



COMMUNICATION PROTOCOL

E8-V5

SIC MARKING

13 route de Limonest
ZAC de la Braille
69380 LISSIEU – France

phone : (+33) 04.72.54.80.00
fax : (+33) 04.78.47.39.40
E-Mail : info@sic-marking.com
<http://www.sic-marking.com>

CONTENTS

1 - GENERAL INFORMATION	5
2 - TEXT PROTOCOL	6
a) String to be sent:.....	6
b) Response of the control.....	6
c) Commun fuctions	6
• LOADFILE	load a marking file..... 6
• SETVAR	set a variable or an increment to the current file..... 6
• RUN	start the marking..... 7
• RESETERROR	if an error occure, you need to clear the machine status..... 7
d) managing files functions.....	8
• NEWFILE	create a new marking file..... 8
• SETFILEOPTION	set diameter for D-AXIS system..... 8
• INSERTTEXTLINE	insert a line to the current file..... 8
• INSERTPAUSELINE	insert a line to the current file..... 9
• INSERTLOGOLINE	insert a logo to the current file..... 9
• INSERTECC200LINE	insert a Data matrix to the current file..... 9
• INSERTAUTOZLINE	make an auto-sensing in current file..... 10
• INSERTMOVEZLINE	insert a line to the current file..... 10
• INSERTROTATIONLINE	insert a line to the current file..... 10
• SETLINEOPTION	Set options for the last inserted line of the current file..... 11
• SAVEFILE	save current file..... 11
• FILEDELETE	erase a file into the control..... 11
e) manual programming functions.....	12
• HOMEPOSITION	go to home position..... 12
• IMPACT	Impact or srcatching..... 12
f) settings functions	13
• SETSHIFTINCVAR	setting setting a shift increment to the current file..... 13
• SETGLOBALVAR	Affectation variable globale..... 13
• SETGLOBALINC	Affectation incrément globale..... 14
• GETVERSION	get the program version..... 14
• GETDATETIME	getting the date and the time..... 14
• SETDATETIME	setting the date and the time..... 14
3 - BINARY PROTOCOL.....	15
a) String to be sent :.....	15
b) The control response to all commands :.....	15
c) commands list :.....	16
d) Commun functions :.....	17
• LOAD FILE	Code 'c' (0x63 in hexadecimal)..... 17
• FILE SET VAR	Code '7' (0x37 in hexadecimal)..... 17
• START MARKING	Code 'g' (0x67 in hexadecimal)..... 17
• RESET_ERREUR	Code 'E' (0x45 in hexadecimal)..... 17
e) managing files functions.....	18
• NEW_FILE	Code 'f' (0x66 in hexadecimal)..... 18
• INSERT_LINE_TO_FILE	Code 'l' (0x6C in hexadecimal)..... 18
• FILE_SET_OPTION	Code 'o' (0x6F in hexadecimal) Marking file Option..... 19
• LINE_SET_OPTION	Code 'a' (0x61 in hexadecimal) last line of marking file Option..... 19
• SAVE_FILE	Code 'e' (0x65 in hexadecimal)..... 20
• FILE LIST	Code 'L' (0x4C in hexadecimal)..... 20
• PC_TO_E6	Code 'G' (0x47 in hexadecimal)..... 21
• E6_TO_PC	Code 'S' (0x53 in hexadecimal)..... 22
• DEL_FILE	Code 'D' (0x44 in hexadecimal)..... 22
f) File options definition:.....	23
• Diameter Option	Setting the part diameter (for the D-Axis use only)..... 23
• Comment Option	Setting the file comment (free text)..... 23
g) Last marking file line option description :.....	23
• Attribut option	Pritable setting..... 23
• Indexor Option	D-axis setting..... 23

h) manual programming	24
• ORIGINE Code 'H' (0x48 in hexadecimal)	24
• IMPACT Code 'P' (0x50 in hexadecimal)	24
• SET_OUPUT Code 'Z' (0x5A in hexadecimal)	24
• GET_INPUTS Code 'Y' (0x59 in hexadecimal)	24
i) Settings functions.....	25
• RESTART Code '*' (0x2A in hexadecimal)	25
• SYNCHRO_DATEHEURE Code 'h' (0x68 in hexadecimal)	25
• VITESSE_COM Code 'v' (0x76 in hexadecimal)	25
• GET_CONFIG_AXE Code 'X' (0x58 in hexadecimal)	26
• SET_CONFIG_IMPACT Code 'i' (0x69 in hexadecimal)	27
• SET_CONFIG_IMPACT Code 'I' (0x49 in hexadecimal)	27
• SET_VAR_GLOBALE Code '8' (0x38 in hexadecimal)	28
• SET_INC_GLOBALE Code '9' (0x39 in hexadecimal)	28
• SET_SHIFT_INC_FICHER Code '0' (0x30 in hexadecimal)	28
• GET_OPTION Code '1' (0x31 in hexadecimal) Control Option.....	28
• SET_OPTION Code '2' (0x32 in hexadecimal) Control Option.....	28
• DEL_OPTION Code '3' (0x33 in hexadecimal) Control Option.....	28
• GET_OPTION_DALLAS Code '4' (0x34 in hexadecimal) Machine Option.....	28
j) Control option definition :	29
• LANGUAGE Option Language.....	29
• Option FICHER File to open at boot time.....	29
• Option VALUE PAR DEFAULT Default values editing fichiers.....	29
• KEYBOARD Option Keyboard setting.....	30
• STATISTICS Option Controls Statistics.....	30
• MACHINE Option machine name (C150, I80, ...).....	30
• SHIFT Option Shift setting.....	30
• MAINTENANCE Option Stylus warning maintenance message setting.....	31
• GLOBALES VAR Option setting the global variables.....	31
• GLOBALES INC Option setting the global increments.....	31
• ALPHA INC BASE Option Setting the alphanumeric increment rules.....	31
• UNIT Option metric or inch system.....	31
k) Machine option definition:.....	32
• auto-sensing Option setting the Auto-sensing.....	32
• Comment Option free text.....	32
• Scratching Option Setting the scribbling machine.....	32
• Portable Option for Portable machine.....	32
• Fast move option Setting the fast move / marking move speed.....	32
• Motor intensity Option setting the motor intensity.....	33
• Input-Output Option Setting the inputs and outputs functions.....	33
4 - COMPATIBILITY WITH THE COMMUNICATION PROTOCOL OF THE 4A VERSION PROGRAM.....	34
a) Communication protocol:.....	34
b) String to be sent :.....	34
c) List of functions :.....	34
d) Examples de communication :.....	35

1 - General information

Guide conventions:

The following typographical conventions are used throughout this manual.

Text in [] is a data, it is written with one byte

Text in [...(n byte)] is a data, it is written with n bytes

Text in <> is a group of data

A char in ' ' is written as itself, for example, '1' is the 49 ASCII code, 'A' is the 65 ASCII code.

Examples:

[0x41] is the 65 ASCII code (in decimal base), which is the letter **A**

[0x42] is the ASCII code 42 in hexadecimal base, which is the letter **B**

[ETX] is ASCII code ETX (0x02 in hexadecimal base)

2 communications protocols are possible :

- TEXT protocol : all data are printable char
 - easy to read
 - it is not possible to transfer every kind of data
 - the string to be transferred is longer
 - ideal for automatism who have trouble sending binary code.

- BINARY Protocol : data are compact and written using the full ASCII table
 - > no limits for the data
 - > the string to be transferred is compact (fast communication)
 - > data are coded, it is not easily readable

- the string to be transferred cannot be longer than 25 000 bytes

Peticular case : the string which run the marking file:

As soon as the control receive the string, it send the [ACK] code then start the marking file

At the last dot marked, the control send [EOT]

At the end of the cycle, and at home position, the control send [ENQ]

On error, the control send [NAK] [ERROR-CODE (3)]

When an error occurs, you must acknowledge the message by sending the RESET ERREUR string

Remark: For historical compatibility, you can use the communication protocol of 4A version (see the end of document)

Le texte écrit avec la police
COURIER ne doit pas être traduit

Text witten with the COURIER font
must not be translated

2 - Text Protocol

a) String to be sent:

<command> [Space char] <Data> [Space char] ... [Space char] <Data> [CR][LF]
or <command> [CR][LF]

where

<Command>	is the command code
<Data>	is the data of the command (without Space char)
[CR]	is the 0x0D char (obsolet)
[LF]	is the 0x0A char

b) Response of the control

<command> [Space char] <Answer> [CR][LF]

the control response to evry command.

c) Commun fuctions

These functions let you select a marking file in the controller, affect variables and run the marking cycle.

• **LOADFILE** *load a marking file*

Syntaxe :

```
LOADFILE <File name> [CR] [LF]
```

Answer :

```
LOADFILE OK [CR] [LF]           → OK  
LOADFILE ERROR [CR] [LF]      → file not found
```

Data :

File name 11 char max, in upper case

Example :

String to send :

```
LOADFILE MYFILE [LF]
```

String received:

```
LOADFILE OK [CR] [LF]           → the file MYFILE is loaded  
LOADFILE ERROR [CR] [LF]      → the file MYFILE is not found
```

• **SETVAR** *set a variable or an increment to the current file*

Syntaxe :

```
SETVAR <Var name> <Value> [CR] [LF]
```

Answer :

```
SETVAR OK [CR] [LF]             → OK  
SETVAR VAR NOT FOUND [CR] [LF] → variable introuvable
```

Data :

Var name in upper case
Value free printable text

Example :

String to send : setting the value 53H805 to the var named OF

```
SETVAR OF 53H805 [LF]
```

String received:

```
SETVAR OK [CR] [LF]
```

•RUN*start the marking*

Syntax :

RUN <Simulation>

Answer :

RUN OK[CR] [LF]	→ start of marking
[EOT]	→ last dot marked
[ENQ]	→ back to home position
[NAK] [Err1] [Err2] [Err3]	→ if an error occurse

Data :

Simulation (optional)
 Marking at force 0 if "Simulation", else marking at normal force

Erri if an error occurse, see fig 1 on annexe

Note :

If there is PAUSE in the marking file :

At a PAUSE line, the control send the char P [0x50] and wait for :

- response p [0x70] from RS232,
- or the Start button to be pressed,

to continue the marking,

Example :

Start marking

String to send :
 RUN [LF]

String received:

RUN OK[CR] [LF]	→ start of marking
[EOT]	→ last dot marked
[NAK] [0x00] [0x88] [0x00]	→ Error : 88 = 80 + 8 = Z axis + Error Sensor : Z axis is not at home position

Start a simulation

String to send :
 RUN SIMULATION [LF]

String received:

RUN OK[CR] [LF]	→ start of marking
[EOT]	→ last dot marked
[ENQ]	→ End of simulation OK

• RESETERROR*if an error occurse, you need to clear the machine status.*

Syntax :

RESETERROR [CR] [LF]

Answer :

RESETERROR OK [CR] [LF]

d) managing files functions

With these functions, you will be able to transfer all marking parameters :
First, create a free file using NEWFILE command, then insert the line with the INSERT*LINE function, then you can save the file for a later use with the SAVEFILE command or run the cycle with the RUN command.

