラス５をペン１にする

```
short ipen[255];
Pen103(2, ipen);
ipen[4] = 1;
Pen003(2, ipen);
```

6.5.13 ペン番号の最大値取得

【関数名】

Pen101

【機能】

ペン番号の最大値を得る。

【呼出し形式】

```
int Pen101(void)
```

【返り値】

ペン番号の最大値 (1 - 255)

例 . ペン番号の最大値を得る

```
penmax = Pen101();
```

6.5.14 ペン割付の種類を取得

【関数名】

Pen102

【機能】

ペン割付の種類を得る。

【呼出し形式】

```
int Pen102(void)
```

【返り値】

ペン割付の種類

0　　：カラー番号
1　　：アイテムタイプ
2　　：クラス
3　　：レビジョン
4　　：線種

5 ：線幅

例 . ペン割付の種類を得る

```
penmod = Pen102();
```

6.5.15 ペン割付状態を取得

【関数名】

Pen103

【機能】

ペン割付状態を得る。

【呼出し形式】

```
int Pen103(int iflg, short *ipen)
```

【入力引数】

iflg 抽出するペン割付の種類
- 1 ：アイテムタイプ
- 2 ：クラス
- 3 ：レビジョン
- 4 ：線種
- 5 ：線幅

【出力引数】

ipen 属性それぞれに対するペン番号 (1-255) の並び。
short ipen[N]
iflg == 1 の時、N は MAXITMTYP
iflg == 2 の時、N は MAXITMCLS
iflg == 3 の時、N は MAXITMREV
iflg == 4 の時、N は MAXITMLFT
iflg == 5 の時、N は MAXITMLWT

【返り値】

0 : 正常
1 : エラー (引数の値が不正)

例 . クラスによるペン割付状態を得る

```
short ipen[MAXITMCLS];
Pen103(2, ipen);
```

6.5.16 ピクチャ縮尺値・ドローイング縮尺値を設定

【関数名】

Scfval001

【機能】

ピクチャ縮尺値またはドローイング縮尺値を設定する。

【呼出し形式】

```
void Scfval001(int ipic, double scf)
```

【入力引数】

```
ipic        ピクチャ縮尺値またはドローイング縮尺値の選択
              -1          : ドローイング縮尺値
              0           : アクティブピクチャのピクチャ縮尺値
            1 - MAXITMPIC   :  指定ピクチャのピクチャ縮尺値
scf         縮尺値
```

例 . ドローイング縮尺値を 1/2 に設定する。

```
Scfval001(-1, 0.5);
```

6.5.17 ピクチャ縮尺値・ドローイング縮尺値の取得

【関数名】

Scfval101

【機能】

ピクチャ縮尺値またはドローイング縮尺値を取り出す。

【呼出し形式】

```
double Scfval101(int ipic)
```

【入力引数】

```
ipic        ピクチャ縮尺値またはドローイング縮尺値の選択
-1          : ドローイング縮尺値
0           : アクティブピクチャのピクチャ縮尺値
1 - MAXITMPIC   :  指定ピクチャのピクチャ縮尺値
```

【返り値】

縮尺値

例 . アクティブピクチャのピクチャ縮尺値を取り出す

```
double picscf = Scfval101(0);
```

6.5.18 ウィンドウ原点を設定

【関数名】

Winorg001

【機能】

ウインドウ原点を設定する。

【呼出し形式】

```
int Winorg001(int ipic, int iwin, DPOINT *org)
```

【入力引数】

ipic	ピクチャ番号 (1 - MAXITMPIC)
iwin	ウインドウ番号 (0, 1 - n)
org	ウインドウ原点座標

【返り値】

0	: 正常
1	: エラー（引数の値が不正）

例 . ピクチャ＃１、ウインドウ＃０のウインドウ原点を <10,20> にする。

```
DPOINT org;
int err;
org.x = 10.0;
org.y = 20.0;
err = Winorg001(1, 0, &org);
```

6.5.19 ウィンドウ原点を取得

【関数名】

Winorg101

【機能】

ウインドウ原点を得る。

【呼出し形式】

```
int Winorg101(int ipic, int iwin, DPOINT *org)
```

【入力引数】

ipic	ピクチャ番号 (1 - MAXITMPIC)
iwin	ウインドウ番号 (0, 1 - n)

【出力引数】

org	ウインドウ原点座標

【返り値】

0	: 正常
1	: 引数の値が不正
2	: 指定されたウインドウが存在しない

例 . ピクチャ＃１、ウインドウ＃０のウインドウ原点を得る。

```
DPOINT org;
int err;
err = Winorg101(1, 0, &org);
```

6.5.20 ウィンドウゾーンを設定

【関数名】

Winzon001

【機能】

ウインドウゾーンを設定する。

【呼出し形式】

```
int Winzon001(int ipic, int iwin, DPOINT *pos, DPOINT *size, double ang)
```

【入力引数】

ipic	ピクチャ番号 (1 - MAXITMPIC)
iwin	ウインドウ番号 (1 - n)
pos	ウインドウ位置（左、下）
size	ウインドウの大きさ（幅、高さ）
ang	ウインドウ角度（度）

【返り値】

0	: 正常
1	: エラー（引数の値が不正）

注意

ウインドウ＃０にはウインドウゾーンは設定できない。

例 .

ピクチャ＃１、ウインドウ＃２のウインドウゾーンを <-10, -20> <100, 200> にする。

```
DPOINT pos, size;
double ang;
int err;
pos.x = -10.0;
pos.y = -20.0;
size.x = 100.0 - pos.x;
size.y = 200.0 - pos.y;
ang = 0.0;
err = Winzon001(1, 2, &pos, &size, ang);
```

6.5.21 ウィンドウゾーンを取得

【関数名】

Winzon101

【機能】

ウインドウゾーンを得る。

【呼出し形式】

```
int Winzon101(int pic, int iwin, DPOINT *pos, DPOINT *size, double *ang)
```

【入力引数】

ipic	ピクチャ番号 (1 - MAXITMPIC)

iwin　　ウインドウ番号（1 - n）

【出力引数】

pos　　ウインドウ位置（左、下）
size　　ウインドウの大きさ（幅、高さ）
ang　　ウインドウ角度（度）

【返り値】

0　　：正常
1　　：引数の値が不正
2　　：指定されたウインドウが存在しない

注意
ウインドウ＃０にはウインドウゾーンは存在しない。

例 .ピクチャ＃１、ウインドウ＃２のウインドウゾーンを得る。

```
例.　DPOINT pos, size;
例.　double ang;
例.　int err;
   err = Winzon101(1, 2, &pos, &size, &ang);
```

6.6 ラバーバンドとドラッグ

● 機能一覧

関数名	機能
Rubset	線分、円／円弧、矩形のラバーバンドを設定または解除する。
Dragopen	ドラッギングするアイテムの登録開始を指示する。
Dragclose	ドラッギングするアイテムの登録終了を指示する。
Dragend	ドラッギングを終了させる。

ラバーバンドとドラッギングは同時には使用できません。どちらか一方だけです。

● ドラッグするアイテムの登録方法

ドラッグするアイテムを登録し、それをドラッグします。登録するとき、アイテムの内容は線分、円弧、Bezier 曲線セグメントに分解して登録します。文字やマークも同様に分解します。ドラッグする図形を保持する領域の大きさには制限があります。容量を超えた部分は保持されません。ドラッグする図形が多すぎると、なめらかなドラッギングができませんので、この容量を大きくするのは意味がありません。
最初に Dragopen 関数でアイテム登録開始を指示します。
次に Rptitems 関数でアイテムの内容を登録します。
最後に Dragclose 関数でアイテム登録を終了させます。この直後にドラッギングが始まります。
ドラッギングを終了させるには、Dragend 関数を使います。

● 例

```
int idptr;

if ((idptr = IdentItem(1, token, 0)) <= 0) {
 /* Error */
} else {
  DPOINT pnts[2] = {{0.0, 0.0}, {0.0, 0.0}};
  Dragopen(1, 1, pnts);
  Rptitems(1, 0, &idptr, 1);
  Dragclose(1);
}
return;
```

6.6.1 ラバーバンドの設定または解除

【関数名】

Rubset

【機能】

線分、円／円弧、矩形、長さ寸法のラバーバンドを設定または解除する。

【呼出し形式】

```
void Rubset(int ictg, int type, int iswt, const double *dprm)
```

【入力引数】

ictg　　コマンドレベル。(1, 3, 4, 5)
type　　ラバーバンドのタイプ番号
iswt　　ラバーバンドの設定／解除
　　0　　：ラバーバンドを解除する。
　　1　　：ラバーバンドを設定する。
　　2　　：ラバーバンドデータを追加する。
dprm　　ラバーバンドデータ
　　iswt == 1 または iswt == 2 のときだけ必要。

ラバーバンドタイプとデータの設定方法

type 1　線分ラバーバンド
　　iswt == 1 で、線分の始点座標 (dprm) を設定すると、ラバーバンドを開始する。
type 2　矩形ラバーバンド
　　iswt == 1 で、矩形のコーナー座標 (dprm) を設定すると、ラバーバンドを開始する。
type 3　円ラバーバンド (中心－円周点)
　　iswt == 1 で、円の中心点座標 (dprm) を設定すると、ラバーバンドを開始する。
type 4　円弧ラバーバンド (始点－通過点－終点)
　　iswt == 1 で、円の始点座標 (dprm) を設定すると、線分ラバーバンドを開始する。
　　iswt == 2 で、円の通過点座標 (dprm) を設定すると、３点円弧ラバーバンドになる。
type 5　長さ寸法
　　iswt == 1 で、サブタイプと寸法両端点、文字枠の矩形 (dprm) を設定すると、寸法線のラバーバンドを開始する。文字枠をつけたくないときは矩形の 4 点を寸法端点 (1 点目) と同じ座標にセットする。
　　サブタイプ (1 : 水平寸法、2 : 垂直寸法、3 : 平行寸法)
type 7　円弧ラバーバンド (始点－終点－通過点)
　　iswt == 1 で、円の始点座標 (dprm) を設定すると、線分ラバーバンドを開始する。
　　iswt == 2 で、円の終点座標 (dprm) を設定すると、始終点を固定した３点円弧ラバーバンドになる。
type 8　線分ラバーバンド
　　X 軸または Y 軸に平行な線分のラバーバンドをする。線分が X 軸と成す角度が、Y 軸との角度よりも小さければ、X 軸に平行な線分のラバーバンドとなる。
　　補助座標系が有効なときは、補助座標系の X 軸、Y 軸を使用する。設定方法は type 1 と同じ。
type 9　矩形ラバーバンド (中心点－対角点)
　　iswt == 1 で、矩形の中心座標 (dprm) を設定すると、矩形の辺が補助座標形の X 軸および Y 軸に平行になるようにラバーバンドを開始する。
type 12　矩形ラバーバンド (対角点－対角点)
　　矩形の辺が補助座標系の X 軸と Y 軸に平行になるようにラバーバンドする。設定方法は type 2 と同じ。
type 13　円弧ラバーバンド (中心－始点－終点)
　　iswt == 1 で、円の中心点座標と半径 (dprm) を設定すると、円弧の始点がラバーバンドになる。
　　iswt == 2 で、円の始点座標 (dprm) を設定すると、円弧の終点がラバーバンドになる。
type 14　円ドラッギング
　　設定するデータはない。iswt == 1 でラバーバンドタイプだけを設定する。現在の半径の値が使用される。

```
/* 線分ラバーバンド開始 */
DPOINT ps = {100.0, 50.0};
Rubset(1, 1, 1, (double *)&ps);
return;
```

```
/* ラバーバンド解除 */
Rubset(1, 1, 0, (double *)NULL);
return;

/* <100,10> - <200,30> を両端点とした垂直寸法のラバーバンド */
/* 文字枠はつけない */
double dprm[13] = { 2, 100.0, 10.0, 200.0, 30.0, 100.0, 10.0, 100.0,
                    10.0, 100.0, 10.0, 100.0, 10.0};
Rubset(1, 5, 1, dprm);
return;
```

6.6.2 ドラッギングアイテムの登録開始

【関数名】

Dragopen

【機能】

ドラッギングするアイテムの登録開始を指示する。

【呼出し形式】

```
void Dragopen(int ictg, int mode, const DPOINT pnts[])
```

【入力引数】

ictg　　コマンドレベル。(1, 3, 4, 5)
mode　　ドラッギングのモード
　　1　　：移動ドラッギング
　　2　　：回転ドラッギング
　　3　　：水平移動ドラッギング
　　4　　：垂直移動ドラッギング
pnts　　ドラッギング基点 (DPOINT pnts[2])
　　pnts[0]　　：　移動中心点座標
　　pnts[1]　　：　回転中心点座標
　　移動ドラッギングのときは、回転中心点座標＝移動中心点座標とすること。

6.6.3 ドラッギングアイテムの登録終了

【関数名】

Dragclose

【機能】

ドラッギングするアイテムの登録終了を指示する。
ドラッギングアイテムの登録終了後、ドラッギングを開始する。

【呼出し形式】

```
void Dragclose(int ictg)
```

【入力引数】

ictg　　コマンドレベル。(1, 3, 4, 5)

6.6.4　ドラッギングの終了

【関数名】

Dragend

【機能】

ドラッギングを終了させる。

【呼出し形式】

```
void Dragend(int ictg)
```

【入力引数】

ictg　　コマンドレベル。(1, 3, 4, 5)

6.7 ハイライトリスト

● 機能一覧

関数名	機能
rptitmadd	アイテムリストにアイテムを追加する。
rptitmrel	アイテムリストからアイテムを削除する。
rptitmnum	アイテムリストのアイテム数を得る。
rptitmptr	アイテムリストのポインタを得る。
rptitmcla	アイテムリストの登録数を 0 にする。
rptpntadd	点列リストに点を追加する。
rptpntrel	点列リストから点を削除する。
rptpntnum	点列リストの点数を得る。
rptpntptr	点列リストのポインタを得る。
rptpntcla	点列リストの登録数を 0 にする。

● ハイライトリスト

ハイライトリストとは選択されたアイテムや点列を画面に白色表示させるための情報を保持しておくもので、ハイライトリストに登録された情報はズームや再表示が行なわれたときにも白色表示される。
ハイライトリストにはアイテムをハイライトさせる「アイテムリスト」と点列をハイライトさせる「点列リスト」がある。

● アイテムリストの使用例

```
int nitm, id;
int *idptrs;
switch (token->typ) {
case TknCMD:          /* コマンド */
    break;
case TokCOD:          /* 座標 */
case TokDIG:          /* デジタイズ座標 */
    id = IdentItem(...);
    nitm = rptitmadd(1, 1, &id, 1);
    break;
case TknBSP:          /* バックスペース */
    nitm = rptitmrel(1, (int *)NULL, -1);
    break;
case TknEOC:          /* CE */
    nitm = rptitmnum(1);
    idptrs = rptitmptr(1);
    /* 処理を行う */
        .
        .
    rptitmcla(1);
```

● 点列リストの使用例

```
int npnt;
DPOINT *pnts;
switch (token->typ) {
case TknCMD:          /* コマンド */
    break;
case TknCOD:          /* 座標 */
case TknDIG:          /* デジタイズ座標 */
    npnt = rptpntadd(1, 1, token->pnt);
    break;
case TknBSP:          /* バックスペース */
    npnt = rptpntrel(1, -1);
    break;
case TknEOC:          /* CE */
    npnt = rptpntnum(1);
    pnts = rptpntptr(1);
    /* 処理を行う */
        .
        .
    rptpntcla(1);
```

6.7.1 アイテムリストにアイテムを追加

【関数名】

rptitmadd

【機能】

アイテムリストにアイテムを追加し、テンポラリプレーンにアイテム形状を表示する。

【呼出し形式】

```
int rptitmadd(int cmdlvl, int hmod, const int idptrs[], int nidptr, int maxitm)
```

【入力引数】

cmdlvl	コマンドレベル。(1, 3, 4, 5)
hmod	表示方法。
	1 : 実際の図形を表示する。
	0 : 実際の図形とノード点を表示する。
	-1 : 始終点にマークを表示する。図形は表示しない
	-2 : 表示しない。
idptrs	追加するアイテムのアイテム識別子の並び。負でも可（負で登録される）。
nidptr	アイテムの数。
maxitm	登録可能なアイテム数の上限値。(0, 1 - n)
	0 : 無制限。

【返り値】

登録されているアイテムの数。(-3, -2, -1, 0 - n)

+n	: 登録されているアイテム数。
-1	: 引数が不正。
-2	: アイテムの数が上限値を超える。 またはスワップ領域が不足。
-3	: 指定されたアイテムは既に登録されている。

6.7.2 アイテムリストからアイテムを削除

【関数名】

rptitmrel

【機能】

アイテムリストからアイテムを削除し、テンポラリプレーンのアイテム形状を消去する。

【呼出し形式】

```
int rptitmrel(int cmdlvl, const int idptrs[], int nidptr)
```

【入力引数】

cmdlvl　コマンドレベル。(1, 3, 4, 5)

idptrs　nidptr が正のとき
　削除するアイテムのアイテム識別子の配列。
　アイテム識別子は負でもよい。
nidptr が負のとき
　使用しない。

nidptr　アイテムの数。(-2, -1, 1 - n)
　+n　：削除するアイテムの数。
　-1　：最後に追加したアイテムを削除する。
　-2　：全てのアイテムを削除する。

【返り値】

登録されているアイテムの数。(-2, -1, 0 - n)
　+n　：登録されているアイテム数。
　-1　：引数が不正。
　-2　：指定されたアイテムが登録されていない。

6.7.3 登録アイテム数の取得

【関数名】

rptitmnum

【機能】

アイテムリストに登録されているアイテム数を得る。

【呼出し形式】

```
int rptitmnum(int cmdlvl)
```

【入力引数】

cmdlvl　コマンドレベル。(1, 3, 4, 5)

【返り値】

登録されているアイテムの数。
+n　　：登録されているアイテム数。
-1　　：引数が不正。

6.7.4　アイテムリストのポインタを取得

【関数名】

rptitmptr

【機能】

アイテムリストのポインタを得る。

【呼出し形式】

```
int  *rptitmptr(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

【返り値】

アイテムリストのポインター。
NULL　　：　引数が不正。

6.7.5　アイテムリストの登録数を 0 にする

【関数名】

rptitmcla

【機能】

アイテムリストの登録数を 0 にする。
アイテムリストの内容は変化しない。
画面は変化しない。

【呼出し形式】

```
void rptitmcla(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

6.7.6　点列リストに点を追加

【関数名】

rptpntadd

【機能】

点列リストに点を追加し、テンポラリプレーンに点列形状を表示する。

【呼出し形式】

```
int rptpntadd(int cmdlvl, int hmod, const DPOINT pnt[])
```

【入力引数】

cmdlvl	コマンドレベル。(1, 3, 4, 5)
hmod	表示方法。
	1 ：点列を結ぶ直線。
	2 ：点数が 2 の時、矩形と１点目―２点目を結ぶ直線。 点数が 3 以上の時、点列を結ぶ直線。
	3 ：点数が 2 の時、WCS にそった矩形と１点目―２点目を結ぶ直線。 点数が 3 以上の時、点列を結ぶ直線。
	0 ：表示しない。
pnt	追加する点（モデル座標系）。

【返り値】

登録されている点列の数。(-2, -1, 0 - n)

+n	：登録されている点数。
-1	：引数が不正。
-2	：スワップ領域が不足。

6.7.7 点列リストから点を削除

【関数名】

rptpntrel

【機能】

点列リストから点を削除し、テンポラリプレーンの点列形状を消去する。

【呼出し形式】

```
int rptpntrel(int cmdlvl, int reltyp)
```

【入力引数】

cmdlvl	コマンドレベル。(1, 3, 4, 5)
reltyp	-1 ： 最後に追加した点を削除する。
	-2 ： 全ての点を削除する。

【返り値】

登録されている点列の数。(-2, -1, 0 - n)

+n	：登録されている点数。
-1	：引数が不正。
-2	：削除すべき点がない。

6.7.8 登録されている点数を取得

【関数名】

rptpntnum

【機能】

点列リストに登録されている点数を得る。

【呼出し形式】

```
int rptpntnum(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

【返り値】

登録されている点列の数。(-1, 0 - n)

+n　　：登録されている点数。
-1　　：引数が不正。

6.7.9　点列リストのポイントを取得

【関数名】

rptpntptr

【機能】

点列リストのポインタを得る。

【呼出し形式】

```
DPOINT *rptpntptr(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

【返り値】

点列リストのポインター。

NULL　　：　引数が不正。

6.7.10　点列リストの点数を 0 にする

【関数名】

rptpntcla

【機能】

点列リストの点数を 0 にする。
点列リストの内容は変化しない。
画面は変化しない。

【呼出し形式】

```
void rptpntcla(int cmdlvl)
```

【入力引数】

cmdlvl　　コマンドレベル。(1, 3, 4, 5)

6.8 アクティブリスト

● 関数一覧

関数名	機能
ActCount	アクティブリスト中のアイテム数を得る。
ActIdptrs	アクティブリスト中のアイテムを得る。
ActClear	アクティブリストを空にする。
ActAdd	アクティブリストにアイテムを追加する。

6.8.1 アクティブリスト中のアイテム数の取得

【関数名】

ActCount

【機能】

アクティブリストに登録されているアイテム識別子の数を得る。

【呼出し形式】

```
int ActCount(void)
```

【返り値】

アクティブリストに登録されているアイテム識別子の数。(0, 1-N)

6.8.2 アクティブリスト中のアイテムの取得

【関数名】

ActIdptrs

【機能】

アクティブリストに登録されているアイテム識別子のポインタ数を得る。

【呼出し形式】

```
int *ActIdptrs(void)
```

【返り値】

アクティブリストに登録されているアイテム識別子配列のポインタ。
NULL : アクティブリストにアイテムは登録されていない。

6.8.3　アクティブリストのクリア

【関数名】

ActClear

【機能】

アクティブリストを空にする。

【呼出し形式】

```
void ActClear(void)
```

6.8.4　アクティブリストにアイテムを追加

【関数名】

ActAdd

【機能】

アクティブリストにアイテムを追加する。

【呼出し形式】

```
int ActAdd(int idptrs, int nitm)
```

【入力引数】

idptrs	アイテム識別子の配列。
nitm	アイテム識別子の個数。

【返り値】

== 0	: 正常
!= 0	: エラー

第 7 章　データベース アクセスモジュール

7.1　データベース

● 関数一覧

関数名	機能
DBGetItmMax	アイテム数の上限値を得る
DBGetSitMax	サブレコード数の上限値を得る
DBGetBufMax	データサイズの上限値を得る
Dbdlitems	アイテムをデータベースから削除する。
Dbdlitmpc	指定ピクチャのアイテムをデータベースから削除する。
DbUundo	UNDO または UN-UNDO する。
Dbudblock	UNDO ブロックを区切る。
DbUnmeset	アイテムに名前を付ける。もしアイテムに名前がついていれば新しい名前に変える。
Dbnmerel	アイテムの名前を削除する。
Dbnmeget	アイテムの名前を得る。
DbUnmesrh	アイテム名からアイテム識別子を得る。
DbUrdopen	アイテムを読み出しのためオープンする。
DbUrdhead	サブレコードのヘッダを読み出す。
DbUrddata	サブレコードのデータを読み出す。
DbUrdclose	アイテムの読み出しを終了する。
DbUrdback	一度参照したサブレコードを再度参照させる。
Dbvrimax	アイテム識別子の最大値（最後に作成したアイテムの識別子）を得る。
Dbvriatr	アイテムの属性を得る。
Dbvrimbx	アイテムの最大／最小値を得る。

7.1.1　アイテム数の上限値を取得

【関数名】

DBGetItmMax

【機能】

アイテムの最大使用可能値を得る

【呼び出し形式】

```
int DBGetItmMax()
```

7.1.2　サブレコード数の上限値を取得

【関数名】

DBGetSitMax

【機能】

アイテムが持つことができるサブレコードの最大値を得る

【呼び出し形式】

```
int DBGetSitMax()
```

7.1.3　データサイズの上限値を取得

【関数名】

DBGetBufMax

【機能】

アイテムを構成するサブレコードのデータサイズ (バイト) の最大値を得る

【呼び出し形式】

```
int DBGetBufMax()
```

7.1.4　アイテムをデータベースから削除する

【関数名】

Dbdlitems

【機能】

アイテムをデータベースから削除する。

【呼び出し形式】

```
int Dbdlitems(const int itmlst[], int nitms)
```

【入力引数】

itmlst　　アイテム識別子のリスト
nitms　　アイテム識別子の数

【返り値】

0　　：正常終了
!0　　：異常終了

注意
削除されたアイテムは画面上から消去される。

例

```
/* アクティブリストのアイテムをデータベースから削除する。*/
if (Dbdlitems(Actldptrs(), ActCount()) == 0)
    Actclear();
```

7.1.5　指定ピクチャのアイテムをデータベースから削除する

【関数名】

Dbdlitmpc

【機能】

指定ピクチャのアイテムをデータベースから削除する。

【呼び出し形式】

```
int Dbdlitmpc(int keypic, int keyasc, int keymsk,
            const short msktyp[], const short mskcls[],
            const short mskrev[], const short mskfnt[],
            const short mskwet[]);
```

【入力引数】

keypic　　ピクチャ制御スイッチ
　　0　　：全てのピクチャのアイテムを削除する。
　　1 - MAXITMPIC　：　指定ピクチャのアイテムを削除する。
keyasc　　アソシエーションアイテム削除スイッチ。
　　0　　：アソシエーションアイテムを削除しない。
　　1　　：アソシエーションアイテムを削除する。
keymsk　　アイテム選択マスク制御スイッチ
　　0　　：選択マスクを無視する。
　　1　　：一時的な選択マスクで指示されたアイテムだけを削除する。

msktyp　　アイテムタイプマスク
mskcls　　クラスマスク
mskrev　　レビジョンマスク
mskfnt　　線種マスク
mskwet　　線幅マスク

【返り値】

0　　：正常終了
!0　　：異常終了

注意

削除されたアイテムは画面上から消去される。
マスク値の詳細は Msk001 の説明を参照のこと。

7.1.6　UNDO 又は UN-UNDO する

【関数名】

DbUundo

【機能】

UNDO または UN-UNDO する。

【呼び出し形式】

```
int DbUundo(int mode, int idptr)
```

【入力引数】

mode　　UNDO 実行モード
　　0　　:UN-UNDO する。
　　1　　：最後の 1 つを UNDO する。
　　2　　：最後の UNDO ブロックを UNDO する。
　　3　　：指示したアイテムを UNDO する。
　　4　　：指示したアイテムを含む UNDO ブロックを UNDO する。
　　5　　：最後の削除 UNDO ブロックを UNDO する。
　　6　　：現在の UNDO ブロックを UNDO する。
idptr　　アイテム識別子
　　UNDO 実行モード mode == 3, mode ==4 のときのみ参照。

【返り値】

0　　：正常終了
!0　　：異常終了

7.1.7　UNDO ブロックを区切る

【関数名】

Dbudblock

【機能】

UNDO ブロックを区切る。

【呼び出し形式】

```
void Dbudblock(void)
```

注意
オペレーション中にコマンドターミネータが入力されたとき自動的に UNDO ブロックは区切られる。UNDO ブロックを明示的に区切りたい場合にのみ使用する。

7.1.8 アイテム名を付ける

【関数名】

DbUnmeset

【機能】

アイテムに名前を付ける。もしアイテムに名前がついていれば新しい名前に変える。

【呼び出し形式】

```
DbUnmeset(int idptr, const char* name)
```

【入力引数】

idptr	アイテム識別子
name	アイテムの名前 (12 文字以内、Null-char terminated)

【返り値】

0 : 正常終了

注意
名前のついたアイテムを削除しても、その時点では削除されたアイテムの名前はまだ消えない。これは削除したアイテムをアンドゥして元に戻したとき、名前が復活できるようにするため。
これを避けるには、アイテムを削除する前にそのアイテムの名前を取り除くことが必要。

例

```
DbUnmeset(1, "PNT001");
```

7.1.9 アイテム名を削除

【関数名】

Dbnmerel

【機能】

アイテムの名前を削除する。

【呼び出し形式】

```
int Dbnmerel(int idptr)
```

【入力引数】

idptr	アイテム識別子

【返り値】

0　　：正常終了

7.1.10 アイテム名を取得

【関数名】

Dbnmeget

【機能】

アイテムの名前を得る。

【呼び出し形式】

```
int Dbnmeget(int idptr, char* name)
```

【入力引数】

idptr　　アイテム識別子

【出力引数】

name　　アイテムの名前 (12 文字以内、Null-char terminated)

【返り値】

0　　：正常終了
!0　　：異常終了、またはアイテムに名前がついていない。

例

```
char name[13];
if (Dbnmeget(1, name))
  return;
```

7.1.11 アイテム名からアイテム識別子を取得

【関数名】

DbUnmesrh

【機能】

アイテム名からアイテム識別子を得る。

【呼び出し形式】

```
int DbUnmesrh(const char* name, int* idptr)
```

【入力引数】

name　　アイテムの名前 (12 文字以内、Null-char terminated)

【出力引数】

idptr　　アイテム識別子

【返り値】

0　　：正常終了
!0　　：異常終了、または該当するアイテムがない。

7.1.12 アイテム読み込みの開始

【関数名】

DbUrdopen

【機能】

アイテムの読み出しのためデータベースをオープンする。

【呼び出し形式】

```
int DbUrdopen(int uid, int idptr)
```

【入力引数】

uid　　データベース参照番号 (1 または 2)
idptr　　アイテム識別子

【返り値】

0　　：正常終了

注意

DbUrdopen でオープンしたデータベース参照番号を DbUrdclose でクローズしないと、その参照番号を使って他のアイテムを参照することはできない。

例

```
#include "acadprm.h"
#include "acadusr.h"

#define SLOT 1


int isno, short sr01;
ItmSR srh;
FPOINT  rpnt;
DPOINT  dpnt;

if (DbUrdopen(SLOT, 1))
  return;
while (DbUrdhead(SLOT, &isno, &srh) == 0) {
    if (srh.typ == SITE01) {
        break;
    } else if (srh.typ == SITCATEG) {
        if (DbUrddata(SLOT, srh.cnt, &sr01))
            break;
    } else if (srh.typ == SITPOINT) {
        if (mode == 3) {
           if (DbUrddata(SLOT, srh.cnt, &dpnt))
                break;
        } else {
```

```
            if (DbUrddata(SLOT, srh.cnt, &rpnt))
                break;
        }
    }
}
DbUrdclose(SOLT);
```

7.1.13 サブレコードヘッダの読み込み

【関数名】

DbUrdhead

【機能】

サブレコードのヘッダを読み出す。

【呼び出し形式】

```
int DbUrdhead(int uid, int *sno, ItmSR *srh)
```

【入力引数】

uid	データベース参照番号

【出力引数】

sno	サブレコードの追い番
srh	サブレコードヘッダ

【返り値】

0	:	正常終了
!0	:	異常終了

例

DbUrdopen の例を参照のこと。

7.1.14 サブレコードの読み込み

【関数名】

DbUrddata

【機能】

サブレコードのデータを読み出す。

【呼び出し形式】

```
int DbUrddata(int uid, int cnt, void *dat)
```

【入力引数】

uid	データベース参照番号

cnt　　サブレコードのデータ数

【出力引数】

dat　　サブレコードのデータを格納する配列

【返り値】

0　　：正常終了

注意

DbUrdhead により参照したサブレコードのデータを得る。
DbUrdhead で得たサブレコードのデータ数 cnt をそのまま DbUrddata に渡せば、そのサブレコードのデータを得る。

例

DbUrdopen の例を参照のこと。

7.1.15 アイテム読み込みを終了

【関数名】

DbUrdclose

【機能】

アイテムの読み出しを終了する。

【呼び出し形式】

```
int DbUrdclose(int uid)
```

【入力引数】

uid　　データベース参照番号。
DbUrdopen でオープンしたデータベース参照番号。

【返り値】

0　　：正常終了

例

DbUrdopen の例を参照のこと。

7.1.16 サブレコードを再参照する

【関数名】

DbUrdback

【機能】

直前に参照したサブレコードを再度参照させる。

直前のサブレコードに対して有効であり、それより前のサブレコードを再度参照させることはできない。

【呼び出し形式】

```
int DbUrdback(int uid)
```

【入力引数】

uid　　データベース参照番号。

【返り値】

0　　: 正常終了

7.1.17 アイテム識別子の最大値を取得

【関数名】

Dbvrimax

【機能】

アイテム識別子の最大値 (最後に作成したアイテムの識別子) を得る。

【呼び出し形式】

```
int Dbvrimax(void)
```

【返り値】

アイテム識別子の最大値。

7.1.18 アイテム属性を取得

【関数名】

Dbvriatr

【機能】

アイテムの属性を得る。

【呼び出し形式】

```
int Dbvriatr(int idptr, ItmAtr *itmatr)
```

【入力引数】

idptr　　アイテム識別子

【出力引数】

itmatr　　アイテム属性

```
itmatr->pic    :  ピクチャ番号       (1 - MAXITMPIC)
itmatr->typ    :  アイテム種類番号   (1 - MAXITMTYP)
itmatr->cls    :  クラス番号         (1 - MAXITMCLS)
```

```
itmatr->rev    :  レビジョン番号     (1 - MAXITMREV)
itmatr->fnt    :  線種番号           (1 - MAXITMLFT)
itmatr->wet    :  線幅番号           (1 - MAXITMLWT)
itmatr->blk    :  ブランクフラグ     (0: 表示 1: 非表示 )
itmatr->color  :  色番号             (1 - MAXITMCLR)
```

【返り値】

0　　: 正常終了

注意

削除されたアイテムの属性は得られない。

7.1.19 アイテムの最大／最小座標値を取得

【関数名】

Dbvrimbx

【機能】

アイテムの最大／最小座標値を得る。

【呼び出し形式】

```
int Dbvrimbx(int idptr, int *flg, DRECT *mbx)
```

【入力引数】

idptr　　アイテム識別子

【出力引数】

flg　　ステータス
　　0　　: アイテムに表示する図形要素がないため、アイテムの最大／最小値が得られない。
　　1　　: アイテムの最大／最小値が得られた。

mbx　　アイテムの最大／最小座標
```
mbx->pmin    :  最小座標
mbx->pmax    :  最大座標
```

【返り値】

0　　: 正常終了

注意

アイテムにテキストまたはマークが含まれている場合は、それを囲む枠がアイテムの最大／最小の対象となる。

7.2　アイテム属性の現在値

アイテム属性の現在値の設定および取得をする関数。

● 関数一覧

関数名	機能
SetItemtype	アイテムタイプの現在値を設定する。
SetItemtclass	クラスの現在値を設定する。
SetItempic	ピクチャの現在値を設定する。
SetItemlwt	線幅の現在値を設定する。
SetItemfont	線種の現在値を設定する。
SetItemrev	レビジョンの現在値を設定する。
Itemtype	アイテムタイプの現在値を得る。
Itemtclass	クラスの現在値を得る。
Itempic	ピクチャの現在値を得る。
Itemlwt	線幅の現在値を得る。
Itemfont	線種の現在値を得る。
Itemrev	レビジョンの現在値を得る。

7.2.1 アイテムタイプの現在値の設定

【関数名】

SetItemtype

【機能】

アイテムタイプの現在値を設定する。

【呼出し形式】

void SetItemtype(int type)

【入力引数】

type　　アイテムタイプ番号。(1 - MAXITMTYP)

7.2.2 クラスの現在値の設定

【関数名】

SetItemclass

【機能】

クラスの現在値を設定する。

【呼出し形式】

```
void SetItemclass(int cls)
```

【入力引数】

cls　　クラス番号。(1 - MAXITMCLS)

7.2.3　ピクチャの現在値の設定

【関数名】

SetItempic

【機能】

ピクチャの現在値を設定する。

【呼出し形式】

```
void SetItempic(int pic)
```

【入力引数】

pic　　ピクチャ番号。(1 - MAXITMPIC)

注意

ピクチャの現在値を変えた場合は使用後に元の値に戻しておかなければならない。

例

```
int crtpic = Itempic();    /* ピクチャの現在値を保持しておく */
SetItempic(31);        /* ピクチャ番号を 31 に変更する */
SetItemtype(ITMLINE);
TmpgOpen1(0, 0);
     :
SetItempic(crtpic);    /* ピクチャの現在値を元に戻す */
```

7.2.4　線幅の現在値の設定

【関数名】

SetItemlwt

【機能】

線幅の現在値を設定する。

【呼出し形式】

```
void SetItemlwt(int lwt)
```

【入力引数】

lwt　　線幅番号。(1 - MAXITMLWT)

7.2.5　線種の現在値の設定

【関数名】

SetItemfont

【機能】

線種の現在値を設定する。

【呼出し形式】

```
void SetItemfont(int lft)
```

【入力引数】

lft　　線種番号。(1 - MAXITMLFT)

7.2.6　レビジョンの現在値の設定

【関数名】

SetItemrev

【機能】

レビジョンの現在値を設定する。

【呼出し形式】

```
void SetItemrev(int rev)
```

【入力引数】

rev　　レビジョン番号。(1 - MAXITMREV)

7.2.7　アイテムタイプの現在値の取得

【関数名】

ItemType

【機能】

アイテムタイプの現在値を得る。

【呼出し形式】

```
short ItemType(void)
```

【返り値】

アイテムタイプ番号。(1 - MAXITMTYP)

7.2.8　クラスの現在値の取得

【関数名】

ltemclass

【機能】

クラスの現在値を得る。

【呼出し形式】

short ltemclass(void)

【返り値】

クラス番号。(1 - MAXITMCLS)

7.2.9　ピクチャの現在値の取得

【関数名】

ltempic

【機能】

ピクチャの現在値を得る。

【呼出し形式】

short ltempic(void)

【返り値】

ピクチャ番号。(1 - MAXITMPIC)

7.2.10　線幅の現在値の取得

【関数名】

ltemlwt

【機能】

線幅の現在値を得る。

【呼出し形式】

short ltemlwt(void)

【返り値】

線幅番号。(1 - MAXITMLWT)

7.2.11 線種の現在値の取得

【関数名】

ltemfont

【機能】

線種の現在値を得る。

【呼出し形式】

short ltemfont(void)

【返り値】

線種番号。(1 - MAXITMLFT)

7.2.12 レビジョンの現在値の取得

【関数名】

ltemrev

【機能】

レビジョンの現在値を得る。

【呼出し形式】

short ltemrev(void)

【返り値】

レビジョン番号。(1 - MAXITMREV)

7.3 テンポラリアイテム

● 関数一覧

関数名	機能
TmpgInit	テンポラリ領域を空にする。
TmpgOpen1	新しいテンポラリアイテムをオープンする。
TmpgAddSr	サブレコードをひとつ追加する。
TmpgClose	テンポラリアイテムをとじる。
TmpgCopy	既存のアイテムをテンポラリアイテムとしてオープンする。
TmpgWrtoDB	テンポラリアイテムをデータベースに書き込む。

関数名	機能
TmpgGetSr	指定した番号のサブレコードのデータを読み出す。
TmpgWrt	サブレコードの番号を指定して、サブレコードを書き出す。
TmpgALast	アイテムの終了を表すサブレコードの直前にサブレコードをひとつ追加する。
Tmpgatr	テンポラリイアイテムのアイテム属性を変更する。
Tmpgback	最後につくったテンポラリアイテムを消去する。
Tmpgcanon	セグメントを得る。
Tmpgdel	サブレコードを削除する。
Tmpgfont	サブレコードのフォントを得る、または変更する。
Tmpgmaxid	テンポラリアイテムの数を得る。
Tmpgmove	サブレコードを移動する。
Tmpgsityp	サブレコードヘッダを得る。
Tmpgrptn	テンポラリアイテムの表示または消去（イレーズ）。
Tmpgstend	テンポラリアイテムの最初と最後のサブレコード番号を得る。

7.3.1 テンポラリ領域の初期化

【関数名】

TmpgInit

【機能】

テンポラリ領域を空にする。

【呼び出し形式】

```
void TmpgInit(void)
```

注意

テンポラリ領域にあるすべてのテンポラリアイテムがなくなる。

7.3.2 新規テンポラリアイテムのオープン

【関数名】

TmpgOpen1

【機能】

新しいテンポラリアイテムをオープンする。

【呼び出し形式】

```
int TmpgOpen1(int keydup, int idptr)
```

【入力引数】

keydup　複製モード

- 0　：更新
  このテンポラリアイテムをクローズしてデータベースに書き込むときに、元のアイテムを更新する。
- 1　：複製
  このテンポラリアイテムをクローズしてデータベースに書き込むときには、新しいアイテムとして追加する。元のアイテムはそのまま。
- 2　：複製
  このテンポラリアイテムをクローズしてデータベースに書き込むときには、新しいアイテムとして追加する。ただし、アイテム属性が現在値に変更される。元のアイテムはそのまま。

idptr　アイテム識別子

アイテム識別子を与えた場合は、このアイテムと同じアイテム属性でテンポラリアイテムをオープンする。
アイテムの指定なし（idptr = 0）の場合は、アイテム属性は現在値を使用する。

【返り値】

0　：正常終了

注意

テンポラリアイテム数が上限をこえるときは、今あるテンポラリアイテムをすべてデータベースに書出し、領域を空にする。
テンポラリアイテムはオープンされるだけで、空のアイテムである。

7.3.3　サブレコードの追加

【関数名】

TmpgAddSr

【機能】

サブレコードをひとつ追加する。

【呼び出し形式】

```
int TmpgAddSr(const ItmSR* sr)
```

【入力引数】

sr　サブレコード

【返り値】

0　：正常終了

7.3.4　テンポラリアイテムを閉じる

【関数名】

TmpgClose

【機能】

テンポラリアイテムを閉じる。
アイテムの終わりを示すサブレコード (SITEOI) を最後のテンポラリアイテムに追加する。

【呼び出し形式】

```
int TmpgClose(int keyrpt)
```

【入力引数】

keyrpt　アイテムを表示するかしないかの指示
- 0　：アイテムを表示しない。
- 1　：アイテムを表示する。

【返り値】

0　：正常終了

7.3.5　既存アイテムをテンポラリアイテムとして開く

【関数名】

TmpgCopy

【機能】

既存のアイテムをテンポラリアイテムとしてオープンする。

【呼び出し形式】

```
int TmpgCopy(int key, int idptr)
```

【入力引数】

key　オープンモード
- 0　：新規
  アイテム属性は現在値を使用する。
- 1　：更新
  指定アイテムと同じアイテム属性でテンポラリアイテムをオープンする。このテンポラリアイテムをクローズしてデータベースに書き込むときに元のアイテムを更新する。
- 2　：複製
  指定アイテムと同じアイテム属性でテンポラリアイテムをオープンする。このアイテムをクローズしてデータベースに書き込むときには、新しいアイテムとして追加する。元のアイテムはそのまま。
- 4　：追加
  指定のアイテムのデータをテンポラリアイテムに追加する。
  あらかじめ、テンポラリアイテムをオープンしておかなければならない。

idptr　テンポラリアイテムにするアイテムの識別子。

【返り値】

0　：正常終了

注意
テンポラリアイテム数が上限を越えるときは、テンポラリアイテムをデータベースに書出し領域を空にする。

key == 0, 4 のとき
アイテムの終わりを示すサブレコード（SITEOI）がつかない。
key == 1, 2 のとき
アイテムをスクリーンに表示する。key = -1, -2 のようにすると表示しない。

7.3.6 テンポラリアイテムをデータベースに登録

【関数名】

TmpgWrtoDB

【機能】

テンポラリアイテムをデータベースに書き込む。

【呼び出し形式】

```
int TmpgWrtoDB(int key)
```

【入力引数】

key	オプション	
	0	: すべてのテンポラリアイテムをデータベースに書き込む。 テンポラリアイテム数は 0 になる。テンポラリスクリーンをイレーズする。
	1	: すべてのテンポラリアイテムをデータベースに書き込む。 テンポラリアイテム数は 0 になる。テンポラリスクリーンはイレーズしない。
	2	: 最後のテンポラリアイテムを除いてデータベースに書き込む。 テンポラリアイテム数は 1、その番号は 1 になる。

【返り値】

0 : 正常終了

注意

空のテンポラリアイテムは無視する。
正しくクローズされていないアイテム (最後のサブレコードが SITEOI でない) は無視する。

7.3.7 指定番号のサブレコードデータを取得

【関数名】

TmpgGetSr

【機能】

指定した番号のサブレコードのデータを取り出す。

【呼び出し形式】

```
int TmpgGetSr(int ipos, ItmSR *sr)
```

【入力引数】

ipos　　サブレコードの番号 (1-)

【出力引数】

sr　　サブレコード

【返り値】

0　　: 正常終了

7.3.8　指定サブレコードの書き換え

【関数名】

TmpgWrt

【機能】

サブレコードの番号を指定してサブレコードを書き出す。

【呼び出し形式】

```
int TmpgWrt(int ipos, const ItmSR* sr)
```

【入力引数】

ipos　　サブレコードの番号 (1-)
sr　　サブレコード

【返り値】

0　　: 正常終了

補足 . TmpgAddSr と TmpgWrt の相違

TmpgAddSr は、TmpgOpen1 でオープンしたテンポラリアイテムにサブレコードを逐次追加するときに使用する。サブレコードはいつもそのアイテムの最後に追加することになり、サブレコードの番号は指定できない。

TmpgWrt は TmpgCopy で既存のアイテムをテンポラリアイテムにしたとき、特定のサブレコードを書き替えるときに使用する。

7.3.9　最終サブレコードの前にサブレコードを追加する

【関数名】

TmpgALast

【機能】

アイテムの終了をあらわすサブレコード (SITEOI) の直前にサブレコードをひとつ追加する。

【呼び出し形式】

```
int TmpgALast(const ItmSR* sr)
```

【入力引数】

sr　　サブレコード

【返り値】

0　　: 正常終了

7.3.10 テンポラリアイテムのアイテム属性変更

【関数名】

Tmpgatr

【機能】

テンポラリイアイテムのアイテム属性を変更する。

【呼び出し形式】

```
void Tmpgatr(int idtmp, int iswt, int newval)
```

【入力引数】

idtmp　テンポラリアイテムの識別番号
　　0　　:最後のテンポラリアイテム
　　1 ≦ ITMPNO ≦ テンポラリアイテムの数
iswt　変更項目番号
　　2　　:アイテムタイプ番号 (1 - MAXITMTYP)
　　3　　:クラス番号 (1 - MAXITMCLS)
　　6　　:線幅番号 (1 - MAXITMLWT)
　　7　　:線種番号 (1 - MAXITMLFT)
　　8　　:レビジョン番号 (1 - MAXITMREV)
　　9　　:ピクチャ番号 (1 - MAXITMPIC)
newval　新設定値

例

```
/* アイテムのレビジョン番号を２にする。*/
if (TmpgCopy(-1, idptr))
  return;
Tmpgatr(0, 8, 2);
TmpgWrtoDB(0);
```

0　　: 正常終了

7.3.11 最後に作成したテンポラリアイテムの消去

【関数名】

Tmpgback

【機能】

最後に作成したテンポラリアイテムを消去する。

【呼び出し形式】

```
int Tmpgback(int iswt, DPOINT *pnt)
```

【入力引数】

iswt	モード	
	0	：なし。
	1	：アイテムの始点を保存する。

【出力引数】

pnt	アイテムの始点。iswt=1 のときだけ有効。

【返り値】

0	：	正常終了
1	：	テンポラリアイテムはない。

7.3.12 セグメントの取得

【関数名】

Tmpgcanon

【機能】

セグメントを得る。

【呼び出し形式】

```
void Tmpgcanon(int ipos, ItmSR *sr)
```

【入力引数】

ipos	サブレコードの番号 (1 -)

