Summa Cutters

Programmer's Guide

Revision: 1220/1

Pdf creation date: 07 december 2020

Revision History

SEP	2000	Original Issue.
JAN	2001	EXTRA_PEN ADDED.
MAR	2001	Software USB handshake added.
SEPT	2003	RELOAD_MARKERS added.
OCT	2003	AUTOMATING OPOS appended.
MAR	2004	Appended RELOAD_MARKERS vs LOAD_MARKERS commands
MAY	2005	S class models, cut media command.
AUG	2005	Added OPOS_LEVEL
SEPT	2005	S140 model Added.
NOV	2005	Added OPOS XY
MAR	2006	Added SummaCut D140 and S class dual TA heads
JUN	2006	OPOS BarCode
OCT	2006	Cutting Through P5 & P6
SEPT	2007	SummaCut-R added.
JAN	2008	Added new SummaUSB.dll
NOV	2008	Added local move commands
JAN	2009	Added Wireless Communication
JAN	2009	Added opos Cam
DEC	2009	Added opos panelled.
JAN	2013	Added tool commando "use current tool no FlexCut" - implemented rules described in 'recommended jobstrcuture' document in samples.
OCT	2013	Elaborated explanation for Ethernet connection. Updated list max velocity/pressure.
DEC	2013	Explanation 'gracefully shutdown socket' extract from MSDN added.
SEP	2015	added EW command deleted RELOAD command. OPOS XY line must be 3mm.
OCT	2015	added new option OPOS paneling (ON4.
AUG	2016	updates drawings
SEP	2016	updated EW command
FEB	2017	Discovering Ethernet devices
AUG	2017	typo in encapsulated command "_LENGTH"
JUN	2018	OPOS XY2 for S Class 2 (was forgotten in version JAN2020
MAY	2020	Added OPOS Xtra, reviewed section 4 on OPOS, uniform pictures and changed some notes.
AUG	2020	Removed old Window references, no more offset for OPOS XY line, Checked OPOS section for 'SPECIAL LOAD' command, removed section on none OPOS alignment modes.
DEC	2020	Removed old RS232 references, changed value of max marks in X direction from 64 to 128.

Notice

Please address all questions, comments or suggestions concerning this and other Summa manuals to:

Rochesterlaan 6
B-8470 Gistel
Belgium

Table of Contents

1 Introduction

The information contained in this package is designed to allow technically competent personnel to develop software and drivers compatible with the cutters from Summa.

1.1 Overview

The information is grouped in 4 principal sections, as follows:

- The vector cutting languages: DM/PL, HP-GL and HP-GL/2.
- The Summa cutter control language: "Encapsulated commands".
- Contour cutting of printed designs using OPOS Outline Cutting.
- Using extra functionality to cut through.
- Implementing USB-communication to control the Summa cutters.
- Information about the wireless communication.

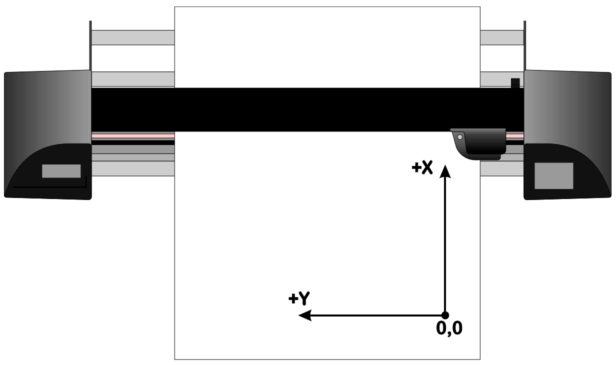
2 Cutter Commands

The cutter understands 3 languages: DM/PL, HP-GL and HP-GL/2. Because HP-GL and HP-GL/2 are almost identical, no difference is made in this document between both languages. Only a small subset of these languages is described. The cutters from Summa do support almost all existing Vector commands from these languages.

The cutter also has a specific language for changing the parameters of the cutters such as the velocity and pressure. This language is called encapsulated language.

The **X-axis** is the direction in which the **media** moves, the maximum size of media in X-direction is 50 meter.
The **Y-axis** is the direction in which the **cut-head** moves, the maximum size is dependent from model.

Illustration 2-1- *Media Orientation*



2.1 DM/PL Language Basics

2.1.1 Introduction.

In order to be fully compatible with the many features incorporated in the cutter itself, such as automatic flex-cut or replotting from the control panel, follow these guidelines:

1. Each DM/PL file can be preceded by encapsulated commands, for example to setup the parameters for OPOS.
2. Each DM/PL file must include following commands in the initialization string:
   - The ;: Select command.
   - The ECx command.
   - The A or R command.
3. Then some optional commands can be used such as the W command or the P Pen/tool select command. These can be used multiple times within a file. It is also possible to request the size of the loaded media, using the ER command.
4. Vector data using U and D commands follow
5. An end of file command is strongly recommended to terminate the file. The end of file commands are: **@**, **Z**, **Fx** or **e**-commands. The e command is recommended.
   If you don’t want to move to the end of cut sign, then use @ or Z command. However if you want to move to the end of the cut sign, then use the “e” command. These end of file commands are necessary if you want to use the flex-cut feature, the paneling feature, some of the OPOS features.
   With the Fx command you have more flexibility on where to move the knife at the end of the file.
6. For the S Class and newer SummaCut, you can optionally add a Cut-off command.

2.1.2 The ;: Select Command

The Select *;:* command (semicolon double point selects the cutter. When the cutter is selected it can receive information until it is deselected by the *e,@ or Z* command.

NOTE: This command is required before every any other data string.

2.1.3 The ECx Coordinate Addressing Command

The Co-ordinate Addressing *ECx* command sets user units to a specified resolution, raises the knife, moves it to the home position.
n selects the addressing resolution, (resolution of the x-y coordinates and is an alphanumeric expression 0, 1, 5, M, or N (see Table 2.1- Coordinate Addressing .

NOTE: This command is required at the start of the data string.

Command	**User Resolution**
EC1	0,001 inch
EC5	0,005 inch
ECM	0,1 mm
ECN	0,025 mm

Table 2.1 - Coordinate Addressing

2.1.4 The A Absolute Addressing Command

The absolute addressing *A* command selects the Absolute addressing mode. In this mode, all x-, y-coordinates are with respect to the origin at the lower left corner of the chart. The device remains in this mode until it receives a Relative *R* command or a deselect command (e, Z or @ .

Example: `;: EC1 A U 5000,5000 D 2000,2000 e`

In this example, the device is selected with the ;: command and 1/1000 inch resolution is selected with the Co-ordinate Addressing *EC1* command. Absolute knife positioning is selected with the *A* command. The knife is sent in up position to absolute co-ordinate 5000,5000. The knife is lowered with the Down *D* command and a line is cut to absolute co-ordinate 2000,2000. The device is deselected with a deselect *e* command.

Note: Before any vector data is processed the Addressing mode A or R MUST be specified. Otherwise, all x-, y-coordinates are ignored.

2.1.5 The R Relative Addressing Command

The Relative Addressing *R* command selects the relative addressing mode. In this mode, all x-, y-co-ordinates are relative to the present knife position. The device remains in this mode until it receives an Absolute Addressing *A* command or a deselect command (e, Z or @ .

Example: `;: EC1 R U 5000,5000 D 2000,2000 e`

In this example, the device is selected with the *;:* command and 1/1000 inch resolution is selected with the Co-ordinate Addressing *EC1* command. Relative knife positioning is selected with the *R* command. The knife is sent to relative co-ordinate 5000,5000. The knife is lowered and a line is cut to relative co-ordinate 2000,2000 (this is absolute co-ordinate 7000,7000. The device is deselected with the deselect *e* command.

Note: Before any vector data is processed the Addressing mode A or R MUST be specified. Otherwise, all x-, y-coordinates are ignored.

2.1.6 The D Down Command

The Down *D* command lowers the knife.
The knife remains in the down position until an Up *U*, Coordinate Addressing *EC*, Home *H*, Frame *F*, Window *W*, End of Plot *e*, or Reset Z command is received.

2.1.7 The U Up Command

The Up *U* command raises the tool (pen or knife.

2.1.8 The x,y Vector Move Command

This command causes the knife to move to a new user position specified by co-ordinate *x,y*. This new position is dependent on the present Absolute Addressing *A* or Relative Addressing *R* knife positioning command in effect, and the current user unit (see 2.1.4 The ECn Coordinate Addressing Command. In Absolute Addressing *A* mode, the new position is x,y user units from the present origin. In Relative Addressing *R* mode, the new position is *x,y* user units from the present position.

Example: `;: ECM A D 0,1000 1000,1000 1000,0 0,0 U e`

In the above sample, the cutter is selected, set to 0.1mm resolution, absolute addressing with the *ECM* command, then the tool is lowered and a square of 10 centimeters is cut. The tool is then raised and the device is deselected with the *e* command.

Note: All coordinates MUST be integer values only.

2.1.9 The e End Of Plot Command

For roll media the origin is moved several mm's beyond the last sign, more precise, beyond the last Down vector. The amount it moves behind the last cut sign is defined with the RECUT_OFFSET parameter (can be set via encapsulated language or via the keypad . The end of plot command also deselects the cutter. If new data needs to be sent, it must be preceded by the ;: select command.

The recut command (encapsulated or through keypad will work correctly when using this command. Note, that the recut command only recuts the last job. All other internal functions such as FlexCut. OPOS barcode, OPOS sheet mode, paneling will work with the e command.

2.1.10 The @ Deselect Command

The Deselect @ command sets the cutter deselected. After the device receives the command, it continues to process the data that is already in the buffer and does not accept new data until it is selected again.
The Deselect command does not affect any of the device's parameters, such as the resolution, window size, origin, etc…

This command will not move the origin to the end of the last sign, unless one of the following options was used: Flex-cut using P5 or P6 command, OPOS Barcode, paneling. The RECUT_OFFSET will be used to determine how far behind the last sign the origin will be moved.
The recut command (encapsulated or through keypad will work correctly when using this command.

2.1.11 The Z Reset Command

The reset Z command deselects the device and resets all parameters that were changed by DMPL commands.

This command will not move the origin to the end of the last sign. The @ is preferred above the Z command. The RECUT feature will work with the Z command, but the flex-cut, paneling and OPOS barcode features will not work when ending only with the Z command.

2.1.12 The BPn Tool Pressure Command

The Tool Pressure BPn command sets a desired pressure. The argument n specifies the pressure in grams. The numeric range for n is an integer from 0 to the maximum pressure, which depends on the device (see 7.3Maximum Pressure And Speed By Model.

2.1.13 The Fn Frame Command

The Frame *Fn* command is used to move the origin in X-axis direction. Upon receipt of this command the tool is raised, the media is advanced to the new home position, and the window and viewport limits (see 2.1.5 The W Window Command are s et to the default size.
n specifies the location of the new home position. This parameter is expressed in user units as set by the present co-ordinate addressing *EC* command. *n* can be positive or negative. If *n* is zero, the present position is defined as the new home position, but the media does not advance.
With *n* negative the media can be ejected out of the cutter, the cutter then goes in a not loaded condition; new media must then be loaded into the cutter.

Example: `;: EC1 F1000`

With *EC1* (see 2.1.4 The Ecn Coordinate Addressing Command the co -ordinate addressing is set to 1/1000 inch, the *F* command sets the new home position 1000 user units (= 1 inch from the present one, and advances accordingly the media with 1 inch.

The Frame command will act as an e-command when flex-cut or paneling is activated. So it will end at the end of the last sign (down vector + RECUT_OFFSET.

2.1.14 The W Window Command

The **optional** window command is used for scaling and/or clipping.

The Window *W wxll,wyll,wxur,wyur,vpxll,vpyll,vpxur,vpyur* sets the device's window and viewport limit co-ordinates. Each is set by specifying lower left and upper right corner points for the desired window or viewport.

wxll, *wyll* specifies the lower left window limit x-, y-coordinate.
wxur, *wyur* specifies the upper right window limit x-, y-coordinate.
vpxll, *vpyll* specifies the lower left viewport limit x-, y-coordinate.
vpxur, *vpyur* specifies the upper right limit x-, y-coordinate.

The units for the co-ordinate values are set with the present Co-ordinate Addressing *EC* command. The default window is the entire maximum cut size. After a *W* command is entered it remains in effect until a new *W* command is issued. The settings are defaulted after an *ECn, Fn, EF, EH, Z* or *e* command is received or after a load procedure issued from the control panel.

A window may be set that tells the cutter what portion of a sign to be cut out by specifying a rectangular area using the lower left *wxll*, *wyll* and upper right *wxur*, *wyur* points.

The viewport *vpxll*, *vpyll*, *vpxur*, *vpyur* specifies an area for the cutter to reproduce the window's sign data. If the area within the viewport limits is equal to the area within the window limits, the size of the sign is equal to the size of the original. If the area within the viewport limits is smaller than the area of the window, then the sign is scaled down. If the area within the viewport limits is larger than the area of the window, then the sign is scaled up. If the viewport limits are not set in proportion with the window limits, the aspect of the plot is changed.

Example: `;: EC1 W 0,0 5000,5000 0,0 10000, 10000`

The *EC1* command sets the user units to 0.001 inch (see 2.1.4 The ECn Coordinate Addressing Command.

The W command sets a 5 X 5 inch window mapped to a viewport of 10 X 10 inch. This effectively sets the user units to 0.002 inch since the viewport is twice the size of the window.

2.1.15 The Px Tool Select Command

The Tool Select *Px* Command selects a given tool. The cutter will pause at execution of the command, and allows manual selection of the tool by the operator if the selected tool is different from the one being used.

The to ol command P5 and P6 turns on the flex-cut option, no intervention of the operator will be needed. When using P5 or P6 (P6 is recommended then an end of file command must be included, the origin will always be moved behind the plot when the job is done.

For the dual head, that incorporates a knife and a pen, no operator intervention will be needed when switching between knife (P1 and the extra pen (P9.

Following table describes the tool-command necessary to select a tool, depending on the cutter-head that the cutter is equipped with.

	Tool selected on machine equipped with :		
Tool selection command	**Drag head**	**Tangential head**	**Dual cutting head equipped with an extra pen-adapter**
P100 U	Use current installed tool		
P0	Drag knife	Tangential knife	Tangential knife
P1	Drag knife	Tangential knife	Tangential knife
P2	Pen	Ballpoint	Ballpoint
P3	Pouncer	Pouncer	Pouncer
P4	x	x	Extra pen
P5	FlexCut(1 Fast mode	FlexCut(1 Fast mode	FlexCut(1 Fast mode with tangential knife
P6	FlexCut(1 Accurate mode	FlexCut(1 Accurate mode	FlexCut(1 Accurate mode with tangential knife
P7	x	x	Extra pen (P9 preferred
P8	x	Drag knife	Drag knife
P9	x	x	Extra pen
P0+	x	Drag knife	Drag knife
P1+	x	Drag knife	Drag knife
P2+	x	x	Extra pen (P9 preferred
P3+ ... P8+	x	x	x

Table 2.2 - DM/PL Tool Select Command

(1 Cutting through media , see Chapter 5

2.1.16 The Vn Velocity Command

The Velocity *Vn* command sets the axial velocity of the tool (down position speed for cutting or plotting. The integer argument n specifies the tool down velocity of the device and depends on the current co-ordinate addressing:

- If the present Co-ordinate Addressing is *EC0*, *EC1* or *EC5*, then the units are in inches per second.

  Example: `;: EC1 A V5 U 0,0 D 0,2500 2500,2500 2500,0 0,0 U e`

  In the above sample, the cutter is selected, set to 1/1000-inch resolution, absolute addressing, a velocity of 5 inches per second with the *V5* command, then the tool is lowered and a square of 2.5 inches is cut. The tool is then raised and the device is deselected with the *e* command.

- If the present Co-ordinate Addressing is *ECM* or *ECN*, the units are in centimeters per second.

  Example: `;: ECM A V50 U 0,0 D 0,2500 2500,2500 2500,0 0,0 U e`

  In this sample, the cutter is again selected, the resolution is now set to 0.1 millimeter, and the velocity is then set to 50 centimeter per second with the *V50* command.

2.1.17 The ER Report Command

The *ER* Report commands allow the device to send information back to the host computer and software through the serial port. The Report *ER* command allows the device to sent its present status to the computer. **It is mainly used to retrieve the size of the inserted media.**

When the device processes a Report *ER* command, the following information is transmitted:

- Status Byte One indicates the number of the last selected pen/knife, the present up or down status of the pen/knife, whether or not the present location of the knife on the cutting surface is inside the window limits, and the present full or half chart format (large or small. Se e also Table 2.4- ER Report Format. Note that bit seven is always set to 0.

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	0 = Large Chart 1 = Small Chart	0 = Inside Window 1 = Outside Window	0 = Pen Up 1 = Pen Down	Pen Number (MSB	Pen Number	Pen Number	Pen Number (LSB

Table 2.3 - Status Byte One

- Status Byte Two is not used (Reserved.
- The remaining bytes indicate: the present x-, y-co-ordinate position of the knife, the present window limit co-ordinates, and the present viewport limit co-ordinates.

Note : All co-ordinates are in user units (see 2.1.4 The Ecn Coordinate Addressing Command and relative to the fixed origin of the present chart size. Positive co-ordinates values are signed with a space and negative co-ordinates with a minus.

The report information is sent in ASCII BCD (binary-coded decimal format. The values that make up the status string are decimal integers. The digits of each decimal integer are represented by their ASCII equivalents. It is the responsibility of the receiving software to convert the ASCII representations of the decimal integers to binary decimal integers before performing any mathematical operations on them.
The report is always enclosed within parentheses (and followed by a carriage return. Commas separate the values. The sign of each x-, y-co-ordinate immediately precedes the co-ordinate value. A space indicates a positive value, while a minus (- indicate s a negative value. Each status byte is three digits in length. The x,y co-ordinate length is 7 digits. Leading zeros are used as required to pad the x-, y-co-ordinate lengths to their appropriate lengths. The total string length is 100 characters.

Summary of data sent :

(	StatusByte 1	Status byte2	Current X pos	Current Y pos	Window Lower Left X	Window Lower Left Y	Window Upper Right X	Window Upper Right Y	Viewport Lower Left X	Viewport Lower Left y	Viewport Upper Left X	Viewport lower Left X	CR

Table 2.4 - ER Report Format

Example : `;: ECN ER`

In this example, the device is selected with the In this example, the device is selected with the *;:* command and 0.025mm resolution is selected with the Co-ordinate Addressing *ECN* command. The cutter is then commanded to send a report with the *ER* command.

