

DPF Format Description

DPF Version 5

November 1998

Document:

DPF Format Description
November 1998

Copyright:

©1998 Barco Graphics nv, Electronic Tooling Systems division

All rights reserved. This document may not be reproduced by any means, in whole or in part, without written permission of the copyright owner.

This document supersedes all previous dated versions. The material in this document is subject to change without notice.

No responsibility is assumed for any errors, which may appear in this document, neither for its use.

Trademarks:

All product names and sites are trade names or registered trademarks of their respective owners.

For more information, contact:

Europe & Asia	North America
Barco nv	Barco Inc
Electronic Tooling Systems division	Electronic Tooling Systems division
Tramstraat 69	30 South Satellite Road
9052 Gent	South Windsor, CT 06074
Belgium	USA
☎ +32 9 21 69 366	☎ +1 (860) 291-7000
☎ +32 9 21 69 870	☎ +1 (860) 291-7021

Email: etsinfo@barco.com

Web Site: <http://www.barco.com/ets/>

Contents

Contents	3
Introduction.....	5
DPF Syntax	6
Global information	6
Conventions	6
File structure	7
HEADER	8
Syntax.....	8
Description.....	8
Example	8
UNIT	9
Syntax.....	9
Description.....	9
Example	9
COMMENT	10
Syntax.....	10
Description.....	10
Example	10
REFPOINTS	11
Syntax.....	11
Description.....	11
Example	11
NETLIST TABLE.....	12
Syntax.....	12
Description.....	12
DATA.....	13
Syntax.....	13
Example	13
OBJATTR TABLE.....	14
Syntax.....	14
Description.....	14
APERTURE DEF.....	15
Circle	15
Rectangle.....	16
Square.....	17
Donut	17
Box	18
Octagon	19
Thermal	19
Text.....	20
Contour.....	20
Complex.....	21

Block	22
APERTURE OPTIONS	23
Mirroring	23
Rotation	23
Name	24
Aperture Attributes	24
Scaling	24
Reverse	25
Pattern	25
PLOT DATA	26
Flash	26
Move	26
Draw	27
Arc	27
Vector Text	27
OBJ ATTR	29
DPF Example	30

Introduction

DPF stands for the **D**ynamic **P**rocess **F**ormat.

The DPF information is part of Barco's JOB database structure. Each JOB contains reference to one or more DPF files.

DPF is the data format, developed by Barco, to represent layer information of a Printed Circuit Board. This format not only describes the image of the layer such as pads, tracks, holes, power and ground planes but also electrical netlist information as well as additional product information represented with attributes.

Developed specially for the Electronics Manufacturing industry, DPF offers a variety of powerful features such as embedded aperture definitions, reverse objects, contour for outline description and block apertures to represent Step & Repeat items.

This document describes the DPF syntax.

DPF Syntax

General information



- DPF uses the right hand coordinate system.
- DPF is case insensitive.
- Separators are only required between two consecutive keywords.
- A keyword stops at the first non-alphabetic character (for instance a space, a line feed or a carriage return) and at the end of a data block.

Conventions

- []
Square brackets: indicate optional file elements. They are not part of the actual format.
- < >
Angular brackets: indicate fixed file elements. They are not part of the actual format.
- { }
Braces: indicate that the options stated are not fixed and may be substituted for other formats. The options enclosed in braces can be separated by a vertical line |. Both the braces and the vertical line are not part of the actual format.
- EOL
Indicates the **end of a line**, this is either a carriage return or a linefeed character. Barco UCAM normally skips end of line characters, but when explicitly required, it is mentioned in the syntax description.
- SPACE
Indicates an explicit space character. Barco UCAM normally skips space characters, but when explicitly required, it is mentioned in the syntax description.

-

Ellipsis: indicate a logical continuation of the syntax. The epilepsy's are not part of the actual format.

- *

Asterisk: separates the fixed part from the optional part of a string of alphabetical characters. IN*CH means that you can use both IN and INCH. The asterisk is not part of the actual format.

- **Bold text**

Quotes elements used in the syntax. Bold Text is normally used in the description and is not part of the actual format.

- All other items should be taken as literal unless described otherwise.

File structure

[HEADER]

[COMMENT]

[OBJATTR TABLE]

[NETLIST TABLE]

[REFPOINTS]

[DATA]

HEADER

DPF files have headers that include some information about the data contained in the file. This header information is generated and added automatically by every program that saves DPF-files.

