matthewsmarking.com

Thank you for choosing an *I·Mark* ink jet printer

We sincerely hope you find this manual to be informative and useful.

Comments regarding our manuals are always welcome and appreciated.

techdoc@swedot.se

I•Mark SX-32e

I•Mark SX-32e ASCII Protocol

Reference Manual

Version: 5 Issue: 1

Order number: MS795-804-06

Matthews™

BRAND SOLUTIONS

Table of Contents

Terms of Use

Contact details - Manufacturer

Sweden
Matthews Swedot
Möbelgatan 4
431 33 Mölndal
Phone: +46 (0)31 338 79 00
Fax: +46 (0)31 84 51 17

USA
Matthews Marking Products
6515 Penn Avenue
Pittsburgh, PA 15206
Phone: +1 412 665-2500
Fax: +1 412 665-2550

China
Kenuohua Matthews
Building C, Jinri Science Park
No.26 Jinyuan Road
Daxing Industrial Development Zone
Beijing 102600
Phone: +86 10 88796525
Fax: +86 10 88796526

Disclaimer

Neither Matthews Marking Products nor Matthews Swedot AB can be held responsible for any direct, indirect, specific, accidental or resultant injuries caused by a fault with the machine system or software, or by an error in the accompanying documentation. In particular, neither Matthews Marking Products nor Matthews Swedot AB can be held responsible for any program or data stored or used with Matthews' products, including the cost of recovering such programs or data.

About this Manual

This manual (Version: 5 Issue: 1) describes version 2.04 of the I•Mark ASCII remote communications protocol. For information regarding the Software version or the Print Engine version refer to "Documentation History" on page 58.

It is intended for use by anyone who needs to use the I•Mark ASCII protocol to send and receive data from a remote host to the controller.

How this Manual is Organized

- **"Introduction" on page 1**
  Here, basic information is given on the different command types available, the command build-up and introductory instructions on how this protocol is used to create messages for the SX-32e printer.

- **"Message Configuration Commands" on page 7**
  These commands are used to create messages for printing as well as the objects (such as texts and graphics) they are made up of.
- **"Printer Configuration Commands" on page 23**
  These commands are used to adapt the controller and print heads to the installation.
- **"Printer Utility Commands" on page 50**
  Printer utility commands are used to immediately return the controller to factory defaults, start and stop printout and clear the error register.

Document Conventions

Text emphasis	Use of emphasis
HELVETICA	For programming code. For example GP DATE
HELVETICA	For response from printer

The following describes the ASCII protocol as implemented in the RS-232 communication port of the SX-32e controller. The protocol is designed to be used with host computers or PLCs with ASCII capability.

Data Transfer

The default configuration for the SX-32e serial port is 19200 baud, eight data bits, one stop bit, and no flow control. This configuration works well when executing commands manually using a terminal emulator, or when the command rate is low.

When the SX-32e is connected to a control device (PC, PLC or similar), it is recommended that hardware flow control[1] is in operation at both the SX-32e and the control device, and that a fully wired RS-232 cable is used[2].

For information on changing the serial port configuration, see "Controller Configuration Properties" on page 32.

All data sent through the serial port, using the ASCII Protocol, is formatted using UTF-8.

Command Types

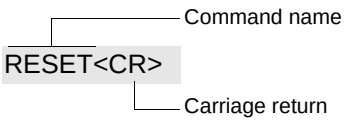
The commands for the SX-32e controller can be divided up into three basic command types:

- **Message configuration commands** - Commands which are used to read or construct messages (for printing) and the objects which they shall contain.
  See "Message Configuration Commands" on page 7.
- **Printer configuration commands** - Commands which read or write controller setup properties such as the print direction for a specific print head and communication settings.
  See "Printer Configuration Commands" on page 23.
- **Printer utility commands** - Commands which manage the controller such as the RESET command which is used to return the controller to its default settings.
  See "Printer Utility Commands" on page 50.

1. When hardware flow control is in operation it is important that the control device software reads the responses from the SX-32e. This will prevent receive buffer overflow (in the control device) thereby avoiding communication failure.
2. Hardware flow control must not be used without fully wired cables.

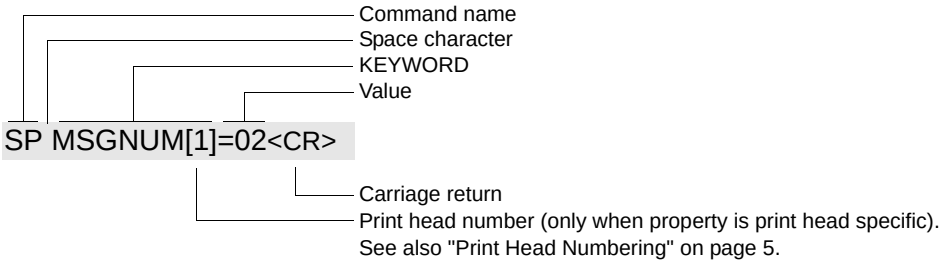
Command Formats

In its simplest form, a command can consist of only the actual command name. An example of this format is the printer utility command used to return the controller to its default settings (RESET). Note that All commands are terminated using a carriage return character (<CR>). A carriage return character has an ASCII decimal value of 13 or a hexadecimal value of 0D.



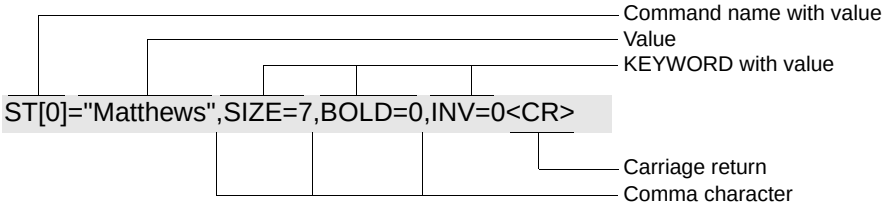
Normally, however, a command will consist of three elements: a command name, and a collection of KEYWORD's and their values. The first example below shows how the printer configuration command SP (set property) is used to set the value of a single property, namely to select message number 02 (=02) for printout at print head number 1 ([1]).

Note that a **space character** (ASCII decimal 32 or hexadecimal 20) must always follow the command name, whenever the command is directly followed by a KEYWORD (as shown below). Although it is possible to include a space character at other points, such as before or after the = symbol, for the purposes of consistency, avoid doing so.



The second example shows how the command for creating a text object (ST) is used to set the value of several properties.

Note that a comma character (ASCII decimal 44 or hexadecimal 2C) is used to separate KEYWORD settings.



- All commands which are sent to the SX-32e controller are terminated using a carriage return character (<CR>). A carriage return character has an ASCII decimal value of 13 or a hexadecimal value of 0D.
- Text data must be enclosed within quotation marks.

Receiving Data

When a command is sent to the controller, it (the controller) will reply with one of the following:

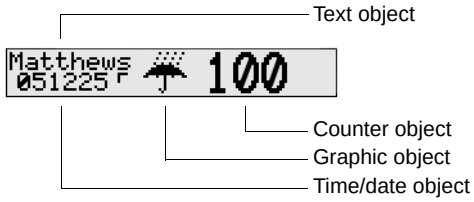
- A value - if the issued command requested the status of a property.
- "ok" - if the issued command sets a property.
- An error message, such as (INVALID PROPERTY) - if there was an error in the command.

The reply string is terminated by a carriage return character (<CR>) and a line feed character (<LF>). A carriage return character has an ASCII decimal value of 13 or a hexadecimal value of 0D. A line feed character has an ASCII decimal value of 10 or a hexidecimal value of 0A.

Creating and Printing Messages

A message is a collection of objects which shall be printed at the same time. Each object is created separately and is linked to a message when the message is created. This is done using the SM[n] (set message) command. See "GM, SM (Get and Set Message)" on page 20.

Below is an example of a message (as shown in the SX-32e display) containing some objects:



The sequence of events one should follow when creating and printing messages is as follows:

1) If necessary, create some objects.
2) Create a message containing one or more objects.
3) Print the message.