•NEWFILE *create a new marking file*

Syntaxe :

```
NEWFILE <Marking speed> <Fast speed> <Crossed zero> <File name> [CR] [LF]
```

Answer :

```
NEWFILE OK[CR] [LF]           → OK  
NEWFILE BAD ARGUMENTS [CR] [LF] → errors in parameters
```

Data :

Marking speed	from 1 to 9
Fast speed	from 1 to 9
Crossed zero	0 for zero not crossed, 1 for crossed zero
File name	(optional) 11 char max, in upper case

Example :

```
String to send :  
NEWFILE 5 7 0 MYFILE [LF]  
String received:  
NEWFILE OK [CR] [LF]
```

•SETFILEOPTION *set diameter for D-AXIS system*

Syntaxe :

```
SETFILEOPTION <Diameter> [CR] [LF]
```

Answer :

```
SETFILEOPTION OK[CR] [LF]           → OK  
SETFILEOPTION BAD ARGUMENTS [CR] [LF] → errors in parameters
```

Data :

Diameter	the part diameter in 10 th of mm
-----------------	---

Example : *to set a 21.5 mm diameter*

```
String to send :  
SETFILEOPTION 215 [LF]  
String received:  
SETFILEOPTION OK [CR] [LF]
```

• INSERTTEXTLINE *insert a line to the current file*

Syntaxe :

```
INSERTTEXTLINE <X> <Y> <Z> <W> <H> <Angle> <Radius> <Space> <Force> <Quality> <Text> [CR] [LF]
```

Answer :

```
INSERTTEXTLINE OK[CR] [LF]           → OK  
INSERTTEXTLINE BAD ARGUMENTS [CR] [LF] → errors in parameters
```

Data :

X, Y, Z	Coordonate of the line, in tenth of mm
W, H	Width and height of the chars, in tenth of mm
Angle	In hundredth of degrees from -18000 to 18000
Radius	In tenth of mm
Space	space between chars (from 0 to 50)
Force	From 0 to 9
Quality	From 1 to 9
Text	Text to be marked

Example : *Line at X=10mm, Y=12mm, 5x7mm chars at force 5, quality double, text=HELLO WORLD*

```
String to send :  
INSERTTEXTLINE 100 120 0 50 70 0 0 2 5 2 HELLO WORLD [LF]  
String received:  
INSERTTEXTLINE OK [CR] [LF]
```

• **INSERTPAUSELINE** *insert a line to the current file*

Syntax :

INSERTPAUSELINE <X> <Y> <Z> [CR] [LF]

Answer :

INSERTPAUSELINE OK [CR] [LF] → OK
 INSERTPAUSELINE BAD ARGUMENTS [CR] [LF] → errors in parameters

Data :

X, Y, Z Coordonate of the line, in tenth of mm

Example : *Line at X=10 mm, Y=12 mm, Z=13 mm*

String to send :
 INSERTPAUSELINE 100 120 130 [LF]

String received:
 INSERTPAUSELINE OK [CR] [LF]

• **INSERTLOGOLINE** *insert a logo to the current file*

Syntax :

INSERTLOGOLINE <X> <Y> <Z> <W> <H> <Angle> <Radius> <Prop.> <Force> <Quality> <Type> <Logo> [CR] [LF]

Answer :

INSERTLOGOLINE OK [CR] [LF] → OK
 INSERTLOGOLINE BAD ARGUMENTS [CR] [LF] → errors in parameters

Data :

X, Y, Z Coordonate of the line, in tenth of mm
W, H Width and height of the chars, in tenth of mm
Angle In hundredth of degrees from -18000 to 18000
Radius In tenth of mm
Prop. 1 to keep proportion of original logo, 0 to stretch the logo within W and H
Force From 0 to 9
Quality From 1 to 9
Type V for Vectorial ?logo, D for Dot marking logo
Logo Name of the logo

Example : *Vectorial logo named SIC at X=10mm, Y=12mm, 10x10mm chars at force 5, quality 4, text=HELLO WORLD*

String to send :
 INSERTLOGOLINE 100 120 0 100 100 0 0 1 5 4 V SIC [LF]

String received:
 INSERTLOGOLINE OK [CR] [LF]

• **INSERTTECC200LINE** *insert a Data matrix to the current file*

Syntax :

INSERTTECC200LINE <X> <Y> <Z> <W> <H> <Angle> <Format> <Force> <Ref> <Speed> <Text> [CR] [LF]

Answer :

INSERTTECC200LINE OK [CR] [LF] → OK
 INSERTTECC200LINE BAD ARGUMENTS [CR] [LF] → errors in parameters

Data :

X, Y, Z Coordonate of the line, in tenth of mm
W, H Width and height of the chars, in tenth of mm
Angle In hundredth of degrees from -18000 to 18000
Format format of the DataMatrix from 0 to 16 :
 0 = AutoSquare, 1=AutoRectangular, 2="10x10", 3="12x12", 4="14x14", 5="16x16", 6="18x18", 7="20x20", 8="22x22",
 9="24x24", 10="26x26", 11=" 8x18", 12=" 8x32", 13="12x26", 14="12x36", 15="16x36", 16="16x48"
Force From 0 to 9
Ref 0 for simple reference, 1 for double reference
Speed From 1 to 9
Text Text to be marked

Example : *Ecc200 at X=15mm, Y=14mm, 10x10mm chars at force 5, Simple reference, speed=3, text to encode=HELLO WORLD*

String to send :
 INSERTTECC200LINE 150 140 0 100 100 0 0 5 0 3 HELLO WORLD [LF]

String received:
 INSERTTECC200LINE OK [CR] [LF]

• **INSERTAUTOZLINE** *make an auto-sensing in current file*

Syntaxe :

INSERTAUTOZLINE <X> <Y> <1st dot> <DZ> <Zmin> <Zmax >[CR] [LF]

Answer :

INSERTAUTOZLINE OK[CR] [LF] → OK
 INSERTAUTOZLINE BAD ARGUMENTS [CR] [LF] → errors in parameters

Data :

X, Y Coordonate of the line, in tenth of mm
 1st dot "1" for YES and "0" for NO
DZ Stylus/Workpiece marking distance, in tenth of mm (from 0mm to 9.9 mm)
Zmin, Zmax minimum move and maximum move, in tenth of mm

Example : *Auto Sensing at 1st dot with part between 50mm and 60 mm and a Stylus/Workpiece marking distance of 6mm*

String to send :

INSERTAUTOZLINE 100 120 1 60 500 600[LF]

String received:

INSERTAUTOZLINE OK[CR] [LF]

Example : running a new file with one AutoZ and marking the text AZERTY

NEWFILE 5 7 0[LF]INSERTAUTOZLINE 100 120 1 60 500 600[LF]
 INSERTTEXTLINE 100 100 0 50 50 0 0 2 2 2 AZERTY[LF]RUN[LF]

• **INSERTMOVEZLINE** *insert a line to the current file*

Syntaxe :

INSERTMOVEZLINE <X> <Y> <DZ> [CR] [LF]

Answer :

INSERTMOVEZLINE OK[CR] [LF] → OK
 INSERTMOVEZLINE BAD ARGUMENTS [CR] [LF] → errors in parameters

Data :

X, Y Coordonate of the line, in tenth of mm
DZ relative coordonate from urrent Z position
 A negative value moves the head up
 A positive value moves the head toward the part

Example : *move 10 mm up*

String to send :

INSERTMOVEZLINE 100 100 -100[LF]

String received:

INSERTMOVEZLINE OK[CR] [LF]

• **INSERTROTATIONLINE** *insert a line to the current file*

Syntaxe :

INSERTROTATIONLINE <X> <Y> <Angle> <PASS TO XY>[CR] [LF]

Answer :

INSERTROTATIONLINE OK[CR] [LF] → OK
 INSERTROTATIONLINE BAD ARGUMENTS [CR] [LF] → errors in parameters

Data :

X, Y Coordonate of the line, in tenth of mm
Angle In hundredth of degrees from -18000 to 18000
Pass to XY 1 : the marking head goes to XY coordonate before executug the rotation,
 0 : the rotation is made without movong the marking head

Example : *rotate of 90 degres*

String to send :

INSERTROTATIONLINE 0 0 9000 0[LF]

String received:

INSERTROTATIONLINE OK[CR] [LF]

• **SETLINEOPTION**

Set options for the last inserted line of the current file

Syntaxe :

```
SETLINEOPTION <FontName> [CR] [LF]
SETLINEOPTION <FontName> <Inclination> [CR] [LF]
SETLINEOPTION <FontName> <Inclination> <Orientation> [CR] [LF]
SETLINEOPTION <FontName> <Inclination> <Orientation> <Mirror> [CR] [LF]
SETLINEOPTION <FontName> <Inclination> <Orientation> <Mirror> <Center> [CR] [LF]
SETLINEOPTION <FontName> <Inclination> <Orientation> <Mirror> <Center> <His> [CR] [LF]
SETLINEOPTION <FontName> <Inclination> <Orientation> <Mirror> <Center> <His> <Speed> [CR] [LF]
```

Answer :

```
SETLINEOPTION OK [CR] [LF] → OK
SETLINEOPTION BAD ARGUMENTS [CR] [LF] → errors in parameters
```

Data :

FontName	OCR, OCRA, COURIER, ARIAL
Inclination	from -120 to 120
Orientation	0 : Abc 1 : α 2 : ⊖q⊕ 3 : ∂
Mirror	0 : Abc 1 : ꞥꞥc 2 : ꞥꞥA
Center	0 : No 1 : Yes
His	0 : No 1 : Yes
Speed	from 1 to 9

Example : *ARIAL + Center*

String to send :

```
SETLINEOPTION ARIAL 0 0 0 1 [LF]
```

String received:

```
SETLINEOPTION OK [CR] [LF]
```

• **SAVEFILE**

save current file

Syntaxe :

```
SAVEFILE <File name> [CR] [LF]
```

Answer :

```
SAVEFILE OK [CR] [LF] → OK
SAVEFILE BAD ARGUMENTS [CR] [LF] → errors in parameters
```

Data :

File name (optional) 11 char max, in upper case

Example :

String to send :

```
SAVEFILE MYFILE [LF]
```

String received:

```
SAVEFILE OK [CR] [LF]
```

• **FILEDELETE**

erase a file into the control

Syntaxe :

```
FILEDELETE <File name> <File kind>
```

Data :

File name File name to delete in upper case, up to 11 chars
File kind : 2 = for a marking file
4 = for a dot logo
5 = for a vectorial logo

Example : *deleting file named : MYFILE*

String to send :

```
FILEDELETE MYFILE 2 [LF]
```

String received:

```
FILEDELETE OK [CR] [LF]
```


f) settings functions

These functions let you access to system functions of the controller, as setting the date-time, setting global variables, get the program version.

- **SETSHIFTINCVAR** *setting setting a shift increment to the current file*

Syntaxe :

```
SETSHIFTINCVAR < Increment name > <Shift> < Value >[CR] [LF]
```

Answer :

```
SETSHIFTINCVAR OK[CR] [LF]           → OK  
SETSHIFTINCVAR VAR NOT FOUND [CR] [LF] → not found
```

Data :

Increment name in upper case
Shift shift number
Value caution, the value is the decimal value of the increment.

Example : *setting the value 12 for the first shift to the increment named NB_PART*
String to send :

```
SETSHIFTINCVAR NB_PART 1 12[LF]
```

String received:

```
SETSHIFTINCVAR OK[CR] [LF]
```

- **SETGLOBALVAR** *Affectation variable globale*

Syntaxe :

```
SETGLOBALVAR <Number> <Value>[CR] [LF]
```

Answer :

```
SETGLOBALVAR OK[CR] [LF]           → OK
```

Data :

Number number of the global var (from 1 to 10)
Value Value to set

Example : *setting the first global variable to FACTORY*
Envoie de :

```
SETGLOBALVAR 1 FACTORY_ONE[LF]
```

String received:

```
SETGLOBALVAR OK[CR] [LF]
```

• **SETGLOBALINC** *Affectation incrément globale*

Syntaxe :

```
SETGLOBALINC <Number> <Value >[CR][LF]
```

Answer :

```
SETGLOBALINC OK[CR][LF] → OK
```

Data :

Number number of the global increment (from 1 to 10)
Value Value to set

Example : *setting the first global increment to 532*

String to send :

```
SETGLOBALINC 1 532[LF]
```

String received:

```
SETGLOBALINC OK[CR][LF]
```

• **GETVERSION** *get the program version*

Syntaxe :

```
GETVERSION (No data)
```

Example :

String to send :

```
GETVERSION [LF]
```

String received:

```
GETVERSION 5-0b4[CR][LF]
```

• **GETDATETIME** *getting the date and the time*

Syntaxe :

```
GETDATETIME (No data)
```

Response :

```
GETDATETIME <Year> <Month> <Day> <Hour> <Minutes> <Secondes>
```

Year 4 digits

Month, Day, Hour, Minutes, Secondes written with 2 digits, with a zero on the left if necessary

Example : *we are on the 5th of jun 2007 and it is 14h25:30*

String to send :

```
GETDATETIME[LF]
```

String received:

```
GETDATETIME 2007 06 05 14 25 30 [CR][LF]
```

• **SETDATETIME** *setting the date and the time*

Syntaxe :

```
SETDATETIME <Year> <Month> <Day> <Hour> <Minutes> <Secondes>
```

Data :

Year written with 4 chars

Month, Day, Hour, Minutes, Secondes written with 2 chars, with a zero on the left if necessary

Example : *we are on the 5th of jun 2003 and it is 14h25:30*

String to send :

```
SETDATETIME 2007 06 05 14 25 30[LF]
```

String received:

```
SETDATETIME OK[CR][LF]
```

3 - Binary protocol

a) String to be sent :

There are two format for the string to be send :(a) : with a check sum
(b) : without the check sum

Control code CheckSum : In order to detect a possible error in the transmission, the CheckSum is calculated depending on the string sent by the main system and receptioned by the machine. If the string has been correctly transmitted, the code calculated by the marking machine is the same as the code sent by the main system..

