Common File Format Revision 2. (cff2).

Introduction:

The common file format was introduced by four independent companies involved in the converting industry in order that users of CAD systems within the carton/corrugated industry could send designs to other, up until now, incompatible CAD systems.

Specification:

The supplier of the CAD system should be able to output a plain ASCII text file to disc in the format described by this document. The system supplier should also provide a means of data transfer from the system via a modem using the KERMIT protocol, which is readily available for most computers at no charge (excluding media) from your local distribution center.

The common file format has been devised in such a way that all the necessary information about a carton design can be presented in a standard or non-standard form in any language. This will enable a user say in Spain to request an order from a French diemaker and both parties will see the requirements in their native language. The messages that are sent in this way are contained in the standard message file (an example of which is shown later), and are generated at the local machine.

Various rule-types and sizes also supported by this file format by way of the linetype and auxiliary linetype, which generally specify what type of rule the line represents, and what size type 46, auxiliary type 3 could represent a 6x6 pt. Perforation rule. Examples of both standard auxiliary types will be outlined later.

Limitations/restrictions on common file contents.

The ASCII characters that are allowed in the file are as follows:

‘A’..’Z..’,’a’..’z’,’0’..’9’,’_’,’-‘,’$’,’:’%’,’,’,’.’

Line parameters are delimited by a comma. Both trailing zeros and all spaces (except in text) should be removed in order to reduce the size of file for transmission.

The Standard message file:

Every machine, which can produce and/or receive the common file format must have a copy of the ‘Standard message file’ on their system. This file is read during production and/or decoding of a common file depending on the directives contained in the file (see later). The Standard message file is an ASCII text file made up of lines in the following format…

msg-id,lang-id,message

where meg-id is an integer used to identify the message from the message-directive in the file, lang-id is a string up to 8 characters identifying the language of the local machine, and the message is a character string up to 80 characters in length. A short example standard message file follows...

```
1,ENGLISH, Customer name
1,FRENCH, (French for customer name)
2,ENGLISH, P.O. Number
2,GERMAN, (German for P.0. Number)
```

The parameter file for linetypes and message formats:

Every machine capable of producing the common database will have one parameter file describing the prompts which get presented to the operator and the types and classes (auxiliary types) of rules to be put into the common file. The construction of this will vary depending of the CAD system but should be documented fully by the system supplier within the parameter file so that the user can easily modify its contents required.

Installations of CAD systems capable of decoding CFF2 will have a matching parameter file for each of its customers (if necessary) so that the receiving computer is able to match his customers' requirements. It will be the receiving computer installation responsibility to ensure that any files remain up to date.

The parameter file will again be a plain ASCII text file in the following format…

```
ORDER
        $Customer name:     CFF2 user number 255
        Dieboard:           Y
        P.O. Number:
        %3:                 500 No
        $%1
END
```

Any text in a location such as this gets ignored and can serve as documentation

```
AUX
    1,1
    2,2
    3,3,1,3,3
    4,3,2,6,6
    5,3,3,10,10
    6,3,4,12,12
END
```

Explanation of the above parameter file format.

(i). **Prompts to the user**

The contents of the lines between the lines ORDER and END described what message in presented to the operator and in what format. The program is directed to do this with the use of the directives, their meaning outlined below…

$ Any line whose first character is a dollar sign is not presented to the user and is passed directly to the common file without the first character (the dollar sign). Such a description would occur if, as in the above example the customer's name would want to be placed in the common file. The user would not want to enter his company name every time he ran the program so to save him having to do this; we make the first character in the line a dollar. Only dollar signs, which are the first character, have this effect so a line with $$600.00 on if for example would put $600.00 into the common file.

%n This causes the %n, where n is an integer message-identifier, to be substituted by the corresponding message from the standard message file in the local machines native language. Any trailing spaces are reduced to one space. For example:
The line '%1 :' in the parameter file would go into the common file as '%1 :' and will be decoded at the receiving site as 'Customer Name :'.

: If the last character on the line is a colon, the text preceding the colon is presented to the operator as a question which must be answered by entering a non-whitespace response (i.e. Mandatory response).

: xxx If the line contains a colon other than as the last character, the text preceding the colon is presented to the user as a question with the text after the colon as the default response. The operator can then either change the default response to his own or simply hit enter to take the default.

Note: in the last two directive descriptions, if there is more than one colon on the line in the parameter file, the last colon in the line is considered to be the directive.