Creating Objects

The first step in creating a message for printing is to create the objects that make up the message. The different object types are:

Text - Counter - Graphic - Barcode -Time/Date - Shift code

In the example below a text object is created (ST) consisting of the text Matthews (="Matthews"). It will be stored in the text memory location 1 ([1]):

ASCII Example command: ST[1]="Matthews"<CR>

Response: *ok<CR><LF>*

Creating Messages

The second step in creating a message for printing is to create the message using the message objects previously created. A message can be created with or without a name depending on whether or not the message shall be made available in the controller via the user interface (display).

Examples of the commands that can be used to build a message with the text field created above are shown below.

In the first example, a message is created in message memory location 2 ([2]) named Msg1 (="Msg1"). This message contains the text field called 1 ((T[1])) which is positioned at x:y coordinates 0:0[1] in the message (@0:0). Both the size and boldness of the text “Matthews” have also been changed from the default. Please note the use of double brackets.

ASCII Example command: SM[2]=((T[1]@0:0,FONT=0,SIZE=16,BOLD=2)),NAME="Msg1"<CR>

Response: *ok<CR><LF>*

Note: when creating messages you should always include the KEYWORD ‘FONT’ (FONT=0), as shown above. The KEYWORD ‘FONT’ act as a trigger to ensure that attributes set in the message construct, as shown above, will be used. This is important to remember, since the actual text object can have its attribute BOLD set to say BOLD=1, whilst in the message construct it is set to BOLD=2 (refer to "GM, SM (Get and Set Message)" on page 20 for a further explanation).

The second example shows a message created in message memory location 2 ([2]) containing the text field called 1 ((T[1])) which is positioned at x:y coordinates 0:0 in the message (@0:0). This message has not been named and will therefore not be accessible via the user interface (display).

ASCII Example command: SM[2]=((T[1]@0:0))<CR>

Response: *ok<CR><LF>*

Selecting a Message for Printout

To print the message which was created above, it must be selected for the print head that will be printing it. An example of the command for selecting a message for printout is shown below.

1. The x:y coordinates refer to the left-to-right (x) and top-to-bottom (y) position of a single dot in the message area, e.g. 0:0 refers to the leftmost, topmost dot respectively.

In this example the message in memory location 2 (=2) has been selected for printout at print head number 0 ([0]). See also "Print Head Numbering" on page 5, and "ENABLE" on page 51.

ASCII Example command: SP MSGNUM[0]=2<CR>

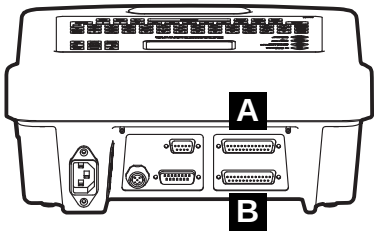
Response: *ok<CR><LF>*

The selected message will be printed the next time a printout is activated.

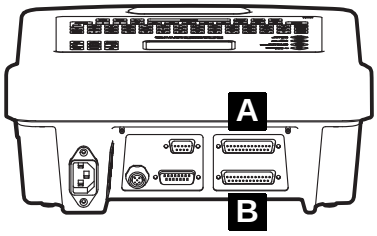
Print Head Numbering

This table shows the different combinations of print heads which can be connected to the SX-32e controller. For each combination, information is given according to the following:

Connector ID - Where to connect the print head on the controller.
Head ID Protocol - How the head is referenced using this protocol.
Head ID GUI - How the head is referenced via the controller's user interface.
Setting Protocol - How the head combination is set at the controller, using this protocol.
Setting GUI - How the head combination is set at the controller, using the user interface.



Head combination	Connector ID	Head ID		Setting	
		Protocol	GUI	Protocol[a]	GUI
7 valve head	A	0	1	1	1x7
7 valve head	A	0	1	2	2x7
7 valve head	B	1	2		
7 valve head	A	0	1	3	4x7
7 valve head		1	2		
7 valve head	B	2	3		
7 valve head		3	4		
16 valve head	A	0	1	4	1x16



Head combination	Connector ID	Head ID		Setting	
		Protocol	GUI	Protocol[a]	GUI
16 valve head	A	0	1	5	2x16
16 valve head	B	2	2		
32 valve head	A	0	1	6	1x32

a. See "CONFIG" on page 35.

Message Configuration Commands

The SX-32e controller stores message objects in a set of tables. Each table has its own Get/Set commands and includes a numerical index to describe a specific message object. The numerical index must be contained within square brackets, with no space following the Get or Set command.

Command	Long Form	Number of Entries	Entry Size	Description/Page
GT[n] ST[n]	GET_TEXT[n] SET_TEXT[n]	500 (n=0-499)	64 chars	Reads and writes text message elements, including their configuration. See page 9.
GG[n] SG[n]	GET_GRAPHIC[n] SET_GRAPHIC[n]	32 (n=0-31)	2000 bytes	Reads and writes a graphic message using the Matthews format for pixel definition. See page 10.
GC[n] SC[n]	GET_COUNTER[n] SET_COUNTER[n]	32 (n=0-31)	10 bytes	Reads and writes a counter message, including configuration of the message. See page 12.
GR[n] SR[n]	GET_REALTIMECLOCK[n] SET_REALTIMECLOCK[n]	32 (n=0-31)	16 chars	Reads and writes a time or date message. See page 15.
GB[n] SB[n]	GET_BARCODE[n] SET_BARCODE[n]	32 (n=0-31)	8 objects	Reads and writes bar code information, to be printed in bar code format. See page 17.
GS[n] SS[n]	GET_SHIFTCODE[n] SET_SHIFTCODE[n]	8 (n=0-7)	384 bytes	8 UTF16 characters per code. Up to 24 codes per construct. See page 19.
GM[n] SM[n]	GET_MESSAGE[n] SET_MESSAGE[n]	300 (n=0-299)[a]	16 objects	Used to define a complete message using the elements in the other tables. See page 20.

a. Message 299 is reserved for the special message [Blank]. Selecting message 299 will produce a blank print-out. The [Blank] message is used to ensure that print heads which are not being tested, do not print.

The command format for message elements allows for all the keywords for a single message element to be described in a single command, with each KEYWORD delimited by a comma[1]. For example, a text object can have a different font size, be bold, or can be printed inverted.[2]

ASCII Example:	ST[1]=”Matthews”,SIZE=16,BOLD=2,INV=1,NAME=”name”<CR>
Response:	*ok<CR><LF>*

In a SET_MESSAGE command, only those KEYWORDS the user wishes to change need to be included in the command. KEYWORDS not included in the command will assume their default values.

If an invalid value for a KEYWORD is included, or an incorrect KEYWORD is used, the response will contain an error message.

Message Constructs

Name	Data Type	Description
Message Name	UTF-16[16]	A string that is used to identify the message.
Message Construct	ObjectType[16]	Format defined below.

A message construct is an array of Message Objects. The number of objects in a message construct is variable, depending on the application, with a maximum of sixteen (16) objects. The Message Construct must not exceed the height of the connected print head(s), nor the length based on the controller option.

ObjectType definition		
Table	Unsigned Byte	The table in the object model that contains the message object.
Record	Unsigned Word	The record within the table of a specific message object.
XLocation	Unsigned Word	The beginning X location (horizontal pixel) for the object.
YLocation	Unsigned Word	The beginning Y location (vertical pixel) for the object.
Font	Unsigned Byte	Which font type to use. (255 for attributes defined in object, binary protocol only)

1. A space character following the comma character is optional.
2. Refer also to ’GM, SM (Get and Set Message)’ on page 20

ObjectType definition		
Size	Unsigned Byte	The font size to use.
Bold	Unsigned Byte	The number of times to repeat a print column (0-9).
XSP	Unsigned Byte	The number of additional spaces to add between columns (0-7)
Mode	Boolean[4]	Bit 0 Print Direction (REV) (0 - Normal, 1 - Reversed) Bit 1 Print Inverted (INV) (0 - Normal, 1 - Inverted) Bit 2 Print Rotate (ROT) (0-Normal, 1-Rotate 90° anti-clockwise) Bit 3 Print Negative (NEG) (0 - Normal, 1 - Negative)

GT, ST (Get and Set Text)

Reads and writes text message elements, including their configuration.