The **CheckSum** corresponds to an "EXCLUSIVE OR" of all codes transmitted in the string, including the STX code and the ETX code.

String (a) : [STX] [NULL] [Version] <Command 1><Command 2> . . . <Command n> [ETX]
String (b) : [STX] [Version] <Command 1> . . . <Command n> [ETX] [Check-sum]

Syntax of a Command:

A command is build with :

The command code (written with 1 byte)
The size of the data (a 16-bit integer Written with 2 bytes)
The data (written with n bytes).

The command code can be any value of the ASCII table between [0x04] and [0xFF]
Any byte of data can take any value of the ASCII code

If you do not wish to specify the size of the data, you can format the command using a break-code :

The command code (written with 1 byte)
The ASCII code 255 to specify you are using a break code (written with 1 byte)
The break-code (written with 1 byte)
The data (written with n bytes)

Make sure that the data do not have the break-code inside

format (1) : [Command Code] [data size (2)] [Data (Size)]
format (2) : [Command Code] [0xFF] [Break-code] [Data (Size)] [Break-code]

Conventions :

Green : start and end of the string
Blue : Command code
pink : Data size
Yellow : Data

[STX] and [ETX] = Start and End of text
[NULL] after [STX] disable the check-sum control
[Version] = '5' (0x35 in hexadecimal) for the protocol version.
[Code] = Command code, written with one byte (from [0x04] to [0xFF])
[Data size(2)] = Size of the data = a 16 bit integer written with 2 bytes, with the most significant byte to the left (Motorola processor)
[0xFF] = char 0xFF for break-code
[Break-code] = any byte from 0x00 to 0xFF
[Data] = data of the command.

Representation of numeric values:

All numbers are written with the most significant byte to the left
Example : the value 513 is written [02] [01]

b) The control response to all commands :

- if an error occurs when the string is analyzed, the control send:

[BS] : for a check-sum error
[HT] : for a bad string syntax
[NAK] : for a Time out

- when the string is correct, the control respond to all commands with the following string.

String sent by control : [STX] <Command 1 answer>< Command 2 answer> . . . <Command n answer> [ETX]
The command are built : [original command code] [answer size(2 bytes)] [anwer(n bytes)]

In many cases, the anwer is the return code of the execution of the command :

[original command code] [0x00] [0x01] [return code (1 byte)]

the [return code] can be one of the following value

[ACK]	successful
[HT]	wrong data syntax
[BEL]	file not found
[LF]	Variable not found

c) **commands list** :

- Commun functions :

These functions let you select a marking file in the controller, affect variables and run the marking cycle.

Command name	Command code	description
LOAD FILE	'c'	load a file
FILE SET VAR	'7'	set a var in the current file
START MARKING	'g'	run the marking file
RESET ERREUR	'E'	reset errors

- Managing files functions :

With these functions, you will be able to:

- transfer all marking parameters :

First, create a free file using `NEWFILE` command, then insert the line with the `INSERT*LINE` function, then you can save the file for a later use with the `SAVEFILE` command or run the cycle with the `RUN` command

- Send and save marking file or logo to the controller:

First, create a free file using `NEWFILE` command, then insert the line with the `INSERT*LINE` function, then you can save the file for a later use with the `SAVEFILE` command or run the cycle with the `RUN` command

Command name	Command code	Ddescription
NEW_FILE	'f'	new empty file
INSERT_LINE_TO_FILE	'l'	add a line into the current file
FILE_SET_OPTION	'o'	add a option to the current file
LINE_SET_OPTION	'a'	add a option to the last line of the current file
SAVE_FILE	'e'	save a file
LISTE_FAT	'L'	list the files in the control
PC_VERS_E6	'G'	Send a file to the control
E6_VERS_PC	'S'	Get a file from the control
DEL_FILE	'D'	delete a file in the control

- Manual programming functions:

These functions are used to take control of the machine :

Send the machine to the origine, go to a specified point then shoot the stylus, set outputs and read inputs.

Command name	Command code	Description
ORIGINE	'H'	go to home position
IMPACT	'P'	make a impact
SET_OUTPUT	'Z'	Set the value of an output
GET_INPUT	'Y'	get the inputs status

- settings functions:

These functions let you access to system functions of the controller, as setting the date-time, setting global variables, get the program version, get the machine configuration.

Command name	Command code	Description
RESTART	'*'	restart the control
SYNCHRO_DATEHEURE	'h'	set the control date/time
VITESSE_COM	'v'	change speed of a control communication port
GET_CONFIG_AXE	'X'	get axis parameter
SET_CONFIG_IMPACT	'i'	set impact parameter
GET_CONFIG_IMPACT	'I'	get impact parameter
SET_VAR_GLOBAL	'8'	set a global var
SET_INC_GLOBAL	'9'	set a global increment
SET_SHIFT_INC_FICHER	'0'	set a shift var to the current file
GET_OPTION	'1'	\
SET_OPTION	'2'	} control option management
DEL_OPTION	'3'	/
GET_OPTION_DALLAS	'4'	machine option management

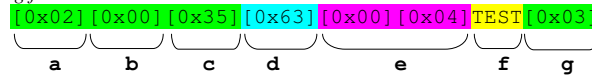
d) **Commun functions :**

• **LOAD FILE** Code 'c' (0x63 in hexadecimal)

Data = [File name (1 to 11 bytes)]
 Answer = [Return code (1)]

Example :

Loading file named TEST



- a = Start of Text [STX]
- b = Desable check-sum
- c = Protocol version
- d = Command code
- e = Data size
- f = File name
- g = End of text [ETX]

• **FILE SET VAR** Code '7' (0x37 in hexadecimal)

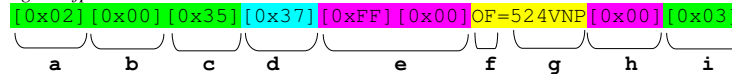
Data = [variable name (20 bytes max)] '=' [Value (1 byte to 127 bytes)]
 Answer = [Return code (1)]

Variable name : in upcase, cannot be longer then 20 char, cannot include spaces
 '=' : char 0x3D in hexadecimal

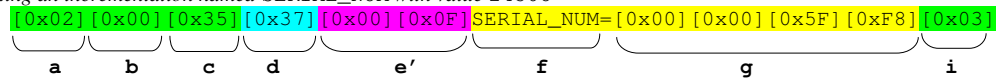
Value : for a alpha-numeric variable, the value is the text of the var
 : for increment variable, the value is a 32-bit integer (the most significant byte to the left)
 : for shift variable, the value is 10 32-bit integer (40 bytes)

Example :

Setting an apphanumeric variable named OF with value 524VNP



Setting an incrementation named SERIAL_NUM with value 24568



- a = Start of Text [STX]
- b = Desable check-sum
- c = Protocol version
- d = Command code
- e = break-code setting
- e' = size
- f = Name de la variable
- g = value
- h = break-code
- i = End of text [ETX]

• **START MARKING** Code 'g' (0x67 in hexadecimal)

Data = [mode (0 ou 1)]
 Answer = [Return code (1)]

Then if Return code is [ACK]:

- [EOT] at end of last impact
- [ENQ] when machine back to home position
- [NAK] [machine status (3)] when an error occurs during marking

Mode : [0x00] marking, [0x01] simulation (no impact, stylus always up)
 if marking mode is empty, then use of normal marking status.

Machine status = see fig 1 page 5

If there is PAUSE in the marking file :

At a PAUSE line, the control send the char P [0x50] and wait for :

- response p [0x70] from RS232,
 - or the Start button to be pressed,
- to continue the marking,

• **RESET_ERREUR** Code 'E' (0x45 in hexadecimal)

Data = no data
 Answer = [Return code (1)]

You must reset the error if the machine status show errors (see fig 1 at the end of the document)

e) managing files functions

•NEW_FILE Code 'f' (0x66 in hexadecimal)

Data = [Marking speed (1)] [fast speed (1)] [crossed zero (1)] [File name (0 to 11 bytes)]
 Answer = [Return code (1)]

Marking speed : speed from [0x01] to [0x09] (or '1' to '9')
Fast speed speed from [0x01] to [0x09] (or '1' to '9')
Crossed zero '1', '0', 1, ou 0
File name optional

Example :

New file named MY_FILE, with marking speed 4 ,fast speed 8 and zero not crossed

[0x02] [0x00] [0x35] [0x66] [0x00] [0x0A] [0x04] [0x08] [0x00] MY_FILE [0x03]

a b c d e f g h i

new file, with marking speed 6 ,fast speed 9 and zero crossed

[0x02] [0x00] [0x35] [0x66] [0x00] [0x03] [0x06] [0x09] [0x01] [0x03]

a b c d e f g i

a = Start of Text [STX] b = Desable check-sum
 c = Protocol version d = Command code
 e = Data size f = speed
 g = crossed zero h = File name
 i = End of text [ETX]

•INSERT_LINE_TO_FILE Code 'l' (0x6C in hexadecimal)

Data = [X (2)][Y (2)][Z (2)][W (2)][H (2)][Esp (1)][F (1)][Qua (1)][Kind (1)][Text (0 to 127 bytes)]
 Answer = [Return code (1)]

X, Y, Z : a 16-bit signed integer (in thenth of mm)
W, H : a 16-bit signed integer (in thenth of mm)
E : space between char : a 8-bit integer from [0x00] to [100]
F : Force of impact : from [0x00] to [0x09]
Qua : Quality of marking from [0x01] to [0x09]
Kind and Text : See below :

Kind text zone format

[0x00] for Text [Font kind (1)] [Font name (11)] [Reserved (1)] [Text (0 to 114 bytes)]
 Font kind = [129] for Font_9x13 (OCR and OCRA), and [131] for Font_TT (COURIER and ARIAL)
 Font name = File name of the font written with 11 bytes, padding with NULL char to the right
 Reserved = a NULL char
 Text = text to print, can include variables, date-time, shift, ...

[0x01] for Logo [Logo kind (1)] [Prop (1)] [Reserved (1)] [Logo name (11)] [Reserved (1)]
 Logo kind = [0x04] for dot logo, and [0x05] for vectorial logo
 prop = [0x01] to keep proportion of the logo Width and Height, [0x00] to stretch the logo to W and H
 Reserved = a NULL char
 Logo name = File name of the logo Written with 11 bytes, padding with NULL char to the right

[0x02] for Ecc200 [Format (1)] [Text (0 to 126 bytes)]
 Format = [0x00] Auto-Square, [0x01] Auto-Rectangle
 Text = text to print, can include variables, date-time, shift, ...

[0x03] for RS232 [Port (1)] [Stop] [TimeOut (2 bytes)] [S-Send] [S-Resp] [Send] [Response]
 Port = [0x00] for the SERIAL port et [0x01] for the HOST port
 Stop = [0x01] Stop marking if timeout is reached before the response is received, [0x00] go to next line after time-out or respose received
 TimeOut = in milli-seconds : a 16-bit integer, the most significant byte to the left.
 S-Send = size of the text to send : a 8-bit integer
 S-Resp = size of the text of the response to wait for: a 8-bit integer
 Send = text to send, can include variables, date-time, shift, ...
 Response = Response to wait for, can include variables, date-time, shift, ...