ER returns:
```
(017,084,  0001000,  0002000,  0000000,  0000000,  2000000,  0014650,
0000000, 0000000, 2000000, 0014650)<CR>
```

In this report we can analyze the following:

- Status Byte One (017 shows that tool 1 was the last requested tool and it is presently in the down position inside the present window limits.
- Status Byte Two (084 is reserved and not used.
- Current X Position (0001000 : the present tool -position x-co-ordinate is
  1000 * 0.025mm = 25 mm.
- Current Y Position (0002000 : the present tool -position y-co-ordinate is
  2000 * 0.025 mm = 50 mm.
- Window Lower Left X (0000000 defines the lower left x -co-ordinate of the window at 0.
- Window Lower Left Y (0000000 defines the lower left y -co-ordinate of the window at 0.
- Window Upper Right X (2000000 defines the upper ri ght x-co-ordinate of the window at 2000000 * 0.025mm = 50000mm = 50m.
- Window Upper Right Y (0014650 defines the upper right x -co-ordinate of the window at 14650 * 0.025mm = 366.25mm.
- Viewport Lower Left X (0000000 defines the lower left x -co-ordinate of the viewport at 0.
- Viewport Lower Left Y (0000000 defines the lower left y -co-ordinate of the viewport at 0.
- Viewport Upper Right X (2000000 defines the upper right x -co-ordinate of the viewport at 2000000 * 0.025mm = 50000mm = 50m.
- Viewport Upper Right Y (0014650 defines the upper right x -co-ordinate of the viewport at

14650 * 0.025mm = 366.25mm.

- A carriage return <*CR*> terminates the report.

Summarized we can say that a knife was selected and that it is in the down position, at x-y location 25mm and 50mm from the origin. There is media loaded which is 366.25mm wide and 50m long. As this is the available media-size, actual media-size may differ, because we need some space for the pinch-rollers to hold the media. There is also an assumption that a full roll of vinyl was present.

2.1.18 The EP Panel Report Command

When paneling is ON, this will return current panel bounds in the same format as the ER command, otherwise it works like ER command.

2.1.19 The c "Cut off media" command

The S class cutters have a "cut off knife" to cut off a sheet out of a roll. This can be added for instance after the e command. The media will be cut through at the current position increased with the CUTMEDIA_OFFSET. This parameter can be set through encapsulated language (see further or through the control panel. A new origin will be set afterwards at the lower right corner of the media. If used after an e, @ or F-command, you must add the selection command ;: before the c command.

Example: `;: EC1 A U 1000,1000 D 2000,2000 2000,0 e ;: c`

- **NOTE:** When using the Cut-off option in combination with OPOS, then the distance between 2 OPOS jobs must be at least 30 mm.

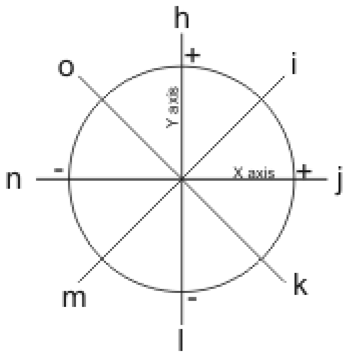
2.1.20 The EW command

The "job length" EW x [,y] sets the length of the job. x specifies the length of the job. This parameter is expressed in user units as set by the co-ordinate addressing EC command. The EW command should be sent before any cut data and after the addressing command (see 2.1.4.
When the cutter receives this command, it will pre-load the distance x, if the media (sheet or end of roll is not big enough, then the firmware will show a warning "job does not fit media area, press exit to abort". If exit/abort is pressed, then the job will be cancelled, so that no media is lost.
If the y is bigger than the media width, then the same warning will be shown. The y is optional.
This command should only be used for non-OPOS jobs. Using the EW command with OPOS jobs makes no sense. OPOS preloads the complete job always unless OPOS panelling is set off.
If OPOS panelling is set to on, then this command would cancel out the benefits from this setting (The biggest advantage of OPOS paneling is that it does not preload the complete job.

Example: ;: EC1 EW40000
With EC1 (see 2.1.4 The ECn Coordinate Addressing Command the co -ordinate addressing is set to 1/1000 inch, the EW command will unroll the media so that a job of 40000 user units (= 40 inch fits on the unrolled media

2.1.21 Local move commands

When a local move command is sent to a Summa cutter, the cutter will start moving it's cutting head in the direction of the specified command, until the cutter receives an end character "#". This is not a standard DMPL command, but an extension for the S class.



DM/PL command	Cutter movement
h	+Y
i	+X,+Y
j	+X
k	+X,-Y
l	-Y
m	-X,-Y
n	-X
o	-X,+Y

2.1.22 The a vector ready command

Following this command all the vectors in the queue will be processed and a '1' will be returned when all vectors are cut. This is not a standard DMPL command, but an extension for the S class.

2.1.23 Sample DM-PL File

```
;: EC1 A U 1000,1000 D 2000,2000 2000,0 e ;: c
```

In this example, the device is selected with the *;:* command and 1/1000 inch resolution is selected with the Co-ordinate Addressing *EC1* command. Absolute knife positioning is selected with the *A* command. The knife is sent in up position to absolute co-ordinate *1000,1000*. The knife is lowered with the Down *D* command and a line is cut to absolute co-ordinate *2000,2000.* Then a line is cut to position *2000,0*. The media is moved to the end of the cut area by the *e* command, so that the next sign will not be placed above the current one. This command also deselects the device. If the cutter has a cut-off knife it will cut-off a sheet, this is indicated with the c command.

2.2 HP-GL Language Basics

Only a subset of HP-GL and HP-GL/2 are presented in this document, the cutters are fully HP-GL and HP-GL/2 compatible, with exception to some commands that a cutter cannot perform (e.g. printing bitmaps with HP-RTL
There is no need for an select command, most often HP-GL files start with an *IN;* or *BP;* command.
The standard resolution for HP–GL is 0.025 mm.
All commands must end with a semicolon;
It is strongly recommended to end a file with a PG; command, otherwise functions such as recut, flex-cut, paneling, OPOS barcode will not work.

2.2.1 The IN; Initialize Command

The *IN;* initialize command resets all parameters that were changed by HP-GL commands.

2.2.2 The BP; Begin Plot Command (only HP–GL/2

The *BP;* Begin Plot command resets all parameters that were changed by HP-GL/2 commands and move the origin to the end of the previous cut sign.

2.2.3 The PAx,y; Absolute Addressing Command

The absolute addressing *PA;* command selects the Absolute addressing mode. In this mode, all x-, y-co-ordinates are with respect to the origin at the lower left corner of the chart. The device remains in this mode until it receives a Relative *PR* command.

Example: `IN; PA; PU 5000,5000;PD 2000,2000;`

In this example, The cutter is initialized with the *IN;* command. Absolute knife positioning is selected with the *PA;* command. The knife is sent in up position to absolute co-ordinate 5000,5000. The knife is lowered with the Down *PD* command and a line is cut to absolute co-ordinate 2000,2000.

2.2.4 The PRx,y; Relative Addressing Command

The relative *PR;* command selects the Relative addressing mode. In this mode, all x-, y-co-ordinates are relative to the present knife position. The device remains in this mode until it receives an Absolute *PA* command.

2.2.5 The PDx,y; Down Command

The Down *PD;* command lowers the knife. This command causes the knife to move to a new user position specified by co-ordinate x,y. This new position is dependent on the present Absolute *PA* or Relative *PR* knife positioning command in effect. In Absolute *PA* mode, the new position is x,y user units from the present origin. In Relative *PR* mode, the new position is x,y user units from the present position.

2.2.6 The PUx,y; Up Command

The Up *PU;* command raises the knife.

This command causes the knife to move to a new user position specified by co-ordinate x,y. This new position is dependent on the present Absolute *PA* or Relative *PR* knife positioning command in effect. In Absolute *PA* mode, the new position is x,y user units from the present origin. In Relative *PR* mode, the new position is x,y user units from the present position.

2.2.7 The VSn; Velocity Command

To set the up and down velocity. n specifies the velocity of the device in cm/s.

2.2.8 The SPn; Pen/Knife Select Command

The *SPn;* command selects a given tool. (e.g. knife pen or pouncer. The cutter will pause at execution of the command, and allows manual selection of the tool by the operator if the selected tool is different from the one being used.

The tool command SP5; and SP6; turns on the flex-cut option, no intervention of the operator will be needed. When using P5 or P6 (P6 is recommended then an end of file command must be included, the origin will always be moved behind the plot when the job is done.

For the dual head, that incorporates a knife and a pen, no operator intervention will be needed when switching between knife (P1 and the extra pen (P9.

Following table describes the tool-command necessary to select a tool, depending on the cutter-head that the cutter is equipped with.

Table 2.5 - HP-GL Tool Select Command describes the tool-command necessary to select a tool, depending on the cutter-head that the cutter is equipped with.

	Tool selected on machine equipped with :		
Tool Selection Command	**Drag head**	**Tangential head**	**Dual Cutting head equipped with an extra pen-adapter**
P100 U	Use current installed tool		
SP0;	Drag knife	Tangential knife	Tangential knife
SP1;	Drag knife	Tangential knife	Tangential knife
SP2;	Pen	Ballpoint	Ballpoint
SP3;	Pouncer	Pouncer	Pouncer
SP4;	x	x	Extra pen
SP5;	FlexCut(1 Fast mode	FlexCut(1 Fast mode	FlexCut(1 Fast mode with tangential knife
SP6;	FlexCut Accurate mode	FlexCut Accurate mode	FlexCut Accurate mode with tangential knife
SP7;	x	x	Extra Pen(SP9 is preferred
SP8;	x	Drag knife	Drag knife
SP9;	x	x	Extra Pen

Table 2. 5 - HP-GL Tool Select Command

(1 Cutting through media see Chapter 5

2.2.9 The FSn; Force Select Command

The Force Select FSn; command sets the tool pressure. It is expressed ingrams. The numeric range for n is an integer from 0 to the maximumpressure that depends on the device (see 7.3 Maximum Pressure AndSpeed By Model.

2.2.10 The OH; Output Hardclip Command

The *OH;* output hardclip command returns the size of the loaded media.
Data returned by the cutter:

Window Lower Left X	Window Lower Left Y	Window Upper Right X	Window Upper Right Y	Carriage Return

Table 2.6 - OH Report Format

2.2.11 The PG; End Of Plot Command

The end of plot *PG;* command allows multi-chart plotting. For roll media the origin is moved one mm beyond the last sign, so that a new sign is not cut on the previous one.

2.2.12 The EC; Enable Cut off Command

The S class cutters have a "cut off knife" to cut off a sheet out of a roll. Sending the EC command will enable this function after a PG or AH or AF command.
So to Cut off the media, send the design, then the EC command and finally the PG; command.

- **NOTE:** When using the Cut-off option in combination with OPOS, then the distance between 2 OPOS jobs must be at least 30 mm.

Note : to disable the Cutoff command after it was enabled, send EC1.

2.2.13 Sample HP-GL File

Example: `IN; PA;PU1000,1000;PD2000,2000;PD 2000,0;PG;`

In this example, the IN; resets the device. Absolute knife positioning is selected with the *PA;* command. The knife is sent in up position to absolute co-ordinate 1000,1000. The knife is lowered with the Down *PD* command and a line is cut to absolute co-ordinate 2000,2000 then a line is cut to position 2000,0. The media is moved to the end of the cut area by the *PG;* command, so that the next sign will not be placed above the current one.

3 Encapsulated Language

The cutter control commands listed below constitute the machine's native control language.
The encapsulated commands are given in the form:

COMMAND option1 | option2 | option3

Where "command" is the command to type in (including spaces and the "=" symbol and "option1", "option2", etc. stand for the values or arguments that work with the command. You can only use one argument at a time. Alternative arguments are shown separated by "|" which is not part of the command or its argument. End each command with a period, or with a carriage return/line feed pair (generated by pressing the <Enter> key on PC-compatible computers.
Possible commands are:

```
SET item_name = option.
END.
MENU [ menu item ].
QUERY.
SET_ORIGIN.
LOAD_MARKERS.
```

The command interpreter is activated when the cutter receives the following sequence of characters:

```
<esc>;@:
```

The sequence begins with the escape character (ASCII 27 in decimal, followed by ASCII 58, 64 and 59. When activated, the command interpreter responds with "READY". It prompts you to enter commands with the ">" symbol.

```
END.
```

After accessing the machines command interpreter, send an END command to the machine to exit the command interpreter. During the dialogue with the machine, send an END command to end the dialogue session. When creating a data encapsulation file, put "END" immediately before the beginning of the data to plot. A period, rather than a carriage return/line feed pair, is the preferred way to terminate the END command in a data encapsulation file, since it provides a visible marker at the end of the encapsulation header.

Note : *After accessing the machines command interpreter, be sure to send an "END." command before sending any data to cut. Otherwise the machine will try to interpret the data as encapsulated control commands, with unpredictable results.*

3.1 Changing Parameters Settings Commands

The **SET** command is used to set the menu items. For more information about the menu items, check the cutter user manual.

3.1.1 MARKER_X_DIS = [a number in the range 1200 to 52000]

Allows for the marker y distance to be specified in 0.025 mm units.

```
<esc>;@:
SET MARKER_X_DIS=2400.
END.
```

This sample defines the X-markers 60 mm apart.

3.1.2 MARKER_Y_DIS = [a number in the range 1200 to 64000]

Allows for the marker y distance to be specified in 0.025 mm units.

```
<esc>;@:
SET MARKER_Y_DIS=4800.
END.
```

3.1.3 MARKER_X_SIZE = [a number in the range 48 to 400]

Allows for the marker x size to be specified in 0.025 mm units.
The default value is 60.

```
<esc>;@:
SET MARKER_X_SIZE=100.
END.
```

This defines the marker size in the X-direction as 2.5 mm.

3.1.4 MARKER_Y_SIZE = [a number in the range 48 to 400]

Allows for the marker y size to be specified in 0.025 mm units.
The default value is 60.

```
<esc>;@:
SET MARKER_Y_SIZE=100.
END.
```

3.1.5 MARKER_X_N = [a number in the range 2 to 128]

Allows for the number of markers along the x axis to be specified. When the special load procedure is called with MARKER_X_N different from 0, then the specified number of markers will be searched for.
If however MARKER_X_N is zero or not specified then the cutter will search for markers until the end of media is found or until the maximum of 64 markers is reached.

```
<esc>;@:
SET MARKER_X_N=6.
END.
```

3.1.6 SPECIAL_LOAD = OPOS |OPOS_XY|OPOS_XY2|OPOS_XTRA

Use this to set select which kind of special loading will be used when calling the special load feature from the keyboard or with the "LOAD_MARKERS" command.
Legacy products will not support the OPOS XY options and newer models do not longer support the none OPOS alignment methods (XY_ADJUST, XY_ALIGN en X_ALIGN

3.1.7 OPOS_SHEET_MODE = [OFF,ON]

Enables the sheet mode for OPOS. When turned on the "SPECIAL_LOAD" procedure will be started automatically from the 2nd sheet on. See the section "AUTOMATING OPOS" for more details.
The default value is OFF.

```
<esc>;@:
SET OPOS_SHEET_MODE ON.
END.
```

3.1.8 OPOS_LEVEL = [a number between 0 and 255]

Sets the level of marker recognition, this is normally measured by the "CALIBRATE MEDIA" test. This level compensates for differences between different kinds of media.
Th default value depends on model and firmware revision.

3.1.9 OPOS_ORIGIN= MARK|XY_LINE|CURRENT_POSITION|CENTER_OF_MEDIA

The cutter can ask the user to set the tool on the sensor or under the XY line. With 'current position' or 'center of media' there is even no user intervention needed after an OPOS load command.

3.1.10 OPOS_PANELLING = [OFF,ON,ON4]

Enables to read the OPOS markers per panel. See the OPOS section (section 4 for more details.

```
<esc>;@:
SET OPOS_PANELLING ON.
END.
```

3.1.11 PANELLING = OFF | ON

Defines if the machine should panel the data.

3.1.12 PANELLING_SIZE = [a number in the range 2 to 250]

Sets the panel-size in cm.

3.1.13 PANEL_REPLOT = [a number in the range 0 to 99]

Defines how many time the cutlines in a panel will be replot. If set to n, the lines will be cut 1+n times.
Default is 0.

3.1.14 RECUT_OFFSET = [a number in the range 0 to 4000]

Sets the offset between 2 'recut' jobs **in mm**.

3.1.15 CUTMEDIA_OFFSET = [a number in the range 0 to 255]

Sets the offset used to cut off a sheet after a job, only available in the newer cutters.

3.1.16 VELOCITY = 50 |100 |150 |200 |250 | 300 | 350 | 400 | 450 | 500 | 550 | 600 |700 |800 |900 |1000

Use this command to specify the velocity in mm/s.
The maximum speed depends on the model (See also Table 7.3 - Maximum Pressure And Speed By Model. If an invalid value is used the command will be ignored and machine velocity will not change. Accepted values for specific model can be requested by using the MENU command (see 3.3.2 .

```
<esc>;@:
SET VELOCITY=600.
END.
```

3.1.17 OVERCUT = [a number in the range 0 to 10]

Use this command to set overcut. One unit is about 0.1 mm.
The default value is 1.

3.1.18 OPTICUT = ON | OFF

Use this command to set the OptiCut.

3.1.19 TOOL = PEN | DRAG_KNIFE | POUNCER (for the D series cutters

For the T series cutters:
TOOL = BALLPOINT | T_DRAG_KNIFE | TANGENTIAL_KNIFE | POUNCER

For the cutters with dual head:
TOOL = BALLPOINT | T_DRAG_KNIFE | TANGENTIAL_KNIFE | EXTRA_PEN | POUNCER

Use this command to select the cutter's tool.

3.1.20 HPGL_ORIGIN = CENTER | RIGHT_FRONT

The HPGL_ORIGIN menu-item is only effective if EMULATION is set to HPGL or HPGL2 (see 3.1.11 The HP/GL ORIGIN submenu is used to set the origin in the center or the bottom-right corner of the loaded media.

3.1.21 FLEX_CUT = OFF | MODE1 | MODE2

Use this to enable FLEX cutting mode, set to mode 1 or mode 2.

3.1.22 FULL_PRESSURE = [a number in the range 0 to 600 (step 5]

Sets the full pressure value for FlexCut mode.

3.1.23 CUT_LENGTH = [a number in the range 10 to 10000]

Sets the length value to cut at full pressure for FlexCut mode specified in 0.025 mm units.

3.1.24 FLEX_PRESSURE = [a number in the range 0 to 600 (increments of 5]

Sets the flex pressure value for FlexCut mode (reduced pressure .