Format: ST[n]="*text string*",KEYWORD=value,KEYWORD=value,….

Text			
KEYWORD	**Data Type**	**Range**	**Description**
NAME	UTF-16[16]		A string that is used to identify the text object.
text string	UTF-16[50]		The string value.
FONT	Unsigned Byte	0	Identification number for various fonts. Currently, only font 0 is supported.
SIZE	Unsigned Byte	5, 7, 9, 14, 16, 21, 32	The font size in number of dots.
BOLD	Unsigned Byte	0...9	Number of times to repeat a column.
XSP	Unsigned Byte	0...7	Number of additional columns to space between characters.
REV	Boolean	0, 1	Reverses the print direction (mirror image).
INV	Boolean	0, 1	The message will be printed inverted (upside down).
ROT	Boolean	0, 1	The characters will be rotated 90° anti-clockwise.
NEG	Boolean	0, 1	The message will be printed using a negative format.

Command:	ST[0]="Matthews",NAME=”MSG 01”,REV=1<CR>
Response:	*ok<CR><LF>*

Where:
- ST[0]="Matthews" - Set text number 0 to contain the text “Matthews”
- NAME=”MSG 01” - The string used to identify the text object
- REV=1 - Set the text to print in reverse (mirror image)

GG, SG (Get and Set Graphic)

Reads and writes up to 32 graphic objects (0–31), including their configuration.
The “graphic format” is defined as an array of bytes using the Matthews graphics format.
A total of 2000 bytes of information gives:
2000 columns of 8 dot high graphics
1000 columns of 16 dot high graphics
500 columns of 32 dot high graphics

Graphics 0-24 are predefined and shipped with the unit, however they can be overwritten via the serial interface. See "Graphic Objects in Memory" on page 11 for more information.

Format: SG[n]=”*bitmap data”*,KEYWORD=value,KEYWORD=value,…

Graphic			
KEYWORD	**Data Type**	**Range**	**Description**
NAME	UTF-16[16]		A string that is used to identify the graphic object.
SIZE	Unsigned Byte	1…32	The graphic height in number of dots.
bitmap data	Byte[2000]	00...FF	Bitmap data in Matthews pixel format.
BOLD	Unsigned Byte	0...9	Number of times to repeat a column.
REV	Boolean	0, 1	Reverses the print direction (mirror image).
INV	Boolean	0, 1	The message will be printed inverted (upside down).
NEG	Boolean	0, 1	The message will be printed using a negative format.

An example of the Get Graphic (GG) command and the Set Graphic (SG) command are provided below (See also Creating Graphics on page 53):

Command:	GG[0]<CR>
Response:	*g[0]="FFFFFFFFC003C203C603CFFBDFFBCFFBC603C203C003C003C203C603CFFBDFFBCFFBC603C203C003FFFFFFFF",name="Up",size=16,bold=0,rev=0,inv=0,neg=0<CR><LF>*

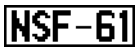
Command:	SG[31]=”FFFFF”,BOLD=1,SIZE=12<CR>
Response:	*ok<CR><LF>*

Graphic Objects in Memory

The following table shows the pre-installed graphic objects and how they are identified in memory.

Number (serial interface)	Name (user and serial interface)	Appearance
0	Up	
1	Fragile	
2	Wet	
3	Knife	
4	Cold	
5	Recyc1	
6	Recyc2	
7	CE	
8	Poison	
9	Heat	
10	Crown	
11	PE-LD	
12	Sun	
13	CST32	

(Sheet 1 of 2)

Number (serial interface)	Name (user and serial interface)	Appearance
14	EUR32	
15	NF32	
16	CSA16	
17	NSF9	
18	NSF6116	
19	NSFDWV16	
20	NSFPW16	
21	R9	
22	ULA16	
23	ULR16	
24	UPCR16	

(Sheet 2 of 2)

GC, SC (Get and Set Counter)

Reads and writes counter objects, including their configuration.

Format: SC[n] KEYWORD=*count*,KEYWORD=value,KEYWORD=value,…

count: 10 character ASCII string that is the current value of the counter

Counter			
KEYWORD	**Data Type**	**Range**	**Description**
NAME	UTF-16[16]		A string that is used to identify the counter object.
count	ASCII[10]		Current value
FMT	ASCII[10]		Each character of the format string represents how the corresponding digit in the counter value is formatted: 9 Numeric (0-9) f Hex lower case (0-9, a-f) F Hex upper case (0-9, A-F) z Lower case alpha (a-z) Z Upper case alpha (A-Z)

(Sheet 1 of 3)

Counter			
KEYWORD	**Data Type**	**Range**	**Description**
FIRST	ASCII[10]		The beginning count value.
LAST	ASCII[10]		The ending count value. When reached, the count will roll over to the Start Value. If the End Value is smaller than the Start Value, the counter will decrement instead of incrementing.
REP	Unsigned Byte	0...255	Number of times to repeat the count before incrementing.
SKIP	Unsigned Byte	0...255	Number of prints to skip before incrementing the counter.
UPDMODE	Unsigned Byte	0...4	0 - Use global update mode – counters are disabled **(default)** 1 - Update at end-of-trigger 2 - Update at end-of-mark 3 - Update on counter cascade 4 - Update on auxiliary input
CASC	Unsigned Byte	0...31	If UPDMODE=3, the counter (b) that is used to increment the actual counter (a). The actual counter (a) is incremented whenever counter (b) rolls over.
OPT	Boolean[3]	0...7	Bit 0 - Skip I for Z formats Bit 1 - Skip O for Z formats Bit 2 - Use spaces for left most padding
RSTMODE	Unsigned Byte	0...4	0 - reset on rollover only 1 - reset on rollover and end of trigger 2 - reserved 3 - reserved 4 - reset on rollover and auxiliary input
FONT	Unsigned Byte	0	Identification number for various fonts. Currently, only font 0 is supported.
SIZE	Unsigned Byte	0, 5, 7, 9, 14, 16, 21, 32	The font size in number of dots.
BOLD	Unsigned Byte	0...9	Number of times to repeat a column.
XSP	Unsigned Byte	0...7	Number of additional columns to space between characters.
REV	Boolean	0, 1	Reverses the print direction (mirror image).
INV	Boolean	0, 1	The message will be printed inverted (upside down).
ROT	Boolean	0, 1	The characters will be rotated 90° anti-clockwise.

(Sheet 2 of 3)

Counter			
KEYWORD	**Data Type**	**Range**	**Description**
NEG	Boolean	0, 1	The message will be printed using a negative format.

(Sheet 3 of 3)

An example of a Set Counter (SC) command is provided below:

Command: SC[0]="09",NAME="S",FMT="99",FIRST="00",LAST="99"<CR>

Response: *ok<CR><LF>*

Where:
- SC[0]="09" - Set the current value of counter number 0 to 09
- NAME=”S” - The string used to identify the counter object
- FMT="99" - Set the counter format to 99
- FIRST="00" - Set the beginning count value to 00
- LAST="99" - Set the highest count value to 99

Examples of other counter formats:

Format	Value [printed]	Next value [printed]	Third value [printed]
999	998 [998]	999 [999]	000 [000]
FF	FE [FE]	FF [FF]	00 [00]
Z99	Z98 [Z98]	Z99 [Z99]	A00 [A00]
999 (space padding on)	008 [_ _8] (where:_ represents a space)	009 [_ _9]	010 [_10]

GR, SR (Get and Set Real-Time Clock)

Reads and writes real-time clock objects, including their configuration.