[0x04] for I/O managing [Output (1)] [Val (1)] [Action (1)] [Input (1)] [Val (1)] [Delay (2)] [Stop [0x01]]
 Output = Output number to set : [0x01] to [9], [0x00] for none
 Val = Value of the output to set : [0x01] Output is close, [0x00] output is open
 Action = [0x01] Set the output to the oposit stat of Val at the end.
 Input = Input number to scan : [0x01] to [9], [0x00] for none
 Delay = in milli-seconds : a 16-bit integer, the most significant byte to the left.
 Stop = [0x01] Stop marking if timeout is reached before the response is received, [0x00] go to next line after time-out or response received

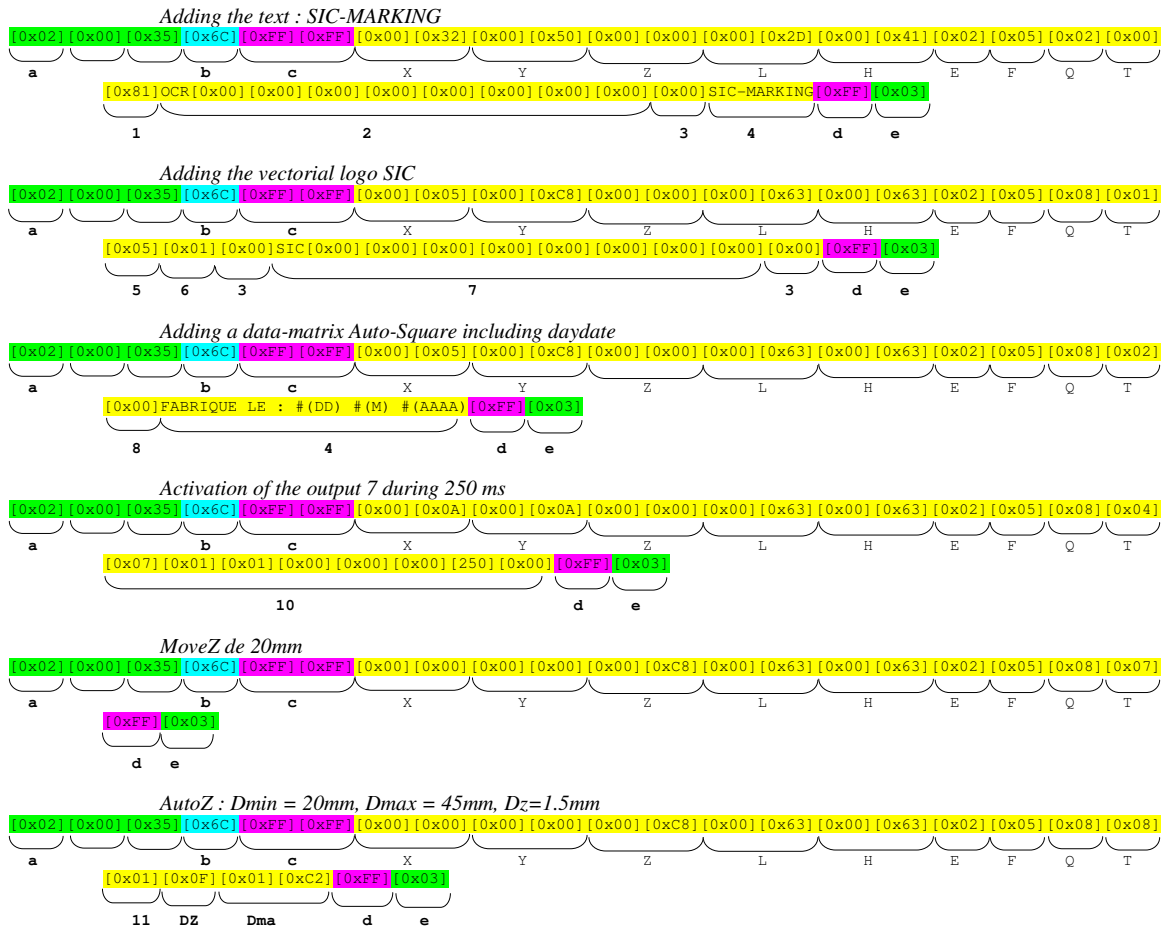
[0x05] for Pause *No data*
 At a PAUSE line, the control send the char P [0x50] and wait for :
 { - response P [0x70] from RS232 , to continue the marking,
 - or the Start button to be pressed

[0x06] for Rotation [Goto XY (1)]
 Goto XY = [0x01] if the stylus goes to the position XY before rotation, [0x00] if the Rotation is done without moving the stylus
 The rotation angle value is set in the line Attribut option (see page 23)

[0x07] for Move Z *No data*
 the Z value is set in the Z field

[0x08] for Auto Z [1st dot (1)] [DZ (1)] [Dmax (2)]
 1st dot = [0x01] to do the Autosensing at the first impact, [0x00] to do the Autosensing at the XY coordonate of that line.
 Dz = from [0x01] to [0x63]; Stylus/part distance in 10th of mm
 Dmax = Maximum move in 10th of mm
 the Dmin move is set in the Z filed

Examples :



- | | |
|--------------------------------|--------------------------|
| a = Start of Text [STX] | 4 = Text |
| b = Command code | 5 = Logo Kind |
| c = break-code setting | 6 = proportional |
| d = break-code | 7 = Logo name |
| e = End of text [ETX] | 8 = Ecc200 format |
| 1 = font kind | |
| 2 = font Name | 10 = I/O setting |
| 3 = reserved [NUL] | 11 = 1st dot |

- FILE_SET_OPTION** Code 'o' (0x6F in hexadecimal) Marking file Option

Data = [Option code (1)] [Data (n)]
 Answer = [Return code (1)]
 See page 23 for more details

- LINE_SET_OPTION** Code 'a' (0x61 in hexadecimal) last line of marking file Option

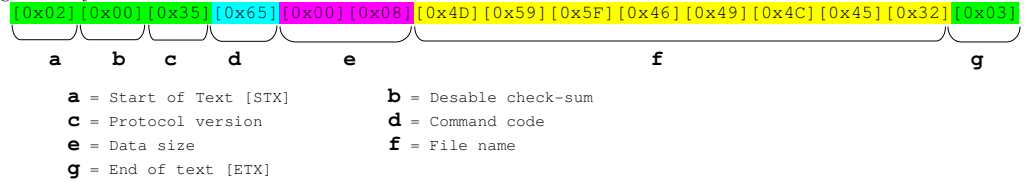
Data = [Option code (1)] [Data (n)]
 Answer = [Return code (1)]
 See page 23 for more details

• **SAVE_FILE** Code 'e' (0x65 in hexadecimal)

Data = [File name (0 to 11 bytes)]
 Answer = [Return code (1)]
File name : optional, if present, its rename current file

Example :

Saving current file to MY_FILE2

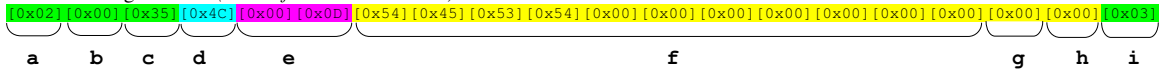


• **FILE LIST** Code 'L' (0x4C in hexadecimal)

Data = Name[0x0B] reserved[0x01] kind[0x01]
 Answer = (Name[0x0B] reserved[0x01] Kind[0x01] Size[0x02] reserved[0x04]) x (number of file found)
Name : file Name to look for, or 11 NULL char for every file
 File name must be 12 bytes long, padding with NULL char to the right
 File name must be in upper-case without space (char 0x20).
reserved : one reserved byte, must be NULL
Kind : 0 = all kinds 2 = marking file 4 = dot logo file 5 = scribbling logo file

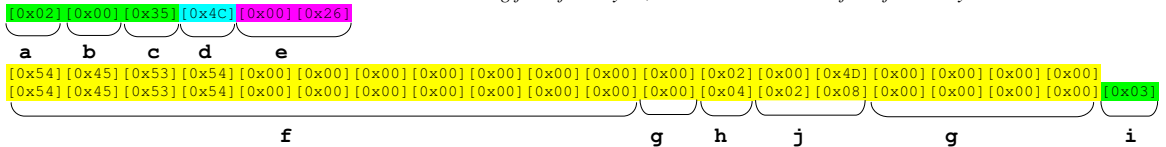
Example :

String to send : (list all file named : TEST)

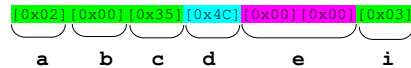


Answer string:

- When there is two files TEST : one marking file of 77 bytes, and one DOT LOGO file of 520 bytes



- When there is no file TEST

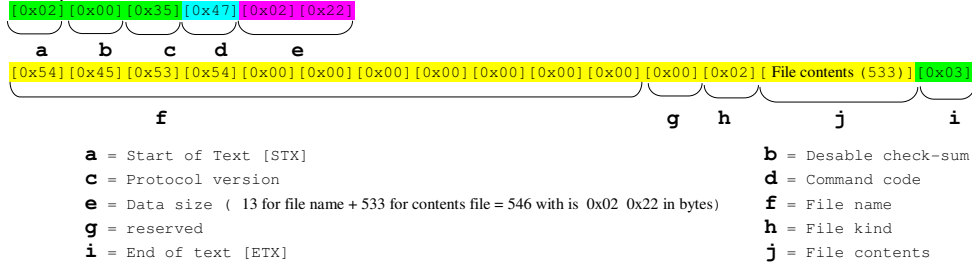


a = Start of Text [STX] b = Desable check-sum
 c = Protocol version d = Command code
 e = Data size f = File name
 g = reserved h = File kind
 i = End of text [ETX] j = File size

• **PC_TO_E6** Code 'G' (0x47 in hexadecimal)

Data = Name [0x0B] reserved [0x01] Kind [0x01] [File contents (n bytes)]
 Answer = [Return code (1)]
Name : file Name to look for, or 12 *NULL* char for every file
 File name must be 12 bytes long, padding with *NULL* char to the right
 File name must be in upper-case without space (char 0x20).
reserved : one reserved byte, must be *NULL*
Kind : 0 = all kinds 2 = marking file
 4 = dot logo file 5 = scribbling logo file
File contents : all the bytes of the file

Example :



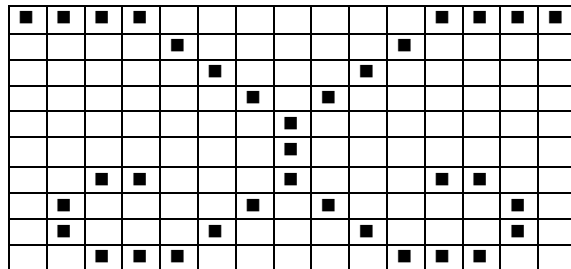
Contents of a Dot logo file :

[Size X][Size Y][X1][Y1][X2][Y2]...[Xn][Yn]

Size X = Logo width from 1 to 255
 Size Y = Logo height from 1 to 255

X_i, Y_i : coordData du point i du logo

Example :



That logo is 15 dots width and 10 dots height
 This is the position of these dots (0 is the bottom left corner)

X	4	3	2	2	3	4	5	6	7	8	9	10	11	12	13	14	13	12	15	14	13	12	11	10	9	8	8	7	6	5	4	3	2	1
Y	4	4	3	2	1	1	2	3	4	3	2	1	1	1	2	3	3	10	10	10	10	9	8	7	6	5	7	8	9	10	10	10	10	

This is the logo file

[0x0F] [0x0A] [0x04] [0x04] [0x03] [0x04] [0x02] [0x03]... [0x02] [0x0A] [0x01] [0x0A]

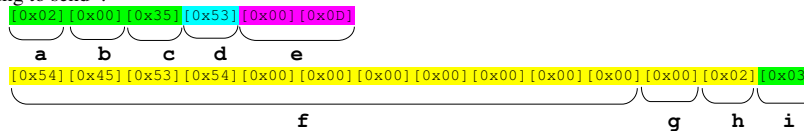
W
H
X1 Y1
X2 Y2
X3 Y3

• **E6_TO_PC** Code 's' (0x53 in hexadecimal)

