COMMUNICATION PROTOCOL

e10-V6

www.sic-marking.com

CONTENTS

I - COMMUNICATION PROTOCOL 6

1 - GENERAL INFORMATION 6

2 - TEXT PROTOCOL 7

a) String to be sent: 7

b) Response of the control 7

c) Commun fuctions 7

- **LOADFILE load a marking file 7**
- **SETVAR set a variable or an increment to the current file 7**
- **RUN start the marking 7**
- **RESETERROR if an error occurse, you need to clear the machine status. 8**

d) managing files functions 8

- **NEWFILE create a new marking file 8**
- **SETFILEOPTION set diameter for D-AXIS system 9**
- **INSERTTEXTLINE insert a line to the current file 9**
- **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 10**
- **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 11**
- **SETLINEOPTION Set options for the last inserted line of the current file 11**
- **SAVEFILE save current file 12**
- **FILEDELETE erase a file into the control 12**

e) manual programming functions 12

- **HOMEPOSITION go to home position 12**
- **IMPACT Impact or srcatching 13**

f) settings functions 13

- **SETSHIFTINCVAR setting settting a shift increment to the current file 13**
- **SETGLOBALVAR Affectation variable globale 13**
- **SETGLOBALINC Affectation incrément globale 13**
- **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 : 16

- **LOAD FILE Code 'c' (0x63 in hexadecimal) 16**
- **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 17

- **NEW_FILE Code 'f' (0x66 in hexadecimal) 17**
- **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) 19**
- **FILE LIST Code 'L' (0x4C in hexadecimal) 19**
- **PC_TO_E6 Code 'G' (0x47 in hexadecimal) 20**
- **E6_TO_PC Code 'S' (0x53 in hexadecimal) 21**
- **DEL_FILE Code 'D' (0x44 in hexadecimal) 21**

f) File options definition: 22

- **Diameter Option Setting the part diameter (for the D-Axis use only) 22**
- **Comment Option Setting the file comment (free text) 22**

g) Last marking file line option description : 22

- **Attribut option Pritable setting 22**
- **Indexor Option D-axis setting 22**

h) Manual programming 23

- **ORIGINE Code 'H' (0x48 in hexadecimal) 23**
- **IMPACT Code 'P' (0x50 in hexadecimal) 23**
- **SET_OUPUT Code 'Z' (0x5A in hexadecimal) 23**

- **GET_INPUTS Code 'Y' (0x59 in hexadecimal)** ... 23

i) Settings functions ... 23

- **RESTART Code '*' (0x2A in hexadecimal)** ... 23
- **SYNCHRO_DATEHEURE Code 'h' (0x68 in hexadecimal)** ... 24
- **VITESSE_COM Code 'v' (0x76 in hexadecimal)** ... 24
- **SET_VAR_GLOBALE Code '8' (0x38 in hexadecimal)** ... 24
- **SET_INC_GLOBALE Code '9' (0x39 in hexadecimal)** ... 24
- **SET_SHIFT_INC_FICHIER Code '0' (0x30 in hexadecimal)** ... 25
- **GET_OPTION Code '1' (0x31 in hexadecimal) Control Option** ... 25
- **SET_OPTION Code '2' (0x32 in hexadecimal) Control Option** ... 25
- **DEL_OPTION Code '3' (0x33 in hexadecimal) Control Option** ... 25
- **RELOAD_CONFIG Code 128 (0x80 in hexadecimal) Reload machine configuration** ... 25
- **GET_MACHINE Code 129 (0x81 in hexadecimal)** ... 25

j) Examples of Machine configuration ... 25

- **Set a p62 system** ... 25
- **Set a c151 system with a standard 50mm Znum axie** ... 26
- **Set a i111s system with lost step control and no 3rd axis** ... 26

k) Control option definition: ... 26

- **LANGUAGE Option Language** ... 26
- **Option FICHIER File to open at boot time** ... 26
- **Option VALUE PAR DEFAUT Default values editing fichiers** ... 26
- **KEYBOARD Option Keyboard setting** ... 27
- **STATISTICS Option Controls Statistics** ... 27
- **MACHINE_NAME Option machine name (C150, I80, …) caution : letters must be in upper case** ... 27
- **MACHINE_ACCESSORY Option 3rd axie configuration** ... 28
- **MACHINE_ADVANCED Option advanced configuration of the system (IO config, axies priority, …)** 28
- **SHIFT Option Shift setting** ... 29
- **GLOBALES VAR Option setting the global variables** ... 29
- **GLOBALES INC Option setting the global increments** ... 29
- **ALPHA INC BASE Option Setting the alphanumeric increment rules** ... 29
- **UNIT Option metric or inch system** ... 30

4 - COMPATIBILITY WITH THE COMMUNICATION PROTOCOL OF THE 4A VERSION PROGRAM ... 30

a) Communication protocol: ... 30

b) String to be sent: ... 30

c) List of functions: ... 30

d) Examples de communication : ... 31

II - SLAVE MODE – COMMUNICATION PROTOCOL – INPUT / OUTPUT ... 35

1 - SERIAL COMMUNICATION ... 35

a) TEXT format of the communication protocol ... 35

- **String format to be sent:** ... 35
- **Response of the system** ... 35
- **Commun functions** ... 35
- **Examples** ... 35

b) V4 protocol ... 36

- **Communication protocol** ... 36
- **String to be sent :** ... 36
- **List of functions** ... 36
- **Examples** ... 36

c) ASCII Codes: ... 37

d) Code sent back from the system (protocol v4 and v5) ... 38

e) Error codes returning on a marking error ... 39

f) Description of the serial ports ... 40

- **"HOST"** ... 40
- **"SERIAL"** ... 40

g) RS232 test ... 41

2 - TOR INPUTS / OUTPUTS ... 41

a) Cableling ... 41

b) Electric limit ... 42

- **Dry contact output (relays)** ... 42
- **Output 4 (transistor)** ... 42
- **Inputs** ... 42

c) File selection ... 42

d) Input/Output test ... 43

e) Operating timing diagram ... 43

- **Wiring of the pneumatic Z-axis** ... 45

• **Use of the I/O with the clamping of a part being marked, and defaut announced by a light. ... 45**

III - CONFIGURATION OF THE COMMUNICATION PORT (OPTIONAL BOARDS): ... 47

Important note: The use of an optional communication boards on the e10 controller disable the RS232 port. ... 47
a) Ethernet ... 47
b) Profibus ... 48
c) Profinet and Ethernet/IP ... 48

IV - FIELDBUS PROTOCOLS ... 50

a) Prerequisite ... 50
b) Integration ... 50
c) Procedure required ... 51
d) Example 1 ... 51
e) Example 2 ... 51
f) Example 3, ... 52

I - COMMUNICATION PROTOCOL

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 possibles:

- 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 40 000 bytes
- Buffer memory is 500 bytes: do not send more than 500 byte too fast else data will be lost.

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)

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*

Syntaxe :

```
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 (optonnal)
Marking at force 0 if "Simulation", else marking at normal force
Err*i* 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.*

Syntaxe :

```
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 :

Dismater the part diameter in 10th 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*=10*mm, Y*=12*mm,* 5x7*mm 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*

Syntaxe :

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*

Syntaxe :

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 streatch the logo within W and H
Force	From 0 to 9
Quality	From 1 to 9
Type	V for Vectorial [2]logo, D for Dot marking logo
Logo	Name of the logo

Example : *Vectorial logo named* `SIC` *at X*=`10`*mm, Y*=`12`*mm,* `10x10`*mm 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]
```

- ***INSERTECC200LINE*** *insert a Data matrix to the current file*

Syntaxe :

```
INSERTECC200LINE <X> <Y> <Z> <W> <H> <Angle> <Format> <Force> <Ref> <Speed> <Text> [CR][LF]
```

Answer :

```
INSERTECC200LINE OK[CR][LF]                  → OK
INSERTECC200LINE 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*=`15`*mm, Y*=`14`*mm,* `10x10`*mm chars at force* `5`, *Simple reference, speed=3, text to encode*=`HELLO WORLD`

String to send :

```
INSERTECC200LINE 150 140 0 100 100 0 0 5 0 3 HELLO WORLD[LF]
```

String received:

```
INSERTECC200LINE 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 (fom 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> <Anle> <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 : ɑ 2 : ɔqɐ 3 : ɔ
Mirror	0 : Abc 1 : ʎqɔ 2 : ɔdA
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]
```

e) 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.

- ***HOMEPOSITION*** *go to home position*

Syntaxe :
```
HOMEPOSITION <Motors>
Answer :
HOMEPOSITION OK[CR][LF]
HOMEPOSITION ERROR <Err1> <Err2> <Err3>[CR][LF]
```

Data :
Motors (optionnal)
1 : X motor only 2 : Y motor only
3 : X et Y motors 4 : Z motor only
7 : X, Y and Z motors
<Err*i*> if an error occurse, see fig 1 on page 6

Example : *go to home position for XY et Z*
String to send :
```
HOMEPOSITION[CR][LF]
```
String received: *(error : the Z axis is not at home position)*
```
HOMEPOSITION ERROR 0 136 0[CR][LF]
```
→*error is 136 = 128 + 8* = Error Z axis + Error Sensor

After resolving the probleme,clear the error and go back to home position for Z
String to send :
```
RESETERROR[CR][LF]HOMEPOSITION 4[CR][LF]
```
String received:
```
RESETERROR OK[CR][LF]
HOMEPOSITION OK[CR][LF]
```

- ***IMPACT*** *Impact or srcatching*

Syntaxe :
`IMPACT <Speed> <X> <Y> <Z> <Force> [CR][LF]`

Answer :
`IMPACT OK[CR][LF]` → *OK*
`IMPACT ERROR <Err1> <Err2> <Err3>[CR][LF]` → *Error*

Data :

Speed	marking speed : de '1' à '9'	
X, Y, Z	Coordonate from home position de l'impact en Pas	
Force	Force of impact : from '0' to'9'	(dot marking machine)
	Or stylus position : '0' for up, and '1' for down	(scratching machine)
Err*i*	if an error occurse, see fig 1 on page 6	

Example :
String to send :
`IMPACT 5 500 500 0 6[LF]`
String received:
`IMPACT OK[CR][LF]` → *OK*

String to send :
`IMPACT 5 500 -500 0 6[LF]`
String received:
`IMPACT ERROR 0 80 0[CR][LF]` → *error : 80 = 64 + 16* = Y axis error + out of bound error

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 settting 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 necesary

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 necesary

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 fuctions :

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_OPTION	'1'	\
SET_OPTION	'2'	} control option management
DEL_OPTION	'3'	/
RELOAD_CONFIG	128	\
GET_MACHINE	129	} Machine management
GET_CONFIG_MACHINE	130	/

- Examples of Machine configuration:

- set a P62 system with no options and and no 3rd axie.
- set a C151 system with no options and a ZNUM axie.
- set a I111s system with lost step control and no 3rd axie.

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

```
[0x02][0x00][0x35][0x63][0x00][0x04]TEST[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
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 afphanumeric variable named OF *with value* 524VNP

```
[0x02][0x00][0x35][0x37][0xFF][0x00]OF=524VNP[0x00][0x03]
  a     b     c     d        e       f   g      h     i
```

Setting an incrementation named SERIAL_NUM *with value* 24568

```
[0x02][0x00][0x35][0x37][0x00][0x0F]SERIAL_NUM=[0x00][0x00][0x5F][0xF8][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 = 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 imact, stylus always up)
if marking mode is empty, then use of normal marking status.
Machine status = see fig 1 page 6

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 'I'* *(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 SERAIL 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 mili-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 mili-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 / - or the Start button to be pressed } , to continue the marking,

`[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 22)

`[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 :

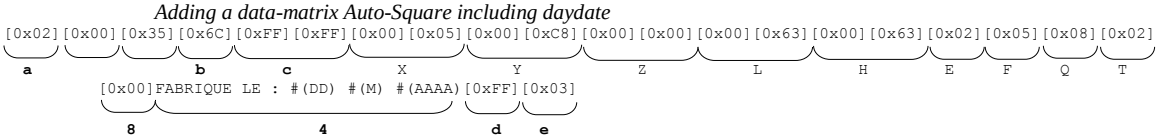
Adding the text : SIC-MARKING

`[0x02][0x00][0x35][0x6C][0xFF][0xFF][0x00][0x32][0x00][0x50][0x00][0x00][0x00][0x2D][0x00][0x41][0x02][0x05][0x02][0x00]`
a **b** **c** X Y Z L H E F Q T
`[0x81]OCR[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]SIC-MARKING[0xFF][0x03]`
1 **2** **3** **4** **d** **e**

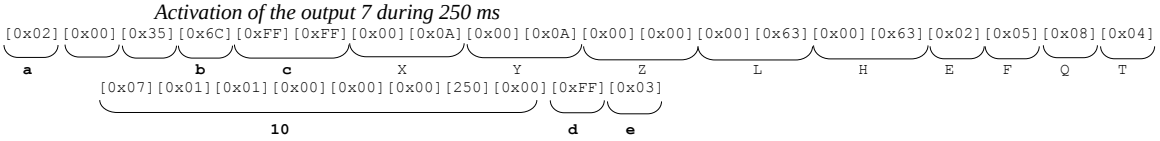
Adding the vectorial logo SIC

`[0x02][0x00][0x35][0x6C][0xFF][0xFF][0x00][0x05][0x00][0xC8][0x00][0x00][0x00][0x63][0x00][0x63][0x02][0x05][0x08][0x01]`
a **b** **c** X Y Z L H E F Q T
`[0x05][0x01][0x00]SIC[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0xFF][0x03]`
5 **6** **3** **7** **3** **d** **e**