Format: SR[n] KEYWORD=value,KEYWORD=value,…

Real-time clock			
KEYWORD	**Data Type**	**Range**	**Description**
NAME	UTF-16[16]		A string that is used to identify the RTC object.
time	ASCII[2048]		The current formatted RTC string, returned following a GR command.
FMT	ASCII[16]		Each character in the format string represents the format for the real-time clock value, based on the following format characters: y Year without century (2-digits) Y Year with century (4-digits) e Year name (4 chars) m Month (01-12) d Day of Month (01-31) c Week of Year name (2 chars) b Month Name (16 char) a Day Name (16 char) j Day of Year (001-366) J Day of Year (000-365) w Week of Year(1-53) H Hour using a 24-hour clock I Hour using a 12-hour clock M Minutes p AM or PM placeholder - Date Separator (see DSEP) : Time Separator (see TSEP) (See "Date and time formats" on page 17).
DSEP	UCS-4		Sets the separator to be used between elements of a date message.
TSEP	UCS-4		Sets the separator to be used between elements of a time message.
DFWD	Unsigned Word	0…7305	The number of days forward from the current date. Used for expiration dates.
MFWD	Unsigned Word	0...1440	The number of minutes forward from the current time/date.
FONT	Unsigned Byte	0	Identification number for various fonts. Currently only one font is supported.
SIZE	Unsigned Byte	5, 7, 9, 14, 16, 21, 32	The height of the font in dots.

(Sheet 1 of 2)

Real-time clock			
KEYWORD	**Data Type**	**Range**	**Description**
BOLD	Unsigned Byte	0...9	Number of times to repeat a column.
XSP	Unsigned Byte	0…7	Number of additional columns to space between characters.
REV	Boolean	0, 1	Reverses the print direction (Mirror image).
INV	Boolean	0, 1	The real-time clock will be printed inverted (upside-down).
ROT	Boolean	0, 1	The characters will be rotated 90° anti-clockwise.
NEG	Boolean	0, 1	The real-time clock will be printed using a negative format.

(Sheet 2 of 2)

An example of a Set Real-Time Clock command is provided below:

Command: SR[0] NAME="Msg 01",FMT="YjHp",REV=1<CR>

Where:
SR[0] - Set real-time clock number 0
NAME="Msg 01" - The string used to identify the RTC object
FMT="YjHP" - The date and time format (See page 17)
REV=1 - Set the print direction to reverse (mirror image)

Date and time formats

Format	Format string
MM~DD~YY	m-d-y
MM~DD~YYYY	m-d-Y
mmm~DD~YYYY	b-d-Y
mmm~DD~YY	b-m-y
DD~MM~YY	d-m-y
DD~MM~YYYY	d-m-Y
DD~mmm~YYYY	d-b-Y
YY~MM~DD	y-m-d
DD~mmm~YY	d-b-y
YYYY~MM~DD	Y-m-d
YYYY~mmm~DD	Y-b-d
DD	d
MM	m
mmm	b
Week (01-53)	w
Week name	c
YY	y
YYYY	Y
Year name	e
JJJ	j
JJJ (European)	J
ddd	a
hh¤mm (24 hour)	H¤M
hh¤mmXM	I¤Mp
hh	I
mm	M
XM (AM or PM)	p

Character key
~: Place holder for date separator
¤: Place holder for time separator
X: A or P (in AM and PM)

GB, SB (Get and Set Barcode)

Reads and writes barcode objects, including their configuration.

Format: SB[n]="*data*"&(Object, Object),KEYWORD=value,KEYWORD=value,…

data: the value to be printed as a bar code (32 character string)

Barcode			
KEYWORD	**Data Type**	**Range**	**Description**
NAME	UTF-16 [16]		A string that is used to identify the bar code object.
data	UTF-16 [32]		Data to be printed in the bar code
OBJECTS	ObjectType[8]		An array of up to eight message objects that are to be encoded in the barcode, where [n] is the actual object. t[n] - Text objects c[n] - Counter objects r[n] - RealTime Clock objects s[n] - Shiftcode objects

(Sheet 1 of 3)

Barcode			
KEYWORD	**Data Type**	**Range**	**Description**
CODE	Unsigned Byte	0…9, 100	Which bar code format to use for the message format. Valid entries and the associated bar code format are: 0 Codabar 1 Interleaved 2 of 5 2 Code 39 3 EAN 8 4 EAN 13 5 UPC A 6 Code 128 B (dynamic objects) 7 Code 128 C (dynamic objects) 8 ITF14 (dynamic objects) 9 SCC14 (dynamic objects) 100 ECC200 Data Matrix (dynamic obj.)
OPT	Boolean	0, 1	Bit 0 - When enabled, human readable text is included when printing the bar code. Font type cannot be specified. Not available with data matrix codes. Bit 1 -When enabled, adds bearer bars on ITF bar codes. Bit 2 - When enabled uses 2x2 pixels in data matrices with 16 rows or less.
SIZE	Unsigned Byte	0…32	For codes 0-9, the height in number of dots. For code 100, the Data Matrix size as follows: 0 Reserved 1 12 x12 2 Reserved 3 16 x 16 4 Reserved 5 20 x 20 6 Reserved 7 24 x 24 8 Reserved 9 32 x 32 10 8 x 18 11 8 x 32 12 12 x 26 13 12 x 36 14 16 x 36 15 16 x 48
BOLD	Unsigned Byte	0...9	Number of times to repeat a column.
REV	Boolean	0, 1	Reverses the print direction (Mirror image).

(Sheet 2 of 3)

Barcode			
KEYWORD	**Data Type**	**Range**	**Description**
INV	Boolean	0, 1	The message will be printed inverted (upside-down).
NEG	Boolean	0, 1	The message will be printed using a negative format.

(Sheet 3 of 3)

An example of a Set Barcode command is provided below:

Command: SB[1]="12345"&(t[n],c[n]),CODE=1<CR>

Where:
- SB[1]="12345"&(t[n],c[n]) - Set the barcode number 1 to encode the data: 12345, plus the Text object t[n] and the Counter Object c[n].
- CODE=1 - Set the barcode type to Interleaved 2 of 5

GS, SS (Get and Set Shiftcode)

Reads and writes shift code objects, including their configuration.

A shift code is a looping, 24 hour sequence consisting of up to 24 periods, each period with a corresponding code of up to eight characters. Each code is only printed during its designated period. See the example below.

Note that all codes must be the same length (consist of the same number of characters).

Format: SS[n]=*value*,KEYWORD=value,KEYWORD=value,…

Shiftcode data value: Up to 24 totalling 96 characters

Shiftcode			
KEYWORD	**Data Type**	**Range**	**Description**
NAME	UTF16[16]		A string that is used to identify the Shiftcode object.
SHIFT	ShiftObject[24]		Shiftcode construct in the following format: • Start • Uns Word • Minutes past midnight • Code • UTF-16 [8] • Text to print for shift.
FONT	Unsigned Byte	0	Identification number for various fonts. Currently, only font 0 is supported.
SIZE	Unsigned Byte	5, 7, 9, 14, 16, 21, 32	The font size in number of dots.

(Sheet 1 of 2)

Shiftcode			
KEYWORD	**Data Type**	**Range**	**Description**
BOLD	Unsigned Byte	0...9	Number of times to repeat a column.
XSP	Unsigned Byte	0...7	Number of additional columns to space between characters.
REV	Boolean	0, 1	Reverses the print direction (mirror image).
INV	Boolean	0, 1	The message will be printed inverted (upside down).
ROT	Boolean	0, 1	The characters will be rotated 90° anti-clockwise.
NEG	Boolean	0, 1	The message will be printed using a negative format.

(Sheet 2 of 2)

An example of a Set Shiftcode command: In this example, a shift code has been set up to print the letters AM from 00:00, and PM from 12:00, until 00:00, when AM shall be printed again and so on.

Command: SS[0]=("AM"@0000,"PM"@1200),NAME=”AP”<CR>

Where: SS[0] - Set shift code number 0
NAME=”AP” - The string used to identify the shift code object

GM, SM (Get and Set Message)

The Get Message (GM) and Set Message (SM) commands are used to assemble different message elements into a single printable message. There can be up to 24 elements in each printable message.