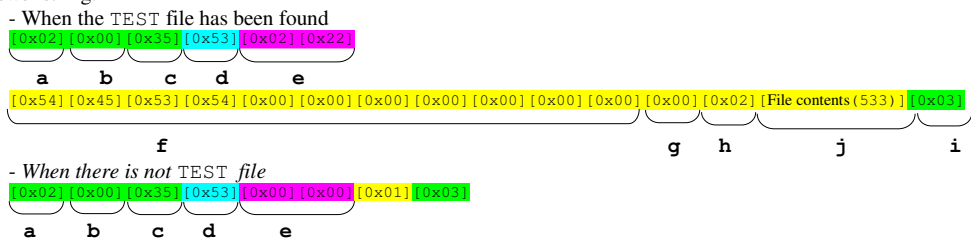
Data = Name [0x0B] reserved [0x01] Kind [0x01]
 Answer = Name [0x0B] reserved [0x01] Kind [0x01] [File contents(n bytes)]
Name : file Name to look for, or 12 *NULL* char for every file
 File name must be 12 bytes long, padding with *NULL* char to the right
 File name must be in upper-case without space (char 0x20).
reserved : one reserved byte, must be *NULL*
Kind : 0 = all kinds 2 = marking file
 4 = dot logo file 5 = scribbling logo file
File contents : all the bytes of the file

Example :

String to send :



Answer string:

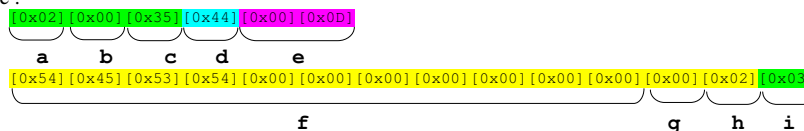


- a = Start of Text [STX]
- b = Desable check-sum
- c = Protocol version
- d = Command code
- e = Data size
- f = File name
- g = reserved
- h = File kind
- i = End of text [ETX]
- j = File contents
- k = Error : file not found

• **DEL_FILE** Code 'D' (0x44 in hexadecimal)

Data = Name[0x0C] Kind[0x01]
 Answer = [Return code(1)]
Name : File name to be deleted (written in upper case)
Kind : 2 = Marking file
 4 = Dot logo file
 5 = Scribbling logo file

Example :



- a = Start of Text [STX]
- b = Desable check-sum
- c = Protocol version
- d = Command code
- e = Data size
- f = File name
- g = reserved
- h = File kind
- i = End of text [ETX]

f) File options definition:

- Diameter Option** Setting the part diameter (for the D-Axis use only)
 Code [0x02] (0x02 in hexadecimal)
 Data=[Diameter (2)]
 Total command data size =3
Diameter in 10th of milimeters : a 16 bit integer, the most significant byte to the left
 Example :
 Setting a 30 mm diameter.

[0x02]	[0x00]	[0x35]	[0x00]	[0x03]	[0x02]	[0x01]	[44]	[0x03]
a	b	c	d	e	f	g	h	
a = Start of Text [STX]					b = Desable check-sum			
	c = Protocol version				d = Command code			
	e = size				f = Option code			
	g = Diameter				h = End of text [ETX]			

- Comment Option** Setting the file comment (free text)
 Code [0x03] (0x03 in hexadecimal)
 Data=[Comment(1 to 39 bytes)] [Reserved(1)]
 Total command data size, from 2 bytes to 41 bytes
Comment : Free text (39 maximum chars)
Reserved : char [0x00] in hexadecimal
 Example :
 Setting the comment : "Part ZCB 1245".

[0x02]	[0x00]	[0x35]	[0xFF]	[0xFF]	[03]	Part	ZCB	1245	[0x00]	[0xFF]	[0x03]
a	b	c	d	e	f	g	h	i	j		
a = Start of Text [STX]					b = Desable check-sum						
	c = Protocol version				d = Command code						
	e = size				f = Option code						
	g = Comment				h = Reserved [NULL]						
	i = break-code				j = End of text [ETX]						

g) Last marking file line option description :

- Attribut option** Pritable setting
 Code [0x01] (0x01 in hexadecimal)
 Data=[B (1)] [I (2)] [A (2)] [R (2)]
 Total command data size =8
B Attribut Byte : let's split that attribut into 8 bits: (R A C I V H 1 2).
 R : reserved A : there is a non zero angle or a non zero raduis
 C : Center the text with the X and Y coordonate I : there is a non zero italic value
 V : Vertical mirror H : Horizontal mirror
 1 : 180° rotate 2 : 90° rotate
I Italic value : a 16-bit signed integer the most significant byte to the left (the value 100 is for a 45° italic)
A Rotate angle : a 16-bit signed integer the most significant byte to the left, in 100th degré (from -18000 to +18000)
R Radius : a 16-bit signed integer the most significant byte to the left, en 10th of mm
 Example :
 Center the line and a 10% italic value.

[0x02]	[0x00]	[0x35]	[0x00]	[08]	[01]	[48]	[0x00]	[0x0A]	[0x00]	[0x00]	[0x00]	[0x00]	[0x03]
a	b	c	d	e	f	B	I	A	R	g			
a = Start of Text [STX]					b = Desable check-sum								
	c = Protocol version				d = Command code								
	e = size				f = Option code								
	g = End of text [ETX]												

- Indexor Option** D-axis setting
 Code 2 (0x02 in hexadecimal)
 Data=[On/Off (1)]
 Total command data size =2
On/Off [0x00] not active, [0x01] Active
 That option is used only if the part diameter is not null
 Example :
 Activate the Indexor mode

[0x02]	[0x00]	[0x35]	[0x00]	[0x02]	[0x02]	[0x01]	[0x03]
a	b	c	d	e	f	g	h

h) manual programming

• **ORIGINE** Code 'H' (0x48 in hexadecimal)

Data = [Motors list (1)] (optional)
 Answer = [machine status (3)]
Motors list = binary mix of these values : 0x01 for X axis, 0x02 for Y axis and 0x04 for accessory axis
 Default value is 0x07
Machine status = see fig 1 page 5

Example :

Home position for X, Y et Z
 [0x02] [0x00] [0x35] H [0x00] [0x00] [0x03]

home position for Y axis
 [0x02] [0x00] [0x35] H [0x00] [0x01] [0x02] [0x03]

a	b	c	d	e	f	g
a	b	c	d	e	f	g
a	b	c	d	e	f	g
a	b	c	d	e	f	g
a	b	c	d	e	f	g
a	b	c	d	e	f	g
a	b	c	d	e	f	g

• **IMPACT** Code 'P' (0x50 in hexadecimal)

Data = [Speed(1)] ([Xi(4)] [Yi(4)] [Zi(4)] [Fi(1)]) ([X2(4)] [Y2(4)] [Z2(4)] [F2(1)]) ...
 Answer = [machine status (3)]

Vitesse : marking speed from [0x01] to [0x09]
Xi : X coordonate of impact number *i* (a 32 bit signed integer, the most significant byte to the left) in step
Yi : Y coordonate of impact number *i* (a 32 bit signed integer, the most significant byte to the left) in step
Zi : Z coordonate of impact number *i* (a 32 bit signed integer, the most significant byte to the left) in step
Fi : - force of impact ([0x00]=no impact, [0x09]=maximum impact) for percussion machine,
 - [0x01] Stylus Up, [0x00] = Stylus Down for scratching machine.

Machine status = see fig 1 page 5

Example :

[0x02] [0x00] [0x35] P [0x00] [0x1B] [0x08] [0x00] [0x00] [0x02] [0x00] [0x00] [0x00] [0x02] [0x00] [0x00] [0x00] [0x00] [0x00] [0x05]
 [0x00] [0x00] [0x01] [0x00] [0x00] [0x00] [0x00] [0x01] [0x00] [0x00] [0x00] [0x00] [0x00] [0x00] [0x05] [0x03]

a	b	c	d	e	f	g	h	i	j	k
a	b	c	d	e	f	g	h	i	j	k
a	b	c	d	e	f	g	h	i	j	k
a	b	c	d	e	f	g	h	i	j	k
a	b	c	d	e	f	g	h	i	j	k
a	b	c	d	e	f	g	h	i	j	k
a	b	c	d	e	f	g	h	i	j	k

• **SET_OUPUT** Code 'Z' (0x5A in hexadecimal)

Data = [Number (1)] [Value]
Number : This is the output number from 0x01 to 0x08
value : the value to set : 0x00 (off) or 0x01 (on)

• **GET_INPUTS** Code 'Y' (0x59 in hexadecimal)

Data = none
 Answer = [Input status (1)]
InputStatus : This is the binary combinason of all inputs 1 to 8
 A return value 0 meens all inputs are inactive.
 A return value 5 meens input 1 and 3 are active.
 A return value 128 meens input 8 is active.

i) Settings functions

• **RESTART** Code '*' (0x2A in hexadecimal)

Data = no data
 Answer = [Return code(1)]

At the beginning of start, the control send : booting... [CR] [LF]
 When it is ready, the control send : e6 ready[CR] [LF]

You must restart the control after setting control options or machine options.

Example :

[0x02] [0x00] [0x35] * [0x00] [0x00] [0x03]

• **SYNCHRO_DATEHEURE** Code 'h' (0x68 in hexadecimal)

Data = AAAA-MM-JJ HH:MM:SS
 Answer = [Return code(1)]
AAAA : Year written with 4 chars
MM : Month written with 2 chars
JJ : Day written with 2 chars
hh : Hour written with 2 chars
mm : Minutes written with 2 chars
ss : Seconds written with 2 chars

Example :

[0x02] [0x00] [0x35] h [0x00] [0x13] 2003-05-14 14:02:31 [0x03]

a	b	c	d	e	f	g
a = Start of Text [STX]	b = Desable check-sum	c = Protocol version	d = Command code	e = Data size (19)	f = Date-Time	g = End of text [ETX]

Note : If the task-bar of the control freeze, restart the control.

• **VITESSE_COM** Code 'v' (0x76 in hexadecimal)

Data = [Port(1)] [Speed(1)]
 Answer = [Return code(1)] then serial port change speed

Port : 'H' ou '1' for the HOST port,
 'S' ou '2', for the SERIAL port
 'C' ou '0' for the current serial port

Vitesse : [0x00] = original speed [0x01] = 300, [0x02] = 600, [0x03] = 1200,
 [0x04] = 2400, [0x05] = 4800, [0x06] = 9600, [0x07] = 19200,
 [0x08] = 38400, [0x09] = 57200

you must wait 0.5 second before using the serial port again

• **GET_CONFIG_AXE** Code 'X' (0x58 in hexadecimal)

Data = [Axis number (1)]
 Answer = [Axis number (1)] [Axis setting (17 byte)] [axis kind (1)]

Axis number : 0x00 for X axis
 0x01 for Y axis
 0x02 for the accessory axis

Axis kind : Axis kind :
 allways 'X' [0x58] for X or Y axis
 for the Accessory : [0x00] : No accessory, the Axis setting must be NULL
 [0x01] : Z-NUM
 [0x02] : Z-BINARY
 [0x03] : D-Axis
 [0x04] : Feeder

Axis setting : Data depend on the axis kind :
 - for X, Y, Z-NUM and D-Axes, the setting is the following :
 [Lmax(4)] [Dec(2)] [Rap(2)] [Vmin(2)] [Vmax(2)] [Vorg(2)] [Accel(1)] [Reserved(1)] [Etat(1)]
 - for Z-BINARY axis, the setting is the following :
 [res(8)] [TpsR(2)] [TpsS(2)] [Reserved(1)] [ER(1)] [ES(1)] [Sortie(1)] [Activ.(1)]
 - for the Feeder, the setting is the following:
 [Lmax(4)] [Offset(2)] [Rapport(2)] [Vmin(2)] [Vmax(2)] [Vorg(2)] [Accel(1)] [RCB(1)] [Etat(1)]