Syntax

```
[UNIT] Xminx,maxxYminy,maxy<EOL>
[REFPOINTS]
[COMMENT]
```

Description

Defines the extents of the DPF data to be within **minx** and **maxx** for the x-range and **miny** and **maxy** for the y-range. The Reference points are defined together with some comments.

Example

```
U=MIL X-39.37,413Y-39.37,39.37
;this is DPF-file info
P1=372,100
```


UNIT

Every DPF command such as move, draw, etc. that executes an action on coordinates also requires the unit in which the coordinates are defined. The unit command is modal, so it does not have to be repeated for every command.

Syntax

$$U = \{MM \mid IN*CH \mid MIL\}$$

Description

Sets the unit in

- **MM:** millimeter, metric
- **INCH:** inch, imperial
- **MIL:** mil, thousandth of an inch

Default = MIL

Example

```
U=MM A1=CIRCLE,1 M0,0D10
;draws a line with a circular aperture of 1mm from position 0,0
to ;10mm,0.
```

COMMENT

This is informational data and has no effect on the visual data. If information was added using Info in the Job Definition dialog box, then this is automatically saved using comments.

Syntax

```
; comment<EOL>  
[ COMMENT ]
```

Description

comment is a string containing any characters. The first character in the comment may not be \$ or # as these have a special meaning (see OBJATTR TABLE). As described in the syntax, you can have multiple comments as long as each comment starts with a semicolon (;) and ends with an EOL.

Example

```
;This is information
```

REFPOINTS

This is a list of reference points. Reference points are similar to DPF Comments as they contain only informational data and have no effect on the visual data. However, some actions in UCAM depend on these reference points and have different results when other reference points are defined.

Reference Points may be used for:

- registration
- output (e.g. Orbot Link)

Syntax

```
Pn=refx,refy[ [EOL]Pn=rex,refy[ ... ] ]<EOL>
```

Description

Define reference points with index **n** and position **refx**, **refy**.

n is a positive number.

refx and **refy** are values in the current unit.

Example

```
P0=0,0P1=372,100P2=200,200
```

Defines 3 reference points.

NETLIST TABLE

DPF can contain netlist information. Netlist information is informational data and has no effect on the visual data. Netlist information is used within UCAM for functions that require electrical connection information or to create test fixtures for electrical test machines. While some of these functions do not specifically require netlist information, overall performance is always faster and more accurate when netlist information is available. Use the NETLIST TABLE section to define the net attributes if required.

Syntax

```
Nn,name="value" [,name="value" [ ... ]]<EOL>
[
Nn,name="value" [,name="value" [ ... ]]<EOL>
[ ... ]
]
```

Description

n: is the net-number that you want to define name-value pairs for.

name: is a normal ASCII string, no special characters are allowed.

value: is a normal ASCII string. If the value contains special characters (like ; and ,), the value should be declared between double quotes.

For each net **n**, **name** or **value**, pairs can be defined. Within the NETLIST TABLE, the net-numbers must increment appropriately.

Example

```
N100,name="power",impedance="50Ohm"
N200,name="ground"
```

DATA

The DATA section gathers all the visual information of the DPF. Within this section shapes are used for plotting. Before a shape can be used for plotting it must be defined by means of an aperture definition.

Syntax

```
[ <UNIT><SPACE> ]<APERTURE DEF><SPACE>[ PLOT DATA ]<EOL>
[
[ <UNIT><SPACE> ]<APERTURE DEF><SPACE>[ PLOT DATA ]<EOL>
[ ... ]
]
```

Example

```
A1=CIRCLE,40 M6000,7000D,2500C5500,2000,5500,2500,CW D1000
C500,2500,1000,2500,CW D,7000C1000,7500,1000,7000,CW D5500
C6000,7000,5500,7000,CW
A2=SQUARE,100,P=L:30:10 M1886,2574D4886
A4=TEXT,("This is an example"),200,R=90 F1200
A5=COMPLEX,(M250,100D,-100D-250D,100D250)
F2150,3075F,3575F,4075F,4575F,5075F,5575
```

OBJATTR TABLE

Each object in the DPF Plot Data can be given an attribute. An attribute always has a name, but the value of the attribute is optional. Within DPF Plot Data, the name and value of the attribute are not fully inserted, instead a reference is used. This reference refers to the OBJATTR TABLE by means of an index. The OBJATTR TABLE must be defined before any attributes can be used.