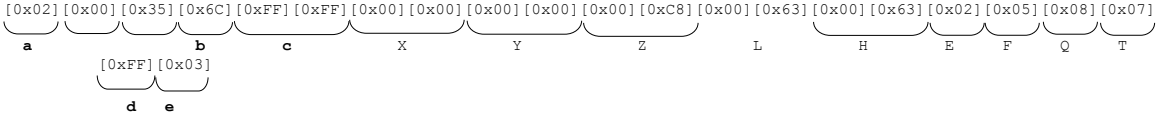
Adding a data-matrix Auto-Square including daydate



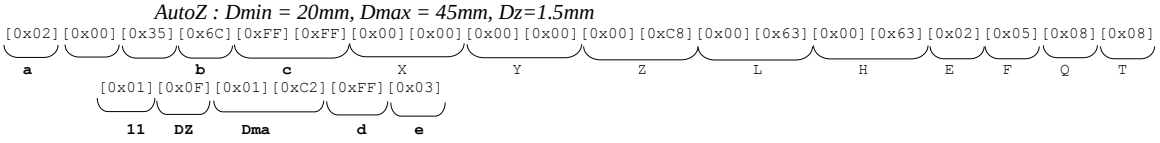
Activation of the output 7 during 250 ms



MoveZ de 20mm



AutoZ : Dmin = 20mm, Dmax = 45mm, Dz=1.5mm



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

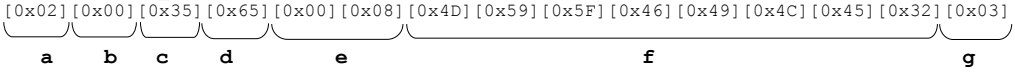
- *LINE_SET_OPTION* *Code 'a'* *(0x61 in hexadecimal) last line of marking file Option*

Data = [Option code (1)] [Data (n)]
Answer = [Return code (1)]
See page 22 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



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 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`)*

```
[0x02][0x00][0x35][0x4C][0x00][0x0D][0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x03]
  a     b     c     d        e                                                                    f                 g     h     i
```

Answer string:

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

```
[0x02][0x00][0x35][0x4C][0x00][0x26]
  a     b     c     d        e
[0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x02][0x00][0x4D][0x00][0x00][0x00][0x00]
[0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x04][0x02][0x08][0x00][0x00][0x00][0x00][0x03]
                                                                                                                                  f                                                                                                                                  g     h     j                 g                 i
```

- When there is no file `TEST`

```
[0x02][0x00][0x35][0x4C][0x00][0x00][0x03]
  a     b     c     d        e        i
```

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 :

```
[0x02][0x00][0x35][0x47][0x02][0x22]
  a     b     c     d        e
[0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x02][ File contents (533)][0x03]
                                                                                                                                  f                                                                                                                                  g     h                 j                 i
```

a = Start of Text [STX] **b** = Desable check-sum
c = Protocol version **d** = Command code
e = Data size (13 for file name + 533 for contents file = 546 with is 0x02 0x22 in bytes) **f** = File name
g = reserved **h** = File kind
i = End of text [ETX] **j** = File contents

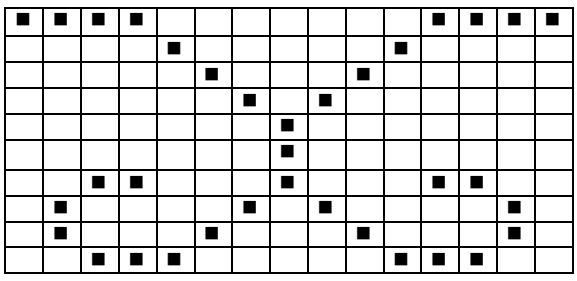
Contents of a Dot logo file :

[Size X][Size Y][X1][Y1][X2][Y2]….[X*n*][Y*n*]

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

X*i*, Y*i* : coorData 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	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 :

`[0x02][0x00][0x35][0x53][0x00][0x0D]`
a b c d e

`[0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x02][0x03]`
f g h i

Answer string:
- When the `TEST` file has been found

`[0x02][0x00][0x35][0x53][0x02][0x22]`
a b c d e

`[0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x02][File contents(533)][0x03]`
f g h j i

- *When there is not* `TEST` *file*

`[0x02][0x00][0x35][0x53][0x00][0x00][0x01][0x03]`
a b c d e

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 = Scribling logo file

Example :

`[0x02][0x00][0x35][0x44][0x00][0x0D]`
a b c d e

`[0x54][0x45][0x53][0x54][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x02][0x03]`
f g h i

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]o[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]o[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]l[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]l[0x00][0x02][0x02][0x01][0x03]
```

h) Manual programming

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

Data = [Motors list (1)] *(optional)*
Answer = [machine status (3)]
Motors list = binary mix of theses values : 0x01 for X axis, 0x02 for Y axis and 0x04 for accessory axis
Default value is 0x07
Machine status = see fig 1 page 6

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

home positon for Y axis
[0x02][0x00][0x35]H[0x00][0x01][0x02][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 (1 or 0)
f = Moteur Y
g = End of text [ETX]

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

Data = [Speed(1)] ([X_1(4)] [Y_1(4)] [Z_1(4)] [F_1(1)]) ([X_2(4)] [Y_2(4)] [Z_2(4)] [F_2(1)]) …
Answer = [machine status (3)]
Vitesse : marking speed from [0x01] to [0x09]
X*i* : X coordonate of impact number *i* (a 32 bit signed integer, the most significant byte to the lest) in step
Y*i* : Y coordonate of impact number *i* (a 32 bit signed integer, the most significant byte to the lest) in step
Z*i* : Z coordonate of impact number *i* (a 32 bit signed integer, the most significant byte to the lest) in step
F*i* : - 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 6
Example :
[0x02][0x00][0x35]P[0x00][0x1B][0x08][0x00][0x00][0x02][0x00][0x00][0x00][0x02][0x00][0x00][0x00][0x00][0x00][0x00][0x05]
[0x00][0x00][0x01][0x00][0x00][0x00][0x01][0x00][0x00][0x00][0x00][0x00][0x00][0x05][0x03]
a b c d e f g h i j k
a = Start of Text [STX]
b = Desable check-sum
c = Protocol version
d = Command code
e = Data size (1 + 2*13)
f = Speed 8
g = X
h = Y
i = Z
j = Depth
k = End of text [ETX]

- *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 begining 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

- *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
c = Protocol version **d** = Command code
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
c = Protocol version **d** = Command code
e = size **f** = variable number
g = value **h** = End of text [ETX]

- *SET_SHIFT_INC_FICHIER* *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
c = Protocol version
d = Command code
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 26 for more details

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

Data = [Option code (1)] [Data (n)]
Answer = [Return code (1)]
See page 26 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 26 for more details

- *RELOAD_CONFIG* *Code 128* *(0x80 in hexadecimal) Reload machine configuration*

This function is used to reset the machine configuration.
The machine configuration is base on these OPTIONS (see SET_OPTION chapiter)
→ MACHINE_NAME Option
→ MACHINE_ACCESSORY Option
→ MACHINE_ADVANCE Option
Data = none
Answer = [Return code (1)]
Example :

```
[0x02][0x00][0x35][0x80][0x00][0x00][0x03]
```

- *GET_MACHINE* *Code 129* *(0x81 in hexadecimal)*

This function is used to get the current machine name and size.
Data = none
Answer =

```
MachineName[11] Reserved[1]
SizeX[4] SizeY[4] SiezZ[4]
Kind3rdAxie[1] Scraching[1] AutoSensing[1] Reserved[1]
full machine name[15]  Reserved[1]
machine serial number[4]
```

Example :

```
Send :   [0x02][0x00][0x35][0c81][0x00][0x00][0x03]

Received : [0x02][0c81][0x00][0x30] C151[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]
           [0x00][0x00][0x06][0x40] [0x00][0x00][0x03][0xE8] [0x00][0x00][0x01][0xF4]
           [0x01][0x00][0x00][0xFC]
           c151 (rev A)[0x00][0x00][0x00][0x00][0x00]
           [0x06][0x2B][0x9A][0x61][0x03]
```

j) Examples of Machine configuration

- *Set a p62 system*

String to send :
1 : Set the "MACHINE_NAME" option to "P62"
2 : Del the "MACHINE_ACCESSORY" option
3 : Del the "MACHINE_ADVANDED" option
4 : Reload the system configuration

```
[0x02][0x00]52[0x00][0x0D]MP62[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]3[0x00][0x01]y3[0x00][0x01]z[0x80][0x00][0x00][0x03]
```

- *Set a c151 system with a standard 50mm Znum axie*

String to send :
1 : Set the “MACHINE_NAME” option to “C151”
2 : Set the “MACHINE_ACCESSORY” option to a standard ZNUM of 50 mm
3 : Del the “MACHINE_ADVANDED” option
4 : Reload the system configuration

```
[0x02][0x00][0x35]2[0x00][0x0D]MC150[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]
                  2[0x00][0x35]y[0x00][0x00][0x1A][0x09][0x00][0x00][0x05][0x35][0x03][0x15][0x01][0x89][0x00][0x00][0x00][0x00]
                               [0x06][0x2B][0x9A][0x61][0x06][0x30][0x97][0x81][0x07][0x34][0x95][0x91][0x08][0x3B][0x93][0xA1][0x09][0x43]
                               [0xA2][0xB1][0x0B][0x4D][0xA2][0xC1][0x0E][0x5B][0xA2][0xD2][0x12][0x6F][0xC2][0xF4][0x16][0x8C][0xC1][0xF5]
                  3[0x00][0x01]z
                  [0x80][0x00][0x00][0x03]
```

- *Set a i111s system with lost step control and no 3rd axis*

String to send :
1 : Set the “MACHINE_NAME” option to “I111S”
2 : Del the “MACHINE_ACCESSORY” option
3 : Set the “MACHINE_ADVANDED” option
4 : Reload the system configuration

```
[0x02][0x00][0x35]2[0x00][0x0D]MI111S[0x00][0x00][0x00][0x00][0x00][0x00][0x00]
                  3[0x00][0x01]y
                  2[0x00][0xAB]z[0x15][0x10][0x00][0x08][0x03][0x10][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][0x01][0x02][0x02][0x02][0x02][0x02][0x02][0x01][0x02][0x03][0x04]
                               [0x05][0x06][0x07][0x00][0x00][0x02][0x20][0x00][0x00][0x00]…[0x00]
                  [0x80][0x00][0x00][0x03]
```

k) 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][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][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)][V(1)][Res(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
V : marking speed [0x01] à [0x09]
Reserved : not use
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][0x03]
f    L          H      E    V   Res    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]

- *KEYBOARD Option* Keyboard setting

Code 'C' (`0x43` in hexadecimal)
Data= `[Reserved (1)] [Kind (1)] [Confirm (1)] [Locked (1)] [Russian (1)] [Slave (1)]`
Total command data size =7

Reserved : char `[0x00]` in hexadecimal
Kind : External keyboard layout: [0x01] =AZERTY, [0x02] =QWERTY, [0x03] =QWERTZ
Confirm : [0x01] confirm message when exit window without saving, [0x00] no confirmation
Lock. : [0x01] lock the keyboard, [0x00] unlock the keyboard
Russian : [0x01] Russian keyboard, [0x00] other keyboard
Slave : [0x00] stand alone mode [0x07] slave mode

If Slave = 0: when user press START button: the system start the marking cycle
If Slave = 7: when user press START button: the system send [0x0D] to serial port and host port

Example:
Setting an AZERTY external keyboard
`[0x02][0x00][0x35]2[0x00][0x07]C[0x00][0x01][0x00][0x01][0x00][0x00][0x03]`

- *STATISTICS Option* *Controls* Statistics

Code 'D' (`0x44` in hexadecimal)
Data= `[Nb Cycle(4)][Nb Char(4)]` 9*(`[Force(4)]`) 3*(`[NbPas(6)]`) 10*(`[Options(6)]`
Total command data size =123

Nb Cycle : Number of cycles: a 32-bit integer, the most significant byte to the left
Nb Char : Number of char: a 32-bit integer, the most significant byte to the left
Force : Number of impact for each force: 9 integers each 32-bit, the most significant byte to the left
NbPas : Number of step ran for each motor: 3 integers each 48-bit, the most significant byte to the left
Option : Number for the options: a table of 10 integers each 48-bit, the most significant byte to the left
1st value for the number of autosensing ran
2nd value, for scratching machine: the number of step ran, stylus up
3rd value, for scratching machine: the number of step ran, stylus down
for now, only these 3 values are used

Example:
Getting the control statistics
`[0x02][0x00][0x35]1[0x00][0x01]D[0x03]`
the answer is