【出力引数】

sr	セグメント セグメントはサブレコードの種類による。下記以外は無効。

sr->typ	sr->dat
線分	(Ps, Pe)
円、円弧	(Ps, Pm, Pe, Pc, radius, angle)
Cubic BEZER	(Q1, Q2, Q3, Q4, length)

注意

ipos の指すサブレコードが 始点 (SITSTART) のときは、次のサブレコードが線分 (SITLINE), 円弧 (SITARC) または Cubic BEZER (SITBZCRV) のときはそれを取り出す。

7.3.13 サブレコードの削除

【関数名】

Tmpgdel

【機能】

サブレコードを削除する。

【呼び出し形式】

```
void Tmpgdel(int ips, int ipe)
```

【入力引数】

ips, ipe　サブレコードの番号
ips 番目から ipe 番目までのサブレコードを取り除く。
0 ＜ ips ≦ ipe であること。

7.3.14 サブレコードのフォントを取得・変更

【関数名】

Tmpgfont

【機能】

サブレコードのフォントを得る、または変更する。

【呼び出し形式】

```
int Tmpgfont(int ipos, int lfnt)
```

【入力引数】

ipos　サブレコードの番号
lfnt　フォント番号
フォント番号を得たいときは、lfnt= -1 と設定する。
フォント番号を設定したいときは、lfnt= 0 ～ 7 に設定する。
各番号の設定内容は以下のとおり。
0 = アイテムと同じフォントで表示
1 = 非表示
2 ～ 7 = フォント番号 1 ～ 6 に対応

【返り値】

フォント番号。フォント番号は 0 ～ 7。-1 が返ったときはエラー。

7.3.15 テンポラリアイテム数を取得

【関数名】

Tmpgmaxid

【機能】

テンポラリアイテムの数を得る。

【呼び出し形式】

```
int Tmpgmaxid(void)
```

【返り値】

テンポラリアイテムの数。

7.3.16 サブレコードの移動

【関数名】

Tmpgmove

【機能】

サブレコードを移動する。

【呼び出し形式】

```
void Tmpgmove(int ipos, int icnt)
```

【入力引数】

ipos　　基準となるサブレコードの番号
icnt　　移動するサブレコード数。
　　icnt > 0
　　　　ipos 番目以降のサブレコードを icnt だけ後に移動する。
　　　　ipos 番目から (ipos+icnt-1) 番目があく。
　　icnt == 0
　　　　なにもしない。
　　icnt < 0
　　　　ipos 番目以降のサブレコードを icnt だけ前に移動する。
　　　　(ipos-icnt) 番目から (ipos-1) 番目のサブレコードが消去される。

7.3.17 サブレコードヘッダの取得

【関数名】

Tmpgsityp

【機能】

サブレコードヘッダを得る。

【呼び出し形式】

```
int Tmpgsityp(int ipos, ItmSR *sr)
```

【入力引数】

ipos　　サブレコードの番号

【出力引数】

sr　　サブレコード

【返り値】

0　　: 正常終了
-1　　: サブレコードの番号 が 0 または 負。

1　　：この番号のサブレコードはない。

7.3.18 テンポラリアイテムの表示・削除

【関数名】

Tmpgrptn

【機能】

テンポラリアイテムの表示または消去 (イレーズ)。

【呼び出し形式】

```
void Tmpgrptn(int ids, int ide, int mode, int vtxswt)
```

【入力引数】

ids	最初のテンポラリアイテムの番号
ide	最後のテンポラリアイテムの番号 1 ≦ ids ≦ ide
mode	表示，消去の指定 0 ： 表示，1 ： 消去
vtxswt	自由曲線の制御点列の表示の指定 1 ： 表示，0 ： 表示しない

7.3.19 テンポラリアイテムの最初と最後のサブレコード番号を取得

【関数名】

Tmpgstend

【機能】

テンポラリアイテムの最初と最後のサブレコード番号を得る。

【呼び出し形式】

```
int Tmpgstend(int idtmp, int *ips, int *ipe)
```

【入力引数】

idtmp	テンポラリアイテムの番号

【出力引数】

ips	このアイテムの先頭のサブレコードの番号
ipe	このアイテムの最後のサブレコードの番号

【返り値】

0　　：正常終了
1　　：この番号のテンポラリアイテムはない。

第 8 章　基本的な図形表示モジュール

この章の関数はデータベースアイテムおよびテンポラリアイテムを表示または消去　するものではありません。座標値を指定して線分、円、円弧、自由曲線、文字列を　表示または消去するものです。

8.1　基本図形表示モジュール

● 関数一覧

関数名	機能
Dsparc	円弧を表示する。
Dsparc3	３点円弧を表示する。
Dspclr	カラーまたは消去モードを設定する。
Dspend	Dspinit で作成した、表示用のサブウインドウを削除する。
Dsperas	表示領域全体または指定領域を消去する。
Dspinit	図形表示の使用環境を設定する。
Dsplft	線種を設定する。
Dspline	線分を表示する。
Dsplwt	線幅を設定する。
Dspmark	マークを表示する。
Dspmsg	メッセージを表示する。
Dspmsgers	メッセージを消す。
Dspscrn	表示プレーンを選択する。
Dspspln	自由曲線を表示または消去する。
Dspspls	3 次 Bezier 曲線を表示または消去する。
Dsptext	文字列を表示する。
Dspwcs	補助座標系を設定する。
Dspzone	表示領域座標を得る。

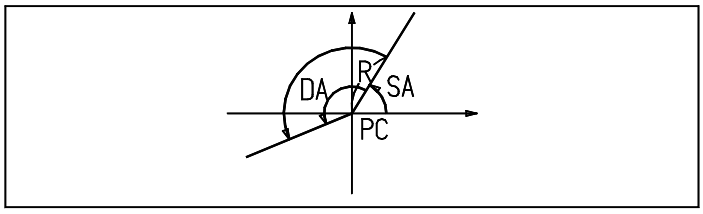
8.1.1　円弧の表示・削除

【関数名】

Dsparc

【機能】

円弧を表示または消去する。



【呼出し形式】

```
void Dsparc(const FPOINT* pc, double r, double sa, double da)
```

【入力引数】

pc	円弧の中心点
r	円弧の半径
sa	円弧の始点角度（度）
da	円弧の内角（度）。 正 ： 反時計回り、負 ： 時計回り、360.0 ： 完全円。

注意

使用に先だって Dspinit 関数で環境設定しなければならない。
補助座標系で指定する。

例 . 中心点 (10,5)、半径 5、の完全円を表示する。

```
FPOINT pc = {10.0, 0.0};
Dsparc(&pc, 5.0, 0.0, 360.0);
```

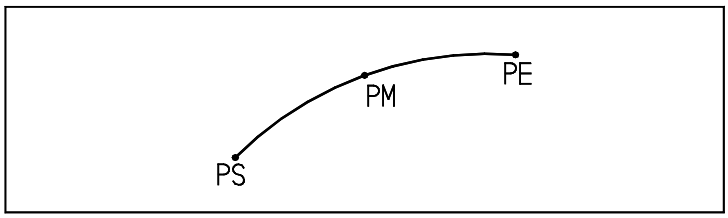
8.1.2　円弧の表示・削除

【関数名】

Dsparc3

【機能】

３点円弧を表示または消去する。



【呼出し形式】

```
void Dsparc3(const FPOINT* ps, const FPOINT* pm, const FPOINT* pe)
```

【入力引数】

ps	円弧の始点座標
pm	円弧の中間点座標
pe	円弧の終点座標

注意

使用に先だって Dspinit 関数で環境設定しなければならない。
補助座標系で指定する。

例

始点 (10,0)、通過点 (20,10)、終点 (30,0) の円弧を表示する。

```
FPOINT ps = {10.0, 0.0};
FPOINT pm = {20.0, 10.0};
FPOINT pe = {30.0, 0.0};
Dsparc3(&ps, &pm, &pe);
```

8.1.3 カラーまたは消去モードの設定

【関数名】

Dspclr

【機能】

表示のカラーまたは消去モードを設定する。

【呼出し形式】

```
void Dspclr(int iclr)
```

【入力引数】

iclr　カラー番号
テンポラリ図形用スクリーン、グリッド用スクリーン、
ラバーバンド用スクリーンが選択されている時
　0　：　以後の表示は消去となる
　1　：　以後の表示は表示となる (default)
実図形用スクリーンが選択されている時
　0　：　以後の表示は消去となる
　1-n　：　以後の表示は指定されたカラーとなる (default==1)

注意
使用に先だって Dspinit 関数で環境設定しなければならない。

8.1.4 表示用サブウィンドウの削除

【関数名】

Dspend

【機能】

Dspinit で作成した表示用のサブウインドウを削除する。
この関数はラスター座標系のときに有効。

【呼出し形式】

```
void Dspend(void)
```

注意
再度、図形表示関数を使用するには Dspinit 関数で環境設定しなければならない。

8.1.5 表示用サブウィンドウの消去

【関数名】

Dsperas

【機能】

表示領域全体または指定領域を消去する。

【呼出し形式】

```
void Dsperas(int iflg, const FRECT* rect)
```

【入力引数】

iflg	消去する部分の選択 0 : 表示領域全体を消去する。この場合、引数 rect は参照しない。 表示領域については Dspinit 関数を参照。 1 : rect で指定された領域内を消去する
rect	消去する領域の左下と右上座標

注意
使用に先だって Dspinit 関数で環境設定しなければならない。
この関数は補助座標系は考慮していない。絶対座標系で指定する。

例

(1) 表示領域全体を消去する。

```
Dsperas(0, (FRECT *)NULL);
```

(2) 領域 (0,0)-(100,20) 内を消去する。

```
FRECT rect = {0.0, 0.0, 100.0, 20.0};
```

```
Dsperas(1, &rect);
```

8.1.6 図形表示関数の使用環境を設定

【関数名】

Dspinit

【機能】

図形表示関数の使用環境を設定する。
これは図形表示関数を使用するに先だって必ず一度は呼び出す必要がある。

【呼出し形式】

```
void Dspinit(int iscr, int iunit)
```

【入力引数】

iscr　プレーンの選択
- 0 : テンポラリ図形用プレーン
- 1 : 実図形用プレーン
- 2 : グリッド用プレーン
- 3 : ラバーバンド用プレーン

iunit　座標系。表示領域はモジュール Dspzone で得ることができる。
- 0 : ラスター座標系
  メニューファイル ACADZON.MEN のキーワード Graphic zone で定義された座標系。
  これが選択されると ACADZON.MEN のウインドウ番号 #1 の大きさのサブウインドウが作成され、図形はサブウインドウの下になり見えなくなる。
  以下図形表示関数はこのサブウインドウに表示する。
  このサブウインドウの行列数は Menuzone 関数で得ることができる。
- 1 : データベース座標系
  アクティブビューポート内のピクチャの表示状態と同じ位置関係となる。

注意

ラスター座標系で使用した場合、最後にサブウインドウを削除するために Dspend を呼び出す必要がある。
Dspend を呼出さない場合でもメインカテゴリのコマンドが要求されると、Dspinit で作成したサブウインドは削除される。

例 . 実図形用スクリーンにデータベース座標系で表示する。

```
Dspinit(1, 1);
```

8.1.7 線種の設定

【関数名】

Dsplft

【機能】

線種を設定する。

【呼出し形式】

```
void Dsplft(int lft)
```

【入力引数】

lft	線種番号。1 - MAXITMLFT (default==1)

注意
使用に先だって Dspinit 関数で環境設定しなければならない。

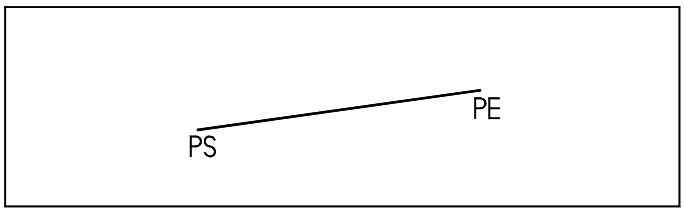
8.1.8 線分の表示・消去

【関数名】

Dspline

【機能】

線分を表示または消去する。



【呼出し形式】

```
void Dspline(const FPOINT* ps, const FPOINT* pe)
```

【入力引数】

ps	線分の始点座標
pe	線分の終点座標

注意
使用に先だって Dspinit 関数で環境設定しなければならない。
補助座標系で指定する。

8.1.9 線幅の設定

【関数名】

Dsplwt

【機能】

線幅を設定する。

【呼出し形式】

```
void Dsplwt(int lwt)
```

【入力引数】

lwt　　線幅番号。1-MAXITMLWT (default==1)

注意
使用に先だって Dspinit 関数で環境設定しなければならない。

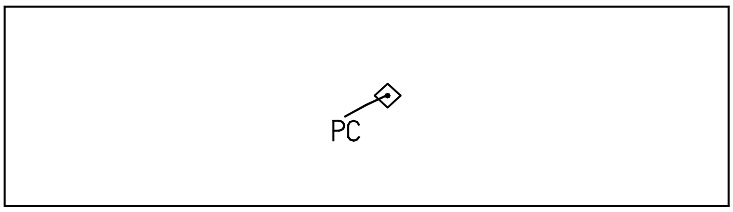
8.1.10 マークの表示・消去

【関数名】

Dspmark

【機能】

マークを表示または消去する。



【呼出し形式】

void Dspmark(const FPOINT* pc, int mark)

【入力引数】

pc　　マークの中心点
mark　　マーク番号

番号	形	大きさ (画面上のミリ)
1	□	4
5	X	4
6	Y	4
14	・	0.5
15	＋	2
16	＊	2
17	○	2
18	X	2
101	◇	2

注意
使用に先だって Dspinit 関数で環境設定しなければならない。
補助座標系で指定する。

8.1.11 メッセージの表示

【関数名】

Dspmsg

【機能】

メッセージを表示する。
この関数はラスター座標系のときに有効。

【呼出し形式】

```
void Dspmsg(int row, int col, int msg, int type, const void* data)
```

【入力引数】

row	行番号
col	列番号
msg	メッセージ番号。0 : メッセージなし
type	データタイプ メッセージの後に表示するデータのタイプ。 0 : なし 1 : short 型 (16 bits) 2 : float 型 3 : double 型 4 : int 型 n*10 : char 型 (n は文字数)
data	データ

注意
使用に先だって Dspinit 関数で環境設定しなければならない。

8.1.12 メッセージの消去

【関数名】

Dspmsgers

【機能】

メッセージを消す。
この関数はラスター座標系のときに有効。

【呼出し形式】

```
void Dspmsgers(int row, int col)
```

【入力引数】

row	行番号。0 : 全ての行
col	列番号。0 : 全ての列

注意

使用に先だって Dspinit 関数で環境設定しなければならない。

8.1.13 プレーンの選択

【関数名】

Dspscrn

【機能】

プレーンを選択する。

【呼出し形式】

```
void Dspscrn(int iscr)
```

【入力引数】

iscr　　プレーンの番号
- 0　：テンポラリ図形用プレーン
- 1　：実図形用プレーン
- 2　：グリッド用プレーン
- 3　：ラバーバンド用プレーン

注意
使用に先だって Dspinit 関数で環境設定しなければならない。

8.1.14 自由曲線の表示・消去

【関数名】

Dspspln

【機能】

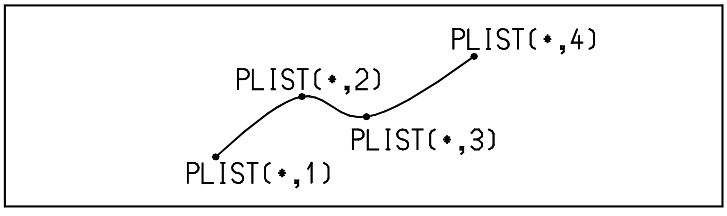
自由曲線を表示または消去する。

【呼出し形式】

```
void Dspspln(const FPOINT plist[], int n, int cswt)
```

【入力引数】

plist　　自由曲線の通過点列 (FPOINT array)
n　　点数。最小は 3、最大はヘッダーファイル acadprm.h 内の MAXSPLPNT。
cswt　　開曲線か閉曲線かの指定。0 : 開曲線、1 : 閉曲線



注意

使用に先だって Dspinit 関数で環境設定しなければならない。
閉曲線の時、終点を含んでも (終点＝始点)、含まなくてもよい。
補助座標系で指定する。

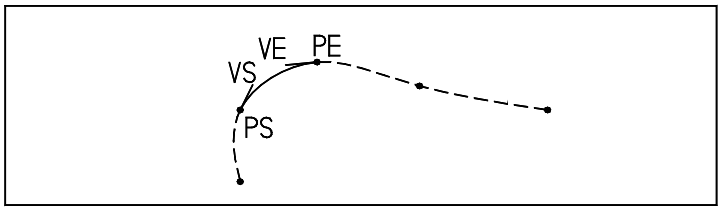
8.1.15 3 次 Bezier 曲線の表示・消去

【関数名】

Dspspls

【機能】

3 次 Bezier 曲線を表示または消去する。



【呼出し形式】

```
void Dspspls(const FPOINT* ps, const FPOINT* vs, const FPOINT* ve, const FPOINT* pe)
```

【入力引数】

ps	3 次 Bezier 曲線の始点
vs	3 次 Bezier 曲線の始点側のバーテックス点
ve	3 次 Bezier 曲線の終点側のバーテックス点
pe	3 次 Bezier 曲線の終点

注意

使用に先だって Dspinit 関数で環境設定しなければならない。
補助座標系で指定する。

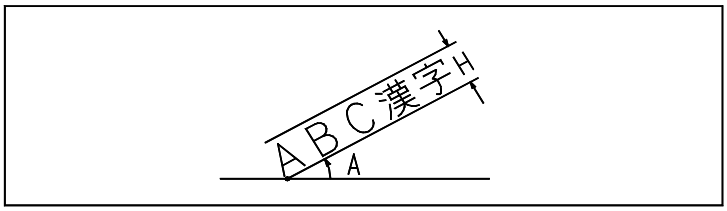
8.1.16 文字列の表示・消去

【関数名】

Dsptext

【機能】

文字列を表示または消去する。



【呼出し形式】

```
void Dsptext(const FPOINT* ps, double h, double a, const char* text)
```

【入力引数】

ps	文字列表示位置（左下）
h	文字高さ
a	表示角度（度）
text	文字列 (NULL-terminated)。256 バイト以下

注意

使用に先だって Dspinit 関数で環境設定しなければならない。
補助座標系で指定する。

例 . 文字列 "ABC 漢字 " を左下座標 (10,5)、高さ 10、角度 30 度で表示する。

```
FPOINT ps = {10.0, 5.0};
dsptext(&ps, 10.0, 30.0, "ABC 漢字 ");
```

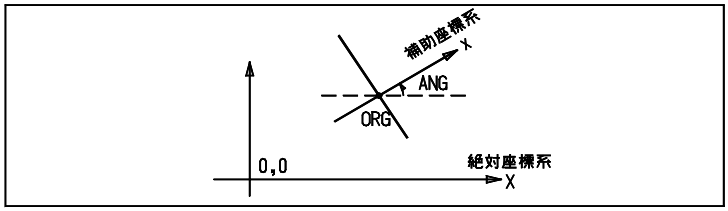
8.1.17 補助座標系を設定

【関数名】

Dspwcs

【機能】

補助座標系を設定する。



【呼出し形式】

```
void Dspwcs(const FPOINT* org, double ang)
```

【入力引数】

org	補助座標原点。(default==0,0)

ang　　補助座標 X 軸の傾き角度（度）。(default==0)

注意
使用に先だって Dspinit 関数で環境設定しなければならない。
原点、傾き角ともに絶対座標系で指定する。
補助座標を設定後、次の関数は補助座標系で処理する。
Dsparc, Dsparc3, Dspline, Dspmark, Dspspln, Dspspls, Dsptext
次の関数は常に絶対座標系で処理する。
Dsperas, Dspzone

例

(1) 絶対座標 (10,20) を原点、X 軸の傾き角を 30 度の補助座標を設定する。

```
FPOINT org = {10.0, 20.0};
Dspwcs(&org, 30.0);
```

(2) 絶対座標系に戻す。

```
FPOINT org = {0.0, 0.0};
Dspwcs(&org, 0.0);
```

8.1.18 表示領域座標を取得

【関数名】

Dspzone

【機能】

表示領域座標を得る。
Dspinit で指定した座標系に基づいた表示領域の座標を得る。

【呼出し形式】

```
void Dspzone(FRECT *zone)
```

【出力引数】

zone　　表示領域の絶対座標
zone->pmin　　：　左下座標
zone->pmax　　：　右上座標

注意
使用に先だって Dspinit 関数で環境設定しなければならない。

第 9 章　アイテム作成・編集モジュール

9.1　製図アイテム作成用定数

製図アイテム作成用定数の設定および取得をする関数。

- **内容 (値の範囲)**
  設定関数
  取得関数

9.1.1　テキストパラメータ

- **英数書体 (1-99、102-125)**
```
void  SetNtextfont(int ival)
short Ntextfont(void)
```

- **日本語書体 (101-125)**
```
void  SetNkanjifont(int ival)
short Nkanjifont(void)
```

- **文字高さ (0.0-327.0 mm)**
```
void  SetTexthight(float fval)
float Texthight(void)
```

- **文字角度 (0.0-360.0 度)**
```
void  SetTextangle(float fval)
float Textangle(void)
```

- **文字原点水平基準位置 (0= 左、1= 中央、2= 右)**
```
void  SetNtextadh(int ival)
short Ntextadh(void)
```

- **文字原点垂直基準位置 (0= 下、1= 中央、2= 上)**
```
void  SetNtextadv(int ival)
short Ntextadv(void)
```

- **水平方向文字間隔 (0-63)**
```
void  SetNtextdh(int ival)
short Ntextdh(void)
```

- **垂直方向文字間隔（行間隔）(0-63)**
```
void  SetNtextdv(int ival)
```

```
short Ntextdv(void)
```

- 文字傾斜角度 (-63 から 63 度)

```
void  SetNtextslant(int ival)
short Ntextslant(void)
```

- 文字枠表示モード (0= なし、1= 枠、2= 下線、3= 枠と下線、4= 可変長下線)

```
void  SetNtextwaku(int ival)
short Ntextwaku(void)
```

- 文字枠ゆとり（横）(0.0-327.0 mm)

```
void  SetTboxdhabax(double fval)
double Tboxdhabax(void)
```

- 文字枠ゆとり（縦）(0.0-327.0 mm)

```
void  SetTboxdhabay(double fval)
double Tboxdhabay(void)
```

- 文字 X 軸反転 (0= 反転しない、1= 反転する)

```
void  SetNtxtmirrx(int ival)
short Ntxtmirrx(void)
```

- 文字 Y 軸反転 (0= 反転しない、1= 反転する)

```
void  SetNtxtmirry(int ival)
short Ntxtmirry(void)
```

- 文字の向き (0= 横書き、1= 縦書き)

```
void  SetNtxttate(int ival)
short Ntxttate(void)
```

- 文字表示水平基準 (0= 左、1= 中央、2= 右)

```
void  SetNtextjsth(int ival)
short Ntextjsth(void)
```

- 行幅整列係数 (0-100 %)

```
void  SetNtextalign(int ival)
short Ntextalign(void)
```

- 文字縦横比 (0.1-10.0)

```
void  SetTextratio(double fval)
double Textratio(void)
```

9.1.2 寸法値パラメータ

- 単寸法／両寸法モード (0= 単一、1=mm/inch、2=inch/mm)

```
void  SetNdimdual(int ival)
short Ndimdual(void)
```

- 長さ寸法の表示形式 (0= 十進、1=feet&inch、2=feet、3=inch)

```
void  SetNdimftyp(int ival)
short Ndimftyp(void)
```

- 長さ寸法の単位 (1=mm、2=cm、3=meter、4=inch、5=feet)

```
void  SetNdimunit(int ival)
short Ndimunit(void)
```

● 長さ寸法の単位表示 (0= 表示しない、1= シンボリック、2= 省略文字)
```
void  SetNdimurep(int ival)
short Ndimurep(void)
```

● 長さ寸法の分数表示の最小単位（長さ寸法の表示形式が十進でないとき）
(0=1、1=1/2、2=1/4、3=1/8、4=1/6、5=1/32、6=1/64)
```
void  SetNdimfres(int ival)
short Ndimfres(void)
```

● 長さ寸法の十進小数桁数（長さ寸法の表示形式が十進のとき）(0-12)
```
void  SetNaccudim(int ival)
short Naccudim(void)
```

● 角度寸法の表示形式
(0= 度分秒、1= 十進で表示単位は Deg、
2= 十進で表示単位なし、3= 十進で表示単位は °)
```
void  SetNadimfrt(int ival)
short Nadimfrt(void)
```

● 角度寸法の度分秒の最小単位（角度寸法の表示形式が度分秒のとき）
(1= 度、2= 度分、3= 度分秒)
```
void  SetNadimfrac(int ival)
short Nadimfrac(void)
```

● 角度寸法の十進小数桁数（角度寸法の表示形式が十進のとき）(0-12)
```
void  SetNaccuadim(int ival)
short Naccuadim(void)
```

● 十進整数部３桁くぎり記号 (0= なし、1= カンマ、2= 空白、3= 点)
```
void  SetNdimmetrc(int ival)
short Ndimmetrc(void)
```

● 十進小数点 (0= 点、1= カンマ)
```
void  SetNdimfracd(int ival)
short Ndimfracd(void)
```

● 十進小数部の後ろの 0 の除去 (0= 除去する、1= 除去しない)
```
void  SetNdimsupz(int ival)
short Ndimsupz(void)
```

● 寸法文字列の表示 (0= 水平：寸法線の間に表示、1= 平行：寸法線の上に表示)
```
void  SetNdimorient(int ival)
short Ndimorient(void)
```

● 直径寸法の寸法補助記号
(0=φxxx、1=xxxφ、2=DIAxxx、3=xxxDIA、
4=φxxx、5=xxxφ、6=DIAxxx、7=xxxDIA：xxx は寸法値)
0 ～ 3 は「直径寸法」と「長さ寸法の φ 追加」の両方に適用。
4 ～ 7 は「長さ寸法の φ 追加」に適用。
```
void  SetNddimmark(int ival)
short Nddimmark(void)
```

● 半径寸法の寸法補助記号 (0=Rxxx、1=xxxR：xxx は寸法値)
```
void  SetNrdimmark(int ival)
```

```
short Nrdimmark(void)
```

● 寸法値倍率（角度寸法を除く）(1.0e-8 以上)

```
void   SetDimscale(double dval)
double Dimscale(void)
```

9.1.3 公差値パラメータ

● ±寸法許容差の文字高さ (0.0-327.0 mm)

```
void  SetTol2hight(double fval)
double Tol2hight(void)
```

● 上下寸法許容差の文字高さ (0.0-327.0 mm)

```
void  SetTollhight(double fval)
double Tollhight(void)
```

● 寸法許容差の小数桁数 (0-12)

```
void  SetNaccutol(int ival)
short Naccutol(void)
```

● 寸法許容差の十進小数部の後ろの 0 の除去 (0= 除去する、1= 除去しない)

```
void  SetNtolsupz(int ival)
short Ntolsupz(void)
```

● デフォルトの±寸法許容差文字列 (最大 32 文字、Null-char terminated)

```
void  SetMtexttolul(const char* cval)
const char* Mtexttolul(void)
```

● デフォルトの上寸法許容差文字列 (最大 32 文字、Null-char terminated)

```
void  SetMtexttolup(const char* cval)
const char*  Mtexttolup(void)
```

● デフォルトの下寸法許容差文字列 (最大 32 文字、Null-char terminated)

```
void  SetMtexttollw(const char* cval)
const char*  Mtexttollw(void)
```

9.1.4 寸法線、寸法補助線パラメータ

● 寸法線の矢の種類 (0= なし、1 ～ n= マーク番号)

```
void  SetNarrowter(int ival)
short Narrowter(void)
```

● 寸法線の矢の大きさ (0.0-327.0 mm)

```
void  SetSizemark(float fval)
float Sizemark(void)
```

● 寸法線スタブ長さ (1.0-320.0 mm)

```
void  SetStubsize(double fval)
double Stubsize(void)
```

● 寸法補助線ギャップ (任意の実数 mm)

```
void  SetExtlgap(double fval)
double Extlgap(void)
```

- 寸法補助線オーバーシュート (任意の実数 mm)

```
void  SetExtlover(double fval)
double Extlover(void)
```

9.1.5 その他の寸法関係パラメータ

- 並列寸法の間隔 (任意の実数 mm)

```
void  SetSizebdimh(double fval)
double Sizebdimh(void)
```

- 半径寸法の円中心側引出線長さ (任意の実数 mm)

```
void  SetSizedmrext(double fval)
double Sizedmrext(void)
```

- 累進寸法の起点記号のマーク番号 (0= なし、1 ～ n= マーク番号)

```
void  SetMarkodm(int ival)
short Markodm(void)
```

9.1.6 リファレンスノート、マークおよび引出線パラメータ

- 風船マークの番号 (0= なし、1 ～ n= マーク番号)

```
void  SetNreftype(int ival)
short Nreftype(void)
```

- 風船マークの大きさ (0.0-327.0 mm)

```
void  SetRefrad(double fval)
double Refrad(void)
```

- 風船引出線の矢の番号 (0= なし、1 ～ n= マーク番号)

```
void  SetNrefawmark(int ival)
short Nrefawmark(void)
```

- 風船引出線の矢の大きさ (0.0-327.0 mm)

```
void  SetSizerefaw(double fval)
double Sizerefaw(void)
```

- 風船の文字の大きさ (0.0-327.0 mm)

```
void  SetSizereftxt(double fval)
double Sizereftxt(void)
```

- 面の指示記号の大きさ (0.0-327.0 mm)

```
void  SetSizesmark(double fval)
double Sizesmark(void)
```

- 溶接記号の大きさ (0.0-327.0 mm)

```
void  SetSizewmark(double fval)
double Sizewmark(void)
```

- 引出線の最初の線分の丸め角度 (0-360 度)

```
void  SetNldrang(int ival)
short Nldrang(void)
```

- 引出線の最後の線分の丸め角度 (0.0-360.0 度)

```
void  SetRoundldr(double fval)
double Roundldr(void)
```

- 引出線の矢の番号 (0= なし、1 ～ n= マーク番号)

```
void  SetMarkldrarw(int ival)
short Markldrarw(void)
```

- 引出線の矢の大きさ (0.0-327.0 mm)

```
void  SetSizeldrarw(double fval)
double Sizeldrarw(void)
```

- 面の指示記号の文字の大きさ (0.0-327.0 mm)

```
void  SetSzsmarktxt(double fval)
double Szsmarktxt(void)
```

- 溶接記号の文字の大きさ (0.0-327.0 mm)

```
void  SetSzwmarktxt(double fval)
double Szwmarktxt(void)
```

9.1.7 幾何公差パラメータ

- 幾何公差記入枠の行の高さ (0.0-327.0 mm)

```
void  SetSizefcsbox(double fval)
double Sizefcsbox(void)
```

- 幾何公差記号の大きさ (0.0-327.0 mm)

```
void  SetSizefcstxt(double fval)
double Sizefcstxt(void)
```

- 幾何公差の引出線の矢の大きさ (0.0-327.0 mm)

```
void  SetSizefcsarw(double fval)
double Sizefcsarw(void)
```

- 幾何公差の引出線の矢の番号 (0= なし、1 ～ n= マーク番号)

```
void  SetMarkfcsarw(int ival)
short Markfcsarw(void)
```

- データム文字記入枠の番号 (0= なし、1 ～ n= マーク番号)

```
void  SetMarkdtmbox(int ival)
short Markdtmbox(void)
```

- データムの引出線の矢の番号 (0= なし、1 ～ n= マーク番号)

```
void  SetMarkdtmarw(int ival)
short Markdtmarw(void)
```

- データムの文字の大きさ (0.0-327.0 mm)

```
void  SetSzfcsdtmtxt(double fval)
double Szfcsdtmtxt(void)
```

9.1.8 切断線パラメータ

- 切断線矢印の番号 (0= なし、1 ～ n= マーク番号)

```
void  SetMarksection(int ival)
short Marksection(void)
```

● 切断線矢印の大きさ (0.0-327.0 mm)

```
void  SetSizesctmark(double fval)
double Sizesctmark(void)
```

● 切断線文字の大きさ (0.0-327.0 mm)

```
void  SetSizescttxt(double fval)
double Sizescttxt(void)
```

● 切断線ラインの表示 (0= なし、1= 表示、2= 折れ曲がり部のみ表示)

```
void  SetNsectline(int ival)
short Nsectline(void)
```

9.1.9 作表パラメータ

● 作表の枠高さ (任意の実数 mm)

```
void  SetWakuhight(double fval)
double Wakuhight(void)
```

● 作表の文字高さ (0.0-327.0 mm)

```
void  SetWakutsize(double fval)
double Wakutsize(void)
```

● 作表の数値部桁数 (負の値：小数部の後ろの 0 除去。桁数は絶対値をとる)

```
void  SetNwakuacc(int ival)
short Nwakuacc(void)
```

● 作表の数値カラム数 (任意の整数)

```
void  SetNwakuacc(int ival)
short Nwakuacc(void)
```

● 作表の面積倍率 (任意の実数)

```
void   SetDareaoff(double dval)
double Dareaoff(void)
```

9.1.10 中心線パラメータ

● 中心線のオーバーシュートの形式
(0= モデル座標系での長さ、1= ドローイング座標系での長さ、2= 半径の比率)

```
void  SetClinovtype(int ival)
short Clinovtype(void)
```

● 中心線のオーバーシュート長さ
(中心線のオーバーシュートの形式が 0 または 1 のときは 任意の実数 mm、中心線のオーバーシュートの形式が 2 のときは -0.5 ～ 0.5)

```
void  SetClinovsize(double fval)
double Clinovsize(void)
```

9.2 製図アイテム

● 関数一覧

ラベルおよび寸法アイテムの作成

関数名	機能
drfdglb	ジェネラルラベルアイテムを作る。
drfdgnt	ジェネラルノートアイテムを作る。
drfdrfn	リファランスノートまたはリファランスラベルアイテムを作る。
Drfexp	製図アイテムを分解して複合アイテムを作る。
Drfxadim	角度寸法アイテムを作る。
Drfxddim	直径寸法アイテムを作る。
Drfxldim	長さ寸法アイテムを作る。
Drfxodim	オーディネイト寸法アイテムを作る。
Drfxrdim	半径寸法アイテムを作る。
Drfxmrun	累進寸法 (Running dimensionning) アイテムを作る。
Dmoutput	寸法アイテムを作る。
gmgenafl	ぬりつぶしアイテムを作成する。
gmexpafl	ぬりつぶしアイテムを分解する。

ユーティリティ

関数名	機能
Drfucvdp	倍精度実数を文字列に変換する。
Drfucvtxt	寸法アイテムの寸法数値の文字列を作る
Drfudtgen	寸法文字列を作る。
Drfulab	ラベルを作る。

● TextDim 構造体

寸法文字列 (寸法値文字列、寸法公差文字列、付加文字列) を関数に渡すときに使用する。
構造体で以下の構造体メンバーを持ちます。

関数名	機能
val	寸法値文字列と文字数 (64 文字以下)
ultol	上下の寸法許容差の文字列と文字数 (32 文字以下)
utol	上の寸法許容差の文字列と文字数 (32 文字以下)

関数名	機能
ltol	下の寸法許容差の文字列と文字数（32 文字以下）
post	付加文字列と文字数（64 文字以下）

9.2.1 ジェネラルラベルアイテムを作成

【関数名】

drfdglb

【機能】

ジェネラルラベルアイテムを作る。

【呼出し形式】

```
void drfdglb(const char* txtstr, const FPOINT pnts[], int npnt)
```

【入力引数】

txtstr　文字列 (Null-char terminated)。
　　複数行のときは各行のくぎりに改行 (ASCII 13) を入れる。
　　最後の行のあとには改行をつけない。
pnts　点列
npnt　点数 (2-32)

例

```
SetItemtype(ITMTEXT);
TmpgOpen1(0, 0)
tmpdglb(省略);
TmpgClose(1);
```

9.2.2 ジェネラルノートアイテムを作成

【関数名】

drfdgnt

【機能】

ジェネラルノートアイテムを作る。

【呼出し形式】

```
void drfdgnt(const char* txtstr, const FPOINT pnts[], int iswt)
```

【入力引数】

txtstr　文字列 (Null-char terminated)。
　　複数行のときは各行のくぎりに改行 (ASCII 13) を入れる。
　　最後の行のあとには改行をつけない。
pnts　点列

iswt　　フラグ (1-7)。意味は以下の表による。

注意
文字列角度と文字高さの決めかた - 入力点数による

iswt	点数	文字列角度	文字の大きさ
1	1	RVP/DRF で設定した値	RVP/DRF で設定した値
2,6	2	第１点から第２点の角度	RVP/DRF で設定した値
3,7	3	第１点から第２点の角度	自動調整
4	2	第１点から第２点の角度	自動調整
5	3	第１点かから第２点の角度	自動調整

文字列の角度の範囲

iswt		文字列角度	
2-5		-85° ＜ angle ≦ 95°	
6-7		-180° ＜ angle ≦ 180°	

例
```
SetItemtype(ITMTEXT);
TmpgOpen1(0, 0);
drfdgnt(省略);
TmpgClose(1);
```

9.2.3　リファランスノートまたはリファランスラベルアイテムを作成

【関数名】

drfdrfn

【機能】

リファランスノートまたはリファランスラベルアイテムを作る。

【呼出し形式】

```
void drfdrfn(const FPOINT pnts[], int npnt, const char* text, double txtang)
```

【入力引数】

pnts　　引出し線の点列
npnt　　引出し線の点数 (1-32)。
　　１点だけを与えると引出し線なしで、その位置にリファランスノートを作る。
　　２点以上与えると引出し線を付け、最後の点の位置にリファランスノートを作る。引出し線の端は風船マークのエッジに乗るように調整される。
text　　文字列 (Null-char terminated)
txtang　　文字列の角度（単位は度）

例

```
SetItemtype(ITMTEXT);
TmpgOpen1(0, 0);
drfdrfn(省略);
TmpgClose(1);
TmpgClose(1);
```

9.2.4 製図アイテムを分解して複合アイテムを作成

【関数名】

Drfexp

【機能】

製図アイテムを分解して複合アイテムを作る。

【呼出し形式】

```
void Drfexp(const int idptrs[], int icnt, int* nitem)
```

【入力引数】

idptrs	アイテム識別子の並び
icnt	アイテムの数

【出力引数】

nitem	複合アイテムの数

注意

(1) 製図アイテム

ITMTEXT	General note / label / Section line
ITMMARK	General Mark
ITMDIM	Dimension
ITMFCS	FCS
ITMHATCH	Hatching

(2) 対象サブレコードシーケンス

SITPOINT
　　点アイテムに変換
SITSTART, { SITLINE, SITARC, SITBZCRV }+
　　ストリングアイテムに変換
SITTXTPRM, SITTMBOX, SITTEXT
　　１文字が１つのストリングアイテムになる。
　　一筆画きにするために、不連続な所は非表示の線分で連結。
SITTMBOX, SITMRKPRM
　　スナップノードは点アイテムになる。
　　それ以外は１つのストリングアイテム。
　　一筆画きにするために、不連続な所は非表示の線分で連結。

(3) １のアイテムを分解してできたアイテムをすべて含む複合アイテムを作る。
もとのアイテムは削除する。

9.2.5 製図アイテムを分解して複合アイテムを作成

【関数名】

Drfxadim

【機能】

角度寸法アイテムを作る。

【呼出し形式】

```
void Drfxadim(int dimtyp, int subtyp, short iswt, const DPOINT* pvtx, const DLINE alin[],
              const DPOINT* porg, const TextDim* dt)
```

【入力引数】

dimtyp	寸法の種類。4 : ３点 , 5 : ２直線
subtyp	寸法の作成方法。0 : 単一寸法 , 1 : 直列寸法 , 2 : 並列寸法
iswt	寸法線 (Arrow line) と寸法補助線 (Witness line) の抑止 bit i = 1 のとき抑止。各ビットの意味は以下のとおり。 iswt = 0 はすべて表示となる。 bit 1 : 第１寸法補助線 bit 2 : 第２寸法補助線 bit 3 : 第１寸法線 bit 4 : 第２寸法線 bit 5 : 第３寸法線 bit 6 : 第４寸法線
pvtx	角度寸法の頂点
alin	角度寸法の開始角線と終了角線 (Pnt,Vec) Pnt は寸法参照点。Vec は頂点から参照点へ向かうベクトル。 pvtx == Pnt となってもよい。
porg	寸法値の位置
dt	寸法文字列 (寸法値文字列、寸法公差文字列、付加文字列)

注意

この関数は、角度寸法アイテムを作成するためにコモンに寸法パラメータを設定するだけである。
アイテムを作るにはつぎのようにする。

```
SetItemtype(ITMDIM);
TmpgOpen1(0,0)
Ddrfxadim( 省略 );
Dmoutput();
TmpgClose(1);
```

9.2.6　直径寸法アイテムを作成

【関数名】

Drfxddim

【機能】

直径寸法アイテムを作る。

【呼出し形式】

```
void Drfxddim(short iswt, double dscf, const DPOINT* pcen,
              const DPOINT pnts[], int npnt, const TextDim* dt)
```

【入力引数】

iswt	寸法線（Arrow line）と寸法補助線（Witness line）の抑止 bit i = 1 のとき抑止。各ビットの意味は以下のとおり。 iswt = 0 はすべて表示となる。 bit 1 : 第１寸法補助線 bit 2 : 第２寸法補助線 bit 3 : 第１寸法線 bit 4 : 第２寸法線 bit 5 : 第３寸法線 bit 6 : 第４寸法線
dscf	寸法値倍率
pcen	円の中心点
pnts	引出線点列。最初の点は円周にあること。
npnts	点数 npnts == 2 のときは　標準直径寸法 npnts == 3 のときは外側の寸法線を折り曲げる。これは JIS モードで寸法テキストが円の外側のときだけ有効。そうでないときは npnts == 2 の場合と同じ。
dt	寸法文字列（寸法値文字列、寸法公差文字列、付加文字列）

注意

このモジュールは直径寸法アイテムを作成するためにコモンに寸法パラメータを設定するだけである。アイテムを作るにはつぎのようにする。

```
SetItemtype(ITMDIM);
TmpgOpen1(0, 0);
Drfxddim(省略);
Dmoutput();
TmpgClose(1);
```

9.2.7　長さ寸法アイテムを作成

【関数名】

Drfxldim

【機能】

長さ寸法アイテムを作る。

【呼出し形式】

```
void Drfxldim(int dimtyp, int subtyp, short iswt, double dscf, const DPOINT pnts[],
              const TextDim* dt)
```

【入力引数】

dimtyp	寸法の種類。1 : DMH, 2 : DMV, 3 : DMP
subtyp	寸法線の作成方法。0 : 単一寸法，1 : 直列寸法，2 : 並列寸法， 4 : 片側寸法
iswt	寸法線と寸法補助線の抑止 bit i == 1 のとき抑止。各ビットの意味は以下のとおり。 iswt == 0 はすべて表示となる。 bit 1 : 第１寸法補助線 bit 2 : 第２寸法補助線 bit 3 : 第１寸法線 bit 4 : 第２寸法線 bit 5 : 第３寸法線

bit 6 : 第４寸法線
dscf 寸法値倍率
pnts 寸法参照点と寸法値テキスト位置。
第１点、第２点は寸法参照点、第３点は寸法値テキスト位置。
dt 寸法文字列（寸法値文字列、寸法公差文字列、付加文字列）

注意
このモジュールは直線寸法アイテムを作成するためにコモンに寸法パラメータを設定するだけである。アイテムを作るにはつぎのようにする。

```
SetItemtype(ITMDIM);
TmpgOpen1(0, 0);
DrfxIdim(省略);
Dmoutput();
TmpgClose(1);
```

例

```
int keyrpl;
TextDim dt;

/* Dimesioning point and text origin */
DPOINT pnts[3] = { {100.0,   20.0}, {200.0,   50.0}, {150.0, 100.0}};

/* Dimension text and post fix text (Convert scalar to character string)
double dimval = pnts[1].x - pnts[0].x;
Drfudtgen("*", &dimval, &keyrpl, &dt.val, &dt.post);

/* Tolerance text - No tolerancing */
dt.ultol.tlen = 0;
dt.utol.tlen  = 0;
dt.ltol.tlen  = 0;

/* Generate a Linear dimension item */
SetItemtype(ITMDIM);
TmpgOpen1(0, 0);
DrfxIdim(1, 0, 0, 1.0, pnts, &dt);
Dmoutput();
TmpgClose(1);
```

9.2.8 オーディネイト寸法アイテムを作成

【関数名】

Drfxodim

【機能】

オーディネイト寸法アイテムを作る。

【呼出し形式】

```
void Drfxodim(int odmtyp, int numdim, const double dscf[], const DPOINT pdim[],
              const DPOINT* porg, const TextDim dt[])
```

【入力引数】

odmtyp 寸法の種類。1 : ODH, 2 : ODV
numdim 寸法の数。

dscf　　寸法値倍率の配列 (double dscf[numdim])。
pdim　　寸法起点と numdim 個の寸法点 (DPOINT pdim[numdim + 1])
porg　　寸法値の位置
dt　　寸法文字列 (寸法値文字列、寸法公差文字列、付加文字列)

注意
この関数はオーディネイト寸法アイテムを作成する。テンポラリアイテムができる。

```
SetItemtype(ITMDIM);
Drfxodim(省略);
```

9.2.9　半径寸法アイテムを作成

【関数名】

Drfxrdim

【機能】

半径寸法アイテムを作る。

【呼出し形式】

```
void Drfxrdim(double dscf, const DPOINT* pcen, const DPOINT pnts[], int npnt,
              int ltyp, const DPOINT* pend, const TextDim* dt)
```

【入力引数】

dscf　　寸法値倍率
pcen　　円の中心点
pnts　　主引出線点列。最初の点は円周にあること。
npnt　　点数
ltyp　　反対側のリーダを付けるかどうか
　　0　　：反対側のリーダなし
　　1　　：反対側のリーダを付ける (矢印なし)
　　2　　：反対側のリーダを付ける (矢印付き)
pend　　反対側のリーダの端点。
　　このリーダの始点は主引出線の第１点を使用する。
　　ltyp == 1, 2 のときだけ与える。
dt　　寸法文字列 (寸法値文字列、寸法公差文字列、付加文字列)

注意
この関数は半径寸法アイテムを作成するためにコモンに寸法パラメータを設定するだけである。アイテムを作るにはつぎのようにする。

```
SetItemtype(ITMDIM);
TmpgOpen1(0, 0);
Drfxrdim(省略);
Dmoutput();
TmpgClose(1);
```

9.2.10　累進寸法アイテムを作成

【関数名】

Drfxmrun

【機能】

累進寸法 (Running dimensionning) アイテムを作る。

【呼出し形式】

```
void Drfxmrun(int dimtyp, int subtyp, int numdim, const double dscf[],
              const DPOINT pnts[], const DPOINT* porg, const TextDim dt[])
```