Any number of directives may be included in a parameter file line. However, the dollar directive supersedes the colon directive and the percentage directive occurs always.

i.e. If the local language is ENGLISH and the line in the parameter file reads

$%1 : ACME CARTONS

Then, using the example standard message file above, the prompt '%1 : ACME CARTONS' would be placed into the common file without being presented to the user and would be decoded at the receiving site as

Customer name: ACME CARTONS

i). Definition of auxiliary linetypes.

The format and contents of this section (contained within the parameter file between the lines AUX and END) is quite system dependant and therefore the example file should be taken as just that. The idea of this section of the parameter file is to output standard linetypes with varying auxiliary types to the common file. In the common file, the linetype represents the type of rule to be used in the die (cut, crease, perf, etc.) and the auxiliary type represents the size of rule to be used (3x3, 6x6, etc.).

The standard linetypes at present along with the number of parameters used to define the size of the rule are as follows…

1	Cut
2	Crease
3	Perforation (2 parameters, cut and gap)
4	Score/half cut.
40	Rillma/Matrix designs.
41	Zipper (3 parameters, length, gap, and angle).
42	Cut/Crease (3 parameters, cut, crease, and land length).
43	Draw but don't burn into the die.
44	Burn but don't rule.
45	Safety edge (2 parameters, height, and pitch).
46	Dimensions.
99	Punch lines representing a shape that the punch would be expected to cut.

The number of parameters represents the amount of additional information required in the parameter file to describe the auxiliary linetypes' sizes (to aid in the description of these, see the enclosed drawings).

There is a special case for linetype 40 (Rillma/matrix) which must give the tool to use. As we do not have a parameter for tools we will use the Pointage parameter as a tool identifier according to the following table

Linetype	***Pointage***	***Description***
40	1	with grain direction
40	2	against grain direction
40	3	location holes
40	4	peripheral cut
40	5	extended chamfer

In addition to user-defined auxiliary linetypes, linetype 46 (dimension lines) has a pre-defined set of auxiliary linetypes for the description of dimension arrows etc. These are as follows:

Linetype	***Aux-type***	***Description***
46	0	unheaded arrow
46	1	arrow head at start of line
46	2	arrow head at end of line
46	3	arrow heads at both ends

Therefore, when describing arrows in the common file, the above linetype of 46 and the relevant auxiliary linetypes must be used.

The actual contents of your parameter file to describe the linetypes you will use must be integer values separated by commas. In our simple example above the first number is the linetype our system uses locally for each design, the following number defines to what type and auxiliary type this is mapped to in the common file (i.e. a linetype of 6 representing a 12x12 perforation rule on our standard system will get output as a linetype 3 auxiliary linetype 4 in the common file).

Geometry and other information in the common file.

A stripped down example of a common file can be sent at the end of this section. Basically, the first part of the file (between the lines ORDER and END) contains all the information required to process the customer's order. This is all the information and responses to the prompts specified in the local parameter file. The next section, between the lines MAIN and $EOF contains the geometric data for all the lines and cartons in the design. As can be seen from the file below, the format supports subroutining with the use of SUB and C (call) commands with transformations as well as lines, and arcs, text, dimensions, etc.

The various methods of describing lines, arcs, arrows, etc. are outlined below…

(i). Straight lines.

Straight lines are described in the common file in the format of:

L, p, t, at, sx, sy, ex, ey, nbridges, wbridge

Where a comma is the delimiter and the various parameters have the following meaning.

L	us the capital letter L.
P	is the pointage in points (1/72 inches).
t	is the common file linetype
at	is the common file auxiliary linetype
sx, sy	is the start coordinate of the line.
ex, ey	is the end coordinate of the line.
nbridges	is the number of bridges in the line.
wbridges	is the width of the bridges in mm.

(ii). **Arcs**.

Arcs are described in the common file in the format of:

A, p, t, at, sx, sy, ex, ey, cx, cy, =/-1, nbridges, wbridges

Where a comma is the delimiter and the various parameters have the following meaning.

A	us the capital letter A.
P	is the pointage in points (1/72 inches).
t	is the common file linetype
at	is the common file auxiliary linetype
sx, sy	is the start coordinate of the arc.
ex, ey	is the end coordinate of the arc.
nbridges	is the number of bridges in the arc.
wbridges	is the width of the bridges in the arc in mm.

(iii). **Text**.

Text is described in the common file in the format of:

T, p, t, at, x, y, angle, height, width
textstring

Where a comma is the delimiter and the various parameters have the following meaning.