```
[0x02]1[0x00][123]D[0x00][0x00][08][62][0x00][0x0C][64][0x00][0x00][0x0C][0x17][0x00][0x0C][0x17][0x00][0x00][0x0C]
[0x00][0x00][0x00][0x0C][0x0C][64][0x00][0x00][0x00][0x0C][0x17][100][0x00][0x00][0x00][0x00][0x01][0x17]
[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]...[0x03]
```

- *MACHINE_NAME Option* machine name (C150, I80, …) caution : letters must be in upper case

Code 'M' (`0x4D` in hexadecimal)
Data= `[Machine Name(11)][Reserved(1)]`
Total command data size =13

MachineName : Name of the machine :
`-S.I.C.-` = Smart Integrated Chip
`SPECIFIC` = Specific configuration
others = standard machine file name

Reserved : char `[0x00]` in hexadecimal

Example:
Setting for the C151

- *MACHINE_ACCESSORY Option* 3rd axie configuration

Code 'y' (0x79 in hexadecimal)
Data=[Lg(4)][Offset(2)][Ratio(2)][P(1)][V(1)][K(1)][C(1)][a(1)][b(1)][c(1)][d(1)][Speed(36)]
Total command data size =53

LG : Size of the axie in step a 32 bit integer
Offset : Offset in step from origine of the axie in step a 16 bit signed integer
Ratio : ratio of the axie in step per centimeter a 16 bit integer
P : for futur use, need to be 3. It is an 8 bit integer
V : Origin speed of the axie time 10. From 15 (=150 step/s) to 100(=1000 step/s) an 8 bit integer
K : 3rd axie kind. It is an 8 bit integer :
- **0**: no 3rd axie
- **1**: Z num
- **2**: Z binary
- **3**: D-Axis
- **4**: Feeder

C : an 8 bit binary combination:
- **Bit 1**: value of the sensor when activated
- **Bits 2,3,4**: value of the motor current in stand by (0=0A, 1=0.75A, 2=1A, 3=1.5A, 4=3A)
- **Bits 5,7,7**: value of the motor current in marking cycle (0=0A, 1=0.75A, 2=1A, 3=1.5A, 4=3A)
- **Bit 8**: reserved

a, b, c, d : optional data depend on the 3rd axie kind. Tey are 8 bit integers each

for Z num :
- **a**: 0 if no Autosensig and 1 if AutoSensing actif
- **b**: internal length of Autosensing piston in 10th mm from 0 to 99
- **c**: input where the autosensing is linked
- **d**: value of the sensor when active (0 or 1)

for Z binary
- **a**: activation mode :0 for file activation and 1 for Line activation
- **b**: input where retract sensor is connected (0 if no sensor)
- **c**: input where out sensor is connected (0 if no sensor)
- **d**: output to drive the axie (from 1 to 8)
- **Offset**: for this axie, Offset represents the delay "IN" in millisecond
- **Ratio**: for this axie, Ratio represents the delay "OUT" in millisecond

for D-Axis
- **a, b, c, d** : not used
- **Ratio**: for this axie, Ratio represents the number of step per turn

for Feeder
- **a**: hidden time return (1 for YES and 0 for NO)
- **b**: part detection sensor (0 for none)
- **c**: output clamping(0 for none)
- **d**: not used
- **Offset**: for this axie, Offset represents the offset from the marking position to the evatuation position in step.

Speed : this is the 9 speed configuration (9 x 4 bytes). For each speed:
- **a**: starting speed time 10 + 90 from 6 (=150 step/s) to 200 (= 2090 step/s)
- **b**: max steed time 15 + 90 from 4 (=150 step/s) to 255 (=3915 step/s)
- **c**: please, use fallowing data
- **d**: please, use fallowing data

Speed	1	2	3	4	5	6	7	8	9
c	0x9A	0x97	0x95	0x93	0xA2	0xA2	0xA2	0xC2	0xC1
d	0x61	0x81	0x91	0xA1	0xB1	0xC1	0xD2	0xF4	0xF5

Example:
Setting for a Z axie of 50 mm

```
[0x02][0x00][0x35]2[0x00][0x35]y[0x00][0x00][0x1A][0x09][0x00][0x00][0x05][0x35][0x03][0x15][0x01][0x89][0x00][0x00][0x00][0x00][0x06][0x2B][0x9A][0x61][0x06][0x30][0x97][0x81][0x07][0x34][0x95][0x91][0x08][0x3B][0x93][0xA1][0x09][0x43][0xA2][0xB1][0x0B][0x4D][0xA2][0xC1][0x0E][0x5B][0xA2][0xD2][0x12][0x6F][0xC2][0xF4][0x16][0x8C][0xC1][0xF5][0x03]
```

- *MACHINE_ADVANCED Option* advanced configuration of the system (IO config, axies priority, …)

Code 'z' (0x7A in hexadecimal)
Data= [Table(30)][Buffer(140)]
Total command data size =171

Table : table of 10 items. Each item contain 3 bytes :
- **a**: Code of the item
- **b**: Length of it's data
- **c**: offset where data start in the Buffer

Buffer : datas of items configurations

Code definition of items

0x15: Input/Output configuration. Data=[Input(8)][Output(8)] Length=16 bytes

Input: settings of the 8 inputs, for each :
- 0 = no settings
- 1 = start the cycle
- 2 = file selection
- 3 = pause validation
- 4 = home position

Output: settings of the 8 output, for each :
- 0 = no settings
- 1 = marking error
- 2 = last dot marked

3 = Cycle in progress
4 = in pause
5 = system ready
6 = autosensing : part detected out of bounds
7 = autosensing : no part detected in bounds

0x16: Axies priority configuration. Data= [First move (1)][Second move (1)][Third move (1)] Length = 3 bytes
for each move, data represent the concerned motors, witch is a binary combination of
X axis: 1
Y axis: 2
3rd axis: 4

0x18: Cycle configuration. Data = [Start of cycle (1)][End of cycle (1)][value (1)] Length = 3 bytes
Start of cycle: to to before first dot is printed. 0 = home position. 1 = Hold
End of cycle: to do after the last dot is printed. 0 = home position. 1 = hold. 2 = lost step test.
value: lost step definition

Example:

Default IO setting + lost step test at end of cycle.

```
[0x02][0x00][0x35]2[0x00][0xAB]z[0x15][0x10][0x00] [0x08][0x03][0x10] [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][0x01][0x02][0x02][0x02][0x02][0x02][0x02] [0x01][0x02][0x03][0x04][0x05][0x06][0x07][0x00]
[0x00][0x02][0x20] [0x00][0x00][0x00]…[0x00][0x03]
```

- *SHIFT Option* Shift setting

Code 'E' (0x45 in hexadecimal)
Data=10*([Name (19)][Reserved (1)][Time (2)])
Total command data size =221

Name : Shift Name, padding with NULL chars to the right :
Reserved : char [0x00] in hexadecimal
Time : begining of shift time, a 16 bit integer which value is : Hour*100 + Minute

Example:

Setting 3 shift named MATIN *starting at 01h30,* MIDI *starting at 12h30 and* SOIR *starting at 20h30*

```
[0x02][00][0x35]2[00][221]EMATIN[00][00][00][00][00][00][00][00][00][00][00][00][00][00][230]
                          MIDI[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][0x05][0x32]
                          SOIR[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][0x07][258]
                          [00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
                          ...
                          [00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][0x03]
```

- *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 *ane* GLOB2.

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

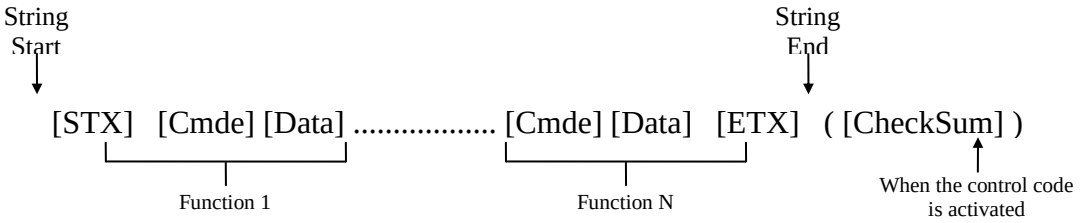
Code 'U' (`0x55` in hexadecimal)
Data= `[`Unit`(1)]`
Total command data size =2
Unit `[0x00]` for inch system and `[0x01]` for metric system

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 *Start Cycle* 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:* X Y Z W H Ang. Radius D O F Text on N characters ***X, Y :*** *Coordinates of the text to be marked (–9999 to +9999 in tenth of mm),* ***Z:*** *Coordinates of the text to be marked (0000 to 9999 in tenth of mm),* ***W, H:*** *Width and height of the character to be marked(000 to 999 in tenth of mm),* ***Ang:*** *marking angle (000 to 359 in degrees),* ***Radiu:*** *Radius of the circle for circular markings (0000 to 9999 in tenth of mm),* ***D:*** *Depth - Impact force (0=no impact to 9=large impact),* ***O:*** *Orientation - Marking direction ('N' or 'I' for Normal or Reverse orientation),* ***F:*** *Character font (1 to 9),* ***Text:*** *Text to be marked*
[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] [ACK]
  ```

  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 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 30 33 30 30
  33 30 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 35 4E 31 41 42 43 64 65 66 0B 2B 30 31 30 30 2B 31 30 30 30 30 30 30 30 30
  30 33 30 30 33 30 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+020000001001000000000005N1[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 30
  35 4E 31 04 4C 6F 67 6F 20 5A 58 33 06 03
  ```

- Datamatrix ECC200 string

  EEC200(*[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+020000001001000000000005N1[EOT]ECC200(30S,12345)
  ```

  In hexadecimal:

  ```
  2B 30 31 30 30 2B 30 32 30 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+020000001001000000000005N1[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
```

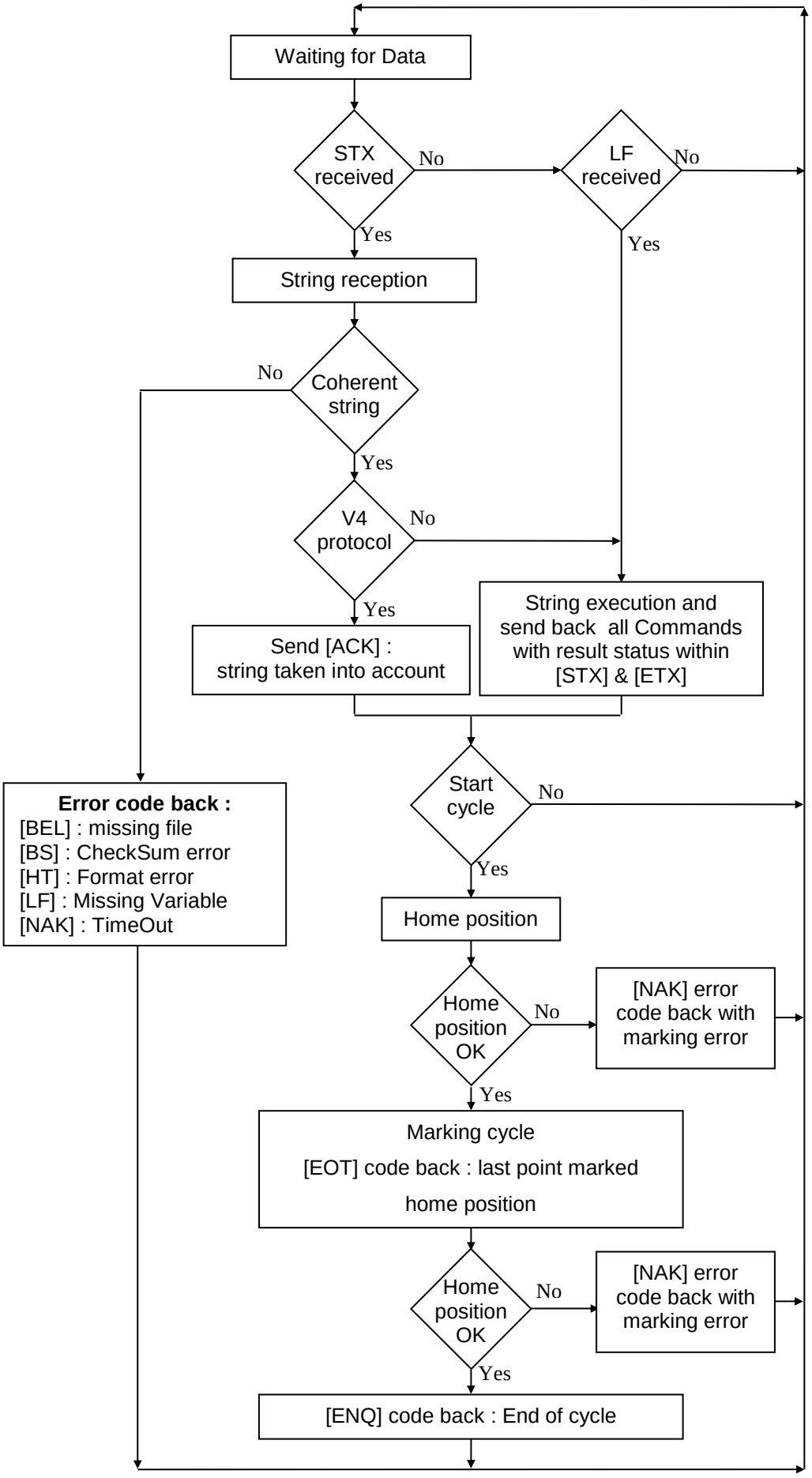
<u>Remark :</u> "Logo " "Pause" and"ECC200" are reserved words ; you absolutely must respect the character case (upper/lower) as well as the space character between "Logo" and the name of the logo file to be marked.

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	ß
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	°
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	Doubble 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

II - SLAVE MODE – COMMUNICATION PROTOCOL – INPUT / OUTPUT

1 - SERIAL COMMUNICATION

a) TEXT format of the communication protocol

ASCII CODES

Codes	Hexa	decimal
[CR]	0D	13
[LF]	0A	10
␣	20	32
[NUL]	00	00
[SOH]	01	01
[STX]	02	02
[ETX]	03	03
[EOT]	04	04
[ENQ]	05	05
[ACK]	06	06
[BS]	08	08
[VT]	0B	11
[NAK]	15	21

- *String format to be sent:*

<Command> ␣ <Data1> ␣ < Data2> [CR][LF]

where

< Command >	is the command code
< Data*i*>	are the data of the command
[CR]	is the `0x0D` in hexadecimal (facultative)
[LF]	is the `0x0A` in hexadecimale
␣	space char

- *Response of the system*

<command> <Answer>[CR][LF]
The control response to every command. Mostly, the anwer will be OK or BAD FORMAT.