Format: SM[n]=((a[n]@x:y,FONT=f,SIZE=s,BOLD=b,XSP=x,MODE=m),(b[n]@x:y, FONT=f,.....)),name=”name”

Where: a[n], b[n] - is one of the following message objects:
T - Text
G - Graphic
B - Barcode
C - Counter
R - Real-time Clock
S - Shift Code

When creating messages you should always include the KEYWORD ‘FONT’ (FONT=0). The KEYWORD ‘FONT’ acts as a trigger to ensure that attributes set in the message construct, as shown above, will be used. This is important to remember as an actual object can have its attribute BOLD set to say BOLD=1, whilst in the message construct it is set to BOLD=2. In addition, the GM (get message) statement will not return information regarding attributes, if the KEYWORD ‘FONT’ is excluded. Provided below are examples that highlight these differences.

In the examples provided, the set message is firstly written without and then with the ‘FONT’ KEYWORD. The response from the GM[1] command is then provided, showing the differences in the attributes used and returned.

‘FONT’ KEYWORD not included

SM: SM[1]=((T[0]@0:0,MODE=6,BOLD=2)),NAME="Msg1"<CR>

GM: m[1]=((t[0]@0:0)),name="Msg1"

Since, in the above example, the statement FONT=0 has not been written, the attributes MODE=6 and BOLD=2 are neither displayed in the response from the GM statement or actual.

‘FONT’ KEYWORD Included

SM: SM[1]=((T[0]@0:0,FONT=0,MODE=6,BOLD=2)),NAME="Msg1"<CR>

GM: m[1]=((t[0]@0:0,font=0,size=0,bold=2,xsp=0,mode=6)),name="Msg1"

In the above example the FONT=0 statement has been written and the attributes have been triggered, so that the text object T[1] will be printed with the properties BOLD=2 and MODE=6. The attribute BOLD of the actual object will not be used.

In addition, attributes that are set as part of an actual object, such as BOLD, will not be reflected in the GUI. It is only when they are set as part of the message construct, that they will be reflected in the GUI.

x:y is the beginning location of the message element in terms of x and y pixel (horizontal and vertical position respectively where 0 equals far left or top of message area).

Font=font type
: Identification number for various fonts. Currently, only font 0 is supported.

size=object size
: The font size in number of dots. 5, 7, 9, 14, 16, 21, 32 heights supported.

Bold=repeated columns
: Number of times to repeat a column (0–9).

xsp=extra spaces
: Number of additional columns to space between characters (0–7).

Mode=object mode <=15

Value	Description	Bit value
Bit 0	reverse (REV)	0 normal, 1 reverse
Bit 1	invert (INV)	0 normal, 1 invert
Bit 2	rotate (ROT)	0 normal, 1 rotate
Bit 3	negate (NEG)	0 normal, 1 negate

Name=message name

A 16 character UTF-16 string that is used to identify the message.

There are no other parameters to be set in the SET_MESSAGE command, since all specific information about each message object is defined as part of each object.

An example of a Set Message command:

Command: SM[1]=((T[0]@0:0,FONT=0,MODE=6),(G[1]@50:0,FONT=0,BOLD=2)),NAME="Msg1"<CR>

Where:

SM[1]	- Set message for location 1
T[0]@0:0	- Position text number 0 at 0:0
FONT=0	- This default setting should always be included.
MODE=6	- The text will be displayed rotated
G[1]@50:0	- Position graphic number 1 at 50:0
FONT=0	- This default setting should always be included.
BOLD=2	- The number of times to repeat a column
NAME="Msg1"	- The 16 character Unicode string, used to identify the message

Printer Configuration Commands

Printer configuration commands are Get Property (GP) and Set Property (SP) commands which use the following groups of properties to read and set the printer configuration related settings:

- **Controller information properties** - These properties are used to get information about the controller.
  See "Controller Information Properties" on page 23.
- **Controller configuration properties** - These properties are used to adapt the controller to the installation.
  See "Controller Configuration Properties" on page 32.
- **Print head configuration properties** - These properties are used to configure how individual print heads print.
  See "Print Head Configuration Properties" on page 45.

Reading a Property

Reading a property is accomplished by using the Get Property (GP) command, followed by the relevant KEYWORD. The value of the property will be returned by the controller in the format KEYWORD=data.

ASCII Example: GP STATUS<CR>

Response: *STATUS=0000000000000010<CR><LF>*

Writing a Property

Writing a property is achieved using the Set Property (SP) command, followed by KEYWORD=data. The controller will return the characters *ok* if the message is accepted.

ASCII Example: SP TIME=“103551”<CR>

Response: *ok<CR><LF>*

Command Error

If a command causes an error, the controller will return an error message.

ASCII Example: SP TIME=10x551<CR>

Response: *Invalid property value<CR><LF>*

Controller Information Properties

The following KEYWORD properties are used in the GP and SP commands to get information about the controller or set the properties.

CATLIST

Returns the controller catalogue list.

Get/Set: GET only

Data type: UTF8[32]

Range:

Default:

ASCII Example:	GP CATLIST<CR>
Response:	*CATLIST="SX"<CR><LF>*

COUNTRY

Sets the unit of measure where 0 is imperial and 1 is metric.

Get/Set:	GET	SET
Data type:	Boolean	
Range:	0,1	
Default:	1	
ASCII Example:	GP COUNTRY<CR>	SP COUNTRY=0<CR>
Response:	*COUNTRY=1<CR><LF>*	*ok<CR><LF>*

LANGUAGE

Sets the display language for the GUI as provided below.

Get/Set:	GET	SET
Data Type:	Unsigned Byte	
Range:	0...24	
Default:	0	
ASCII Example:	GP LANGUAGE<CR>	SP LANGUAGE=2<CR>
Response:	*LANGUAGE=1<CR><LF>*	*ok<CR><LF>*

List of available languages:

Value	Language
0	Undefined (default)
1	English
2	Swedish
3	German
4	French
5	Spanish
6	Italian
7	Dutch
8	Czech
9	Polish
10	Turkish
11	Russian

(Sheet 1 of 2)

Value	Language
12	Japanese
13	Greek
14	Romanian
15	Persian
16	Arabic
17	Hebrew
18	Danish

(Sheet 2 of 2)

BRIGHTNESS

Used by GUI for the display's brightness intensity level.

Get/Set:	GET	SET
Data Type:	Unsigned Byte	
Range:	0-255	
Default:	128	
ASCII Example:	GP BRIGHT<CR>	SP BRIGHT=200<CR>
Response:	*BRIGHT=1<CR><LF>*	*ok<CR><LF>*

LEVEL

Used by the GUI for setting the user level, as provided below.

Get/Set:	GET	SET
Data Type:	Unsigned Byte	
Range:	0-3	
Default:	2	
ASCII Example:	GP LEVEL<CR>	SP LEVEL=3<CR>
Response:	*LEVEL=2<CR><LF>*	*ok<CR><LF>*

List of user levels:

Value	User level
0	Operator
1	Foreman
2	Full access
3	Obsolete (not used)

DATE

Sets the current date, using the format: DDMMYYYY.

Get/Set:	GET	SET
Data type:	Unsigned Byte[4]	
Range:	DDMMYYYY	
Default:		
ASCI Example:	GP DATE<CR>	SP DATE="08042010"
Response:	*DATE="08042010"*<CR><LF>	*ok<CR><LF>*

FAULT

Returns print fault information according to the bit table below. All errors listed are classified as FAILURES for the purpose of indication.

Get/Set:	GET only
Data type:	Boolean[32] - transmitted as a binary character string.
Range:	00000000h - FFFFFFFFh
Default:	
ASCII Example:	GP FAULT<CR>
Response:	*FAULT=00000000000000000000000000000000<CT><LF>*

Print fault bits:

Bit[a]	Description
0-3	Not assigned
4	Serial power missing
5	Over temp (valve drive)
6-31	Not assigned

a. Bits are read from right to left, i.e., bit 0 refers to the rightmost bit.