3.1.25 FLEX_LENGTH = [a number in the range 10 to 10000]

Sets the length value to cut at flex pressure for FlexCut mode specified in 0.025 mm units.

3.1.26 FLEX_VELOCITY = 50|100|200|300|400|500|600|700|800|900|1000|AUTO

Use this command to specify the velocity when cutting through in flex-mode, this in mm/s. The maximum speed depends on the model (see Table 7.3 - Maximum Pressure And Speed By Model on page 66. When set to Auto, it will follow the VELOCITY settings.

3.1.27 SORTING_ENABLE = OFF | ON | START_POINT

Defines if vector optimization is done or not.

3.1.28 UP_VELOCITY = 50 |100 |150 |200 | 300 | 400 | 500 | 600 |700 |800 |900 |1000 | AUTO

Use this command to specify the velocity of vectors in up position, this in mm/s.
The maximum speed depends on the model (see Table 7.3 - Maximum Pressure And Speed By Model on page 66. When set to Auto, it will follow the VELOCITY settings.

3.1.29 UP_ACCELERATION_ = 1 |2 |3 |5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | AUTO

Use this command to specify the acceleration of vectors in up position, this in 0.1 g. When set to Auto, it will follow the VELOCITY settings.

3.1.30 DOWN_ACCELERATION_ = 1 |2 |3 |5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | AUTO

Use this command to specify the acceleration of vectors in up position, this in 0.1 g.
When set to Auto, it will follow the VELOCITY settings.

3.1.31 X_CALIBRATION = [a number in the range 0 to 65535]

Defines the calibration factor for the X-axis (media movement. Mechanical tolerances from the media and/or the cutter can be compensated using this command.
To calibrate the cutter, draw a line of known length. Measure the length of the drawn line and use the following formula to calculate the new calibration factor:

$$N = \frac{65536*(D + \frac{C*D}{65536})}{M}$$

where:
N = New calibration factor to be computed.
D = The intended length of the line.
C = The current calibration factor, used while drawing the line.
M = The measured length of the line.
NewX = (**short unsigned int**(65536.0 * ((**double**DrawnX + (**double**OldX * (**double**DrawnX / 65536.0 / (**double**MeasuredX ;

```
<esc>;@:
SET X_CALIBRATION=64188.
END.
```

Un-calibrated cutters default to 64197.

***Note:** In the X-axis, draw the calibration-line in the middle of the media to compensate for left-to-right mechanical tolerance-differences.*

3.1.32 Y_CALIBRATION = [a number in the range 0 to 65535]

Defines the calibration factor for the Y-axis (cutter-head movement. Mechanical tolerances from the media and/or the cutter can be compensated using this command.
To calibrate the Y-axis, use the same method as explained in the X_CALIBRATION command. Un-calibrated cutters default to 0.

***Note :** Make calibration-lines long enough to minimize the effect of measuring-inaccuracies.*

3.2 Executive Commands

3.2.1 SET_ORIGIN = X,Y

This command moves the origin relatively from the current position. If no X and Y are specified the current position will be the new origin. X/Y can be either positive or negative. The unit of the coordinates is 0.025 mm.

```
<esc>;@:
SET_ORIGIN=0,6000.
END.
```

3.2.2 LOAD_MARKERS

Allows the user to execute the "SPECIAL LOAD" procedure otherwise activated by pressing the "RESET/LOAD" key a few times on the cutter's keypad.
If the LOAD_MARKERS procedure is aborted or the cutter failed to sense all the markers properly this command will return "ERROR : LOAD_MARKERS ABORTED OR MARKERS NOT PROPERLY SENSED."
If the LOAD_MARKERS procedure properly senses all markers, "MARKERS LOADED." will be returned.

```
<esc>;@:
LOAD_MARKERS.
END.
```

3.2.3 SET_CAM_ORIGIN.

When sending `<ESC>;@: SET_CAM_ORIGIN.END.`

The cutter will ask for the user to set the knife above the first marker. When pressing the apply button, the cutter will return "ORIGIN_LOADED" and the head will move its camera above the first marker. Note that the full width of the cutter is now available, now that the head is moved above the first marker. It is the task of the software to detect the marker, to remember its position, to go to the other markers, to detect them and to do all necessary transformations on the cutting data.

NOTE: *ONLY FOR OPOS-CAM feature*

3.2.4 EXTENDED_LOAD.

This will set the origin at the right side of machine, past the pinch rollers, so that entire width of the machine can be used when moving the head.

NOTE: *ONLY FOR OPOS-CAM feature*

3.2.5 SETSYS camera_led 0|1|2

Controls the light/LED on the CAM-OPOS:
0 = LED is off
1 = LED is ON
2 = LED is turned on or off in function of OPOS procedure. So it will beturned on after a LOAD_MARKERS has been initiated and turned off after the markers have been red. It is advised to turn off the led after markers has been read.

NOTE: *ONLY FOR OPOS-CAM feature*

3.2.6 RECUT n

Recuts the last file n times.

```
<esc>;@:
RECUT 12.
END.
```

3.3 Response Commands

Response commands are used to elicit information from the cutter. Response commands are useful only during dialogues with the cutter's command interpreter, in which the cutter's responses can be received. See Dialogue Example below for an example of their use.

3.3.1 MENU.

When the command interpreter receives the MENU command by itself, the interpreter responds with the following information:

item count ITEMS-
item name1 : type = current value
item name2 : type = current value
...
where "item count" is the number of configuration settings: "item name1", "item name2", etc. are the names of the settings; "type" is the type of value allowed for each setting; and "current value" is the current value for each setting.

Send the following to the cutter:

```
<esc>;@:
MENU.
END.
```

This could result in the following prompt, dependent of the capabilities of the connected cutter:

```
READY.

>54 ITEMS-
  KNIFE_PRESSURE : numeric{0..600} = 50
    PEN_PRESSURE : numeric{0..600} = 80
     DRAG_OFFSET : numeric{0..100} = 43
        VELOCITY : enumtext{50,100,200,300,400,500,600,700,800,900,1000} = 800
         OVERCUT : numeric{0..10} = 0
   CONCATENATION : numeric{0..20} = 0
         OPTICUT : enumtext{OFF,ON} = OFF
       SMOOTHING : enumtext{OFF,ON} = OFF
       EMULATION : enumtext{DMPL,HPGL,HPGL_2,AUTO} = AUTO
            TOOL : enumtext{BALLPOINT,T_DRAG_KNIFE,TANGENTIAL_KNIFE,POUNCER} = T_DRAG_KNIFE
      MENU_UNITS : enumtext{ENGLISH,METRIC} = METRIC
  DMPLADDRESSING : enumtext{EC1,EC5,ECN,ECM,EC0} = ECN
     HPGL_ORIGIN : enumtext{CENTER,RIGHT_FRONT} = RIGHT_FRONT
       BAUD_RATE : enumtext{2400,4800,9600,19200,38400,57600} = 38400
          PARITY : enumtext{NONE,MARK,EVEN,ODD} = NONE
         RTS/DTR : enumtext{TOGGLE,HIGH} = TOGGLE
   TOOL_COMMANDS : enumtext{ACCEPT,IGNORE} = ACCEPT
        AUTOLOAD : enumtext{ON,OFF,ASK} = ON
    SPECIAL_LOAD : enumtext{OPOS,XY_ADJUST,XY_ALIGN,X_ALIGN,OPOS_XY,OPOS_BARCODE,FORCE_OPOSXY} =
OPOS
 OPOS_SHEET_MODE : enumtext{OFF,ON} = OFF
        FLEX_CUT : enumtext{OFF,MODE1,MODE2} = OFF
   FULL_PRESSURE : numeric{0..600} = 70
      CUT_LENGTH : numeric{10..10000} = 144
   FLEX_PRESSURE : numeric{0..600} = 30
     FLEX_LENGTH : numeric{10..10000} = 19
   FLEX_VELOCITY : enumtext{50,100,200,300,400,500,600,700,800,900,1000,AUTO} = AUTO
      CONFIGUSER : numeric{1..8} = 1
     MEDIA_SENSE : enumtext{OFF,ON,FRONT_OFF} = FRONT_OFF
    MARKER_X_DIS : numeric{1200..52000} = 2400
    MARKER_Y_DIS : numeric{1200..64000} = 21600
   MARKER_X_SIZE : numeric{48..400} = 80
   MARKER_Y_SIZE : numeric{48..400} = 80
```

```
       MARKER_X_N : numeric{2..128} = 14
     40G_PRESSURE : numeric{0..250} = 26
    400G_PRESSURE : numeric{0..250} = 200
     DRAG_LANDING : numeric{0..250} = 17
      PEN_LANDING : numeric{0..250} = 17
     TANG_LANDING : numeric{0..250} = 17
     RECUT_OFFSET : numeric{0..4000} = 0
  CUTMEDIA_OFFSET : numeric{0..255} = 0
         LANGUAGE : enumtext{ENGLISH,FRENCH,GERMAN,SPANISH,ITALIAN,DUTCH,POLISH,CZECH} = ENGLISH
   POUNC_PRESSURE : numeric{0..600} = 120
        POUNC_GAP : numeric{1..50} = 1
      UP_VELOCITY : enumtext{50,100,200,300,400,500,600,700,800,900,1000,AUTO} = AUTO
UP_ACCELERATION_ : enumtext{1,2,3,5,10,20,25,30,35,40,AUTO} = AUTO
DOWN_ACCELERATION_ : enumtext{1,2,3,5,10,20,25,30,35,40,AUTO} = AUTO
  VEL_LONG_VECTOR : enumtext{AUTOMATIC,NORMAL} = AUTOMATIC
         TURBOCUT : enumtext{QUALITY,SPEED} = SPEED
   QUALITY_TUNING : enumtext{OFF,ON} = ON
          ROLL_UP : enumtext{OFF,END_OF_JOB,END_OF_PANEL} = OFF
        PANELLING : enumtext{OFF,ON} = OFF
   PANELLING_SIZE : numeric{0..250} = 50
     PANEL_REPLOT : numeric{0..99} = 0
   SORTING_ENABLE : enumtext{OFF,ON} = OFF

      >
```

3.3.2 MENU [item name].

When you add the name of a setting (item name to the MENU command, the command interpreter responds with the type of value and the current value for the setting named:

item name: type = current value

Send this:

```
<esc>;@:
MENU VELOCITY.
END.
```

to receive this :

```
READY

>
    VELOCITY  :  enumtext{50,100,200,300,400,500,600,700,800,900,1000}  =
600
>
```

3.3.3 QUERY.

When you send the QUERY command, the command interpreter responds with the cutter's model name and ROM number.

Send the following to the cutter :

```
<esc>;@:
QUERY.
END.
```

You will receive something like this :

```
>
T610_PRO
9955017 9955017 1473001
>
```

3.4 Encapsulated File Example

The following is an example of a data encapsulation file header, placed before a plot. The encapsulation header contains cutter control commands that configure the cutter for a plot following the header. Comments in *italics* are not part of the example file.

```
<esc>;@:                     Initialize encapsulation
SET VELOCITY = 600.          Set velocity to 600 mm/s
END.                         Terminate encapsulation
```

Note : *In this example, a period, rather than a carriage return/line feed pair, has been used to terminate a command. The init-string doesn't need a period.*

4 OPOS Outline Cutting

4.1 Introduction

Summa supports several systems for positioning / aligning media in their cutting machines. These alignment systems are used to do the contour cutting of printed designs. The most accurate and automatic method is called OPOS.
This chapter first describes shortly the principle of the OPOS alignment systems and which support the sign-making software should include.
The second part describes in detail how to implement the OPOS alignment system in the sign-making software.
The last part describes the other 3 manual alignment schemes.
This chapter is mainly targeted for sign-making software that supports both printing and cutting. Implementing the OPOS system can add a surplus value to the software.

4.2 Contour Cutting Problems

Depending on the selected alignment method, the cutters can counterbalance the following irregularities:

1. Rotated Design

If the printed design is not loaded straight into the unit, the contour can be rotated equally to fit the printed graphic.

2. Skewed Design

If the X and Y-axes of the printed design are not perpendicular, the contour can be skewed to fit the printed design.

3. Incorrectly Scaled Design

If the print size is different from the original design in your software due to media expansion or shrinkage, or due to printing inaccuracies, the contour can be scaled to fit the printed graphic.

NOTE: *The scaling can only be adjusted by a few percent**.***

Any combination of the three above irregularities can be handled too.
The parameter SPECIAL_LOAD in the user configuration menu determines which alignment method is used.

4.3 The OPOS Alignment Method

The SummaCut, S Class and DC-series have an option called OPOS. This accurate Optical POsitioning System guarantees precise contour cutting. The registration markers are read automatically with an optical sensor.

The basic idea for printing and cutting with OPOS is described hereafter in several steps:

1. Create a design for printing and cutting.
2. Add registration markers to the design. The sign-making software should facilitate this procedure for the user. The software should automatically place the markers at the correct place and with the correct size.
3. Print the design together with the markers.
4. Insert the design in the cutter.
5. Send commands to the cutter, which inform the cutter about the amount, the size and the position of the markers.
6. Start the OPOS load procedure on the cutter; this procedure will register the markers. The software can initiate this procedure.
7. The next step will be for the user to manually position the optical sensor near the first marker.
8. Finally the software must send the outline of the design to the cutter (in the DM/PL or HP-GL vector language. The cutter will then cut the design precisely and compensate er rors from alignment, calibration differences between printer and cutter, rotation of media and skewing.

NOTE : *The information that the cutter requires (info about markers (5, starting the OPOS load procedure (6 and the cutting data (8 can all be sent at once to the cutter after the printed design is inserted in the cutter.*

In this document there are several references to the X and Y-axis. With the X-axis is meant the axis in which the media moves, the Y-axis correspond to the movement of the cutting head. This is the same co-ordinate system as for the DM/PL and HP-GL vector language used for the cutting data. The origin of the cutter is at the lower right corner of the media and corresponds to co-ordinate X=0 and Y=0.

4.4 Registration Markers

To have full advantage of the OPOS system the placement of the markers should be done fully automatically by the software. The software should have an option that places all the necessary markers for the current design. The software should then also automatically send the info about the markers to the cutter (step 5.

4.4.1 Shape

The markers must be black rectangles, all of the same size.

4.4.2 Size

It is advised to use square markers of 3 mm by 3 mm. This means that the X-size is 3 mm and the Y-size is 3 mm. Do not use markers smaller than 2 mm and do not use markers bigger than 5 mm.

Note : *Marks up to 10mm large are allowed if a standard OPOS sensor is used. The 5mm limit is for the use of OPOSCAM.*

4.4.3 Position

Make sure there is a white margin of about 3-4 times the marker size around the marker. If anything is printed within this margin the sensor may have trouble to locate the markers.
The first marker must be positioned at the origin of the drawing, at co-ordinate (0,0. More precisely the lower right corner of the marker must be at co-ordinate (0,0. All cutting data must then be related to the lower right corner of the marker.

Note : *Make sure there is at least 1 cm, preferably 2 cm margin between the markers and the edge of the media, on each side and in the front.*

Exactly 2 rows of markers along the Y-axis are necessary. The first row includes the origin marker and is placed on the right side of the design along the Y-axis. The Y-co-ordinate of the right side of these markers must be 0 (see illustration 4.1 - OPOS Markers Layout. The second row is also placed along the Y-axis at position Y-distance at the left side of the picture. The right side of these markers must be at position Y-distance (see illustration 4.1 - OPOS Markers Layout.
The Y-distance between the markers can go up to 1600 mm. This is the widest media that a cutter from Summa can accommodate.
There are 2 or more markers along the X-axis necessary. The distance between 2 markers along the X-axis (called X-distance must be the same for all markers. The advised X -distance is 400 mm. Do not go beyond 1000 mm.

Note : *When printing on sheets, make sure that there is a margin of at least 8 cm (3.15" at the end of the sheet*

The distance between 2 markers is measured from lower right corner to lower right corner in both X and Y direction as shown in.
It is advised to put markers till the end of the drawing (in the X-direction. The last marker h owever does not have to be past the design. The OPOS system will extrapolate the information as needed.