- *Commun functions*

Command	Description	Data to send	Response of the system
LOADFILE	Select and lod a file	Name of the file to be loaded (11 chars max)	`OK` : file loaded `ERROR` : File nor found
SETVAR	Set a variable to the current file	<Name of the var> <Value>	`OK` `VAR NOT FOUND`
RUN	Start the marking *Start cycle* 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,	*No data*	`OK` → *cycle in process* `[EOT]` → *last dot marked* `[ENQ]` → *back to home position* `[NAK][Err1][Err2][Err3]` → if an error occurse (see apendix)
RESETERROR	If an error occurse, you need to clear the machin status (Same as if you press the start button)	*No data*	`OK`
NEWFILE	Create a new file	Marking speed (1 à 9), Fast speed (1 à9), Crossed zéro (0 ou 1), Name of the fole (optional)	`OK` : empty file created `BAD ARGUMENT`
INSERTTEXTLINE	Add a line to the current file	X, Y, Z, W, H (in 10th of mm) Angle (in hendredth of dedrees from –18000 to 18000) radius (in 10th of mm) Space between chars (from 0 to 10) Force (0 à 9) Quality (1 à 9) Text to be printed	`OK` `BAD ARGUMENT`:

- *Examples*

Select the '`AB12`' file without running the cycle

Send the string : **`LOADFILE AB12`**[CR][LF]
in hexadecimal : 4C 4F 41 44 46 49 4c 45 20 41 42 31 31 0D 0A

Select the '`AB12`' file and start the cycle

Send the string: **`LOADFILE AB12`**[CR][LF]**`RUN`**[CR][LF]
in hexadecimal: 4C 4F 41 44 46 49 4c 45 20 41 42 31 31 0D 0A 52 55 4E 0D 0A

Select the '`AB12`' file, set the '`OF`' variable with '`12345`' value and start the cycle

Send the string: **`LOADFILE AB12`**[CR][LF]**`SETTEXTVAR OF 12345`**[CR][LF]**`RUN`**[CR][LF]
in hexadecimal: 4C 4F 41 44 46 49 4c 45 20 41 42 31 31 0D 0A 53 45 54 54 45 58 54 56 41 52 20
4F 46 20 31 32 33 34 35 0D 0A 52 55 4E 0D 0A

Sent a complete marking trame in a '`TEMP`' named file and start the cycle
X=`10`*mm, Y*=`12`*mm, Char of 5*`x7`*mm, Force*=`5`*, Quality=double with text*=`HELLO WORLD`

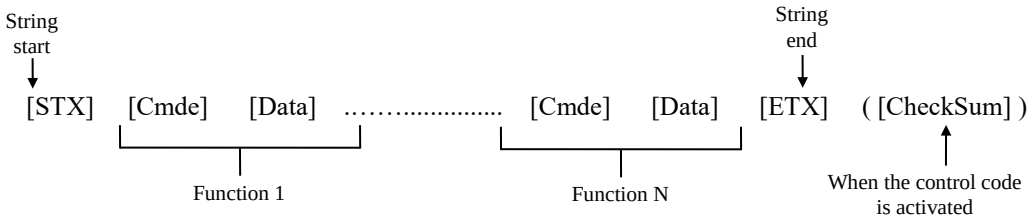
Send the string: **`NEWFILE 5 7 0 TEMP`**[CR][LF]
`INSERTTEXTLINE 100 120 0 50 70 0 0 2 5 2 HELLO WORLD` [CR][LF]
`RUN`[CR][LF]
in hexadecimal: 4E 45 57 46 49 4C 45 20 35 20 37 20 30 20 54 45 4D 50 0D 0A
49 4E 53 45 52 54 54 45 58 54 4C 49 4E 45 20 31 30 30 20 31 32 30 20 30 20 35 30 20 37 30 20
30 20 30 20 32 20 35 20 35 20 48 45 4C 4C 4F 20 57 4F 52 4C 44 0D 0A 52 55 4E 0D 0A

b) V4 protocol

•*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).

•*String to be sent :*

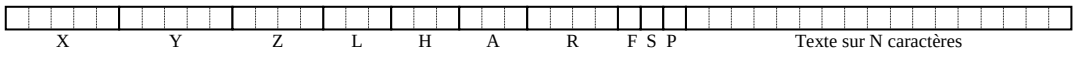


Control code CheckSum : In order to detect a possible error in the transmission, the CheckSum is calculated depending on the string sent by the main system and receptioned by the machine. If the string has been correctly transmitted, the code calculated by the marking machine is the same as the code sent by the main system. When a check sum error is detected, the syctem send back the [BS] code.

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

•*List of functions*

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

Cmde	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 *Start Cycle* 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:*  X Y Z L H A R F S P Texte sur N caractères ***X, Y :*** *Coordinates of the text to be marked (–9999 to +9999 in tenth of mm),* ***Z:*** *Coordinates of the text to be marked (0000 to 9999 in tenth of mm),* ***W, H:*** *Width and height of the character to be marked(000 to 999 in tenth of mm),* ***Ang:*** *marking angle (000 to 359 in degrees),* ***Radiu:*** *Radius of the circle for circular markings (0000 to 9999 in tenth of mm),* ***D:*** *Depth - Impact force (0=no impact to 9=large impact),* ***O:*** *Orientation - Marking direction ('N' or 'I' for Normal or Reverse orientation),* ***F:*** *Character font (1 to 9),* ***Text:*** *Text to be marked*

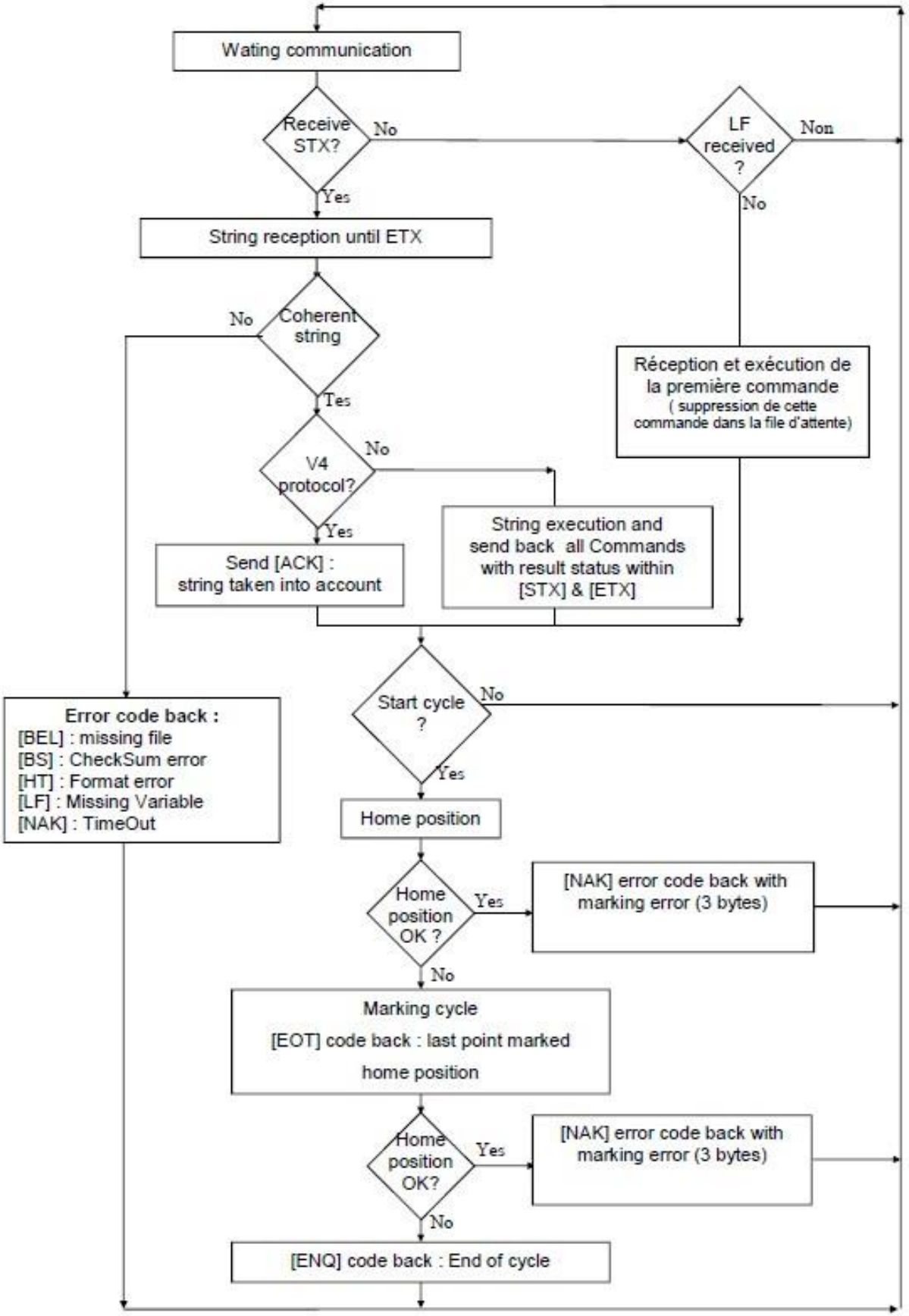
•*Examples*

- Select the '`AB12`' file without checksum and without running the cycle:
  Send the string:
  [STX] [NUL] [SOH] AB12 [ETX]
  in hexadecimal:
  02 00 01 41 42 31 32 03
- Affect the '`OF`' variable in the current file and start the cycle:
  Send the string:
  [STX] [ENQ] OF=142BH05 [ACK] [ETX] [Check-summ]
  in hexadecimal:
  02 05 4F 46 3D 31 34 32 42 48 30 35 06 03 0E
- Sent a complete marking trame, and start the cycle, and select speed 9:
  Send the string:
  [STX] [NUL] [LF] 9 [VT] +0100+0500000002002000000005N1ABCdef [VT] +0100+1000000003003000000005N2Z123 [ACK] [ETX]
  in hexadecimal:
  02 00 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 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 30 33 30 30 33 30 30 30 30 30 30 30 30 30 35 4E 32 5A 31 32 33 06 03

c) ASCII Codes:

Dec Code	Hex Code	Val	Dec Code	Hex Code	Val	Dec Code	Hex Code	Val	Dec Code	Hex Code	Val	Dec Code	Hex Code	Val	Dec Code	Hex Code	Val	Dec Code	Hex Code	Val
00	00	NUL	01	01	SOH	02	02	STX	03	03	ETX	04	04	EOT	05	05	ENQ	06	06	ACK
07	07	BEL	08	08	BS	09	09	HT	10	0A	LF	11	0B	VT	12	0C	FF	13	0D	CR
14	0E	SO	15	0F	SI	16	10	DLE	17	11	DC1	18	12	DC2	19	13	DC3	20	14	DC4
21	15	NAK																		
32	20		64	40	@	96	60	`							192	C0	🡐			
33	21	!	65	41	A	97	61	a	129	81	ü				193	C1	🡑	225	E1	ß
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												
38	26	&	70	46	F	102	66	f	134	86	å									
39	27	'	71	47	G	103	67	g												
40	28	(	72	48	H	104	68	h												
41	29	)	73	49	I	105	69	i												
42	2A	*	74	4A	J	106	6A	j												
43	2B	+	75	4B	K	107	6B	k												
44	2C	,	76	4C	L	108	6C	l												
45	2D	-	77	4D	M	109	6D	m												
46	2E	.	78	4E	N	110	6E	n	142	8E	Ä									
47	2F	/	79	4F	O	111	6F	o	143	8F	Å									
48	30	0	80	50	P	112	70	p												
49	31	1	81	51	Q	113	71	q												
50	32	2	82	52	R	114	72	r												
51	33	3	83	53	S	115	73	s												
52	34	4	84	54	T	116	74	t	148	94	ö									
53	35	5	85	55	U	117	75	u												
54	36	6	86	56	V	118	76	v												
55	37	7	87	57	W	119	77	w												
56	38	8	88	58	X	120	78	x										248	F8	°
57	39	9	89	59	Y	121	79	y	153	99	Ö									
58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü									
59	3B	;	91	5B	[	123	7B	{	155	9B	ø							251	FB	¹
60	3C	<	92	5C	\	124	7C	|										252	FC	³
61	3D	=	93	5D	]	125	7D	}	157	9D	Ø							253	FD	²
62	3E	>	94	5E	^	126	7E	~												
63	3F	?	95	5F	_							191	BF	🡒						

d) Code sent back from the system (protocol v4 and v5)



e) Error codes returning on a marking error

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 **Or** - Error with binary axis
02 00 00	02 00 00	0000 0010 0000 0000 0000 0000	- Error empty feeder **Or** - Error with Autosensing : part detected out of bound
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	Historic 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

Note : SIC TERMINAL use the decimal display of the ASCII codes .

f) Description of the serial ports

- *"HOST"*

SUB-D 9Pts female	
Pin	RS232
1	NC
2	RX
3	TX
4	NC
5	O Volt
6	NC
7	NC
8	NC
9	NC

Notes :

- This serial port is set at 9600 baud, with 8 data bits, 1 stop bit, and with no parity.
- It is allocated to the loading of the "controller" software and to the saving of the parameter setting files.
- It can also be used for the slave operating mode but it is only in standard type RS232 and the parameters can not be modified.