Lmax : maximum length of the axis, in step : from 0 to 4 000 000 000 (0 for the D-Axis)
Dec : Home shift in step, from -32 000 à +32 000
Rap : Ratio in step per centimeter (or step per revolution for the D-Axis) from 0 to 32 000
Vmin : Maximum speed in step per secondes, from 150 to 4000
Vmax : Minimum speed in step per secondes, from 150 to 4000
Vorg : home speed in step per secondes, from 150 to 4000
Accel : Acceleration factor, from de 0 to 255
Etat : Home position sensor value : 0x00 = Invert, 0x01 = Normal, 0x02 = No sensor
Reserved Reserved : must be NULL
Sortie Number of the exit to move the binary Axis
TpsR Time Out to reach the home position
TpsS Time Out to reach the out position
ER Number of the input where the home position sensor is connected (0 for no sensor)
ES Number of the input where the out position sensor is connected (0 for no sensor)
Activ. [0x01] for line activation, [0x00] for file activation
Offset : Evacuation position of the feeder (from the marking position) : from -32000 to 32000 step
RCB : this byte is build as follow $RCB=100*R+10*C+B$: where
R : =1 for going to home position during the cycle, and 0 to keep the feeder at Lmax :
C : =number of the input where the part sensor is connected (0 for no sensor) :
B : =number of the exit where the Part clamping is connected(0 for no clamping)

• **SET_CONFIG_IMPACT** Code 'i' (0x69 in hexadecimal)

Data = ([Pause before (2)] [Length(2)] [Pause after (2)] [Number of time (1)] [Delai (1)])*9
 or = DEFAULT
 Answer = [Return code(1)]

Pause before : Pause between end of motor move and beginning of stylus impact (in μ secondes, from 0 to 64000)
Length : length of the pulse in μ secondes, de 0 à 15000
Pause after : Pause between pulse beginning and beginning of motos move (in μ secondes, from 0 to 64000)
Number of time : How many time the impact has to be repeated.
Delai : minimum time between two impacts, de 0 à 255 in mili-second

Example :

Factory settings

```

[0x02] [0x00] [0x35] i [0x00] [72] [0x00] [250] [05] [020] [04] [126] [0x01] [02]
[0x00] [250] [07] [058] [06] [164] [0x01] [03]
[0x00] [250] [0x0A] [190] [0x0A] [040] [0x01] [05]
[0x00] [250] [0x0C] [178] [0x0C] [028] [0x01] [06]
[0x00] [250] [0x10] [154] [0x10] [004] [0x01] [08]
[0x00] [250] [0x15] [124] [0x14] [230] [0x01] [0x0B]
[0x00] [250] [33] [052] [0x1F] [158] [0x01] [0x11]
[0x00] [250] [39] [116] [38] [222] [0x01] [0x14]
[0x00] [250] [46] [224] [46] [074] [0x01] [0x18] [0x03]
    a      b      c d      e      f      g      h      i      j
    
```

Reinitialisation

```

[0x02] [0x00] [0x35] i [0x00] [0x07] DEFAULT [0x03]
    a      b c d      e      j
    
```

a = Start of Text [STX]	b = Desable check-sum
c = Protocol version	d = Command code
e = Data size	f = Pause Avant
g = Durée	h = Pause Après
i = Délai	j = End of text [ETX]

• **GET_CONFIG_IMPACT** Code 'I' (0x49 in hexadecimal)

Data = none
 Answer = ([Pause before (2)] [Lenth(2)] [Pause after (2)] [Number of time (1)] [Delai (1)])*9

Pause before : Pause between end of motor move and beginning of stylus impact (in μ secondes, from 0 to 64000)
Length : length of the pulse in μ secondes, de 0 à 15000
Pause after : Pause between pulse beginning and beginig of motos move (in μ secondes, from 0 to 64000)
Number of time : how many time the impact has to be repeated.
Delai : minimum time between two impacts, de 0 à 255 in mili-second

Example :

Getting impact definition:

```

[0x02] [0x00] [0x35] I [0x00] [0x00] [0x03]
    a      b      c d      e      j
    
```

Control answer:

```

[0x02] [0x00] [0x35] I [0x00] [72] [0x00] [250] [05] [020] [04] [126] [0x01] [02]
[0x00] [250] [07] [058] [06] [164] [0x01] [03]
[0x00] [250] [0x0A] [190] [0x0A] [040] [0x01] [05]
[0x00] [250] [0x0C] [178] [0x0C] [028] [0x01] [06]
[0x00] [250] [0x10] [154] [0x10] [004] [0x01] [08]
[0x00] [250] [0x15] [124] [0x14] [230] [0x01] [0x0B]
[0x00] [250] [33] [052] [0x1F] [158] [0x01] [0x11]
[0x00] [250] [39] [116] [38] [222] [0x01] [0x14]
[0x00] [250] [46] [224] [46] [074] [0x01] [0x18] [0x03]
    a      b      c d      e      f      g      h      i      j
    
```

a = Start of Text [STX]	b = Desable check-sum
c = Protocol version	d = Command code
e = Data size	f = Pause Avant
g = Durée	h = Pause Après
i = Délai	j = End of text [ETX]

- **SET_VAR_GLOBALE** Code '8' (0x38 in hexadecimal)

Data = [variable number (1)] [Value (0 byte to 25 bytes)]

Answer = [Return code (1)]

Variable number : number of the global variable, from [0x00] to [0x09] (or '0' to '9')

Value : value of the global variable

Example :

Setting second global variable with value : VNP

[0x02] [0x00] [0x35] 8 [0x00] [0x04] 1VNP [0x03]

a	b	c	d	e	f	g	h
a = Start of Text [STX]	b = Desable check-sum				d = Command code		
c = Protocol version	e = size				f = variable number		
g = value					h = End of text [ETX]		

- **SET_INC_GLOBALE** Code '9' (0x39 in hexadecimal)

Data = [variable number (1)] [Value (4)]

Answer = [Return code (1)]

Variable number : number of the global variable, from [0x00] to [0x09] (or '0' to '9')

Value : value of the global increment : a 32-bit integer (the most significant byte to the left)

Example :

Setting second global increment with value : 24568

[0x02] [0x00] [0x35] 9 [0x00] [0x05] [0x01] [0x00] [0x00] [95] [248] [0x03]

a	b	c	d	e	f	g	h
a = Start of Text [STX]	b = Desable check-sum				d = Command code		
c = Protocol version	e = size				f = variable number		
g = value					h = End of text [ETX]		

- **SET_SHIFT_INC_FICHER** Code '0' (0x30 in hexadecimal)

Data = [variable name (20 bytes max)] [shift number(1)] [Value (4)]

Answer = [Return code (1)]

Variable name : in upcase, cannot be longer then 20 char, cannot include spaces

Indice Equipe : number of the shift, from [0x00] to [0x09] (or '0' to '9')

Value : value of the global increment : a 32-bit integer (the most significant byte to the left)

Example :

Setting increment named INCSHIFT for the second shift with the value 111

[0x02] [0x00] [0x35] 0 [0xFF] [0xFF] INCSHIFT1 [0x00] [0x00] [0x00] [111] [0xFF] [0x03]

a	b	c	d	e	Name	f	Value	g	h
a = Start of Text [STX]	b = Desable check-sum				d = Command code				
c = Protocol version	e = Break-code mode				f = shift number				
g = Break-code					h = End of text [ETX]				

- **GET_OPTION** Code '1' (0x31 in hexadecimal) Control Option

Data = [option code (1)]

Answer = [option code (1)] [Data (n)]

See page 29 for more details

- **SET_OPTION** Code '2' (0x32 in hexadecimal) Control Option

Data = [Option code (1)] [Data (n)]

Answer = [Return code (1)]

See page 29 for more details

- **DEL_OPTION** Code '3' (0x33 in hexadecimal) Control Option

Data = [Option code (1)] [Data (n)] ou ALL to delete all options

Answer = [Return code (1)]

See page 29 for more details

- **GET_OPTION_DALLAS** Code '4' (0x34 in hexadecimal) Machine Option

Data = [Option code (1)]

Answer = [Option code (1)] [Data (n)]

See page 32 for more details

j) Control option definition :

Note : After setting or deleting option, you need to restart the control.

• **LANGUAGE Option**

Language

Code 'L' (0x4C in hexadecimal)

Data=[traduction file name (11)] [Reserved]

Total command data size=13

Traduction file name : File name of the font written with 11 bytes, padding with NULL char to the right
Reserved : char [0x00] in hexadecimal

Example :

Setting the US language

[0x02][0x00][0x35]2[0x00][0x0D]LUS[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x03]

a b c d e f g h i

a = Start of Text [STX] b = Desable check-sum
c = Protocol version d = Command code
e = size f = Option code
g = File name langue h = reserved [NULL]
i = End of text [ETX]

• **Option FICHIER**

File to open at boot time

Code 'F' (0x46 in hexadecimal)

Data=[File name (11)] [Reserved]

Total command data size =13

File name : Written with 11 bytes, padding with NULL char to the right
Reserved : char [0x00] in hexadecimal

Example :

Setting the marking file TEST

[0x02][0x00][0x35]2[0x00][0x0D]FTEST[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x03]

a b c d e f g h i

a = Start of Text [STX] b = Desable check-sum
c = Protocol version d = Command code
e = size f = Option code
g = File name h = reserved [NULL]
i = End of text [ETX]

• **Option VALUE PAR DEFAUT**

Default values editing fichiers

Code 'f' (0x66 in hexadecimal)

Data=[L(2)] [H(2)] [E(1)] [Vm(1)] [Vd(1)] [F(1)] [Qua(1)] [Zero(1)] [Kind(1)] [Name(11)] [Reserved(1)] [Reserved(1)]

Total command data size =23

W, H : Width, Height in 10th of mm : a 16-bit integer, the most significant byte to the lest
E : space between char
Vm : marking speed [0x01] à [0x09]
Vd : fast speed [0x01] à [0x09]
Qua : marking quality : [0x01] for a 5x7 grid, [0x02] for a 9x13 grid, [0x03] to [0x09] vectoriel marking
Zero : [0x01] for crossed zero, [0x00] for not crossed zero
Kind : Font kind : [129] for Font_9x13, [131] for Font_TT
Name : Font name written with 11 bytes, padding with NULL char to the right
Reserved : char [0x00] in hexadecimal

Example :

[0x02][0x00][0x35]2[0x00][0x17]

f [0x00][0x2D][0x00][0x41][0x02][0x07][0x09][0x03][0x02][0x00][129]OCRA[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x03]

f L H E Vm Vd F Q Z Kind g h h i

a = Start of Text [STX] b = Desable check-sum
c = Protocol version d = Command code
e = size f = Option code
g = File name h = reserved [NULL]
i = End of text [ETX]

- **MAINTENANCE Option** Stylus warning maintenance message setting

Code 'Z' (0x5A in hexadecimal)

Data=[Number (4)] [Warning (4)] [Stop (4)]

Total command data size =9

Number Number of impact since the last stylus has been changed : a 32-bit integer (the most significant byt to the left)
Warning Number of impact after which a warnig message is shown : a 32-bit integer (the most significant byt to the left)
Stop Number of impact after which the marking is blocked : a 32-bit integer (the most significant byt to the left)

- **GLOBALES VAR Option** setting the global variables

Code 'G' (0x47 in hexadecimal)

Data=10*([Size (1)]) 10*([Value (23)]) [Reserved (1)]]

Total command data size =251

Size the size of each variable : 10 integers 8-bits from [0x00] to [0x17]

Value value of the variable : ten strings of 23 char each (the variable are padding with NULL char to the right)

Reserved : char [0x00] in hexadecimal

Example : *setting the two first global var to value GLOB1 and GLOB2.*

```
[0x02][0x00][0x35]2[0x00][251]G[0x05][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]
GLOB1[0x00][0x00][0x00][0x00][0x00][0x00][0x00]...[0x00]
GLOB2[0x00][0x00][0x00][0x00][0x00][0x00][0x00]...[0x00]
[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]...[0x00]
[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]...[0x00][0x03]
```

- **GLOBALES INC Option** setting the global increments

Code 'I' (0x49 in hexadecimal)

Data=10*([Size (1)]) 10*([Value (4)])

Total command data size =51

Size the size of each increment: 10 integers 8-bits from [0x00] to [0x17]

Value values of the increment : 10 integers 32-bits , the most significant byte to the left

Example : *Setting the 1 and 2 increments (the rest of the 8 increments are not used)*

```
[0x02][0x00][0x35]2[0x00][251]G[0x05][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]
[0x00][0x00][0x03][0x19]
[0x00][0x01][0x05][129]
[0x00][0x00][0x00][0x00]
[0x00][0x00][0x00][0x00][0x03]
```

- **ALPHA INC BASE Option** Setting the alphanumeric increment rules

Code 'B' (0x42 in hexadecimal)

Data=[Base (from 2 to 64 bytes)]

Total command data size =3 à 65

Base base definition

Example : *setting the hexadecimal base.*

```
[0x02][0x00][0x35]2[0x00][0x11]B0123456789ABCDEF[0x03]
```

- **UNIT Option** metric or inch system

Code 'U' (0x55 in hexadecimal)

Data=[Unit (1)]

Total command data size =2

Unit [0x00] for inch system and [0x01] for metric system

k) Machine option definition:

- **auto-sensing Option** setting the Auto-sensing

Code 1 (0x01 in hexadecimal)
 Data=[Active (1)] [Stroke (1)] [Input (1)] [Mode (1)]
 Total command data size =5

Active [0x00] no autosensing, [0x01] autosensing present
Stroke from [0x01] to [0x63] : internal stroke before the part is detected (in 10th of mm)
Input from [0x01] to [0x08] : input which is connected to the autosensing sensor
Mode from [0x00] to [0x01] : value of the input when the part is detected

Example :

Setting the autosensing with a 0.6 mm stroke, the 8th input with a normaly close contact.