Note : *It can be useful to place a sign somewhere indicating the origin marker. This is useful when the printed design needs to be loaded in the cutter. Also make sure there is enough white space around every marker.*

Illustration 4-1 - OPOS Marks Layout

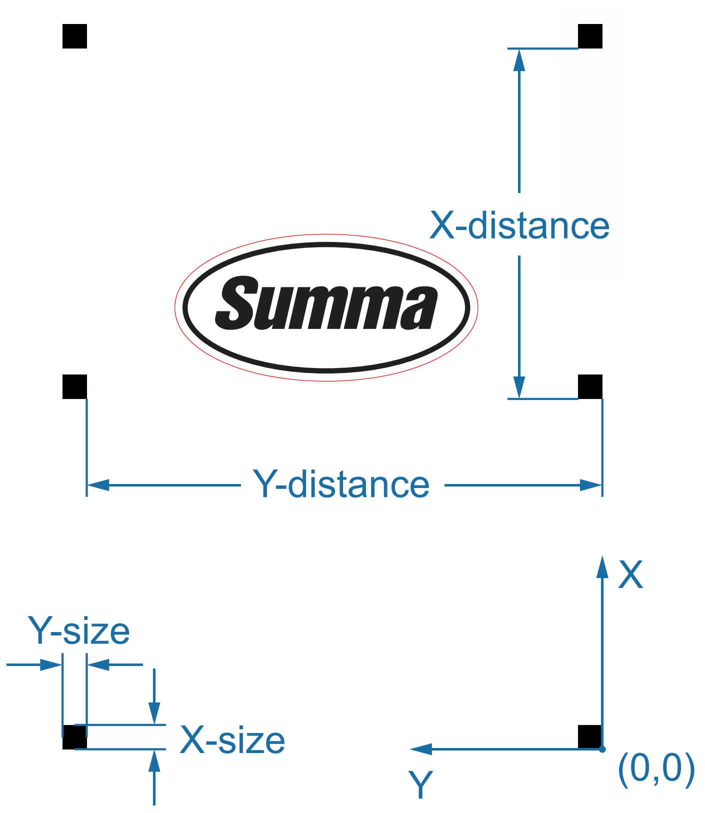
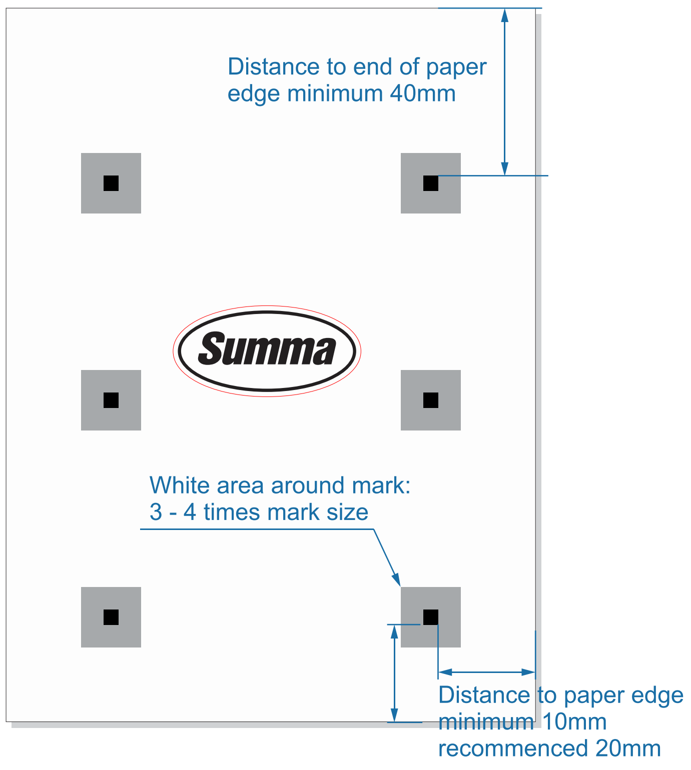


Illustration 4-2 - OPOS Markers placement


Distance to end of paper edge minimum 40mm
White area around mark: 3 - 4 times mark size
Distance to paper edge minimum 10mm recommenced 20mm

4.4.4 OPOS XY

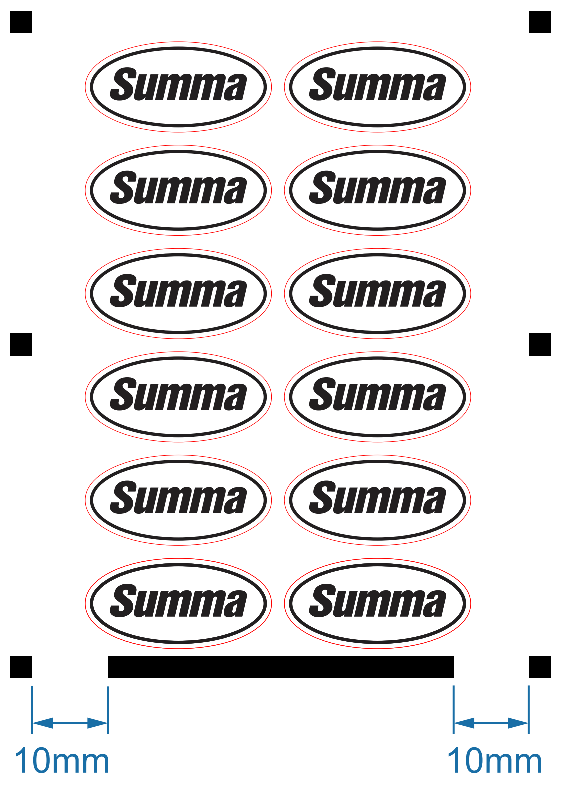
The S Class and new SummaCut cutters also support an extra correction along the Y-axis, this feature is called OPOS XY. It will compensate for misalignment between what is printed and what is cut along the Y-axis due for example for curved print-outs along this axis.

In order to do this, a line must be added along the Y-axis between or slightly beyond the first markers.

The left and right margin between the line and the markers should be 10 mm for optimal sensing. It may however vary between 10 and 15 mm.

The thickness of the line should be 3mm.

Illustration 4-3 - OPOS XY



The OPOS-XY is an extra alignment mode. So it can be set by means of the front panel or by means of the "SPECIAL_LOAD "-encapsulated command. (see 4.5.1

```
[ESC];@:
SET SPECIAL_LOAD OPOS_XY.
END.
```

When enabled, the line will automatically be sensed several times in function of the widths of the sign.

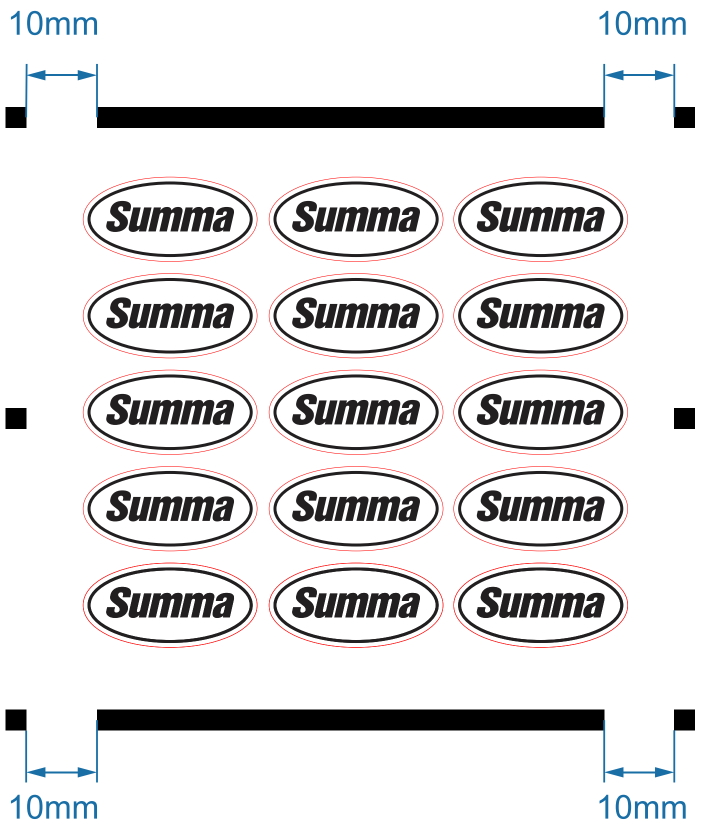
4.4.5 OPOS XY2

A first enhancement for OPOS XY is called OPOS XY2. If a line is printed between the two top marks, then this line can be scanned also, this feature is called OPOS XY2. The combination with the front and back XY-line, allows the system to contour cut wider jobs more accurately. It will compensate for misalignment between what is printed and what is cut along the Y-axis due for example for curved print-outs along this axis, and adjust along the X axis if the curvature is not the same at the bottom as at the top.

In order to do this, a line must be added along the Y-axis between the last marks.

The left and right margin between the line and the markers should be 10 mm for optimal sensing. It may however vary between 10 and 15 mm.The thickness of the line should be 3mm.

Illustration 4-4- OPOS XY2



The OPOS-XY2 is an extra alignment mode for S Class 2 only. It can only be controlled from the RIP software.

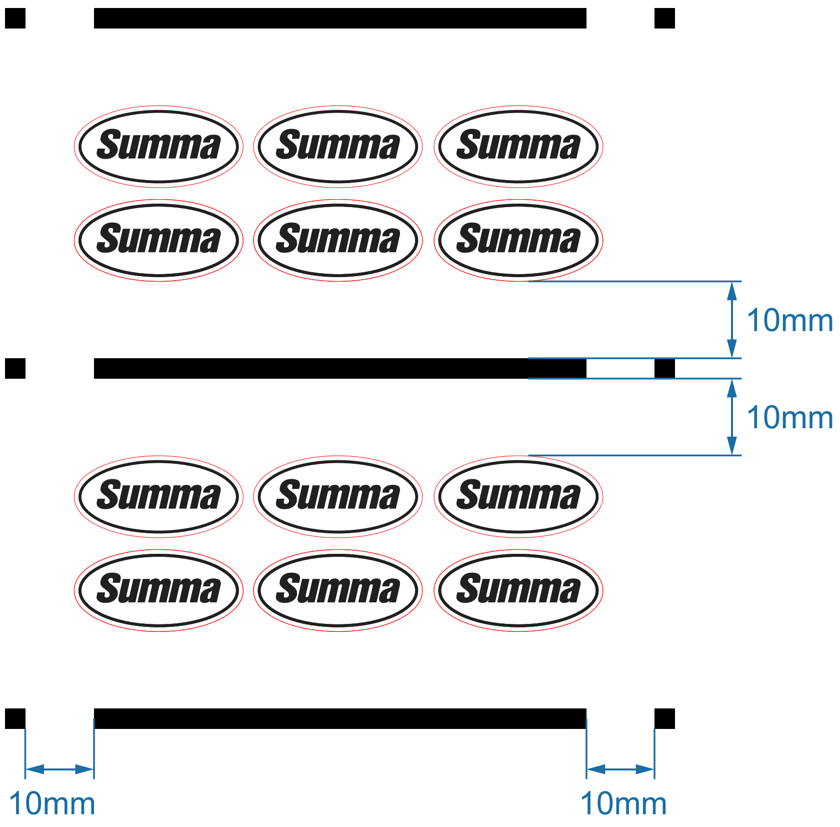
```
[ESC];@:
SET SPECIAL_LOAD OPOS_XY2.
END.
```

When enabled, the front and rear line will automatically be sensed several times in function of the widths of the sign.
OPOS XY2 is automatically converted to OPOS_XY if OPOS paneling is used.

4.4.6 OPOS Xtra

The newest option for compensating misalignment along the Y axis is OPOS Xtra. This is again an enhancement of the OPOS XY function that allows the system to contour cut wider jobs with even more accuracy.
It will compensate for misalignment between what is printed and what is cut along the Y-axis due for example for curved print-outs along this axis, and adjust along the X axis if the curvature changes in the printing direction.
For this option it is required to print a line (analogue to the OPOS XY line between the left and right OPOS mark. All these lines will be scanned be sensed several times in function of the widths of the sign in order to compensate.
The left and right margin between the line and the markers should be 10 mm for optimal sensing. It may however vary between 10 and 15 mm. The thickness of the line should be 3mm.

Illustration 4-5- OPOS Xtra



This OPOS option can only be controlled from the RIP software.

```
[ESC];@:
SET SPECIAL_LOAD OPOS_XTRA.
END.
```

When enabled, the lines will automatically be sensed several times in function of the widths of the sign.

Note : *Because of its definition this is an OPOS option that will be used when using multiple copies in the RIP.*

4.4.7 OPOS PANELLING.

The OPOS system is capable of sensing the OPOS markers per panel, not all at once, but only when needed. First all OPOS parameters are sent to the cutter as usual. When receiving the command to start OPOS loading, (LOAD MARKERS only the first 2 markers (on each side will be sensed, the other markers will be sensed when needed, this means when cutting data is received that goes behind the last sensed markers in X - Direction. This allows to cut longer opos jobs per panel.

If OPOS_PANELLING is set to ON, then the cutter will only read one mark left and right of the next panel it cuts.
If OPOS_PANELLING is set to ON4, then the last mark left and right of the previous panel will be re-read.

The command OPOS_PANELLING = [OFF,ON,ON4] enables to read the OPOS markers per panel.

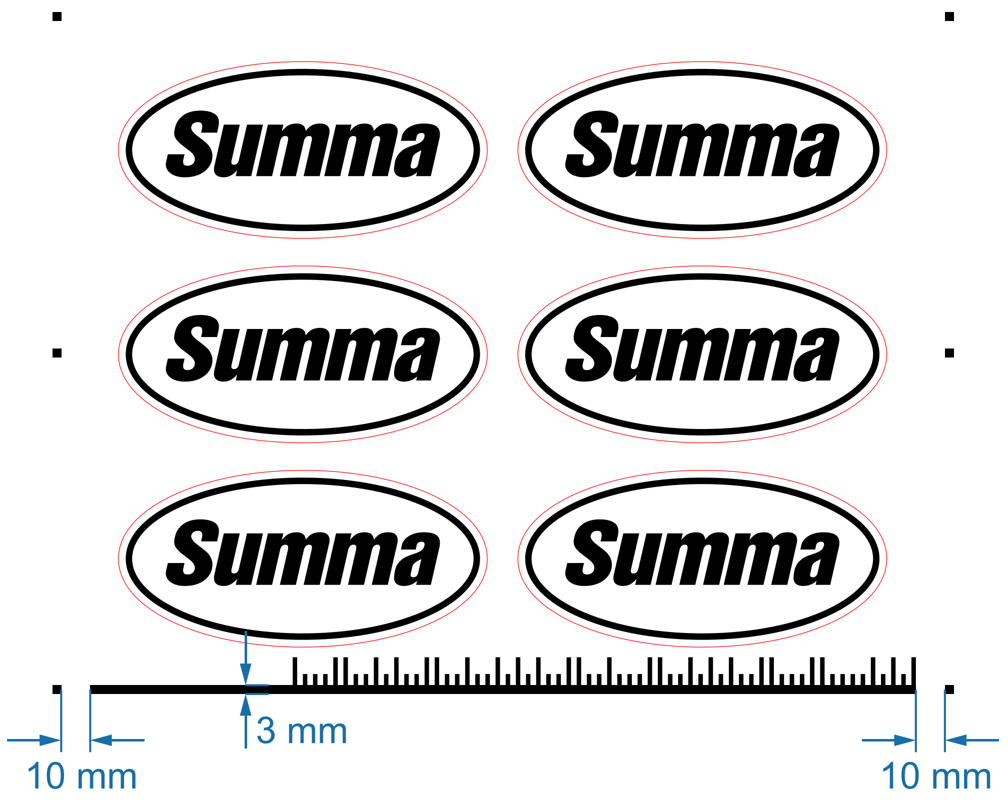
```
<esc>;@:
SET OPOS_PANELLING ON.
END.
```

4.4.8 OPOS BARCODE

The OPOS system is also capable of reading a barcode in order to identify the job. This way multiple jobs can be processed after each other without operator intervention. See section "AUTOMATED OPOS" (4.6 for more info on how to use OPOS Barcode in a workflow.

The Summa barcode consists of a line along the Y-axis with a POSTNET code on top of it. This line should be 3mm thick, aligned to the bottom of the markers and leave 10mm of white space between the markers and this line. On wider jobs this line will also be used as OPOS XY – line (see 0. On top of this line the barcode must be placed aligned to the right nearby the marker indicating the origin.

Illustration 4-6- Basic OPOS barcode



The barcode is based upon 'Postnet' used by the US post (http://en.wikipedia.org/wiki/POSTNET.

Summarized definition:

Illustration 4-7 - US Post Barcode

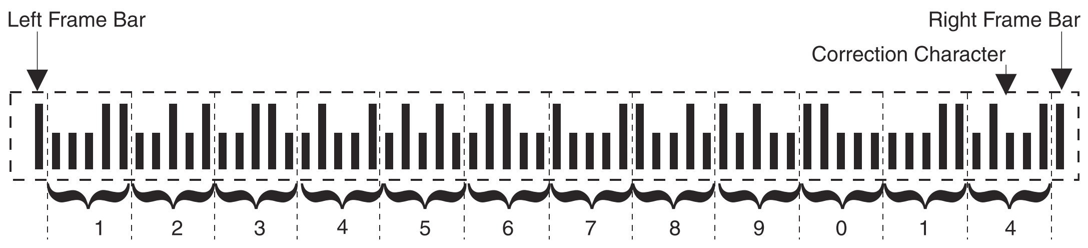


Illustration 6 - Barcode values

Numeric Value	**Binary Code Value** 7 4 2 1 0	**Barcode Value** 74210
1	00011	ııı||
2	00101	ıı|ı|
3	00110	ıı||ı
4	01001	ı|ıı|
5	01010	ı|ı|ı
6	01100	ı||ıı
7	10001	|ııı|
8	10010	|ıı|ı
9	10100	|ı|ıı
0	11000	||ııı

- The first and last full bars in a barcode—the frame bars—do not count.
- Each digit (numeric value is represented by five bars.
- Value is 11 digit number.
- The last five bars in the barcode make up the correction character. All barcodes, when added together, must equal a multiple of 10.

The OPOS barcode should be three times the official size and placed on a line.

Illustration 4-8 - Size of OPOS Barcode Digit

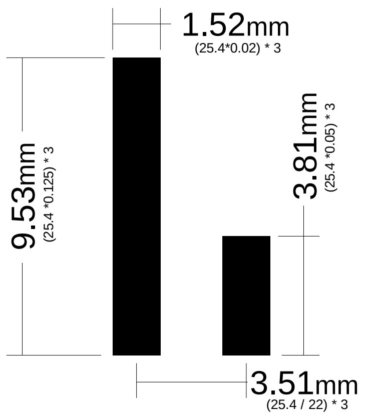
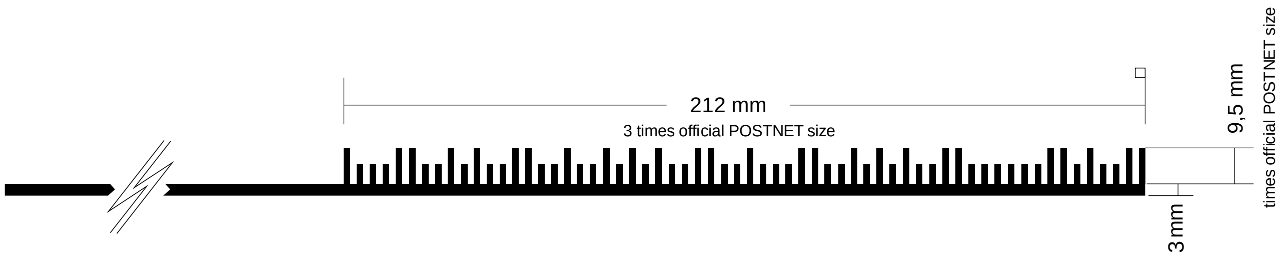


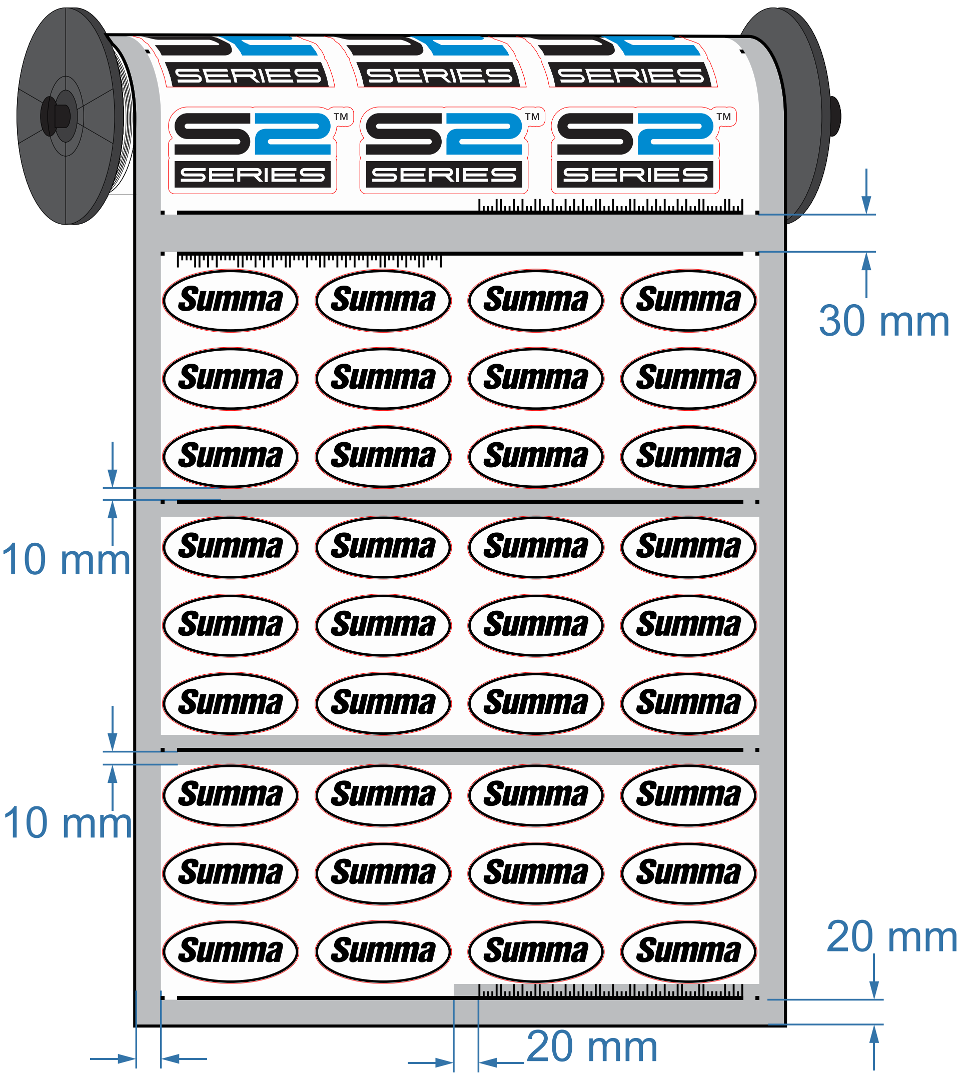
Illustration 4-9 - OPOS Barcode Size



Note : *In order not to mislead the OPOS sensor while looking for the job the area outside the markers should remain blank. All extra's (job-references, color-bars, .. should be placed inside the marker area.*

In following figure the areas that need to stay blank are made gray.