【入力引数】

dimtyp	寸法の種類。1 : 水平 (abscissa), 2 : 垂直 (ordinate)
subtyp	起点寸法抑止スイッチ。0 : 起点寸法も記入, 1 : 起点寸法は記入しない
numdim	寸法の数
dscf	寸法値倍率の配列 (double dscf[numdim])
pnts	起点と寸法点 (DPOINT pnts[numdim + 1])。最初の点は起点。
porg	寸法線の位置
dt	寸法文字列 (寸法値文字列、寸法公差文字列、付加文字列) の並び (TextDim dt[numdim + 1]) 最初の寸法テキストは起点の寸法値。JIS では起点記号を白丸とするときは寸法値 '0' は記述しないことになっている。このときは、空白文字 (space) 1 文字を設定する。

注意

この関数は累進寸法アイテムを作成する。テンポラリアイテムができる。

```
SetItemtype(ITMDIM);
Drfxmrun( 省略 );
```

9.2.11 寸法アイテムを作成

【関数名】

Dmoutput

【機能】

寸法アイテムを作る。Drtxadim, Drfxddim, Drfxldim, Drfxrdim の後に呼出し、テンポラリアイテムとして寸法アイテムを作る。

【呼出し形式】

```
void Dmoutput(void)
```

9.2.12 倍精度実数を文字列に変換

【関数名】

Drfucvdp

【機能】

倍精度実数を文字列に変換する。

【呼出し形式】

```
int Drfucvdp(double dp, int maccu, int mtdlm, int mfdlm, int mtzer, char *cstr, int clen)
```

【入力引数】

```
dp        倍精度実数。abs(dp) ≦ 1.0e16
maccu     十進少数桁数 (0 ≦ maccu ≦ 12)。
             0        :少数部なし
mtdlm     十進整数部区切り記号。
             0        :なし
             1        :Comma (,)
             2        :Space
             3        :Dot (.)
mfdlm     十進少数記号 (INTEGER*2)。
             0        :Dot (.)
             1        :Comma (,)
mtzer     十進少数部の下の零 (0) の除去。
             0        :除去する
             1        :除去しない
clen      配列 cstr の長さ。
```

【出力引数】

```
cstr      寸法値文字列 (NULL-terminated)
```

【返り値】

0　　: 正常終了

9.2.13 整数を形式に従って文字列に変換

【関数名】

Drfucvtxt

【機能】

数値を形式に従って文字列に変換する。

【呼出し形式】

```
int Drfucvtxt(int iform, const double* scalar, char *cstr, int clen)
```

【入力引数】

```
iform     形式を示す番号
             0        :単純変換
             1        :長さ寸法
             4        :角度寸法。Nadimfrt() の値による。
                0              :  度(°)分(')秒(")表示
                1              :  十進度表示。寸法値の後に文字 Ｄｅｇ を付ける
                2              :  十進度表示 (単位表示なし)
             6        :半径寸法。Nrdimmark() の値による。
                0              :  寸法値の前に寸法補助記号 Ｒ を付ける (JIS)
                1              :  寸法値の後に寸法補助記号 Ｒ を付ける
                2              :  寸法補助記号なし
             7        :直径寸法。Nddimmark() の値による。
                0              :  寸法値の前に寸法補助記号 φ を付ける (JIS)
```

1 : 寸法値の後に寸法補助記号 φ を付ける
2 : 寸法値の前に文字 DIA を付ける
3 : 寸法値の後に文字 DIA を付ける
4 : 補助記号、文字なし
8 :長さ寸法許容差。許容値の前に符号±を付ける
9 :長さ上または下の寸法許容差
許容値の前に符号＋またはーを付ける
正の値の時許容値の前に符号＋を付ける
負の値の時許容値の前に符号ーを付ける
値が零に近いとき符号は付けずに空白を付ける
10 :寸法補助記号□。寸法値の前に記号□を付ける (JIS)
11 :一定寸法。寸法値の後ろに文字一定を付ける。
12 :45 度面取り寸法。寸法値の前に記号 C を付ける (JIS)
13 :角度寸法許容差
14 :角度寸法上または下の許容差
21 :座標 "(x, y)"

scalar 数値。iform == 21 の場合、X 座標, Y 座標を含む配列。
clen 配列 cstr の長さ。

【出力引数】

cstr 寸法値文字列 (NULL-terminated)

【返り値】

0 : OK
1 : 変換エラー

注意

(1) 寸法値の表現

十進少数桁数
十進整数部区切り記号
十進少数記号
十進少数部の下の零 (0) の除去
は各々 Naccudim(), Ndimmetrc(), Ndimfracd(), Ndimsupz() の現在値による。

(2) 寸法値の丸め

最後の桁において 0.5 以上は切り上げ。
たとえば、十進少数桁数が３のとき
123.4565 は 123.257
123.4564 は 123.256

(3) 単一寸法／両寸法

Ndimdual() の現在値による。これは長さ寸法 (iform == 1, 6, 7, 10,11, 12) のときだけ有効。
0 : 単寸法表示 (Metric)
1 : 両寸法表示 (Metric/English)
2 : 両寸法表示 (English/Metric)

9.2.14 寸法文字列を作成

【関数名】

Drfudtgen

【機能】

寸法文字列を作る。

【呼出し形式】

```
int Drfudtgen(const char* cstr, int keycnv, const double* dimval, int* keyrpl,
              TextDimVal* dtv, TextDimPst *dtp)
```

【入力引数】

cstr	入力文字列 (Null-char terminated)	
keycnv	寸法数値の変換形式	
	1	: 長さ寸法
	4	: 角度寸法
	6	: 半径寸法
	7	: 直径寸法
	10	: 寸法補助記号（角）付き長さ寸法
	11	: 一定付き長さ寸法
	12	: ４５度面取り寸法
	21	: 座標寸法
dimval	寸法数値（実数))。座標寸法 (keycnv == 21) のときは、点座標 (x,y) を含む配列。	

【出力引数】

keyrpl	寸法値文字列置換フラグ	
	0	: 寸法値文字列は寸法数値を含んでいる。
	1	: 寸法値文字列は入力文字列で置き換えられ、寸法数値を含まない。
dtv	寸法値文字列と文字数（バイト数）(1<= 文字数 <=64)	
dtp	寸法付加文字列と文字数（バイト数）(1<= 文字数 <=64)	

【返り値】

0 : 正常

補足

(1) 入力文字列で特別な意味を持つ文字

"&"	寸法値文字列と寸法付加文字列とを区切る文字。"&" 文字の前が寸法値文字列で、後が寸法付加文字列。最初に現れる "&" だけが有効。
"*"	寸法値文字列の最初に現れた文字 "*" は、寸法数値で置き換えられる。
"¥c"	文字 "c" を意味する。"&" や "*" は特別な意味を持つので、それを打ち消すために "¥" を前におく。"¥*" は文字 "*", "¥&" は文字 "&" ,"¥¥" は文字 "¥" を表わし通常の文字になる。(¥ はバックスラッシュ)
¥MDtext¥MZ	寸法値文字列内の寸法数値は特別な文字 (¥MD) と (¥MZ) で囲まれている。

(2) 入力寸法値文字列が、

寸法数値 (¥MDtext¥MZ) を含むとき

寸法数値は更新しない。

文字 "*" を含み寸法数値 (¥MDtext¥MZ) がないとき

文字 "*" が寸法数値で置き換えられる。

文字 "*" も寸法数値 (¥MDtext¥MZ) もないとき

寸法値文字列は入力文字列で置き換られ、寸法数値を含まない。

例

```
double dimval = 100.0;
TextDimVal dtv;
TextDimPst dtp;
int keyrpl;
Drfudtgen("ABC*DEF&GHI", 1, &dimval, &keyrpl, &dtv, &dtp);

/* この結果は以下のようになる。
dtv.txt  = "ABC¥MD100¥MZDEF";
dtv.tlen = 13;
dtp.txt  = "GHI";
dtp.tlen =  3;
```

```
*/
```

9.2.15 ラベル作成

【関数名】

Drfulab

【機能】

ラベルを作る。

【呼出し形式】

```
void Drfulab(const char* pfxtxt, int num, int lnum, char* label)
```

【入力引数】

pfxtxt	接頭語 (Null-char terminated)	
	接頭語なしにするには空文字列か null ポインタを渡す。	
num	番号。	
	num < 0	接頭語と番号の間にハイフン (-) を置く
	num ≧ 0	接頭語と番号の間にハイフン (-) なし
lnum	番号の桁数	
	lnum == 0	番号なし
	1 ≦ lnum ≦ 5	番号の前の零 (0) を除去しない。
	-5 ≦ lnum ≦ -1	番号の前の零 (0) を除去する。

【出力引数】

abtxt	ラベル (接頭語＋番号)(Null-char terminated)

例

つぎのように４種類のラベルを作ることができる。
接頭語 "PNT" で、
番号が正 (num == 1) の場合
　桁数 lnum == 3　　ラベル "PNT001"
　　lnum == -3　　ラベル "PNT1"
番号が負 (num == -1) の場合
　桁数 lnum == 3　　ラベル "PNT-001"
　　lnum == -3　　ラベル "PNT-1"

9.2.16 塗り潰しアイテムを作成

【関数名】

gmgenafl

【機能】

塗り潰しアイテム (Area fill item) を作成する。

【呼出し形式】

```
int gmgenafl(const int idptrs[], int nitm, const float* aflprm, int dupswt)
```

【入力引数】

idptrs	塗り潰しアイテムの境界を構成するアイテムのアイテム識別子の並び (int idptrs[nitm])。 境界を構成するアイテムはそれぞれ閉曲線でなければならない。円、閉自由曲線、閉ストリングアイテムまたは塗り潰しアイテムでもよい。
nitm	境界を構成するアイテムの数。
aflprm	塗り潰しアイテムのパラメータ 詳細はサブレコード 18 を参照のこと。
dupswt	塗り潰しアイテム作成後、もとのアイテムを削除するかどうか。 FALSE : 元のアイテムは削除する。 TRUE : 元のアイテムを残す。

【返り値】

0	: 正常
1	: エラー

9.2.17 塗り潰しアイテムの分解

【関数名】

gmexpafl

【機能】

塗り潰しアイテムを分解する。

【呼出し形式】

```
int gmexpafl(const int idptrs[], int icnt, int dupswt)
```

【入力引数】

idptrs	分解する塗り潰しアイテムのアイテム識別子の並び(int idptrs[icnt])
icnt	アイテムの数。
dupswt	塗り潰しアイテム分解後、その塗り潰しアイテムを削除するかどうか。 FALSE : 元のアイテムは削除する。 TRUE : 元のアイテムを残す。

【返り値】

nitem	塗り潰しアイテム分解してできたアイテムの数。

9.3 複合アイテム

● 関数一覧

関数名	機能
gmexpcmp	複合アイテムを分解する
gmgencmp	複合アイテムを作る

9.3.1 複合アイテムを分解

【関数名】

gmexpcmp

【機能】

```
int gmexpcmp(const int idptrs[], int icnt, int dupswt, int atrswt)
```

【機能】

複合アイテムを分解し構成アイテムを独立したアイテムとする。

【入力引数】

idptrs　複合アイテムの識別子のリスト（int idptrs[icnt]）
icnt　複合アイテムの数
dupswt　複合アイテムを残すかどうか
　FALSE　：　複合アイテムを消去する
　TRUE　：　複合アイテムを残す
atrswt　分解してできるアイテムの属性
　FALSE　：　もとのアイテムのアイテム属性
　TRUE　：　表示に基ずくアイテム属性

【返り値】

分解してできたアイテムの数。新しくできたアイテムはデータベースに格納される。

9.3.2 複合アイテム作成

【関数名】

gmgencmp

【機能】

複合アイテムを作成する。

【呼出し形式】

```
int gmgencmp(const int idptrs[], int icnt, int dupswt)
```

【入力引数】

idptrs	複合アイテムに含めるアイテムの識別子のリスト(int idptrs[icnt])	
icnt	アイテムの数	
dupswt	もとのアイテムを残すかどうか	
	FALSE	: アイテムを消去する
	TRUE	: アイテムを残す

【返り値】

0	: 正常終了
1	: エラー

注意
新しくできたアイテムはデータベースに格納される。

9.4 シンボル

● 関数一覧

関数名	機能
SymGenI	シンボルファイルを作成する
SymPutI	シンボルアイテムをテンポラリバッファに作成する。
SymFrfHdrI	シンボルファイルのヘッダー情報を参照する。
SymInpHedI	シンボルアイテムのヘッダー情報を参照する。
SymDspWinI	シンボルを指定ウインドウ領域内に表示する。
SymFrfNdpI	シンボルファイルのノードポイントを参照する。

9.4.1 シンボルファイルを作成

【関数名】

SymGenI

【機能】

シンボルファイルを作成する。

【呼出し形式】

```
int SymGenI(const char* name, int keygen, const DPOINT* org, const int ibstxt[],
            int iasclst[][5], int nasc)
```

【入力引数】

name　シンボル名 (NULL terminated)。シンボル名に、シンボルファイルの拡張子”.SYM”を含める必要はない。

keygen　シンボルファイルの作成モード
- 0　：アクティブリストのアイテムからシンボルファイルを作成する。
- ≧ 1　：モデルのピクチャ #keygen 上のアイテムからシンボルファイルを作成する。

org　シンボル原点

ibstxt　ベーステキストの指定 (int ibstxt[2])
- ibstxt[0]　：ベーステキストとするテキストセグメントを持つアイテムのアイテム識別子
- ibstxt[1]　：テキストセグメントの直前の分類サブレコードの相対番号。シンボルにベーステキストを持たせない場合は、ibstxt[0] と ibstxt[1] を 0 とする。

iasclst　ノードテキストの指定リスト (int iasclst[nasc][5])
- iasclst[][0]　：ノードテキストとするテキストセグメントを持つアイテムのアイテム識別子
- iasclst[][1]　：テキストセグメントの直前の分類サブレコードの相対番号
- iasclst[][2]　：ノードテキストを付加するポイントセグメントを持つアイテムのアイテム識別子
- iasclst[][3]　：ポイントセグメントの点サブレコードの相対番号
- iasclst[][4]　：ノードテキストの分類番号（分類番号 :1 ～ 3)

nasc　　ノードテキストの指定の総数。ノードポイントにノードテキストを付加しない場合は、nasc を 0 とする。ノードテキストの指定の総数 nasc の最大値は 768 である。

【返り値】

0　　：正常終了

注意

シンボルファイルの作成の対象となるアイテムに含まれる非表示のポイントセグメントは、すべてシンボルのノードポイントとなる。シンボルのノードポイントの最大数は 255 。これを越えるポイントセグメントがある場合は、シンボルファイルは作成されない。

以下の場合はシンボルファイルは作成されない。

- シンボルファイルに含めるアイテムが持つサブレコードの総数が 16384 を越える場合
- サブレコードデータのデータ量が 256 Kword を越える場合

ベーステキストおよびノードテキストとするテキストセグメントを持つアイテムは、シンボルファイルの作成の対象となるアイテムに含まれていなければならない。
同一のテキストセグメントをベーステキストと複数のノードポイントのノードテキストとすることはできない。
通常、グラフィックステキストアイテムが 1 つのテキストセグメントだけからなる場合、テキストセグメントの直前の分類サブレコードの相対番号は 1 になる。
また、点アイテムが 1 つのポイントセグメントだけからなる場合、ポイントセグメントの点サブレコードの相対番号は 1 になる。
IdentItem 関数 でアイテムをピックした場合、ピックされたアイテムのアイテム識別番号、テキストセグメントの直前の分類サブレコードの相対番号およびポイントセグメントの点サブレコードの相対番号は、IdentInfo 関数で得ることができる。

9.4.2　シンボルアイテムをテンポラリバッファに作成

【関数名】

SymPutI

【機能】

シンボルアイテムをテンポラリバッファに作成する。

【呼出し形式】

```
int SymPutI(const char* name, const DPOINT* org, const double scf[], double ang)
```

【入力引数】

name　　シンボル名 (NULL-terminated)。シンボル名に、シンボルファイルの拡張子 ".SYM" を含める必要はない。
org　　シンボル配置位置 (シンボル原点の位置)
scf　　シンボル縮尺値 (double scf[2])
　　scf[0]　　：X 縮尺値 (シンボルの X 軸方向に対する縮尺値)。
　　　　scf[0] が負である場合には、シンボルの Y 軸に対して反転する。
　　scf[1]　　：Y 縮尺値 (シンボルの Y 軸方向に対する縮尺値)
　　　　scf[1] が負である場合には、シンボルの X 軸に対して反転する。
　　拡大／縮小しないときは、scf[0] = scf[1] == 1.0 とする。
ang　　シンボル回転角度 (単位：度)

【返り値】

0　　：正常終了

注意
この関数は、指示された配置パラメタをもとにして、シンボルアイテムをテンポラリバッファに追加する。テンポラリバッファのオープンとクローズは、この関数内で行なわれているため、TmpgOpen1 と TmpgClose を使う必要はない。
シンボルアイテムを作るにはつぎのようにする。

```
TmpgWrtoDB(0);
SymPutl(iname, org, scf, ang);
TmpgWrtoDB(0);
```

9.4.3　シンボルファイルのヘッダー情報を参照

【関数名】

SymFrfHdrl

【機能】

シンボルファイルのヘッダー情報を参照する。

【呼出し形式】

```
int SymFrfHdrl(const char* name, short icdate[], char* cusrnme, DPOINT box[])
```

【入力引数】

name	シンボル名 (NULL-terminated)。シンボル名に、シンボルファイルの拡張子 ".SYM" を含める必要はない。

【出力引数】

idate	シンボルファイル作成日時 (short idate[6]) idate[0] : 日、idate[1] : 月、idate[2] : 年 idate[3] : 時、idate[4] : 分、idate[5] : 秒
cusrnme	シンボルファイルのユーザ名 (char cusrnme[17], NULL-terminated)
box	シンボルボックス (DPOINT box[4]) box[0] : シンボルボックスの左下の点 box[1] : シンボルボックスの右下の点 box[2] : シンボルボックスの左上の点 box[3] : シンボルボックスの右上の点

【返り値】

0　　：正常終了

9.4.4　シンボルアイテムのヘッダー情報を参照

【関数名】

SymInpHedl

【機能】

シンボルアイテムのヘッダー情報を参照する。

【呼出し形式】

```
int SymInpHedl(int idptr, char* name, short idate[], DPOINT* org, double scf[],
               double* ang, DPOINT box[]);
```

【入力引数】

idptr　　シンボルアイテムのアイテム識別子

【出力引数】

name　　シンボル名 (NULL-terminated)。
idate　　シンボルアイテム作成日時
　　idate[0] : 日、idate[1] : 月、idate[2] : 年
　　idate[3] : 時、idate[4] : 分、idate[5] : 秒
org　　シンボル配置位置（シンボル配置位置：シンボル原点の位置）
scf　　シンボル縮尺値 (double scf[2])
　　scf[0]　　: X 縮尺値（シンボルの X 軸方向に対する縮尺値）。負の場合は、シンボルの Y 軸に対して反転されていることを示す。
　　scf[1]　　: Y 縮尺値（シンボルの Y 軸方向に対する縮尺値）。負の場合は、シンボルの X 軸に対して反転されていることを示す。
ang　　シンボル回転角度（単位：度）
box　　シンボルボックス
　　box[0]　　: シンボルボックスの左下の点
　　box[1]　　: シンボルボックスの右下の点
　　box[2]　　: シンボルボックスの左上の点
　　box[3]　　: シンボルボックスの右上の点

【返り値】

0　　: 正常終了

9.4.5　シンボルを指定ウィンドウ領域内に表示

【関数名】

SymDspWinl

【機能】

シンボルを指定ウインドウ領域内に表示する。

【呼出し形式】

```
int SymDspWinl(const char* name, int keyscr, int keyers, int iwindsp, const short iwinzon[]);
```

【入力引数】

iname　　シンボル名。(NULL-terminated)。シンボル名にシンボルファイルの拡張子”.SYM”を含める必要はない。
keyscr　　表示プレーン制御スイッチ
　　0　　:単色表示。テンポラリ図形用プレーンにシンボルを表示する。
　　1　　:カラー表示。実図形用プレーンにシンボルを表示する。
keyers　　ウインドウ領域消去スイッチ
　　0　　:ウインドウ領域内を消去しないで、シンボルを表示する。
　　1　　:ウインドウ領域内を消去して、シンボルを表示する。

iwindsp　表示ウインドウ番号
　　1　　：グラフィックウインドウ
　　2　　：サブグラフィックウインドウ（ＭＳＣ エリア）
iwinzon　表示ウインドウ領域
　iwinzon[0]　　：　ウインドウ領域の左下 Ｘ 座標（ラスター）
　iwinzon[1]　　：　ウインドウ領域の左下 Ｙ 座標（ラスター）
　iwinzon[2]　　：　ウインドウ領域の右上 Ｘ 座標（ラスター）
　iwinzon[3]　　：　ウインドウ領域の右上 Ｙ 座標（ラスター）

【返り値】

0　　：正常終了

注意
以下はサブグラフィックウインドウ (MSC エリア) にシンボルを表示するときの例。

```
short iwinzon[16];
int iwinno = 2;
Menuzone(2, iwinno, iwinzon);
SymDspWinl(iname, keydsp, keyers, iwinno, iwinzon);
```

9.4.6　シンボルファイルのノードポイントを参照

【関数名】

SymFrfNdpl

【機能】

シンボルファイルのノードポイントを参照する。

【呼出し形式】

```
int SymFrfNdpl(const char* name, short* ndpnt, DPOINT pnts[]);
```

【入力引数】

name　　シンボル名。(NULL-terminated)。シンボル名に、シンボルファイルの拡張子 ".SYM" を含める必要はない。

【出力引数】

ndpnt　　シンボルファイルのノードポイント数 (0 ≦ NDPNT ≦ 255)
pnts　　シンボルファイルのノードポイントの座標 (DPOINT pnt[256])

【返り値】

0　　：正常終了

注意
シンボルファイルのノードポイントの座標は、シンボルの原点を (0,0) としたときの座標である。

9.5 モデル

● 関数一覧

関数名	機能
Mdlinit	モデルを初期化する。
Mdlwrite	アクティブモデルをモデルファイルに保存する。
Mdlread	既存のモデルファイルを読み込む。
Mdlread1	既存のモデルファイルを読み込む。
Mdlrplpic	モデルピクチャ書き込みを行なう。
Mdl101	モデルファイルのヘッダー情報を参照する。
Mdl102	アクティブモデルのモデル名またはモデル名ファイルを参照する。
Mdl103	モデルファイルのヘッダー情報をグラフィックウインドウに表示する。
Mdl104	モデルファイルのモデルタイトルを参照する。
Mdlname001	モデル名を設定／解除する。
Mdlname101	モデル名を得る。
Subgen	サブモデルファイルを作成する。
Sub101	サブモデルアイテムをテンポラリバッファに作成する。
Subput	サブモデルアイテムをデータベース内に作成する。

9.5.1 モデルの初期化

【関数名】

Mdlinit

【機能】

モデルを初期化する。

【呼出し形式】

```
int Mdlinit(void)
```

【返り値】

0　　：正常終了

9.5.2 アクティブモデルをモデルファイルに保存

【関数名】

Mdlwrite

【機能】

アクティブモデルをモデルファイルに保存する。

【呼出し形式】

```
int Mdlwrite(char* name, int mode, int keychk, int keyinf,
             int keypcvt, const short ipcvtbl[], bool picRef = false)
```

【入力引数】

name	モデル名 (NULL-terminated)	
	モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。	
mode	モデルの保存モード	
	-1	：アクティブリスト中のアイテムだけを保存する。アクティブリストのアイテムがあるピクチャについてだけピクチャ属性を保存する。
	0	：モデルの全てのピクチャを保存する。
	≧1	：保存するピクチャの番号。
keychk	作成モデルファイル検証スイッチ	
	0	：検証しない
	1	：サイズを検証
keyinf	モデル情報ファイルの出力制御スイッチ	
	0	：モデル情報ファイルを出力しない。
	1	：モデル情報ファイルを出力する。
	2	：モデル情報ファイルを更新出力する。
keypcvt	ピクチャ番号変換スイッチ	
	0	：ピクチャ番号を変換しない。
	1	： ピクチャ番号を変換する。
ipcvtbl	ピクチャ番号変換テーブル	
picRef	ピクチャ参照機能の制御スイッチ	
	(ピクチャ参照についてはコマンドリファレンスマニュアルを参照)	
	false	：ピクチャ参照機能を使用しない。
	true	：ピクチャ参照機能を使用する。

【返り値】

0 ： 正常終了

注意

モデルの保存モード mode が 0 のとき、アクティブモデルのモデル名が name で指定された名前となる。

9.5.3 モデルファイルの読み込み

【関数名】

Mdlread

【機能】

既存のモデルファイルを読み込む。

【呼出し形式】

```
int Mdlread(const char* name, int mode, int keymsk, int ipicfrm, int ipicdst,
            double ang, bool picRef = false)
```

【入力引数】

name	モデル名 (NULL-terminated) モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。
mode	モデルの読み込みモード 0 ：新規開始してからモデルを読み込む。 1 ：現在作業中のモデルにアイテムだけを読み込む。 2 ：現在作業中のモデルにアイテムを読み込み、ピクチャの縮尺値をモデルファイルと同じにする。
keymsk	選択マスクの制御スイッチ 0 ：現在の選択マスクを参照せずに、全てのアイテムを読み込む。 1 ：現在のアイテム選択マスクとクラス選択マスクに従って、アイテムを読み込む。
ipicfrm	入力ピクチャ番号 0 ：全てのピクチャのデータを読み込む。 ＞0 ：ピクチャ #ipicfrm のデータだけ読み込む。
ipicdst	格納先ピクチャ番号 0 ：アイテムを各々のピクチャに読み込む。 ＞0 ：アイテムをピクチャ #ipicdst に読み込む。
ang	読み込み時の回転角度
picRef	ピクチャ参照機能の制御スイッチ （ピクチャ参照についてはコマンドリファレンスマニュアルを参照） false ：ピクチャ参照機能を使用しない。 true ：ピクチャ参照機能を使用する。

【返り値】

0 ： 正常終了

注意

入力引数の値が以下のとき、アクティブモデルのモデル名が name で指定された名前となる。

モデルの読み込みモード	mode == 0
選択マスクの制御スイッチ	keymsk == 0
入力ピクチャ番号	ipicfrm == 0
格納先ピクチャ番号	ipicdst == 0

9.5.4 モデルファイルの読み込み

【関数名】

Mdlread1

【機能】

既存のモデルファイルを読み込む。

【呼出し形式】

```
int Mdlread1(const char* name, int mode, int picscf, int romode, int keymsk,
             const short ipcvtbl[], double ang, bool picRef = false)
```

【入力引数】

name　　モデル名 (NULL-terminated)
　　モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。
mode　　モデルの読み込みモード
　　0　　：新規開始してからモデルを読み込む。
　　1　　：現在作業中のモデルにアイテムだけを読み込む。
picscf　　picture scale 制御 (常に 0 を渡すこと)
　　0　　：ピクチャの縮尺値は変更しない。
　　1　　：ピクチャの縮尺値をモデルファイルと同じにする。
romode　　読み込み専用モード
　　0　　：通常モード
　　1　　：読み込み専用モードでモデルファイルを読み込む。
keymsk　　選択マスクの制御スイッチ
　　0　　：現在の選択マスクを参照せずに、全てのアイテムを読み込む。
　　1　　：現在のアイテム選択マスクとクラス選択マスクに従って、アイテムを読み込む。
ipcvtbl　　ピクチャ変換テーブル (short ipcvtbl[MAXITMPIC])
　　ipcvtbl[i - 1]:ピクチャ #i をピクチャ #ipcvtbl[i - 1] に変換。
　　(i == 1, MAXITMPIC)
　　ipcvtbl[i] の値は、1 から MAXITMPIC までで、かつ ipcvtbl[j] (j<>i) と重複してはならない。
ang　　読み込み時の回転角度
picRef　　ピクチャ参照機能の制御スイッチ
　　(ピクチャ参照についてはコマンドリファレンスマニュアルを参照)
　　false　　：ピクチャ参照機能を使用しない。
　　true　　：ピクチャ参照機能を使用する。

【返り値】

0　　：正常終了

注意

入力引数の値が以下のとき、アクティブモデルのモデル名が iname で指定された名前となる。

モデルの読み込みモード	mode == 0
選択マスクの制御スイッチ	keymsk == 0
ピクチャ変換テーブル	ピクチャ変換を全くしないとき

9.5.5　モデルピクチャ書き込み

【関数名】

MdlrplpicI

【機能】

モデルピクチャ書き込みを行なう。

【呼出し形式】

```
int MdlrplpicI(const char* name, int keychk, int ipicmdf, int ipicmdl,
               bool picRef = false)
```

【入力引数】

name　　モデル名 (NULL-terminated)
　　モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。
　　既存のモデルファイルでなければならない。

keychk　作成モデルファイル検証スイッチ
　　0　　：検証しない
　　1　　：サイズを検証
ipicmdf　書き込み先ピクチャ
　　モデルファイルのピクチャ番号
ipicmdl　書き込み元ピクチャ
　　モデルのピクチャ番号
picRef　ピクチャ参照機能の制御スイッチ
　　（ピクチャ参照についてはコマンドリファレンスマニュアルを参照）
　　false　：ピクチャ参照機能を使用しない。
　　true　：ピクチャ参照機能を使用する。

【返り値】

0　：正常終了

9.5.6　モデルファイルのヘッダー情報参照

【関数名】

Mdl101

【機能】

モデルファイルのヘッダー情報を参照する。

【呼出し形式】

```
int Mdl101(const char* name, short *icdate, short *iudate,char *cusrnme,
           short *lusrnme, short *iunttyp, char *cmdlttl, short *lmdlttl,
           char *cpicnme, int *nttlitm, int *npicitm, int *nascitm)
```

【入力引数】

name　モデル名 (NULL-char-terminated)。モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。

【出力引数】

icdate　モデルファイル作成日時 (short icdate[6])
　　icdate[0] : 日, icdate[1] : 月, icdate[2] : 年
　　icdate[3] : 時, icdate[4] : 分, icdate[5] : 秒
iudate　モデルファイル更新日時 (short iudate[6])
　　iudata[0] : 日, iudata[1] : 月, iudata[2] : 年
　　iudata[3] : 時, iudata[4] : 分, iudata[5] : 秒
cusrnme　モデルファイルのユーザ名 (char cusrnme[17])
lusrnme　モデルファイルのユーザ名の文字数
iunttyp　モデルファイルの単位系
　　1　　：ミリ単位系
　　4　　：インチ単位系
cmdlttl　モデルファイルの主タイトル
lmdlttl　モデルファイルの主タイトルの文字数
cpicnme　モデルファイルのピクチャ名 (char cpicnme[MAXITMPIC][257])
　　ピクチャ #1 からピクチャ #MAXITMPIC までのピクチャ名。(NULL-char terminated)
nttlitm　総アイテム数
npicitm　ピクチャ毎のアイテム数 (int npicitm[MAXITMPIC])
nascitm　アソシエーションアイテム数

【返り値】

0　　：正常終了

9.5.7　アクティブモデルのモデル名又はモデル名ファイルを参照

【関数名】

Mdl102

【機能】

アクティブモデルのモデル名またはモデル名ファイルを参照する。

【呼出し形式】

```
int Mdl102(int key, char *mname)
```

【入力引数】

key　　モデル名の制御スイッチ。
　　0　　：モデルファイル名を参照する。
　　1　　：モデル名を参照する。

【出力引数】

mname　　モデル名

【返り値】

モデル名の文字数。アクティブモデルのモデル名が設定されていない場合は、0 になる。

9.5.8　モデルファイルのヘッダー情報を表示

【関数名】

Mdl103

【機能】

モデルファイルのヘッダー情報をグラフィックウインドウに表示する。

【呼出し形式】

```
int Mdl103(const char* name, short *icdate, short *iudate, char *cusrnme,
           short *lusrnme, short *iunttyp, char *cmdlttl, short *lmdlttl,
           char *cpicnme, int *nttlitm, int *npicitm, int *nascitm)
```

【入力引数】

name　　モデル名 (NULL-char-terminated)。モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。

【出力引数】

icdate　モデルファイル作成日時 (short icdate[6])
　　icdate[0] : 日, icdate[1] : 月, icdate[2] : 年
　　icdate[3] : 時, icdate[4] : 分, icdate[5] : 秒
iudate　モデルファイル更新日時 (short iudate[6])
　　iudata[0] : 日, iudata[1] : 月, iudata[2] : 年
　　iudata[3] : 時, iudata[4] : 分, iudata[5] : 秒
cusrnme　モデルファイルのユーザ名 (char cusrnme[16])
lusrnme　モデルファイルのユーザ名の文字数
iunttyp　モデルファイルの単位系
　　1　　:ミリ単位系
　　4　　:インチ単位系
cmdlttl　モデルファイルの主タイトル
lmdlttl　モデルファイルの主タイトルの文字数
cpicnme　モデルファイルのピクチャ名 (char cpicnme[MAXITMPIC][257])
　　ピクチャ#1 からピクチャ #MAXITMPIC までのピクチャ名。(NULL-char terminated)
nttlitm　総アイテム数
npicitm　ピクチャ毎のアイテム数 (int npicitm[MAXITMPIC])
nascitm　アソシエーションアイテム数

【返り値】

0　　: 正常終了

9.5.9　モデルファイルのモデルタイトルを参照

【関数名】

Mdl104

【機能】

モデルファイルのモデルタイトルを参照する。

【呼出し形式】

```
int Mdl104(const char* name, short *ittllst, int nttl, MDLTTLTXT *cttltxt,
          short *lttltxt)
```

【入力引数】

name　モデル名。(NULL-terminated)。モデル名にモデルファイルの拡張子 ".MDL" を含める必要はない。
ittllst　参照するモデルタイトル番号の並び (short ittllst[nttl])
nttl　参照するモデルタイトル番号の総数

【出力引数】

cttltxt　モデルファイルのモデルタイトル (MDLTTLTXT cttltxt[nttl])
lttltxt　モデルファイルのモデルタイトルの文字数 (short lttltxt[nttl])

【返り値】

0　　: 正常終了

9.5.10 モデル名を設定または解除

【関数名】

Mdlname001

【機能】

現在のモデル名を設定または解除する。
具体的には以下の２つの処理を行う。
(1) モデルタイトル項目番号２０２を設定／解除。
(2) モデルのロックファイルの作成／解除。

【呼出し形式】

```
int Mdlname001(char *mdlnme)
```

【入力引数】

nmlnme　モデル名。
解除する場合は（char *)NULL または長さ０の文字列（""）にする。

【返り値】

0　：正常終了。
-n　：正常であるが指定されたモデル名は以下の状態にある。
　-50　：指定されたモデルが既に存在する。
　-51　：書き込み禁止のファイル／ディレクトリが指定された。
　　または呼出し専用オプション（-ro）で起動された。
　-52　：指定されたモデルは他のユーザによって使用されている。
+n　：指定されたモデル名がモデルタイトル２０２として不正。
　2　：モデルタイトルテンプレートで規定されている最小文字数より短い。
　3　：モデルタイトルテンプレートで規定されている最大文字数より長い。
　4　：モデルタイトルの総文字数が制限を超える。
　5　：モデルタイトルテンプレートで規定されている使用禁止文字が含まれている。

例

```
int ier;
ier = Mdlname001("TEST");              /* モデル名を TEST にする */
ier = Mdlname001((char *)NULL);        /* モデル名を解除する */
```

9.5.11 現存のモデル名を抽出

【関数名】

Mdlname101

【機能】

現在のモデル名を抽出する。

【呼出し形式】

```
int Mdlname101(int iswt, char *mdlnme, size_t l_mdlnme)
```

【入力引数】

iswt　　スイッチ。
　　1　　：入力された状態のモデル名を得る。
　　2　　：フルパスのモデル名を得る。
　　3　　：ディレクトリとファイル拡張子を取り除いたモデル名を得る。
l_mdlnme　出力引数 mdlnme の大きさ（バイト数）。

【出力引数】

nmlnme　　モデル名。

【返り値】

0　　：正常終了。
1　　：入力引数 iswt の値が不正。
2　　：モデル名が l_mdlnme を超える。
3　　：モデル名は設定されていない。

例

```
int ier;
char mdlnme1[257], mdlnme2[257], mdlnme2[257];
(void)Mdlname001("test");
ier = Mdlname101(1, mdlnme1, sizeof(mdlnme1));
ier = Mdlname101(2, mdlnme1, sizeof(mdlnme2));
ier = Mdlname101(3, mdlnme1, sizeof(mdlnme3));

mdlnme1、mdlnme2、mdlnme3 は以下のような文字列になる。
mdlnme1 : "test"
mdlnme2 : "/acad/files/TEST.MDL"
mdlnme3 : "TEST"
```

9.5.12 サブモデルファイルの作成

【関数名】

Subgen

【機能】

サブモデルファイルを作成する。

【呼出し形式】

```
int Subgen(const char* name, int mode, int keychk, int keyrpl, int keyinf,
          int keypcvt, const short ipcvtbl[],
          const DPOINT org[], const double ang[])
```

【入力引数】

name　　サブモデル名 (NULL terminated)。サブモデル名に、モデルファイルの拡張子”.MDL”を含める必要はない。
mode　　サブモデルファイルの作成モード
　　-1　　：アクティブリスト中のアイテムだけをサブモデルのアイテムとする。
　　0　　：全アイテムをサブモデルのアイテムとする。
　　≧1　　：モデルのピクチャ #mode 上のアイテムをサブモデルのアイテムとする。
keychk　作成サブモデルファイル検証スイッチ
　　0　　：検証しない
　　1　　：サイズを検証

keyrpl　サブモデル変換制御スイッチ
　0　：サブモデルのアイテムとしたアイテムをサブモデルアイテムに変換しない。
　1　：サブモデルのアイテムとしたアイテムをサブモデルアイテムに変換する。
keyinf　モデル情報ファイルの出力制御スイッチ
　0　：モデル情報ファイルを出力しない。
　1　：モデル情報ファイルを出力する。
　2　：モデル情報ファイルを更新出力する。
keypcvt　ピクチャ変換制御スイッチ
　0　：ピクチャ変換しない。
　1　：アイテムのピクチャ番号とピクチャ縮尺値をピクチャ変換テーブルにもとづき変換する。
ipcvtbl　ピクチャ変換テーブル (short ipcvtbl[MAXITMPIC])
ピクチャ変換制御スイッチ KEYPCVT == 1 のときだけ参照される。
ipcvtbl[i - 1] ： ピクチャ #i をピクチャ #ipcvtbl[i - 1] に変換。(i == 1, MAXITMPIC)
ipcvtbl[i] の値は、1 から MAXITMPIC までで、かつ ipcvtbl[j] (j◇i) と重複してはならない。
org　サブモデル原点の配列 (DPOINT org[MAXITMPIC])
ang　サブモデル基準水平角度の配列 (単位：度)(double ang[MAXITMPIC])

【返り値】

0　：正常終了

9.5.13 サブモデルアイテムをテンポラリバッファに作成

【関数名】

Sub101

【機能】

サブモデルアイテムをテンポラリバッファに作成する。

【呼出し形式】

```
int Sub101(const char* name, int keymsk, int keydim, int keydrf, int ipicfrm,
           const DPOINT& org, const double scf[], double ang, int itmtyp)
```

【入力引数】

name　サブモデル名 (NULL terminated)。サブモデル名に、モデルファイルの拡張子”.MDL”を含める必要はない。
keymsk　アイテム選択マスク制御スイッチ
　0　：選択マスクを無視する。
　1　：一時的な選択マスクで指示されたアイテムだけをサブモデルアイテム化する。
keydim　寸法の調整制御スイッチ
　0　：サブモデルに含まれる寸法を入力パラメータ (ORG, SCF, ANG) に合わせて調整しない。
　1　：サブモデルに含まれる寸法を入力パラメータ (ORG, SCF, ANG) に合わせて調整する。
keydrf　製図図形要素の縮尺制御スイッチ
　0　：製図図形要素は縮尺しない。幾何図形要素だけをサブモデル縮尺値 (scf) に基づいて縮尺する。
　1　：製図図形要素も幾何図形要素と同様にサブモデル縮尺値 (scf) に基づいて縮尺する。
ipicfrm　サブモデル配置ピクチャ番号。サブモデル名で指定されたモデルファイルのピクチャ# ipicfrm 上のアイテムがサブモデルとなる。
org　サブモデル配置位置 (サブモデル原点の位置)

scf　　サブモデル縮尺値
　　scf[0]　　：　X 縮尺値（サブモデルの X 軸方向に対する縮尺値）負である場合には、サブモデルの Y 軸に対して反転する。
　　scf[1]　　：　Y 縮尺値（サブモデルの Y 軸方向に対する縮尺値）負である場合には、サブモデルの X 軸に対して反転する。
　　拡大／縮小しないときは、scf[0] = scf[1] = 1.0 とする
ang　　サブモデル回転角度（単位：度）
ltmtyp　ITMSUBMDL

【返り値】

0　　：正常終了。異常終了の際は、サブモデルアイテムは作成されない。

注意

この関数は、指示された配置パラメータをもとにして、サブモデルアイテムをテンポラリバッファに追加する。テンポラリバッファのオープンとクローズは、このモジュール内で行なわれているため、TmpgOpen1 と TmpgClose を使う必要はない。サブモデルアイテムを作るにはつぎのようにする。

```
#include "acadprm.h"
TmpgWrtoDB(0);
Sub101(iname, keymsk, keydim, keydrf, ipicfrm, org, scf, ang, ITMSUBMDL);
TmpgWrtoDB(0);
```

9.5.14 サブモデルアイテム作成

【関数名】

Subput

【機能】

サブモデルアイテムを作成する。作成したアイテムはデータベース内に格納される。

【呼出し形式】

```
int Subput(const char* name, int mode, int keymsk, int keydim, int keydrf,
          int ipicfrm, const DPOINT& org, const double scf[], double ang,
          bool picRef = false)
```

【入力引数】

name　　サブモデル名 (NULL terminated)。サブモデル名に、モデルファイルの拡張子”.MDL”を含める必要はない。
mode　　サブモデル配置モード
　　0　　：サブモデルとして配置する。
　　1　　：複合アイテムとして配置する。
　　2　　：サブモデルを分解して配置する。
keymsk　アイテム選択マスク制御スイッチ
　　0　　：選択マスクを無視する。
　　1　　：一時的な選択マスクで指示されたアイテムだけをサブモデルアイテム化する。
keydim　寸法の調整制御スイッチ
　　0　　：サブモデルに含まれる寸法を入力パラメータ (ORG, SCF, ANG) に合わせて調整しない。
　　1　　：サブモデルに含まれる寸法を入力パラメータ (ORG, SCF, ANG) に合わせて調整する。
keydrf　製図図形要素の縮尺制御スイッチ

0　：製図図形要素は縮尺しない。幾何図形要素だけをサブモデル縮尺値（scf）に基づいて縮尺する。
1　：製図図形要素も幾何図形要素と同様にサブモデル縮尺値（scf）に基づいて縮尺する。

ipicfrm　サブモデル配置ピクチャ番号。サブモデル名で指定されたモデルファイルのピクチャ＃ ipicfrm 上のアイテムがサブモデルとなる。

org　サブモデル配置位置（サブモデル原点の位置）

scf　サブモデル縮尺値

scf[0]　：X 縮尺値（サブモデルの X 軸方向に対する縮尺値）負である場合には、サブモデルの Y 軸に対して反転する。

scf[1]　：Y 縮尺値（サブモデルの Y 軸方向に対する縮尺値）負である場合には、サブモデルの X 軸に対して反転する。

拡大／縮小しないときは、scf[0] = scf[1] = 1.0 とする

ang　サブモデル回転角度（単位：度）

picRef　ピクチャ参照機能の制御スイッチ
（ピクチャ参照についてはコマンドリファレンスマニュアルを参照）

false　：ピクチャ参照機能を使用しない。
true　：ピクチャ参照機能を使用する。

【返り値】

0　：正常終了。異常終了の際は、サブモデルアイテムは作成されない。

注意

Sub101() 関数との引数の違いは第 2 引数のみです。第 2 引数を指定することにより、サブモデルの配置モードを指定することができます。
また、Sub101() 関数はサブモデルをテンポラリバッファに作成しますが、この Subput() 関数はサブモデルをデータベースに追加します。

9.6 アソシエイトアイテム

● 関数一覧

関数名	機能
Ascadd	アソシエイトアイテムにメンバーを追加する。
Ascbreak	アソシエイトの解除。
Ascdelete	アソシエイトアイテムとそのメンバーアイテムを削除する
Ascnameal	すべてのアソシエイトアイテムの名前を得る。
Ascnamegt	指定したアソシアエイトアイテムの名前を得る。
Ascnameid	与えた名前のアソシエイトアイテムのアイテム識別子を得る。
Ascnew	アソシエイトアイテムを作成する。
Ascrelese	あるアイテムをそれをメンバーとして含む全てのアソシエイトアイテムからはずす。
Ascrename	アソシエイトアイテムの名前を変更する。
Ascs29gt	アイテム識別子 idptr にアソシエイトされている idptr リストと、そのアソシエイトカテゴリ番号を取出す。
Ascctg001	カテゴリ番号の格納。
Ascctg101	カテゴリ番号の取り出し。
ascs29idp	アイテムがどのアソシエートアイテムに属するかを調べる。

9.6.1 アソシエイトアイテムにメンバー追加

【関数名】

Ascadd

【機能】

アソシエイトアイテムにメンバーを追加する。

【呼出し形式】

```
int Ascadd(int idptrasc, const int idptrlist[], int ndptrcnt)
```

【入力引数】

```
idptrasc   アソシエイトアイテムのアイテム識別子
idptrlist  追加するアイテムの識別子リスト
           (int idptrlist[ndptrcnt])
ndptrcnt   追加アイテム数
```

【返り値】

0　　：正常終了
1　　：idptrasc がアソシエイトアイテムでない

注意
追加アイテムに付けられるアソシエイトカテゴリ番号は idptrasc が持っているアソシエイトカテゴリ番号が追加される。

9.6.2　アソシエイトの解除

【関数名】

Ascbreak

【機能】

アソシエイトの解除。アソシエイトアイテムは削除されるが、そのアソシエイトアイテムのメンバーであったアイテムは削除されない。

【呼出し形式】

```
int Ascbreak(int idptrasc)
```

【入力引数】

idptrasc　アソシエイトアイテムのアイテム識別子

【返り値】

0　　：正常終了
1　　：idptrasc がアソシエイトアイテムでない

9.6.3　アソシエイトアイテム及びメンバーアイテムをデータベースから削除

【関数名】

Ascdelete

【機能】

アソシエイトアイテムおよびそのメンバーであるアイテムを全てデータベースから削除する。

【呼出し形式】

```
int Ascdelete(int idptrasc)
```

【入力引数】

idptrasc　アソシエイトアイテムアイテム識別子

【返り値】

0　　：正常終了
1　　：idptrasc がアソシエイトアイテムでない

9.6.4　全アソシエイトアイテム名を取得

【関数名】

Ascnameal

【機能】

すべてのアソシエイトアイテムの名前を得る。

【呼出し形式】

```
int Ascnameal(int icatasc, char itemname[][33])
```

【入力引数】

icatasc　アソシエイトカテゴリ番号
　0　　：全てのアソシエイト名
　n　　：アソシエイトカテゴリ番号 n を持つ全てのアソシエイトアイテムの名前

【出力引数】

itemname　アソシエイト名リスト (char itemname[][33])
　１つの名前は 32 文字以下。NULL terminated.