HWDATE

Returns the date of manufacture in the format MMYY.

Get/Set: GET only

Data type: UTF8[4]

Range: MMYY

Default:

ASCII Example:	GP HWDATE<CR>
Response:	*HWDATE="1106"<CR><LF>*

HWVERS

Returns the hardware version of the controller, where aaa is the assembly type, bbbbb is the unique assembly number and cc is the revision.

Get/Set: GET only

Data type: UTF8[12]

Range: aaa-bbbbb-cc

Default:

ASCII Example:	GP HWVERS<CR>
Response:	*HWVERS="175-00137-00"<CR><LF>*

PRODUCT

Returns the name of the product.

Get/Set:	GET only
Data type:	UTF8[32]
Range:	
Default:	I-Mark SX Controller

ASCII Example:	GP PRODUCT<CR>
Response:	*PRODUCT="I-Mark SX Controller"<CR><LF>*

SERNUM

Unique internal serial number, assigned during manufacture (not the product number).

Get/Set:	GET only
Data type:	Unsigned Long
Range:	$0 - 2^{32} - 1$
Default:	

ASCII Example:	GP SERNUM<CR>
Response:	*SERNUM=678123<CR><LF>*

STATUS

Returns the status of the controller according to the bit table below.

Get/Set:	GET only
Data type:	Boolean[16] - transmitted as a string of binary characters
Range:	0000000000000000b - 1111111111111111b
Default:	

ASCII Example:	GP STATUS<CR>
Response:	*STATUS=0000000000000010<CR><LF>*

Controller status bits:

Bit	Hex Value	Description
0	0001h	Valve set to Cleaner
1	0002h	Printer Enabled
2	0004h	
3	0008h	

(Sheet 1 of 2)

Bit	Hex Value	Description
4	0010h	Print Cycle On
5	0020h	Printer Running
6	0040h	Printer Warning (see WARN)
7	0080h	Printer Failure (see FAULT)
8	0100h	
9	0200h	
10	0400h	
11	0800h	
12	1000h	
13	2000h	
14	4000h	
15	8000h	

(Sheet 2 of 2)

SWDATE

Returns the firmware (software) compilation date in the format MMYY.

Get/Set: GET only

Data type: UTF8[4]

Range: MMYY

Default:

ASCII Example:	GP SWDATE<CR>
Response:	*SWDATE="0806"<CR><LF>*

SWVERS

Returns the software version of the controller. Where aaa is 173, b is the major release and c is the minor release .

Get/Set: GET only

Data type: UTF8[12]

Range: aaa.b.c

Default:

ASCII Example:	GP SWVERS<CR>
Response:	*SWVERS="173.2.1"<CR><LF>*

TIME

Sets the current time, using the format HHMMSS.

Get/Set:	GET	SET
Data type:	Unsigned Byte[3]	
Range:	HHMMSS	
Default:		
ASCII Example:	GP TIME<CR>	SP TIME="134200"<CR>
Response:	*TIME="134200"<CR><LF>*	*ok<CR><LF>*

WARN

Returns print warning information according to the bit table below. All errors listed are classified as WARNINGS for the purpose of indication.

Get/Set:	GET only
Data type:	Boolean[32] - transmitted as a string of binary characters.
Range:	00000000h - FFFFFFFFh
Default:	
ASCII Example:	GP WARN<CR>
Response:	*WARN=00000000000000000000101100000000<CR><LF>*

Print warning bits:

Bit	Description
0	RTC Battery (0=Good, 1=Low)
1-7	not assigned
8	Ink Pressure (0=Good, 1=Low)
9-31	not assigned

VENDOR

Returns the name of the manufacturer.

Get/Set:	GET only
Data type:	UTF8[32]
Range:	
Default:	MATTHEWS
ASCII Example:	GP VENDOR<CR>
Response:	*VENDOR="Matthews"<CR><LF>*

ZERO

Sets the format of the zero character, either without a slash O or with a slash Ø

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0, 1	
Default:	0 (no slash)	
ASCI Example:	GP ZERO<CR>	SP ZERO=*0*<CR>
Response:	*ZERO=0*<CR><LF>	*ok<CR><LF>*

Controller Configuration Properties

The following property KEYWORDS are used in the GP and SP commands to configure the controller for the system being implemented.

KEYWORD	Description	See page
ADDR	Sets address for external serial interface	33
AUXIN	Returns auxiliary input states.	34
BAUDRATE	Sets the baud rate of the auxiliary serial port.	34
COLSPAC	Sets the distance between columns of dots.	35
CONFIG	Sets the print head configuration.	35
DOTSIZE	Set the size of the dots.	36
ENBL	Enable or disable a serial port	36
ENCFACT	Sets the number of pulses generated by the speed encoder.	36
ENCMODE	Sets the functionality of the encoder input	37
FLOW	Sets the method for serial flow control.	37
HEADTYPE	Sets the print head type.	38
MASTER	Sets the printer mode to either Master or Slave	38
MULTI	Enable RS 232/RS 485 Multidrop	39
PARITY	Sets the parity used in the serial communications.	39
PRINTHT	Sets the print height.	40
RELAY	Sets the state of the relay output.	40
INKTYPE	Sets the ink type.	40
RELAYMODE	Sets the functionality of the relay output.	41
SERMODE	Sets the functionality of the auxiliary serial port.	41

(Sheet 1 of 2)

KEYWORD	Description	See page
TARGSPD	Sets the speed of the print target.	42
TILTASP	Sets the aspect ratio of a tilted print.	42
TRIGEND	Sets the action when the trigger is removed.	43
TRIGMODE	Sets the functionality of the trigger input.	43
TRIGSKIP	Sets how many trigger inputs must occur before the printer begins printing.	43
UPDMODE	Sets the event that causes messages to be rebuilt.	44

(Sheet 2 of 2)

ADDR

Sets the printer address for the external serial interface when used in a networked environment. Where the digit enclosed in square brackets is the serial port number (always 0 on this printer) and the argument, following the equals sign, is the address (1...32).

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…31	
Default:	0	
ASCII Example:	GP ADDR[0]<CR>	SP ADDR[0]=3<CR>
Response:	*ADDR[0]=0<CR><LF>*	*ok<CR><LF>*

When the address is set to a non-zero value, the address must prefix each serial command and will prefix each response.

ASCII Example:	[3]GP TIME<CR>
Response:	*[3]TIME=”124530”<CR><LF>*

Address 0 is reserved to communicate to all nodes on a network. For example, if all print controllers on a network are tied to the same line speed, then the TARGSPD property could be set in all units with the single command, as follows:

ASCII Example:	[0]SP TRAGSPD=1000<CR>
Response:	*ok<CR><LF>*

AUXIN

Returns the auxiliary input states.

Get/Set:	GET only
Data type:	Unsigned Byte
Range:	0…15
Default:	00000000

ASCII Example:	GP AUXIN<CR>
Response:	*AUXIN=0<CR><LF>*

BAUDRATE

Sets the baud rate of the auxiliary serial port where the digit enclosed in square brackets is the serial port number (always 0 on this printer) and the argument, following the equals sign, is the baud rate.
The printer must be restarted for a new setting to take effect.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	1…7	
Default:	2 for port 0 7 for port 3 (serial interface for an optional daughter card)	

ASCII Example:	GP BAUDRATE[0]<CR>	SP BAUDRATE[0]=1<CR>
Response:	*BAUDRATE[0]=2<CR><LF>*	*ok<CR><LF>*

Baud rate values:

Value	Baud rate
1	9600
2	19200
3[a]	28800
4	38400
5	57600
6	115200
7	230400

a. Do not set the baud rate to value 3 when using HyperTerminal to communicate with the printer. HyperTerminal does not support this baud rate and you will be unable to change it back.

COLSPAC

Sets the distance between columns of dots when PRINTHT=100 (see also TILTASP).