[0x02] [0x00] [0x35] 5 [0x00] [0x05] [0x01] [0x01] [0x06] [0x08] [0x01] [0x03]

- **Comment Option** free text

Code 11 (0x0A in hexadecimal)
 Data=[Comment (from 1 to 39 char)] [Reserved (1)]
 Total command data size =2 à 41

Comment : Free text (39 maximum chars)
Reserved : char [0x00] in hexadecimal

Example :

Setting comment « Last revision mai 03 ».

[0x02] [0x00] [0x35] 5 [0x00] [0x16] [0x0B] Last revision mai 03 [0x00] [0x03]

- **Scratching Option** Setting the scribbling machine

Code 12 (0x0B in hexadecimal)
 Data=[Up (1)] [Down (1)] [Action (1)] [Sise (1)]
 Total command data size =5

Up time for the stylus to move up (a 8-bit integer in ms)
Down time for the stylus to move down (a 8-bit integer in ms)
Action [0x01] = stylus command (newer machines), [0x00] = output 4 (older machines)
Size simple segment size : [0x02]=1.67 mm, [0x03]=1 mm, [0x04]=0.58 mm,... , [0x09]=0.05 mm

Example :

Setting default value for scratching machine

[0x02] [0x00] [0x35] 5 [0x00] [0x05] [0x0C] [0x14] [0x1D] [0x01] [0x07] [0x03]

- **Portable Option** for Portable machine

Code 13 (0x0C in hexadecimal)
 Data=[Time (1)]
 Total command data size =2

Time delay for stopping the marking in 10th of second : a 8-bit integer.

Example :

For the p60

[0x02] [0x00] [0x35] 5 [0x00] [0x02] [0x0D] [0x14] [0x03]

- **Fast move option** Setting the fast move / marking move speed

Code 14 (0x0D in hexadecimal)
 Data=[Step (2)] [Tempo (2)]
 Total command data size =5

Step Number of steps from which an axis runs at fast speed, below that number, the axis runs at marking speed
Tempo delay the motor has to wait after having run at fast speed (in mili-second)
 Thoses two numbers are 16-bit integer writing the most significant byte to the left

Example :

Setting the limit to 5120 step and a delay of 100 mili-second

[0x02] [0x00] [0x35] 5 [0x00] [0x05] [0x0E] [0x14] [00] [0x00] [100] [0x03]

- **Motor intensity Option** setting the motor intensity

Code 15 (0x0E in hexadecimal)

Data=[MaxX(1)][MaxZ(1)][SleepX(1)][SleepZ(1)]

Total command data size =5

MaxX Intensity for the X and Y motor when moving from [0x00] à [0x04]
MaxZ Intensity for the Z motor when moving from [0x00] à [0x04]
SleepX Intensity for the X and Y motor when waiting at home position from [0x00] à [0x04]
SleepZ Intensity for the Z motor when waiting at home position from [0x00] à [0x04]

for XY axis : [0x00] = 0A, [0x01]=0.50A, [0x02]=0.67A, [0x03]=1.0A et [0x04]=2 Amperes
for the Z axis : [0x00] = 0A, [0x01]=0.75A, [0x02]=1A, [0x03]=1.5A et [0x04]=3 Amperes

Example :

Default setting

[0x02][0x00][0x35]5[0x00][0x05][0x0F][0x04][0x04][0x02][0x02][0x03]

- **Input-Output Option** Setting the inputs and outputs functions

Code 16 (0x0F in hexadecimal)

Data=[Inputs(8)][Outputs(8)]

Total command data size =17

Inputs values of the 8 inputs
Outputs values of the 8 outputs

Value for each input

[0x00] :	Input not used	[0x01] :	start marking
[0x02] :	Use for file selection	[0x03] :	pause validation
[0x03] :	Air pressure control		

it is not possible to change input 1 and 2, be sure to always keep [0x00] sans quoi le système ne risque de ne pas fonctionner

Value for each output

[0x00] :	Output not used	[0x01] :	Error
[0x02] :	last dot marked	[0x03] :	in marking
[0x04] :	in Pause	[0x05] :	ready to run
[0x06] :	auto-sensing : error Part	[0x07] :	auto-sensing : Error no part
[0x08] :	stylus need to be change		

Example :

Factory setting

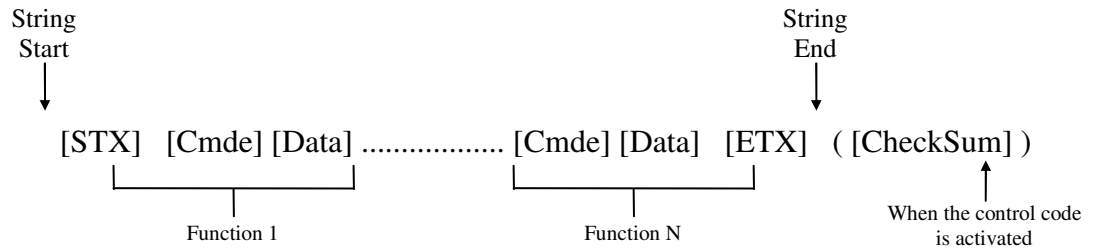
[0x02][0x00][0x35]5[0x00][0x11][0x10][0x00][0x00][0x02][0x02][0x02][0x02][0x02][0x02][0x02][0x01][0x02][0x03][0x04][0x00][0x00][0x00][0x00][0x03]

4 - Compatibility with the communication protocol of the 4A version program

a) Communication protocol:

The string sent to the controller must start with the characters **STX** (Start TeXte: 02h), followed by a list of functions (described on the next pages) and the string must end with **ETX** (End TeXte: 03h).

b) String to be sent :



Control code CheckSum : In order to detect a possible error in the transmission, the CheckSum is calculated depending on the string sent by the main system and receptioned by the machine. If the string has been correctly transmitted, the code calculated by the marking machine is the same as the code sent by the main system.

The **CheckSum** corresponds to an "EXCLUSIVE OR" of all codes transmitted in the string, including the STX code and the ETX code.

c) List of functions :

Each function starts with a control code, followed by the data corresponding to the function to be used :

Code	Description											
[NUL]	Deactivate the control code CheckSum [No data]											
[SOH]	File Selection [Data: Name of the file to be loaded (max 11 characters)]											
[ACK]	Marking release right after receiving the string without waiting for the confirmation through the <i>Start Cycle</i> button. [No data]											
[ENQ]	Definition of a variable in the valid file or after loading a file through the [SOH] function [Data: name of the variable + '=' + value to be attributed]											
[LF]	Definition of the marking speed (standard is speed 5) [Data : 0 to 9]											
[VT]	Transfer all marking parameters Data: Each line of the file is transferred with the following format: <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td style="width: 10%;">X</td> <td style="width: 10%;">Y</td> <td style="width: 10%;">Z</td> <td style="width: 10%;">W</td> <td style="width: 10%;">H</td> <td style="width: 10%;">Ang.</td> <td style="width: 10%;">Radius</td> <td style="width: 10%;">D</td> <td style="width: 10%;">O</td> <td style="width: 10%;">F</td> <td style="width: 40%;">Text on N characters</td> </tr> </table> <p> <i>X, Y :</i> Coordinates of the text to be marked (-9999 to +9999 in tenth of mm), <i>Z :</i> Coordinates of the text to be marked (0000 to 9999 in tenth of mm), <i>W, H :</i> Width and height of the character to be marked(000 to 999 in tenth of mm), <i>Ang :</i> marking angle (000 to 359 in degrees), <i>Radiu :</i> Radius of the circle for circular markings (0000 to 9999 in tenth of mm), <i>D :</i> Depth - Impact force (0=no impact to 9=large impact), <i>O :</i> Orientation - Marking direction ('N' or 'T' for Normal or Reverse orientation), <i>F :</i> Character font (1 to 9), <i>Text :</i> Text to be marked </p>	X	Y	Z	W	H	Ang.	Radius	D	O	F	Text on N characters
X	Y	Z	W	H	Ang.	Radius	D	O	F	Text on N characters		
[FF]	Transfer of the part diameter, when a D-axis is used. [Data: 0000 to 2500, in tenth of mm]											
[CR]	Selection of the spacing between characters [Data: 0 to 9]											
[SO]	Selection of the character font. [Data: 0 (OCR) or 1 (OCR-A)]											
[SI]	Selection of the marking mode in mirror [Data: 0 (not in mirror mode), 1 (horizontal mirror model) or 2 (vertical mirror mode)]											
[DLE]	Saving of the actual parameters in a file. [Data: name of the file to be saved (max 11 characters)]											

d) *Examples de communication :*

- Select the 'AB12' file, without CheckSum and without launching the marking :
Send the string :
[STX] [NUL] [SOH] AB12 [ETX]
or in hexadecimal:
02 00 01 41 42 31 32 03
- Select the 'AB12' file, with CheckSum and launching the marking :
Send the string :
[STX] [SOH] AB12 [ACK] [ETX] K
or in hexadecimal:
02 01 41 42 31 32 06 03 4B(checksum)
- Transfer of a complete string with marking launching and selection of speed 9:
Send the string :
[STX] [NUL] [LF] 9 [VT] +0100+0500000002002000000005N1ABCdef [VT]
+0100+1000000003003000000005N2Z123 [ACK] [ETX]
or in hexadecimal:
02 00 0A 39 0B 2B 30 31 30 30 2B 30 35 30 30 30 30 30 30 32 30 30 32 30 30 30 30 30
30 30 30 35 4E 31 41 42 43 64 65 66 0B 2B 30 31 30 30 2B 31 30 30 30 30 30 30 30 33
30 30 33 30 30 30 30 30 30 30 35 4E 32 5A 31 32 33 06 03
- Save the parameters in a file called "XJK" :
Send the string :
[STX] [NUL] [DLE] XJK [ETX]
or in hexadecimal:
02 00 10 58 4A 4B 03
- Transfer of a complete string in OCR-A font, spacing 5 between characters and parameters saving in a "XJK"
file :
Send the string :
[STX] [NUL] [SO] 1 [CR] 5 [VT] +0100+0500000002002000000005N1ABCdef
[VT] +0100+1000000003003000000005N2Z123 [DLE] XJK [ETX]
or in hexadecimal:
02 00 0E 31 0D 35 0A 39 0B 2B 30 31 30 30 2B 30 35 30 30 30 30 30 30 30 32 30 30 32 30
30 30 30 30 30 30 30 35 4E 31 41 42 43 64 65 66 0B 2B 30 31 30 30 2B 31 30 30 30 30 30
30 30 33 30 30 33 30 30 30 30 30 30 30 35 4E 32 5A 31 32 33 10 58 4A 4B 03
- Marking of a logo called "ZX3" in X=10, Y=20:
Send the string :
[STX] [NUL] [VT] +0100+0200000010010000000005N1[EOT]Logo ZX3 [ACK] [ETX]
or in hexadecimal:
02 00 0B 2B 30 31 30 30 2B 30 32 30 30 30 30 30 30 31 30 30 31 30 30 30 30 30 30 30
30 35 4E 31 04 4C 6F 67 6F 20 5A 58 33 06 03
- Datamatrix ECC200 string
ECC200([Speed] [Format] [Reference], [text])
Speed : one digit from '0' to '9'
Reference = 'S' for Simple or 'D' for Double
Format : one character depending on the following:
AutoSquare:0 - AutoRect:1 - 10x10:A - 12x12: B - 14x14: C - 16x16: D - 18x18: E - 20x20: F -
22x22: G- 24x24: H - 26x26: I - 8x18: J - 8x32: K - 12x26: L - 12x36: M - 16x36: N - 16x48:O
Example with marking speed 3, AutoSquare, simple reference and including the text 12345:
Send the string:
+0100+0200000010010000000005N1[EOT]ECC200(30S,12345)
In hexadecimal:
2B 30 31 30 30 2B 30 32 30 30 30 30 30 30 31 30 30 31 30 30 30 30 30 30 30 30 30 35 4E
31 04 45 43 43 32 30 30 28 33 30 53 2C 31 32 33 34 35 29
- Pause string:
Send the following string:
[VT] +0100+0200000010010000000005N1[EOT]Pause
In hexadecimal:
2B 30 31 30 30 2B 30 32 30 30 30 30 30 30 31 30 30 31 30 30 30 30 30 30 30 30 30 35 4E
31 04 50 61 75 73 65