【返り値】

名前の数　(≦ 4096)

9.6.5　指定したアソシエイトアイテムの名前を取得

【関数名】

Ascnamegt

【機能】

指定したアソシアエイトアイテムの名前を得る。

【呼出し形式】

```
int Ascnamegt(int idptrasc, char *itemname)
```

【入力引数】

idptrasc　アイテム識別子

【出力引数】

itemname　アソシエイト名。名前は最大 32 文字。NULL terminated.

【返り値】

名前の文字数

9.6.6　アソシエイトアイテム名からアイテム識別子を取得

【関数名】

Ascnameid

【機能】

与えた名前のアソシエイトアイテムのアイテム識別子を得る。

【呼出し形式】

```
int Ascnameid(const char* itemname)
```

【入力引数】

itemname　アソシエイト名。NULL terminated.

【返り値】

アイテム識別子。アソシエイトアイテムでなければ 0 となる。

9.6.7　アソシエイトアイテムを作成

【関数名】

Ascnew

【機能】

アソシエイトアイテムを作成する。

【呼出し形式】

```
int Ascnew(const char* itemname, int icatasc, const int idptrlist[], int ndptrcnt,
           int keyundo)
```

【入力引数】

```
itemname  アソシエイトアイテムの名前。NULL terminated. 32 文字以下。
icatasc   アソシエイトカテゴリ番号 (1 - 255)
idptrlist アソシエイトアイテムのメンバーにするアイテムの、アイテム識別番号のリスト。
          (int idptrlist[ndptrcnt])
ndptrcnt  アイテム数
keyundo   UNDO をカウントアップするスイッチ
             0        :カウントアップしない
             1        :カウントアップする
```

【返り値】

アソシエイトアイテムのアイテム識別子。エラーの場合 0 となる。

9.6.8　アソシエイトアイテムからメンバーを外す

【関数名】

Ascrelese

【機能】

アソシエイトアイテムからメンバーをはずす。

【呼出し形式】

```
int Ascrelese(int idptrasc, const int idptrlist[], int idptrcnt)
```

【入力引数】

idptrasc　アソシエイトアイテムのアイテム識別子。
idptrlist 除去するアイテムのアイテム識別子。(int idptrlist[idptrcnt])
idptrcnt　除去するアイテム数。

【返り値】

0　　：正常終了
1　　：このアイテムはどのアソシエイトアイテムのメンバーでもない

9.6.9　アソシエイトアイテム名を変更

【関数名】

Ascrename

【機能】

アソシエイトアイテムの名前を変更する。

【呼出し形式】

```
int Ascrename(const char* itemname, int idptrasc)
```

【入力引数】

itemname　新しい名前。NULL terminated. 32 文字以下。
idptrasc　アソシエイトアイテムのアイテム識別子

【返り値】

0　　：正常
1　　：エラー

9.6.10　アイテム識別子のリストとカテゴリ番号の取得

【関数名】

Ascs29gt

【機能】

アイテム識別子 idptr にアソシエイトされているアイテムのアイテム識別子のリストと、そのアソシエイトカテゴリ番号を取り出す。

【呼出し形式】

```
int Ascs29gt(int idptr, int icatasc, int items29[][2])
```

【入力引数】

idptr	アイテム識別子。アソシエイトアイテムまたはアソシエイトアイテムのメンバーであるアイテム。
icatasc	対象とするアソシエイトカテゴリ番号。 0 : 全部のアソシエイトカテゴリを対象とする n : n 番のアソシエイトカテゴリを対象とする

【出力引数】

items29	アソシエイト相手先データ (short items29[4096][2])	
	items29[][0]	byte 1 : ソシエイトカテゴリ番号 byte 2 : 親子関係 1 : items29[][1] がこのアイテムの親である 2 : items29[][1] がこのアイテムの子である
	items29[][1]	アソシエイト相手先アイテム識別子

【返り値】

アソシエイト相手先数。アソシエイトされていなければ 0 となる。

9.6.11 カテゴリ番号を格納

【関数名】

Ascctg001

【機能】

カテゴリ番号を格納する。

【呼出し形式】

```
int Ascctg001(int category1, int category2)
```

【入力引数】

category1	カテゴリ番号 1
category2	カテゴリ番号 2

【返り値】

カテゴリ番号。

9.6.12 カテゴリ番号を取得

【関数名】

Ascctg101

【機能】

カテゴリ番号を取り出す。

【呼出し形式】

```
int Ascctg101(int category, int *category1, int *category2)
```

【入力引数】

category　カテゴリ番号

【出力引数】

category1 カテゴリ番号 1 格納領域
category2 カテゴリ番号 2 格納領域

【返り値】

0　　: 正常
1　　: エラー

9.6.13 アソシエートアイテムのアイテム識別子の取得

【関数名】

ascs29idp

【機能】

アイテムがどのアソシエートアイテムに属するかを調べる。

【呼出し形式】

```
int ascs29idp(int idptr, int icat)
```

【入力引数】

idptr　　調べるアイテムのアイテム識別子
icat　　アソシエートのカテゴリ番号
　　= 0　　:全部のアソシエイトカテゴリを対象とする。
　　= n　　:n 番のアソシエイトカテゴリを対象とする。

【返り値】

アソシエートアイテムのアイテム識別子
0　　: アイテムはアソシエートされていない。

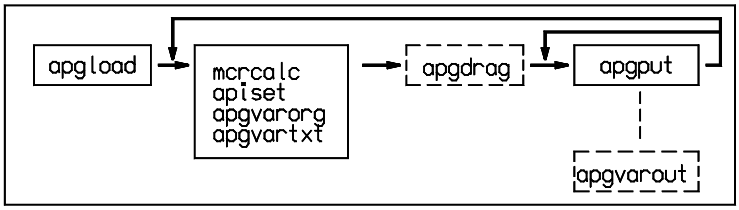
9.7 APG アイテム

● 関数一覧

apgload	**APG ファイルを APG データ領域にロードする。 ロードしたデータは、次の APG ファイルがロードされるまで保持される。**
apgput	**APG 配置。 APG アイテムをデータベースに書き込む。**

注 . apgput の前に mcrcalc 関数を使用して APG パラメータに値をセットすること。
mcrcalc については本書「12. ユーティリティ関数」を参照して下さい。
計算機変数とは、Advance CAD 中の計算（コマンド中での計算式，マクロ中での外部変数，apgput 時に使用された変数）で使用された変数のこと。
APG パラメータ名は６文字以内であること。

apgvarorg	**APG ファイル作成時の値を 計算機変数にセットする。 APG ファイルをロードした直後に呼出すこと。 apgdrag，apgput したあとに aogvarorg を呼出すと、配置またはドラッグの計算結果が計算機変数に設定される。**
apgvartxt	**APG ファイル作成時のパラメータ名を取り出す。**
apgvarout	**APG パラメータ変数出力 APG パラメータ名、およびその配置時の値をファイルに出力する。**
apgdrag	**APG ドラッグ APG データをドラッグモードにする。APG Rev. 4.0 以降のみ使用可。**



9.7.1 APG ファイルの読み込み

【関数名】

apgload

【機能】

APG ファイル を APG データ領域にロードする。
一度ロードしたデータはつぎの APG ファイルがロードされるまで保持される。
したがって、繰り返し位置を変えたりパラメータの値を変えたりして、連続して使用できる。

【呼出し形式】

```
int apgload(char *apgfile)
```

【入力引数】

apgfile　APG ファイル名 (NULL-terminated)。ファイル名に拡張子 ".APG" を含める必要はない。

【返り値】

0　　: 正常終了

9.7.2　APG アイテムをデータベースに書き込み

【関数名】

apgput

【機能】

APG データ領域にセットされた APG パラメータの値を、計算機変数中にセットされた値をもとに再計算する。そしてその値を使用して全図形を作成し、データベースに書き込む。
apgput を呼び出すと、クラス 251 で定義された寸法変数の値が計算され、計算機変数にその値がセットされる。

【呼出し形式】

```
void apgput(int keyapg, int itemapg, int keyupd, int keydim, int keytol,
            const DPOINT& porg, const DPOINT& dvec, int newcls, int skp254,
            int clsofs, int outpic, int mirx, int miry, int keyspc,
            int keymsk, int keyuno)
```

【入力引数】

keyapg　Advance CAD データベースに作成するアイテムタイプの定義
　　0　　:APG ファイル作成時と同じアイテムタイプで作成される。
　　1　　:全図形データが 1 つの APG アイテムとして作成される。
itemapg　既存の APG アイテムをアップデートする場合の APG アイテムの アイテム識別子。新規作成の場合 = 0
keyupd　必ず 0 をセットすること。
keydim　寸法線データ出力キー。
　　0　　:寸法線を出力しない。1 : 寸法線を出力する。
keytol　パラメタライズ計算時に寸法公差を使用するためのスイッチ。
　　0　　:使用しない。1 : 使用する。
porg　図形作成時の原点座標
dvec　図形作成時の回転ベクトル。たとえば、回転角度 30 度の場合は、

```
dvec->x = cos(30.0*PI/180.0);
dvec->y = sin(30.0*PI/180.0);
```

newcls　新規データ作成時のクラス番号
　　0　　:　APG ファイル作成時のクラス番号を使用する。
　　1-255　　:　データは、セットされたクラス番号で作成される。
skp254　APG 作成時にクラス 254 で作成されたデータをスキップするキー。
　　0　　:CLS 254 のデータをデータベースに書き込む。
　　1　　:CLS 254 のデータをデータベースに書き込まない。
clsofs　新規データの作成時に、APG ファイル作成時のクラス番号に加算するクラス番号。
クラス番号は、以下の式で決定される。

```
MAX( MOD(original class + clsofs), 256), 1)
```

outpic　出力ピクチャ番号をセットする (1 ～ MAXITMPIC)

0 : カレントピクチャ番号

mirx APG ファイル作成時の X 軸に対して反転してデータベースに書き込む。
0 :反転しない
1 :反転する

miry APG ファイル作成時の Y 軸に対して反転してデータベースに書き込む。
0 :反転しない
1 :反転する

keyspc APG アイテムを新規作成したときに、特性データバッファの内容を追加して、データベースに書き込むためのスイッチ。
0 :書き込まない。
1 :書き込む。

keymsk アイテムタイプ、クラス、レビジョン、線種、線幅のマスク
0 :無効
1 :有効

keyuno UNDO BLOCK をくぎるかどうか
0 :くぎらない
1 :くぎる

9.7.3 APG データをドラッグモードにする

【関数名】

apgdrag

【機能】

APG データ領域にセットされた APG パラメータの値を計算器変数の値を使用して再計算する。そしてその値を使用して図形データを作成し、外形データ、最大ボックス、図形データそのままのいづれかをドラッグする。
apgdrag を呼び出すと、クラス 251 で定義された寸法変数の値が計算され、計算器変数にその値が設定される。

【呼出し形式】

```
void apgdrag(int drgtyp, int keytol, const DPOINT& dvec, int skp254,
             int mirx, int miry, int keymsk)
```

【入力引数】

drgtyp ドラッグモード。
1 :外形データ
2 :最大ボックス
3 :図形データそのまま

keytol パラメタライズ計算時に寸法公差を使用するためのスイッチ。
0 :使用しない, 1 : 寸法公差を使用してパラメタライズ計算する。

dvec 図形作成時の回転ベクトル。たとえば回転角度 30 度の場合、

```
dvec.x = cos(30.0*PI/180.0);
dvec.y = sin(30.0*PI/180.0);
```

skp254 APG 作成時にクラス 254 で作成されたデータをスキップするかどうか。
0 :スキップしない
1 :スキップする

imirx X 軸反転。
0 :反転しない
1 :反転する

imiry Y 軸反転。
0 :反転しない
1 :反転する

keymsk　　アイテムタイプ、クラス、レビジョン、線種、線幅のマスク。
　　0　　：無効
　　1　　：有効

9.7.4　APG ファイルパラメータ値を計算機変数に設定

【関数名】

apgvarorg

【機能】

APG ファイル作成時のパラメータの値を計算機変数にセットする

【呼出し形式】

```
void apgvarorg(void)
```

注意

APG ファイルをロードした直後に呼び出すこと。
apgdrag, apgput したあとに aogvarorg を呼び出すと、配置またはドラッグの計算結果の値が計算機変数にセットされる。

9.7.5　APG ファイル作成時のパラメータ名を取得

【関数名】

apgvartxt

【機能】

APG ファイル作成時のパラメータ名を取り出す。

【呼出し形式】

```
int apgvartxt(int type, char nlst[][6], int maxlst)
```

【入力引数】

type　　2
maxlst　　配列 nlst の長さ

【出力引数】

nlst　　パラメータ名を受ける配列 (char nlst[maxlst][6])

【返り値】

パラメータ名の数

9.7.6　配置時の APG パラメータ名と値をファイルに出力

【関数名】

apgvarout

【機能】

配置時の APG パラメータ名とその値をファイルに出力する。

【呼出し形式】

```
void apgvarout(const char* fname)
```

【入力引数】

fname　　出力ファイル名 (NULL-terminated)。ファイル名に拡張子 ".APP" を含める必要はない。

例

```
/
/ ** Advance CAD APG rev 4.0 Parameter Output **
/
AG1   = 50
H1    = 70
H2    = 50
R12   = 12
R30   = 30
R50   = 50
T     = 15
V1    = 45
V2    = 35
```

9.7.7　APG パラメータファイルのパラメータを計算機変数に設定

【関数名】

apiset

【機能】

APG パラメータファイルのパラメータを、計算機変数にセットする。

【呼出し形式】

```
int apiset(const char* fname)
```

【入力引数】

fname　　APG パラメータファイル名 (NULL-terminated)。拡張子 ".API" を含める必要はない。

【返り値】

0　　: 正常終了

9.8 アイテム編集

● 関数一覧

edtcoplst	**アイテムを複製する**
edtmovlst	**アイテムを移動する**
edtmirlst	**アイテムを反転する**
edtrotlst	**アイテムを回転する**
edtstrset	**伸縮領域を設定する。**
edtstrlst	**アイテムを伸縮する**
edtexplst	**アイテムを拡大縮小する**
edtaryrec	**アイテムの矩形配列を作成する**
edtarycir	**アイテムの回転配列を作成する**
edtaryrad	**アイテムの回転配列 (向心) を作成する**

9.8.1 アイテムを複製する

【関数名】

edtcoplst

【機能】

指定されたアイテムの複製アイテムを作成する。

【呼出し形式】

```
int edtcoplst(int keydup, int keypic, const int itmlst[], int nitm)
```

【入力引数】

keydup　複製アイテム制御スイッチ
- 0　：複製アイテムを作成しない。
- 1　：複製アイテムを作成する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、対応するアイテムと同一とする。
- 2　：複製アイテムを作成する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、現在のデフォルトのアイテム属性とする。

keypic　出力ピクチャ制御スイッチ
- 0　:keydup が 0 であれば、指定されたアイテムをそれが存在するピクチャに出力する。keydup が 0 でなければ、複製アイテムをアクティブピクチャに出力する。
- > 0　：指定されたアイテムまたは複製アイテムをピクチャ #keypic に出力する。

itmlst　アイテム識別子のリスト（int itmlst[nitm]）

nitm　アイテム識別子の数

【返り値】

0　：正常終了

9.8.2 アイテムを移動する

【関数名】

edtmovlst

【機能】

指定されたアイテムを移動する。

【呼出し形式】

```
int edtmovlst(int keydup, int keypic, const int itmlst[], int nitm,
              const DPOINT& vec)
```

【入力引数】

keydup　複製アイテム制御スイッチ
- 0　：複製アイテムを作成せずに指定されたアイテムを移動する。
- 1　：複製アイテムを作成し、それを移動する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、対応するアイテムと同一とする。
- 2　：複製アイテムを作成し、それを移動する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、現在のデフォルトのアイテム属性とする。

keypic　出力ピクチャ制御スイッチ
- 0　:keydup が 0 であれば、指定されたアイテムをそれが存在するピクチャに出力する。keydup が 0 でなければ、複製アイテムをアクティブピクチャに出力する。
- ＞0　：指定されたアイテムまたは複製アイテムをピクチャ #keypic に出力する。

itmlst　アイテム識別子のリスト（int itmlst[nitm]）

nitm　アイテム識別子の総数

vec　移動ベクトル

【返り値】

0　：正常終了

9.8.3 アイテムを回転する

【関数名】

edtrotlst

【機能】

指定されたアイテムを回転する。

【呼出し形式】

```
int edtrotlst(int keydup, int keypic, const int itmlst[], int nitm,
              const DPOINT& org, double ang)
```

【入力引数】

keydup　複製アイテム制御スイッチ
- 0　：複製アイテムを作成せずに指定されたアイテムを回転する。

1　：複製アイテムを作成し、それを回転する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、対応するアイテムと同一とする。
2　：複製アイテムを作成し、それを回転する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、現在のデフォルトのアイテム属性とする。

keypic　出力ピクチャ制御スイッチ
0　：keydup が 0 であれば、指定されたアイテムをそれが存在するピクチャに出力する。keydup が 0 でなければ、複製アイテムをアクティブピクチャに出力する。
＞0　：指定されたアイテムまたは複製アイテムをピクチャ #keypic に出力する。

itmlst　アイテム識別子のリスト（int itmlst[nitm]）
nitm　アイテム識別子の数
org　回転中心点
ang　回転角度（単位：度）

【返り値】

0　：正常終了

9.8.4　アイテムを反転する

【関数名】

edtmirlst

【機能】

指定されたアイテムを反転する。

【呼出し形式】

```
int edtmirlst(int keydup, int keypic, const int itmlst[], int nitm,
              const DPOINT& org, const DPOINT& uvec)
```

【入力引数】

keydup　複製アイテム制御スイッチ
0　：複製アイテムを作成せずに指定されたアイテムを反転する。
1　：複製アイテムを作成し、それを反転する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、対応するアイテムと同一とする。
2　：複製アイテムを作成し、それを反転する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、現在のデフォルトのアイテム属性とする。

keypic　出力ピクチャ制御スイッチ
0　:keydup が 0 であれば、指定されたアイテムをそれが存在するピクチャに出力する。keydup が 0 でなければ、複製アイテムをアクティブピクチャに出力する。
>0　：指定されたアイテムまたは複製アイテムをピクチャ #keypic に出力する。

itmlst　アイテム識別子のリスト（int itmlst[nitm]）
nitm　アイテム識別子の数
org　反転基準線の始点
uvec　反転基準線の向きをしめす正規ベクトル

【返り値】

0　：正常終了

9.8.5　アイテムを伸縮する

【関数名】

edtstrlst

【機能】

指定されたアイテムを伸縮する。

【呼出し形式】

```
int edtstrlst(int keydup, int keypic, const int itmlst[], int nitm,
              const DPOINT& vec)
```

【入力引数】

keydup　複製アイテム制御スイッチ
- 0　：複製アイテムを作成せずに指定されたアイテムを伸縮する。
- 1　：複製アイテムを作成し、それを伸縮する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、対応するアイテムと同一とする。
- 2　：複製アイテムを作成し、それを伸縮する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、現在のデフォルトのアイテム属性とする。

keypic　出力ピクチャ制御スイッチ
- 0　：keydup が 0 であれば、指定されたアイテムをそれが存在するピクチャに出力する。keydup が 0 でなければ、複製アイテムをアクティブピクチャに出力する。
- >0　：指定されたアイテムまたは複製アイテムをピクチャ #keypic に出力する。

itmlst　アイテム識別子のリスト（int itmlst[nitm]）

nitm　アイテム識別子の数

vec　伸縮ベクトル

【返り値】

0　：正常終了

9.8.6　伸縮領域の設定

【関数名】

edtstrset

【機能】

伸縮領域を設定する。

【呼出し形式】

```
int edtstrset(const FPOINT plypnt[], int npnt)
```

【入力引数】

plypnt　多角形の伸縮領域（FPOINT plypnt[npnt]）
npnt が 2 の場合は、長方形の伸縮領域の対角点とみなす。

npnt　多角形の伸縮領域のコーナーの総数
(2 <= npnt <= 256)

9.8.7　アイテムの拡大縮小

【関数名】

edtexplst

【機能】

指定されたアイテムを拡大縮小する。

【呼出し形式】

```
int edtexplst(int keydup, int keypic, int keydrf, const int itmlst[], int nitm,
              const DPOINT& org, const double scf[])
```

【入力引数】

keydup　複製アイテム制御スイッチ
- 0　：複製アイテムを作成せずに指定されたアイテムを拡大縮小する。
- 1　：複製アイテムを作成し、それを拡大縮小する。ただしアイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、対応するアイテムと同一とする。
- 2　：複製アイテムを作成し、それを拡大縮小する。ただし複製アイテムのアイテム属性（クラス、レビジョン、線種、線幅）は、現在のデフォルトのアイテム属性とする。

keypic　出力ピクチャ制御スイッチ
- 0　:keydup が 0 であれば、指定されたアイテムをそれが存在するピクチャに出力する。keydup が 0 でなければ、複製アイテムをアクティブピクチャに出力する。
- >0　：指定されたアイテムまたは複製アイテムをピクチャ #keypic に出力する。

keydrf　テキストアイテム、マークアイテムも縮尺するかどうかの制御スイッチ。
- 0　：縮尺しない。
- 1　：縮尺する。

itmlst　アイテム識別子のリスト（int itmlst[nitm]）

nitm　アイテム識別子の数

org　拡大縮小基準点

scf　縮尺値

scf[0] ： X 縮尺値。負の場合は Y 軸に対して反転する。
scf[1] ： Y 縮尺値。負の場合は X 軸に対して反転する。
縮尺値は、現在の座標系に対するものである。

【返り値】

0　：正常終了

9.8.8　アイテムの矩形配列を作成

【関数名】

edtaryrec

【機能】

指定されたアイテムの矩形配列を作成する。

【呼出し形式】

```
int edtaryrec(int keypic, const int itmlst[], int nitm,
              const short nele[], const DPOINT vec[])
```

【入力引数】

keypic　出力ピクチャ制御スイッチ

```
                0       :矩形配列をアクティブピクチャに作成する。
                >0      :矩形配列をピクチャ #keypic に作成する。
    itmlst      アイテム識別子のリスト (int itmlst[nitm])
    nitm        アイテム識別子の数
    nele        矩形配列の列の要素数
                    nele[0]         :  列１の要素数
                    nele[1]         :  列２の要素数
    vec         矩形配列の列の間隔 (DPOINT vec[2])
                    vec[0]          :  列１のピッチ
                    vec[1]          :  列２のピッチ
```

【返り値】

0　　: 正常終了

9.8.9　アイテムの回転配列を作成

【関数名】

edtarycir

【機能】

指定されたアイテムの回転配列を作成する。

【呼出し形式】

```
int edtarycir(int keypic, const int itmlst[], int nitm, int nele,
              const DPOINT& org, double ang, const DPOINT &bse)
```

【入力引数】

```
    keypic      出力ピクチャ制御スイッチ
                0       :回転配列をアクティブピクチャに作成する。
                >0      :回転配列をピクチャ #KEYPIC に作成する。
    itmlst      アイテム識別子のリスト (int itmlst[nitm])
    nitm        アイテム識別子の数
    nele        回転配列の要素数
    org         回転配列の回転中心点
    ang         回転配列の角度ピッチ（単位：度）
    bse         回転配列の回転基準点
```

【返り値】

0　　: 正常終了

9.8.10　アイテムの回転配列 (向心) を作成

【関数名】

edtaryrad

【機能】

指定されたアイテムの回転配列 (向心) を作成する。

【呼出し形式】

```
int edtaryrad(int keypic, const int itmlst[], int nitm, int nele,
              const DPOINT& org, double ang)
```

【入力引数】

keypic	出力ピクチャ制御スイッチ 0 ：回転配列（向心）をアクティブピクチャに作成する。 >0 ：回転配列（向心）をピクチャ #keypic に作成する。
itmlst	アイテム識別子のリスト（int itmlst[nitm]）
nitm	アイテム識別子の数
nele	回転配列の要素数
org	回転配列の回転中心点
ang	回転配列の角度ピッチ（単位：度）

【返り値】

0 ： 正常終了

第 10 章　幾何演算モジュール

10.1 図形アイテム

図形アイテムとは以下のアイテムをいいます。
左はヘッダーファイル acadprm.h に宣言してあるパラメータ名です。

ITMPOINT	**点アイテム**
ITMLINE	**線分アイテム**
ITMARC	**円弧アイテム**
ITMFREE	**自由曲線アイテム**
ITMSTRING	**ストリングアイテム**

線分、円弧、自由曲線、ストリングを総称して曲線アイテムとよびます。

曲線アイテムは、１つ以上の図形要素からなっています。
最初が曲線の始点のサブレコード
SITSTART
つぎが曲線図形要素のサブレコードの並び
SITLINE, SITARC または、SITBZCRV
これらの並びの間に他のサブレコードが入ってはなりません。
線分アイテムは複数の線分サブレコード SITLINE だけを含めることができ、それらは矛盾なく１つの直線を構成しなければなりません。つまり全ての点が一直線上にあり、始点から終点にむかって順番になければなりません。
円弧アイテム、自由曲線アイテムも同様に矛盾のないものでなければなりません。
ストリングアイテムだけは各種の曲線図形要素サブレコードを含むことができます。

10.1.1 図形要素の標準形

以下の関数で使用する主な図形要素の名称、データおよび型名はつぎの通りです。
型宣言はヘッダーファイル acaddef.h にあります。

点	P	DPOINT
ベクトル	V	DPOINT
線分	Ps, Uv, len	DVLINE
円弧	Ps, Pm, Pe, Pc, rad, ang	DARC
３次 Bezier 曲線	P0, P1, P2, P3, len	DBZC
線分（始点、終点表現）	Ps, Pe	DLINE
二次曲線	Po, a, b, Uv, ts, te	DCON
上記を含む共用体		DGEOM

ここで、P, Ps, Pm, Pe, Pc, Pi はいずれも点 (x,y) を表わし、Uv は正規化されたベクトル (i,j)、V はベクトル (dx,dy) を表わします。

len は長さを表わします。rad は半径を表わします。len, rad > 0 でなければなりません。

ang は円弧の中心角度を表わします。
中心点から始点への半直線を基準に中心点から終点への半直線までの角度をとります。
単位はラジアン。円弧の向きにより角度の符号が異なります。
　正の値 反時計廻り
　負の値 時計廻り
　0 < 絶対値 (ang) <= 2pi でなければなりません。

簡便のため、線分の別形式として (始点、終点) 表現を受けつけるモジュールもあります。そのときは明示してあります。

10.1.2 パラメータ

曲線 (線分、円弧、３次 Bezier, ２次曲線) をパラメータ表現した場合にパラメータは曲線上のある点を特定します。
ここでのパラメータは次のように定めてあります。

線分のパラメータ
　t = 線分の始点からの距離
　始点　t = 0
　線分の向きと同方向 0 < t
　線分の向きと逆方向 t < 0

円弧のパラメータ
　t = 円弧始点から、円弧の進行方向に沿って測った中心角
　始点　t = 0
　円弧の向きと同方向　0 ＜ t ≦ 2pi
　円弧の向きと逆方向 -2pi ≦ t ＜ 0

３次 Bezier 曲線のパラメータ
B(t) = (1-t)^3*Q1 + 3*t*(1-t)^2*Q2 + 3*t^2*(1-t)*Q3 + T^3*Q4
0 <= t <= 1
始点　t = 0
終点　t = 1

二次曲線の種類
PARABOLA : 放物線
ELLIPSE : 楕円
HYPERBOLA : 双曲線

二次曲線 (Po, a, b, Uv, ts, dt)

放物線
点 (Po) を原点、ベクトル (Uv) を正の X 軸とする座標系での曲線の方程式

$$\begin{cases} x : a*t\hat{}2 \\ y = 2*a*t \end{cases}$$

Po	頂点
a, b	放物線の方程式の定数（a > 0, b = 0）
Uv	頂点から焦点の向き（長さ１のベクトル）
ts	始点におけるパラメータ
dt	終点におけるパラメーター始点におけるパラメータ

楕円
点 (Po) を原点、ベクトル (Uv) を正の X 軸とする座標系での曲線の方程式

$$\begin{cases} x = a*\cos(t) \\ y = b*\sin(t) \end{cases} \qquad 0 <= t <= 2pi$$

Po	中心点
a, b	楕円の方程式の定数（長軸と短軸の半径）(a,b > 0)
Uv	正の長軸の向き（長さ１のベクトル）
ts	始点の離心角（ラジアン）
dt	終点の離心角ー始点の離心角（ラジアン）

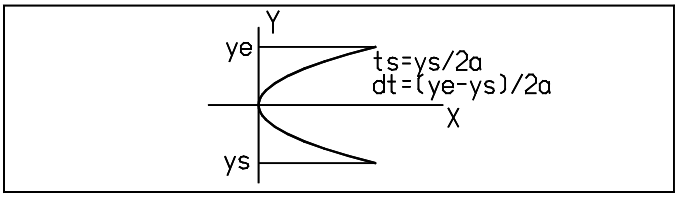
双曲線
点 (Po) を原点、ベクトル (Uv) を正の X 軸とする座標系での曲線の方程式

$$\begin{cases} x = a*\sec(t) \\ y = b*\tan(t) \end{cases} \qquad -pi/2 < t < pi/2$$

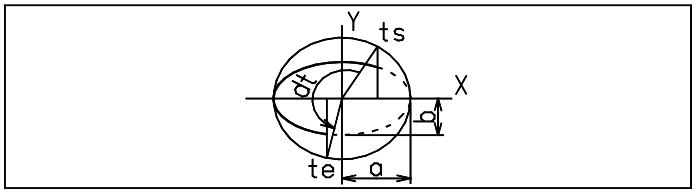
t = -pi/2, pi/2 付近では x,y の値が極めて大きくなるので注意。

Po	中心点
a, b	双曲線の方程式の定数（a = 補助円の半径）(a,b > 0)
Uv	中心点から頂点の向き（長さ１のベクトル）
ts	始点の離心角（ラジアン）
dt	終点の離心角ー始点の離心角（ラジアン）

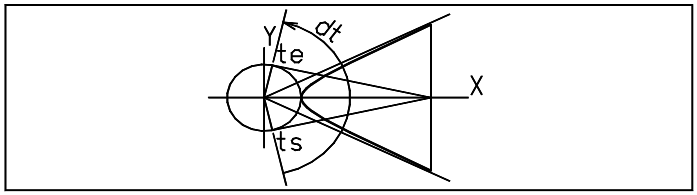
放物線



楕円



双曲線



10.1.3 定数

以下の関数では使用する定数があらかじめヘッダーファイル acadprm.h に宣言してあります。

関数名	機能
SITPOINT	点
SITLINE	線分。始点、終点表現の場合は（-SITLINE）とする。
SITARC	円弧
SITBZCRV	３次 Bezier 曲線
SITVECTOR	ベクトル
CCW	反時計廻り
CW	時計廻り
NARROW	せまい角度を取る
LEFT	左側
RIGHT	右側
INSIDE	内側

関数名	機能
OUTSIDE	外側
HALFPI	pi/2
PI	pi
TWOPI	2pi
DGRTORAD	度をラジアンに単位変換
RADTODGR	ラジアンを度に単位変換

10.2 幾何アイテム

● 機能

以下のモジュールは図形アイテムレベルでの図形処理を行ないます。図形要素レベルの処理ではありません。
モジュールによっては、引数として図形要素データを渡すものがあります。このとき、線分要素は原則としてデータベースの線分要素の形式と合せるため、パラメータ表現ではなく始点終点の形式です。

線分　(Ps, Pe)

自由曲線アイテムのトリムされた部分は処理対象とはなりません。

● 関数一覧

点

関数名	機能
gmpin02	交点
gmpmn00	最短点
gmpntdef	定義点
gmpntpdg	デジタイズ点
gmpntpdv	等分割点
gmpntpnd	端点
gmpon02	投影点
gmpntse	曲線アイテムの端点を得る

線分

関数名	機能
gmltn02	接線

円弧

関数名	機能
gmfil02	接円弧

自由曲線

関数名	機能
tmpgfree	与えられた点列から自由曲線アイテムを作る

アイテムの操作

関数名	機能
gmofs	オフセットアイテムを作る
gmexpcrvs	曲線アイテムの分解
gmsptcrv	曲線アイテムの２分割

ユーティリティ

関数名	機能
gmdbseg	アイテムから図形要素を取り出す。
gmitmlst	アイテムのリストを作る
gmlength	曲線長

10.2.1 指定アイテムから順次図形要素を取得

【関数名】

gmdbseg

【機能】

指定のアイテムから順次図形要素を取り出す。

【呼出し形式】

```
int gmdbseg(Gm_dbseg *cblk, short *type, short *sfnt, DGEOM *geom)
```

【入力引数】

cblk　制御データ。最初の呼び出しのときだけ次の値を指定する。
- cblk->idptr　：アイテムの識別子
- cblk->slot　：データベース参照番号（1 または 2）
- cblk->snum　：最初の呼び出しであることを示す値。
- -2　：自由曲線のトリムされた部分も取り出す。
  自由曲線以外は -2 のほうがよい。
- -1　：自由曲線のトリムされた部分は取り出さない。

【出力引数】

cblk　制御データ。次の呼び出し時に使用するので内容を変更しないこと。
- cblk->snum　：サブレコードの相対番号（1-）.

type　図形要素の種類
SITPOINT, SITLINE, SITARC, SITBZCRV

sfnt　図形要素の線種類
- 1　：非表示
- 0　：アイテムと同じ
- 2-7　：線種（1-6）

geom　　図形要素

【返り値】

0　　: 正常終了
1　　: エラー
2　　: アイテムの終了

注意

(1)　このモジュールは図形要素だけ取り出す。

(2)　複数のアイテムを同時にオープンするときは、cblk->slot に設定する値を違えること。その場合は制御データはそれぞれべつに必要。

(3)　データベースアイテムのクローズ
このモジュールは最初の呼び出し (cblk->snum < 0) のときデータベースアイテムをオープンする。順次図形データを取出し、終了 (返り値 2) またはエラーが発生 (返り値 1) したときは、データベースアイテムをクローズする。返り値 0 のときは次の図形データ呼出しのため、オープンしたままである。もしこの状態でアイテムの読み出しを終了したいならば、必ずクローズ処理をしなければならない。DbUrdclos を使う。

例

```
short type, sfnt;
Gm_dbseg cblk;
DGEOM seg;

cblk.snum = -2;
cblk.idptr = idptr;
cblk.slot = 1;
while (gmdbseg(&cblk, &type, &sfnt, &seg) == 0) {
  printf("Segment type number is %d¥n", type);
}
```

10.2.2 曲線アイテムの分解

【関数名】

gmexpcrvs

【機能】

曲線アイテムの分解

【呼出し形式】

```
int gmexpcrvs(const int idptrs[], int icnt)
```

【入力引数】

idptrs　　分解するアイテムのアイテム識別子の並び (int idptrs[icnt])
icnt　　アイテムの数。

【返り値】

分解してできたアイテムの数。
新しくできたアイテムはデータベースに格納される。

注意

(1) アイテムの中の１つの図形要素が１つのアイテムとなる。
ただし、３次 Bezier 曲線要素が連続している場合は別である。連続する３次 Bezier 曲線要素は１つの自由曲線アイテムとする。連続していても図形要素のラインフォントが変わるときは、そこまでで、ひとつの自由曲線となる。
(2) 分解してできるアイテムの線種はその図形要素の線種である。
(3) 自由曲線のトリムされている部分、曲線アイテムの部分ブランクはアイテムとはならない。
(4) もとのアイテムは消去される。

10.2.3 ２アイテムの接円弧を計算

【関数名】

gmfil02

【機能】

ふたつのアイテムの接円弧を計算する。
アイテムのピック位置に近い接円弧 (もしあれば複数) を計算する。

【呼出し形式】

```
int gmfil02(const int itmptr[][3], const DGEOM seg[], double rad,
            int maxarc, int srseq[][2], DARC arcs[])
```

【入力引数】

itmptr　アイテムのデータ（itmptr[2][3]）
　itmptr[][0]　アイテム識別子
　itmptr[][1]　サブレコードの相対番号
　itmptr[][2]　図形要素の種類
　　SITPOINT、SITLINE、SITARC、SITBZCRV
seg　図形要素データ（DGEOM seg[2]）

以上はアイテムをピックした位置を与えるもので、Ident00 の副作用として得られる。

rad　半径
maxarc　接円弧を受け取る配列の大きさ

【出力引数】

srseq　円弧の両端が接するサブレコードの相対番号
（int srseq[maxarc][2]）
　srseq[][0]　j 番目の接円弧の始点が接する第一番目の曲線のサブレコードの番号
　srseq[][1]　j 番目の接円弧の終点が接する第二番目の曲線のサブレコードの番号
arcs　接円弧（DARC arcs[maxarc]）

【返り値】

接円弧の数

注意
(1) 最初に与えられた図形要素データで接円弧を計算する。
接円弧が得られれば、それで終了。
(2) 次にピックされた図形要素の隣の図形要素をとりだし計算する。
接円弧が得られれば、それで終了。
そうでなければ、さらに隣の図形要素をとりだし計算する。
これは、アイテムの一方が自由曲線のときのみ行う。
このとき、自由曲線を扱い易いように作業領域に展開する一各図形要素を直接参照できるようにす

る。

例

```
#define   MAXARC 16
int itmptr[2][3], srseq[MAXARC][2];
DGEOM seg[2];
DARC farc[MAXARC];
TOKEN token;

/* Pick two curve items */
tknclear(&token);
token.typ = TknDIG;
for (int j = 0; j < 2; ++j) {
  token.pnt.x += 100.0;
  token.pnt.y += 100.0;
  if ((idptr = IdentItem(1, &token, 1)) < = 0)
    return;
  if (j == 1 && idptr == itmptr[0][0])
    return;

  const IDENTINFO* info = IdentInfo();
  itmptr[j][0] = info[0].idptr;
  itmptr[j][1] = info[0].snum;
  itmptr[j][2] = info[0].styp;
  seg[j] = info[0].geom;
}
/* Calculate Fillet arcs */
gmfil02(itmptr, seg, 15.0, MAXCNT, srseq, farc);
```

10.2.4 ピクチャ・アイテムタイプ・クラス・レビジョン・線種・線幅を取得

【関数名】

gmitmlst

【機能】

指定のピクチャ、アイテムタイプ、クラス、レビジョン、線種、線幅のアイテムを取り出す。

【呼出し形式】

```
int gmitmlst(int iswt, const short mpic[], const short mtyp[],
             const short mcls[], const short mrev[],
             const short mlft[], const short mlwt[],
             int maxnum, int idptrs[])
```

【入力引数】

iswt	探索する対象。0 : データベース、1 : アクティブリスト。
mpic	ピクチャマスク (short array, MAXITMPIC bits)
mtyp	アイテムマスク (short array, MAXITMTYP bits)
mcls	クラスマスク (short array, MAXITMCLS bits)
mrev	レビジョンマスク (short array, MAXITMREV bits)
mlft	線種マスク (short array, MAXITMLFT bits)
mlwt	線幅マスク (short array, MAXITMLWT bits)

mpic, mtyp, mcls, mrev, mlft, mlwt はビット列。

bit i == 0 : 選択しない
== 1 : 選択する。
maxnum 最大アイテム数。配列 idptrs の大きさ

【出力引数】

idptrs 選択したアイテム識別子のならび（idptrs[maxnum]）

【返り値】

選択したアイテム識別子の数

10.2.5 曲線アイテムの長さを計る

【関数名】

gmlength

【機能】

曲線アイテムの長さを得る。

【呼出し形式】

```
double gmlength(int idptr, int spos, int tpos)
```

【入力引数】

idptr アイテム識別子
spos 開始サブレコードの相対番号
tpos 終端サブレコードの相対番号
spos == 0, tpos == 0 とすると曲線アイテム全体の長さを得る。

【返り値】

曲線長さ

10.2.6 ２曲線アイテムの接線を計算

【関数名】

gmltn02

【機能】

ふたつの曲線アイテムの接線を計算する。
曲線アイテムのピック位置に近い接線 (もしあれば複数) を計算する。

【呼出し形式】

```
int gmltn02(const int itmptr[][3], const DGEOM seg[],
            int maxlin, DLINE lins[])
```

【入力引数】

itmptr アイテムのデータ（itmptr[2][3]）

itmptr[][0]　　アイテム識別子
itmptr[][1]　　サブレコードの相対番号
itmptr[][2]　　図形要素の種類
SITPOINT、SITLINE、SITARC、SITBZCRV

seg　　図形要素データ (DGEOM seg[2])

以上はアイテムをピックした位置を与えるもので、Ident00 の副作用として得られる。
ベクトルは、アイテムの中には含まれていないので次のようにする。
itmptr[][0] = 0
itmptr[][1] = 0
点もベクトルと同じようにして良い。

maxlin　接線を受け取る配列の大きさ

【出力引数】

lins　　接線 (Ps, Pe)

【返り値】

接線数

注意

(1)　最初に与えられた図形要素データで接線を計算する。
接線が得られれば、それで終了。

(2)　つぎにピックされた図形要素の隣の図形要素をとりだし計算する。
接線が得られれば、それで終了。
そうでなければ、さらに隣の図形要素をとりだし計算する。
これは、アイテムの一方が自由曲線のときだけ行なう。
このとき、自由曲線を扱い易いように作業領域に展開する－各図形要素を直接参照できるようにする。

(3)　ベクトルを使用するときは他方はベクトル以外でなければならない。

例

```
#define   MAXLIN 16
short itmptr[2][3];
DGEOM seg[2];
DLINE lins[MAXLIN];
TOKEN token;

/* Set point (0,0) */
itmptr[0][0] = 0;
itmptr[0][1] = 0;
itmptr[0][2] = SITPOINT;
seg[0].pnt.x = 0.0;
seg[0].pnt.y = 0.0;

/* Pick curve item */
tknclear(&token);
token.typ = TknDIG;
token.pnt.x = 100.0;
token.pnt.y = 100.0;
if(IdentItem(1, &token, 1) <= 0)
  return;
const IDENTINFO* info = IdentInfo();
itmptr[1][0] = info[0].idptr;
itmptr[1][1] = info[0].snum;
```

```
itmptr[1][2] = info[0].styp;
seg[1] = info[0].geom;

/* Calculate tangent */
gmltn02(itmptr, seg, MAXLIN, lins);
```

10.2.7 曲線アイテムのオフセット

【関数名】

gmofs

【機能】

曲線アイテムのオフセット

【呼出し形式】

```
int gmofs(int idptr, const int sqnum[], int ofsdir, double ofsdst,
          int ofstyp, double ratio, int ifcswt, int ifwswt)
```

【入力引数】

idptr　曲線アイテムのアイテム識別番号

sqnum　結合アイテム中のひとつの線分、円弧、ストリング、自由曲線をオフセットするとき使用する。

sqnum[0]　：　オフセットする部分の最初のサブレコードのシーケンス番号

sqnum[1]　：　オフセットする部分の最後のサブレコードのシーケンス番号

1 <= sqnum[0] < sqnum[1] < そのアイテムの最後のサブレコードのシーケンス番号

線分アイテム、円弧アイテム、ストリングアイテム、自由曲線アイテムで、曲線全体をオフセットする場合は、sqnum[0]＝0 と設定する。

ofsdir　オフセット方向。LEFT ： 左，RIGHT ： 右

ofsdst　オフセット距離

ofstyp　補間モード。隣り合うオフセットセグメントがつながらないときの接続方法を指定する。

0　：セグメントの端点での接線の交点を使用する。
セグメントの終点－交点－次の始点 を結ぶ線分を挿入する。
接線の交点がない場合は、']' 形の３本の線分を挿入する。
']' は (ratio * ofsdst) だけ突き出す。

1　：上記と同じ。ただし交点がもとのセグメントから大きく離れるときはその突出しを切り取る。

2　：円弧を挿入する

ratio　狩り込み比 (1.0 < ratio < 10.0)。ofstyp = 0,1 の場合のみ参照。
突出しの高さが (ratio * ofsdst) 以上になると突出しを切り取る。

ifcswt　干渉除去スイッチ。0 ： 干渉除去しない，1 ： 干渉除去する

ifwswt　部分線種／線幅継承スイッチ。0 ： 継承しない，1 ： 継承する

【返り値】

0　：正常終了

1　：エラー

7　：正常終了だが、線分セグメントまたは円弧セグメントの挿入が行なわれたのでオフセットカーブはストリングアイテムで作成した。

注意

(1)　オフセットしてできるアイテムはテンポラリアイテム。アイテムタイプ、クラスなどの属性は、事前に設定しておくこと。

(2) もとの曲線が閉じたストリングアイテムのときは、オフセットしてできたストリングアイテムも自動的に閉じる。
(3) 自由曲線アイテムをオフセットするとき、補間のために直線セグメントまたは円弧セグメントが挿入される場合は、ストリングアイテムとして作成され、返り値には 7 がセットされる。

例

```
ItmAtr itmatr;

int idptr = 1;
int sqnum[] = {0, 0};
int ofsdir = LEFT;
int ofstyp = 2;
int ifcswt = 0;
int ifwswt = 0;
double ofsdst = 5.0;
double ratio = 2.0;

/* アイテムタイプを調べる */
if (Dbvriatr(idptr, &itmatr))
  return;
if (itmatr.typ < ITMLINE || itmatr.typ > ITMSTRING)
  return;

/* オフセット */
SetItemtype(itmatr.typ);
int ier = gmofs(idptr, sqnum, ofsdir, ofsdst, ofstyp, ratio, ifcswt, ifwswt);
if (ier)
  return;
TmpgWrtoDB(0);
```

10.2.8 ２つの曲線アイテムの交点を計算

【関数名】

gmpin02

【機能】

ふたつの曲線アイテムの交点を計算する。
曲線アイテムのピック位置に近い交点 (もしあれば複数) を計算する。

【呼出し形式】

```
int gmpin02(const int itmptr[][3], const DGEOM seg[],
            int maxpnt, DPOINT pnts[])
```

【入力引数】

itmptr　アイテムのデータ (itmptr[2][3])
　itmptr[][0]　アイテム識別子
　itmptr[][1]　サブレコードの相対番号
　itmptr[][2]　図形要素の種類
　SITLINE、SITARC、SITBZCRV
seg　図形要素データ (DGEOM seg[2])

以上はアイテムをピックした位置を与えるもので、Ident00 の副作用として得られる。

maxpnt　交点を受け取る配列の大きさ

【出力引数】

pnts　　交点を受け取る配列

【返り値】

交点数

注意

(1)　最初に与えられた図形要素データで交点計算する。
　　交点が得られれば、それで終了。

(2)　つぎにピックされた図形要素の隣の図形要素をとりだし計算する。
　　点が得られれば、それで終了。
　　そうでなければ、さらに隣の図形要素をとりだし計算する。