Get/Set:	GET	SET
Data type:	UNS WORD	
Range:	1…12000 [mm/1000]	
Default:	4000	
ASCII Example:	GP COLSPAC<CR>	SP COLSPAC=4000<CR>
Response:	*COLSPAC=1000<CR><LF>*	*ok<CR><LF>*

CONFIG

Sets the print head configuration according to the config values table below. See "Print Head Numbering" on page 5.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	1…6	
Default:	1	
ASCII Example:	GP CONFIG<CR>	SP CONFIG=1<CR>
Response:	*CONFIG=1<CR><LF>*	*ok<CR><LF>*

Config values:

	Value					
Y Position	**1**	**2**	**3**	**4**	**5**	**6**
0...7	7 valve	7 valve	7 valve	16 valve	16 valve	32 valve
8...15		7 valve	7 valve			
16...23			7 valve		16 valve	
24...31			7 valve			

DOTSIZE

Sets the size of the dots by setting the time that a valve is turned on. The value is in increments in periods of 10 microseconds (1=10 microseconds).

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	5…500	
Default:	50	
ASCII Example:	GP DOTSIZE<CR>	SP DOTSIZE=30<CR>
Response:	*DOTSIZE=50<CR><LF>*	*ok<CR><LF>*

ENBL

Enable or disable a serial port. The value enclosed in square brackets [0] is the port that the command is sent to.

Get/Set:	GET	SET
Data type:	Boolean	
Range:	0, 1	
Default:	1 for port 0 (ASCII) 0 for port 3 (serial interface for an optional daughter card)	
ASCII Example:	GP ENBL[0]<CR>	SP ENBL[0]=1<CR>
Response:	*enbl[0]=1<CR><LF>*	*ok<CR><LF>*

ENCFACT

Sets the number of pulses generated by the speed encoder per meter of travel.
See "ENCMODE" on page 37.

Get/Set:	GET	SET
Data type:	Unsigned Word	
Range:	1…50 000 [pulses per meter]	
Default:	10 000	
ASCII Example:	GP ENCFACT<CR>	SP ENCFACT=25000<CR>
Response:	*ENCFACT=10000<CR><LF>*	*ok<CR><LF>*

ENCMODE

Sets the functionality of the speed encoder, based on the values provided below.
See "ENCFACT" on page 36.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…5	
Default:	0	
ASCII Example:	GP ENCMODE<CR>	SP ENCMODE=1<CR>
Response:	*ENCMODE=0<CR><LF>*	*ok<CR><LF>*

Encmode values:

Value	Function
0	Disabled (time is used).
1	Single ended - A Input
2	Single ended - B Input
3	Double Count (A + B)
4	FWD Quadrature
5	Rev Quadrature

FLOW

Sets the method for flow control used in the serial communications, where the digit enclosed in square brackets is the serial port number (always 0 on this printer). The argument, following the equals sign, is the flow setting.
The printer must be restarted for a new setting to take effect.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…2	
Default:	0 for port 0 2 for port 3 (serial interface for an optional daughter card)	
ASCII Example:	GP FLOW[0]<CR>	SP FLOW[0]=0<CR>
Response:	*FLOW[0]=1<CR><LF>*	*ok<CR><LF>*

Flow values:

Value	Flow control
0	None
1	Xon/Xoff
2	RTS/CTS (not active in RS-485)

For the best possible communication reliability, it is strongly recommended to use the setting FLOW[0]=2.

HEADTYPE

Gets the print head type, as listed below. Note that the configuration will be carried out automatically if the appropriate cable is used.

Get/Set:	GET only
Data type:	Unsigned Byte
Range:	0…2
Default:	2
ASCII Example:	GP HEADTYPE<CR>
Response:	*HEADTYPE=0<CR><LF>*

Head type values:

Value	Head type
0	Standard
1	3000
2	8000

MASTER

Set the printer to either Master or Slave mode. In Master mode the name of the printer's selected message is transmitted through its serial port.

In Slave mode the message for print is selected according to the message name received through the printer's serial port.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0, 1	
Default:	0	
ASCII Example:	GP MASTER<CR>	SP MASTER=1<CR>
Response:	*master=1<CR><LF>*	*ok<CR><LF>*

Master values:

Value	ModeB
0	Slave
1	Master

MULTI

Enables RS-232/RS-485 Multidrop. The value enclosed in angle brackets [0] is the port that the command is sent to.as follows:

0 - ASCII

3 - Optional daughter card

Get/Set:	GET	SET
Data type:	Boolean	
Range:	0, 1	
Default:	0	
ASCII Example:	GP MULTI[0]<CR>	SP MULTI[0]=1<CR>
Response:	*multi[0]=0<CR><LF>*	*ok<CR><LF>*

PARITY

Gets/sets the parity used in the serial communications, where the digit enclosed in square brackets is the serial port number (always 0 on this printer) and the argument, following the equals sign, is the parity setting. If parity is none (set to 0), then the number of data bits is 8. All other parity settings results in 7 data bits. Stop bits is always 1.
The printer must be restarted for a new setting to take effect.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…4	
Default:	0	
ASCII Example:	GP PARITY[0]<CR>	SP PARITY[0]=1<CR>
Response:	*PARITY[0]=0<CR><LF>*	*ok<CR><LF>*

Parity values:

Value	Parity
0	None
1	Odd
2	Even
3	Mark
4	Zero

PRINTHT

Gets/sets the print height. Less than 100% requires tilting the head. When 100% the printer uses COLSPAC, when less than 100% TILTASP is used.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	10(%)…100(%)	
Default:	100	
ASCII Example:	GP PRINTHT<CR>	SP PRINTHT=20<CR>
Response:	*PRINTHT=100<CR><LF>*	*ok<CR><LF>*

RELAY

Sets the state of the relay output (See "RELAYMODE" on page 41).

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0,1	
Default:	0	
ASCII Example:	GP RELAY<CR>	SP RELAY=1<CR>
Response:	*RELAY=0<CR><LF>*	*ok<CR><LF>*

INKTYPE

Sets the ink type for ‘standby compensation’ (head open time).

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0...9	
Default:	0	
ASCII Example:	GP INKTYPE<CR>	SP INKTYPE=1<CR>
Response:	*INKTYPE=1<CR><LF>*	*ok<CR><LF>*

Ink Types:

Ink Type	Description
0	Default, no HOT (head open time) compensation
1	Minimum HOT (head open time) compensation
2	Increasing HOT (head open time) compensation
3	Increasing HOT (head open time) compensation
4	Increasing HOT (head open time) compensation
5	Increasing HOT (head open time) compensation

6	Increasing HOT (head open time) compensation
7	Increasing HOT (head open time) compensation
8	Increasing HOT (head open time) compensation
9	Maximum HOT (head open time) compensation

RELAYMODE

Sets the functionality of the relay output.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…4	
Default:	0	
ASCII Example:	GP RELAYMODE<CR>	SP RELAYMODE=1<CR>
Response:	*RELAYMODE=0<CR><LF>*	*ok<CR><LF>*

Relay mode values:

Value	Parity
0	Disabled/Software control (refer to ’RELAY’ on page 40).
1	Fail/Warn
2	Fail Only
4	Print status

SERMODE

Sets/gets the functionality of the auxiliary serial port where the digit enclosed in square brackets is the serial port number (always 0 on this printer) and the argument, following the equals sign, is the SERMODE setting.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0, 1	
Default:	0	
ASCII Example:	GP SERMODE[0]<CR>	SP SERMODE[0]=1<CR>
Response:	*SERMODE[0]=0<CR><LF>*	*ok<CR><LF>*

Serial mode values:

Value	Parity
0	I•Mark ASCII Protocol
1	DOD•2002 Protocol - Limited Set

TARGSPD

Sets the rate at which the print target is moving. Only relevant whenever a speed encoder is not being used (ENCMODE is set to 0).

