

MacDraw Format

Every couple of months someone requests this information on info-mac. Attached is the front end to

a MacDraw-to-Imagen translator. It is in the C language header format. Microsoft graphics programs also output MacDraw format.

```
/*
 *      Description of MacDraw file
 *
 *      <MacDraw file> ::= HeadPacket <ObjectList> <End Object>
 *      <Object List> ::= <Object List> <Object> | <Object>
 *      <Object> ::= <Complex Object> | <Simple Object>
 *      <Complex Object> ::= <Nest Object> <Object List> <End Object>
 *      <Simple Object> ::= HeadWord <Body>
 *      <Object Body> ::= endObject | textObject | gridlineObject |
 *                      lineObject | rectObject | roundrectObject |
 *                      ovalObject | arcObject | freehandObject |
 *                      polyObject | nestObject
 *      <Nest Object> ::= HeadWord nestObject
 *      <End Object> ::= HeadWord endObject
 */

/* integer types */
typedef unsigned char    int8;
typedef short int       int16;
typedef long int        int32;

#define NOBJECTS        11

/* packet at head of MacDraw file */
struct HeadPacket
{
    int16    unknown1[85];
    int16    PlotWidth;
    int16    PlotHeight;
    int16    PageWidth;
    int16    PageHeight;
    int16    unknown2[167];
} HeadPacket;
```

```

/* word at beginning of each graphical object */
struct HeadWord
{
    int8    ObjectType;
    int8    Lock;
    int16   unknown;
} HeadWord;

/* ObjectType values */
#define endObject      0
#define textObject    1
#define gridlineObject 2
#define lineObject     3
#define rectObject     4
#define roundrectObject 5
#define ovalObject     6
#define arcObject      7
#define freehandObject 8
#define polyObject     9
#define nestObject    10
/* Object #11, Paint format bitmaps is not defined here */

/* Lock values */
#define unlocked      0
#define locked       1

/* end object delimiter */
struct End
{
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    unknown;
} End;

/* LineFat values */
#define NFat          6
#define invisibleLine 1
#define thinLine     2
#define mediumLine   3
#define thickLine    4

```

```
#define fatLine      5
#define defaultLine  2
/* fatness in rasters */
float FatTable[NFat] = {0.,0.,1.,2.,3.5,5.};

/* LinePat, FillPat values */
#define NPat      37
#define noPat     1
#define whitePat  2
#define blackPat  3
#define darkgrayPat  4
#define medgrayPat  5
#define lightgrayPat  6
#define coarsedotsPat  7
#define dotsPat     8
#define sparsedotsPat  9
#define topshinglePat 10
#define brickPat    11
#define slantbrickPat 12
#define leftdiagPat 13
#define thicleftdiagPat 14
#define dashleftdiagPat 15
#define narrowleftdiagPat 16
#define heavyleftdiagPat 17
#define dualdiagPat 18
#define horzdashpat 19
#define horzlinePat 20
#define circlePat   21
#define fourwayPat  22
#define smallhatchedPat 23
#define smalldiamondPat 24
#define rightdiagPat 25
#define thickrightdiagPat 26
#define dashrightdiagPat 27
#define narrowrightdiagPat 28
#define heavyrightdiagPat 29
#define trianglePat 30
#define vertdashpat 31
#define vertlinePat 32
```

```

#define rightshinglePat    33
#define heartPat          34
#define largehatchedPat   35
#define largediamentPat   36

/* pattern masks */
#define MPat              8
unsigned char Pat[NPat][MPat] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xBB,0xEE,0xBB,0xEE,0xBB,0xEE,0xBB,0xEE,
0x55,0xAA,0x55,0xAA,0x55,0xAA,0x55,0xAA,
0x88,0x22,0x88,0x22,0x88,0x22,0x88,0x22,
0x88,0x00,0x22,0x00,0x88,0x00,0x22,0x00,
0x80,0x00,0x08,0x00,0x80,0x00,0x08,0x00,
0x08,0x00,0x00,0x00,0x80,0x00,0x00,0x00,
0x80,0x80,0x41,0x3E,0x08,0x08,0x14,0xE3,
0x08,0x1C,0x22,0x41,0x80,0x01,0x02,0x04,
0xFF,0x80,0x80,0x80,0xFF,0x08,0x08,0x08,
0x01,0x80,0x40,0x20,0x10,0x08,0x04,0x02,
0x81,0xC0,0x60,0x30,0x18,0x0C,0x06,0x03,
0x11,0x88,0x44,0x00,0x11,0x88,0x44,0x00,
0x11,0x88,0x44,0x22,0x11,0x88,0x44,0x22,
0x33,0x99,0xCC,0x66,0x33,0x99,0xCC,0x66,
0x01,0x80,0x40,0x00,0x02,0x04,0x08,0x00,
0x66,0x00,0x00,0x00,0x99,0x00,0x00,0x00,
0xFF,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,
0x50,0x20,0x20,0x20,0x50,0x88,0x27,0x88,
0x84,0x9F,0x80,0x80,0x04,0x04,0xE7,0x84,
0x01,0x01,0x01,0xFF,0x01,0x01,0x01,0xFF,
0x55,0x88,0x55,0x22,0x55,0x88,0x55,0x22,
0x80,0x01,0x02,0x04,0x08,0x10,0x20,0x40,
0xC0,0x81,0x03,0x06,0x0C,0x18,0x30,0x60,
0x88,0x11,0x22,0x00,0x88,0x11,0x22,0x00,
0x88,0x11,0x22,0x44,0x88,0x11,0x22,0x44,
0xCC,0x99,0x33,0x66,0xCC,0x99,0x33,0x66,
0x20,0x50,0x00,0x00,0x02,0x05,0x00,0x00,