これは、アイテムの一方が自由曲線のときのみおこなう。
このとき、自由曲線を扱い易いように作業領域に展開する－各図形要素を直接参照できるようにする。

例

```
#define MAXPNT 16
short itmptr[2][3];
DGEOM seg[2];
DPOINT pnts[MAXPNT];
TOKEN token;

/* Pick two curve items */
tknclear(&token);
token.typ = TknDIG;
for (int j = 0; j < 2; ++j) {
  token.pnt.x += 100.0;
  token.pnt.y += 100.0;
  if (IdentItem(1, &token, 1) <= 0)
    return;
  const IDENTINFO* info = IdentInfo();
  if (j == 1 && info[0].idptr == itmptr[0][0])
    return;

  itmptr[j][0] = info[0].idptr;
  itmptr[j][1] = info[0].snum;
  itmptr[j][2] = info[0].styp;
  seg[j] = inofo[0].geom;
}

/* Calculate intersection points */
gmpin02(itmptr, seg, MAXPNT, pnts);
```

10.2.9　２つの曲線アイテムの最短点を計算

【関数名】

gmpmn00

【機能】

ふたつの曲線アイテムの最短点を計算する。

【呼出し形式】

```
int gmpmn00(int iswt, const int itmptr[][3], const DGEOM seg[], const DPOINT* pner,
            double* dst, DPOINT pnts[])
```

【入力引数】

iswt　　線分、円弧を延長するかどうか指示する。
　　0　　：延長する。線分は無限直線、円弧は円として処理する。
　　1　　：延長しない。

itmptr　　アイテムのデータ（itmptr[2][3]）
　　itmptr[][0]　　アイテム識別子
　　itmptr[][1]　　サブレコードの相対番号
　　itmptr[][2]　　図形要素の種類。SITPOINT, SITLINE, SITARC, SITBZCRV

seg　　図形要素データ (DGEOM seg[2])

以上はアイテムをピックした位置を与えるもので、Ident00 の副作用として得られる。

pner　　点。平行なとき、最短点の得たい場所を指示する。

【出力引数】

dst　　最短距離。返り値が 3 のときは２直線の角度（ラジアン）
pnts　　最短点 (DPOINT pnts[2])

【返り値】

0　　：最短距離が求まった
1　　：交差している。
2　　：この組み合わせは不可。
3　　：２つの線分が交差している。

注意

(1)　延長モード（ISWT==0）で、入力が点、線分、円、ストリングの時は配列 seg の図形要素を使用する。このとき、線分は直線とし、円弧は円として処理する。
入力が自由曲線のときは配列 seg は参照せず曲線全体を対象にする。

(2)　平行な二直線や２つの同心円のように、全てにわたって平行の場合には点 pner をそれぞれに投影した点を最短点とする。

例

```
int itmptr[2][3];
DGEOM seg[2];
DPOINT pnts[2];
TOKEN token;

/* Set point */
itmptr[0][0] = 0;
itmptr[0][1] = 0;
itmptr[0][2] = SITPOINT;
seg[0].pnt.x = 0.0;
seg[0].pnt.y = 0.0;

/* Pick curve */
tknclear(&token);
token.typ = TknDIG;
token.pnt.x = 100.0;
token.pnt.y = 100.0;
if (IdentItem(1, &token, 1) <= 0)
  return;
const IDENTINFO* info = IdentInfo();
```

```
itmptr[1][0] = info[0].idptr;
itmptr[1][1] = info[0].snum;
itmptr[1][2] = info[0].styp;
seg[1] = info[0].geom;

gmpmn00(0, itmptr, &p, dummy, &dst, pnts);
```

10.2.10 デジタイズ点の計算

【関数名】

gmpntpdg

【機能】

デジタイズ点を計算する。

【呼出し形式】

```
void gmpntpdg(const DPOINT* pnt, DPOINT* pout)
```

【入力引数】

pnt	位置

【出力引数】

pout	点

注意

グリッドが有効ならば、入力位置に最も近いグリッド点を返す。

10.2.11 アイテムの定義点を取得

【関数名】

gmpntdef

【機能】

アイテムの定義点を得る。

【呼出し形式】

```
int gmpntdef(int idptr, int maxpnt, DPOINT *pnts)
```

【入力引数】

idptr	アイテム識別子
maxpnt	最大点数

【出力引数】

pnts	定義点

【返り値】

点数

注意
線分　　始点、[終点]+
円弧　　始点、[中間点, 終点]+
自由曲線　始点、[終点]+

[]+ は、アイテムが複数の図形要素を含むときの繰返し記号。

10.2.12 曲線アイテムを等分割する点を計算

【関数名】

gmpntpdv

【機能】

曲線アイテムを等分割する点群を計算する。

【呼出し形式】

```
int gmpntpdv(int iswt, int idptr, int maxpnt, double dst,
             const DPOINT* bpnt, DPOINT pnts[])
```

【入力引数】

iswt　　分割方法
　　-3　　：基準点とピッチを指示（基準点から始点へ計算する）
　　-2　　：基準点とピッチを指示（基準点から終点へ計算する）
　　1　　：点数指示
　　2　　：ピッチ指示（余りは終点側）
　　3　　：ピッチ指示（余りは始点側）
idptr　　アイテム識別子

maxpnt　点数 (0 < maxpnt)。
　　iswt == 1　点数
　　maxpnt == 1　　中点
　　maxpnt == 2　　始点と終点
　　閉じた曲線アイテムでは最初と最後の点は同じ点になる。
　　iswt == -2, -3, 2, 3 最大点数
dst　　ピッチ (iswt == -3, -2, 2, 3 のときのみ)
bpnt　　基準点 (iswt == -3, -2 のときのみ)

【出力引数】

pnts　　点列

【返り値】

出力点数。
-1　　: 基準点が曲線アイテム上に存在しない
　　(iswt == -3, -2 のときのエラー)

10.2.13 アイテムの端点を取得

【関数名】

gmpntpnd

【機能】

アイテムの端点を得る。

【呼出し形式】

```
int gmpntpnd(int idptr, double dst, DPOINT *pnts)
```

【入力引数】

idptr	アイテム識別子
dst	端点からの距離 正の値　他の端点にむかって ゼロ　　真の端点 負の値　他の端点から離れる方にむかって

【出力引数】

pnts	端点 (DPOINT pnts[2])

【返り値】

点数 (0 or 2)

10.2.14 曲線アイテムの端点を取得

【関数名】

gmpntse

【機能】

曲線アイテムの端点を得る。

【呼出し形式】

```
int gmpntse(int idptr, DPOINT pnts[])
```

【入力引数】

idptr	アイテム識別子。ストリングアイテム、自由曲線、円弧、または線分アイテム。

【出力引数】

pnd	始点と終点 (DPOINT pnts[2])

【返り値】

0	: 正常終了
1	: エラー

10.2.15 点を曲線アイテムに投影

【関数名】

gmpon02

【機能】

点を曲線アイテムに投影する。
曲線アイテムのピック位置に近い点 (もしあれば複数) を計算する。

【呼出し形式】

```
int gmpon02(const int itmptr[][3], const DGEOM seg[], const DPOINT* pnt,
            int maxpnt, DPOINT pnts[])
```

【入力引数】

itmptr　アイテムのデータ (itmptr[2][3])
　itmptr[][0]　アイテム識別子
　itmptr[][1]　サブレコードの相対番号
　itmptr[][2]　図形要素の種類。SITLINE、SITARC、SITBZCRV
seg　図形要素データ (DGEOM seg[2])

以上はアイテムをピックした位置を与えるもので、Ident00 の副作用として得られる。

pnt　投影する点
maxpnt　点を受け取る配列の大きさ

【出力引数】

pnts　投影点

【返り値】

点数

注意

(1)　最初に与えられた図形要素データで投影点計算する。
点が得られれば、それで終了。

(2)　つぎにピックされた図形要素の隣の図形要素をとりだし計算する。
点が得られれば、それで終了。
そうでなければ、さらに隣の図形要素をとりだし計算する。
これは、アイテムが自由曲線のときのみおこなう。
このとき、自由曲線を扱いやすいように作業領域に展開するー各図形要素を直接参照できるようにする。

例

```
#define MAXPNT 16

int itmptr[1][3];
TOKEN token;
DPOINT pnts[MAXPNT];

/* Pick curve */
tknclear(&token);
token.typ = TknDIG;
token.pnt.x = 100.0;
token.pnt.y = 100.0;
if (IdentItem(1, &token, 1) <= 0)
  return;
```

```
/* Calculate projection */
const IDENTINFO* info = IdentInfo();
itmptr[0][0] = info[0].idptr;
itmptr[0][1] = info[0].snum;
itmptr[0][2] = info[0].styp;
gmpon02(itmptr, &info[0].geom, &pbase, MAXPNT, pnts);
```

10.2.16 曲線アイテムを指定位置で２分割

【関数名】

gmsptcrv

【機能】

曲線アイテムを指定位置で２分割する。

【呼出し形式】

```
int gmsptcrv(const Gm_brkpos* brkpos, int idptr)
```

【入力引数】

brkpos　分割位置を指示するデータ

brkpos.idptr	アイテムの識別子
brkpos.snum	サブレコードの相対番号
brkpos.type	図形要素の種類。SITPOINT, SITLINE, SITARC, SITBZCRV
brkpos.pnt	図形要素をピックした位置 (DPOINT)
brkpos.geom	図形要素データ (DGEOM)

これらのデータはアイテムを分割する位置を与えるもので、Ident00 の副作用として得られる。

idptrs　分割されるアイテムの識別子

【返り値】

0　：正常終了
1　：エラー

注意

(1)　新しくできたアイテムはデータベースに格納される。
もとのアイテムは消去する。

(2)　分割位置指定が点のときは投影点で２分割する。
分割位置指定が曲線のときは交点で２分割する。

例 . 分割位置を指示するデータの設定方法

```
DPOINT p = {50.0, 50.0};
int idptr = 1;

Gm_brkpos brkpos;

if (boundary is point) {
  brkpos.geom.pnt = p;
  brkpos.idptr = 0;
  brkpos.snum = 0;
  brkpos.type = SITPOINT;
  brkpos.pnt = brkpos.seg.pnt;
```

```
} else if (boundary is picked curve) {
  TOKEN token;
  tknclear(&token);
  token.typ = TknDIG;
  token.pnt = p;
  if (IdentItem(1, &token, 1) <= 0)
    return;
  const IDENTINFO* info = IdentInfo();
  brkpos.idptr = info[0].idptr;
  brkpos.snum = info[0].snum;
  brkpos.type = info[0].styp;
  brkpos.geom = info[0].geom;
  brkpos.pnt = p;
} else {
  return;
}
gmsptcrv(&brkpos, idptr);
```

10.2.17 自由曲線を作成

【関数名】

tmpgfree

【機能】

自由曲線を作成する。
３次元点列のとき、曲線はアクティブピクチャ上の２次元曲線になる。

【呼出し形式】

```
int tmpgfree(int ndim, int npnt, const double pnts[], int cswt, int sswt)
```

【入力引数】

ndim	入力点列の次元	
	2	：２次元
	3	：３次元
npnt	入力点数 (3 <= npnt <= 512)	
pnts	点列 (pnts[npnt][ndim])	
cswt	閉曲線スイッチ	
	1	：閉曲線。閉曲線を指定したときは最初の点と最後の点を同じにしないこと。
	0	：開曲線
sswt	始点サブレコード (SITSTART) を出力しないときのスイッチ	
	0	：出力
	1	：出力しない

【返り値】

0	： 正常
1	： 入力点数が制限をこえた
2	： 重複点がある
3	： すべての点が直線上にある
4	： 計算不能

注意
このモジュールは、アクティブ テンポラリ アイテムに自由曲線のレコードを追加する。

例.

```
SetItemtype(ITMFREE);
TmpgOpen1(0, 0);
tmpgfree(省略);
TmpgClose(1);
```

10.3 面積計算

● 関数一覧

gmpro	**領域の面積を計算する**
gmpro2	**回転体の表面積と体積を計算する**

10.3.1 領域面積の計算

【関数名】

gmpro

【機能】

領域の面積を計算する。

【呼出し形式】

```
int gmpro(int numitm, const int idptrs[], double prop[])
```

【入力引数】

numitm　領域の外周を構成するアイテムの数。

idptrs　領域の外周を構成するアイテムのアイテム識別子の並び。
(int idptrs[numitm])
外周は捻れていたり、交差するようなものであってはならない。

単一アイテムのとき (numitm == 1)
閉じたストリングアイテム、閉じた自由曲線 または円でなければならない。

複数アイテムのとき (numitm ≧ 2)
複数のアイテムでひとつの閉じた外周を構成する。含めてもよいアイテムタイプはストリングアイテム、自由曲線、円弧、線分アイテム。隣合うアイテムは端点が連接していなければならない。また、外周全体の向きと逆向きのアイテムは負のアイテム識別子とすること。

【出力引数】

prop　計算結果 (double prop[11])
モーメントは補助座標 (WCS) に対して計算する。
[0] 周長
[1] 面積
[2] 断面一次モーメント WCS X-軸
[3] 断面一次モーメント WCS Y-軸
[4] 断面二次モーメント WCS X-軸
[5] 断面二次モーメント WCS Y-軸
[6] 断面相乗モーメント
[7] 縁距離。WCS-X 軸の最小座標
[8] 縁距離。WCS-Y 軸の最小座標

[9] 縁距離。WCS-X 軸の最大座標
[10] 縁距離。WCS-Y 軸の最大座標

【返り値】

0 : エラー
1 : 領域外周の向きは反時計廻り (CCW)
2 : 領域外周の向きは時計廻り (CW)

例

```
void uarea(TOKEN *token) {
  ItmAtr itmatr;
  int idptr;
  double precision prop[11];
  DPOINT p[2];

  /* 境界アイテムのピック */
  if ((idptr = IdentItem(1, token, 1)) <= 0)
    return;

  /* アイテムタイプの確認 */
  if (Dbvriatr(idptr, &itmatr))
    return;
  if (itmatr.typ < ITMARC || itmatr.typ > ITMSTRING)
    return;

  /* 閉じていることを確認 */
  if (gmpntse(idptr, p))
    return;
  if (fabs(p[0].x - p[1].x) > 1.0e-3 ||
      fabs(p[0].y - p[1].y) > 1.0e-3)
    return;

  /* 計算 */
  if (gmpro(1, &idptr, prop) == 0)
    return;

  /* 結果 */
  printf("Perimeter   L = %g", prop[0]);
  printf("Area        A = %g", prop[1]);
  printf("Moment     Gx = %g", prop[2]);
  printf("           Gy = %g", prop[3]);
  printf("Inertia    Ix = %g", prop[4]);
  printf("           Iy = %g", prop[5]);
  printf("          Ixy = %g", prop[6]);
  printf("Center     Cx = %g", prop[3] / prop[1]);
  printf("           Cy = %g", prop[2] / prop[1]);
  printf("Principal axis= %g",
          RADTODGR * 0.5 * atan2(-2.0 * prop[6]), prop[4] - prop[5]);
}
```

10.3.2 回転体の体積を計算

【関数名】

gmpro2

【機能】

回転体の体積を計算する。
断面を補助座標 (WCS) の X 軸に対して回転してできる回転体の体積を計算する。

【呼出し形式】

```
int gmpro2(int numitm, const int idptrs[], double prop[])
```

【入力引数】

numitm　回転体の断面を構成するアイテムの数。
idptrs　回転体の断面を構成するアイテムの識別子の並び。
(int idptrs[numitm])
複数アイテムで断面を構成する場合は、隣合うアイテムは端点が連接していなければならない。また、断面全体の向きと逆向きのアイテムは負のアイテムの識別子とすること。
含めてもよいアイテムタイプはストリング、自由曲線、円弧、線分アイテム。
断面は捻れていたり、交差するようなものであってはならない。
断面は補助座標 (WCS) の X 軸に対して回転するので、その軸と交差していると計算できない。

【出力引数】

prop　計算結果 (double prop[2])
[0] 表面積
[1] 体積

【返り値】

0　: 正常終了
1　: 計算できません

10.4 図形要素

● 関数一覧

点の作成

gmupin	**交点**
gmupon	**投影点**
gmutolc	**円弧上に点を作成する**
gmutolz	**３次 Bezier 曲線上点を作成する**

線分の作成

gmuline	**２点**
gmultn	**接線**

円の作成

gmucir	**中心点と円周の点**
gmucrc	**中心点と半径**
gmuctn2g	**２つの図形要素の接円**
gmuctn3g	**３つの図形要素の接円**
gmuctn	**点を中心とする接円を計算する**
gmufil	**２つの図形に接する接円を計算する**

円弧の作成

gmuarc	**中心点、始点と終点**
gmuctp	**円上の３点**

オフセット

gmuoffc	**円弧のオフセット**
gmuoffl	**線分のオフセット**
gmuoffz	**３次 Bezier 曲線のオフセット**

判定

gmuinout	**点が多角形の内部にあるか**
gmuside	**点が図形要素の左側にあるか**
gmuontest	**点が図形要素に乗っているか**
gmurelat	**２つの多角形の関係をしらべる**

距離

gmudst	**２点間の距離**
gmudstcrv	**２つの図形要素の最短距離**
gmudstpnt	**点と図形要素の最短距離**

角度

gmuang2v	**２つのベクトルの成す角度**
gmuang3p	**３つの点の成す角度**

クリッピング

gmuclip1 **線分を矩形領域でクリップする**
gmuclip2 **線分を凸多角形領域でクリップする**
gmuclip3 **閉多角形を多角形領域でクリップする**

その他
gmueval **図形要素の点をきめる**
gmudiv **図形要素の等分割**
gmucmp **補円弧を作成する**
gmurvs **図形要素の方向を反転する**
gmusptcrv **図形要素を指定位置で２つに分割する**
gmuexten **図形要素を包む最小矩形**
gmuqtoz **２次曲線を近似する自由曲線を計算する**

10.4.1 ２つのベクトルの成す角度を計算

【関数名】

gmuang2v

【機能】

２つのベクトルの成す角度を計算する

【呼出し形式】

```
double gmuang2v(const DPOINT* uv1, const DPOINT* uv2, int dir)
```

【入力引数】

uv1,uv2	正規ベクトル		
dir	角度測定方向		
	CCW	:	反時計廻り
	CW	:	時計廻り
	NARROW	:	せまい角度を取る

【返り値】

uv1 を基準として測った角度 (ラジアン)
正の値は反時計廻り、負の値は時計廻り。

10.4.2 ３点の成す角度を計算

【関数名】

gmuang3p

【機能】

３点の成す角度を計算する

【呼出し形式】

```
int gmuang3p(const DPOINT* pc, const DPOINT* ps, const DPOINT* pe, int dir, double* ang)
```

【入力引数】

pc, ps, pe ３点
dir　　角度測定方向
　　CCW　　：　反時計廻り
　　CW　　：　時計廻り
　　NARROW　　：　せまい角度を取る

【出力引数】

ang　　pc を中心として ps から pe へ測った角度（ラジアン）
　　正の値は反時計廻り、負の値は時計廻り。

【返り値】

0　　：正常終了
1　　：角度をはかれない

10.4.3 中心点・始点・終点から円弧を計算

【関数名】

gmuarc

【機能】

中心点、始点、終点 を与えて円弧を計算する

【呼出し形式】

```
int gmuarc(const DPOINT* pc, const DPOINT* ps, const DPOINT* pe, int dir, DARC* arc)
```

【入力引数】

pc, ps, pe　中心点、始点、終点
dir　　円弧の方向
　　CCW　　：　反時計廻り
　　CW　　：　時計廻り
　　NARROW　　：　せまい弧を取る

【出力引数】

arc　　円弧。半径は pc と ps の 距離

【返り値】

0　　：正常終了
1　　：円弧が作れない

10.4.4 中心点と始点から円を計算

【関数名】

gmucir

【機能】

中心点と始点 を与えて円を計算する

【呼出し形式】

```
int gmucir(const DPOINT* pc, const DPOINT* ps, DARC* cir)
```

【入力引数】

pc, ps　　中心点、始点

【出力引数】

cir　　円（半径は pc と ps の距離）

【返り値】

0　　：正常終了
1　　：円が作れない

10.4.5 与えた弧の補弧を計算

【関数名】

gmucmp

【機能】

与えた弧の補弧を計算する。

【呼出し形式】

```
void gmucmp(int ctyp, const DGEOM* crv, DGEOM* cmpcrv)
```

【入力引数】

ctyp　　弧の種類。SITARC ： 円弧、ELLIPSE ： 楕円弧。
crv　　弧

【出力引数】

cmparc　補弧

注意
補円弧の方向は与えた円弧の方向と逆になる。
円を与えたときは逆向きの円が得られる。

10.4.6 中心点と半径で円を計算

【関数名】

gmucrc

【機能】

中心点と半径 を与えて円を計算する

【呼出し形式】

```
int gmucrc(const DPOINT* pc, double rad, DARC* cir)
```

【入力引数】

pc	中心点
rad	半径

【出力引数】

cir	円

【返り値】

0	: 正常終了
1	: 円が作れない

10.4.7 与えられた点を中心とする接円を計算

【関数名】

gmuctn

【機能】

与えられた点を中心とする接円を計算する。

【呼出し形式】

```
int gmuctn(int ctyp, const DGEOM* crv, const DPOINT* pc, DARC* cir)
```

【入力引数】

ctyp	接する図形要素種類。SITPOINT, SITLINE, SITARC, SITBZCRV
crv	接する図形要素
pc	中心点

【出力引数】

cir	接円

【返り値】

接円数 (0-2)

10.4.8 ２つの図形要素に接する円を計算

【関数名】

gmuctn2g

【機能】

２つの図形要素に接する円を計算する

【呼出し形式】

```
int gmuctn2g(const int ctyp[], const DGEOM crvs[], double rad, int maxitr,
             double deps, DPOINT* pc, DARC* cir, DPOINT p[], double t[])
```

【入力引数】

ctyp	図形要素の種類 (int ctyp[2])。SITPOINT, SITLINE, SITARC, SITBZCRV
crvs	図形要素 (DGEOM crvs[2])
rad	接円の半径
maxitr	最大繰返し数
deps	許容誤差
pc	円中心点の近似値

【出力引数】

pc	円中心点
cir	円
p	二接点 (DPOINT p[2])
t	二接点のパラメータ (double t[2])

【返り値】

0	: 正常終了
1	: 円が作れない

10.4.9 ３つの図形要素に接する円を計算

【関数名】

gmuctn3g

【機能】

３つの図形要素に接する円を計算する

【呼出し形式】

```
int gmuctn3g(const short ctyp[], const DGEOM crvs[], int maxitr, double deps,
             const DPOINT* pc, const DPOINT pner[],
             DARC* cir)
```

【入力引数】

ctyp	図形要素の種類 (short ctyp[3])。SITPOINT, SITLINE, SITARC, SITBZCRV
crvs	図形要素 (DGEOM crvs[3])
maxitr	最大繰返し数
deps	許容誤差
pc	円中心点の近似値
pner	三接点の近似値 (DPOINT pner[3])

【出力引数】

pc	中心点
cir	円

【返り値】

0　　：正常終了
1　　：円が作れない

注意
複数の接円が存在する場合に、与えられた円中心点の近似値 pc 付近に中心点を持つ接円を求める。また与えられた３つの円の接円では与えられた中心点の近似値 pc 付近に中心点を持つものが複数ある場合がある。このとき、求める接円は三接点の近似値 pner 付近を通るものとする。pner は図形要素が円弧のときだけ参照する。

10.4.10 ２つの図形要素に接する接円を計算

【関数名】

gmufil

【機能】

２つの図形要素に接する接円を計算する。

【呼出し形式】

```
int gmufil(const short ctyp[], const DGEOM seg[], double rad, DARC farc[])
```

【入力引数】

ctyp	図形要素種類 (short ctyp[2])。SITPOINT, SITLINE, SITARC, SITBZCRV
seg	図形要素 (DGEOM seg[2])
rad	半径

【出力引数】

farc	接円弧

【返り値】

接円弧数

注意
接円弧の始点は接する図形要素種類の番号の小さい方の図形要素上にある。
たとえば、ctyp[0] が SITARC, ctyp[1] が SITLINE なら、接円弧の始点は２番目の図形要素 SITLINE 上にある。
接する図形要素種類が同じであれば、接円子の始点は１番目の図形要素上にある。
図形要素種類の番号順は次のとおり。
SITPOINT ＜ SITLINE ＜ SITARC ＜ SITBZCRV

10.4.11 3 点を通る円弧を計算

【関数名】

gmuctp

【機能】

３点を通る円弧を計算する

【呼出し形式】

```
int gmuctp(const DPOINT* p1, const DPOINT* p2, const DPOINT* p3, DARC* arc)
```

【入力引数】

p1, p2, p3 円弧の始点、通過点、終点

【出力引数】

arc　　円弧

【返り値】

0　　：正常終了。円弧
1　　：正常終了。円
2　　：円弧がつくれない

注意
始点と終点を等しくしたときは、円を得る。直径は p1 と p2 の距離。

10.4.12 一つの曲線を等分割した曲線群を計算

【関数名】

gmudiv

【機能】

ひとつの曲線を等分割した曲線群を計算する

【呼出し形式】

```
void gmudiv(int ctyp, const DGEOM* crv, int ncrv, DGEOM crvs[])
```

【入力引数】

ctyp　　図形要素の種類。SITLINE, -SITLINE, SITARC
crv　　分割する図形要素
ncrv　　分割数 (2-)

【出力引数】

crvs　　分割曲線群 (DGEOM crvs[ncrv])

10.4.13 ２点間距離を計算

【関数名】

gmudst

【機能】

２点間の距離を計算する

【呼出し形式】

```
double gmudst(const double p1[], const double p2[], int ndim, int ntyp)
```

【入力引数】

```
p1, p2     ２点
ndim       点の次元。2 : ２次元、3 : ３次元
ntyp       測度の種類
              1        :距離
              2        :自乗距離
              3        :最大成分の絶対値
              4        :成分の絶対値の和
```

【返り値】

距離

例

```
if (gmudst(&ps, &pe, 2, 3) < 1.0e-6)
  return ;
```

10.4.14 ２つの図形要素間での最短距離

【関数名】

gmudstcrv

【機能】

２つの図形要素間の最短距離を測る

【呼出し形式】

```
int gmudstcrv(int iswt, int typ1, const DGEOM* crv1, int typ2, const DGEOM* crv2,
              double* dst, DPOINT pout[])
```

【入力引数】

```
iswt       線分、円弧を延長するかどうか指示する。
              0        :延長する。線分は無限直線、円弧は円として処理する。
              1        :延長しない。
ctyp1,ctyp2      図形要素の種類。SITPOIN, SITLINE, SITARC, SITBZCRV
crv1, crv2       図形要素
```

【出力引数】

```
dst        最短距離。返り値が 4 のときは２直線の角度（ラジアン）
pout       最短距離点。(DPOINT pout[2])
```

【返り値】

```
0       : 正常終了
1       : 正常終了。平行または同心円
2       : エラー。交差
3       : エラー。この組み合わせは不可
4       : エラー。２直線が交差
```

10.4.15 点と図形要素間での最短距離

【関数名】

gmudstpnt

【機能】

点と図形要素間の最短距離を測る

【呼出し形式】

```
int gmudstpnt(int iswt, const DPOINT* pnt, int gtyp, const DGEOM* geom,
              double* dst, DPOINT* pout)
```

【入力引数】

iswt	線分、円弧を延長するかどうか指示する。 0 ：延長する。線分は無限直線、円弧は円として処理する。 1 ：延長しない。
pnt	点
ctyp	図形要素の種類。SITPOINT, SITLINE, SITARC, SITBZCRV
crv	図形要素

【出力引数】

dst	最短距離
pout	図形要素上の最短距離点 (DPOINT pout[2])

返り値

0 : 正常終了
1 : エラー

10.4.16 曲線のエバリュータ

【関数名】

gmueval

【機能】

曲線のエバリュータ

【呼出し形式】

```
int gmueval(int ctyp, const DGEOM* crv, int iswt, double s, int iout, DPOINT out[])
```

【入力引数】

ctyp	図形要素の種類。SITLINE, -SITLINE, SITARC, SITBZCRV
crv	図形要素
iswt	パラメータの種類。1 : 曲線にそった距離、2 : パラメータ
s	距離またはパラメータ
iout	出力スイッチ。1 : 点、2 : 点と一階微分、3 : 点、一階微分と二階微分。

【出力引数】

out　　出力

【返り値】

0　　：正常終了
1　　：エラー。

10.4.17 図形要素を包む最小矩形を求める

【関数名】

gmuextent

【機能】

図形要素を包む最小矩形を求める

【呼出し形式】

```
void gmuextent(int ctyp, const DGEOM* crv, DRECT* extent)
```

【入力引数】

ctyp　　図形要素の種類。SITLINE, -SITLINE, SITARC, SITBZCRV
crv　　図形要素 (DPFP array)

【出力引数】

extent　　図形要素を包む最小矩形の左下点と右上点

10.4.18 点が多角形の内部にあるか判定

【関数名】

gmuinout

【機能】

点が多角形の内部にあるかどうか判定する。

【呼出し形式】

```
int gmuinout(const DPOINT ply[], int num, const DRECT* box, const DPOINT* pnt)
```

【入力引数】

ply　　単純多角形の点列 (DPOINT ply[num])。終点を始点と同じにしないこと。
num　　点数 (= 辺数)
box　　多角形を包む最小矩形の左下点と右上点。
pnt　　判定する点

【返り値】

判定結果

-1　　：多角形の内部
0　　：多角形の辺上
1　　：多角形の外部

注意
多角形の点列に重複点を含めてはならない。
点数が２以下の時は多角形が構成されないので多角形判定はしない。
そのときは、点が最小矩形の内部にあるかどうかだけを判定する。

10.4.19 ２点を結ぶ線分を計算

【関数名】

gmuline

【機能】

２点を結ぶ線分を計算する

【呼出し形式】

```
int gmuline(const DPOINT* ps, const DPOINT* pe, DVLINE* lin)
```

【入力引数】

ps, pe　　２点

【出力引数】

lin　　線分

【返り値】

0　　：正常終了
1　　：線分の長さがゼロ

10.4.20 ２つの図形要素に接する線分の計算

【関数名】

gmultn

【機能】

２つの図形要素に接する線分を計算する

【呼出し形式】

```
int gmultn(const short gtyp[], const DGEOM geom[], DLINE lins[])
```

【入力引数】

ctyp　　図形要素の種類 (short ctyp[2])。SITPOINT, SITLINE, SITARC, SITBZCRV
crvs　　図形要素 (DGEOM geom[2])

【出力引数】

lins　　接線（Ps, Pe）

【返り値】

接線の数

10.4.21 円弧をオフセットする

【関数名】

gmuoffc

【機能】

円弧をオフセットする (半径変更) 。

【呼出し形式】

```
int gmuoffc(const DARC* arc, int dir, double drad, DARC* offarc)
```

【入力引数】

arc　　円弧
dir　　オフセット方向。OUTSIDE : 外側 , INSIDE　: 内側
drad　　オフセット距離

【出力引数】

offarc　　オフセットした円弧

【返り値】

0　　: 正常終了
1　　: 半径が負
2　　: エラー（半径ゼロまたは誤入力）

10.4.22 線分ををオフセットする

【関数名】

gmuoffl

【機能】

線分をオフセットする。

【呼出し形式】

```
void gmuoffl(const DVLINE* lin, int dir, double dst, DVLINE* offlin)
```

【入力引数】

lin　　線分
dir　　オフセット方向。LEFT　: 左 , RIGHT : 右。

dst　　オフセット距離

【出力引数】

offlin　オフセット線分

10.4.23 ３次 Bezier 曲線をオフセットする

【関数名】

gmuoffz

【機能】

３次 Bezier 曲線をオフセットする。

【呼出し形式】

```
int gmuoffz(const DBZC* bzc, int dir, double dst, int lswt, DBZC ofs[])
```

【入力引数】

bzc　　３次 Bezier 曲線
dir　　オフセット方向。LEFT ： 左, RIGHT ： 右。
dst　　オフセット距離
lswt　　曲線長計算。0 ： 曲線長を計算しない、1 ： 曲線長を計算する。

【出力引数】

ofs　　オフセットした３次 Bezier 曲線 (DBZC ofs[2])

【返り値】

曲線の数。入力曲線が変曲点を持つときはオフセット曲線は２つに分かれる。
0　　: オフセットできない
1-2　: OK。

10.4.24 点が図形要素上に有るか判定

【関数名】

gmuontest

【機能】

点が図形要素上にあるかどうか判定する

【呼出し形式】

```
int gmuontest(int ctyp, const DGEOM* crv, const DPOINT* pnt, int iopt)
```

【入力引数】

ctyp　　図形要素の種類。SITLINE, -SITLINE, SITARC
crv　　図形要素
pnt　　点 (DPFP array)

iopt　　オプション
　　1　　：区間判定
　　線分　　：　線分をそれと直交に移動してできる区間に入るかどうか
　　円弧　　：　円弧の中心から始点側にのばした半直線と、中心点から終点にのばした半直線が作る領域に含まれるかどうか
　　2　　：曲線判定
　　点が確実に図形要素上に乗っているか

【返り値】

判定結果。
0　　：点は図形要素上にない
1　　：点は図形要素上にある。

10.4.25 ２つの図形要素の交点を計算

【関数名】

gmupin

【機能】

ふたつの図形要素の交点を計算する

【呼出し形式】

```
int gmupin(const short ctyp[], const DGEOM crvs[], double t[][2], DPOINT pnts[])
```

【入力引数】

ctyp　　図形要素の種類 (short ctyp[2])。SITLINE, SITARC, SITBZCRV
crvs　　図形要素 (DGEOM crvs[2])

【出力引数】

t　　交点の位置のパラメータ
　　t[j]　　j-番目の交点のパラメータ
　　t[j][0]　　最初の図形要素のパラメータ
　　t[j][1]　　二番目の図形要素のパラメータ
pnts　　交点

【返り値】

交点の数

10.4.26 点を図形要素上に投影する

【関数名】

gmupon

【機能】

点を図形要素の上に投影する (投影点を計算する)

【呼出し形式】

```
int gmupon(int ctyp, const DGEOM* crv, const DPOINT* pnt, int iswt, DPOINT pans[], double t[])
```

【入力引数】

ctyp	図形要素の種類。SITLINE, SITARC, SITBZCRV
crv	図形要素
pnt	投影する点
iswt	出力スイッチ。 0 ：すべての投影点を得る 1 ：入力点に近い投影点をひとつだけ得る

【出力引数】

pans	投影点
t	投影点の位置のパラメータ

【返り値】

ICNT	投影点の数

10.4.27 二次曲線を近似する自由曲線を計算

【関数名】

gmuqtoz

【機能】

二次曲線を近似する自由曲線を計算する (三次 Bezier 曲線の並び)。

【呼出し形式】

```
int gmuqtoz(int ctyp, const DCON* con, DBZC bzc[], short iflgs[])
```

【入力引数】

ctyp	二次曲線の種類。 PARABOLA : 放物線 ELLIPSE : 楕円 HYPERBOLA : 双曲線
crv	二次曲線

【出力引数】

bzc	三次 Bezie 曲線の並び
iflgs	各三次 Bezie 曲線の表示／非表示フラグ iflgs[j] 0 : 表示 , 1 : 非表示 非表示の部分はトリムされた部分であることを表わす。 たとえば楕円で、始点の離心角 =pi/2, 終点－始点の角 =pi とすれば、つぎのようになる。 0 - pi/2 最初の 1/4 は非表示 pi/2 - 3pi/2 中央の 1/2 は表示 (× 2) 3pi/2 - 2pi 最後の 1/4 は非表示

【返り値】

三次 Bezie 曲線の数。
エラー : 0

放物線　: 1
楕円　　: 4 - 6
双曲線　: 6 - 7

例

```
#define MAXSEG 8

DBZC bzc[MAXSEG];
short iflgs[MAXSEG];
ItmSR sr;

/* 楕円 */
const DCON crv = {
  100.0, 100.0,  /* 中心点 (100,100) */
   50.0,  25.0,   /* 長軸半径 50, 短軸半径 25 */
    0.0,   1.0,    /* 長軸の向きは正 Y 軸方向 (0,1) */
    0.0, TWOPI};   /* 始点の離心角, 終点の離心角ー始点の離心角 =2pi(全周) */

/* 楕円を近似する自由曲線を計算する */
int nseg = gmuqtoz(ELLIPSE, &crv, bzc, iflgs);
if (nseg < 0)
  return;

/* 楕円を近似する自由曲線を作る */
SetItemtype(ITMFREE);
TmpgOpen1(0, 0);
SETITMSR(sr, SITSTART, 3, 2, 0, 0, &bzc.p0);
TmpgAddSr(&sr);
for (int j = 0; j < nseg; ++j) {
  SETITMSR(sr, SITBZCRV, 3, 7, iflgs[j], 0, &bzc[j].p1);
  TmpgAddSr(&sr);
}
TmpgClose(1);
```

10.4.28 ２つの単純多角形の関係を調べる

【関数名】

gmurelate

【機能】

２つの単純多角形の関係を調べる

【呼出し形式】

```
int gmurelate(const DPOINT p1[], int n1, const DRECT* box1,
              const DPOINT p2[], int n2, const DRECT* box2)
```

【入力引数】

p1, p2　　単純多角形の点列 (DPOINT p[n])
n1, n2　　単純多角形の点列の数 (=辺の数)
box1, box2単純多角形を包む最小矩形の左下点と右上点

【返り値】

判定結果

0　　：２つの多角形は離れている
1　　：２つの多角形は重なっている
2　　：多角形 p2 は　多角形 p1 を含む
3　　：多角形 p1 は　多角形 p2 を含む

注意
点数 n1 または n2 が２以下のときは多角形を構成しないので、多角形判定は行なわない。最小矩形に対してだけ判定する。
たとえば、n1=1, n2=1 ならば２つの矩形の関係だけを判定することになる。

10.4.29 図形要素を逆向きにする

【関数名】

gmurvs

【機能】

図形要素の向きを逆にする

【呼出し形式】

```
void gmurvs(int ctyp, DGEOM *crv)
```

【入力引数】

ctyp	図形要素の種類。SITLINE, (-SITLINE), SITARC, SITBZCRV
crv	図形要素

【出力引数】

crv	図形要素

10.4.30 点が図形要素のどちらにあるか判定

【関数名】

gmuside

【機能】

点が図形要素のどちらにあるか判定する

【呼出し形式】

```
int gmuside(int ctyp, const DGEOM* crv, const DPOINT* pnt)
```

【入力引数】

ctyp	図形要素の種類。SITLINE, -SITLINE, SITARC, SITBZCRV
crv	図形要素
pnt	判定する点

【返り値】

判定結果。
1　　：左
0　　：図形要素上
-1　　：右

10.4.31 図形要素を指定した位置で二つに分割

【関数名】

gmusptcrv

【機能】

図形要素を指定した位置でふたつに分割する

【呼出し形式】

```
int gmusptcrv(int ctyp, const DGEOM* crv, int iswt, const double pos[],
              DGEOM* crv1, DGEOM* crv2)
```

【入力引数】

ctyp	図形要素の種類。SITLINE, -SITLINE, SITARC, SITBZCRV
crv	図形要素
iswt	分割位置の種類。1 : パラメータ、2 : 点。
pos	分割位置。 iswt == 1 なら pos[0] にパラメータ iswt == 2 なら pos に点

【出力引数】

crv1, crv2 分割してできた２つの図形要素

【返り値】

0　　：正常終了
1　　：警告 分割点が図形要素の始点。
　　crv1　　：空
　　crv2　　:crv と同じ
2　　：警告 分割点が図形要素の終点。
　　crv1　　:crv と同じ
　　crv2　　：空
3　　：分割できない

10.4.32 円弧上の点列を取得

【関数名】

gmutolc

【機能】

円弧上の点列を得る。

【呼出し形式】

```
int gmutolc(const DARC* arc, int mode, double tol, int maxpnt, DPOINT pnts[])
```

【入力引数】

arc	円弧
mode	許容誤差の種類
	1 ：相対誤差 (0.0001 < tol < 0.25)
	2 ：絶対誤差 (モデルユニット) (0.00001 < tol/rad < 1)
tol	許容誤差 (maximum deviation)
	maxpnt 最大点数 (3-)

【出力引数】

pnts	点列 (DPOINT pnts[maxpnt])

【返り値】

点数	: (3 ～ MAXPNT)
0	: エラー

注意

点数の見積り (円に対して)

相対誤差 (0.0001 < tol < 0.25)

```
TOL : 円弧高 / 弦の長さ
tol : 0.0001        n : 7854
tol : 0.001         n :  785
tol : 0.01          n :   79
tol : 0.02          n :   39
tol : 0.05          n :   16
tol : 0.1           n :    8
```

絶対誤差 (モデルユニット) (0.00001 < tol/rad < 1)

```
TOL : 円弧高
tol/rad : 0.00001   n :  702
tol/rad : 0.0001    n :  222
tol/rad : 0.001     n :   70
tol/rad : 0.01      n :   22
tol/rad : 0.02      n :   16
tol/rad : 0.05      n :   10
tol/rad : 0.1       n :    7
```

10.4.33 ３次 Bezier 曲線上の点列を取得

【関数名】

gmutolz

【機能】

３次 Bezier 曲線上の点列を得る。

【呼出し形式】

```
int gmutolz(const DBZC* bzc, int mode, double tol, int maxpnt, DPOINT pnts[])
```

【入力引数】

bzc　　３次 Bezier 曲線
mode　　許容誤差の種類
　　1　　：相対誤差 (0.0 < tol < 0.25)
　　2　　：絶対誤差（モデルユニット）
tol　　許容誤差 (maximum deviation)
maxpnt　　最大点数

【出力引数】

pnts　　点列 (DPOINT pnts[maxpnt])

【返り値】

点数

10.4.34 線分を矩形領域でクリップ

【関数名】

gmuclip1

【機能】

線分を矩形領域でクリップする。

【呼出し形式】

```
int gmuclip1(const DRECT* rect, const DLINE* lin, DLINE* lout)
```

【入力引数】

rect　　矩形領域（左下点 と右上点）
lin　　クリップする線分

【出力引数】

lout　　クリップされた線分。lin とおなじ変数でもよい。

【返り値】

-1　　：線分は矩形領域の外にある。
0　　：線分は矩形領域の内にある。
1 -　　：線分は矩形領域でクリップされた。

例

三つの線分を領域 (-10,-10), (10, 10) でクリップする。
(-5, 15), (0,20)　領域の外
(-5, 0), (0, 5)　領域の内
(-5,-15), (15, 5)　クリップされた (0,-10),(10,0)

```
const DRECT rect  = {-10, -10, 10, 10};
const DLINE lin[] = {{-5,  15,  0, 20}, {-5, 0, 0, 5}, {-5, -15, 15, 5}};
DLINE out;
for (int j = 0; j < sizeof(lin) / sizeof(lin[0]); ++j) {
  if (gmuclip1(&rect, &line[j], &out) >= 1)
    printf("clipped line %g %g %g %g¥n",
           out.ps.x, out.ps.y, out.pe.x, out.pe.y);
```

10.4.35 線分を凸多角形領域でクリップ

【関数名】

gmuclip2

【機能】

線分を凸多角形領域でクリップする。

【呼出し形式】

```
int gmuclip2(int nvtx, const DPOINT cbry[], const DPOINT vnrm[],
             const DRECT* cext, int iswt, const DLINE* lorg,
             int* key, DLINE* lout)
```

【入力引数】

nvtx	凸多角形の頂点の数 (= 辺の数)
cbry	凸多角形の頂点の座標の並び
vnrm	凸多角形の辺の法線ベクトルの並び。 vnrm[j] は 辺 cbry[j]->cbry[j+1] の法線ベクトル。 j = nvtx - 1 のとき辺は cbry[nvtx - 1] -> cbry[0]。 (j = 0 から nvtx - 1)
cext	凸多角形を含む最小矩形
iswt	0
lorg	クリップする線分

【出力引数】

key	結果 -1 : 線分は矩形領域の外にある。 0 : 線分は矩形領域の内にある。 1 : 線分は矩形領域でクリップされた。
lout	クリップされた線分。lorg とおなじ変数でもよい。

【返り値】

lout	クリップされた線分の数 (0、1)

10.4.36 閉多角形を多角形領域でクリップ

【関数名】

gmuclip3

【機能】

閉多角形を多角形領域でクリップする。

【呼出し形式】

```
int gmuclip3(int nw, const DPOINT pw[], int np, DPOINT pp[], int maxpnt, DPOINT pq[])
```

【入力引数】

nw	クリップウインドウ多角形の頂点の数（= 辺の数）
pw	クリップウインドウ多角形の頂点の座標の並び。 多角形の頂点列は反時計回りであること。
np	クリップしたい閉多角形の頂点の数（ = 辺の数）
pp	クリップしたい閉多角形の頂点の座標の並び。 多角形の頂点列は反時計回りであること。
maxpnt	配列 pp, pq の長さ。 クリップされた閉多角形の頂点数は、もとの多角形の頂点数より多くなることもある。たとえば、正方形のウインドウで菱形をクリップした結果は八角形となる。配列 pp, pq の大きさはこれを考慮すること。
pq	中間結果の格納に使用する。

【出力引数】

pp	クリップされた閉多角形の頂点の座標の並び

【返り値】

クリップされた閉多角形の頂点の数または辺の数
クリップしたい閉多角形が、クリップウインドウ多角形と交差していないときは 0。

10.5 ベクトルと座標交換

● 関数一覧

xinvt	**座標逆変換**
xquadeq	**２次方程式を解く**
xtrfm	**座標変換**
xuvsub	**ベクトルの差**
xvadd	**ベクトルの和**
xvcr	**ベクトルの外積**
xvdot	**ベクトルの内積**
xvprj	**ベクトル U をベクトル V に投影する**
xvscpadd	**Q = P + s*V (２次元ベクトルのみ)**
xvsub	**ベクトルの差**
xvunit	**ベクトルを正規化する (ベクトルの大きさを１にする)**
xmir	**軸対称点を計算する。**

10.5.1 座標逆変換

【関数名】

xinvt

【機能】

座標逆変換

【呼出し形式】

```
void xinvt(const DPOINT* xdir, const DPOINT* pnt, const DPOINT* porg, DPOINT* pout)
```

【入力引数】

xdir	座標系 X 軸方向余弦。xdir->x = cos(s), xdir->y = sin(s)
pnt	変換する座標
prog	座標系の原点座標

【出力引数】

pout	変換された座標

注意

ローテーションマトリックス

$$M = \begin{bmatrix} L1 & L2 \\ M1 & M2 \end{bmatrix} = \begin{bmatrix} xdir \rightarrow x & -xdir \rightarrow y \\ xdir \rightarrow y & xdir \rightarrow x \end{bmatrix}$$

```
pout = transpose-Matrix M * (pnt - porg)
```

10.5.2 ２次方程式を解く

【関数名】

xquadeq

【機能】

２次方程式を解く

【呼出し形式】

```
int xquadeq(const double c[], double t[])
```

【入力引数】

c	係数の並び f(t) = a*t^2 + b*t + c = 0 c[0] = a, c[1] = b and c[2] = c

【出力引数】

t	根

【返り値】

根の数 (0-2)

10.5.3 座標変換

【関数名】

xtrfm

【機能】

座標変換

【呼出し形式】

```
void xtrfm(const DPOINT* xdir, const DPOINT* pnt, const DPOINT* porg, DPOINT* pout)
```