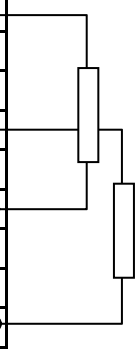
- *"SERIAL"*

SUB-D 9Pts female	
Pin	RS232
1	NC
2	RX
3	TX
4	NC
5	O Volt
6	NC
7	NC
8	NC
9	NC

Notes :

- In the standard version, this serial port is set in RS232 but it can also be set in RS422 using the tabs.
- It is strictly usable in *Slave Operating Mode.*.
- In order to set the configuration, open the hatch located under the marking controller, which will give you access to the configuration tabs. See diagrams below :

SUB-D 9Pts female	
Pin	RS422
1	RX+
2	NC
3	NC
4	TX-
5	O Volt
6	RX-
7	NC
8	NC
9	TX+



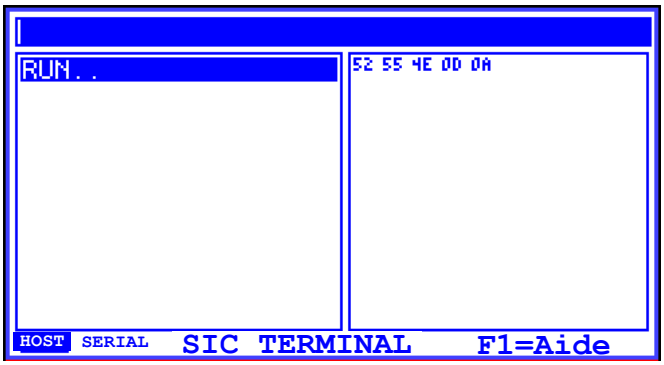
For RS422, it is need to add two end line resistances of 120 ohm 1/4W

RS232, RS422 RS485

The SERIAL port can be set to RS232 (default), RS422 or RS485.
See SERIAL COMMUNICATION PARAMETER under E10 v5 software.

g) RS232 test

You can manually test the RS232 of the e6. Goes to menu ? → SIC TERMINAL, Follow screen will appear :



This function let you test serial communication with the e6. You can send string and see received string

- The edition filed is the string to be sent
- the left list correspond to the last received string
- the right text is the selected line of the left list RECEIVED DATA ARE IN HEXADECIMAL

↓ ↑ : Select a line within the received string list

F6 Send string to the Selectionned RS port

F7 Add check sum

F8 Clear the received string

F9 SERIAL Port <-> HOST Port

F12 or Esc return to previus SCREEN

2 - TOR InputS / Outputs

a) Cableling

N°	Name	function
1	Input 1 (NC)	Power off
2	Input 2	Cycle start
3	Input 3	*file selection*
4	Input 4	*file selection*
5	Input 5	*file selection*
6	Input 6	*file selection*
7	Input 7	*file selection*
8	Input 8	*file selection*
9	Gnd_ Input	Communs input
10	Gnd_ Input	Communs input
11	output 6 dry contact	" AutoSensing error: close when no part was detected "
12	output 6 dry contact	" AutoSensing error: close when no part was detected "
13	output 7 dry contact	"AutoSensing error: close when a obstacle has been detected "
14	output 7 dry contact	"AutoSensing error: close when a obstacle has been detected "
15	output 5 dry contact	" closed when e6 ready "
16	output 5 dry contact	" closed when e6 ready "
17		
18	GND du 5V	Ground
19	+5V	5V (500 mA max)
20	output 1 dry contact	" closed during the marking cycle "
21	output 1 dry contact	" closed during the marking cycle "
22	output 2 dry contact	" impulsion at the last printed dot, time 250ms "

23	output 2 dry contact	" impulsion at the last printed dot, time 250ms "
24	output 3 dry contact	" Default "
25	output 3 dry contact	" Default "
26		
27	output 4	« Pause »
28	Ground 24 V	Ground 24V
29	24V	+24V (500 mA max)
30		
31		
32	output 8 dry contact (NC)	" open when stylus need to be changed "
33	output 8 dry contact (NC)	" open when stylus need to be changed "
34		
35		
36		
37		

b) Electric limit

•*Dry contact output (relays)*

- Switched current max.. = 1 A.
- Switched voltage max. = 48 V.
- Contact resistance < 100 mΩ.

•*Output 4 (transistor)*

This transistor pluh the pin #27 to the ground (pin #28).

- Switched current max. = 1 A.
- Switched voltage max. = 30 V.

•*Inputs*

Activating an input by grounding its pin (#1 to #8) to pin #9 or #10

c) File selection

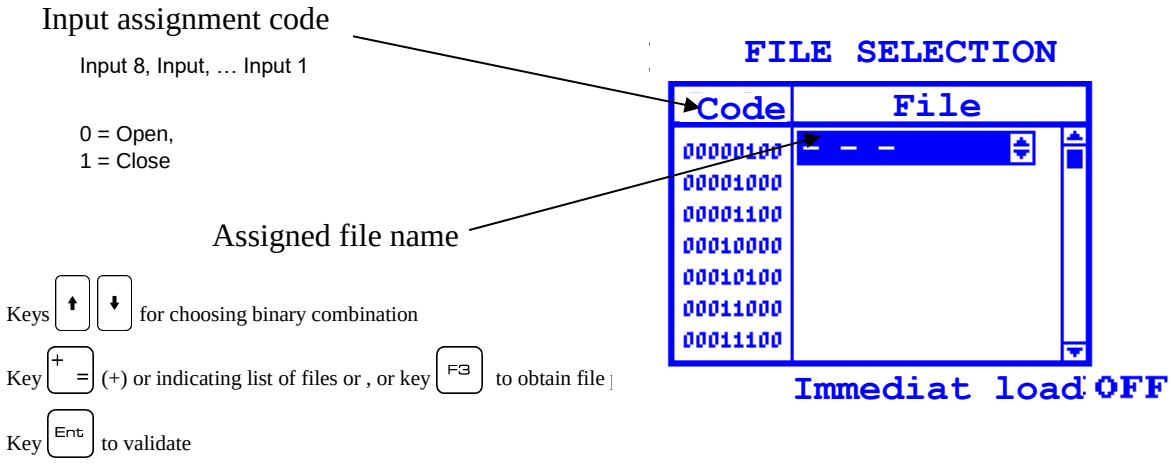
The selection of some of the controller input/output board inputs allows automatic opening of pre-recorded marking files (up to 63 files).

A maximum of 6 inputs are assigned to this task (inputs 3 to 8). Each "input" may be open (state "0") or closed (state "1") by dry contacts (*) to form a binary code.

(*) A dry contact can be assimilated to an open or closed switch connecting contact 9 or 10 (input common) to one of the inputs, without any voltage.

This binary chain, converted into decimal values, corresponds to a number varying from 0 to 63 (maximum). This is the value which will be associated with a marking file.

You will be able to assign a marking file to each binary input combination.



Example for the assignment code

Input	8	7	6	5	4	3	2	1		Binary code	Decimal value
State	1	0	1	0	1	0	0	0		10101000	168

Remarks:

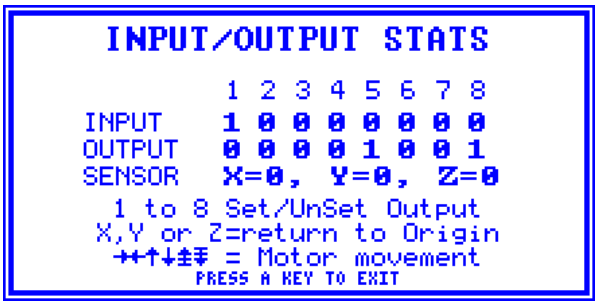
- The state of the inputs is scanned only when the cycle starts. Be sure to set the state of the inputs before activating the cycle start.
- Some accessories or options use inputs. These inputs will no longer be available for selecting the file which is limited.

d) Input/Output test

You can manually test the input/output stats of the e6. Goes to menu :

.

Follow screen will appear :



INPUT line gives the input stats of the e6 :
1 active (contact is close with commun)
0 inactive (contact is open)

OUTPUT line let you set the stats: use keys 1 to 8.

F10 key will display on live IO status in marking mode screen

F11 key will cancel previous display.

e) Operating timing diagram

During the marking cycle the controller outputs are activated as follows:

Start of cycle :	The « contoller ready » output changes to the inactive state The current cycle output changes to the active state
Last marked point:	The last marked point output changes to the active state
Return to origin	The last marked point output changes to the inactive state The current cycle output changes to the inactive state The « Controller ready » output changes to the active state
Error during cycle:	The « Fault » output changes to the active state The current cycle output changes to the inactive state The last marked point output changes to the active state The ready controller output remains inactive.
Validation of pause message:	The machine returns to the origin then, The current cycle output changes to the inactive state

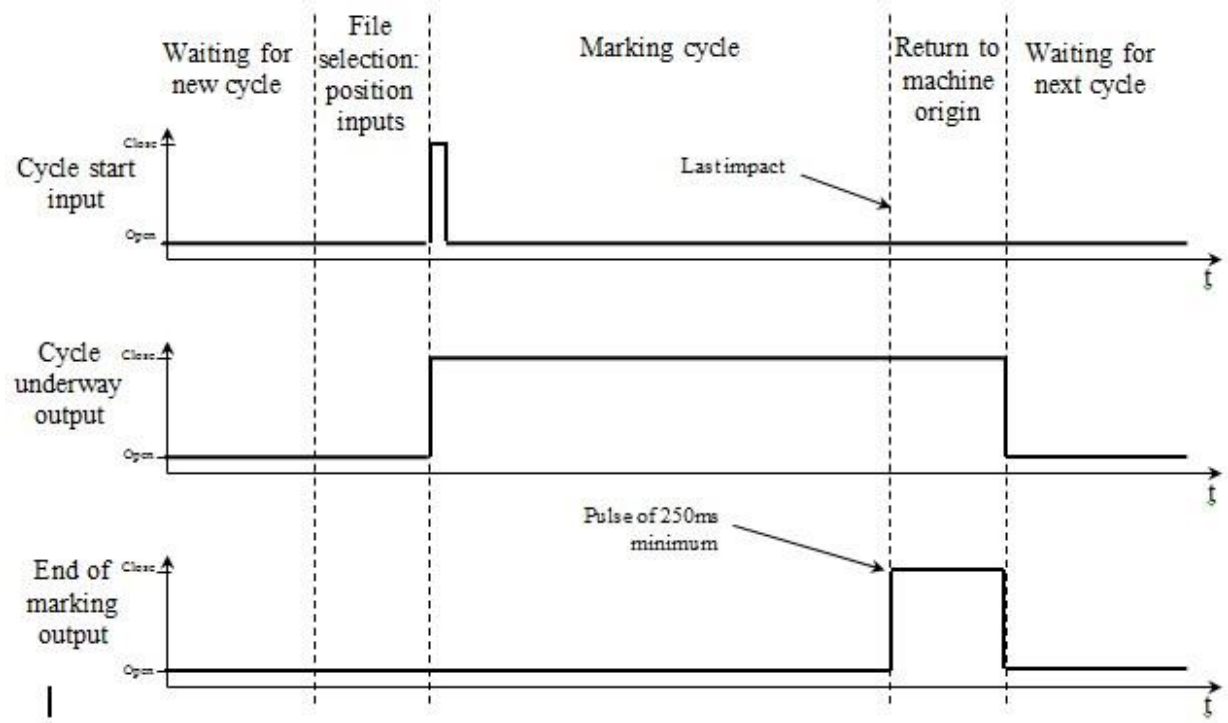
The « controller ready » output changes to the active state
The « Fault » output changes to the active state

Pause during cycle: The « Pause » output changes to the active state

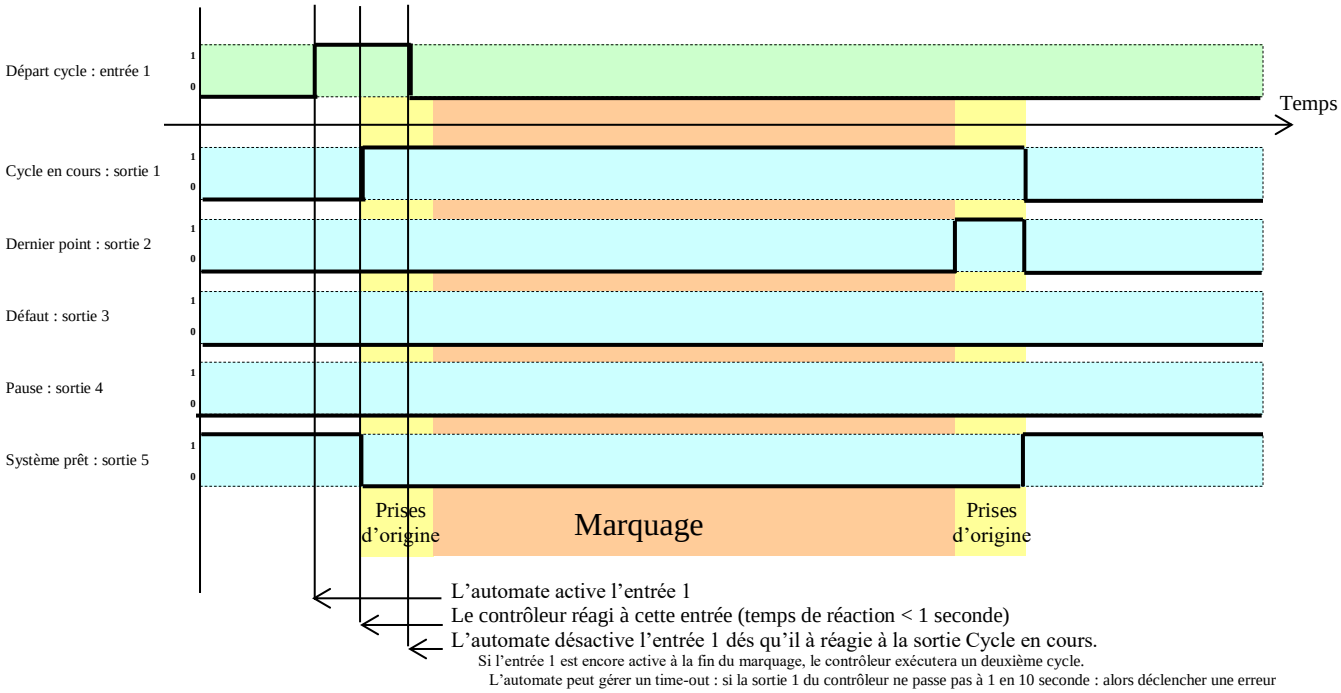
Validation of pause message: The « Pause » output changes to the inactive state