Illustration 4-10: OPOS Barcode Margins



NOTE: *When using the Cut-off option, then the distance between 2 jobs must be at least 30 mm more than the cut off value.*

The OPOS Barcode workflow can be started from the software or from the cutter control panel (preferred. To start it from the software a command is used similar to setting the alignment method.

```
[ESC];@:
SET SPECIAL_LOAD = OPOS_BARCODE.
LOAD_MARKERS.
END.
```

NOTE: *Use the "SET SPECIAL_LOAD = OPOS_BARCODE." in combination with "LOAD MARKERS." only to start the barcode workflow.*
Do not *use this command in the cutfiles, as it is similar to the alignment mode command, the command interpreter of the cutter might misinterpret it as an alignment mode.*

4.5 Data sent to the cutter

4.5.1 OPOS Commands.

Before sending the cutting data (in DM/PL or HP-GL to the cutter, the cutter must get information about the markers and the registration of the markers must be initiated. To do this the “encapsulated” language is used.

Following information must be sent to the cutter:

1. The size of the markers, both X-size and Y-size.
2. The distance between 2 markers in both directions.
3. The number of markers (in 1 row.
4. The type of registration procedure (in our case OPOS.
5. A command to start the registration of the markers.

This is done using the next encapsulated commands:

SET SPECIAL_LOAD = [OPOS| XY_ADJUST|X_ALIGN|XY_ALIGN|OPOS_XY|OPOS_XTRA]

This command specifies which alignment method to use. In our case it will be OPOS.

SET MARKER_X_DIS = [a number in the range 1200 to 52000]

This command sets the marker X distance. The unit is 0.025 mm.

SET MARKER_Y_DIS = [a number in the range 1200 to 64000]

This command sets the marker Y distance. The unit is 0.025 mm.

SET MARKER_X_SIZE = [a number in the range 48 to 400]

This command sets the marker X Size. The unit is 0.025 mm.

SET MARKER_Y_SIZE = [a number in the range 48 to 400]

This command sets the marker Y Size. The unit is 0.025 mm.

SET MARKER_X_N = [a number in the range 2 to 64]

This command specifies the number of markers **in 1 row** along the X-axis. So it does not specifies the total number of markers but only half of them.

SET SPECIAL_LOAD = OPOS_BARCODE.

This command in combination with the following command start up the barcode workflow.

LOAD_MARKERS

This command starts (initiates the procedure to register the markers on the cutter. After this command is sent to the cutter, the cutter will ask the user to set the optical sensor near the first marker. Then all the markers will be sensed automatically.

4.5.2 Cutting Data

The outline of the design can be sent in either DM/PL or HP-GL to the cutter. Important is that the origin of the cutting data is situated at the same place as the origin marker and more precisely at the Lower Right corner of this marker. To be sure that the origin is correct you can include the first marker in the outline data.

4.5.3 Sample File

Now follows an example of a cutting file that matches a printed design where markers of 2mm by 2mm where used. The distance between the markers is 400 mm in the X-direction and 1200 mm in the Y-direction. There are a total of 3 markers in one row, or 6 markers in total.

```
<esc>;@:
SET SPECIAL_LOAD=OPOS.
SET MARKER_X_DIS=16000.
SET MARKER_Y_DIS=48000.
SET MARKER_X_SIZE=80.
SET MARKER_Y_SIZE=80.
SET MARKER_X_N=3.
LOAD_MARKERS.
END.
```

<contour cutting information in DMPL, HPGL or HPGL\2>

4.6 AUTOMATING OPOS.

4.6.1 Introduction:

There are 4 cases supported to facilitate the use of several jobs that must be cut with OPOS. The main purpose to automate OPOS is to reduce the user intervention and time. Only for the first job the user will have to select the first marker. Then no more user intervention will be needed.

1. Identical jobs on a roll.
2. Different jobs on a roll.
3. Identical jobs on several sheets.
4. Using BarCode system

The scenarios are described in the following sections.

4.6.2 Identical jobs on a roll

Use the following procedure to cut several identical signs printed on a roll.

- Send the OPOS parameters that fit the sign as described in section 4.5.
- Tell how far 2 signs are from each other. For this purpose set the RECUT_OFFSET parameter. The RECUT_OFFSET parameter must be equal to the distance between the last marker on the first sign and the first marker on the next sign.
  NOTE: When using the Cut-off option, then the distance between 2 jobs must be at least 30 mm more than the cut off distance.
- Then send the "LOAD_MARKERS" command.
- Then send the contour data (DMPL. End this DMPL file with the DMPL 'e' command (end of plot command.
- NOTE do not use the DMPL Frame command

Sample1 shows how to automate small jobs (less than 14 Mbyte
Sample 2 shows how to automate bigger jobs.

SAMPLE 1:

```
<ESC>;@:
SET MARKER_X_SIZE 80.
SET MARKER_Y_SIZE 80.
SET MARKER_X_DIS 4400.
SET MARKER_Y_DIS 3920.
SET MARKER_X_N 2.
SET RECUT_OFFSET 40.
LOAD_MARKERS.
END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 e @

<ESC>;@:
RECUT 3
END.
```

SAMPLE 2:

```
<ESC>;@:
SET MARKER_X_SIZE 80.
SET MARKER_Y_SIZE 80.
SET MARKER_X_DIS 4400.
SET MARKER_Y_DIS 3920.
SET MARKER_X_N 2.
SET RECUT_OFFSET 40.
LOAD_MARKERS.
END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 e @

<ESC>;@:.RELOAD_MARKERS.END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 e @
```

4.6.3 Different jobs on a roll

Use the following procedure to cut several different signs printed on a roll.

1. Send the OPOS parameters that fit the first sign as described in section 4.5.
2. Send the "LOAD_MARKERS" command.
3. Then send the contour data of the first sign (DMPL. DO NOT End this DMPL file with the DMPL 'e' command (end of plot command
4. Now use the SET_ORIGIN command to tell where the first marker of the next sign is compared to your current position.
5. Send the OPOS parameters that fit the second sign.
6. Then send the "LOAD_MARKERS" command.
7. Then send the contour data of the second sign (DMPL. DO NOT End th is DMPL file with the DMPL 'e' command (end of plot command
8. Repeat step 4 to 7 as much as necessary.

SAMPLE 3:

```
ESC;@:
SET MARKER_X_SIZE 120.
SET MARKER_Y_SIZE 120.
SET MARKER_X_DIS 3200.
SET MARKER_Y_DIS 3200.
SET MARKER_X_N 2.
LOAD_MARKERS.
END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 0 0,0 @.

<ESC>;@: SET_ORIGIN 300,4400.
SET MARKER_X_SIZE 80.
SET MARKER_Y_SIZE 80.
SET MARKER_X_DIS 2200.
SET MARKER_Y_DIS 2200.
SET MARKER_X_N 2.
RELOAD_MARKERS.
END.
;: ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 0,0 @
```

4.6.4 Identical jobs on several sheets

Use the following procedure to cut several identical signs printed on several sheets.

1. Send the OPOS parameters that fit the sign as described in section 4.5.
2. Turn on the OPOS_SHEET_MODE.
3. Send the "LOAD_MARKERS" command.
4. Send the contour data (DMPL.
5. Now the user will have to remove the sheet of media and when he inserts a new sheet, the markers will be sensed and the same sign will be cut again. This will occur until the user presses RESET on the front panel.

SAMPLE 4

```
<ESC>;@:
SET MARKER_X_SIZE 120.
SET MARKER_Y_SIZE 120.
SET MARKER_X_DIS 3200.
SET MARKER_Y_DIS 3200.
SET MARKER_X_N 2.
SET RECUT_OFFSET 40.
SET OPOS_SHEET_MODE ON.
LOAD_MARKERS.
END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 0 @.
```

4.6.5 OPOS BARCODE

The barcode system is used when several jobs are printed after each other. Each job has a barcode that indicates the jobnumber.

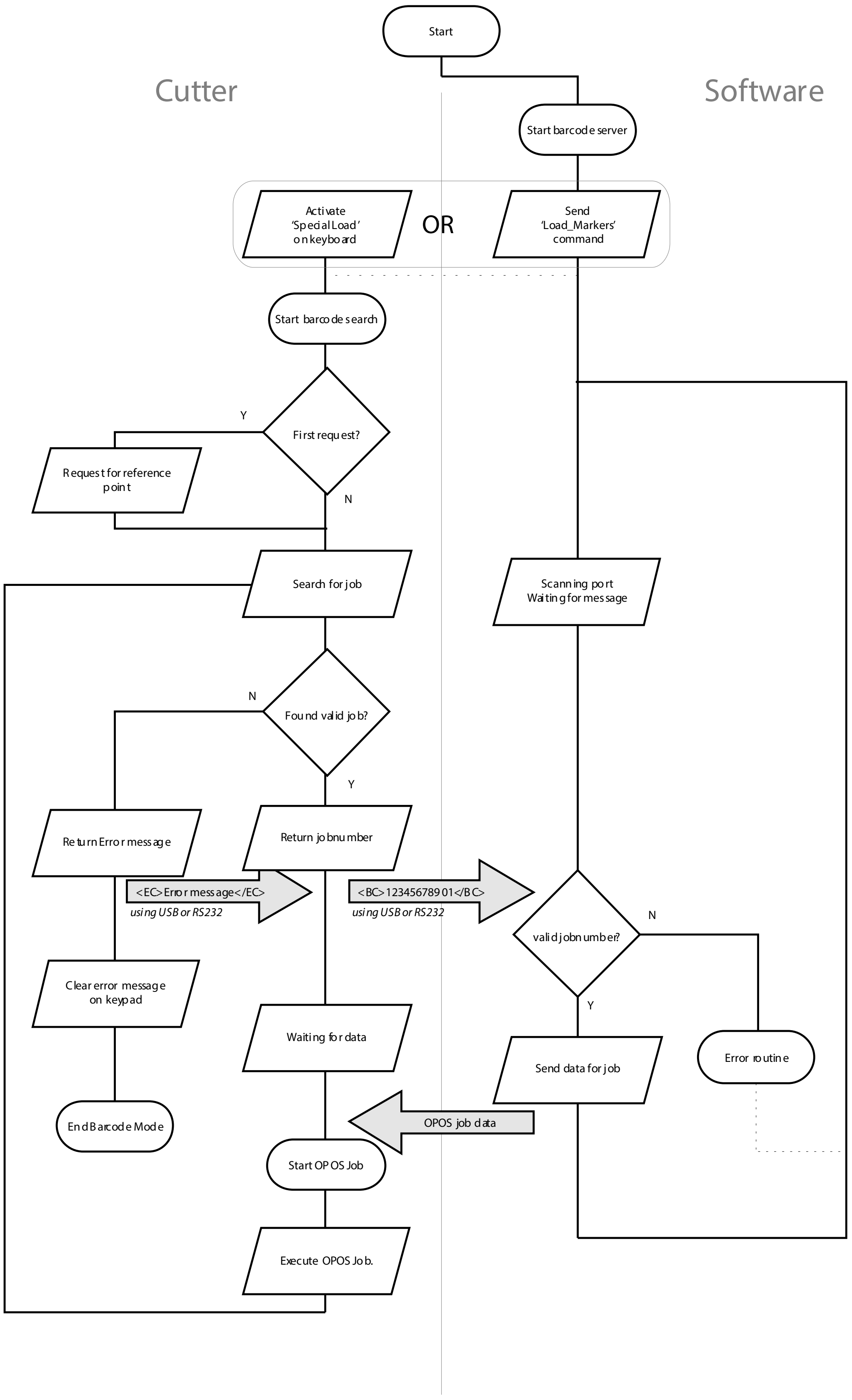
After activation, OPOS will start looking for the first job and scan the barcode. The jobnumber will be sent to the computer. On the computer a kind of **'Barcode server'** should be running, polling the port for incoming jobnumbers. On receipt of the jobnumber the software should select the correct contour job and sent the data as a standard OPOS job. OPOS will start scanning the job (with the OPOS information it received in the file and cut the contours.

When the job is finished, it will automatically start looking for the next job. When it finds a new job with valid barcode it will sent the jobnumber to the computer and wait for the data.

This continues until no new valid job is found or if the computer doesn't return any data.

<u>Barcode server in Summa Cutter Control</u>:
Summa Cutter Control is a Windows based utility to monitor all parameters of the cutter from the computer. In version 4.8 (or later a barcode server is implemented. This can be used for testing during implementation. But can also be used as a final solution in combination with other software. For the barcode server, the OPOS jobs need to be stored in ready to sent data files. File name should be the barcode number and the extension can be *.plt *.dmpl *.dmp *.hpgl *.hpg or *.prn (eg: *12345678901.plt*

Illustration 4-11 - OPOS Barcode workflow



Use the following procedure to activate the barcode system.

1. Send the "SPECIAL_LOAD = OPOS_BARCODE" command.
2. Send the "LOAD_MARKERS" command.
   Note: these first two steps can also so be executed on the keyboard of the Summa Cutter.
3. Wait until a jobnumber (11-digit number is receipt.
   Returnformat: `<BC>###########</BC>`
4. Send the OPOS parameters that fit the jobnumber as described in section 4.5.
5. Send the "LOAD_MARKERS" command.
6. Then send the contour data of the first sign (DMPL. End this DMPL file with the DMPL **'@'** command (deselect command.

- After cutting the contour and receipt of the '@' command, OPOS will start looking a next job and return the next jobnumber. Repeat step 3 to 6 as much as necessary.
- If no new job is found or the barcode is invalid a error string is returned:
  `<EC>error message</EC>.`

SAMPLE:

```
<ESC>;@:
SET SPECIAL_LOAD = OPOS_BARCODE
LOAD_MARKERS.
END.
```

Wait for job number

Returned: `<BC>12345678901</BC>`

```
<ESC>;@:
SET SPECIAL_LOAD OPOS_XY
SET MARKER_X_SIZE 80.
SET MARKER_Y_SIZE 80.
SET MARKER_X_DIS 4400.
SET MARKER_Y_DIS 3920.
SET MARKER_X_N 2.
STORE NVRAM.
LOAD_MARKERS.
END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 @
```

Wait for job number

Returned: `<BC>10987654321</BC>`

```
<ESC>;@:
SET SPECIAL_LOAD OPOS
SET MARKER_X_SIZE 80.
SET MARKER_Y_SIZE 80.
SET MARKER_X_DIS 4500.
SET MARKER_Y_DIS 3960.
SET MARKER_X_N 3.
STORE NVRAM.
LOAD_MARKERS.
END.
;:ECN A U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 @
```

Wait for job number ...

Error Messages

```
<EC>error message</EC>
```

If OPOS doesn’t find a new job (=horizontal-line it will return following error message:

```
<EC>Origin has not been set properly!</EC>
```

If OPOS reach the end of page it will return following error message:

```
<EC>Markers exceed sheet size!</EC>
```

Error messages specific for the Barcode are:

```
<EC>Not enough samples!</EC>
<EC>Barcode stop frame not found!</EC>
<EC>Barcode start frame not found!</EC>
<EC>Number of bars does not match: FULL: ... (26) - HALF: ...
(36)!</EC>
<EC>Number of FULL bars does not match for digit ...!</EC>
<EC>Checksum failed!</EC>
```

All these messages will be followed by a general OPOS Barcode error:

```
<EC>Unable to sense OPOS Barcode.</EC>
```

e.g: if the checksum is incorrect following is returned:

```
<EC>Checksum failed!</EC><EC>Unable to sense OPOS Barcode.</EC>
```

Other OPOS messages are:

```
<EC>Marker distance doesn't match with theoretical value!</EC>
<EC>OPOS not available.</EC>
<EC>Media not loaded!</EC>
<EC>Loading media failed!</EC>
<EC>Markers are too close to right wall.</EC>
<EC>Loading OPOS cancelled.</EC>
<EC>At least 2 markers must be sensed.</EC>
<EC>Not able to sense an equal number of markers left as right.</EC>
<EC>Unable to sense OPOS XY correction line.</EC>
```

5 OPOS CAM.

5.1 Introduction.

OPOS CAM is a system with a camera mounted on the cutting head of the cutter, this to do fast recognition of pre-printed markers for contour cutting.