Syntax

```
;$name_index<SPACE>name<EOL>
[
;$name_index<SPACE>name<EOL>
[ ... ]
]
[
;#value_index<SPACE>"value"<EOL>
[
;#value_index<SPACE>"value"<EOL>
[ ... ]
]]
```

Description

Two tables, one which always links a **name_index** with a **name** and one which always links a **value_index** with a **value**.

name_index and **value_index** are numbers larger then or equal to zero.

name: defines an attribute name and should not contain any special characters.

value: defines an attribute value.

Example

```
;$0 nonplated
;$1 plated
;$2 smd
;#0 "true"
;#1 " "
```

APERTURE DEF

This section defines all apertures. Apertures must be defined before they can be used in the Plot Data. The definition of an aperture consists of its shape, size and extra options (mirroring, rotation, scaling, fill pattern and aperture attributes).

Regular aperture shapes are:

- Circle
- Rectangle
- Square
- Donut
- Box
- Octagon
- Thermal
- Text
- Contour

Special **composed** aperture shapes are:

- Complex
- Block

Syntax

An=DEFINITION

Description

Defines an aperture with aperture number **n**. This aperture number can be any value between zero and one billion, (including zero). The aperture number does not have to be unique, meaning that apertures are distinguished based on their location in the file. All possibilities for **DEFINITION** are described below.

Circle

Defines a circular aperture, this is the most common aperture used for pads and for drawing tracks with rounded edges. Circular flashes are also commonly used to represent drilling holes.

Syntax

`An=C*IRCLE , d [, <APERTURE OPTIONS>]`

Description

- **d**: Defines a circular aperture with diameter d and is expressed in the current unit (See DPF Unit Change).
- **n**: is the aperture number

Example

`A10=C , 10`

`A11=CIRCLE , 10`

Rectangle

This defines a rectangular aperture, which can be used for placing SMD's or for drawing tracks with rectangular edges.

Syntax

`An=R*ECTANGLE , sx , sy [, <APERTURE OPTIONS>]`

Description

Defines a rectangular aperture as follows:

- **sx**: the size in the x-direction
- **sy**: the size in the y-direction
- **n**: the aperture number

Example

`A12=R , 100 , 50`

`A13=RECTANGLE , 100 , 50`

Square

A square is a special case of a rectangular aperture, where the x-size is equal to the y-size. It is most commonly used for drawing tracks with square edges.

Syntax

```
An=S*QUARE, s [ , <APERTURE OPTIONS> ]
```

Description

This defines a square aperture with size **s**. This is the same as a rectangular aperture where **sx** and **sy** are both equal to **s**.

Example

```
A14=S, 20  
A15=SQUARE, 20
```

Donut

A donut can be used in combination with rectangular apertures to define targets. When used as a reversed flash, it can separate copper. The netlist of the separated copper is the netlist of the donut flash, and not the underlying copper.

Syntax

```
An=D*ONUT, douter, dinner [ , {RR|SS|SR} ] [ , <APERTURE OPTIONS> ]
```

Description

Defines a donut with:

- **douter**: the outer diameter or size of the donut
- **dinner**: the inner diameter or size of the donut

3 different kinds of donut may be defined as follows:

- **RR**: circular outer and circular inner shape (round/round), this is the default
- **SS**: square outer and square inner shape (square/square)
- **SR**: square outer and circular inner shape (square/round)

Where **n** is the aperture number

Example

```
A16=D, 200, 50, RR  
A17=DONUT, 200, 50, SR
```

Box





A box is mainly used to define SMD-pads. It is a rectangle with featured corners.