In a standard configuration

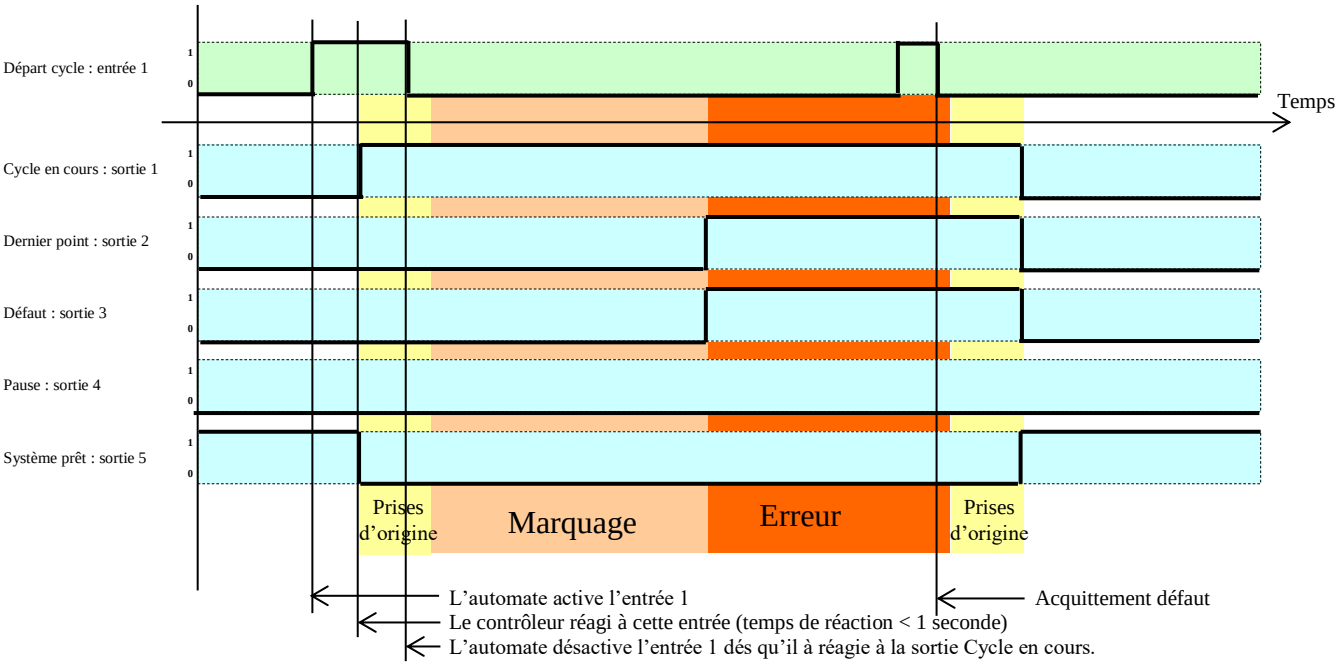
Output « Cycle in progress »	is active when the contact is close between bornes 20 et 21
Output « End of marking »	is active when the contact is close between bornes 22 et 23
Output « default »	is active when the contact is close between bornes 24 et 25
Output « pause »	is active when the contact is close between bornes 27 et 29
Output « e6 Ready »	is active when the contact is close between bornes 15 et 16



Synoptique : cas d'un marquage sans erreur

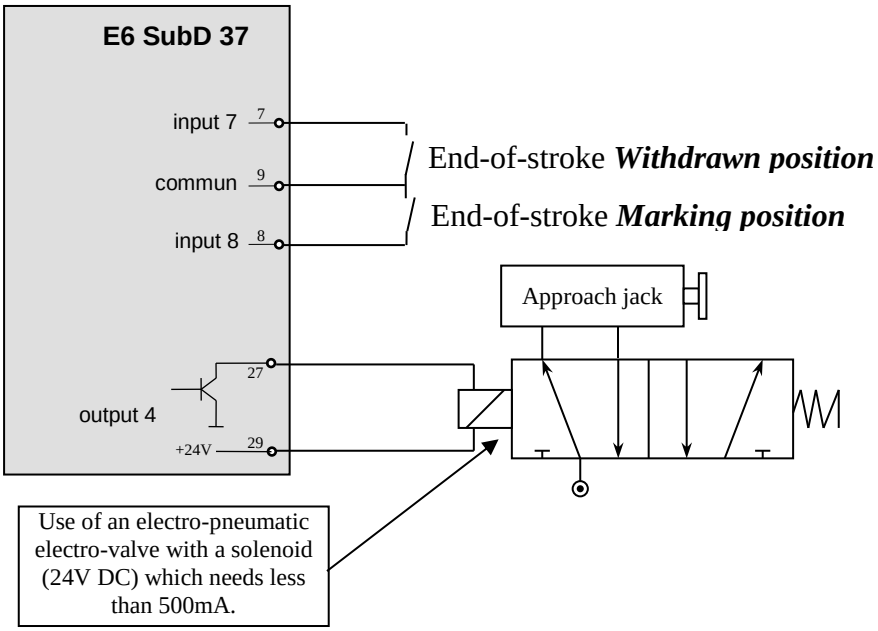


Synoptique : cas d'un marquage avec erreur

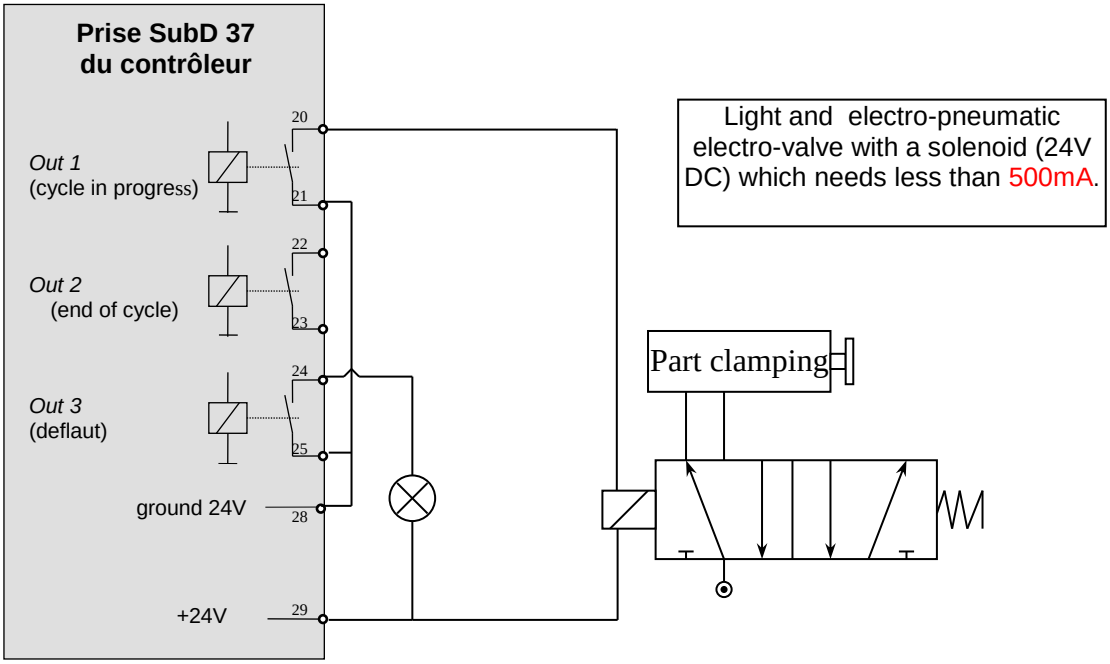


Cabling examples

- *Wiring of the pneumatic Z-axis*



- *Use of the I/O with the clamping of a part being marked, and defaut announced by a light.*



In the case of I/O communication with a PLC, the inputs of the controller must be dry contacts with no tension.

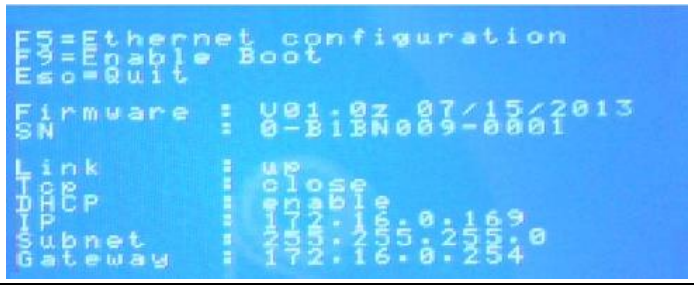
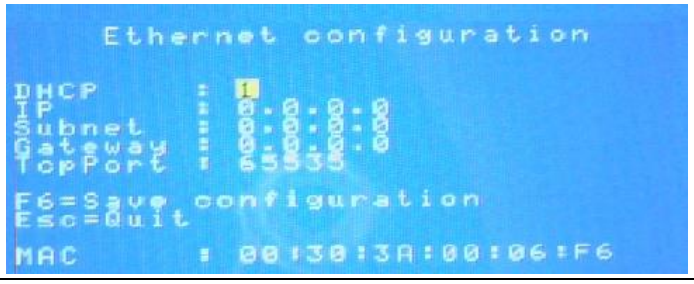
To avoid any trouble, we recommend that you interface the static relay output by an external mechanical relay.

III - CONFIGURATION OF THE COMMUNICATION PORT (OPTIONAL BOARDS):

Important note: The use of an optional communication boards on the e10 controller disable the RS232 port.

a) Ethernet

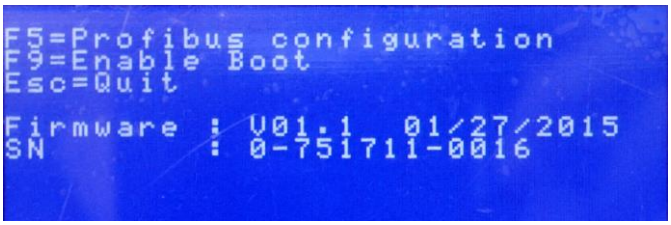
Power up the E10 controller (switch 15, figure 1) by pressing on F1 at the same time

	Press F12 If the communication card is not present or not detected, F12 = Communication board is not displayed
	Press F5
	DHCP: 1, Address defined by the server TcpPort must be: 65535 DHCP: 0, Manual configuration
	Enter a compatible network address Press F6 to save
	Press Esc to exit

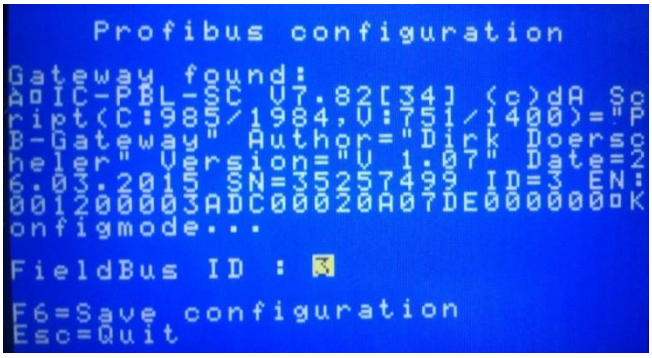
b) Profibus

- Switch the controller in Boot mode: press F1 at the beginning of the boot:

- Press F12 to enter on the COM board boot mode:



- Press F5 to enter on the Profibus configuration:



From now you can change the fieldbus ID:

1. Enter a number from 1 to 126
2. Press F6 to save and ESC two time to reboot the controller

c) Profinet and Ethernet/IP

- Switch the controller in Boot mode: press F1 at the beginning of the boot:
- Press F12 to enter on the COM board boot mode:
- Press F5 to enter on the Profinet or Etehrnet/IP configuration menu and check the IP address (Change it if necessary). It will be used further to set up the connection.

Example:

DHCP	0: The controller IP address is set manually (DHCP disabled) 1: The controller IP address will be set by the DHCP controller (DHCP enabled)
IP	The IP address of the controller
Subnet	The IP subnet to be used in conjunction with the IP address
Gateway	The IP address of the gateway
Qos	0: No quality of service used 1: Quality of service used

Note: On this screen if "Author= Deutschmann automation…" is displayed, it means the board is not ready to work. The SIC script must be loaded.

- Press "F6" to save a modification.
- Press "ESC" several times to quit and reboot the controller.

IV - FIELDBUS PROTOCOLS

The marking system can be controlled by Profibus, Profinet or Ethernet/IP protocol with a bus width of 16 bytes (standard).
The protocols cannot send the complete data string at once to the E10 controller, as the bus width is restricted to 16 bytes.
Please refer to the previous pages for the commands list.

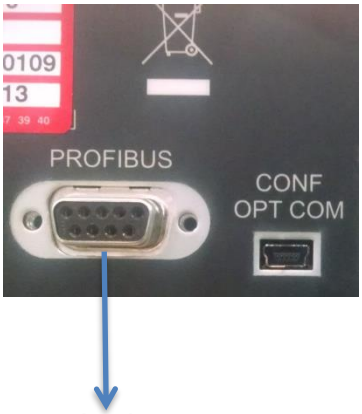
a) Prerequisite

The PROFIBUS module requires the e10v6.1.001 (2-3 axes), e10v6.0.011 (4 axes) software or later versions installed on the controller.
The PROFINET and ETHERNET/IP modules require the e10v6.1.002 (2-3 axes), e10v6.0.012 (4 axes) software or later versions installed on the controller.

b) Integration

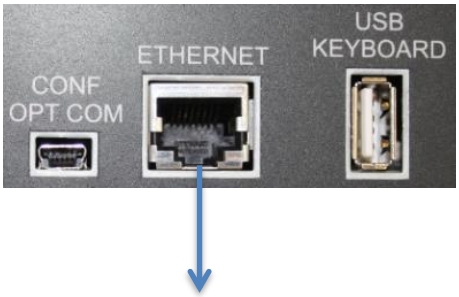
These modules are integrated as COM boards inside the e10 controller.