The camera is not connected to the controller of the cutter, it is directly connected to the computer via a USB 2.0 port. USB drivers are only available for Microsoft windows operating systems (Windows XP and Windows VISTA, both 32–bit and 64 bit. There are no drivers available for Mac OS or Linux operating system.

OPOS CAM can be driving using Summa Camera Control or by writing custom software. This latest option allows more sophisticated use of markers. (Different shapes, different positioning of markers,….

5.2 Driving Opos Cam using Summa Camera control.

Summa has written a windows program called *Summa Camera control*. This program allows the use of existing software, which supports the standard OPOS features, with OPOS CAM. Normally no adaptation needs to be made to existing software!!!!

How *Summa Camera Control* works:

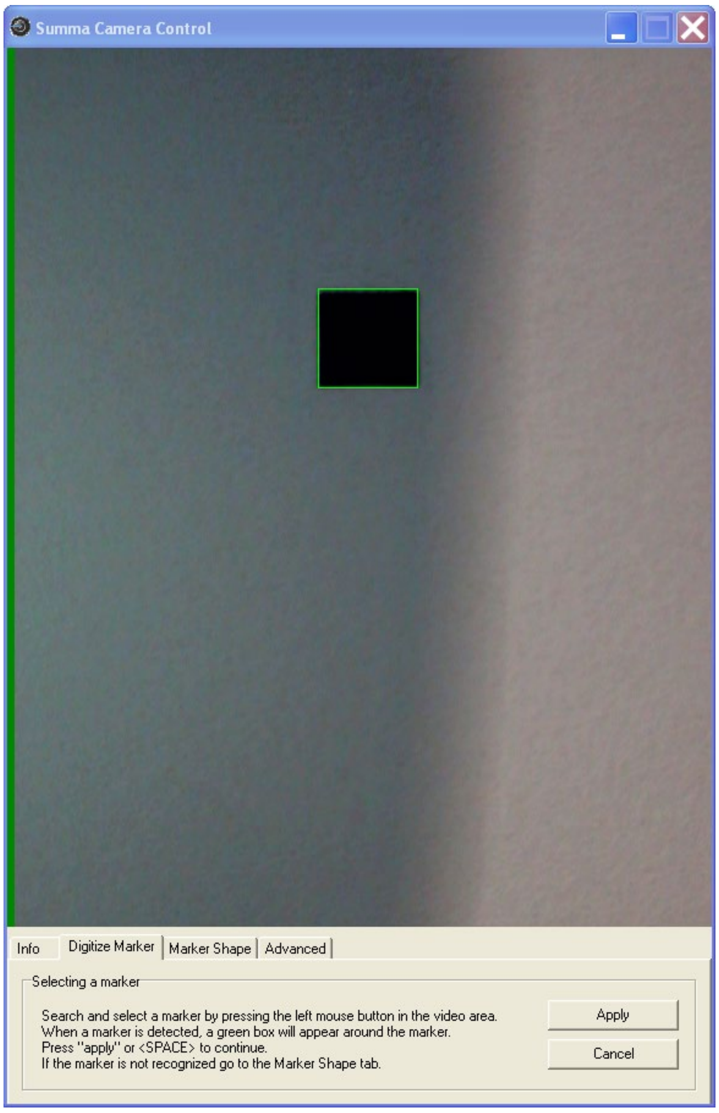
For older Summa S Class 1 cutters it is a stand alone program called Camera Control program. For the S Class 2 the program OPOS camera control is embedded in the program Summa Cutter Control.

A contour job (DMPL or HPGL, containing the OPOS information (see the previous section OPOS , is sent to the USB port or to the serial port of the cutter using sign software.

The cutter, that did receive the OPOS information, will use Summa Camera control on the PC. Summa Camera control will pop-up in the foreground. The user will be asked to point to the first marker. The first marker can be pointed on the PC screen or on the cutter keypad.

Summa camera control will now scan all the markers and send their position to the cutter.

The cutter will now cut the contour job and do all necessary transformation in order to have accurate contour cutting.



Selecting a marker

Search and select a marker by pressing the left mouse button in the video area.
When a marker is detected, a green box will appear around the marker.
Press "apply" or <SPACE> to continue.
If the marker is not recognized go to the Marker Shape tab.

Screenshot of camera Control.

5.3 Driving Opos Cam using Custom software.

It is possible to read the video stream directly into your software. This is done using the **DirectShow** feature of the Windows operating software. Commercial libraries are also available to speed up this process.

You need to do the following in order to do contour cutting using OPOS CAM:

Read the markers using DirectShow and by moving the cutting head to the next markers (using DMPL or HPGL commands.
Transform the cutting data prior to sending it to the cutter, correction for rotation, origin, scaling will be necessary.

Send the transformed data to the cutter.

All of this this is quite some work, but it allows the use of other markers, it allows more flexibility in your software, it allows better integration within your software.

A new encapsulated command has been added to facilitate the positioning of the first marker.

SET_CAM_ORIGIN.

When sending `<ESC>;@: SET_CAM_ORIGIN.END.`

The cutter will ask for the user to set the knife above the first marker. When pressing the apply button, the cutter will return "ORIGIN_LOADED" and the head will move its camera above the first marker.

Note that the full width of the cutter is now available, that the head is moved above the first marker. It is now the task of the software to detect the marker, to remember its position, to go to the other markers, to detect them and to do all necessary transformations on the cutting data.

6 Cutting Through.

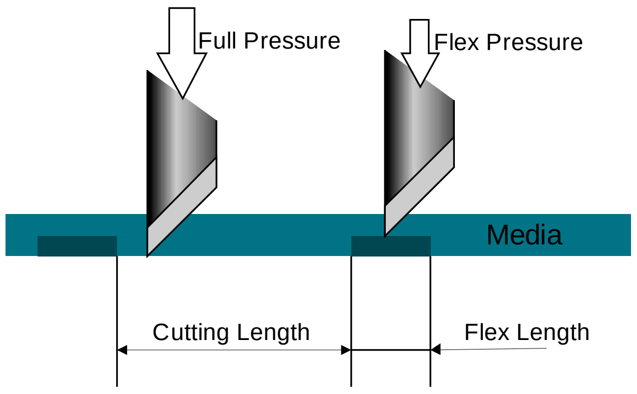
6.1 Introduction.

Cutting material completely through on a drum-plotter is not an easy thing to do. It should be avoided that cut pieces fall out while proceeding the job because this causes material crashes. In order to avoid these crashes Summa implemented 'Flex-Cut' more than 10 years ago. An interrupted cutting line makes sure that the material remains together thanks to the small media 'bridges'. When the job is finished the cut pieces can be torn out.
Although Flex-Cut helps a lot, it still has several limitations. Often, it is difficult to find the correct balance between cutting deep enough making sure the pieces can be taken out easily, and not cutting too deep making sure the material keeps it strength while cutting. Sometimes this balance doesn't exist meaning that this material can't be cut.
On the latest Summa models new functionality has been implemented. A single tool-command activates Flex-Cut, paneling and intelligent vector optimization routines. This way more materials can be cut-through reliably.

6.2 Flex-Cut

When Flex-Cut is activated on the Summa cutter. The cutting line will become an interrupted line. Flex-Cut can be switched OFF (default of set in MODE1 (fast mode, or MODE2 (accurate mode (see 3.1.21. In general MODE2 is recommended as it is most reliable. The different lengths and pressures of the cut line can be set with encapsulated commands

Illustration 14 – FlexCut



Note: *The 'FULL_PRESSURE parameter is only available in most recent models. Older models use the 'KNIFE_PRESSURE' instead.*

Note: *It is not recommended to use encapsulated command to activate FlexCut mode. Use instead the tool 5 and 6 as explained in next paragraph, the use of tools activates extra useful features.*

6.3 Tool 5 & Tool 6

When activating tool 5 or tool 6, recent Summa cutters will combine the Flex-cut mode with paneling and optimization routines. Tool 5 will use Flex-Cut MODE1, tool 6 :MODE2.
The panel size can be set with encapsulated commands (see 3.1.12.

For some other applications it may be of interest to use the paneling and optimization routines as well. Therefore, these can be (de-activated by encapsulated commands (see 3.1.11 & 3.1.27 . However, when using tool 5 or 6 this is done automatically.

It is important that all objects that will be cut with this tool are **grouped together at the end** of the cutting data.

Only lines to cut through:

```
;:ECN A P6 U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 e
```

First using default settings and then cutting through:

```
;:ECN A U 102 102 D 1835 102 1835 1717 102 1717 102 102 P6 U 2 2 D
1935 2 1935 1817 2 1817 2 2  U 1935 1000 e
```

In combination with OPOS:

```
<ESC>;@:
SET SPECIAL_LOAD OPOS
SET MARKER_X_SIZE 80.
SET MARKER_Y_SIZE 80.
SET MARKER_X_DIS 4400.
SET MARKER_Y_DIS 3920.
SET MARKER_X_N 2.
LOAD_MARKERS.
END.
;:ECN A U 102 102 D 1835 102 1835 1717 102 1717 102 102 P6 U 2 2 D
1935 2 1935 1817 2 1817 2 2  U 1935 1000 e
```

6.4 Guidelines

The settings for cutting through depend mainly on the material but also on the type of cutting-head (drag / tangential and the wear of the knife. In order to cut through a standard vinyl following settings should be a good start (feel free to use other settings.

Cutting Length : 10mm
Flex Length : 1 mm
Full pressure: 150-200gr

In order to avoid high knife-wear and damage to the cutting strip, full pressure may not be too high. Machine operator should also make sure the knife depth is set accordingly.

Flex pressure: 80-130gr
Vary this parameter in order to get an acceptable flex-cut line.
Panel size: 5-10cm (on more difficult materials (eg: papers it may be recommend to reduce panel - size to 2cm

```
<ESC>;@:
SET PANELLING_SIZE 5.
END.
;:ECN A P6 U 2 2 D 1935 2 1935 1817 2 1817 2 2 U 1935 1000 e
```

***Note:** Be careful when sending parameters to the cutter with every job. In some cases the operator may have calibrated the cutter correctly not knowing that the software will overrule his settings! Summa recommends that the operator can choose in the software if parameters are sent or not.*

The cutting through functionality is focused to cut simple shapes (e.g. rectangles. However it will process any data it receives in this mode. But it is obvious that the more complex the shape is, the more difficult it will be to get an acceptable result.
When cutting through, it is recommended that parallel lines are at least 1 cm away from each other. Otherwise, while cutting the second line, the first line may come loose and cause trouble.
In case the cutter is not capable of cutting the material in one pass. It is possible to cut each panel several times (See 3.1.13

***Note:** Do not send encapsulated commands once actual cutdata has been send (more in particular after the ";:" command. Doing so interferes with features like recut, panelling and FlexCut.*

7 TCP/IP

7.1 Introduction

As from 2009 the S class model of Summa has a Wireless option, which is compatible with the Wi-Fi protocol. This communication is bi-directional, so this means you can read the media size and send cutting data to the cutter. In order to talk to the wireless port you need to implement a TCP/IP connection, as if the data was sent to an Ethernet port (also called socket programming. Besides the IP address, you also need to write or read to/from a certain port. For the wireless option of Summa this port is set fixed to **port 9100.** The IP address can be set to any IP address.

The Wireless option was replaced in the S Class 2 series, the second generation S Class cutter, by a wired Ethernet connection. The IP address can be set to any IP address and the port is set fixed to **port 9100.**

The usage of the Wireless connection is the same as with the wired Ethernet connection.

Recommendations when using TCP/IP communication:

When polling for the media size, build in a time-out of 10 seconds, it may take a couple of seconds before the cutter reacts with the media size.

On some operating systems care must be taken when closing a socket. Closing the socket too soon may result in data not being sent to the cutter. e.g. windows will not send the last chunk of data for large files. Windows calls this ‘closing a socket gracefully’. Example code below shows how to implement a graceful close in socket communication for windows applications.

7.2 Description by MSDN on graceful close:

To assure that all data is sent and received on a connected socket before it is closed, an application should use shutdown to close connection before calling closesocket. One method to wait for notification that the remote end has sent all its data and initiated a graceful disconnect uses the WSAEventSelect function as follows :

1. *Call WSAEventSelect to register for FD_CLOSE notification.*
2. *Call shutdown with how=SD_SEND.*
3. *When FD_CLOSE received, call the recv or WSARecv until the function completes with success and indicates that zero bytes were received. If SOCKET_ERROR is returned, then the graceful disconnect is not possible.*
4. *Call closesocket.*

If this cannot be implemented then a trick can be used to prevent losing the last chunk of data, by adding a delay of 3 - 4 seconds before closing the socket. No guaranty can be given that all data will have been sent to the cutter.
Also do not close and open the socket all the time, leave it opened until all data is sent.

In order to use the wireless buffered option, an end of file command must be sent.
For DMP these are: e, @ , Z, F
For HPGL these are: PG; AF ; AH ; FR ;

Recommended implementation

Through this example code, the following general handle is used for the client socket.

```
SOCKET ConnectSocket = INVALID_SOCKET;
```

You have to include following header to support sockets.

```
#include <Winsock2.h>
```

Your project should be linked to the ws2_32.lib to support the code shown below.

```
#pragma comment(lib, "ws2_32.lib"
```

Opening a socket

```
int iResult;
WSADATA wsaData;

struct sockaddr_in clientService;


// Initialize Winsock
iResult = WSAStartup(MAKEWORD(2,2, &wsaData;
if (iResult != NO_ERROR {
//          wprintf(L"WSAStartup failed with error: %d\n", iResult;
ConnectSocket = INVALID_SOCKET;
return;
}

// Create a SOCKET for connecting to server
ConnectSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP;
if (ConnectSocket == INVALID_SOCKET {
//          wprintf(L"socket failed with error: %ld\n", WSAGetLastError(;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
}

// The sockaddr_in structure specifies the address family, IP address, and port of clientService.sin_family
= AF_INET;
clientService.sin_addr.s_addr = inet_addr( AnsiString(this->edHost->Text.c_str( ;
clientService.sin_port = htons( this->edPort->Text.ToInt( ;

// Connect to server.
iResult = connect( ConnectSocket, (SOCKADDR* &clientServi ce, sizeof(clientService ;
if (iResult == SOCKET_ERROR {
//          wprintf(L"connect failed with error: %d\n", WSAGetLastError( ;
closesocket(ConnectSocket;
WSACleanup(;
ConnectSocket = INVALID_SOCKET;
return;
}
```

Put socket in Blocking-mode

TCP/IP sockets are default set to blocking mode. To be sure you can use the code below to setup sockets in blocking-mode.

```
// Set the socket I/O mode: In this case FIONBIO enables or disables the blocking mode for the socket
// If iMode = 0, blocking is enabled;
// If iMode != 0, non-blocking mode is enabled.
iMode = 0;
iResult = ioctlsocket(ConnectSocket, FIONBIO, &iMode;
if (iResult == SOCKET_ERROR {
//          printf("ioctlsocket failed with error: %ld\n", iResult;
}
```

Set send timeout

When using sockets in blocking mode, you should set a timeout for sending data, to avoid the main program from freezing.

```
// Set send timeout.
iOptVal = 10000;
iResult = setsockopt( ConnectSocket, SOL_SOCKET, SO_SNDTIMEO, (char * &iOptVal, iOptLen ;
if (iResult == SOCKET_ERROR {
//          wprintf(L"setsockopt for SO_SNDTIMEO failed with error: %u\n", WSAGetLastError(;
            closesocket(ConnectSocket;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
}
```

For sending encapsulated data, this is no problem.
But for the cutting data you should keep sending data, for example when the Cutter is offline, and not processing any data anymore. You can achieve this in 2 ways:

- Send the file in a different thread than the main thread and setting iOptVal = 0.
- Send the file in the main thread, setting iOptVal = 10000, and retry each time. Decrease the delay to make you main program more responsive.

Set receive timeout

When using sockets in blocking mode, you should set a timeout for receiving data, to avoid the main program from freezing.

```
// Set receive timeout.
iOptVal = 500;
iResult = setsockopt( ConnectSocket, SOL_SOCKET, SO_RCVTIMEO, (char * &iOptVal, iOptLen ;
if (iResult == SOCKET_ERROR {
//          wprintf(L"setsockopt for SO_SNDTIMEO failed with error: %u\n", WSAGetLastError(;
            closesocket(ConnectSocket;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
}
```

Closing the socket

```
int iResult;
WSAEVENT NewEvent;

long events;
WSANETWORKEVENTS wsaevents;


// shutdown the connection since no more data will be sent
iResult = shutdown(ConnectSocket, SD_SEND;
if (iResult == SOCKET_ERROR {
//          wprintf(L"close failed with error: %d\n", WSAGetLastError(;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
}

// wait for other side to close the socket
NewEvent = WSACreateEvent(;
WSAEventSelect( ConnectSocket, NewEvent, FD_CLOSE;
do
{
            WaitForSingleObject(NewEvent, 50;
            WSAEnumNetworkEvents(ConnectSocket, NewEvent, &wsaevents;
            events = wsaevents.lNetworkEvents;
} while ( !(events & FD_CLOSE ;

// cleanup
iResult = closesocket(ConnectSocket;
if (iResult == SOCKET_ERROR {
//          wprintf(L"close failed with error: %d\n", WSAGetLastError(;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
}

WSACleanup(;
ConnectSocket = INVALID_SOCKET;
```

Sending data

When send succeeds it returns the number of bytes it has sent. To send files we split up the data in packets of 1024 bytes.

```
iResult = send( ConnectSocket, buffer,sizeof(buffer, 0 ;
if (iResult == SOCKET_ERROR {
//          wprintf(L"send failed with error: %d\n", WSAGetLastError(;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
}
```

Receiving data

```
  iResult = recv(ConnectSocket, ReadBuffer, 1500, 0;
  if (iResult == SOCKET_ERROR {
  //        wprintf(L"recv failed with error: %d\n", WSAGetLastError(;
            WSACleanup(;
            ConnectSocket = INVALID_SOCKET;
            return;
  }
```

For HPGL these are: PG; AF ;   AH ;   FR ;

7.3 Discovering Ethernet devices

The S Class 2 and some Summacut cutters support discovery on a local network using broadcast messages. This is again done by using the Winsock2 library.
Broadcasting is only supported in the UDP layer, so you will need to create an UDP socket.