Syntax

```
An=BO*X, sx, sy, {R*OUNDED | S*TRAIGHT | A*NTIQUE | C*UT}=xcutoff[:ycutoff] [, <APERTURE OPTIONS>]
```

Description

Defines a box, where the type of corner can be one of the following:

- **ROUNDED:** rounded corners e.g. . When **xcutoff** is half **sx** and **ycutoff** is half **sy**, an ellipse is defined.
- **STRAIGHT:** a flattened corner where the edge consists of only one line e.g. . When **xcutoff** is half **sx** and **ycutoff** is half **sy**, a diamond shape is defined.
- **ANTIQUE:** inward rounded corners e.g. 
- **CUT:** two inwardly perpendicular lines as corners e.g. 
- **x:** the x size of the box
- **y:** the y size of the box
- **cutoff:** defines the x size of the corner
- **cutoff:** defines the y size of the corner
- **cutoff:** should be larger than zero and less or equal to half sx
- **cutoff:** should be larger than zero and less or equal to half sy
- **n:** the aperture number

Note: When **ycutoff** is omitted, it is defaulted to the value of **xcutoff**

Example

```
A18=BO, 200, 100, S=50:25
```

```
A19=BOX, 200, 100, R=50
```

Octagon

An octagon is a special case of a Box with STRAIGHT corners, where each of the 8 sides has an equal size.

Syntax

```
An=O*CTAGON, size[ , <APERTURE OPTIONS> ]
```

Description

Defines an octagon, with size **size**. This is the same size that has to be used for **sx** and **sy** when using a BOX aperture to define the same shape.

Example

```
A20=O, 200  
A21=OCTAGON, 200
```

Thermal

A thermal is a shape that is used as a thermal isolation for pins that are connected to large copper areas. Thermals should always have the reverse polarity of the data used to define copper.

Syntax

```
An=THE*ERMAL, outer, inner, gap, ngap, startangle, {RR|RS|SS|SR}[ , <APERTURE OPTIONS> ]
```

Description

Defines a thermal where:

- **outer**: the outer diameter or size
- **inner**: the inner diameter or size
- **gap**: the size of the gap (or the size of the remaining copper when used as reverse flash as it should be)
- **ngap**: the number of gaps equally spaced around the thermal
- **startangle**: is the angle of the first gap. Zero means to the right side of the center point
- **n**: the aperture number

The type of thermal can be one of the following:

- **RR**: round with rounded edges
- **RS**: round with square edges
- **SR**: square with rounded edges

- **SS:** square with square edges

Example

```
A22=THE,200,180,20,4,45,rr
A23=THERMAL,200,180,20,3,0,rs
```

Text

Text is more than mere information, text also represents material (copper on signal layers).

Syntax

```
An=T*EXT,("text_string"),[height[:width]][,<APERTURE OPTIONS>]
```

Description

Defines a text aperture where:

- **text_string:** the string of text characters that is displayed. As this string is enclosed between double quotes, a double quote inside the string has to be avoided by inserting it twice.
- **height:** the height of the text, when the height is omitted, then the height defaults to 100 mil.
- **width:** the width of the text, when no width is defined, then the width defaults to 84.21% of the text height.
- **n:** the aperture number.

Example

```
A24=T,("Hello"),200:168.42
A25=TEXT,("Say ""Hello"""),200:200
```

Contour

A contour aperture is used to describe large copper areas.

Syntax

```
An=CON*TOUR[,ST*ROKE=st][,<APERTURE OPTIONS>]
```

Description

A contour aperture is considered as a pen with size = 0. This aperture is used to draw contour lines. After the selection of a contour aperture, you can add an enumeration of contour lines described by means of move, draw and arc. An overlap of two contour lines is not allowed.

You can add a stroke with size **st**, which results in a contour with a solid line on top of the contour. The stroke value is the width of this line.

Example

```
A26=CON  
A27=CONTOUR , STROKE=10
```

Complex

A complex is used to define shapes that have no primitive in DPF. This way an aperture can be defined with a triangular shape.

Syntax

```
An=COM*PLEX, (<contour>[<contour>[ ... ]])[, <APERTURE OPTIONS> ]  
An=COM*PLEX, Ac[, <APERTURE OPTIONS> ]  
contour :  
<MOVE> { <DRAW> | <ARC> } [ { <DRAW> | <ARC> } [ ... ] ] { <DRAW> | <ARC> }
```

Description

The first syntax defines a complex using an enumeration of contour descriptions. Each contour is described using the DPF plot commands for move, draw and arc. Each contour starts with a move command, is followed by at least one draw or arc command and ends with a draw or arc to close the contour.

The second syntax defines a complex using the same contour definition as the last complex defined with aperture number = **c**. This is useful for reusing a complex and applying different aperture options.

n: the aperture number of the newly created complex.