```

```
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,
0x04,0x04,0x40,0x40,0x04,0x04,0x40,0x40,
0x03,0x84,0x48,0x30,0x0C,0x02,0x01,0x01,
0x0A,0x11,0xA0,0x40,0x00,0xB1,0x4A,0x4A,
0x40,0x40,0x40,0xFF,0x40,0x40,0x40,0x40,
0x41,0x22,0x14,0x08,0x14,0x22,0x41,0x80
};
```

```
/* text object */
struct Text {
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    unknown1;
    int16   BoxDx;
    int16   BoxDy;
    int8    Style;
    int8    Font;
    int8    Size;
    int8    LineSpace;
    int8    Justify;
    int8    Orient;
    int8    unknown2;
    int8    CharCount;
    int16   Top;
    int16   Left;
    int16   Bottom;
    int16   Right;
    /* plus CharCount bytes */
} Text;
char    TextString[256];
```

```
/* Style values */
#define plainStyle    0
#define boldStyle    1
#define italicStyle   2
#define underlineStyle 4
#define outlineStyle  8
#define shadowStyle   16
```

```
#define defaultStyle      0

/* Font values */
#define ChicagoFont      1
#define GenevaFont      2
#define NewYorkFont     3
#define MonocoFont      4
#define VeniceFont      5
#define LondonFont      6
#define AthensFont      7
#define defaultFont     1

/* Size values */
#define size9            1
#define size10           2
#define size12           3
#define size14           4
#define size18           5
#define size24           6
#define size36           7
#define size48           8
#define NTextSize       9

/* LineSpace values */
#define singleSpace     1
#define halfSpace      2
#define doubleSpace    3
#define defaultSpace   1

/* Justify values */
#define leftJustify     1
#define centerJustify  2
#define rightJustify    3
#define defaultJustify  1

/* Orient values */
#define deg0Orient      0
#define deg90Orient     3
#define deg180Orient    2
```

```

#define deg270Orient          1
#define reflect0Orient       4
#define reflect90Orient      6
#define reflect180Orient     5
#define reflect270Orient     7

/* grid line object */
struct GridLine {
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    Arrow;
    int16   y1;
    int16   unknown1;
    int16   x1;
    int16   unknown2;
    int16   y2;
    int16   unknown3;
    int16   x2;
    int16   unknown4;
} GridLine;

/* Arrow values */
#define noArrow              0
#define rightArrow          1
#define leftArrow           2
#define bothArrow           3
#define defaultArrow        0
/* arrow length in rasters */
float ArrowSize[NFat] = {0.,0.,5.,10.,17.,25.};
/* arrow angle in radians */
#define ARROW_ANGLE         .5

/* line object */
struct Line {
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    Arrow;