T	us the capital letter T.
P	is the pointage in points (1/72 inches).
t	is the common file linetype
at	is the common file auxiliary linetype
x, y	is the bottom left position of the text design.
Angle	is the rotation about x, y of the text design.
Height	is the height of the characters in mm.
Width	is the width of the characters including the inter-character gap in mm.

(iv). **Dimension lines and text.**

Dimension lines and/or text are simply described using the formats using the formats for lines, arcs and text as described above with a linetype of 46. Arrows are described using the line format with a type of 46 and the relevant auxiliary linetype as described above under 'Definition of auxiliary linetypes'.

(v). **Subroutines**.

Geometric data for a subroutine is enclosed between the lines 'SUB subname' and 'END' and all coordinates between these lines should be relative to the design's position as defined in the call to that subroutine.

The first line in the definition of a subroutine is in the format of:

SUB, subname

Where a comma is the delimiter and the two parameters have the following meaning.

SUB	is the word 'SUB' in capitals.
subname	is a unique identifier for that subroutine of no more than eight characters in length.

The last line in the definition of a subroutine is END.

(vi). **Subroutine calls**.

Subroutine calls are described in the common file in the format of:

C, subname, x, y, a, scale-x, scale-y

Where a comma is the delimiter and the various parameters have the following meaning.

C	is the capital letter C.
subname	is the reference to a subroutine name of no more than 8 chars.
x, y	is the absolute position of the subroutine in the design.
a	is the angle through which the design is rotated about its position x, y.
scale-x	is the scale in the x direction (see later about scales).
scale-y	is the scale in the y direction.

Scale parameters in the file.

Some of the lines in the common file take scales as one of their parameters. These take the form of a scale in the x direction followed by a scale in the y direction separated by a comma. A negative scale denotes that the item being described should be negated in that axis and the value of the number denotes the actual scale to make the object in that axis.

e.g. a scale of –1,1 would mean full size mirrored in the x-axis, a scale of .9881, -1.5 would mean scale the design by .9881 in the x direction and 1.5 in the y direction and mirror the design in the y axis.

The parameter file to be used for the example is shown below.

```
ORDER
   $%1          : COMMON FIEL USER 1
    %2          :
    %3          :
    %4          : Y
    %5          : 23.80
    %6          : 23.50
    Wood type   : Birch.
End
```

The following lines between AUX and END the auxiliary linetypes.

```
AUX
    1,1
    4,3,3,6,6
    5,42,1,10,10,8
    6,3,4,12,12
END
```

An example common file listing.

```
$BOF
V2
%1 : COMMON FILE USER 1
%2 : 1234ABC
%3 : ASAP
%4 : Y
%5 : 23.80
%6 : 23.50
Wood type : Birch.
END
MAIN, design-name
UM
LL, <lower left X>, <lower left Y>,
UR  <upper right X>, <upper right Y>,
SCALE, 1, 1,
L, 2, 1, 0, 12.5, 0, 0, 125, 2, 4
C, DESIGN 1, 0, 0, 0, 1, 1
C, DESIGN 2, 150, 0, 0, 1, 1
C, DESIGN 2, 150, 200, 180, 1, 1
END
SUB, DESIGN 1
L, 2, 1, 0, 40, 100, 140, 100, 2, 6
A, 2, 3, 3, 0, 0, 100, 0, 50, 0, -1, 0, 0
END
SUB DESIGN 2
L, 6, 1, 0, 100, 100, 200, 100, 2, 12
A, 2, 3, 3, 0, 0, 100, 0, 50, 0, +1, 0, 0
END
$EOF
```

The standard message file used in the above example was

```
1, English, Customer Name
2, English, P.O. Number
3, English, Required delivery
```

4, English, Die Required
5, English, Cut rule height
6, English, Crease rule height

Note:

All values should be accurate to two decimal places in mm. And if slight inaccuracies in arc generation occur then center of the arc may be moved within +/- .05 mm. To allow for any slight geometric rounding error.

In decoding the common file, items that can be scaled and rotated must be rotated, then negated if required.

As you can see from the information given on the previous page, there are several lines which have no explanation, their explanation follows

The second line of the file is ‘V2’, this signifies that the Common Format file contains information that is compatible with this release of CFF and will allow for future generations of CFF to be differentiated between.

After the MAIN statement there are 3 lines of information, the first line is ‘UM’ which signifies units METRIC, if you want to have imperial units then the line should read ‘UI’ but it is only the geometry which is converted (start, end, center, bridge width, etc.) pointages should remain in points (1/72 inch units). The next 2 lines are the drawing limits as given LL for lower left and UR as upper right.