Example

```
A28=COM, (M0, 0D100D, 100, D0D, 0)  
A29=COMPLEX, A28
```

Block

A block contains DPF data that has to be repeated several times.

Syntax

```
An=B*LOCK, ([ COMMENT ][ OBJATTR TABLE ][ DATA ]) [ , <APERTURE OPTIONS> ]  
An=B*LOCK, Ab [ , <APERTURE OPTIONS> ]
```

Description

The first syntax defines the block that contains the DPF commands between round brackets. These DPF commands can also contain other block definitions.

The second syntax defines a block that has the same definition as the last block defined with aperture number = **b**. This is useful to define a block with the same definitions but with different aperture options.

n: the aperture number of the newly created block.

Example

```
A30=B, (U=MM A1=C, 0.1M0, 0D5D, 5D0D, 0)  
A31=BLOCK, A30
```

APERTURE OPTIONS

Every aperture in DPF can also have some extra options. These options are associated with the aperture definition and the available options are common for each kind of aperture. The options are separated by a comma.

Mirroring

Syntax

M*IRROR={ X | Y | XY }

Description

- **X**: mirror along the x-axis
- **Y**: mirror along the y-axis
- **XY**: mirror along both the x- and y-axis

Example

A10=TEXT, "ABC", 10:8.42, M=X

Rotation

Syntax

R*OTATION=r

Description

r: is an angle specified in degrees. A positive value indicates a counterclockwise rotation while a negative value indicates a clockwise rotation.

Example

A10=RECTANGLE, 10, 20, R=90

Name

Syntax

NAME="name"

Description

name: is the optional name of an aperture.

Example

A10=CIRCLE,10,NAME="special plated"

Aperture Attributes

Syntax

A*ATTRIBUTES=(name="value" [,name="value" [...]])

Description

Associate **name-value** attribute pairs to the aperture

name: may not contain any special characters

Example

10=CIRCLE,10,ATTR=(attr1="1",attr2="2")

Scaling

Syntax

S*SCALE=s

Description

s: the scaling factor.

Note: UCAM only use scale factors for block and complex apertures, as they can refer to a previously defined definition. A scaling factor on any other aperture has no function, and UCAM automatically adapts the dimensions of the aperture.

Example

A10=BLOCK,A9,SCALE=0.2

Reverse

Syntax

RE*VERSE

Description

Whenever this keyword is used, the aperture is plotted in reverse, meaning white on black.

Example

A10=CIRCLE, 10, REVERSE

Pattern









Syntax

P*ATTERN={V|H|VH|L|R|LR|C45|C90}: [step[:width]]

Description

- **step**: the distance between two pattern elements, the default is 10 mil.
- **width**: the width of a pattern element, the default is 5 mil.

The pattern type can be defined as follows:

	V	Vertical
	H	Horizontal
	VH	horizontal and vertical
	L	diagonal slant from left
	R	diagonal slant from right
	LR	diagonal slant from right and left
	C45	45 degree circular dots
	C90	90 degree circular dots

Example

A10=CIRCLE, 10, P=LR:1:0.5

PLOT DATA

The DPF Plot Commands actually execute plotting and uses defined apertures to make flashes, arcs, draws and contours.

Note: In DPF, the X and Y values are modal, and therefore they only need to be specified whenever their value actually changes.

Flash

Syntax

[<UNIT><SPACE>] [Nn] F [x] [, y] [OBJ ATTR]

Description

This command inserts a flash at position **x**, **y**, with an optional net number **n** and optional object attributes. The net number is modal and remains for all following plot commands unless modified.

Move

Although a move command is not directly available in UCAM, it constitutes part of a draw or an arc.

Syntax

[<UNIT><SPACE>] [Nn] M [x] [, y]

Description

Moves to position **x**, **y** without performing any plotting. This is done to mark the start point of a single arc or draw, or to mark the start of a series of draws and arcs.

There is no option for adding attributes here as object attributes are only useful when defined for whole objects.

It is however possible to assign an option net-number **n**, and it counts for any draw or arc commands that follow, until modified in the next flash, move or vector text command.

Draw

Syntax

[<UNIT><SPACE>]D[x][,y][OBJ ATTR]

Description

Draw from the current position (usually from the previous move, draw or arc command) to position **x**, **y**. There is no net-number expected here, as it is only meaningful for the first move command.