【入力引数】

xdir	座標系 X 軸方向余弦。xdir->x = cos(s), xdir->y = sin(s)
pnt	変換する座標
porg	座標系の原点座標

【出力引数】

pout	変換された座標

注意

ローテーションマトリックス

$$M = \begin{bmatrix} L1\ L2 \\ M1\ M2 \end{bmatrix} = \begin{bmatrix} xdir \rightarrow x & -xdir \rightarrow y \\ xdir \rightarrow y & xdir \rightarrow x \end{bmatrix}$$

pout = M * pnt + porg

10.5.4 ベクトルの差 UV = unitize(PE - PS)

【関数名】

xuvsub

【機能】

ベクトルの差　UV = unitize(PE-PS)

【呼出し形式】

double xuvsub(const double ps[], const double pe[], double uv[], int ndim)

【入力引数】

ps, pe　　ベクトル
ndim　　ベクトルの次数。2 : ２次, 3 : ３次

【出力引数】

uv　　正規化したベクトル

【返り値】

ベクトルの大きさ (長さ)

10.5.5 ベクトルの和

【関数名】

xvadd

【機能】

ベクトルの和　W = U + V

【呼出し形式】

void xvadd(const double u[], const double v[], double w[], int ndim)

【入力引数】

u, v　　ベクトル
ndim　　ベクトルの次数。2 : ２次, 3 : ３次

【出力引数】

w　　ベクトル

10.5.6 ベクトルの外積

【関数名】

xvcr

【機能】

ベクトルの外積

【呼出し形式】

```
void xvcr(const DPOINT3* a, const DPOINT3* b, DPOINT3* c)
```

【入力引数】

a, b　　３次ベクトル

【出力引数】

c　　３次ベクトル

10.5.7 ベクトルの内積

【関数名】

xvdot

【機能】

ベクトルの内積

【呼出し形式】

```
double xvdot(const double u[], const double v[], int ndim)
```

【入力引数】

u, v　　ベクトル
ndim　　ベクトルの次数。
　　2　　：２次
　　3　　：３次

【返り値】

ベクトルの内積

10.5.8 ベクトル A をベクトル U に投影

【関数名】

xvprj

【機能】

ベクトル A をベクトル U に投影する

【呼出し形式】

```
void xvprj(const double a[], const double u[], double p[], int ndim)
```

【入力引数】

a	投影するベクトル
u	基準ベクトル（正規化）
ndim	ベクトルの次数。2 ： ２次，3 ： ３次

【出力引数】

p	投影されたベクトル

10.5.9 Q=P+s*V

【関数名】

xvscpadd

【機能】

Q = P + s*V (２次元ベクトルのみ)

【呼出し形式】

```
void xvscpadd(const DPOINT* p, const DPOINT* v, double s, int dir, DPOINT* q)
```

【入力引数】

p	点	
v	ベクトル	
s	ベクトルの倍率	
dir	方向。	
	ASIS	： そのまま
	LEFT	： 左（ベクトル v を左 90 度回転）
	RIGHT	： 右（ベクトル v を右 90 度回転）

【出力引数】

q	点

10.5.10 ベクトルの差　　V = PE - PS

【関数名】

xvsub

【機能】

ベクトルの差　V = PE - PS

【呼出し形式】

```
void xvsub(const double ps[], const double pe[], double w[], int ndim)
```

【入力引数】

pe, ps	ベクトル
ndim	ベクトルの次数。2 : ２次, 3 : ３次

【出力引数】

v	ベクトル

10.5.11 ベクトルを正規化

【関数名】

xvunit

【機能】

ベクトルを正規化する (ベクトルの大きさを１にする)

【呼出し形式】

```
double xvunit(const double v[], double u[], int ndim)
```

【入力引数】

v	ベクトル
ndim	ベクトルの次数。2 : ２次, 3 : ３次

【出力引数】

u	正規化したベクトル

【返り値】

入力ベクトルの大きさ (長さ)

10.5.12 軸対称点を計算

【関数名】

xmir

【機能】

軸対称点を計算する。

【呼出し形式】

```
void xmir(const DPOINT* umir, const DPOINT* pnt, const DPOINT* pmir, DPOINT* pout)
```

【入力引数】

umir	対称軸の単位ベクトル
pnt	軸対称する点
pmir	対称軸の位置

【出力引数】

pout	軸対称点

第 11 章　図形以外のモデル情報

11.1 ピクチャ回転マトリックス

● 関数一覧

isopmtget	**ピクチャ回転マトリックステーブルに設定されているピクチャ回転マトリックスを得る。**
isopmtset	**ピクチャ回転マトリックステーブルに設定されているピクチャ回転マトリックスを変更する。**
isoprjget	**マトリクスの投影タイプを得る。**
isoprjset	**マトリクスの投影タイプを変更する。**
isomtxstd	**標準ピクチャ回転マトリックスを得る。**
isomtxrot	**回転基準軸に対する回転角度からピクチャ回転マトリックスを得る。**
isomtxmlt	**２つのピクチャ回転マトリックスの積を得る。**

11.1.1 ピクチャ回転マトリックスを取得

【関数名】

isopmtget

【機能】

ピクチャ回転マトリックステーブルに設定されているピクチャ回転マトリックスを得る。

【呼出し形式】

```
int isopmtget(int imtx, double pmtx[][3])
```

【入力引数】

imtx　　ピクチャ回転マトリックス番号 (= ピクチャ番号)

【出力引数】

pmtx　　ピクチャ回転マトリックス (double pmtx[3][3])

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} PMTX(1,1) & PMTX(2,1) & PMTX(3,1) \\ PMTX(1,2) & PMTX(2,2) & PMTX(3,2) \\ PMTX(1,3) & PMTX(2,3) & PMTX(3,3) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(x,y,z) ： ３次元空間座標
(X,Y,Z) ： ピクチャ座標

【返り値】

0　　： 正常終了

11.1.2 ピクチャ回転マトリックスを変更

【関数名】

isopmtset

【機能】

ピクチャ回転マトリックステーブルに設定されているピクチャ回転マトリックスを変更する。

【呼出し形式】

```
int isopmtset(int imtx, const double pmtx[][3], int idsp)
```

【入力引数】

imtx　　ピクチャ回転マトリックス番号 (= ピクチャ番号)
pmtx　　ピクチャ回転マトリックス (double pmtx[3][3])
idsp　　0 を設定する。

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} PMTX(1,1) & PMTX(2,1) & PMTX(3,1) \\ PMTX(1,2) & PMTX(2,2) & PMTX(3,2) \\ PMTX(1,3) & PMTX(2,3) & PMTX(3,3) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(x, y, z) ： ３次元空間座標
(X, Y, Z) ： ピクチャ座標

【返り値】

0　　：正常終了

11.1.3 マトリックスの投影タイプを取得

【関数名】

isoprjget

【機能】

マトリックスの投影タイプを得る。

【呼出し形式】

```
int isoprjget(void)
```

【返り値】

投影タイプ。投影タイプについては isomtxstd 関数を参照のこと。

11.1.4 マトリックスの投影タイプを変更

【関数名】

isoprjset

【機能】

マトリックスの投影タイプを変更する。

【呼出し形式】

```
int isoprjset(int prjtyp)
```

【入力引数】

prjtyp　　投影タイプ

【返り値】

0　　：正常終了

注意

このモジュールを呼出すと、ピクチャ１から７のローテーションマトリクスは各タイプの標準マトリクスに設定される。

投影タイプについては isomtxstd 関数を参照のこと。

11.1.5 標準ピクチャ回転マトリックスを取得

【関数名】

isomtxstd

【機能】

標準ピクチャ回転マトリックスを得る。

【呼出し形式】

```
void isomtxstd(int prjtyp, int mtxtyp, double pmtx[][3])
```

【入力引数】

prjtyp　マトリックス投影タイプ
- 0　：投影タイプ 0
- 1　：投影タイプ 1

mtxtyp　ピクチャ回転マトリックスタイプ

投影タイプが 0 のとき
- 1　：平面　(no rotation)
- 2　：正面　(rotate X-90)
- 3　：右側面　(rotate Y-90)
- 4　：底面　(rotate X180)
- 5　：背面　(rotate X90)
- 6　：左側面　(rotate Y90)
- 7　：アイソメトリック (rotate Z45, X-54.74)

投影タイプが 1 のとき
- 1　：平面　(no rotation)
- 2　：正面　(rotate X-90)
- 3　：右側面　(rotate X-90, Y-90)
- 4　：底面　(rotate X180)
- 5　：背面　(rotate X-90, Y180)
- 6　：左側面　(rotate X-90, Y90)
- 7　：アイソメトリック (rotate Z45, X-54.74)

【出力引数】

pmtx　ピクチャ回転マトリックス (double pmtx[3][3])

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} PMTX(1,1) & PMTX(2,1) & PMTX(3,1) \\ PMTX(1,2) & PMTX(2,2) & PMTX(3,2) \\ PMTX(1,3) & PMTX(2,3) & PMTX(3,3) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(x,y,z) ： ３次元空間座標
(X,Y,Z) ： ピクチャ座標

11.1.6 開店後のピクチャ回転マトリックスを取得

【関数名】

isomtxrot

【機能】

回転基準軸に対する回転角度からピクチャ回転マトリックスを得る。

【呼出し形式】

```
void isomtxrot(int axis, double ang, double pmtx[][3])
```

【入力引数】

axis　回転基準軸。
- 1　：X 軸
- 2　：Y 軸
- 3　：Z 軸

ang　回転角度（単位：度）

【出力引数】

pmtx　ピクチャ回転マトリックス (double pmtx[3][3])

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} PMTX(1,1) & PMTX(2,1) & PMTX(3,1) \\ PMTX(1,2) & PMTX(2,2) & PMTX(3,2) \\ PMTX(1,3) & PMTX(2,3) & PMTX(3,3) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(x,y,z) ： ３次元空間座標
(X,Y,Z) ： ピクチャ座標

X 軸に対するピクチャ回転マトリックス

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & COS(ANG) & -SIN(ANG) \\ 0 & SIN(ANG) & COS(ANG) \end{bmatrix}$$

Y 軸に対するピクチャ回転マトリックス

$$\begin{bmatrix} COS(ANG) & 0 & SIN(ANG) \\ 0 & 1 & 0 \\ -SIN(ANG) & 0 & COS(ANG) \end{bmatrix}$$

Z 軸に対するピクチャ回転マトリックス

$$\begin{bmatrix} COS(ANG) & -SIN(ANG) & 0 \\ SIN(ANG) & COS(ANG) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

11.1.7　２つのピクチャ回転マトリックスの積を取得

【関数名】

isomtxmlt

【機能】

２つのピクチャ回転マトリックスの積を得る。

【呼出し形式】

```
void isomtxmlt(const double pmtx1[][3], const double pmtx2[][3], double pmtx3[][3])
```

【入力引数】

pmtx1　　ピクチャ回転マトリックス (double pmtx1[3][3])
pmtx2　　ピクチャ回転マトリックス (double pmtx2[3][3])

【出力引数】

pmtx3　　ピクチャ回転マトリックスの積 (double pmtx3[3][3])
pmtx3 = pmtx2 * pmtx1

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} PMTXi(1,1) & PMTXi(2,1) & PMTXi(3,1) \\ PMTXi(1,2) & PMTXi(2,2) & PMTXi(3,2) \\ PMTXi(1,3) & PMTXi(2,3) & PMTXi(3,3) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$i = 1,2,3$$

(x,y,z) : ３次元空間座標
(X,Y,Z) : ピクチャ座標

注意
ピクチャ回転マトリックスの積 PMTX3 は、ピクチャ回転マトリックス PMTX1 を更にピクチャ回転マトリックス PMTX2 で回転させたピクチャ回転マトリックスである。

11.2 モデルタイトル

● 関数一覧

TtItxtput	**モデルタイトルを設定する**
TtItxtget	**モデルタイトルを取り出す**

11.2.1 モデルタイトルを設定

【関数名】

TtItxtput

【機能】

モデルタイトルを設定する

【呼出し形式】

int TtItxtput(int ttlno, const char* text)

【入力引数】

ttlno　　設定するモデルタイトル番号 (1 - 201)
text　　設定するモデルタイトル (NULL-Terminated)
設定できる文字数は最大 256 バイトまで。

【返り値】

0　　: 正常
1　　: 指定されたモデルタイトル番号は存在しない。
2　　: モデルタイトルの文字数が短かすぎる。
3　　: モデルタイトルの文字数が長すぎる。
4　　: モデルタイトルの総長が長すぎる。

5　　：モデルタイトルのデータの型が定義にあわない。

注意
モデルタイトル 202 のモデル名にはこの関数では設定できない。モデル名を設定するには Mdlname001() 関数を使用する。

11.2.2 モデルタイトルを取得

【関数名】

Ttltxtget

【機能】

モデルタイトルを取り出す

【呼出し形式】

```
int Ttltxtget(int ttlno, char *text)
```

【入力引数】

ttlno	取り出すモデルタイトル番号 (1 - 255)

【出力引数】

text	取り出されたモデルタイトル (NULL-Terminated)。 text の大きさは、257 バイト必要。

【返り値】

モデルタイトルの文字数。
-1　　：指定されたモデルタイトル番号は存在しない。

11.3 特性データ

● 関数一覧

Spc005	**アイテムに付加された特性データを取り出す。**
Spc101	**カレントの特性データのファイル番号とレコード番号を変更する。**
Spc102	**カレントの特性データにデータをセットする。**
Spc103	**カレントの特性データをアイテムに追加する。**
Spc104	**カレントの特性データをアイテムから削除する。**

11.3.1 アイテムに付加された特性データを取得

【関数名】

Spc005

【機能】

アイテムに付加された特性データを取り出す

【呼出し形式】

```
int Spc005(int idptr, int rec, int num, double *dval, int *c_dval,  char *cval,
           size_t l_cval)
```

【入力引数】

idptr	アイテム識別番号
rec	特性データのレコード番号
num	特性データの項目番号
c_dval	dval の配列の個数
l_cval	cval の大きさ（バイト単位）

【出力引数】

dval	付加された特性データ（数値の場合）
c_dval	dval に格納した特性データの個数（数値の場合）
cval	付加された特性データ（文字列の場合。NULL terminated）

【返り値】

0	: 文字列。
1	: 16 ビット整数。
2	: 単精度実数。
3	: 倍精度実数。
-1	: アイテム識別子が正しくない。
-2	: アイテムには特性データが付加されていない。
-3	: 指定されたレコード番号の特性データが付加されていない。
-4	: 指定された項目番号の特性データが付加されていない。
-5	: 特性データのレコード番号が正しくない。
-6	: 特性データの項目番号が正しくない。

注意

取り出される特性データが数値（16 ビット整数、単精度実数または倍精度実数）の場合、数値は倍精度実数に変換されて、配列 dval に格納される。c_dval には用意した配列 dval の個数を指定する。c_dval には取り出した特性データの個数が設定される。取り出される特性データが文字列の場合、cval に格納される。取り出された文字列は NULL terminated となるので、少なくとも、取り出される文字列の長さに 1 を足した領域を、cval として用意する必要がある。l_cval には、用意した領域の大きさをバイト単位で指定する。

11.3.2 カレント特性データのファイル番号とレコード番号を変更

【関数名】

Spc101

【機能】

カレントの特性データのファイル番号とレコード番号を変更する。

【呼出し形式】

```
int Spc101(int fleno, int recno, int keyint)
```

【入力引数】

fleno　特性データファイル番号
recno　特性データレコード番号
keyint　特性データ初期化スイッチ
　　0　　：カレントの特性データを初期化しない。
　　1　　：カレントの特性データを初期化する。

【返り値】

0　　：正常終了

11.3.3 カレント特性データにデータを設定

【関数名】

Spc102

【機能】

カレントの特性データにデータをセットする。

【呼出し形式】

```
int Spc102(int kno, int type, int cnt, const char* text, const double slst[])
```

【入力引数】

kno　　特性データの項目番号
type　　入力データのタイプ。
　　0　　：文字列でデータをセットする
　　1　　：数値をセットする。
cnt　　入力データの数。
　　type == 0 のとき文字列の長さ、type == 1 のとき数値の数。
text　　文字列
slst　　数値の並び

【返り値】

0　　：正常終了

11.3.4 カレント特性データをアイテムに追加

【関数名】

Spc103

【機能】

カレントの特性データをアイテムに追加する。

【呼出し形式】

```
int Spc103(int idptr)
```

【入力引数】

idptr　　アイテムの識別番号

【返り値】

0　　：正常終了

11.3.5 カレント特性データをアイテムから削除

【関数名】

Spc104

【機能】

カレントの特性データをアイテムから削除する。

【呼出し形式】

```
int Spc104(int idptr)
```

【入力引数】

idptr　　アイテム識別番号

【返り値】

0　　：正常終了
-1　　：アイテム識別番号が正しくない
-2　　：アイテムには特性データが付加されていない。

11.4 ステータス

● 関数一覧

Radius001	**半径値を設定する。**
Radius101	**半径値を取り出す。**
Atr001	**コマンド別のアイテム属性テーブルの設定。**
Atr101	**コマンド別のアイテム属性テーブルの内容を抽出する。**
Lst001	**各ピクチャのアイテム数を得る**
Lst002	**アイテム属性（クラス、レビジョン、線種、線幅）およびアイテム種別の使用状況を得る**
Lst003	**スクリーンレイアウト情報を得る**

11.4.1 半径値を設定

【関数名】

Radius001

【機能】

半径値を設定する。

【呼出し形式】

int Radius001(double rad, int dspflg)

【入力引数】

rad　　設定する半径値
dspfg　　ステータス領域に半径値を
　　0　　：表示しない
　　1　　：表示する

【返り値】

0　　：正常
1　　：rad が不正

11.4.2 半径値を設定

【関数名】

Radius101

【機能】

半径値を取り出す。

【呼出し形式】

double Radius101(void)

【入力引数】

なし

【出力引数】

なし

【返り値】

半径値

11.4.3 アイテム属性テーブルの設定

【関数名】

Atr001

【機能】

コマンド別のアイテム属性テーブルの設定

【呼出し形式】

int Atr001(int iflg, int type, int val)

【入力引数】

iflg	分類。	
	1	：線種
	2	：線幅
	3	：クラス
	4	：レビジョン。
type	コマンド分類	
	1	：幾何図形作成コマンド
	2	：寸法作成コマンド
	3	：注記作成コマンド
	4	：記号作成コマンド
	5	：風船作成コマンド
	6	：仕上記号作成コマンド
	7	：切断線作成コマンド
	8	：円中心線作成コマンド
	9	：ハッチング作成コマンド
val	テーブルの値。意味については『コマンドリファレンス』の線種線幅定数を参照のこと。	

【返り値】

0 ： 正常

11.4.4 アイテム属性テーブルの内容を抽出

【関数名】

Atr101

【機能】

コマンド別のアイテム属性テーブルの内容を抽出する。

【呼出し形式】

```
int Atr101(int iflg, int type)
```

【入力引数】

iflg	分類。	
	1	：線種
	2	：線幅
	3	：クラス
	4	：レビジョン。
type	マンド分類	
	1	：幾何図形作成コマンド
	2	：寸法作成コマンド
	3	：注記作成コマンド
	4	：記号作成コマンド
	5	：風船作成コマンド
	6	：仕上記号作成コマンド
	7	：切断線作成コマンド
	8	：円中心線作成コマンド
	9	： ハッチング作成コマンド

【返り値】

テーブルの値。『コマンドリファレンス』の線種線幅定数を参照のこと。

11.4.5 各ピクチャのアイテム数を取得

【関数名】

Lst001

【機能】

各ピクチャのアイテム数を得る

【呼出し形式】

```
int Lst001(int itms[], bool dlopic = false)
```

【入力引数】

dlopic　false を設定する。この引数を省略すると flase。

【出力引数】

itms　各ピクチャのアイテム数（int itms[MAXITMPIC]）

【返り値】

全ピクチャのアイテム数

11.4.6 アイテム属性・アイテム種別の使用状況を取得

【関数名】

Lst002

【機能】

アイテム属性 (クラス、レビジョン、線種、線幅) およびアイテム種別の使用状況を得る

【呼出し形式】

```
int Lst002(int iflg, int ipic, int itms[])
```

【入力引数】

iflg　使用状況を得る属性の種類
- 1　：アイテム種別（点、線、円弧...）の使用状況を得る
- 2　：クラスの使用状況を得る
- 3　：レビジョンの使用状況を得る
- 4　：線種の使用状況を得る
- 5　：線幅の使用状況を得る

ipic　使用状況を調べるピクチャのピクチャ番号
- -1　：全ピクチャ
- 0　：アクティブピクチャ
- 1 - MAXITMPIC　：ピクチャ番号

【出力引数】

itms　　アイテム数の並び
　　int itms[N]
　　iflg == 1 の時、N は MAXITMTYP
　　iflg == 2 の時、N は MAXITMCLS
　　iflg == 3 の時、N は MAXITMREV
　　iflg == 4 の時、N は MAXITMLFT
　　iflg == 5 の時、N は MAXITMLWT

【返り値】

0　　：正常

例

```
/* 全ピクチャのクラスの使用状況を得る */
int itms[MAXITMCLS];
Lst002(2, -1, itms);

/* クラス１のアイテム数が５、クラス３のアイテム数が７で、
   他のクラスにアイテムが存在しない時は次の値が戻る。
itms[0] : 5
itms[1] : 0
itms[2] : 7
itms[3] ～ itms[MAXITMCLS-1] : 0
*/
```

11.4.7 スクリーンレイアウト情報の取得

【関数名】

Lst003

【機能】

スクリーンレイアウト情報を得る

【呼出し形式】

```
int Lst003(int islo, int tbl[][3], double zon[][4])
```

【入力引数】

islo　　情報を抽出するスクリーンレイアウト番号
　　-1　　：全スクリーンレイアウト
　　0　　：アクティブスクリーンレイアウト
　　1 - 31　　：スクリーンレイアウト番号

【出力引数】

tbl　　スクリーンレイアウト情報（int tbl[128][3]）
　　スクリーンレイアウト番号、ビューポート番号の小さい順に並べられる。
　　tbl[][0] : スクリーンレイアウト番号
　　tbl[][1] : ビューポート番号
　　tbl[][2] : ピクチャ番号
zon　　表示範囲テーブル（double zon[128][4]）
　　zon[][0] : 表示範囲 最小 X 座標（モデル座標）
　　zon[][1] : 表示範囲 最小 Y 座標（モデル座標）

zon[][2] : 表示範囲 最大 X 座標（モデル座標）
zon[][3] : 表示範囲 最大 Y 座標（モデル座標）

【返り値】

ビューポート数 (1 - 128)。エラーの場合は 0。

例

```
/* アクティブスクリーンレイアウトの状態を得る */
int tbl[128][3];
double zon[128][4];
Lst003(0, tbl, zon);
```

第 12 章　ユーティリティ関数

12.1 ユーティリティ関数

● 関数一覧

macroload	**マクロファイルをコンパイル、ロード、実行する。**
mcrcalc	**計算器**
cvstof	**文字列を数値の並びに変換する。**
detab	**文字列の水平タブを適切な数の空白文字でおきかえる。**
FilName	**フルパスのファイル名を求める。**
FilSelect	**ファイル名を画面に表示し、該当するものを選ぶ。**
gcdate	**現日時を得る。**
ifld	**ビット列の取り出し**
insfld	**ビット列の挿入**
ibyte	**バイトの取り出し**
insbyte	**バイトの挿入**
xsrt	**内部ソート**
xswap	**二次元配列の列を交換する。**
sjis2euc	**MS 漢字コード文字列を日本語 EUC 文字列に変換する。**
euc2sjis	**日本語 EUC 文字列を MS 漢字コード文字列に変換する。**

12.1.1 マクロファイルをコンパイル・ロード・実行する

【関数名】

macroload

【機能】

マクロファイルをコンパイル、ロード、実行する。

【呼出し形式】

```
int macroload(int swt, const char* name)
```

【入力引数】

swt　　再ロードするかどうかのスイッチ (0, -2)
　　0　　：指定されたマクロが既にロードされていれば再ロードしない。実行のみする。

-2　：指定されたマクロ既にロードされているかいないに関わらず再度コンパイル、ロードし実行する。

name　マクロ名。(Null-char terminated)
ディレクトリ名とファイル拡張子は付けない。ACAD.SET の #MACRO# で指定されているディレクトリ名とファイル拡張子を、この関数自身で付加している。

【返り値】

0　：正常。
1　：コンパイルエラー。
2　：エラー（マクロが存在しない。リスタート中。）

注意
この関数は指定されたマクロソースをコンパイルし、エラーがなければロードし、マクロの実行フラグをオンにする。実行はこの関数を呼び出したユーザプログラムが処理を終了して Advance CAD の入力処理に戻ったときにロードされたマクロを実行する。
マクロのコンパイル結果は Advance CAD の起動ディレクトリの macro.lis に出力される。

12.1.2 計算器

【関数名】

mcrcalc

【機能】

計算器。この関数は Advance CAD の計算器で、キーボードから計算式を '[' と ']' でくくって入力すると呼び出される。
ユーザプログラムでこの関数を使用するのは主に以下の目的である。

- 計算器の変数に値を割り付ける。
- 計算器の変数の値を参照する。

計算器の変数はマクロの外部変数となる。マクロの内部変数にはアクセスできない。

【呼出し形式】

```
int mcrcalc(const char* expr, double* dpfp, char* cstr, size_t cstr_len)
```

【入力引数】

expr　計算する式 (Null-char terminated)
(1)　'[' と ']' は不要
(2)　計算器の変数名は英大文字と数字のみ
(3)　数値定数
整数型、実数型いずれでもよいが、すべて実数型に変換してから計算する。
(4)　文字定数はクォーテーション (") で囲む
cstr_len　配列 cstr の長さ。

【出力引数】

dpfp　計算結果が数値のとき値が設定される。
cstr　計算結果が文字列のときにそれを格納する配列。(NULL-Terminated)

【返り値】

計算結果のデータの型またはエラーコード
0　：なし　（たとえば clear() の結果）
3　：数値　（たとえば a=2*pi の結果）

4　　：文字列　（たとえば a="ABC" の結果）
30　　：配列　　（たとえば a=array(3) の結果）
　　dpfp は配列のサイズ。

-201　　：式が誤り
-205　　：式に未定義のレジスタがある
-208　　：式に未定義の変数がある
-212　　：式に未定義の組み込み関数がある

例

```
char *s;
int type;
char cstr[16];
double dpval;

/* 計算器の変数 A に数値 10 を割り付ける。*/
s = "a=10";
mcrcalc(s, strlen(s), &dpval, cstr, sizeof(cstr));

/* 計算器の変数 B に文字列 'ABC' を割り付ける。*/
s = "a=¥"ABC¥"";
mcrcalc(s, strlen(s), &dpval, cstr, sizeof(cstr));

/* 計算器の変数 A の値を得る。*/
s = "a";
type = mcrcalc(s, strlen(s), &dpval, cstr, sizeof(cstr));
switch (type) {
  case 0: /* No output */
    break;
  case 3: /* Number */
    break;
  case 4: /* Text */
    break;
  case 30: /* Array */
    break;
  default:
    break;
}
```

12.1.3 文字列を数値の並びに変換

【関数名】

cvstof

【機能】

文字列を数値の並びに変換する。

【呼出し形式】

```
int cvstof(const char* src, int maxcnt, double dpfp[])
```

【入力引数】

src　　入力文字列 (NULL-terminated)
maxcnt　　最大変換数（配列 dpfp の長さ）。

【出力引数】

dpfp　　数値を格納する配列 (double dpfp[maxcnt])

【返り値】

数値の数 (0 - maxcnt)

【入力文字列】

(1)　数値の区切りはカンマ','
(2)　数値の区切りカンマ',' の間に何も文字がないときや、空白だけのときは数値 0 とする。また入力文字列の最初にカンマが表われたとき、最後にカンマが表われた時も同様。
(3)　一つの数値の文字列長さは 20 文字以内

例

```
'1, 2.0 ,3 '     ->      1.0, 2.0, 3.0
', 2, ,4,'       ->      0.0, 2.0, 0.0, 4.0, 0.0
'1e2,,3d-1'      ->    100.0, 0.0, 0.3
```

12.1.4 文字列の水平タブを適切な数の空白文字に変換

【関数名】

detab

【機能】

文字列の水平タブ (\t) を適切な数の空白文字でおきかえる。

【呼出し形式】

```
int detab(int tabsiz, const char* src, char* dst, size_t dst_len)
```

【入力引数】

tabsiz　　水平タブサイズ
　　tabsiz ＞ 0　: 水平タブサイズ
　　tabsiz = 0　: 水平タブサイズ = 8
　　tabsiz ＜ 0　: 水平タブサイズ = 絶対値 (tabsiz)
　　日本語の空白文字も、二つの ASCII 空白文字におきかえる。
src　　入力文字列 (NULL-terminate)
dst_len　　配列 dst の長さ。

【出力引数】

dst　　出力文字列 (NULL-terminate)

12.1.5 フルパスのファイル名を取得

【関数名】

Filname

【機能】

ファイル名に、ディレクトリ定義ファイル ACAD.SET からディレクトリとファイル拡張子を取り出し、フルパスの名前を求める。

【呼出し形式】

```
int Filname(const char* fctg, char* fname, size_t fname_len, int key)
```

【入力引数】

fctg	ファイルの種類 (NULL-terminated)。たとえば "#MODEL#", "#SYMBOL#"。
fname	ファイル名 (NULL-terminated)
fname_len	配列 fname の長さ。
key	0。

【出力引数】

fname	ディレクトリとファイル拡張子を付けたファイル名 (NULL-terminated)

【返り値】

0	:	正常
1	:	指定のファイルの種類は登録されていない。
2	:	ファイル名が長いため fname に収まらない。

注意

(1) 入力のファイル名にディレクトリ名がふくまれていれば、ディレクトリ名はそのままである。ファイル拡張子も同様。

入力	結果
"ABC"	"/usr/acad/files/ABC.SYM"
"ABC.TST"	"/usr/acad/files/ABC.TST"
"/tmp/ABC"	"/tmp/ABC.SYM"
"/tmp/ABC.TST"	"/tmp/ABC.TST"

(2) UNIX でのディレクトリ

~/	ホームディレクトリ
./	カレントディレクトリ
~xxx/	ユーザ xxx のホームディレクトリ

(3) WINDOWS でのディレクトリ

~/	ホームディレクトリ
./	カレントディレクトリ
drive:/dir	ドライブ drive のディレクトリ /dir
//node/dir	ホスト名 node のディレクトリ /dir

例

```
/* シンボル ABC のフルパスの名前を求める */
char fname[256];
strcpy(fname, "ABC");
Filname("#SYMBOL#", fname, sizeof(fname), 0)
```

12.1.6 ファイル名の一覧を表示

【関数名】

FilSelect

【機能】

ファイル名を画面に表示し、該当するものを選ぶ。

【呼出し形式】

```
int FilSelect(int seltyp, const char* caption, const char* fctg, const char* fmask,
              char* fname, size_t fname_len)
```

【入力引数】

seltyp　選択の種類（ダイアログを使用する場合にのみ有効）
- 0 = 呼び出し用　：ダイアログの OK ボタンは「開く」と表示される。既存ファイルの選択および入力が可能。
- 1 = 書き込み用　：ダイアログの OK ボタンは「保存」と表示される。既存ファイルの選択および新ファイル名の入力が可能。
- 2 = 書き込み用　：ダイアログの OK ボタンは「ＯＫ」と表示される。既存ファイルの選択および新ファイル名の入力が可能。

caption　ダイアログのキャプション（ダイアログを使用する場合にのみ有効）
ダイアログの左上に表示される文字列を指定する。全角文字は EUC。
NULL ポインターまたは文字列長さが 0 のときはデフォルトのキャプションが表示される。
デフォルトのキャプション ：″ファイルの選択″

fctg　ファイルの種類 (NULL-char-terminated)。たとえば "#MODEL#", "#SYMBOL#"。

fmask　ファイル名 (NULL-char-terminated)
- ワイルドカード '*'　：任意の文字列
- ワイルドカード '?'　：任意の一文字

fname_len　配列 fname の大きさ。

【出力引数】

fname　選択されたファイル名 (NULL-char-terminated)
ディレクトリパスおよびファイル拡張子は ACAD.SET の fctg での定義と一致する部分は fname には含まない。
たとえば ACAD.SET でのディレクトリが "D:/acad/files/"、拡張子が !.MDL! のとき以下のようになる。

```
D:/acad/files/ABC.MDL          ->  ABC
D:/acad/files/subdir/ABC.MDL   ->  subdir/ABC
D:/acad/files/ABC.PLT          ->  ABC.PLT
C:/temp/ABC.MDL                ->  C:/temp/ABC
```

フルパスのファイル名を得るには、この結果を Filname() 関数に渡す。(例を参照)

【返り値】

- 0　：正常に選択された。または 入力ファイル名にワイルドカードが含まれていない。
- 1　：ファイルが選択されなかった。
- 3　：ファイル名が長いため fname に収まらない。

注意

(1) 返り値が 1、3 のとき、メッセージ領域にエラーメッセージを表示している。
(2) ファイルが選択されたとき、そのファイル名が入力されたものとして、セッションファイルに記録される。

例

```
int seltyp = 0;                     /* 呼び出し用 */
const char* caption = NULL;         * デフォルトのキャプション */
const char* fctg    = "#MODEL#";    /* モデルファイル */
const char* fmask   = "ABC*";       /* ファイル名が ABC で始まるもの */
char fname[256];
int errcod = FilSelect(seltyp, caption, fctg, fmask, fname, sizeof(fname));
if (errcod)
  return 1; /* 選択されなかった */
errcod = Filname(fctg, fname, sizeof(fname), 0); /* フルパスにする */
if (errcod == 0) {
```

```
  /* 正常 */
} else if (errcod == 1) {
  /* #MODEL# が ACAD.SET に定義されていない */
  /* 起動ディレクトリで拡張子なしのファイル名になっている */
  /* 正常としてもよい場合もある */
} else {
  /* フルパスの名前が作成できなかった */
}
```

12.1.7 現日時を取得

【関数名】

gcdate

【機能】

現日時を得る。

【呼出し形式】

```
void gcdate(short idate[], short itime[])
```

【出力引数】

idate	日付
	idate[0] : 日 (1 - 31)
	idate[1] : 月 (1 - 12)
	idate[2] : 西暦年
itime	時刻
	itime[0] : 時 (0 - 23)
	itime[1] : 分 (0 - 59)
	itime[2] : 秒 (0 - 59)

12.1.8 ビット列の取り出し

【関数名】

ifld

【機能】

ビット列の取り出し

【呼出し形式】

```
short ifld(short *idat, int ks, int ke)
```

【入力引数】

idat	データの取り出し元
is	取り出すビットの先頭位置 (1 <= is)
ie	取り出すビットの最終位置 (is <= ie, ie - is + 1 <= 16)

【返り値】

取り出したビット列

注意
is と ie は idat の同じ short ワード内でなければならない。
また、ビット列の長さは 16 bits 以内でなければならない (ie - is + 1 <= 16)。
例えば ifld(idat, 15、17) は idat[0] と idat[1] にまたがるのでエラーとなる。
エラーの時の返り値は 0 になる。

例
```
short idat[2] = {0, 6};
ifld(idat, 30, 31);
```

12.1.9 ビット列の挿入

【関数名】

insfld

【機能】

ビット列の挿入

【呼出し形式】

```
void insfld(short *idat, int is, int ie, int ival)
```

【入力引数】

is	挿入先のビットの先頭位置 (1 <= is)
ie	挿入先のビットの最終位置 (is <= ie)
ival	挿入するビットデータ

【入出力引数】

idat	データの挿入先

注意
ival の下位 N ビット (N = ie - is + 1) を、idat の is ビット目から
ie ビット目にコピーする。idat の他のビットは変更されない。
is と ie は idat の同じ short ワード内でなければならない。
また、ビット列の長さは 16 bits 以内でなければならない (ie - is + 1 <= 16)。

例
```
/* mask の５ビット目を１にする */
short mask = 0;
insfld(&mask, 5, 5, 1);
```

12.1.10 バイトの取り出し

【関数名】

ibyte

【機能】

バイトの取り出し

【呼出し形式】

```
short ibyte(char *idat, int ipos)
```

【入力引数】

idat	データの取り出し元
ipos	取り出すバイト位置 (1 <= ipos)

【返り値】

バイトの値

例

```
short idat[2] = {0, 259};
ibyte((char *)idat, 4);
```

12.1.11 バイトの挿入

【関数名】

insbyte

【機能】

バイトの挿入

【呼出し形式】

```
void insbyte(char *idat, int ipos, int ival)
```

【入力引数】

ipos	挿入先のバイト位置 (1 <= ipos)
ival	挿入するバイトの値 (0 <= ival <= 255)

【入出力引数】

idat	データの挿入先 idat の ipos バイト目が更新される。他のバイトのは変更されない。

例

```
/* mask の３バイト目を 255 にする */
short mask[2] = {0, 0};
insbyte((char *)mask, 3, 255)
```

12.1.12 内部ソート

【関数名】

xsrt

【機能】

内部ソート

【呼出し形式】

```
void xsrt(char *cstr, short *nrec, short *nkey, short keys[][4], short *iopt,
          short (*usrcmp)(char *cstr, short *nkey, short keys[][4],
                          short *irec1, short *irec2, size_t cstr_len),
          short *ier, size_t cstr_len)
```

【入力引数】

cstr　　ソートするレコードの並び（配列）
cstr_len　レコード長 (256 バイト以下)。
nrec　　レコード数
nkey　　ソートキーの数 (1-)
keys　　ソートキー。
ユーザ比較ルーチンを指定するときは、その比較ルーチンにおいて必要な任意の内容を設定してかまわない。
デフォルト比較ルーチンを使用する場合は下記のパラメータを設定すること。パラメータの値の整合性確認をおこなう。一つのキーは４つのパラメータからなる。複数のキーを指定するときは、優先順位の高いキーから低いキーの順に記述すること。

keys[j][0] データの種類
　　0 : 文字列
　　1 : short (16 bits)
　　2 : int (32 bits)
　　3 : float
　　4 : double
keys[j][1] ソートオーダ
　　ASCENDING : 昇順
　　DESCENDING : 降順
keys[j][2] キーの開始位置（バイト, 1- レコード長）
keys[j][3] キーの長さ（バイト, 1- レコード長）

iopt　　オプションスイッチ
0 : 指定なし
1 (bit 16) : 重複レコード除去
2 (bit 15) : ユーザ比較ルーチンを指定
bit 14-1　　未使用

usrcmp　　ユーザ比較ルーチン
レコードの順序をきめるために２つのレコードを比較するルーチン。
比較結果を以下の整数値 (short) で返す関数であること。

-1	：最初のレコードが二番目のレコードより前
0	：ふたつのレコードは（キーが）同じ
1	：最初のレコードが二番目のレコードより後

比較ルーチンの引数は以下のとおり。

```
short ucmp(char *cstr, short *nkey, short keys[][4],
           short *irec1, short *irec2, size_t cstr_len)
```

cstr, nkey, keys, cstr_len	xsrt の引数がそのまま渡る。
irec1, irec2	比較するレコードの番号 (1-nrec)

【出力引数】

cstr　　ソートされたレコードの並び
nrec　　レコード数
重複レコード除去を指示したときには変更される。
ier　　エラーステータス

0	：正常終了
1	：レコード数が０以下
2	：レコード長が０以下または 256 より大きい。

```
3       ：キーの数が０以下
4       ：キーのデータ種類が不正
5       ：ソートオーダが不正
6       ：キーの開始位置が不正
7       ：キーの長さが不正
```

例１

６つの点の座標 (X,Y) を X 座標で昇順、つぎに Y 座標で降順に並べるとする。

```
もとの点列  →  ソート後の点列
                重複あり        重複なし
(100, 150)      (100, 150)      (100, 150)
(150, 100)      (100, 100)      (100, 100)
(150, 150)      (150, 150)*     (150, 150)
(200, 100)      (150, 150)*     (150, 100)
(100, 100)      (150, 100)      (200, 100)
(150, 150)      (200, 100)
```

```
short ier;
short nkey = 2; /* キーは２つ */
short keys[][4] = {{4, ASCENDING, 1, 8}, {4, DESCENDING, 9, 8}};
short iopt = 1; /* 座標の同じ点があったらそれを除去する */
short nrec = 6; /* レコード数 (= 点の数 ) */
DPOINT pbuf[6] = {{100.0, 150.0}, {150.0, 100.0}, {150.0, 150.0},
                  {200.0, 100.0}, {100.0, 100.0}, {150.0, 150.0}};

/* ユーザ比較関数は使用しないので NULL とする。
 * S_fp の typedef は acaddef.h にある。*/
xsrt((char *)pbuf, &nrec, &nkey, keys, &iopt, (S_fp)NULL,
      &ier, sizof(DPOINT));
```

例２

比較関数を指定する例としては、原点にもっとも近い点から遠い点へとならべるような場合がある。このときはふたつの点の原点からの距離を比較し、距離の短い方を前とする判定をすれば良い。この例では、引数 nkey, keys の内容は参照しない。

```
short ier;
short nkey = 0;   /* 参照しない */
short keys[1][4]; /* 参照しない */
short iopt = 3;   /* 比較モジュール指定 (2) + 重複除去 (1) */
short nrec = 6;   /* レコード数 (= 点の数 ) */
DPOINT pbuf[6] = {{100.0, 150.0}, {150.0, 100.0}, {150.0, 150.0},
                  {200.0, 100.0}, {100.0, 100.0}, {150.0, 150.0}};
xsrt((char *)pbuf, &nrec, &nkey, keys, &iopt, usrcmpxy,
      &ier, sizof(DPOINT));

/* 比較関数 */
short usrcmpxy(char *cstr, short *nkey, short keys[][4],
               short *irec1, short *irec2, size_t cstr_len)
{
  DPOINT *p;
  double d1, d2;

  p = (DPOINT *)(cstr + cstr_len * (*irec1 - 1));
  d1 = p->x * p->x + p->y * p->y;

  p = (DPOINT *)(cstr + cstr_len * (*irec2 - 1));
  d2 = p->x * p->x + p->y * p->y;

  if (d1 < d2) {
    return -1;
  } else if (d1 == d2) {
```

```
    return 0;
  } else {
    return 1;
  }
}
```

12.1.13 二次元配列の列を交換

【関数名】

xswap

【機能】

二次元配列の列を交換する。

【呼出し形式】

```
void xswap(int type, void *ibuf, int ncol, int nrow1, int nrow2)
```

【入力引数】

type	データの型 0 :short (16 bits) 1 :int (32 bits) 2 :float 3 :double
ibuf	データの配列 ibuf[][ncol]
ncol	列の長さ
nrow1	交換する列の番号 (1 -)
nrow2	交換する列の番号 (1 -)

【出力引数】

ibuf	列を交換した配列

12.1.14 SJIS から EUC に漢字コード変換

【関数名】

sjis2euc

【機能】

MS 漢字コード文字列を日本語 EUC 文字列に変換する。

【呼出し形式】

```
int sjis2euc(const char* src, char* dst, size_t len_dst)
```

【入力引数】

src	MS 漢字コード文字列 (NULL terminated)。
len_dst	配列 dst の長さ。

【出力引数】

dst　　日本語 EUC 文字列 (NULL terminated)。

【返り値】

dst に格納された文字列の長さ。

注意

MS 漢字コード文字列には半角カナ (1 バイト)、JIS コードに存在しない MS 拡張漢字コードを含めてはいけない。

12.1.15 EUC から SJIS に漢字コード変換

【関数名】

euc2sjis

【機能】

日本語 EUC 文字列を MS 漢字コード文字列に変換する。

【呼出し形式】

```
int euc2sjis(const char* src, char* dst, size_t len_dst)
```

【入力引数】

src　　日本語 EUC 文字列 (NULL terminated)。
len_dst　　配列 dst の長さ。

【出力引数】

dst　　MS 漢字コード文字列 (NULL terminated)。

【返り値】

dst に格納された文字列の長さ。

第 13 章　ユーザ定数の設定

13.1 概要

ユーザのデータ領域を持つことができます。内容を会話形式で設定変更できます。
また設定した値はモデルに保存、取り出しができます。

以下に、必要な関数、およびメニューファイルの例を示します。これらを変更して使用できます。

メニューファイル

USERCMD.MEN	：値の設定を行うためのコマンドを定義する。
USEROSM.MEN	：会話形式で値の設定を行うためのメニューを定義する。

ソースコード

usrcom.c	：以下の３つの関数を記述します。
comusrint	：定数を初期化する関数。
comusrset	：会話形式で値の設定、表示をする関数。
comusrio	：定数をファイル（定数ファイル・モデルファイル）に書き込む、またはファイルから読み込む関数。

これらの関数が呼ばれるのは以下の場合です。

comusrint 関数
(1)　Advance CAD の起動時
(2)　定数の呼出しコマンド
(3)　モデルの新規開始コマンド
(4)　モデルの呼出しコマンド（新規モード）でモデルの読み込みの前

comusrset 関数
値の設定を行うユーザ定義コマンド (コマンド番号 53,98,*)

comusrio 関数
(1)　定数ファイルからの読み込み。comusrint 関数の 1) から 4)。
(2)　定数ファイルへの書込み。（定数の保存コマンド）
(3)　モデルファイルからの読み込み。（モデル呼出しコマンド一新規モード）
(4)　モデルファイルへの書込み。（モデルの保存コマンドおよびサブモデルの作成コマンド）

13.2 メニューの作成

ここでは USERCMD.MEN と USEROSM.MEN について説明します。

13.2.1 USERCMD.MEN の作成

USERCMD.MEN の例

```
/ AdvanceCAD ver 15 User defined command list
/
/ + [dispatcher#, driver#, form#] !command_name!
/ V [dispatcher#, driver#, form#] !command_name!
/       command name :== [A-Z][A-Z0-9/_]*
/
/ [ 48, n, n] are reserved for User defined modifier
/ [ 32, n, n] are reserved for User defined command
/ [ 53, 98, n] are reserved for User defined command RVP/USER
Command
V [ 53, 98,   0] !RVP/USER!
V [ 53, 98,   1] !USER/INT1!
V [ 53, 98,   2] !USER/INT2!
V [ 53, 98,   3] !USER/TXT!
V [ 53, 98,   4] !USER/RAL1!
V [ 53, 98,   5] !USER/RAL2!
V [ 53, 98,   6] !USER/DBL!
/ End of file
```

RVP/USER 会話形式で値の設定を行うためのメニューを呼出すコマンドの、コマンド名とコマンド番号を定義します。コマンド番号は 53,98,0 固定です。

USER/INT1 から USER/DBL には、個々の定数を変更するためのコマンドを定義します。
コマンドのディスパッチャ番号は 53, ドライバ番号は 98 で固定です。
フォーム番号は 1 から番号を割り当てます。フォーム番号は comusrset 関数がどの定数かを判別するのに使います。

13.2.2 USEROSM.MEN の作成

USEROSM.MEN の例

```
/
/ Advance CAD ver 15 User Onscreen menu definition
/ Menu [menu#,category#,screen#,colour#]
/ + <row#,col#> "text" !command! [menu1#,menu2#,colour#]
/ L <row#,col#> "text" !letter!  [menu1#,menu2#,colour#]
/ T <row#,col#> "text" !text!    [menu1#,menu2#,colour#]
/ N <row#,col#> "text" [menu1#,menu2#,colour#,type#,value]
/       type 0=scalar, 1=Mark number, 2=colour number,
/            3=key number, 4=Line font number, 5=Line weight number
/
/ Pagename = rvp_user
Menu    [rvp_user,  3, 10,  c0]
+ < 1, 1> " 例題 "        !RVP/USER!      [rvp_user, none, c0]
+ < 2, 1> " 整数型＃１          "        !USER/INT1!
+ < 3, 1> " 整数型＃２          "        !USER/INT2!
```

```
  +< 4, 1>"文字型（１－４文字）"        !USER/TXT!
  +< 5, 1>"実数型＃１              "    !USER/RAL1!
  +< 6, 1>"実数型＃２              "    !USER/RAL2!
  +< 7, 1>"倍精度実数型            "    !USER/DBL!
  +<10, 1>"終了"                        !RVP/END!
  /
  / End of file

Menu [rvp_user,  3, 10,  c0]
```