Because there is a difference in Ethernet hardware between the S Class 2 and the Summacut, different messages have to be sent and received through a different Ethernet port:

- S Class 2:
  - Message to send: PingForSummaS2
  - Received message:
    MODEL_NAME;SERIAL NUMBER;IP ADRES
    e.g. S2 CLASS T140;751601-10001;246.124.250.100
  - Port 9000
- Summacut:
  - Message to send: 0xFF, 0x01, 0x01, 0x02 (hex
  - Received message (total of 36 bytes:
    Byte 0 → should be 0xFF
    Byte 1 → should be 0x24 (= packet length: 36 bytes
    Byte 5-8 → IP ADRES
    Byte 19-34 → MODEL_NAME
  - Port 1901

However, the basic principle of sending and receiving the messages remains the same. This is illustrated below. It is advised to search for both types of cutters.

CODE SAMPLE:

Using Winsock2

```
#include <Winsock2.h>
#include <string> // for using std::string
#pragma comment(lib, "ws2_32.lib"
```

Setting the device type

```
enum E_SUMMA_DEVICE
{
  S2_CLASS,
  SUMMACUT
};
E_SUMMA_DEVICE eDevice = S2_CLASS; // change to SUMMACUT if you want to discover Summacut
```

Setting the message to send

```
std::string sMsg;
if( eDevice == S2_CLASS
  sMsg = “PingForSummaS2”;
else
{
  char msg[] = { 0xFF, 0x01, 0x01, 0x02, '\0' };
  sMsg = msg;
}
```

Setting the ethernet port

```
USHORT  ushPort = (eDevice==S2_CLASS? 9000 : 1901;
```

Creating and opening the socket for UDP broadcasting

```
int iResult;
WSADATA wsaData;
SOCKET ConnectSocket = INVALID_SOCKET;

struct sockaddr_in clientService;

// Initialize Winsock
iResult = WSAStartup(MAKEWORD(2,2, &wsaData;
if (iResult != NO_ERROR
{
  ConnectSocket = INVALID_SOCKET;
  return;
}

// Create a SOCKET for connecting to broadcast
ConnectSocket = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP;
if (ConnectSocket == INVALID_SOCKET
{
  WSACleanup(;
  ConnectSocket = INVALID_SOCKET;
    return;
}

// The sockaddr_in structure specifies the address family,
// IP address, and port
clientService.sin_family = AF_INET;
clientService.sin_addr.s_addr = htonl(INADDR_ANY;
clientService.sin_port = htons(ushPort;

// Bind the socket to any address and the port
iResult = bind(ConnectSocket, (SOCKADDR* &sockAddr, sizeof(sockAddr;
if (iResult != 0
{
  WSACleanup(;
  ConnectSocket = INVALID_SOCKET;
```

```
    return;
}

// Enable broadcast
bool bOptVal = true;
iResult = setsockopt(ConnectSocket, SOL_SOCKET, SO_BROADCAST, (char * &bOptVal, sizeof (bOptVal ;
if (iResult == SOCKET_ERROR
{
  WSACleanup(;
  ConnectSocket = INVALID_SOCKET;
    return;
}

// Set receive timeout (here set to 600ms
iOptVal = 600;
iResult = setsockopt(ConnectSocket, SOL_SOCKET, SO_RCVTIMEO, (char * &iOptVal, sizeof(iOptVal ;
if (iResult == SOCKET_ERROR
{
  WSACleanup(;
  ConnectSocket = INVALID_SOCKET;
  return;
}
```

Sending the broadcast message

```
// Local variables
int iResult;
const int BufLen = 512;
int iLastErr;
sockaddr_in raddr;
int rlen = sizeof (raddr;

// Clear ping info
this->vInfo.clear(;

// The sockaddr_in structure specifies the address family,
// IP address, and port of the target socket
sockaddr_in        sockAddr;
sockAddr.sin_family = AF_INET;
sockAddr.sin_addr.s_addr = htonl(INADDR_BROADCAST;
sockAddr.sin_port = htons( ushPort;

  // Send ping command
iResult = sendto(ConnectSocket, sMsg.c_str(,sMsg. size(, 0, (SOCKADDR *  &sockAddr, sizeof(sockAddr;
if (iResult == SOCKET_ERROR
{
  WSACleanup(;
  ConnectSocket = INVALID_SOCKET;
  return;
}
```

Receiving data

```
// Read buffer
char ReadBuffer[512];

// Wait time
const clock_t MAX_PING_WAIT_TIME = 600;

// Read data
clock_t starttime = clock(;
clock_t time;
do
{
  iResult = recvfrom(ConnectSocket, ReadBuffer, BufLen, 0, (SOCKADDR*&raddr, &rlen;
  if( iResult == SOCKET_ERROR
  {
            iLastErr = WSAGetLastError(;
            if( iLastErr != WSAETIMEDOUT
            {
                      WSACleanup(;
                      ConnectSocket = INVALID_SOCKET;
                      return;
            }

            // Add NULL char to terminate data in read buffer for Summacut
            if( iResult > 0
                      ReadBuffer[iResult] = '\0';

            // AT THIS POINT, YOU SHOULD PARSE THE DATA IN READBUFFER AS DESCRIBED BEFORE
   }

    Sleep(10;

   time = clock(  - starttime;
}
while( time < MAX_PING_WAIT_TIME ;
```

Disconnecting the socket

```
ConnectSocket = INVALID_SOCKET
WSACleanUp(;
```

8 USB

8.1 Introduction.

This document describes the basics to communicate with the USB bus on the cutters from Summa, this for the windows operating system.
First some basics about the USB bus and the support of windows for this bus are described.
Then the specific details of the software-architecture for communication with the cutters are described.
The document ends with a sample written in C that clarifies the theory.

Note: *The explanation is done for 32 bit applications, For 64 bit applications it is completely the same providing you use the 64 bit DLL and drivers.*

8.2 USB and Windows

USB is a standard bus available on all PC's. This bus offers serial communication at high speed. It is hot pluggable, which means devices can be attached or removed at any time from the PC.

The Win32 API (or let say Windows offers some functions to communicate. These functions are `CreateFile()`, `ReadFile()`, `WriteFile()` and `CloseHandle()`.
More info about these functions can be found in your development tools (Visual C++ or Borland C++ or MSDN. The term 'file' also includes communications resources such as the USB port.
The `CreateFile()` is used to get a handle to a file or communication resource. The `WriteFile()` is then used to send data to the opened 'file'. The `CreateFile()` function can **NOT** be used directly with the USB bus. Instead a function (`open_file()` provided by Summa must be used. This function does the same as `CreateFile()`; it returns a handle to the USB bus for the cutter. Then the `WriteFile()` and `ReadFile()` functions can be used.

8.2.1 USB Pipes

Each USB device (in our case the cutter communicates to the host software (= software on PC through what is called pipes. Most USB devices do have several pipes implemented. The cutter has 3 pipes implemented. This means that there are 3 communications channels between the cutter and the host software.

The first one is called the Default Control Pipe, the operating software (Windows uses this pipe for plug and play. This pipe is used to get information about the USB configuration of the cutter. Third party developers do not need this pipe.

The two most important pipes are `PIPE00` and `PIPE01`. `PIPE01` is an **unidirectional** pipe that transports data from the host PC to the cutter. This data is the cut data (DM/PL or HP-GL. `PIPE00` is also **unidirectional** and carries information from the cutter to the host. For example you can get the size of the media through this pipe (in DM/PL for example.

Sending data has to be handled over `PIPE01` and receiving data will be handled over `PIPE00`. Each pipe will need a different handle.
As an example to get the paper size of the cutter (in DM/PL you will have to send `' ;: ER '` on `PIPE01` and you will have to read the response on `PIPE00`.

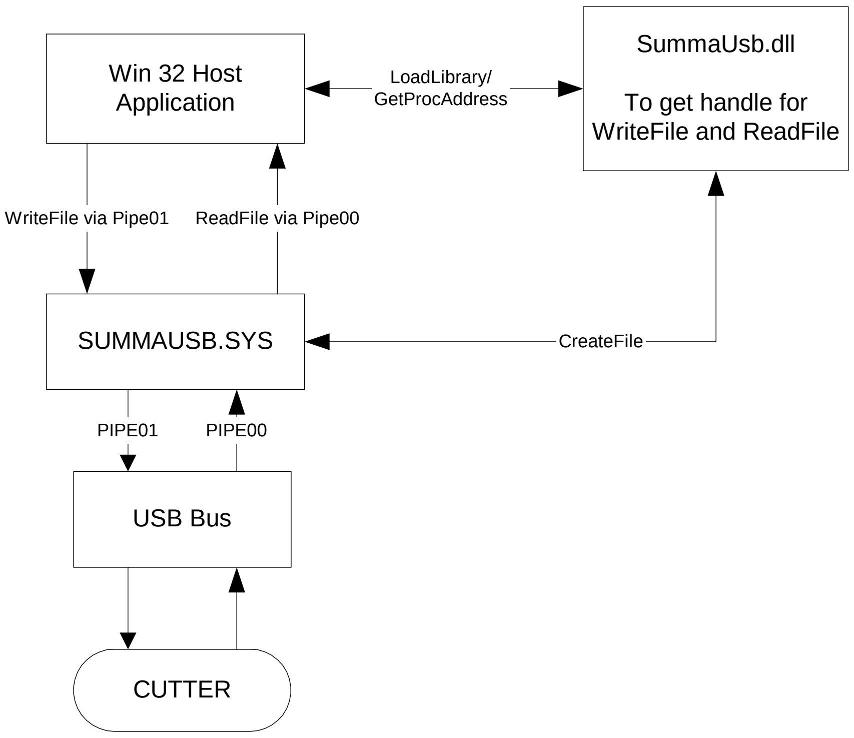
The third pipe has been called `PIPE02`. Reading from this pipe returns the free space in the internal buffer of the cutter. This pipe can be used to implement a software handshake method. This can be useful as the `WriteFile()` function only returns when the data that has to be sent is completely transmitted.

8.3 Win32 Software Components

Summa has written a windows driver (SummaUsb.sys that allows Win dows programs to communicate with the cutter over the USB bus with the standard Win32 API functions `ReadFile()`, `WriteFile()` and `CloseHandle()`.
Before using this functions, you must get a handle to the USB cutter device. For this use '`open_file()`', a function that Summa has written. This function is made available through the SummaUsb.dll. (see the sample – section 8.6 – on how to use the DLL.

You must use the `WriteFile()` and `ReadFile()` functions to communicate over the USB bus.

Illustration 1 - The different USB software parts



8.3.1 SummaUsb.sys

The Plug and play Manager from windows automatically load this WDM driver when it detects that a cutter has been plugged into the USB bus. This driver a be downloaded from the Summa website.

NOTE : *The USB bus allows hot plugging and unplugging of the devices attached to its bus. The cutter may be plugged in after your program has started. You will not get a handle when the cutter is not plugged in or is powered off. You will only get a handle to the device after the cutter is powered on and plugged in the USB bus.*

Get a handle to the USB device just before sending/receiving data. It is preferred not to get a handle at initialization of your program.

8.3.2 SummaUsb.dll

SummaUsb.dll is a dynamic link library that helps you to get an easy access to the handles of the USB pipes from the cutter. You can get 2 handles to the cutter, one for each pipe. After you have a handle to the USB cutter, you don't need SummaUsb.dll anymore. You can use the standard win32 API functions `ReadFile()` and `WriteFile()` to communicate with the cutter.

You will have to include this file in the installation of your product.
This DLL has 4 functions called `open_file()`, `open_file2()`, `open_file3()` and `open_file4()` that does the same as `CreateFile()`:
One function for each of the 4 available USB ports. The USB port number can be set through the control panel of the cutter.

HANDLE __stdcall open_file (**char*** lpFileName, **DWORD** *dwFlagsAndAttributes* ;
HANDLE __stdcall open_file2 (**char*** lpFileName, **DWORD** *dwFlagsAndAttributes* ;
HANDLE __stdcall open_file3 (**char*** lpFileName, **DWORD** *dwFlagsAndAttributes* ;
HANDLE __stdcall open_file4 (**char*** lpFileName, **DWORD** *dwFlagsAndAttributes* ;

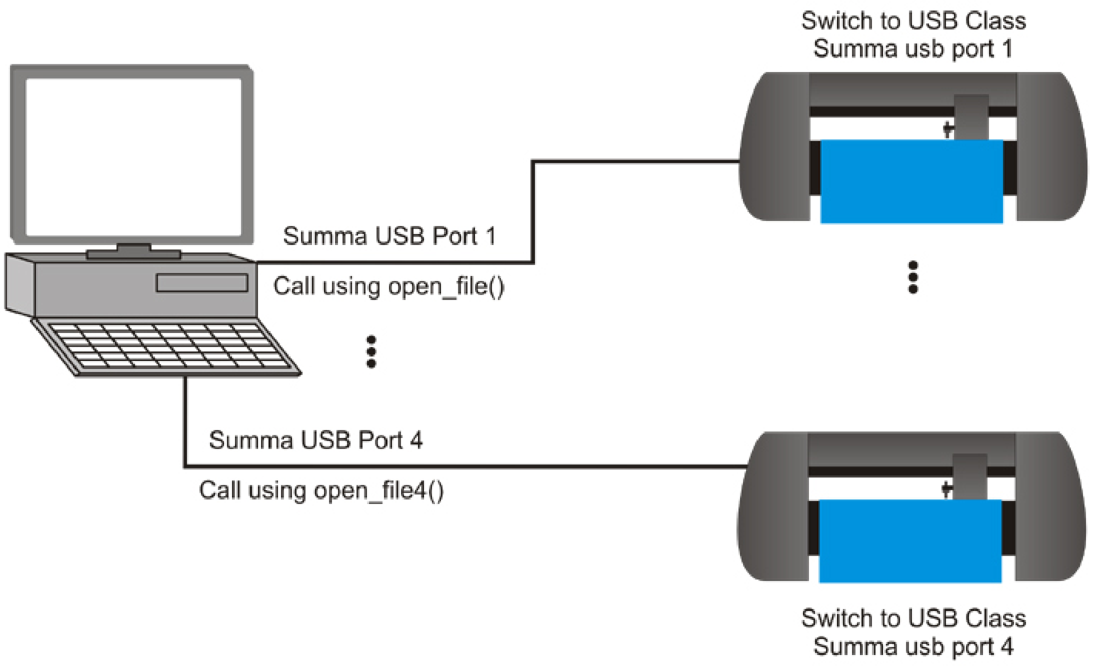
Parameters

lpFileName
Points to a string that specifies the name of the pipe. The only valid values for that parameter are the strings "`PIPE00`" and "`PIPE01`". "`PIPE00`" to read data from the cutter and "`PIPE01`" to write data to the cutter.

dwFlagsAndAttributes
Instructs the system whether or not to use asynchronous communication. See information about `CreateFile()`.

Values: `0(NULL)` for synchronous operation or `FILE_FLAG_OVERLAPPED` for asynchronous operation.

Illustration 2 - Using multiple USB cutter devices



Return Values

If the function succeeds, the return value is an open handle to the specified pipe. If the function fails, the return value is `INVALID_HANDLE_VALUE`. To get extended error information, call `GetLastError()`. The pipe can not be opened when the cutter is powered off or when the cutter is not connected to the USB bus.

***Note :** This function must be declared as* `__stdcall.`

`WriteFile()` and `ReadFile()` can then use the handle returned by `open_file()`.

8.3.3 Win32 Host Application

There are some restrictions on the Win32 API functions `ReadFile()` and `WriteFile()`.

Writing Data To Cutter.

You must send the data in little packets (e.g. 256 bytes. This gives muc h better performance. When sending a file of 1 Megabyte, for instance, the USB driver stack will take up too much time and your computer will seem to hang. When sending little chunks it is also easier to cancel a current job being sent.

When the data is split in packets of 256 bytes, the last packet will probably be smaller than 256 bytes, this is no problem.

If the input buffer of the cutter is full, the `WriteFile()` function will not return, but it will wait until some place in the buffer is available.

- **Reading Data From Cutter**

  The timeout principle doesn't exists on the USB bus, at least not in the same way as in the serial port. So when querying info from the cutter, you normally first send data on `PIPE01` and then read data on `PIPE00`. The cutter needs some time to put data on the USB bus after receiving the request. While the data is not ready the cutter will respond to any query with a zero length packet. This means the `ReadFile()` function will return with 0 bytes read. You will have to call `ReadFile()` until you get the desired response from the cutter. Calling `ReadFile()` only once will not be enough, the cutter needs in most cases to have 2 `ReadFile()` commands, the first one will respond with 0 bytes, the second one with the desired response (if the cutter is ready to answer.

  So when getting an answer from the cutter of zero bytes or less than expected, does not mean that no data is available anymore. The cutter needs some time to put the data in its output buffer. This problem is handled in the sample by adding some delays (see function `delayms()`.

  <u>Reading data must be done by multiples of 16</u>. Do not read 1 byte at a time with the `ReadFile()` function! When reading more data than available, the `ReadFile()` will return with success, but the number of bytes read will be set to what is really read.

8.4 Plug And Play Considerations

USB claims to be plug and play, this means that the user just should plug in the cutter and that all drivers are installed automatically. This is the case for the low level USB driver (SummaUsb.sys. But the user still needs to install the correct vector driver within your program(e.g. a DM/PL or HP-GL driver for Summa, the user also needs to select to which port the cutter/printer is installed.

When using USB you can also introduce plug and play for your software. This means that your software can automatically choose the correct driver when a cutter from summa is attached on the USB port without the intervention of the user. To do this you need to check if a cutter from Summa is attached on the USB port. This will be the case when you get a handle to the USB port. When getting this handle you know for sure that a cutter from Summa is attached to the USB port. You could check this for example every time your program is started. Once the cutter is found, you can then install the correct vector driver and select the USB port. The installation of the driver should only occur once of course and not every time your program is started.