It is possible to assign object attributes.

Arc

Syntax

[<UNIT><SPACE>]C[x][,y],mx,my,[{CW|CCW}][OBJ ATTR]

Description

Draws an arc from the current position to position **x**, **y** with center point **mx**, **my**.

CW: clockwise arc

CCW: counter-clockwise arc (default)

As object attributes are only meaningful on the whole object and this command is only a part of it, there is no option to add these attributes here.

It is possible to assign an optional net-number **n**, and it counts for any following draw or arc command, until changed in the next flash, move or vector text command.

Vector Text

Syntax

[<UNIT><SPACE>]V[x][,y],("text_string"),width[:space],fontname,
(<expanded text>)[VTX OPTIONS][OBJ ATTR]

Description

- **x / y**: are the coordinates that the text is positioned on. It is the position of the first character of the text defined in **text_string**.
- **text_string**: represents the string of text characters that should be displayed. As this string is enclosed between double quotes, a double quote inside the string has to be avoided by placing it twice.
- **width**: is the horizontal offset between the starting points of two consecutive characters. In

this case the text is mono-spaced and can be used for tabular text.

When **width** is 0 then **space** defines the white space between two characters.

- **fontname:** is the name of the vector font. As no path names are allowed, the font is searched on \$UFNTDIR by default.
- **<expanded text>:** are the actual draws of the text, so that the font file is not required to visualize the DPF file. As vector text consist of multiple characters, it is not meaningful to assign a net-number to it.

Optional objet attributes can be assigned.

The possible VTX OPTIONS are mirroring, scaling and rotating and are the same as for the DPF Aperture Options.

Example

```
A9=CIRCLE,10 V400,100,("1"),800,vtx,(U=MM  
M0,75,21.75D,075),S=0.1
```

OBJ ATTR

On each Object in DPF Plot Data, extra attributes can be added. This can be done manually by using the Object Attribute Editor, or automatically by means of a tool.

Before associating any object with an attribute name/value pair, all the attribute names and values used must be defined in a DPF Object Attributes Table.

Syntax

```
;$name_id[=value_id][,name_id[=value_id][ ...]]<EOL>
```

Description

Each **name_id** and **value_id** refers to a name and value defined in the DPF Object Attributes Table.

When the **value_id** is omitted, this means that an attribute has been assigned without any value. This is not the same as an empty value, as the latter results in a reference into the DPF Object Attributes Table. Attributes are non-modal and have to be assigned for each object.

Example

```
A1=CIRCLE,10 F10,0;$1=0,2,3=1  
F20;$1=1
```

DPF Example

```
U=MIL X0,6520Y0,8520
;$0 nonplated
;$1 plated
;$2 smd
;#0 "true"
;#1 ""
;data
A1=CIRCLE,40 M6000,7000D,2500C5500,2000,5500,2500,CW D1000
C500,2500,1000,2500,CW D,7000C1000,7500,1000,7000,CW D5500
C6000,7000,5500,7000,CW
A2=SQUARE,100,P=L:30:10 M1886,2574D4886
A4=TEXT,("This is an example"),200,R=90 F1200
A5=COMPLEX,(M250,100D,-100D-250D,100D250)
F2150,3075F,3575F,4075F,4575F,5075F,5575
A7=BLOCK,(A1=CIRCLE,40 M500,0D0D500,500D,0)
F2886,3074F,4074F,5074
A8=BLOCK,A7,R=180 F4386,5574F,4574;$0=0,2
F,3574
A10=CON,P=HV:30:10,STROKE=20
M4886,6074D,6574D1886D,6074D3386D4386D4886
A17=TEXT,("Figure 5"),100 F1448,1188
A18=CIRCLE,20 M10,8510D,10D6010D6510D,8510D10
A19=OCTAGON,300 F4800,3000
A20=DONUT,240,180 F,3500 A21=DONUT,240,180,SS F5100
A22=DONUT,240,180,SR F5400
A23=THERMAL,300,240,30,4,0,RS F4800,4200
A24=THERMAL,300,240,30,4,0,SS F5400
A25=THERMAL,300,240,30,4,0,RR F4800,4800
A26=THERMAL,300,240,30,4,0,SR F5400
A27=THERMAL,300,240,30,4,45,RS F4800,5400
A28=THERMAL,300,240,30,4,45,SS F5400
```