メニューページ名、カテゴリ番号、表示領域、表示色の定義をします。
カテゴリは 3、表示領域は 10 固定です。メニューの表示色は無視されます。

```
+< 1, 1>"例題"  !RVP/USER!  [rvp_user, none, c0]
```

会話形式で値の設定を行うためのメニューを呼出すコマンドを割付けます。
このコマンドが選択されたときに表示するメニューページ名が必要です。
メニューページ名の指定が無いとメニューは表示しません。

```
+< 2, 1>"整数型＃１              "    !USER/INT1!
```

個々の定数を変更するコマンドをメニューページに割付けます。

```
+<10, 1>"終了"                        !RVP/END!
```

会話形式で値の設定を終了させるコマンドのを割付けます。
これが選択されると会話形式での値設定を終わり、画面は以前の状態に戻ります。

13.3 ソースコードの作成

13.3.1 ユーザのデータ領域

ユーザのデータ領域を定義します。
この例では、ソースコードの先頭にファイルスコープの静的変数として定義しています。また、ひとつの構造体として型宣言していますが、構造体でなくてもかまいません。構造体にする場合は double, float, short, char の順に並べると、構造体要素間にパディングが入らず安全です。文字列は長さは偶数にします。
データ領域を決める場合は、将来の変更・拡張を考慮しておくことを勧めます。たとえばユーザデータ領域の先頭に、データのバージョン番号などをつけておけば、将来データ領域を変更するときに整合させることができます。

```
/*
 * Filename : comusr.cpp
 * Category : User Inrerface
 *            comusrint, comusrio, comusrset
 */

#include <math.h>
#include "acadusr.h"
#include "acadupi.h"

/* User defined data */
typedef struct {
  double dbl;           /* Double precision data */
  float  rarr[2];       /* Real data array */
  short  iarr[2];       /* Integer data array */
  char   txt[4];        /* Text */
} COMUSR;

static COMUSR comusr;   /* User common area */
```

```
static int dblwds = sizeof(comusr.dbl) / 2;  /* word size of double */
static int ralwds = sizeof(comusr.rarr) / 2; /* word size of float array */
static int intwds = sizeof(comusr.iarr) / 2; /* word size of short array */
static int txtwds = sizeof(comusr.txt) / 2;  /* word size of text */
static int totalwds = sizeof(comusr) / 2;    /* Total word size */
```

13.3.2 comusrint 関数

ユーザデータを初期化する関数を記述します。

```
/************************************************************************
  Purpose
    Set the default values to the User defined data.
*/
void comusrint(void) {
  comusr.dbl = 100.0;
  comusr.rarr[0] = 10.0f;
  comusr.rarr[1] = 20.0f;
  comusr.iarr[0] = 1;
  comusr.iarr[1] = 2;
  memcpy(comusr.txt, "ABCD", 4L);
} /* comusrint */
```

13.3.3 comusrset 関数

会話形式で値の設定、表示をする関数を記述します。

```
/************************************************************************
  Purpose
    Set and/or Display parameters for the User RVP menu.
    RVP/USER
  Input Arguments
      KEYANS  Input data type.
              3 : Scalar
              4 : Text string
      FORMNO  Command form number.
              1 = USER/INT1
              2 = USER/INT2
              3 = USER/TXT
              4 = USER/RAL1
              5 = USER/RAL2
              6 = USER/DBL
      SC      Scalar.
      NTEXT   Text string. (NULL-terminated)
      NCH     Text length.
      KEYSET  Setting control switch.
              0 : Display only.
              1 : Set and display.
  Output Arguments
      IER     Error code.
              0 : No error
*/
void comusrset(short *keyans, short *formno, double *sc, short *ntext,
               short *nch, short *keyset, short *ier)
{
  #define RVPCTG 3
```

```
#define DSPTNO 53
#define DRIVNO 98
int mode, im3swt;
char itext[4097];
double v;

*ier = 1;

if (*keyset == 0)
  goto L100;

/* Set new value to the common area. */
switch (*formno) {
  case 1: /* Integer data #1 */
    if (*keyans != TknSCL)
      return;
    if (fabs(*sc) > 32767.0)
      return;
    comusr.iarr[0] = (short)(*sc);
    break;
  case 2: /* Integer data #2 */
    if (*keyans != TknSCL)
      return;
    if (fabs(*sc) > 32767.0)
      return;
    comusr.iarr[1] = (short)(*sc);
    break;
  case 3: /* Text data #1 */
    if (*keyans != TknTXT)
      return;
    if (*nch < 1 || *nch > sizeof(comusr.txt))
      return;
    memset(comusr.txt, ' ', sizeof(comusr.txt));
    memcpy(comusr.txt, ntext, *nch);
      break;
  case 4: /* Real data #1 */
    if (*keyans != TknSCL)
      return;
    comusr.rarr[0] = (float)(*sc);
    break;
  case 5: /* Real data #2 */
    if (*keyans != TknSCL)
      return;
    comusr.rarr[1] = (float)(*sc);
    break;
  case 6: /* Double precision data #1 */
    if (*keyans != TknSCL)
      return;
    comusr.dbl = *sc;
    break;
  default:
    return;
}

/* Display. */
L100:
switch (*formno) {
  case 1: /* Integer data #1 */
    v = (double)comusr.iarr[0];
    mode = 3;
    im3swt = 0;
    break;
```

```
    case 2: /* Integer data #2 */
      v = (double)comusr.iarr[1];
      mode = 3;
      im3swt = 0;
      break;
    case 3: /* Text data #1 */
      memcpy(itext, comusr.txt, sizeof(itext));
      mode = sizeof(itext) * 10;
      im3swt = 0;
      break;
    case 4: /* Real data #1 */
      v = (double)comusr.rarr[0];
      mode = 3;
      im3swt = 1;
      break;
    case 5: /* Real data #2 */
      v = (double)comusr.rarr[1];
      mode = 3;
      im3swt = 1;
      break;
    case 6: /* Double precision data #1 */
      v = comusr.dbl;
      mode = 3;
      im3swt = 1;
      break;
    default:
      return;
  }

  /* Display parameter */
  if (*formno == 3) {
    Rvpdisp(RVPCTG, DSPTNO, DRIVNO, *formno, mode, itext, *keyset, im3swt);
  } else {
    Rvpdisp(RVPCTG, DSPTNO, DRIVNO, *formno, mode, &v, *keyset, im3swt);
  }

  *ier = 0; /* No error */
} /* comusrset */
```

13.3.4 comusrio 関数

ユーザデータをファイル (定数ファイル・モデルファイル) に書き込み、またはファイルから読み込む関数を記述します。

この関数は最初にユーザデータのサイズを調べるため使用されます (*key = 0)。
このときサイズが 0 であれば、ユーザデータの書き込み (*key = 1)、または読み込み (*key = 2) はおこないません。
以下の例では、ユーザデータの入出力をしないように、ユーザデータのサイズを 0 に設定しています。

```
    *iwdsize = 0;
```

この行を変更して、正しいユーザデータのサイズを設定すればファイルへの入出力を行います。
ユーザデータは文字列、整数、実数、倍精度実数ごとに分けて処理しなければなりません。

```
/*************************************************************************
  Purpose
    Read/Write the User defined data in the model.
  Input Arguments
```

```
        key     I/O control switch.
                0 : Get the word size of the user common area.
                1 : Write the user common area to the file.
                2 : Read the user common area from the file.
  Output Arguments
        iwdsize Word size of the user common area.
        ier     Error code.
                0 : No error
*/
void comusrio(short *key, short *iwdsize, short *ier) {
  *iwdsize = 0;
  /* *iwdsize = totalwds; */

  switch (*key) {
    case 0: /* Get the word size of the user common area. */
      *ier = 0;
      break;
    case 1: /* Write the user common area to the file. */
      *ier = MdlfWrite(dblwds, 8, &comusr.dbl);
      *ier = MdlfWrite(ralwds, 4, comusr.rarr);
      *ier = MdlfWrite(intwds, 2, comusr.iarr);
      *ier = MdlfWrite(txtwds, 1, comusr.txt);
      break;
    case 2: /* Read the user common area from the file. */
      *ier = MdlfRead(dblwds, 8, &comusr.dbl);
      *ier = MdlfRead(ralwds, 4, comusr.rarr);
      *ier = MdlfRead(intwds, 2, comusr.iarr);
      *ier = MdlfRead(txtwds, 1, comusr.txt);
      break;
    default:
      *ier = 1;
  }
} /* comusrio */
```

13.4 ここで使用する関数

13.4.1 ファイルにデータを書き込む

【関数名】

MdlfWrite

【機能】

ファイルにデータを書き込む

【呼出し形式】

```
int MdlfWrite(int nwds, int kind, const void* idata)
```

【入力引数】

nwds　　書込むデータサイズ (2byte で 1 サイズ。nwds=(書込む byte 数 + 1) / 2 となる)
kind　　書込むデータ種類
　　1　　:char

```
                2       :short
                4       :float, int
                8       :double
```
idata　書込むデータ

【返り値】

0　：　正常終了

注意

comusrio 以外のところでは使用できません。
違ったデータ種類を１度に書いてはいけません。

13.4.2 ファイルからデータを読込む

【関数名】

MdlfRead

【機能】

ファイルからデータを読込む

【呼出し形式】

```
int  MdlfRead(int nwds, int kind, void *idata)
```

【入力引数】

nwds　読込むデータサイズ（2byte で 1 サイズ。nwds=（読込む byte 数 + 1）/ 2 となる）
kind　読込むデータ種類

```
                1       :char
                2       :short
                4       :float, int
                8       :double
```

【出力引数】

idata　データを格納する配列

【返り値】

0　：正常終了

注意

comusrio 以外のところでは使用できません。
違ったデータ種類を１度に読んではいけません。

13.4.3 定数を表示する

【関数名】

Rvpdisp

【機能】

定数を表示する。

【呼出し形式】

```
void Rvpdisp(int ictg, int idsp, int idrv, int iform, int mode, void *vdata, int ieras,
             int im3swt)
```

【入力引数】

ictg	コマンドのカテゴリ番号。常に 3 にする。
idsp	コマンドのディスパッチャ番号。53 にする。
idrv	コマンドのドライバ番号。98 にする。
ifor	コマンドのフォーム番号。
mode	表示するデータの種類。 3 : 数値 文字数*10 : 文字列
vdata	表示するデータ 倍精度実数 (double *) または 文字列 (char *)
ieras	表示する前に表示部分を消すかどうか。 1 :消す 0 :消さない
im3swt	数値 (mode=3) の時に小数点を表示するかどうか。 1 :表示する 0 :表示しない

注意

comusrset 以外のところでは使用できません。

13.5 モデル管理のカスタマイズ

ソースコード usrmdm.c には、以下の関数が含まれています。これらの関数はこのままでは何も処理しない空の関数ですが、関数を書き換えることによりユーザ独自の処理を追加できます。

- 関数一覧

mdm001 **ユーザが作成したファイル管理プログラムを Advance CAD から起動するためのもの。**

mdm101 **モデルファイルおよびサブモデルファイルに対するオペレーションごとにユーザ独自の処理を追加する。**

13.5.1 ファイル管理プログラムの起動

【関数名】

mdm001

【機能】

この関数は、ユーザが作成したファイル管理プログラムを Advance CAD から起動するためのものである。必要ならば、それから得られたファイル名などを割り込みレベルの低いコマンドに対するテキスト入力の代わりにすることもできる。

この関数は割り込みコマンド MDM/TOOL が選択されたときに呼び出されるので、ユーザは必要な処理を行なうようにこの関数を書き換えて、希望の処理を割り込みコマンド MDM/TOOL に追加できる。

【呼出し形式】

```
void mdm001(char *ctext, short *ltext, size_t len_ctext)
```

【入力引数】

len_ctext 配列 ctext の長さ (2048)

【出力引数】

ctext　　コマンド用のテキスト (char ctext[len_ctext])
ltext　　コマンド用のテキストの文字数
　　0　　：コマンド入力無し。
　　ltext が 2048 を越えている場合には 2048 と見なされる。

13.5.2 独自処理の追加

【関数名】

mdm101

【機能】

この関数は、モデルファイルおよびサブモデルファイルに対するオペレーションごとにユーザ独自の処理を追加するとき使用する。
この関数は各オペレーションが正常終了した時に呼び出される。この関数をユーザ独自の処理を行なうように書き換えて、通常のオペレーションに対して希望の処理を追加できる。

【呼出し形式】

```
void mdm101(short *otyp, char *fname, short *lfname,char *oname, short *loname,
            size_t len_fname, size_t len_oname)
```

【入力引数】

otyp　　オペレーションタイプ
　　100　　：アクティブモデルの保存
　　101　　：アクティブモデルの部分的な保存。またはサブモデルファイルの作成。
　　102　　：ピクチャ書き込み
　　103　　：所有者名の変更
　　104　　：削除
　　105　　：ファイル名の変更
　　106　　：コピー
　　107　　：テープからのコピー
　　200　　：呼出し
　　201　　：部分的な呼出し。またはサブモデルの配置や更新による再配置。
fname　　モデルファイル名
lfname　　モデルファイル名の文字数
oname　　新しい所有者名・元のモデルファイル名
loname　　新しい所有者名・元のモデルファイル名の文字数
　　所有者名の変更 (otyp == 103)
　　ファイル名の変更 (otyp == 105)
　　コピー (otyp == 106) のときだけ設定される。
len_fname 配列 fname の長さ

len_oname 配列 oname の長さ

Appendix A バージョン 15 での変更点

バージョン 15 で変更した機能や関数についての概要を説明します。

A.1 ユーザディスパッチャ関数などの関数プロトタイプ宣言ファイル

以前の関数プロトタイプ宣言ファイル
```
usrtype.h
```

以後の関数プロトタイプ宣言ファイル
```
acadupi.h
```

usrtype.h は acadupi.h に名前を変更しました。
usrtype.h 内にユーザ独自の関数プロトタイプなどを追加していた場合は acadupi.h 内の記述と重複する部分を usrtype.h から削除してください。
usrtype.h に何も記述を追加していない場合は usrtype.h は不要ですので削除してください。

A.2 ユーザディスパッチャ関数の引数

以前の関数宣言
```
void dspatch32(int keyans, DPOINT *pnt, double sc, char *text)
```

以後の関数宣言
```
void dspatch32(TOKEN *token)
```

新しい引数 token との対応は次のとおりです。
```
keyans    token->typ
pnt       token->pnt
sc        token->scl
text      token->txt
```

dspatch64、dspatch80、dspatch88 も同様です。

ソースコード
dspatch32.c、dspatch64.c、dspatch80.c、dspatch88.c

A.3 構造体 TOKEN

アプリケーションのトークンを配送するための構造体です。トークンは、コマンドが選択された、マウスボタンがクリックされた、文字列が入力された、コマンドが終了するなどのイベントが発生したときにそれをコマンド処理関数に通知するために使用します。

TOKEN 構造体にはトークンの種類を表すメンバー変数 typ があります。
他にはトークンの種類に応じて、点座標、文字列あるいは数値データを保持するメンバー変数があります。バックスペーストークンなど、これらのメンバー変数を使用しない（設定しない）トークンもあります。

トークンはアプリケーションの制御部がコマンド処理関数に配送するもので、読み出し専用です。しかしユーザディスパッチャの引数 token は const 修飾していませんので構造体のメンバー変数の内容を変更できますが、変更しないで下さい。

構造体 TOKEN は ヘッダーファイル acaddef.h で宣言しています。
トークンの種類は acadprm.h に宣言があります。

ソースコード
acaddef.h、acadprm.h

A.4 トークン

次のトークンが追加／変更されています。
コマンド処理関数を新しい方法に従って書き換える必要があります。

追加されたトークン
　　修飾子　　TknMDF
　　コマンド終了　　TknEXIT
　　アイテム選択TknITM（後述：アイテム選択関数で説明）
　　テンポラリポイントTknPNT（後述：テンポラリポイント関数で説明）

変更のあるトークン
　　ディジタイズ座標TknDIG

● 修飾子トークン (TknMDF)

以前はコマンドトークンと修飾子トークンは共にコマンドトークン（トークン種類番号 =1）としていましたが、修飾子トークンを分離しました。
以前はコマンドトークンと修飾子トークンを区別するためにコマンド識別番号を使用しなければなりませんでしたが、今後はトークン種類で判定できます。

コマンドトークンと修飾子トークンはコマンド識別番号を持っています。

TOKEN 構造体メンバー変数 cid	
cid[0]	ディスパッチャ番号
cid[1]	ドライバ番号
cid[2]	フォーム番号

それ以外のトークンのときにコマンドや有効になっている修飾子の識別番号を得るためには下記の関数を使用します。

コマンドの識別番号を得る関数
引数の 1 はコマンドレベルを表します。
コマンドレベルは dspatch32 配下では 1、dspatch64 配下では 3 などです。

ldispatch(1)	ディスパッチャ番号
ldriver(1)	ドライバ番号
lformat(1)	フォーム番号

修飾子の識別番号を得る関数

引数の値はコマンドレベルに関係なく常に 2 です。

Idispatch(2)	ディスパッチャ番号
Idriver(2)	ドライバ番号
Iformat(2)	フォーム番号

次のようなコマンド処理関数を例に考えてみます。

```
void ucmd01(int keyans, DPOINT *pnt, double sc, char *text)
{
  switch (keyans) {
  case 1:
    if (command_.idispatch2 == 0) {
      /* コマンドの呼び出しのとき一回だけ実行される。初期化する */
    } else if (command_.idispatch2 == 34 &&
               command_.idriver2 == 1 &&
               command_.iformat2 == 1) {
      /* 修飾子 USEACT の処理 */
    } else if (command_.idispatch2 == 34 &&
               command_.idriver2 == 1 &&
               command_.iformat2 == 2) {
      /* 修飾子 ADD の処理 */
    } else {
      /* 間違った修飾子をクリア */
      command_.idispatch2 = 0;
    }
    break;
    （略）
}
```

これは引数 token を使って次のように書き換えます。

```
void ucmd01(TOKEN *token)
{
  switch (token->typ) {
  case TknCMD:
  /* コマンドの呼び出しのとき一回だけ実行される。初期化する */
    break;
  case TknMDF:
    if (token->cid[0] == 34 &&
        token->cid[1] == 1 &&
        token->cid[2] == 1) {
      /* 修飾子 USEACT の処理 */
    } else if (token->cid[0] == 34 &&
               token->cid[1] == 1 &&
               token->cid[2] == 2) {
      /* 修飾子 ADD の処理 */
    } else {
      /* 間違った修飾子をクリア */
      cmdmdfcla(1);
    }
    break;
    （略）
}
```

グローバル変数のコマンド識別番号は使っていません。また、修飾子をクリアするのにグローバル変数の値を変更するのではなく、関数 cmdmdfcla を使用することに注意してください。引数の 1 はコマンドレベルです。

● コマンド終了トークン (TknEXIT)

コマンド処理関数ではコマンドが終了する直前に何らかの後始末をするのが一般的です。
以前はコマンド終了は ＣＥ トークン（トークン種類番号 =5）でグローバル変数 command_.iccinsert の値が 1 のときという判定でした。
この方法はわかりにくいものでした。新しいコマンド終了トークンでわかり易くなります。

次のようなコマンド処理関数を例に考えてみます。

```
void ucmd01(int keyans, DPOINT *pnt, double sc, char *text)
{
  switch (keyans) {
    （略）
  case 5:
    if (command_.iccinsert == 1) {
      /* コマンドの終了。後始末をする */
    } else {
      /* <CE> の処理 */
    }
    break;
    （略）
}
```

これは引数 token を使って次のように書き換えます。

```
void ucmd01(TOKEN *token)
{
  switch (token->typ) {
    （略）
  case TknEXIT:
    /* コマンドの終了。後始末をする */
    break;
  case TknEOC:
    /* <CE> の処理 */
    break;
  （略）
}
```

● ディジタイズ座標トークン (TknDIG)

ディジタイズが行われたビューポートの番号を TOKEN 構造体のメンバー変数 vp に設定します。
以前はディジタイズが行われたビューポート番号を知りたい時は関数 tknvie101 を使用していましたが、これは廃止しました。
またディジタイズが行われたビューポート内のピクチャ番号を知りたい時は関数 tknpic101 を使用していましたが、関数 Pic102 を使用します。
tknpic101 は廃止しました。

次のようなコマンド処理関数を例に考えてみます。

```
void ucmd01(int keyans, DPOINT *pnt, double sc, char *text)
{
  switch (keyans) {
    （略）
  case 6:
    if (tknvie101() == 1) {
      /* ビューポート番号が 1 のとき、ピクチャ番号を得る */
      int pic = tknpic101();
    }
    break;
    （略）
```

```
}
```

これは引数 token を使って次のように書き換えます。

```
void ucmd01(TOKEN *token)
{
  switch (token->typ) {
  (略)
  case TknDIG:
    if (token->vp == 1) {
      /* ビューポート番号が 1 のとき、ピクチャ番号を得る */
      int pic = Pic102(token->vp);
    }
    break;
    (略)
}
```

A.5 アイテムの選択

以前の関数
　Ident00
以後の関数
　IdentItem、IdentItemCandidate、IdentInfoCount、IdentInfo

アイテムの選択関数 Ident00 は、次候補アイテムの処理などを追加した新しい関数 IdentItem に書き換えます。Ident00 は廃止しました。
選択されたアイテムの詳細情報は、以前は comident.h のグローバル変数を参照していましたが、関数 IdentInfoCount、IdentInfo を使用します。comident.h は廃止しました。詳細は『プログラミングマニュアル』６章の IdentItem および２章の例を参照してください。

A.6 複数アイテムの自動選択

以前の関数
　identgany
V15 の関数
　IdentItems、IdentItemsCandidate
関連関数
　rptitmnum、rptitmptr

複数アイテムの自動選択関数 identgany は、次候補アイテムの処理や選択されているアイテムの排除処理などを追加した新しい関数 IdentItems に書き換えます。
identgany は廃止しました。
選択されたアイテムのアイテム識別子は comident.h のグローバル変数を参照していましたが、ハイライトアイテムリストにセットされています。comident.h は廃止しました。
この書き換えは単純ではありません。IdentItems の結果として新しいトークンタイプ TknITM が通知されます。詳細は『プログラミングマニュアル』６章の IdentItems および２章の例を参照してください。

A.7 テンポラリポイント

以前の関数
　identpnt
以後の関数
　IdentPoint

テンポラリポイント作成関数 identpnt は、座標値トークンの処理などを追加した新しい関数 IdentPoint に書き換えます。
この書き換えは単純ではありません。IdentPoint の結果として新しいトークンタイプ TknPNT が通知されます。詳細は『プログラミングマニュアル』６章の IdentPoint および２章の例を参照してください。

A.8 グローバル変数の関数アクセス化

以前の公開グローバル変数は全て関数でアクセスする方法に変更になりました。
グローバル変数定義ヘッダーファイルはなくなります。
アクセッサ関数宣言はヘッダーファイル acadusr.h にあります。

廃止ソースコード
　comactive.h
　comidptr.h
　comident.h
　command.h
　comdimrvp.h

● comactive_ のメンバー変数に対応するアクセッサ

comactive_

メンバー変数	取得関数	設定関数
count	ActCount()	ActClear(), ActAdd()
idptrs	ActIdptrs()	

ActIdptrs() は int * を返します。このポインタを使って内容を変更してはいけません。
アクティブリストを空にするには ActClear 関数を使います。

● comidptr_ のメンバー変数に対応するアクセッサ

comidptr_

メンバー変数	取得関数	設定関数
itemtype	Itemtype()	SetItemtype()
itemclass	Itemclass()	SetItemclass()
itempic	Itempic()	SetItempic()
itemlwt	Itemlwt()	SetItemlwt()
itemfont	Itemfont()	SetItemfont()
itemrev	Itemrev()	SetItemrev()

● compick_ のメンバー変数に対応するアクセッサ

compick_

メンバー変数	取得関数	設定関数
count	IdentCount()	無し
u.idptrs[]	IdentIdptrs()	無し
u.info[]	IdentInfo() IdentInfoCount()	無し

ピック結果の詳細情報を得るには IdentInfo() 関数を使います。
IdentInfo() の返り値は IDENTINFO 構造体の配列へのポインタです。
有効な配列要素数は IdentInfoCount 関数で得られます。
構造体 IDENTINFO は comident.h の構造体 PICKINFO と同じです。
構造体 IDENTINFO の定義は acaddef.h にあります。

● command_ のメンバー変数に対応するアクセッサ

command_

メンバー変数	取得関数	設定関数
idispatch1	Idispatch(1)	無し
idriver1	Idriver(1)	無し
iformat1	Iformat(1)	無し
idispatch2	Idispatch(2)	無し
idriver2	Idriver(2)	無し
iformat2	Iformat(2)	無し
idispatch3	Idispatch(3)	無し
idriver3	Idriver(3)	無し
iformat3	Iformat(3)	無し
idispatch4	Idispatch(4)	無し
idriver4	Idriver(4)	無し
iformat4	Iformat(4)	無し
idispatch5	Idispatch(5)	無し
idriver5	Idriver(5)	無し
iformat5	Iformat(5)	無し
iccinsert	廃止（TknEXIT で通知する）	

コマンド処理関数ではコマンド番号を変更（設定）することはできません。
割り込みコマンド（レベル 3,4,5）を終了させるには次の関数を呼び出します。
コマンドレベルを間違えないように注意しましょう。

```
cmdidcla(int level);
```

修飾子をクリアするには次の関数を呼び出します。(レベル 1,3,4,5)

```
cmdmdfcla(int level);
```

● comdimrvpi_ のメンバー変数に対応するアクセッサ

comdimrvpi_

メンバー変数	取得関数	設定関数
ntextfont	Ntextfont()	SetNtextfont()
ntextdh	Ntextdh()	SetNtextdh()
ntextdv	Ntextdv()	SetNtextdv()
ntextadh	Ntextadh()	SetNtextadh()
ntextadv	Ntextadv()	SetNtextadv()
ntextslant	Ntextslant()	SetNtextslant()
ntextwaku	Ntextwaku()	SetNtextwaku()
naccudim	Naccudim()	SetNaccudim()
nadimfrt	Nadimfrt()	SetNadimfrt()
nadimfrac	Nadimfrac()	SetNadimfrac()
naccuadim	Naccuadim()	SetNaccuadim()
naccutol	Naccutol()	SetNaccutol()
ndimorient	Ndimorient()	SetNdimorient()
nddimmark	Nddimmark()	SetNddimmark()
nrdimmark	Nrdimmark()	SetNrdimmark()
narrowter	Narrowter()	SetNarrowter()
ndimdual	Ndimdual()	SetNdimdual()
ndimunit	Ndimunit()	SetNdimunit()
ndimmetrc	Ndimmetrc()	SetNdimmetrc()
ndimfracd	Ndimfracd()	SetNdimfracd()
nreftype	Nreftype()	SetNreftype()
nrefawmark	Nrefawmark()	SetNrefawmark()
mtexttolul[16]	Mtexttolul()	SetMtexttolul()
mchtolul	廃止	
mtexttollw[16]	Mtexttollw()	SetMtexttollw()
mchtollw	廃止	
mtexttolup[16]	Mtexttolup()	SetMtexttolup()
mchtolup	廃止	
nwakuacc	Nwakuacc()	SetNwakuacc()
nwakucol	Nwakucol()	SetNwakucol()

comdimrvpi_

メンバー変数	取得関数	設定関数
ntxtmirrx	Ntxtmirrx()	SetNtxtmirrx()
ntxtmirry	Ntxtmirry()	SetNtxtmirry()
ntxttate	Ntxttate()	SetNtxttate()
ndimsupz	Ndimsupz()	SetNdimsupz()
markodm	Markodm()	SetMarkodm()
nldrang	Nldrang()	SetNldrang()
ndimurep	Ndimurep()	SetNdimurep()
ndimftyp	Ndimftyp()	SetNdimftyp()
ndimfres	Ndimfres()	SetNdimfres()
ntextjsth	Ntextjsth()	SetNtextjsth()
markldrarw	Markldrarw()	SetMarkldrarw()
nkanjifont	Nkanjifont()	SetNkanjifont()
ntolsupz	Ntolsupz()	SetNtolsupz()
ntextalign	Ntextalign()	SetNtextalign()

寸法許容差のデフォルト値について

Mtexttolul(), Mtexttollw(), Mtexttolup() は char * を返します。

この文字列は '\0' 文字で終了していますので、文字数 (バイト数) を調べるには次のようにします。

```
strlen(Mtexttolul())
```

この文字列ポインタを通して内容を直接変更してはいけません。

設定は SetMtexttolul(),SetMtexttollw(),SetMtexttolup() で行います。

```
SetMtexttolul(char *value, int length)
```

● comdimrvpr_ のメンバー変数に対応するアクセッサ

comdimrvpr_

メンバー変数	取得関数	設定関数
texthight	Texthight()	SetTexthight()
textangle	Textangle()	SetTextangle()
tollhight	Tollhight()	SetTollhight()
sizemark	Sizemark()	SetSizemark()
stubsize	Stubsize()	SetStubsize()
extlgap	Extlgap()	SetExtlgap()
extlover	Extlover()	SetExtlover()
refrad	Refrad()	SetRefrad()
roundldr	Roundldr()	SetRoundldr()

comdimrvpr_

メンバー変数	取得関数	設定関数
tboxdhabax	Tboxdhabax()	SetTboxdhabax()
tboxdhabay	Tboxdhabay()	SetTboxdhabay()
sizesmark	Sizesmark()	SetSizesmark()
sizewmark	Sizewmark()	SetSizewmark()
sizeldrarw	Sizeldrarw()	SetSizeldrarw()
dimscale	Dimscale()	SetDimscale()
tol2hight	Tol2hight()	SetTol2hight()
szsmarktxt	Szsmarktxt()	SetSzsmarktxt()
szwmarktxt	Szwmarktxt()	SetSzwmarktxt()
szfcsdtmtxt	Szfcsdtmtxt()	SetSzfcsdtmtxt()
textratio	Textratio()	SetTextratio()

● comfcsrvpi_ のメンバー変数に対応するアクセッサ

comfcsrvpi_

メンバー変数	取得関数	設定関数
markfcsarw	Markfcsarw()	SetMarkfcsarw()
marksection	Marksection()	SetMarksection()
nsectline	Nsectline()	SetNsectline()
markdtmbox	Markdtmbox()	SetMarkdtmbox()
markdtmarw	Markdtmarw()	SetMarkdtmarw()
clinovtype	Clinovtype()	SetClinovtype()

● comfcsrvpr_ のメンバー変数に対応するアクセッサ

comfcsrvpr_

メンバー変数	取得関数	設定関数
sizefcsbox	Sizefcsbox()	SetSizefcsbox()
sizefcstxt	Sizefcstxt()	SetSizefcstxt()
sizefcsarw	Sizefcsarw()	SetSizefcsarw()
sizesctmark	Sizesctmark()	SetSizesctmark()
sizebdimh	Sizebdimh()	SetSizebdimh()
sizedmrext	Sizedmrext()	SetSizedmrext()
sizerefaw	Sizerefaw()	SetSizerefaw()
wakuhight	Wakuhight()	SetWakuhight()

comfcsrvpr_

メンバー変数	取得関数	設定関数
wakutsize	Wakutsize()	SetWakutsize()
wakuyokospace	Wakuyokospace()	SetWakuyokospace()
dareaoff	Dareaoff()	SetDareaoff()
sizereftxt	Sizereftxt()	SetSizereftxt()
sizescttxt	Sizescttxt()	SetSizescttxt()
clinovsize	Clinovsize()	SetClinovsize()

Appendix B　ヘッダーファイル

以下の４つのヘッダーファイルを提供しています。
これらの内容を変更してはいけません。

acaddef.h	**共通のデータ型宣言、マクロ定義**
acadprm.h	**共通定数宣言**
acadupi.h	**ユーザディスパッチャ関数のプロトタイプ宣言**
acadusr.h	**公開関数のプロトタイプ宣言**

● Version 14 までとの相違点

acadupi.h　V14 までの usrtype.h は acadupi.h に名前を変更しました。
usrtype.h にユーザ独自の関数プロトタイプなどを追加していた場合は acadupi.h 内の記述と重複する部分を usrtype.h から削除してください。
usrtype.h に何も記述を追加していない場合は usrtype.h は不要ですので削除してください。

広域変数　以下の５つのヘッダーファイルは廃止しました。各々該当する公開関数で置き換えてください。

command.h	6.1 コマンド識別番号
comident.h	6.2 アイテムのピック
comactive.h	6.8 アクティブリスト
comidptr.h	7.2 アイテム属性の現在値
comdimrvp.h	10.1 製図アイテム作成用定数

Appendix C　ユーザコマンドの登録例

以下は、ユーザコマンドの登録に必要なファイルの例です。

コマンド定義ファイル	USERCMD.MEN
メニューファイル	USEROSM.MEN
メッセージファイル	MSG90.TXT
エラーメッセージファイル	ERR90.TXT
ユーザコマンドディスパッチャ	dspatch32.c
ユーザコマンドドライバー	udr01.c
コマンド USER1 処理	ucmd01.c
コマンド USER2 処理	ucmd02.c
コマンド USER3 処理	ucmd03.c
コマンド USER4 処理	ucmd04.c

バージョン 12 よりメッセージファイルとエラーメッセージファイルが変更になりました。
今までメッセージは 300 個の制限がありましたが大幅に緩和されました。
DISPATCH32.MSG は MSG90.TXT、DISPATCH32.ERR は ERR90.TXT となります。
メッセージ番号は 90xxxxx で 9000000 から 9999999 までとなりますので、番号を振り直してください。
メッセージファイルの書式は今までと同じです。

filename : USERCMD.MEN

```
/
/ Advance CAD USER command list.
/
Command
/
/ + [dispatcher#, driver#, form#] !command_name!
/ command name :== [A-Z][A-ZO-9/_]*
/
/ [ 32, n, n] User defined command
/ [ 48, n, n] User defined command modifier
/
/ User commands
+ [32, 1, 0] !USER!
+ [32, 1, 1] !USER1!
+ [32, 1, 2] !USER2!
+ [32, 1, 3] !USER3!
```

```
+ [32, 1, 4] !USER4!
/
/ User command modifiers
+ [48, 1, 1] !OPTDUP!
+ [48, 1, 2] !OPTOFF!
/
/ End of file
```

filename : USEROSM.MEN

```
/
/ Advance CAD USER On screen menu definition
/
/ Menu [menu#,category#,screen#,colour#]
/
/ + <row#,col#> "text" !command! [menu1#,menu2#,colour#]
/ L <row#,col#> "text" !letter!  [menu1#,menu2#,colour#]
/ T <row#,col#> "text" !text!    [menu1#,menu2#,colour#]
/ N <row#,col#> "text" [menu1#,menu2#,colour#,type#,value]
/        type 0=scalar, 1=Mark#, 2=colour#, 3=key#
/
Menu [user_cmd,  1,  4,  1]
+ < 1, 1> "ユーザコマンド" !USER!   [user_cmd, user_mdf, 3]
+ < 3, 1> "ユーザ１"       !USER1!  [user_cmd, user_mdf, 4]
+ < 5, 1> "ユーザ２"       !USER2!  [user_cmd, user_mdf, 4]
+ < 7, 1> "ユーザ３"       !USER3!  [user_cmd, user_mdf, 4]
+ < 9, 1> "ユーザ４"       !USER4!  [user_cmd, user_mdf, 4]
/
Menu        [user_mdf,  1, 17,  1]
+ <10, 1> "  複製－ＯＮＮ" !OPTDUP!
+ <11, 1> "  複製－ＯＦＦ" !OPTOFF!
/
/ End of file
```

filename : MSG90.TXT

```
/ Filename : MSG90.TXT
/
/ ACAD message file for dispatcher 32
/
/        Message number        9000000 - 9999999
/
+ (9000000) "〈ＣＥ〉または〈ＢＳ〉を入力"
+ (9000001) "ボックスサイズの変更／ボックスの配置位置を指示"
+ (9000002) "移動するラインアイテムの選択／移動ベクトルの変更"
+ (9000003) "移動するストリングアイテムの選択／移動ベクトルの変更"
/
+ (9000010) "ボックスサイズ"
+ (9000011) "移動量　Ｘ"
+ (9000012) "移動量　Ｙ"
+ (9000013) "複製モードー ＯＦＦ"
+ (9000014) "複製モードー ＯＮ"
/
/ End of file
```

filename : ERR90.TXT

```
/ Filename : ERR90.TXT
```

```
/
/ ACAD error message file for dispatcher 32
/
/        Error number        9000000 - 9999999
/
+ (9000000) "ボックスサイズが有効でない。"
+ (9000001) "テンポラリポイントがえられない。"
+ (9000002) "アイテムが選択できない。"
+ (9000003) "アイテムがラインアイテムでない。"
+ (9000004) "ベクトルの長さが０である。"
+ (9000005) "選択されたアイテムがストリングアイテムでない。"
/
/ End of file
```

filename : dspatch32.c

```
/*
  Filename : dspatch32.c
  Category : User dispatcher
*/

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

/*************************************************************************
Purpose
     Example of user dispatcher
Inputs
     token  Token
Outputs
     None
*/
void dspatch32(TOKEN *token)
{
  switch (Idriver(1)) {
  case 1:
    udr01(Iformat(1), token);
    break;
#if 0
  case 2:
    udr02(Iformat(1), token);
    break;
  case 3:
    udr03(Iformat(1), token);
    break;
#endif
  default:
    break;
  }
} /* dspatch32 */
```

filename : udr01.c

```
#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"
```

```
/************************************************************************
  Purpose
      Example of user driver #1
  Inputs
      token   Token
  Outputs
      None
*/
void udr01(int form, TOKEN *token)
{
  switch (form) {
  case 1:
    ucmd01(token);
    break;
  case 2:
    ucmd02(token);
    break;
  case 3:
    ucmd03(token);
    break;
  case 4:
    ucmd04(token);
    break;
  default:
    break;
  }
} /* udr01 */
```

filename : ucmd01.c

```
#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

/************************************************************************
  Purpose
      USER1 command handler
      Dispatcher      32
      Driver           1
      Form             1
  Command Syntax
      Create two line items and one string item.
      USER1 <CE>
  Inputs
      token   Token
  Outputs
       None
*/
void ucmd01(TOKEN *token)
{
  static DPOINT dpnts[] = { { 100.0,  0.0}, {  0.0,  100.0},
                            {-100.0,  0.0}, {  0.0, -100.0},
                            { 100.0,  0.0} };
  DPOINT p;
  ItmSR sr;
  int j;
  int ier = 0;
```

```
  switch (token->typ) {
  case TknCMD: /* Command */
    TmpgWrtoDB(0);
    TmpgInit();
    /* Create Two line items. */
    SetItemtype(ITMLINE);
    for (j = 0; j < 2; ++j) {
      TmpgOpen1(0, 0);
      SETITMSR(sr, SITSTART, 3, 2, 0, 0, &dpnts[j]);
      TmpgAddSr(&sr);
      SETITMSR(sr, SITLINE,  3, 2, 0, 0, &dpnts[j+1]);
      TmpgAddSr(&sr);
      TmpgClose(1);
    }
    /* Create the string item. */
    SetItemtype(ITMSTRING);
    TmpgOpen1(0, 0);
    SETITMSR(sr, SITSTART, 3, 2, 0, 0, &dpnts[2]);
    TmpgAddSr(&sr);
    for (j = 3; J < 5; ++j) {
      SETITMSR(sr, SITLINE, 3, 2, 0, 0, &dpnts[j]);
      TmpgAddSr(&sr);
    }
    TmpgClose(1);
    break;
  case TknBSP: /* Back Space */
    if (Tmpgback(0, &p))
      ier = 100;
    break;
  case TknEOC:  /* CE */
  case TknEXIT: /* Exit */
    TmpgWrtoDB(0);
    TmpgInit();
    if (token->typ == TknEXIT)
      return;
    break;
  default:
    ier = 100
    break;
  }
  if (ier)
    Errorb(); /* Ring the error bell. */
  Opmsgcode(1, 9000000); /* Display the operation message. */
}
```

filename : ucmd02.c

```
#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

/*********************************************************************
  Purpose
    USER2 command handler
    Dispatcher      32
    Driver           1
    Form             2
  Command Syntax
    Create the box string item at the specified position.
```

```
    USER2 [[s] P]+ <CE>
            s       Box size.
            P       Lower-left corner of the box.
  Inputs
    token   Token
  Outputs
    None
*/
void ucmd02(TOKEN *token)
{
  static double boxsize;
  ItmSR sr;
  DPOINT boxcnr[4], p;
  int ier = 0;
  int j;

  switch (token->typ) {
  case TknCMD: /* Command */
    TmpgWrtoDB(0);
    TmpgInit();
    /* Initialize the default box size. */
    boxsize = 10.0;
    /* Display the current box size on the command status area. */
    Mesageras(MSGZONE, 0, 0);
    Mesagdisp(MZONECOLOR1, 1, 1, 9000010, 3, &boxsize);
    break;
  case TknCOD: /* Coordinate */
  case TknDIG: /* Digitize */
  case TknPNT: /* Temporary Point */
    if (identpoint(token, &p) == IDENT_SUCCESS) {
      /* Set the corner points of the box. */
      boxcnr[0].x = p.x + boxsize;
      boxcnr[0].y = p.y;
      boxcnr[1].x = p.x + boxsize;
      boxcnr[1].y = p.y + boxsize;
      boxcnr[2].x = p.x;
      boxcnr[2].y = p.y + boxsize;
      boxcnr[3] = p;
      /* Create the temporary box string item. */
      SetItemtype(ITMSTRING);
      TmpgOpen1(0, 0);
      SETITMSR(sr, SITSTART, 3, 2, 0, 0, &boxcnr[3]);
      TmpgAddSr(&sr);
      for (j = 0; j < 4; ++j) {
        SETITMSR(sr, SITLINE,  3, 2, 0, 0, &boxcnr[j]);
        TmpgAddSr(&sr);
      }
      TmpgClose(1);
      /* Change the class number of the last temporary item. */
      Tmpgatr(0, 3, 4);
    }
    break;
  case TknSCL: /* Scalar */
    if (sc < 1.0) {
      ier = 1;
      break;
    }
    /* Display the current box size on the command status area. */
    boxsize = sc;
    Mesageras(MSGZONE, 0, 0)
    Mesagdisp(MZONECOLOR1, 1, 1, 9000010, 3, &boxsize);
    break;
```

```
  case TknBSP: /* Back Space */
    /* Delete the last temporary item. */
    if (Tmpgback(0, &p))
      ier = 100;
    break;
  case TknEOC:  /* CE */
  case TknEXIT: /* Exit */
    TmpgWrtoDB(0);
    TmpgInit();
    if (token->typ == TknEXIT)
      return;
    break;
  default:
    ier = 100;
    break;
  }
  /* Handle the error. */
  if (ier) {
    if ( 0 < ier && ier < 100) {
      Errorcode(ier + 9000000 - 1);
    } else {
      Errorb(); /* Ring the error bell */
    }
  }
  /* Display the operation message. */
  Opmsgcode(1, 9000001);
}
```

filename : ucmd03.c

```
#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

/*****************************************************************************
  Purpose
    USER3 command handler
    Dispatcher      32
    Driver           1
    Form             3
  Command Syntax
    Move the selected line item by the specified vector.
    USER3 [[vec] IS]+ <CE>
            vec       Vector to move line items.
            IS        Ident the line item to be moved.
  Inputs
    token   Token
  Outputs
    None
*/
void ucmd03(TOKEN *token)
{
  static DPOINT vec;

  ItmAtr itmatr;
  ItmSR sr;
  int idptr;
  int isno;
  short idata[1024];
```

```
FPOINT fp;
DPOINT dp;

int ier = 0;

switch (token->typ) {
case TknCMD: /* Command */
  TmpgWrtoDB(0);
  TmpgInit();
  vec.x = 10.0;
  vec.y = 10.0;
  /* Display the current vector on the command status area. */
  Mesageras(MSGZONE, 0, 0);
  Mesagdisp(MZONECOLOR1, 1, 1, 9000011, 3, &vec.x);
  Mesagdisp(MZONECOLOR1, 2, 1, 9000012, 3, &vec.y);
  break;
case TknDIG: /* Digitize */
  TmpgWrtoDB(0);
  TmpgInit();
  /* Ident the item. */
  if ((idptr = IdentItem(CMDLVL, token, 1)) <= 0) {
    ier = 3;
    break;
  }
  /* Check the item type of the idented item. */
  if (Dbvriatr(idptr, &itmatr) || itmatr.typ != ITMLINE) {
    ier = 4;
    break;
  }
  /* Move the item by the current vector. */
  TmpgOpen1(0, idptr);
  DbUrdopen(1, idptr);
  while (DbUrdhead(1, &isno, &sr) == 0 && sr.typ != SITEOI) {
    switch (sr.typ) {
    case SITSTART:
    case SITLINE:
      if (sr.mod == 3) {
        DbUrddata(1, sr.cnt, &dp);
        dp.x += vec.x;
        dp.y += vec.y;
        sr.dat = &dp;
        TmpgAddSr(&sr);
      } else {
        DbUrddata(1, sr.cnt, &fp);
        fp.x += vec.x;
        fp.y += vec.y;
        sr.dat = &fp;
        TmpgAddSr(&sr);
      }
      break;
    default:
      DbUrddata(1, sr.cnt, idata);
      sr.dat = idata;
      TmpgAddSr(&sr);
      break;
    }
  } /* End of while loop */
  DbUrdclose(1);
  TmpgClose(1);
  break;
case TknVEC: /* Vector */
  if (pnt->x == 0.0 && pnt->y == 0.0) {
```

```
      ier = 5;
      break;
    }
    vec = *pnt;
    /* Display the current vector on the command status area. */
    Mesageras(MSGZONE, 0, 0);
    Mesagdisp(MZONECOLOR1, 1, 1, 9000011, 3, &vec.x);
    Mesagdisp(MZONECOLOR1, 2, 1, 9000012, 3, &vec.y);
    break;
  case TknBSP: /* Back Space */
    /* Delete the last temporary item. */
    if (Tmpgback(0, &dp))
      ier = 100;
      break;
  case TknEOC:  /* CE */
  case TknEXIT: /* Exit */
    TmpgWrtoDB(0);
    TmpgInit();
    if (token->typ == TknEXIT)
      return;
      break;
  default:
    ier = 100;
    break;
  }

  /* Handle the error. */
  if (ier) {
    if (0 < ier && ier < 100) {
      Errorcode(ier + 9000000 - 1);
    } else {
      Errorb(); /* Ring the error bell */
    }
  }

  /* Display the operation message. */
  Opmsgcode(1, 9000002);
}
```

filename : ucmd04.c

```
#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

/*****************************************************************
  Purpose
    USER4 command handler
         Dispatcher      32
         Driver           1
         Form             4
    Command Syntax
         Move the selected string items by the specified vector.
         USER4 [[{OPTDUP, OPTOFF}] [vec] ISauto]+
               OPTDUP        Duplication mode on.
               OPTOFF        Duplication mode off.
               vec           Vector to move string items.
               ISauto        Ident the string items to be moved.
    Inputs
```

```
        token   Token
    Outputs
        None
*/
void ucmd04(TOKEN *token)
{
  static int keydup;
  static DPOINT vec;

  int i, nstritm;
  int ier = 0;

  switch (token->typ) {
  case TknCMD: /* Command */
    TmpgWrtoDB(0);
    TmpgInit();
    vec.x = 10.0;
    vec.y = 10.0;
    keydup = 0;
    /* Display the current parameters on the command status area. */
    Mesageras(MSGZONE, 0, 0);
    Mesagdisp(MZONECOLOR1, 1, 1, 9000011, 3, &vec.x);
    Mesagdisp(MZONECOLOR1, 2, 1, 9000012, 3, &vec.y);
    Mesagdisp(MZONECOLOR2, 3, 1, (keydup == 0) ? 9000013 : 9000014, 0, (void *)NULL);
    break;
  case TknMDF: /* Modifier */
    if (Idispatch(2) == 48 && Idriver(2) == 1 && (Iformat(2) == 1 || Iformat(2) == 2)) {
      /* OPTDUP [48, 1, 1] Duplication mode. */
      /* OPTOFF [48, 1, 2] No duplication mode. */
      keydup = (Iformat(2) == 1) ? 1 : 0;
      Mesageras(MSGZONE, 3, 0);
      Mesagdisp(MZONECOLOR2, 3, 1, (keydup == 0) ? 9000013 : 9000014, 0, (void *)NULL);
    } else {
      ier = 100;
    }
    cmdmdfcla(1);
    break;
  case TknMZN: /* Message Zone */
    if (pnt->x == 1 && pnt->y == 3) { /* Change the duplication mode */
      keydup = (keydup == 0) ? 1 : 0;
      Mesageras(MSGZONE, 3, 0);
      Mesagdisp(MZONECOLOR2, 3, 1, (keydup == 0) ? 9000013 : 9000014, 0, (void *)NULL);
    } else {
      ier = 100;
    }
    break;
  case TknDIG: /* Digitize */
  case TknITM: /* Item Selection */
    /* Ident the items. */
    if (IdentItems(1, token, 0) != IDENT_SUCCESS)
      break;
    /* Move the items. */
    TmpgWrtoDB(0);
    TmpgInit();
    nstritm = 0; /* The number of the string items. */
    for (i = 0; i < rptitmnum(1); ++i) {
      short idata[1024];
      FPOINT fp;
      DPOINT dp;
      ItmAtr itmatr;
      ItmSr sr;
      int *idptrs = rptitmptr(1);
```

```
    int idptr = idptrs[i];

    /* Check the item type of the idented item. */
    if (Dbvriatr(idptr, &itmatr) || itmatr.typ != ITMSTRING)
      continue;
    /* Move the item by the current vector. */
    TmpgOpen1(keydup, idptr);
    DbUrdopen(1, idptr);
    while (DbUrdhead(1, &isno, &sr) == 0 && sr.typ != SITEOI) {
      switch (sr.typ) {
      case SITSTART:
      case SITLINE:
        if (sr.mod == 3) {
          DbUrddata(1, sr.cnt, &dp);
          dp.x += vec.x;
          dp.y += vec.y;
          sr.dat = &dp;
          TmpgAddSr(&sr);
        } else {
          DbUrddata(1, sr.cnt, &fp);
          fp.x += vec.x;
          fp.y += vec.y;
          sr.dat = &fp;
          TmpgAddSr(&sr);
        }
        break;
      default:
        DbUrddata(1, sr.cnt, idata);
        sr.dat = idata;
        TmpgAddSr(&sr);
        break;
      }
    } /* End of while loop */
    DbUrdclose(1);
    TmpgClose(0);
    ++nstritm;
  } /* End of for loop */
  if (nstritm <= 0)
    ier = 6;
  TmpgWrtoDB(0); /* Write the all temporary item to the Data Base. */
  TmpgInit();
  token->typ = TknCMD;
  (void)IdentItems(1, token, 0);
  break;
case TknVEC: /* Vector */
  if (pnt->x == 0.0 && pnt->y == 0.0) {
    ier = 5;
    break;
  }
  vec = *pnt;
  /* Display the current vector on the command status area. */
  Mesageras(MSGZONE, 1, 0);
  Mesagdisp(MZONECOLOR1, 1, 1, 9000011, 3, &vec.x);
  Mesageras(MSGZONE, 2, 0);
  Mesagdisp(MZONECOLOR1, 2, 1, 9000012, 3, &vec.y);
  break;
case TknEOC:  /* CE */
  break;
case TknEXIT: /* Exit */
  return;
default:
  ier = 100;
```

```
      break;
    } /* End of switch (token->typ) */

    /* Handle the error. */
    if (ier) {
      if (0 < ier && ier < 100) {
        Errorcode(ier + 9000000 - 1);
      } else {
        Errorb(); /* Ring the error bell */
      }
    }
    /* Display the operation message. */
    Opmsgcode(1, 9000003)
  }
```