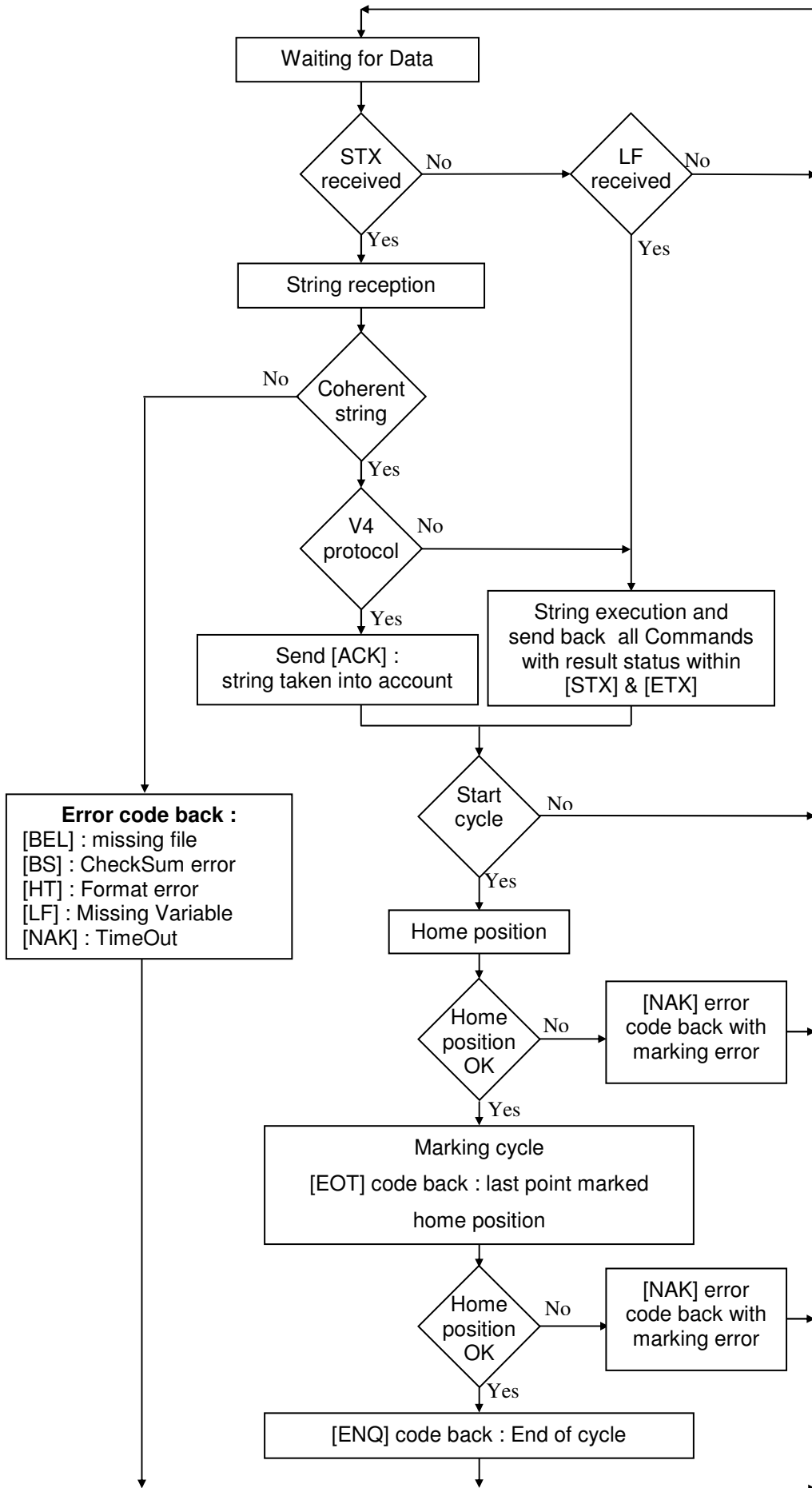
Remark : "Logo " "Pause" and"ECC200" are reserved words ; you absolutely must respect the character case (upper/lower) as well as the space character between "Logo" and the name of the logo file to be marked.

ASCII Codes:

[NUL] = 0x00	[SOH] = 0x01	[STX] = 0x02	[ETX] = 0x03
[EOT] = 0x04	[ENQ] = 0x05	[ACK] = 0x06	[BEL] = 0x07
[BS] = 0x08	[HT] = 0x09	[LF] = 0x0A	[VT] = 0x0B
[FF] = 0x0C	[CR] = 0x0D	[SO] = 0x0E	[SI] = 0x0F
[DLE] = 0x10	[DC1] = 0x11	[DC2] = 0x12	[DC3] = 0x13
[DC4] = 0x14	[NAK] = 0x15		

Code Dec	Code Hex	Val	Code Dec	Code Hex	Val	Code Dec	Code Hex	Val	Code Dec	Code Hex	Val	Code Dec	Code Hex	Val	Code Dec	Code Hex	Val	Code Dec	Code Hex	Val
32	20		64	40	@	96	60	`							192	C0	←			
33	21	!	65	41	A	97	61	a	129	81	ü				193	C1	↑	225	E1	Б
34	22	"	66	42	B	98	62	b							194	C2	↓			
35	23	#	67	43	C	99	63	c							195	C3	↕			
36	24	\$	68	44	D	100	64	d	132	84	ä				196	C4	↕			
37	25	%	69	45	E	101	65	e							197	C5	ч			
38	26	&	70	46	F	102	66	f	134	86	å	166	A6	б						
39	27	'	71	47	G	103	67	g				167	A7	в						
40	28	(72	48	H	104	68	h							200	C8	ш			
41	29)	73	49	I	105	69	i							201	C9	щ			
42	2A	*	74	4A	J	106	6A	j				170	AA	г	202	CA	ь			
43	2B	+	75	4B	K	107	6B	k				171	AB	д	203	CB	ы			
44	2C	,	76	4C	L	108	6C	l				172	AC	ж	204	CC	э			
45	2D	-	77	4D	M	109	6D	m							205	CD	ю			
46	2E	.	78	4E	N	110	6E	n	142	8E	Ä	174	AE	з	206	CE	я	238	EE	л
47	2F	/	79	4F	O	111	6F	o	143	8F	Å	175	AF	и	207	CF	Б	239	EF	ю
48	30	0	80	50	P	112	70	p				176	B0	й	208	D0	æ	240	F0	п
49	31	1	81	51	Q	113	71	q				177	B1	к	209	D1	Æ	241	F1	ф
50	32	2	82	52	R	114	72	r				178	B2	л				242	F2	ц
51	33	3	83	53	S	115	73	s				179	B3	м	211	D3	Ё	243	F3	у
52	34	4	84	54	T	116	74	t	148	94	ö	180	B4	н				244	F4	ш
53	35	5	85	55	U	117	75	u							213	D5	Г	245	F5	щ
54	36	6	86	56	V	118	76	v										246	F6	ь
55	37	7	87	57	W	119	77	w										247	F7	Ы
56	38	8	88	58	X	120	78	x										248	F8	о
57	39	9	89	59	Y	121	79	y	153	99	Ö	185	B9	п	217	D9	Д	249	F9	Ъ
58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü	186	BA	т	218	DA	Ж	250	FA	Э
59	3B	;	91	5B	[123	7B	{	155	9B	ø	187	BB	ф	219	DB	З	251	FB	¹
60	3C	<	92	5C	\	124	7C					188	BC	ц	220	DC	И	252	FC	³
61	3D	=	93	5D]	125	7D	}	157	9D	ø				221	DD	Й	253	FD	²
62	3E	>	94	5E	^	126	7E	~				190	BE	у				254	FC	Я
63	3F	?	95	5F	_							191	BF	→	223	DF	К			

Codes sent back from the system



Error codes returning on a marking error (fig 1)

Decimal Code (3 bytes)	Hexa Code (3 bytes)	Binary value	description
00 00 01	00 00 01	0000 0000 0000 0000 0000 0001	Error with marking font
00 00 02	00 00 02	0000 0000 0000 0000 0000 0010	Error with dot logo
00 00 04	00 00 04	0000 0000 0000 0000 0000 0100	Error with vectorial logo
00 00 08	00 00 08	0000 0000 0000 0000 0000 1000	Error with Ecc200
00 00 16	00 00 10	0000 0000 0000 0000 0001 0000	Error with the syntax of text zone
00 00 32	00 00 20	0000 0000 0000 0000 0010 0000	Error with variable
00 00 64	00 00 40	0000 0000 0000 0000 0100 0000	Error with I/O
00 00 128	00 00 80	0000 0000 0000 0000 1000 0000	Error with RS232
00 01 00	00 01 00	0000 0000 0000 0001 0000 0000	Error : Stop button activated
00 02 00	00 02 00	0000 0000 0000 0010 0000 0000	Error with stylus
00 04 00	00 04 00	0000 0000 0000 0100 0000 0000	Error with motor
00 08 00	00 08 00	0000 0000 0000 1000 0000 0000	Error with sensor
00 16 00	00 10 00	0000 0000 0001 0000 0000 0000	Error out of marking window bound
00 32 00	00 20 00	0000 0000 0010 0000 0000 0000	Error with the X axis
00 64 00	00 40 00	0000 0000 0100 0000 0000 0000	Error with the Y axis
00 128 00	00 80 00	0000 0000 1000 0000 0000 0000	Error with the accessory axis
01 00 00	01 00 00	0000 0001 0000 0000 0000 0000	- Error blocked feeder Or - Error with Autosensing : no part detection
02 00 00	02 00 00	0000 0010 0000 0000 0000 0000	- Error empty feeder Or - Error with Autosensing : part detected out of bound Or - Error with binary axis
04 00 00	04 00 00	0000 0100 0000 0000 0000 0000	The marking head did loose steps
08 00 00	08 00 00	0000 1000 0000 0000 0000 0000	Error with external motor
16 00 00	10 00 00	0001 0000 0000 0000 0000 0000	Historique full
32 00 00	20 00 00	0010 0000 0000 0000 0000 0000	Double detected for historique
64 00 00	40 00 00	0100 0000 0000 0000 0000 0000	Error : stylus need to be changed
128 00 00	80 00 00	1000 0000 0000 0000 0000 0000	Error : stylus as to be changed

Examples :

00 01 00 : Emergency stop activated

00 30 00 (= 00 10 00 + 00 20 00) : Marking is out of bounds for X axie

00 48 00 (= 00 08 00 + 00 40 00) : Error with origin sensing on Y