```

```

    int16    y1;
    int16    unknown1;
    int16    x1;
    int16    unknown2;
    int16    y2;
    int16    unknown3;
    int16    x2;
    int16    unknown4;
} Line;

/* rectangle object */
struct Rect
{
    int8     LineFat;
    int8     LinePat;
    int8     FillPat;
    int8     Corner;
    int16    Top;
    int16    unknown1;
    int16    Left;
    int16    unknown2;
    int16    Bottom;
    int16    unknown3;
    int16    Right;
    int16    unknown4;
} Rect;

/* Corner values */
#define NCorner      6
#define zeroCorner  0
#define one8Corner   1
#define three16Corner 2
#define one4Corner   3
#define five16Corner 4
#define three8Corner 5
/* radii in inches */
float RadiusTable[NCorner] = {0.,.125,.1875,.25,.3125,.375};

/* rounded rectangle object */
struct RoundRect
{

```

```
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    Corner;
    int16   Top;
    int16   unknown1;
    int16   Left;
    int16   unknown2;
    int16   Bottom;
    int16   unknown3;
    int16   Right;
    int16   unknown4;
} RoundRect;
```

```
/* oval object */
struct Oval {
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    unknown;
    int16   Top;
    int16   unknown1;
    int16   Left;
    int16   unknown2;
    int16   Bottom;
    int16   unknown3;
    int16   Right;
    int16   unknown4;
} Oval;
```

```
/* arc object */
struct Arc {
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    unknown;
    int16   Top;
    int16   unknown1;
    int16   Left;
```

```

        int16    unknown2;
        int16    Bottom;
        int16    unknown3;
        int16    Right;
        int16    unknown4;
        int16    StartAngle;
        int16    NDegree;
    } Arc;

/* point objects */
#define NPoint          256
struct Point    {
    int16    y;
    int16    unknown1;
    int16    x;
    int16    unknown2;
} Point[NPoint];

struct Delta    {
    char    dx;
    char    dy;
} Delta[NPoint];

/* freehand line object */
struct FreeHand    {
    int8    LineFat;
    int8    LinePat;
    int8    FillPat;
    int8    unknown1;
    int16   unknown2;
    int16   Bytes;
    int16   PointCount;
    int16   Top;
    int16   unknown3;
    int16   Left;
    int16   unknown4;
    int16   Bottom;
    int16   unknown5;
    int16   Right;

```

```

    int16    unknown6;
    int16    unknown7;
    int16    y1;
    int16    unknown8;
    int16    x1;
    int16    unknown9;
    /* plus Bytes-28 bytes or Bytes/2-14 dx,dy pairs */
} FreeHand;

/* polygon object */
struct Poly {
    int8     LineFat;
    int8     LinePat;
    int8     FillPat;
    int8     unknown1;
    int16    unknown2;
    int16    Bytes;
    int16    PointCount;
    int16    unknown3;
    int16    unknown4;
    int16    Top;
    int16    unknown5;
    int16    Left;
    int16    unknown6;
    int16    Bottom;
    int16    unknown7;
    int16    Right;
    /* plus Bytes-20 or PointCount*4 bytes, PointCount x,y pairs */
} Poly;

/* nest object delimiter */
struct Nest {
    int8     LineFat;
    int8     LinePat;
    int8     FillPat;
    int8     unknown1;
    int16    unknown2;
    int16    ObjectCount;
    int16    unknown3;

```

```

    int16    Bytes;
    int16    Top;
    int16    unknown4;
    int16    Left;
    int16    unknown5;
    int16    Bottom;
    int16    unknown6;
    int16    Right;
    int16    unknown7[5];
} Nest;

/*
 *    count Object lengths
 *
 *    #include "MacDraw.h"
 *    main(){
 *        printf ("HeadPacket=%d\n", sizeof(HeadPacket));
 *        printf ("HeadWord=%d\n", sizeof(HeadWord));
 *        printf ("End=%d\n", sizeof(End));
 *        printf ("Text=%d\n", sizeof(Text));
 *        printf ("GridLine=%d\n", sizeof(GridLine));
 *        printf ("Line=%d\n", sizeof(Line));
 *        printf ("Rect=%d\n", sizeof(Rect));
 *        printf ("RoundRect=%d\n", sizeof(RoundRect));
 *        printf ("Oval=%d\n", sizeof(Oval));
 *        printf ("Arc=%d\n", sizeof(Arc));
 *        printf ("FreeHand=%d\n", sizeof(FreeHand));
 *        printf ("Poly=%d\n", sizeof(Poly));
 *        printf ("Nest=%d\n", sizeof(Nest));
 *    };
 */

```