8.5 Summary

- Make a 32-bit program (Win32 using `ReadFile()` and `WriteFile()` to communicate.
- Load the DLL SummaUsb.dll using `LoadLibrary()`.
- Get the address of the function `open_file()` using `GetProcAddress()`.
- Use `open_file()` from SummaUsb.dll to get a handle to the USB port instead of `CreateFile()`.
- Open 2 handles, one for reading data and one for writing data.
- Send data in little chunks (e.g. 256 bytes with `WriteFile()`.
- Read data in multiples of 16 bytes with `ReadFile()`.
- When reading data the cutter may respond with 0 bytes, read more than once, before deciding the cutter doesn't respond.
- USB has no time-out function (like the serial port

8.6 Win32 Sample Application

The sample is compiled using Visual C++ 5.0. The sample is a win 32 console application (runs in a 'DOS' box. Files are distributed together with this document. The RWSumma.c file gives an example on how to open handles to the USB pipes and how to send and read data. The sample also shows how to use SummaUSb.dll and its function "`open_file()`".
The method used in this sample is the so-called Run-Time Dynamic Linking. It has the advantage that the process can continue running even if the DLL is not available. The program can then notify the user of an error. If the user can provide the full path of the missing DLL, the process can use this information to load the DLL even though it is not in the normal search path.
This sample does not allow to cancel data or stop the execution of the `ReadFile()` or `WriteFile()` function. It has no time-out function implemented.

```
/*++
Copyright (c) 1999  Summa N.V.
Module Name:
 RWSumma.c
Abstract:
 Console test app for SummaUsb.sys driver
Environment:
 user mode only
Notes:
     Copyright (c) 1999 Summa N.V.  All Rights Reserved.

Revision History:
 11/11/99: created
--*/

#include <windows.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>
#include <sys\timeb.h>
#include <basetyps.h>

char inPipe[32] = "PIPE00";      // pipe name for bulk input pipe on
our test board
char outPipe[32] = "PIPE01";     // pipe name for bulk output pipe
on our test board
int gDebugLevel = 1;     // higher == more verbose, default is 1, 0
turns off all
int WriteLen = 0;        // #bytes to write
int ReadLen = 0;         // #bytes to read

// prototype for open_file function from summausb.dll
typedef HANDLE (__stdcall *OPENFILE)(char*, DWORD);
OPENFILE POpenFile;

// functions
/*++
Routine Description:
 waits for a time ms (in milliseconds)
Arguments:
 ms : time to wait in milliseconds
Return Value:
 Zero
--*/
void delayms(double ms)
{
 double elapsed_time = 0;
 struct _timeb start, finish;

 _ftime( &start );
 _ftime( &finish );

 do
 {
      _ftime( &finish );
      elapsed_time = ((finish.time - start.time) * 1000) +
finish.millitm
                                                                  -
start.millitm;
 } while (elapsed_time < ms);
}

int _cdecl main(
 int argc,
 char *argv[])
/*++
Routine Description:
 Entry point to RWsumma.exe
 Sends data to the USB cutter and reads response.
Arguments:
 argc, argv standard console  'c' app arguments
Return Value:
 Zero
--*/
{
 char *pinBuf = NULL, *poutBuf = NULL;
 int nBytesRead, nBytesWrite,TotalBytesRead;
 ULONG i;
 UINT success;
 HANDLE hRead = INVALID_HANDLE_VALUE, hWrite = INVALID_HANDLE_VALUE;
 ULONG totalBytes = 0L;
 char DebugString2[] = " \x1B;@:.MENU. ";
 char DebugString[] = ";: EC1 ER ";

 UINT bResult = 0;

 HINSTANCE hinstLib;
 BOOL fRunTimeLinkSuccess = FALSE;

 // First the SummaUsb Dll will be loaded
 // then we will get the adress of the function needed in the dll
	// Get a handle to the DLL module.
	hinstLib = LoadLibrary("summausb.dll");

	// If the handle is valid, try to get the function address.
	if (hinstLib != NULL)
	{
		POpenFile = (OPENFILE) GetProcAddress(hinstLib, "open_file");

		// If the function address is invalid, print a error message

		fRunTimeLinkSuccess = (POpenFile != NULL);
		// If unable to call the DLL function, DLL must be corrupted?
		if (! fRunTimeLinkSuccess)
			printf("Error : could not open function OpenFile");
	}
	else
	{
		printf("Error : could not open Summausb.dll\n");
		return (1);
	}

	//
	// open the Read and Write Pipes
	//

	hWrite = (POpenFile)( outPipe, (DWORD)NULL);

	if (hWrite == INVALID_HANDLE_VALUE)
	{
		printf("could not open %s", outPipe);
		return 0;
	}

	hRead = (POpenFile)(inPipe, (DWORD)NULL);

	if (hRead == INVALID_HANDLE_VALUE)
	{
		printf("could not open %s", inPipe);
		return 0;
	}

	//
	// put some data in the output buffer
	//

	WriteLen = sizeof(DebugString);
	poutBuf = malloc(WriteLen);
	sprintf(poutBuf, DebugString);

	// allocates some memory to put read data.
	ReadLen = 64;
	pinBuf = malloc(ReadLen + 1);

	if ( poutBuf && hWrite != INVALID_HANDLE_VALUE)
 {
     // skip any data that is still left in the buffer of the cutter
     // by reading data from the cutter, until no more data is
available

      do
      {
           success = ReadFile(hRead, pinBuf, ReadLen, &nBytesRead,
NULL);
           delayms(10);
           /* add some kind of timeout or abort procedure ? */
      } while (nBytesRead);

        //
        // send data to the cutter
      //
      WriteFile(hWrite, poutBuf, WriteLen, &nBytesWrite, NULL);
        printf("<%s> W (%04.4d) : request %06.6d bytes -- %06.6d
byte\n",
                                 outPipe, i, WriteLen,
nBytesWrite);
 }

  if (pinBuf)
 {
      nBytesRead =0;

     /* read untill we get some answer from the cutter */
     /* the cutter will return nBytesRead = 0 as long as no data is
available*/

      while (!nBytesRead)
      {
            success = ReadFile(hRead, pinBuf, ReadLen, &nBytesRead,
NULL);
            /* add some kind of timeout or abort procedure ? */
      }

      // we have some data, process it
      pinBuf[nBytesRead] = 0;
      printf(pinBuf);
      printf("\n\r");
      delayms(20);    // add some delay to allow the cutter to
prepare its next data

      TotalBytesRead = nBytesRead;

      /*if more data needed read, read more data,
       The cutter will return 0 bytes, while it is preparing its
data !!
      */

      while (nBytesRead)
      {
            success = ReadFile(hRead, pinBuf, ReadLen, &nBytesRead,
NULL);
            TotalBytesRead += nBytesRead;
            pinBuf[nBytesRead]=0;
            printf(pinBuf);
            delayms(20);
      }

      printf("\n<%s> R (%04.4d) : %06.6d bytes read\n", inPipe, i,

TotalBytesRead);
 }

 if (pinBuf)
 {
      free(pinBuf);
 }

 if (poutBuf)
 {
      free (poutBuf);
 }


 // close devices if needed
 if (hRead != INVALID_HANDLE_VALUE)
      CloseHandle(hRead);
 If (hWrite != INVALID_HANDLE_VALUE)
      CloseHandle(hWrite);


 // free the dll Library
 if (hinstLib)
      FreeLibrary(hinstLib);

 return 0;
}
```

8.7 Handshake Sample

The following application shows how to implement a handshaking method. It also uses the Run-Time Dynamic Linking method.
The program starts with creating a handle `fp` to the file specified in the command-line argument `argv[1]`. It then loads the library SummaUSB.dll and stores the handle in `hLib`. This handle is then used to retrieve the address of the function `open_file` to get a handle to an USB-pipe. This address `pOpenFile` is then used to get a handle to the USB output-pipe `hWrite` and status-pipe `hStatus`.
Now we are ready to transmit the data to the cutter.
First, we check if we're not at the end of a file, then we read a chunk of data (up to 256 bytes. If we could get the data, we have to check if there is space for it in the cutter's buffer. This is accomplished by reading form the USB status-pipe `hStatus` with the `ReadFile()` function. The data `szBuffer` returned represents the free space in the cutter's internal buffer. The converted value `nBufferFree` is compared against the amount of data `iCount` we've previously read. When there's not enough space for the data, we enter a loop until space becomes available. As soon as there's enough space for the data, we can send it to the cutter using the `WriteFile()` function with a handle to the USB output-pipe `hWrite`.
All this is repeated until all data has been sent to the cutter.
Next, all handles are closed and the Summa USB library `hLib` is unloaded.

```
// File2USB.cpp : Defines the entry point for the console
application.

#include <windows.h>
#include <stdlib.h>
#include <stdio.h>

char      inPipe[32] = "PIPE00" ;  // pipe to read responces from the cutter
char     outPipe[32] = "PIPE01" ;  // pipe to send data to the cutter
char statusPipe[32] = "PIPE02" ;  // pipe to query free space in the cutter's
internal buffer

// prototype for open_file function from summausb.dll
typedef HANDLE (__stdcall * OPENFILE)(char *, DWORD) ;
OPENFILE pOpenFile ; // pointer to the open_file function

int main(int argc, char* argv[])
{
 FILE *fp ;
 HMODULE hLib ;              // The SummaUSB library handle
 HANDLE hWrite ;             // A handle to the USB output pipe
 HANDLE hStatus ;            // A handle to the USB status pipe
 char szData[256] ;          // The data buffer
 unsigned int iCount ;       // Amount of data we've read
 char szBuffer[16] ;
 unsigned long n ;
 unsigned long nBufferFree ; // Free space in the cutter's internal buffer
 BOOL bSuccess ;


 // A filename is required
 if (argc != 2)
 {
        printf("Usage : File2USB <filename>\n");
        return 1 ;
```

```
      }

      // Open the specified file
      fp = fopen(argv[1], "rb") ;

      if (fp == NULL)
      {
            printf("Error: could not open file \"%s\"\n", argv[1]) ;
            return 2 ;
      }

      // Load the library which connects our app to the USB driver
      hLib = LoadLibrary("SummaUSB.dll") ;

      if (hLib == NULL)
      {
            fclose(fp) ;
            printf("Error: could not open SummaUSB.dll\n") ;
            return 3 ;
      }

      // Get the address of the function which gives us a handle to the specified pipe
      pOpenFile = (OPENFILE)GetProcAddress(hLib, "open_file") ;

      if (pOpenFile == NULL)
      {
            fclose(fp) ;
            printf("Error: could not retrieve address of open_file function\n",
     outPipe) ;
            return 4 ;
      }

      // Get a handle to the pipe where whe can put our data
      hWrite = (pOpenFile)(outPipe, 0) ;

      if (hWrite == INVALID_HANDLE_VALUE)
      {
            fclose(fp) ;
            printf("Error: could not open pipe \"%s\"\n", outPipe) ;
            return 5 ;
      }

      // Get a handle to the pipe where whe can read the free space in the cutter's
     internal buffer
      hStatus = (pOpenFile)(statusPipe, 0) ;

      if (hStatus == INVALID_HANDLE_VALUE)
      {
            fclose(fp) ;
            printf("Error: could not open pipe \"%s\"\n", statusPipe) ;
            return 6 ;
      }

      bSuccess = TRUE ;

      // Repeat until no more data available or when there are problems on the USB port
      while (!feof(fp) && bSuccess)
      {
            // Get some data to send
            iCount = fread(szData, sizeof(char), sizeof(szData), fp) ;

            // If there is something to send
            if (iCount > 0)
            {
                    // Use software handshaking
                    do
                    {
                            // Read data from USB status pipe
                            bSuccess = ReadFile(hStatus, szBuffer, sizeof(szBuffer), &n,
    0) ;
                            szBuffer[n] = 0 ;               // Terminate string
                            nBufferFree = atol(szBuffer) ;  // Free space in internal
    buffer of the cutter

                    } while (bSuccess && nBufferFree < iCount) ; // Repeat until space
    becomes available

                    // Send the data through the USB output pipe
                    bSuccess = WriteFile(hWrite, szData, iCount, &n, 0) ;
            }
     } ;

     CloseHandle(hStatus) ;
     CloseHandle(hWrite) ;
     fclose(fp) ;

     // Release the SummaUSB library
     FreeLibrary(hLib) ;
     return 0;
    }
```

9 Appendix

9.1 DM/PL Commands Summary

Command	Description	Page
;:	Select cutter.	2
@	Deselect cutter.	4
Z	Reset cutter.	5
ECn	Co-ordinate Addressing defines programmable resolution: EC0, EC1 : 0.001″, EC5 : 0.005″, ECN : 0.025mm, ECM : 0.1mm.	3
W	Window and Viewport co-ordinates define scaling and clipping.	6
A	Absolute addressing: co-ordinates following this command are interpreted as absolute positions.	3
R	Relative addressing: co-ordinates following this command are relative to the previous position.	3
D	Lower the tool. Co-ordinates following this command will move the cutter-head with the tool down.	4
U	Raise the tool. Co-ordinates following this command will move the cutter-head with the tool up.	4
x,y	Vector move command. Moves the cutter-head to the specified location.	4
Pn	Tool selection.	7
Vn	Velocity.	8
BPn	Tool pressure.	5
Fn	Frame command. Moves the media.	5
ER	Report command. Used to query media-sizes.	8
e	End of plot command. Deselects the cutter and moves the media past the sign.	4
EW n	Sets length of the job	10
c	Cut off command	10

Table 9.1 - DM/PL Commands

9.2 HP-GL Commands Summary

Command	Description	Page
IN;	Initialize cutter. Reset parameters.	12
BP;	Begin plot. Moves media to the end of the previous sign. HP-GL/2 only.	12
PAx,y;	Move cutter-head to specified absolute location.	12
PRx,y;	Move cutter-head relative to previous location.	12
PDx,y;	Move cutter-head with the tool down to the specified location.	12
PUx,y;	Move cutter-head with the tool up to the specified location.	12
VSn;	Velocity.	13
SPn;	Tool selection.	13
FSn;	Tool pressure.	14
OH;	Output Hardclip : Used to query media-sizes.	14
PG;	End Of Plot: Moves the media past the sign.	14
EC;	Enable Cut off command	14

Table 9.2 - HP-GL Commands

9.3 Maximum Pressure and Speed by Model

The maximum tool-pressure, speed and media width depends on the cutter-model, and is summarized in the following table:

Legacy Products:

Device	Max. Pressure	Max. Speed	Max Cutting Width
SummaCut D520	400 gr.	600 mm/s	500 mm
SummaCut D620	400 gr.	600 mm/s	600 mm
SummaCut D760	400 gr.	600 mm/s	740 mm
SummaCut D1020	400 gr.	600 mm/s	1000 mm
SummaCut D1220	400 gr.	600 mm/s	1200 mm
SummaSign D610	400 gr.	600 mm/s	585 mm
SummaSign D750	400 gr.	600 mm/s	703 mm
SummaSign D1010	400 gr.	600 mm/s	995 mm
SummaSign D1300	400 gr.	600 mm/s	1195 mm
SummaSign T610	600 gr.	600 mm/s	585 mm
SummaSign T750	600 gr.	600 mm/s	720 mm
SummaSign T1010	600 gr.	600 mm/s	995 mm
SummaSign T1300	600 gr.	600 mm/s	1195 mm
SummaSign T1010_PLUS	600 gr.	600 mm/s	995 mm
SummaSign T1010A	600 gr.	600 mm/s	995 mm
SummaSign T1300A	600 gr.	600 mm/s	1195 mm
SummaSign D610_PRO	400 gr.	1000 mm/s	585 mm
SummaSign D750_PRO	400 gr.	1000 mm/s	703 mm
SummaSign D1010_PRO	400 gr.	1000 mm/s	995 mm
SummaSign D1300_PRO	400 gr.	1000 mm/s	1195 mm
SummaSign D1400_PRO	400 gr.	1000 mm/s	1395 mm
SummaSign D1600_PRO	400 gr.	1000 mm/s	1575 mm
SummaSign D140	400 gr.	1000 mm/s	1395 mm
SummaSign T610_PRO	600 gr.	1000 mm/s	585 mm
SummaSign T750_PRO	600 gr.	1000 mm/s	720 mm
SummaSign T1010_PRO	600 gr.	1000 mm/s	995 mm
SummaSign T1300_PRO	600 gr.	1000 mm/s	1195 mm

SummaSign T1400_PRO	600 gr.	1000 mm/s	1395 mm
SummaSign T1600_PRO	600 gr.	1000 mm/s	1595 mm
SummaSign T140	600 gr.	1000 mm/s	1395 mm
SummaCut D60	400 gr.	800 mm/s	600 mm
SummaCut D60 FX	400 gr.	800 mm/s	575 mm
SummaCut D120	400 gr.	800 mm/s	1200 mm
SummaCut D140	400 gr.	800 mm/s	1350 mm
SummaCut D160	400 gr.	800 mm/s	1575 mm
S Class 75 D	400 gr.	1000 mm/s	792
S Class 120 D	400 gr.	1000 mm/s	1250
S Class 140 D	400 gr.	1000 mm/s	1400
S Class 160 D	400 gr.	1000 mm/s	1630
S Class 75 T	600 gr.	1000 mm/s	792
S Class 120 T	600 gr.	1000 mm/s	1250
S Class 140 T	600 gr.	1000 mm/s	1400
S Class 160 T	600 gr.	1000 mm/s	1630
S Class 75 TA	600 gr.	1000 mm/s	727
S Class 120 TA	600 gr.	1000 mm/s	1185
S Class 140 TA	600 gr.	1000 mm/s	1335
S Class 160 TA	600 gr.	1000 mm/s	1565
S Class 75T OPOS CAM	600 gr.	1000 mm/s	792
S Class 140T OPOS CAM	600 gr.	1000 mm/s	1400
S Class 160T OPOS CAM	600 gr.	1000 mm/s	1630

Table 7.3a - Maximum Pressure And Speed By Model

Current products:

Device	**Max. Pressure**	**Max. Speed**	**Max Cutting Width**
SummaCut-R D60	400 gr.	800 mm/s	630 mm
SummaCut-R D60 FX	400 gr.	800 mm/s	630 mm
SummaCut-R D75	400 gr.	800 mm/s	780 mm
SummaCut-R D120	400 gr.	800 mm/s	1230 mm
SummaCut-R D140	400 gr.	800 mm/s	1380 mm
SummaCut-R D140 FX	400 gr.	800 mm/s	1380 mm
SummaCut-R D160	400 gr.	800 mm/s	1605 mm
S Class 2 75 D	400 gr.	1000 mm/s	792
S Class 2 120 D	400 gr.	1000 mm/s	1250
S Class 2 140 D	400 gr.	1000 mm/s	1400
S Class 2 160 D	400 gr.	1000 mm/s	1630
S Class 2 75 T	600 gr.	1000 mm/s	792
S Class 2 120 T	600 gr.	1000 mm/s	1250
S Class 2 140 T	600 gr.	1000 mm/s	1400
S Class 2 160 T	600 gr.	1000 mm/s	1630
S Class 2 75 TA	600 gr.	1000 mm/s	727
S Class 2 120 TA	600 gr.	1000 mm/s	1185
S Class 2 140 TA	600 gr.	1000 mm/s	1335
S Class 2 160 TA	600 gr.	1000 mm/s	1565
S Class 75 T OPOS CAM	600 gr.	1000 mm/s	792
S Class 140 T OPOS CAM	600 gr.	1000 mm/s	1400
S Class 160 T OPOS CAM	600 gr.	1000 mm/s	1630

Table 8.3b - Maximum Pressure And Speed By Model