- **PROFIBUS :**



Standard PROFIBUS SUBD 9 connector

- **PROFINET and Etehrnet/IP:**



Standard PROFINET or Ethernet/IP connector

c) Procedure required

1. Separate your data string into parts of 15 bytes each
2. Put a sequential counting byte (hex (01)-(02)-(03)- etc.) at the beginning of each sequence, append the 15 bytes of data.
3. On the last sequence use hex (FF) as the first byte.
4. If there is only one sequence, use hex (FF).
5. Any sequences sent and not completely filled with data have to be completed using zeroes to fill the 16 bytes (see example 2).

Important note: To be active, two identical and successive sending must be separate by a neutral sending (null string). The system read only sendings after an I/O's state change.

d) Example 1

Select file 'SIC', no checksum and do not start the marking process:
Send this string: [STX] [NUL] [SOH] SIC [ETX]
First sequence is the only one.
Hex values: **FF** 02 00 01 53 49 43 03 00 00 00 00 00 00 00 00

Note : the selected file must be in the controller.

e) Example 2

Send a complete string including start command for the marking sequence and select speed 9:
send this string: [STX] [NUL] [LF] 9 [VT] +0100+0500000002002000000005N1 SICTST [VT] +0100+1000000003003000000005N2 MARKING [ACK] [ETX]

Hex values:
Seq. **1** : **01** 02 00 0A 39 0B 2B 30 31 30 30 2B 30 35 30 30
Seq. **2** : **02** 30 30 30 30 30 32 30 30 32 30 30 30 30 30 30
Seq. **3** : **03** 30 30 35 4E 31 53 49 43 54 53 54 0B 2B 30 31
Seq. **4** : **04** 30 30 2B 31 30 30 30 30 30 30 30 30 33 30 30
Seq. **5** : **05** 33 30 30 30 30 30 30 30 30 35 4E 32 4D 41 52
Seq. **6** : **FF** 4B 49 4E 47 06 03 ***00 00 00 00 00 00 00 00 00***

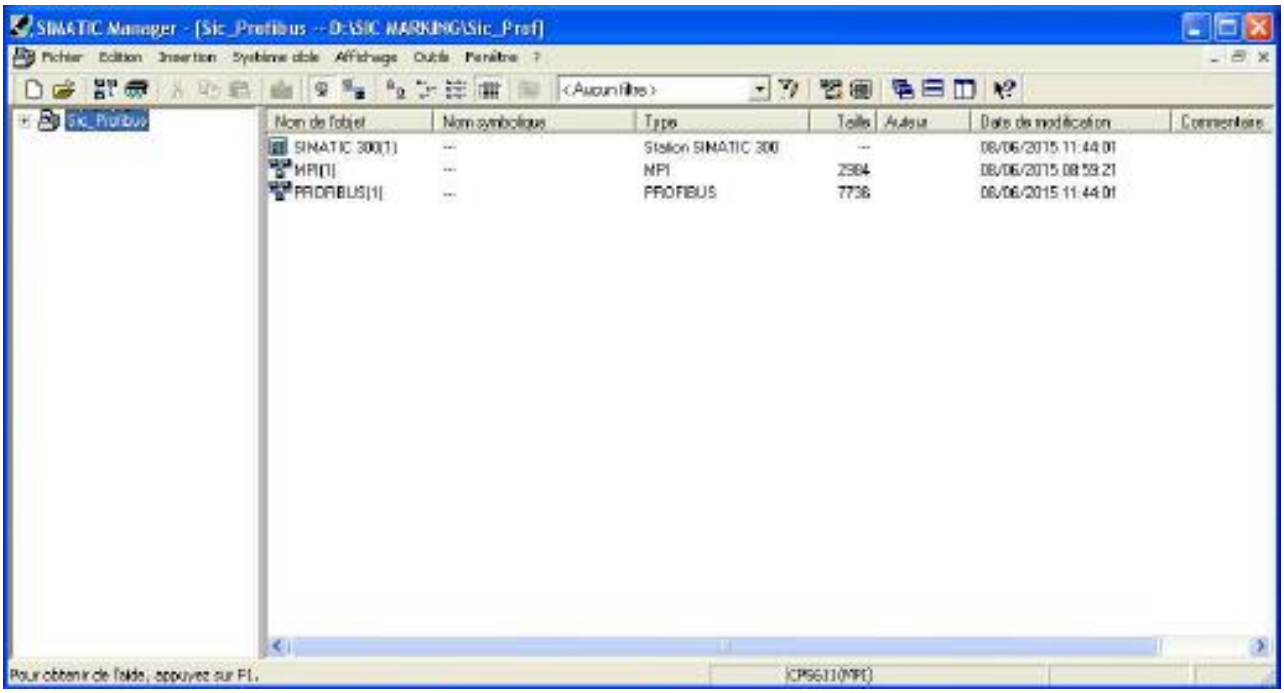
It is possible to create a data string made up of up to 8 single packets.
After the receipt of one packet the Gateway returns the packet number followed by the answer of the command (limited to 16 byte).

f) Example 3,

- **<u>with SIEMENS S700 PLC</u>**

- Open Siemens Step 7 software.
- Open the file needed with the file menu.

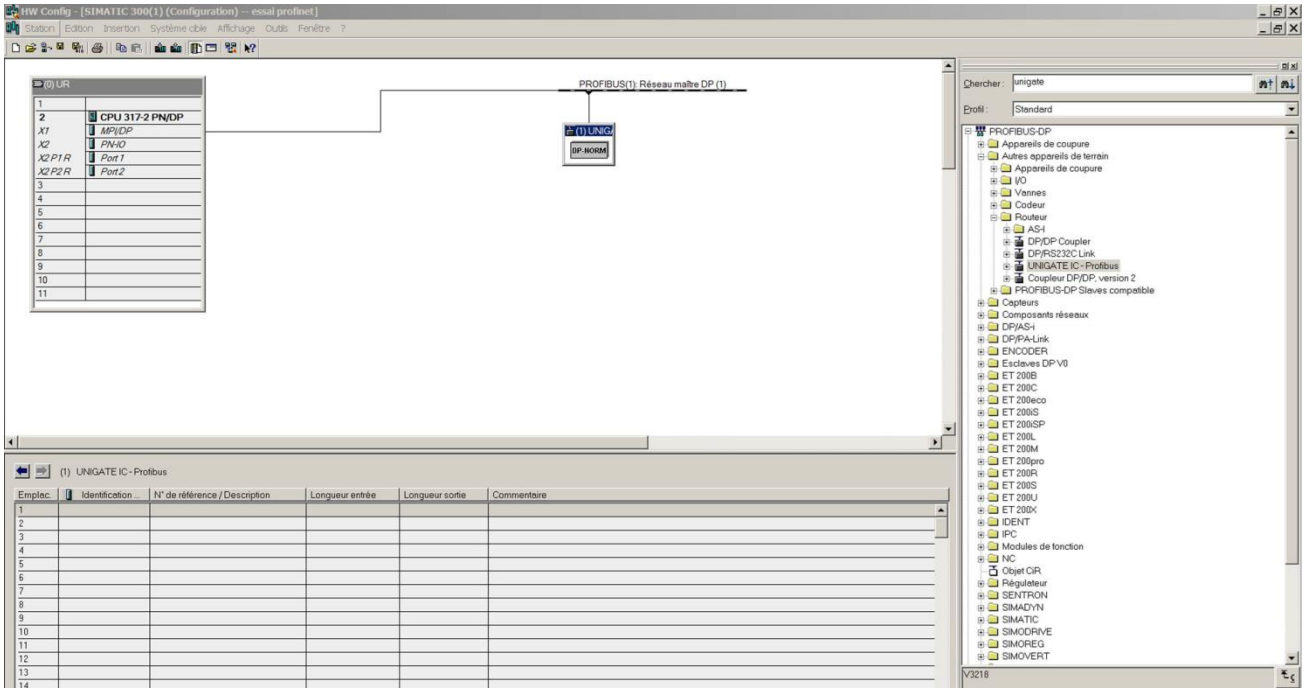
This file is on the e10 CD in the **Profibus\sic_prof** folder.



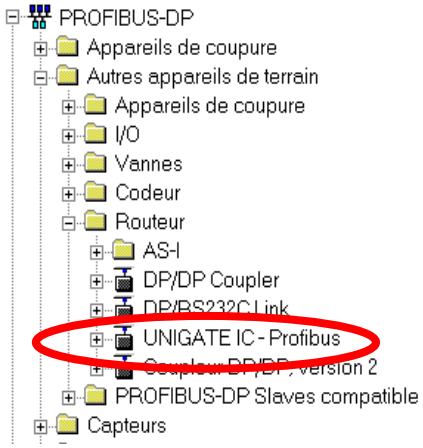
Hardware configuration:

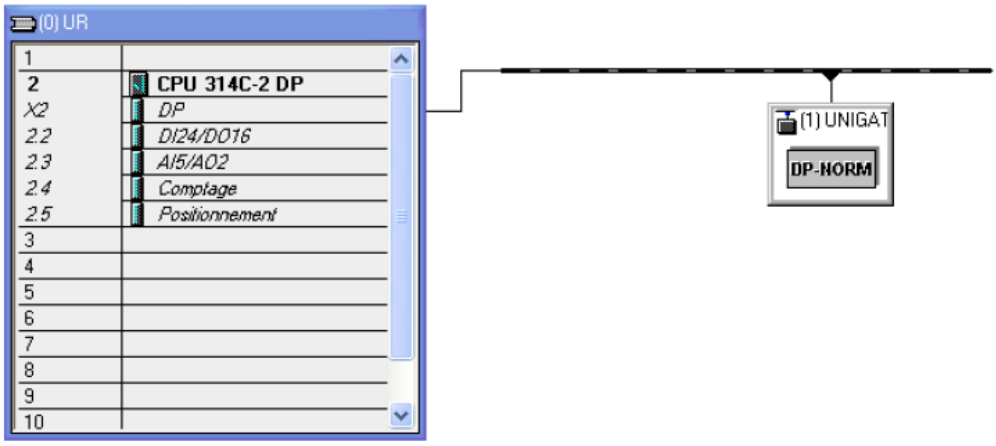
Double click on the hardware element to open the configuration window (HW config).

- **<u>PROFIBUS : (Example SIMATIC)</u>**

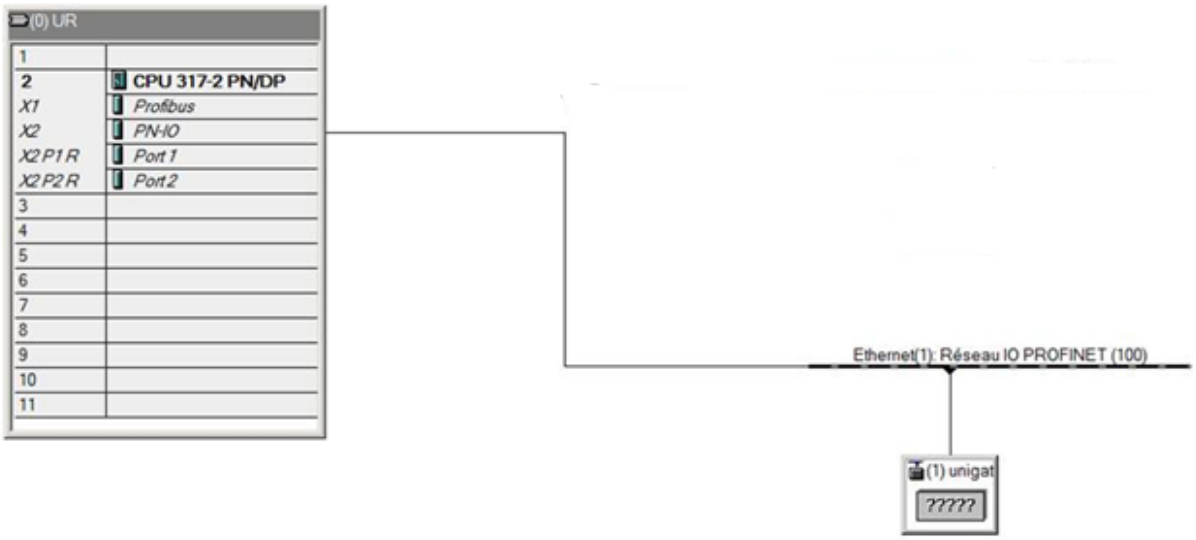


- The file “UGIC3218.gsd” is available in the SIC files directory.



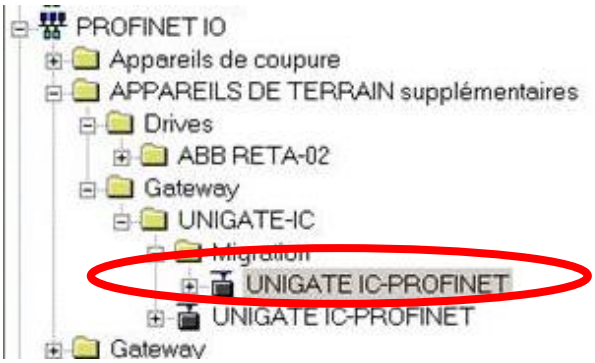


- **PROFINET : (Example SIMATIC)**



- The file “GSDML-V2.3-Deutschmann-UNIGATE-IC-PN-20150504-110100” is available in the SIC files directory.

- Take UNIGATE from the HW config Library.



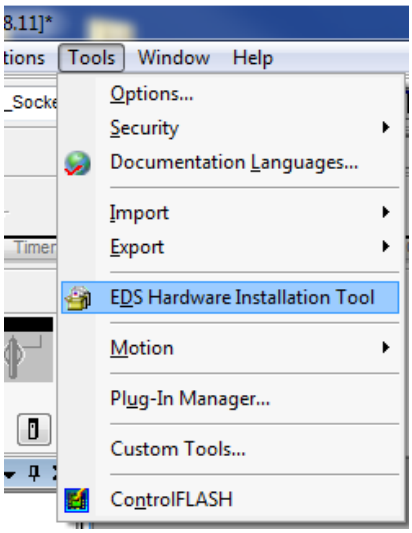
- Integrate the GSDML on the I/O PROFINET network previously created.