Appendix D　X プロパティ (UNIX 版)

D.1　概要

Advance CAD とユーザアプリケーション間の通信手段として、X window のプロパティを利用することができます。
この機能を使用すると、ユーザアプリケーションから Advance CAD へコマンドやパラメータを送信することによってユーザアプリケーションから Advance CAD をコントロールすることができます。
また逆に、Advance CAD からユーザアプリケーションへデータを送信することもできます。
この機能を使用するには、X Window の知識が必要です。

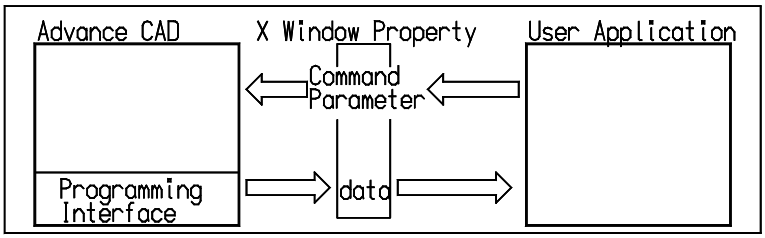
プロパティを介した通信は、次のような手順で行います。

(1)　送信側アプリケーションがプロパティの内容を更新する。いいかえれば、プロパティにデータを書き込む。
(2)　X window system は、プロパティが変更されたことを通知する。
(3)　受信側はプロパティを読む。

- X window プロパティ

  プロパティは X server のルートウインドウに作ります。プロパティを識別するためにプロパティに名前を付けます。通信しようとするアプリケーション同士はお互いに使用するプロパティの名前を取り決めておかなければなりません。次のプロパティ名は Acadhelp で使用しますので、使用しないでください。
  __ACAD_COM_NAM
  __ACAD_HELP_EXIST

  構成図



D.2　ユーザアプリケーションから Advance CAD への送信

- X window プロパティ名

Advance CAD 起動時にコマンド引数として与えます。
%acad -nPropertyName
PropertyName : 受信用プロパティ名

送信側のアプリケーションは、このプロパティ名に対して送信します。

- **プロパティ**

Advance CAD が受信できるデータタイプは、文字列 (XA_STRING) とし、一度に受信できる文字数は 80 バイト以下です。80 バイトを越えるとすべて無視されます。文字列の内容は Advance CAD のコマンドストリームでなければなりません。

Advance CAD は指定された名前を持つプロパティが更新されると、そのプロパティを読み取ります。そして、それがキーボードから入力されたコマンドストリームとみなして解釈、処理をします。従ってこの場合、Advance CAD 側はなんら変更する必要はありません。

送信側アプリケーションは作成しなければなりません。例としてサンプルプログラム 1 を参考にしてください。
サンプルアプリケーション xvsend はコマンドリストファイルを読み込み、リストボックスに表示します。マウスで選択した行を Advance CAD へ送信するプログラムです。

D.3 Advance CAD からユーザアプリケーションへの送信

プロパティ名、プロパティのデータタイプ、内容などは任意です。アプリケーションの目的に応じて決めます。
Advance CAD 側ではプログラミングインタフェースを使って、送信する機能を追加します。これに対応した受信側アプリケーションを作ります。例としてサンプルプログラム 2 を付けましたので、参考にしてください。

Advance CAD 側
ソースコード　　dspatch32.c, userprop.c
処理内容　　レジスタの内容、マクロ変数の内容を文字列に変換してプロパティを更新する。
受信側アプリケーション
ソースコード　　rec.c
処理内容　　プロパティの内容を読み出し、表示する。

D.4 サンプルプログラム 1

filename : xvsend.c

```
/*
  File : xvsend.c
  Advance CAD X Property Sample Program
  Solaris 2.X, Use : Motif, Date: 04/17/2002

    cc -xarch=generic64 -I/usr/dt/include -I/usr/openwin/include -o xvsend xvsend.c
       -L/usr/dt/lib -lMrm -lXm -L/usr/openwin/lib -lXt -lX11

  Advance CAD Execution Option
        acad =1125x863+0+0 -n__ACADPROP_TEST
```

```
*/

#include <stdio.h>
#include <string.h>
#include <X11/StringDefs.h>
#include <X11/Intrinsic.h>
#include <Xm/Xm.h>
#include <Xm/Form.h>
#include <Xm/RowColumn.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/Text.h>
#include <Xm/PushB.h>
#include <X11/Xatom.h>

static char *ACADAtom = "__ACADPROP_TEST";
static Atom Acadatm;
static Display *Dpy;

static Widget frame;       /* Toplevel */
static Widget panel;       /* Container, XmForm */
static Widget panel_label; /* Label of container, XmLabel */
static Widget List1;       /* Contents of the command file, XmScrolledList */
static Widget dir_label;   /* Label of directory, XmLabel */
static Widget Directory;   /* Directory name of the command file, XmText */
static Widget file_label;  /* Label of filename, XmLabel */
static Widget Filename;    /* File name of the command file, XmText */
static Widget button_panel;/* Container for button, XmRowColumn */
static Widget wbutton[3];  /* Open, send and quit, XmPushButton */
static Widget textsw;      /* Command log to be issued, XmScrolledText */

static char DirName[256] ;
static char FilName[256] = "ACADSEND.DAT";

static char OpenFilName[512];
typedef struct {
  char  DspName[40];
  char  SndStr[256];
} LstSnd;

#define MAXLISTSND 300
static LstSnd SendList[MAXLISTSND];
static int SendListCnt = 0;
static int List1Sel = -1;
static char msgbuf[1024];

static int GetWidth(int num_char)
{
  XFontStruct* font = XLoadQueryFont(XtDisplay(frame), "fixed");
  return (font != (XFontStruct*)NULL)
          ? XTextWidth(font, " ", 1) * num_char : 0;
}

static void textsw_insert(Widget textsw, char* msgbuf, size_t length)
{
  static XmTextPosition text_position = 0;
  XmTextInsert(textsw, text_position, msgbuf);
  text_position += strlen(msgbuf);
  XtVaSetValues(textsw, XmNcursorPosition, text_position, NULL);
  XmTextShowPosition(textsw, text_position);
}
```

```
/* Check Command Stream */
static void ChkListCmd(char *dp, char *sp)
{
  while (*sp)
    *dp++ = *sp++;
  *dp = '¥0';
} /* ChkListCmd */

/* Set Command Stream from File */
static void SetList(FILE *fp)
{
  int n;
  Arg args[10];

  /* Clear List */
  {
    XmString* items = (XmString*)NULL;
    int itemCount   = 0;
    n = 0;
    XtSetArg(args[n], XmNitems,     &items);     ++n;
    XtSetArg(args[n], XmNitemCount, &itemCount); ++n;
    XtGetValues(List1, args, n);
    if (items != (XmString*)NULL) {
      XtFree(items);
      /* n = 0; */
      /* XtSetArg(args[n], XmNitems,     NULL); ++n; */
      /* XtSetArg(args[n], XmNitemCount, 0);    ++n; */
      /* XtSetValues(List1, args, n); */
    }
  }

  /* Read File */
  SendListCnt = 0;
  for (;;) {
    int slen;
    char *tabp;
    if (fgets(msgbuf, sizeof(msgbuf), fp) == (char *)NULL)
      break;
    slen = strlen(msgbuf);
    if (msgbuf[slen-1] == '¥n')
      msgbuf[slen-1] = '¥0';
    slen = strlen(msgbuf);
    if (slen == 0)
      continue;
    tabp = strchr(msgbuf, '¥t');
    if (tabp == (char *)NULL)
      continue;
    *tabp = '¥0';
    strcpy(SendList[SendListCnt].DspName, msgbuf);
    ChkListCmd(SendList[SendListCnt].SndStr, tabp+1);
    ++SendListCnt;
    if (SendListCnt >= MAXLISTSND)
      break;
  }

  /* Set List */
  if (SendListCnt > 0) {
    int i;
    XmString* items = (XmString*)XtMalloc(sizeof(XmString) * SendListCnt);
    for (i = 0; i < SendListCnt; ++i) {
      items[i] = XmStringCreate(SendList[i].DspName, XmSTRING_DEFAULT_CHARSET);
    }
```

```
    n = 0;
    XtSetArg(args[n], XmNitems,     items);       ++n;
    XtSetArg(args[n], XmNitemCount, SendListCnt); ++n;
    XtSetValues(List1, args, n);
  }

} /* SetList */

/* File Command Open and Read */
static void proc_Open(Widget w,
                      caddr_t client_data, XmAnyCallbackStruct* call_data)
{
  int slen;
  char *pdirname, *pfilename;
  FILE *fp;

  pdirname  = (char *)XmTextGetString(Directory);
  pfilename = (char *)XmTextGetString(Filename);

  strcpy(OpenFilName, pdirname);
  slen = strlen(OpenFilName);
  if (slen == 0) {
    strcpy(OpenFilName, "./");
  } else if ( OpenFilName[slen-1] != '/') {
     strcat(OpenFilName, "/");
  }
  strcat(OpenFilName, pfilename);
  slen = strlen(OpenFilName);

  if (pdirname  != (char*)NULL) XtFree(pdirname);
  if (pfilename != (char*)NULL) XtFree(pfilename);

  if (slen == 0){
    sprintf(msgbuf, "please Set File Name ¥n");
    textsw_insert(textsw, msgbuf, strlen(msgbuf));
    return;
  }
  fp = fopen(OpenFilName, "r");
  if (fp == (FILE *)NULL) {
    sprintf(msgbuf, "File Open Error : %s¥n", OpenFilName);
    textsw_insert(textsw, msgbuf, strlen(msgbuf));
    return;
  } else {
    sprintf(msgbuf,"Open File : %s¥n",OpenFilName);
    textsw_insert(textsw, msgbuf, strlen(msgbuf));
  }
  SetList(fp);
  fclose(fp);
} /* proc_Open */

/* Send Command Stream to Advance CAD */
static void SendToACAD(char *str)
{
   XChangeProperty(Dpy, DefaultRootWindow(Dpy), Acadatm, XA_STRING, 8,
                   PropModeReplace, str, strlen(str));
} /* SendToACAD */

static void proc_Send(Widget w,
                      caddr_t client_data, XmAnyCallbackStruct* call_data)
{
  if (List1Sel < 0 || List1Sel >= SendListCnt)
    return;
```

```
  sprintf(msgbuf, "Send: %d: %s¥n", List1Sel, SendList[List1Sel].SndStr);
  textsw_insert(textsw, msgbuf, strlen(msgbuf));
  SendToACAD(SendList[List1Sel].SndStr);
} /* proc_Send */

/* Quit Function */
static void proc_Quit(Widget w,
                      caddr_t client_data, XmAnyCallbackStruct* call_data)
{
  /* textsw_reset(textsw, 0, 0); */
  /* xv_destroy_safe(frame); */
  XtDestroyWidget(frame);
  XtCloseDisplay(XtDisplay(frame));
  exit(0);
}

static void proc_List1(Widget w,
                      caddr_t client_data, XmListCallbackStruct* call_data)
{
  if (call_data->reason == XmCR_BROWSE_SELECT)
    List1Sel = call_data->item_position - 1;
  else
    List1Sel = -1;
} /* proc_List1 */

/* Create GUI by Xview */
static void CreateSendUI(void)
{
  int row, col;
  int n;
  Arg args[16];
  XmString compound;

  /* Create Panel */
  panel = XtCreateManagedWidget("panel", xmFormWidgetClass, frame, NULL, 0);

  /* Title Message */
  row = 1;
  col = 25;
  n   = 0;
  compound = XmStringCreate("Advance CAD Property Test",
                            XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[n], XmNtopAttachment,   XmATTACH_POSITION); ++n;
  XtSetArg(args[n], XmNtopPosition,     row);               ++n;
  XtSetArg(args[n], XmNleftAttachment, XmATTACH_POSITION); ++n;
  XtSetArg(args[n], XmNleftPosition,    col);               ++n;
  XtSetArg(args[n], XmNlabelString,     compound);          ++n;
  panel_label = XtCreateManagedWidget("panel_label",
                                      xmLabelWidgetClass, panel, args, n);

  /* Create List */
  row = 6;
  col = 25;
  n   = 0;
  XtSetArg(args[n], XmNtopAttachment,     XmATTACH_POSITION); ++n;
  XtSetArg(args[n], XmNtopPosition,       row);               ++n;
  XtSetArg(args[n], XmNleftAttachment,    XmATTACH_POSITION); ++n;
  XtSetArg(args[n], XmNleftPosition,      col);               ++n;
  XtSetArg(args[n], XmNwidth,             GetWidth(40));      ++n;
  XtSetArg(args[n], XmNvisibleItemCount, 9);                  ++n;
  XtSetArg(args[n], XmNscrollingPolicy,   XmAUTOMATIC);       ++n;
  List1 = XmCreateScrolledList(panel, "List1", args, n);
```

```
XtAddCallback(List1, XmNbrowseSelectionCallback, proc_List1, NULL);
XtManageChild(List1);

/* Directory label */
row = 48;
col = 1;
n   = 0;
compound = XmStringCreate("Directory : ", XmSTRING_DEFAULT_CHARSET);
XtSetArg(args[n], XmNtopAttachment,  XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNtopPosition,    row);               ++n;
XtSetArg(args[n], XmNleftAttachment, XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNleftPosition,   col);               ++n;
XtSetArg(args[n], XmNlabelString,    compound);          ++n;
dir_label = XtCreateManagedWidget("dir_label",
                                  xmLabelWidgetClass, panel, args, n);

/* Directory */
row = 47;
col = 25;
n   = 0;
XtSetArg(args[n], XmNtopAttachment,  XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNtopPosition,    row);               ++n;
XtSetArg(args[n], XmNleftAttachment, XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNleftPosition,   col);               ++n;
XtSetArg(args[n], XmNvalue,          DirName);           ++n;
XtSetArg(args[n], XmNmaxLength,      64);                ++n;
XtSetArg(args[n], XmNcolumns,        40);                ++n;
Directory = XtCreateManagedWidget("Directory",
                                  xmTextWidgetClass, panel, args, n);

/* Filename label */
row = 58;
col = 1;
n   = 0;
compound = XmStringCreate("File      : ", XmSTRING_DEFAULT_CHARSET);
XtSetArg(args[n], XmNtopAttachment,  XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNtopPosition,    row);               ++n;
XtSetArg(args[n], XmNleftAttachment, XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNleftPosition,   col);               ++n;
XtSetArg(args[n], XmNlabelString,    compound);          ++n;
file_label = XtCreateManagedWidget("dir_label",
                                   xmLabelWidgetClass, panel, args, n);

/* Filename */
row = 57;
col = 25;
n   = 0;
XtSetArg(args[n], XmNtopAttachment,  XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNtopPosition,    row);               ++n;
XtSetArg(args[n], XmNleftAttachment, XmATTACH_POSITION); ++n;
XtSetArg(args[n], XmNleftPosition,   col);               ++n;
XtSetArg(args[n], XmNvalue,          FilName);           ++n;
XtSetArg(args[n], XmNmaxLength,      64);                ++n;
XtSetArg(args[n], XmNcolumns,        40);                ++n;
Filename = XtCreateManagedWidget("Filename",
                                  xmTextWidgetClass, panel, args, n);

/* Create Button */
{
  int position[] = {1, 45, 90};
  char* button[] = { " Open ", " Send ", " Quit " };
  XtCallbackProc button_callback[] = { proc_Open, proc_Send, proc_Quit };
```

```
    row = 65;
    for (col = 0; col < XtNumber(wbutton); ++col) {
      n = 0;
      XtSetArg(args[n], XmNtopAttachment,  XmATTACH_POSITION); ++n;
      XtSetArg(args[n], XmNtopPosition,    row);               ++n;
      XtSetArg(args[n], XmNleftAttachment, XmATTACH_POSITION); ++n;
      XtSetArg(args[n], XmNleftPosition,   position[col]);     ++n;
      wbutton[col] =
        XtCreateManagedWidget(button[col],
                              xmPushButtonWidgetClass, panel, args, n);
      XtAddCallback(wbutton[col],
                    XmNactivateCallback, button_callback[col], NULL);
    }
  }

  /* Message window */
  row = 72;
  col = 0;
  n   = 0;
  XtSetArg(args[n], XmNtopAttachment,    XmATTACH_POSITION); ++n;
  XtSetArg(args[n], XmNtopPosition,      row);               ++n;
  XtSetArg(args[n], XmNleftAttachment,   XmATTACH_POSITION); ++n;
  XtSetArg(args[n], XmNleftPosition,     col);               ++n;
  XtSetArg(args[n], XmNrows,             8);                 ++n;
  XtSetArg(args[n], XmNcolumns,          80);                ++n;
  XtSetArg(args[n], XmNeditable,         FALSE);             ++n;
  XtSetArg(args[n], XmNeditMode,         XmMULTI_LINE_EDIT); ++n;
  XtSetArg(args[n], XmNscrollingPolicy,  XmAUTOMATIC);       ++n;
  XtSetArg(args[n], XmNwordWrap,         True);              ++n;
  XtSetArg(args[n], XmNscrollHorizontal, False);             ++n;
  XtSetArg(args[n], XmNblinkRate,        0);                 ++n;
  XtSetArg(args[n], XmNautoShowCursorPosition, True);        ++n;
  XtSetArg(args[n], XmNcursorPositionVisible, False);        ++n;
  textsw = XmCreateScrolledText(panel, "textsw", args, n);
  XtManageChild(textsw);

} /* CreateSendUI */

/* Main  */
int main(int argc, char **argv)
{
  /* Initialize Xview */
  frame = XtInitialize(argv[0], "Xvsend", NULL, 0, &argc, argv);

  /* Create GUI compornents */
  CreateSendUI();

  /* Set Atom */
  Dpy     = XtDisplay(frame);
  Acadatm = XInternAtom(Dpy, ACADAtom, False);

  /* Open & read the default command file. */
  proc_Open((Widget)NULL, (caddr_t)0, (XmAnyCallbackStruct*)NULL);

  /* Realize widget */
  XtRealizeWidget(frame);

  /* Set window name */
  {
    char* window_name = "Sample Send";
    XTextProperty text;
    XStringListToTextProperty(&window_name, 1, &text);
```

```
    XSetWMName(XtDisplay(frame), XtWindow(frame), &text);
  }

  /* Disable to resize */
  {
    Dimension width;
    Dimension height;
    XtVaGetValues(frame, XmNwidth, &width, XmNheight, &height, NULL);
    {
      XSizeHints hints;
      hints.min_width  = hints.max_width  = width;
      hints.min_height = hints.max_height = height;
      hints.flags = PMinSize | PMaxSize;
      XSetNormalHints(XtDisplay(frame), XtWindow(frame), &hints);
    }
  }

  /* Xview Loop */
  XtMainLoop();

  return 0;
}
```

D.5 サンプルプログラム２

filename ： dspatch32.c

```
/*
  Filename : dspatch32.c
  Category : User dispatcher
*/

#include "acaddef.h"
#include "acadprm.h"
#include "acadusr.h"
#include "acadupi.h"

/*****************************************************************************
   Purpose
        Example of user dispatcher
   Inputs
        token   Token
   Outputs
        None
*/
void dspatch32(TOKEN *token)
{
  switch (Idriver(1)) {
  case 2:
    udr02(Iformat(1), token);
    break;
  default:
    break;
  }
} /* dspatch32 */

/*****************************************************************************
```

```
   Purpose
        Example of user driver #2
   Inputs
        form    Form number
        token   Token
   Outputs
        None
*/
void udr02(int form, TOKEN *token)
{
  switch (form) {
  case 1:
    ucmd21(token);
    break;
  default:
    break;
  }
} /* udr02 */

/*************************************************************************
  Purpose
        Process UREG Command
*/
void ucmd21(TOKEN *token)
{
  static int ipropini = 0;
  int ier = 0;

  switch (token->typ) {
  case TknCMD: /* Command */
   if (ipropini == 0) {
     ier = xpropopen();
     if (ier == 0)
       ipropini = 1;
   }
   break;

  case TknTXT: /* Text */
    {
      int type;
      double dval;
      char cstr[81];
      type = mcrcalc(token->txt, strlen(token->txt), &dval, cstr, sizeof(cstr));
      if (type == 3 || type == 4) {
        if (type == 3)
           sprintf(cstr, "%g", dval);
        Mesageras(MSGZONE, 2, 1);
        Mesagdisp(MZONECOLOR1, 2, 1, 0, 10 * strlen(cstr), cstr);
        if (ipropini)
          xpropsend(cstr);
      } else if (type < 0) {
        ier = 2;
      }
    }
    break;

  case TknEOC: /* CE */
    break;

  case TknEXIT: /* Exit */
    return;
```

```
  default:
    ier = 100;
    break;
  } /* End of switch */

  if (ier)
    Errorb();
} /* ucmd21 */
```

filename : userprop.c

```
/*
  Filename : userprop.c
  X Property Sample Code
  Use with Advance CAD Programming Interface */

#include    <stdio.h>
#include    <X11/Xlib.h>
#include    <X11/Xutil.h>
#include    <X11/keysym.h>
#include    <X11/Xatom.h>
#include    <X11/Xresource.h>

static     Display        *Dpy;         /* X Display */
static     Window         Root;         /* X Root Window */
static     Atom           atom_prop;    /* Property Atom */
static     char *MyProp = "__ACADPROP_USERSEND"; /* Sample Name */

/* Open Property for user Application */
int xpropopen(void)
{
  if ((Dpy = XOpenDisplay ("")) == NULL)
    return 1;
  Root = DefaultRootWindow(Dpy);
  atom_prop = XInternAtom(Dpy, MyProp, False);
  return 0;
} /* xpropopen */

/* Send Data from Advance CAD to user Application */
void xpropsend(char *send)
{
  XChangeProperty(Dpy, Root, atom_prop, XA_STRING,  8, PropModeReplace,
                  send, strlen(send));
  XFlush(Dpy);
} /* xpropsend */
```

filename : rec.c

```
/*
  Filename: rec.c
  Advance CAD X Property Sample Program
    cc -xarch=generic64 -DOW_I18N -I${OPENWINHOME}/include -o rec rec.c
       -L${OPENWINHOME}/lib -lX11
*/

#include    <stdio.h>
#include    <X11/Xlib.h>
#include    <X11/Xutil.h>
#include    <X11/keysym.h>
```

```
#include    <X11/Xatom.h>
#include    <X11/Xresource.h>

Display        *Dpy;
Window          Win,Root;
int             Scr;
GC      kanjigc,asciigc;
XFontStruct             *kanjifont;
XFontStruct             *asciifont;

/* Rec Main Program */
main(int ac, char *av)
{
  XSizeHints      SizeHints;
  unsigned long   Fg,Bg;
  XEvent          Event;
  Atom            atom_prop,atom_delete;
  Atom            atom_type;
  KeySym          key;
  char            ebuf[10];
  int             count;
  GC              Gc;
  int             done;
  XWMHints        hints;
  Atom            ret_atom;
  int             ret_format;
  unsigned long   ret_len, ret_after;
  unsigned char  *ret_prop;

  if ((Dpy = XOpenDisplay ("")) == NULL) {
    fprintf(stderr, "could not connect to [%s].¥n", XDisplayName(NULL));
    exit (1);
  }

  Scr = DefaultScreen (Dpy);
  Root = DefaultRootWindow (Dpy);
  Bg = BlackPixel (Dpy, Scr);
  Fg = WhitePixel (Dpy, Scr);

  SizeHints.x = 100;
  SizeHints.y = 100;
  SizeHints.width = 100;
  SizeHints.height = 100;
  SizeHints.flags = PPosition|PSize;

  Win = XCreateSimpleWindow (Dpy, Root,
                  SizeHints.x, SizeHints.y,
                  SizeHints.width, SizeHints.height,
                  1, Fg, Bg
                 );
  XSetStandardProperties(Dpy,Win,"Rec", "Rec", None, av, ac, &SizeHints);
  Gc = XCreateGC(Dpy, Win, 0, 0);
  XSetBackground(Dpy, Gc, Bg);
  XSetForeground(Dpy, Gc, Fg);
  XSelectInput(Dpy,Win,
              ButtonPressMask | KeyPressMask|ExposureMask);
  XSelectInput(Dpy,Root,PropertyChangeMask);
  XMapRaised(Dpy,Win);

  atom_prop = XInternAtom(Dpy, "__ACADPROP_USERSEND", False);
  atom_delete=XInternAtom (Dpy, "WM_DELETE_WINDOW", False);
  (void) XSetWMProtocols (Dpy, Win, &atom_delete, 1L);
```

```
  hints.input = True;
  hints.flags = InputHint;
  XSetWMHints (Dpy, Win, &hints);
  Fontinit();
  done=0;
  while(!done){
    XNextEvent (Dpy, &Event);
    switch(Event.type){
    case ClientMessage:
      done++;
      break;
    case Expose:
      /* printf("Expose¥n"); */
      break;
    case PropertyNotify:
      if (Event.xproperty.window == Root &&
          Event.xproperty.atom == atom_prop){
        XGetWindowProperty(Dpy,
                           Root,
                           atom_prop,
                           0L,
                           8192,
                           False,
                           XA_STRING,
                           &ret_atom,
                           &ret_format,
                           &ret_len,
                           &ret_after,
                           &ret_prop);
        if (ret_len > 0) {
          ret_prop[ret_len] = NULL;
          drawstring(Win, 10, 50, ">                                    ");
          drawstring(Win, 30, 50, ret_prop);
          XFree(ret_prop);
        }
      }
      break;
    }
  }
} /* main */

/* Initialize font */
Fontinit(){
  kanjigc = XCreateGC(Dpy,Win, 0, 0);
  asciigc = XCreateGC(Dpy,Win, 0, 0);
  kanjifont = XLoadQueryFont(Dpy,"k14");
  asciifont = XLoadQueryFont(Dpy,"7x14");
  if (kanjifont == NULL) {
    printf("Kanji Font Open Error¥n");
  } else{
    XSetFont(Dpy,kanjigc,kanjifont->fid);
  }
  if (asciifont == NULL){
    printf("Ascii Font Open Error¥n");
  } else{
    XSetFont(Dpy,asciigc,asciifont->fid);
  }
} /* Fontinit */

/* Draw String */
drawstring(Window win, int x, int y, u_char *str)
{
```

```
    int     i,j,flag;
    char    ascii[256],*ap;
    u_char  ch,cl;
    XChar2b kanji[256],*kp,kch;
    union{
      short   i;
      char    c[2];
    } u;

    ap = ascii;
    kp = kanji;
    for (j = 0, flag = 0; ;){
      ch= *str++;
      if (ch == NULL)
        break;
      if (ch < 0x80){
        if (flag == 1) {
          flag = 0;
          if (j > 0){
            XDrawImageString16(Dpy, win, kanjigc, x, y, kanji, j/2);
            x += XTextWidth16(kanjifont, kanji, j/2);
          }
          ap = ascii;
          j = 0;
        }
        *ap++=ch;
        j++;
      } else {
        if (flag == 0) {
          flag = 1;
          if (j > 0) {
            XDrawImageString(Dpy, win, asciigc, x, y, ascii, j);
            x += XTextWidth(asciifont, ascii, j);
          }
          kp = kanji;
          j = 0;
        }
        cl = *str++;
        if (kanjifont->min_byte1 != 0 && kanjifont->max_byte1 != 0) {
          u.i = (ch-0xa1) * 94 + (cl-0xa1);
          i = kanjifont->max_char_or_byte2 - kanjifont->min_char_or_byte2 +1;
          (*kp).byte1= u.i / i + kanjifont->min_byte1;
          (*kp).byte2= u.i % i + kanjifont->min_char_or_byte2;
        } else {
          u.i = (ch-0xa1) * 96 + (cl-0xa0) + 127;
          (*kp).byte1=u.c[0];
          (*kp).byte2=u.c[1];
        }
        kp++;
        j += 2;
      }
    }
    if (j > 0) {
      if (flag == 1) {
        XDrawImageString16(Dpy, win,kanjigc,x,y,kanji,j/2);
      } else {
        XDrawImageString(Dpy, win,asciigc,x,y,ascii,j);
      }
    }
  } /* drawstring */
```

Appendix E　共有メモリ (Windows 版)

E.1　概要

Advance CAD とユーザアプリケーションとの通信手段として、Windows の共有メモリを利用することができます。この機能を使用してユーザアプリケーションから Advance CAD へコマンドやパラメータを送信すると、ユーザアプリケーションから Advance CAD を制御することができます。また逆に、Advance CAD からユーザアプリケーションへデータを送信することもできます。この機能を使用するには、Windows の共有メモリの知識が必要です。

共有メモリを使用する通信は、次の手順で行います。

● 送信側

(1) ファイルマッピングオブジェクトを作成する（またはオープンする)。
(2) ファイルマッピングオブジェクトをメモリにマップする。
(3) マップされたメモリにデータを書き込む。
(4) 書き込んだことを受信側へ通知する (つまり、メッセージを送信する)。
(5) 受信側からの応答を待つ（メッセージが通知される)。
(6) 通信したいデータがなくなるまで、(3)、(4)、(5) を繰り返す。
(7) メモリにマップされたファイルマッピングオブジェクトをアンマップする。
(8) ファイルマッピングオブジェクトをクローズする。

● 受信側

(1) ファイルマッピングオブジェクトを作成する（またはオープンする)。
(2) ファイルマッピングオブジェクトをメモリにマップする。
(3) 送信側からの通知を待つ（メッセージが通知される)。
(4) マップされたメモリからデータを読み込む。
(5) 読み込んだことを送信側へ通知する (つまり、メッセージを送信する)。
(6) 受信したい間、(3)、(4)、(5) を繰り返す。
(7) メモリにマップされたファイルマッピングオブジェクトをアンマップする。
(8) ファイルマッピングオブジェクトをクローズする。

共有メモリはファイルマッピングオブジェクト名によって識別されます。通信しようとするアプリケーション同士はお互いに使用するファイルマッピングオブジェクトの名前を取り決めておかなければなりません。次のファイルマッピングオブジェクト名は Acadhelp で利用しますので、使用しないでください。

```
__ACAD_COM_NAM
__ACAD_HELP_EXIST
```

また Advance CAD は他のユーザアプリケーションと通信するために 0xD000 からのメッセージ番号を使用します。プログラミングインターフェイスを使用する場合、メッセージ番号が同じにならないように注意してください。0xE000 からのメッセージ番号を使用することを推奨します。

E.2 ユーザアプリケーションから Advance CAD への送信

- **ファイルマッピングオブジェクト名**
  Advance CAD 起動時にコマンド引数として与えます。

  ```
  % acad -nFileMappingObjectName
  ```

  FileMappingObjectName：ファイルマッピングオブジェクト名

  送信側のアプリケーションは、このファイルマッピングオブジェクト名を使用して通信を行います。

- **Advance CAD の処理**
  Advance CAD が受信できるデータタイプは文字列で、一度に受信できる文字数は 80 バイト以下です。
  80 バイトを越えるとすべて無視されます。文字列の内容は、Advance CAD のコマンドストリームでなければなりません。Advance CAD は、指定された名前を持つ共有メモリにデータが書き込まれたことを通知するメッセージを受け取ると、共有メモリからデータを読み取ります。そして、それがキーボードから入力されたコマンドストリームとみなして解釈、処理します。したがって、この場合、Advance CAD 側はなんら変更する必要はありません。

- **通信側アプリケーションの処理**
  通信側アプリケーションは作成しなければなりません。作成する通信アプリケーションが行わなければならないことを、プログラム例で示します。プログラムの例は全体ではなく一部分です。また、エラーチェックなどを省略しています。プログラムの流れだけを確認してください。

```
#include <windows.h>
              :
              :
#define WM_ACADMSG 0xD000              /* Advnace CAD へ通知するメッセージの番号 */
              :
              :
static char lpszMapName[256];          /* ファイルマッピングオブジェクト名 */
              :
              :
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
  static HANDLE hMapFile = NULL;
  static LPVOID lpMapped = NULL;
  static BOOL   bSendOk  = TRUE;
              :
              :
 switch (message) {
 case WM_CREATE:
   /* ファイルマッピングオブジェクトを作成する。既に存在する場合はオープンする。*/
   hMapFile = CreateFileMapping((HANDLE)0xffffffff, NULL, PAGE_READWRITE,
                                0, 81, lpszMapName);
   if(hMapFile == NULL) {
     /* ここでエラー処理を行う */
   }

   /* ファイルマッピングオブジェクトをメモリにマップする。*/
   lpMapped = MapViewOfFile(hMapFile, FILE_MAP_WRITE, 0, 0, 0);
```

```
    if(lpMapped == NULL){
      /* ここでエラー処理を行う */
    }
    break;
  case WM_LBUTTONDOWN:
  case WM_MBUTTONDOWN:
  case WM_RBUTTONDOWN:
    /* データを共有メモリに書き込み、Advance CAD へメッセージを送信する。*/
    if(hMapFile == NULL || lpMapped == NULL) break;
    if(bSendOk) {
      strcpy((char *)lpMapped, "LBP <0,0> <100,100> <CE>");
      bSendOk = FALSE;
      PostMessage(HWND_BROADCAST, WM_ACADMSG, (WPARAM)hWnd, 0L);
    }
    break;
              :
              :
  case WM_ACADMSG:
    if((HWND)wParam == hWnd || (int)lParam == 0L) {
      /* 自分が発行したメッセージを受信したので、ここでは何もしない。*/
    } else {
      /* Advance CAD が発行したメッセージを受信したので、次のメッセージを送信できる。*/
      bSendOk = TRUE;
    }
    break;
              :
              :
  case WM_DESTROY:
    /* ファイルマッピングオブジェクトをアンマップし、クローズする。*/
    if(lpMapped != NULL)
      UnmapViewOfFile(lpMapped);
    if(hMapFile != NULL)
      CloseHandle(hMapFile);
      PostQuitMessage(0);
    break;
  default:
    return DefWindowProc(hWnd, message, wParam, lParam);
  }
  return 0;
}
```

E.3 Advance CAD からユーザアプリケーションへの送信

Advance CAD からユーザアプリケーションへの送信には、一部制限がありますが、上記で述べた共有メモリや Windows がサポートするプロセス間通信の機能を使用することができます。

Advance CAD 側では、プログラミングインタフェースを使って、送信する機能を追加します。これに対応した受信側アプリケーションを作成します。例として共有メモリを使用したプログラムを添付しますので、参考にしてください。

- Advance CAD 側

  ソースコード　:dspatch32.c, userprop.c
  処理内容　: レジスタの内容、マクロ変数の内容を文字列に変更して、共有メモリに書き込む。

● 受信側アプリケーション

ソースコード　　:rec.c

処理内容　　：共有メモリの内容を読み出し、表示する。

filename ： dspatch32.c

```
/*
  Filename : dspatch32.c
  Category : User dispatcher
*/

#include <stdio.h>
#include "acaddef.h"
#include "acadprm.h"
#include "acadupi.h"
#include "acadusr.h"

/* External Functions */
int xpropopen(void);
void xpropsend(char * data);

/* Static Functions */
static void udr02(int form, TOKEN *token);
static void ucmd21(TOKEN *token);

/*****************************************************************************
   Purpose
        Example of user dispatcher
   Inputs
        token   Token
   Outputs
        None
*/
void dspatch32(TOKEN *token)
{
  switch (Idriver(1)) {
  case 2:
    udr02(Iformat(1), token);
    break;
  default:
    break;
  }
} /* dspatch32 */

/*****************************************************************************
   Purpose
        Example of user driver #2
   Inputs
        form    Form number
        token   Token
   Outputs
        None
*/
static void udr02(int form, TOKEN *token)
{
  switch (form) {
  case 1:
    ucmd21(token);
    break;
```

```
  default:
    break;
  }
} /* udr02 */

/*************************************************************************
  Purpose
        Process UREG Command
*/
static void ucmd21(TOKEN *token)
{
  static int ipropini = 0;
  int ier = 0;

  switch (token->typ) {
  case TknCMD: /* Command */
   if (ipropini == 0) {
     ier = xpropopen();
     if (ier == 0)
       ipropini = 1;
   }
   cmdmdfcla(1);
   break;

  case TknTXT: /* Text */
    {
      int type;
      double dpval;
      char cstr[81];
      type = mcrcalc(token->txt,
                     strlen(token->txt), &dpval, cstr, sizeof(cstr));
      if (type == 3 || type == 4) {
        if (type == 3)
           sprintf(cstr, "%g", dpval);
        Mesageras(MSGZONE, 2, 1);
        Mesagdisp(MZONECOLOR1, 2, 1, 0, 10 * strlen(cstr), cstr);
        if (ipropini)
          xpropsend(cstr);
      } else if (type < 0) {
        ier = 2;
      }
    }
    break;

  case TknEOC: /* CE */
    break;

  case TknEIXT: /* Exit */
    return;

  default:
    ier = 100;
    break;
  } /* End of switch */

  if (ier)
    Errorb();
} /* ucmd21 */
```

filename : userprop.c

```
/*
  Filename : userprop.c
  File-mapping object Sample Code
  Use with Advance CAD Programming Interface
*/

#include <stdio.h>
#include <windows.h>
#include "acadusr.h"

#define WM_USERMSG 0xE000       /* Message # to communicate Advance CAD
                                   Programming Interface with User
                                   Application */

static HANDLE hMapFile = NULL;  /* Handle of the File-Mapping Object */
static LPVOID lpMapped = NULL;  /* Starting Address of the Mapped View */
static char   *MyMmap  = "__ACADMMAP_USERSEND";
                                /* Name of the File-Mapping Object */

/* Create and Map File-Mapping Object for User Application */
int xpropopen(void)
{
  hMapFile = CreateFileMapping((HANDLE)0xffffffff, NULL, PAGE_READWRITE,
                               0, 1024, MyMmap);
  if(hMapFile == NULL)
    return 1;
  lpMapped = MapViewOfFile(hMapFile, FILE_MAP_WRITE, 0, 0, 0);
  if(lpMapped == NULL) {
    CloseHandle(hMapFile);
    return 1;
  }
  return 0;
}

/* Send Data from Advance CAD to User Application */
void xpropsend(char *data)
{
  if(lpMapped == NULL) return;
  euc2sjis(data, data, strlen(data));
  strcpy(lpMapped, data);
  PostMessage(HWND_BROADCAST, WM_USERMSG, 0L, 0L);
}
```

filename : rec.c

```
/*
  Filename: rec.c
  Advance CAD File-Mapping Object Sample Program
*/
#include <windows.h>
#include <string.h>
#include <stdio.h>

#define WM_USERMSG 0xE000       /* Message # to communicate Advance CAD
                                   Programming Interface with User
                                   Application */

LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM, LPARAM);
```

```
static char szWinName[] = "ACADTEST";            /* Name of Window Class */
static char MyMmap[]    = "__ACADMMAP_USERSEND"; /* Name of the File-Mapping Object */

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst,
                   LPSTR lpszArgs, int nWinMode)
{
    HWND hWnd;
    MSG msg;
    WNDCLASS wcl;

    /* define Window Class */
    wcl.hInstance     = hInst;          /* Handle of current instance */
    wcl.lpszClassName = szWinName;      /* Name of window class */
    wcl.lpfnWndProc   = WindowFunc;     /* Window procedure */
    wcl.style         = 0;              /* Default style */

    wcl.hIcon         = LoadIcon(NULL, IDI_APPLICATION); /* Icon style */
    wcl.hCursor       = LoadCursor(NULL, IDC_ARROW);     /* Cursor style */
    wcl.lpszMenuName  = 0;              /* No menu */

    wcl.cbClsExtra    = 0;              /* Additional Information */
    wcl.cbWndExtra    = 0;              /* None */

    /* set window to light-gray */
    wcl.hbrBackground = GetStockObject(LTGRAY_BRUSH);

    /* register window class */
    if (!RegisterClass(&wcl)) return 0;

    /* create window */
    hWnd = CreateWindow(
        szWinName,              /* Window Class Name */
        "WindowsNT Skeleton",   /* Title */
        WS_OVERLAPPEDWINDOW,    /* Window Style - Normal */
        100,                    /* x */
        100,                    /* y */
        100,                    /* width */
        100,                    /* height */
        NULL,                   /* Handle of Parent - None */
        NULL,                   /* No Menu */
        hInst,                  /* Handle of program's this instance */
        NULL                    /* End of parameter */
        );

    /* visualize window */
    ShowWindow(hWnd, nWinMode);
    UpdateWindow(hWnd);

    /* message loop */
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg); /* enable keyboard */
        DispatchMessage(&msg);  /* return to WindowsNT */
    }
    return msg.wParam;
}

/*
    This procedure is called by WindowsNT, passes a message
    from the message queue.
*/
LRESULT CALLBACK WindowFunc(HWND hWnd, UINT message, WPARAM wParam,
                            LPARAM lParam)
```

```
{
    static HANDLE hMapFile = NULL; /* Handle of the File-Mapping Object */
    static LPVOID lpMapped = NULL; /* Starting Address of the Mapped View */
    HDC hdc;

    switch (message) {
    case WM_CREATE:
        hMapFile = CreateFileMapping((HANDLE)0xffffffff, NULL, PAGE_READWRITE,
                                     0, 1024, MyMmap);
        if(hMapFile == NULL) break;
        lpMapped = MapViewOfFile(hMapFile, FILE_MAP_WRITE, 0, 0, 0);
        if(lpMapped == NULL) CloseHandle(hMapFile);
        SetWindowPos(hWnd,HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE|SWP_NOSIZE);
        break;
    case WM_USERMSG:
        if(lpMapped != NULL) {
          RECT r;
          int w, h;
          char buf[1024];
          GetWindowRect(hWnd, &r);
          w = r.right  - r.left;
          h = r.bottom - r.top;
          hdc = GetDC(hWnd);
          SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
          PatBlt(hdc, 0, 0, w, h, PATCOPY);
          strcpy(buf, "> ");
          strcat(buf, lpMapped);
          SetBkMode(hdc, TRANSPARENT);
          TextOut(hdc, 10, 50, buf, strlen(buf));
          ReleaseDC(hWnd, hdc);
        }
        break;
    case WM_CHAR:
        if ((char)wParam != 'q') break;
    case WM_DESTROY:            /* exit program */
        if(lpMapped != NULL) UnmapViewOfFile(lpMapped);
        if(hMapFile != NULL) CloseHandle(hMapFile);
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

索引

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z