- **ETHERNET/IP : (Example RSLogix 5000)**

In order to address properly in an EtherNet/IP network, an eds file is provided with the EtherNet/IP card (« sicmarking.eds » available in the SIC files directory). This file contains the information so that a programmable logic controller (PLC) can send information to the e10 controller and read back information from it.

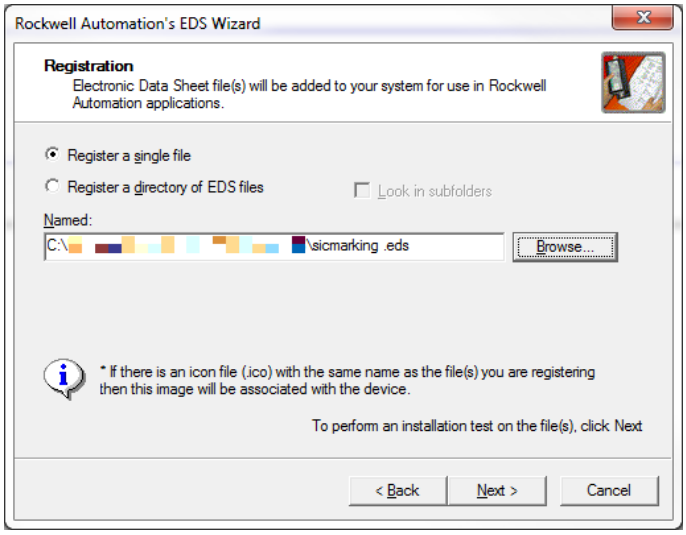
The eds file contains the definition of the e10 controller. So before adding an e10 controller in the network, the e10 definition must be extracted from the eds file and added to the list of device already recognized by the development environment. The following steps are done in the Logix Designer Studio 5000.

From the “Tools” menu, select the “EDS Hardware Installation Tool” option :

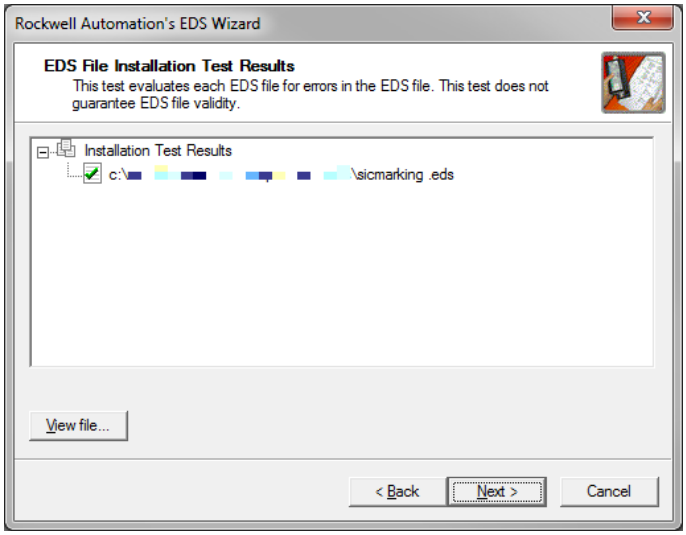


From the Rockwell Automation EDS Wizard, select the “Register an EDS file” option and click the “Next” button :

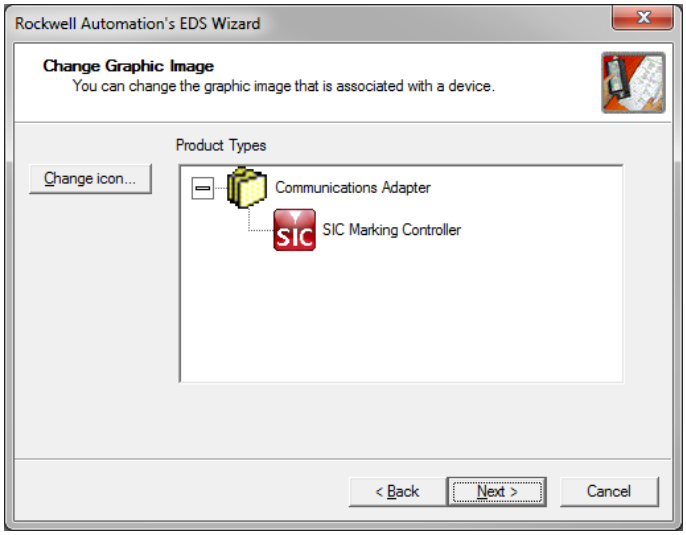
In the next step, select the “Register a single” option. Either enter directly the path of the provided eds file, or click the “Browse…” to select the file. Once the file is entered in the “Named” text box, click the “Next” button again :



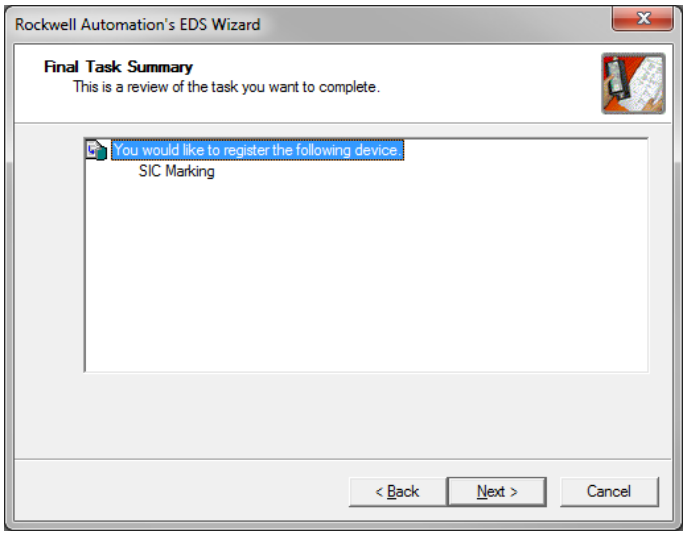
The following step in the Rockwell Automation EDS wizard is a confirmation page to ensure that you have selected the appropriate file. If you are satisfied with your choice, click the “Next” button to proceed with the extraction of the information about the e10 controller :



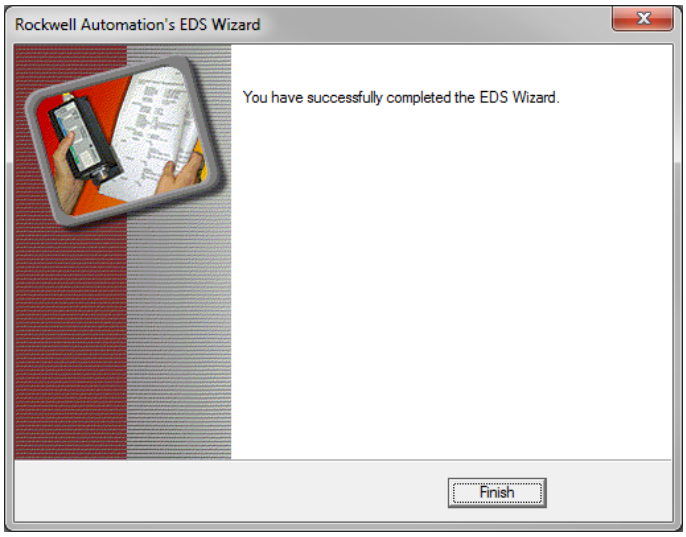
When the eds file is read by the wizard, the definition of the SIC Marking controller will be presented to the user in the next wizard’s page. Click the “Next” button to confirm the product type extraction :



The last step is to insert the device definition contained in the eds file and to add the device in the database. Again click the “Next” button to proceed with the insertion of the SIC Marking e10 controller in the database :



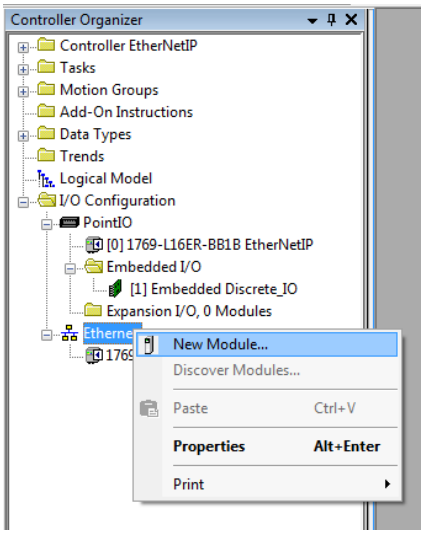
This final wizard page is to confirm the SIC Marking e10 controller definition was properly extracted from the EDS file and added to the Logix Designed devices definition. Click the “Finish” button to complete the process :



Adding the e10 controller to the Network :

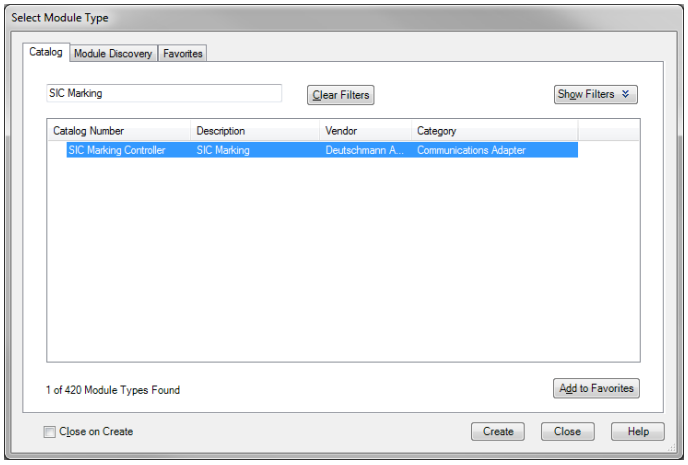
With a SIC Marking e10 controller with an EtherNet/IP card installed and the product definition added to the Logix Designer, you can now proceed with adding an e10 controller in your industrial network. The following steps are done the Logix Designer.

In the Controller Organizer view, do a right-click on the “Ethernet” branch. Select the “New Module…” to add a new device to your network.



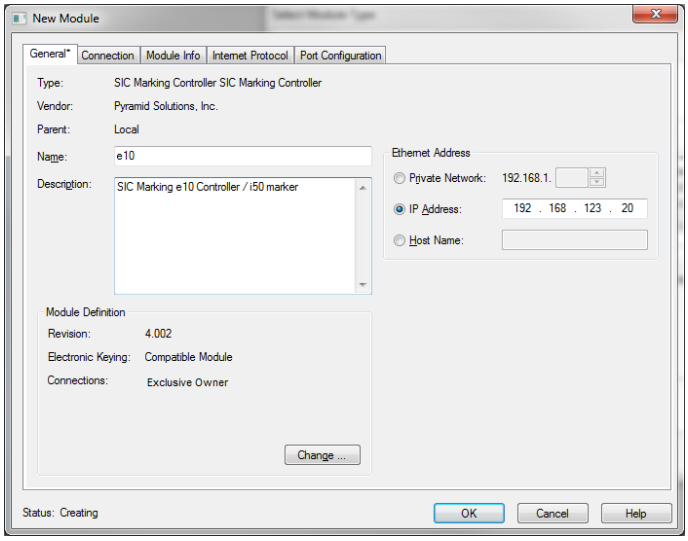
You must now select the SIC Marking device from the catalog. The easiest way is to perform a search is by searching for “SIC Marking” as the keyword. If the eds file was properly loaded (see previous section) you should be able to see the definition.

Click on the “Create” button to add a new e10 controller in your network :

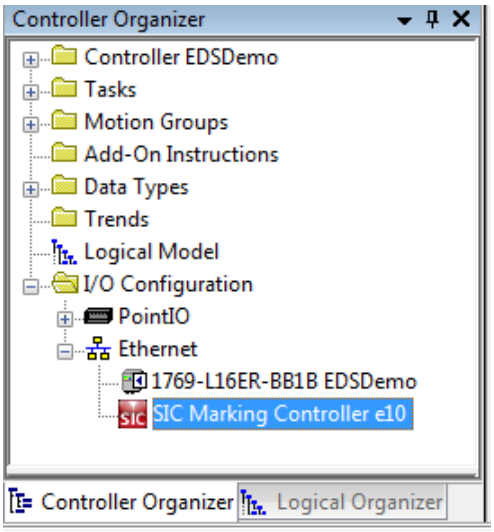


Give your new e10 controller in your network a meaningful name in the “Name” textbox. The other critical information is the Ethernet address. The information you enter in the Ethernet address must match what you have entered in the controller itself (first section of this document).

Click the “Ok” button when the appropriate information is entered :



Your new e10 controller will now appear in the Controller Organizer window of the Logix Designer. Your device is now part of your industrial network and you are now ready to send information and directive to it to perform you various marking needs :



NOTE: Depending on the version of your RSLogix Software, the connection tab may also present the option "Use Unicast Connection over EtherNet/IP. Please make sure to disable this option as the device does not support it.

SIC MARKING Headquarters

ZAC Bel Air
195 rue des Vergers
69480 POMMIERS
France
Ph.: + 33 472 548 000
info@sic-marking.com

SIC MARKING USA

3812 William Flynn Hwy
Allison Park, PA 15101
USA
Ph.: + 1 412 487 1165
info@sic-marking.com

SIC MARKING CANADA

35-2, rue De Lauzon
Boucherville Qc J4B 1E7
Canada
Ph.: +1 450-449-9133
info@sic-marking.com

SIC MARKING CHINA

No. 601, No. 4 Building, No. 258 /上海浦东新区金藏路258号4号楼601室
Jinzang Rd. Shanghai
Shanghai 201206
China
Ph: +86 (0) 21 6164 5600
info@sic-marking.cn

SIC MARKING GERMANY

Am Bruch 21 - 23
Remscheid D-42857
Germany
Tel.: +49 (0) 2191 46240-0
info@sic-marking.de

http://www.sic-marking.com