Get/Set:	GET	SET
Data type:	Unsigned Word	
Range:	20…3500 millimetres per second	
Default:	200	
ASCII Example:	GP TARGSPD<CR>	SP TARGSPD=200<CR>
Response:	*TARGSPD=4<CR><LF>*	*ok<CR><LF>*

TILTASP

Sets the aspect ratio of a tilted print (PRINTHT < 100). The value is the number of dot columns. The table below shows some possible settings for PRINTHT and TILTASP that maintain a good aspect ratio.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	1…10	
Default:	1	
ASCII Example:	GP TILTASP<CR>	SP TILTASP=5<CR>
Response:	*TILTASP=4<CR><LF>*	*ok<CR><LF>*

TILTASP/PRINTHT example values:

TILTASP	PRINTHT
1	70
2	50
3	30
9	10

TRIGEND

Sets the action to occur when the trigger is removed. A value of 0 means continue printing until the message is complete, a value of 1 means terminate printing when the trigger is removed.

Get/Set:	GET	SET
Data type:	Boolean	
Range:	0,1	
Default:	0	
ASCII Example:	GP TRIGEND<CR>	SP TRIGEND=1<CR>
Response:	*TRIGEND=0<CR><LF>*	*ok<CR><LF>*

TRIGMODE

Selects the source of the trigger input, based on the table below.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…2	
Default:	1	
ASCII Example:	GP TRIGMODE<CR>	SP TRIGMODE=1<CR>
Response:	*TRIGMODE=0<CR><LF>*	*ok<CR><LF>*

Trigger mode values:

Value	Description
0	Software only (TRIGON/TRIGOFF)
1	One print cycle per sensor 1 input.
2	One print cycle per sensor 2 input.

TRIGSKIP

Sets how many trigger inputs must occur before the printer begins printing. Allows for skipping objects on a line.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…100	
Default:	0	
ASCII Example:	GP TRIGSKIP<CR>	SP TRIGSKIP=2<CR>
Response:	*TRIGSKIP=0<CR><LF>*	*ok<CR><LF>*

UPDMODE

Sets the event that causes messages to be rebuilt, per the table below. Set to 2 in continuous print mode.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…2	
Default:	1	
ASCII Example:	GP UPDMODE<CR>	SP UPDMODE=2<CR>
Response:	*UPDMODE=1<CR><LF>*	*ok<CR><LF>*

UPDMODE values:

Value	Description
0	Message is built once on the first trigger, and then remains the same (objects that are in a message will not be updated).
1	End of Trigger. The message is rebuilt when the trigger signal transitions to OFF.
2	End of Mark. The message is rebuilt after every mark. Useful when printing multiple marks on a target, or printing continuously.

Print Head Configuration Properties

These are properties which are used in the GP and SP commands to configure the printing of individual print heads.

The controller supports up to four print heads which are labelled 0, 1, 2 or 3. The print head that the command is intended for is stipulated in square brackets. For example, the command: GP MARGIN[1] will be sent to print head 1.

KEYWORD	Description	See page
MARGIN	Sets the size of the margin.	45
MARKEND	Sets the number of printouts to be repeated.	46
MARKGAP	Sets the distance between multiple printouts.	46
MARKLEN	Sets the target length.	46
MSGNUM	Sets the message number to be printed.	47
MSGSEL	Sets the method of message selection.	47
PRINTDIR	Sets the print direction.	48
PRINTNEG	Set the printing of a print head to negative format. The printout will be inverted.	48
PRINTCNT	Returns the total number of printouts for a specific print head.	49

MARGIN[head]

Sets the size of the margin to be implemented after a trigger is received and before the message begins printing.
The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…4000 [mm]	
Default:	0	
ASCII Example:	GP MARGIN[0]<CR>	SP MARGIN[0]=0<CR>
Response:	*MARGIN[0]=200<CR><LF>*	*ok<CR><LF>*

MARKEND[head]

Sets the number of printouts to be repeated after a trigger has been received. The printing will discontinue when the trigger is removed, even if the number of marks has not been completed. Set to zero (0) for continuous print mode.
The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…100	
Default:	1	
ASCII Example:	GP MARKEND[0]<CR>	SP MARKEND[0]=0<CR>
Response:	*MARKEND[0]=1<CR><LF>*	*ok<CR><LF>*

MARKGAP[head]

Sets the distance between printouts when multiple printouts are being printed during a single print cycle (see also MARKEND).
The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Unsigned Word	
Range:	0…4000 [mm]	
Default:	0	
ASCII Example:	GP MARKGAP[0]<CR>	SP MARKGAP[0]=250<CR>
Response:	*MARKGAP[0]=200<CR><LF>*	*ok<CR><LF>*

MARKLEN[head]

Sets the total length of the print target. Setting MARKLEN to 0 (zero) switches MARKLEN off.
The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Unsigned Word	
Range:	0…16000 [mm]	
Default:	0=AUTOLENGTH	
ASCII Example:	GP MARKLEN[0]<CR>	SP MARKLEN[0]=2000<CR>
Response:	*MARKLEN[0]=1000<CR><LF>*	*ok<CR><LF>*

MSGNUM[head]

Sets the message number to be printed for each print head. The format of the message is defined in the SET_MESSAGE(SM) command.
The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Unsigned Word	
Range:	0…299	
Default:	0	
ASCII Example:	GP MSGNUM[0]<CR>	SP MSGNUM[1]=2<CR>
Response:	*MSGNUM[0]=1<CR><LF>*	*ok<CR><LF>*

Message 299 is reserved for the special message [Blank]. Selecting message 299 will produce a blank printout. The [Blank] message is used to ensure that print heads which are not being tested, do not print.

MSGSEL[head]

Sets the method by which the message number to be printed is selected, based on the table below.
The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Unsigned Byte	
Range:	0…2	
Default:	0	
ASCII Example:	GP MSGSEL[0]<CR>	SP MSGSEL[0]=2<CR>
Response:	*MSGSEL[0]=1<CR><LF>*	*ok<CR><LF>*

MSGSEL values:

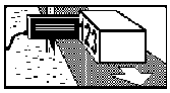
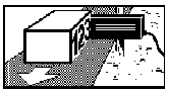
Value	Description
0	Message selected by MSGNUM[head].
1	Message selected by DOD2002 command (EscSM).
2	Message selected by Auxiliary Inputs (0…15)

PRINTDIR[head]

Sets the print direction for each print head attached to the SX-32e controller. The number enclosed in square brackets specifies the applicable print head.

Get/Set:	GET	SET
Data type:	Boolean	
Range:	0,1	
Default:	0	
ASCII Example:	GP PRINTDIR[0]<CR>	SP PRINTDIR[0]=1<CR>
Response:	*PRINTDIR[0]=0<CR><LF>*	*ok<CR><LF>*

PRINTDIR values:

Value	Description
0	Left 
1	Right 

PRINTNEG[head]

The print head can be set to print in a negative format. The black and white colors are inverted.

Get/Set:	GET	SET
Data Type:	Boolean	
Range:	0, 1	
Default:	0	
ASCII Example:	GP PRINTNEG[0]<CR>	SP PRINTNEG[0]=0<CR>
Response:	*printneg[0]=0<CR><LF>*	*ok<CR><LF>*

PRINTCNT[head]

Returns the total number of print cycles for the print head.This is a read only property and can only be reset through re-activation of the control unit (restoring factory settings).

Get/Set:	GET only
Data Type:	Unsigned Long
Range:	$1 \ldots (2^{32} - 1)$
Default:	

ASCII Example:	GP PRINTCNT[0]<CR>
Response:	*printcnt[0]=12345<CR><LF>*

Printer Utility Commands

Printer utility commands are used to manage the controller.

KEYWORD	Description	See page
CDB	Clears all records in the database	50
CE	Clears the error register.	50
CLEANER	Selects cleaner fluid at print head	50
DISABLE	Disables printing.	51
ENABLE	Enables printing.	51
FLUSH	Flushes print heads	51
INK	Selects ink at print head	51
RESET	Resets the controller to factory defaults.	51
TRIGOFF	Terminates a software activated trigger.	51
TRIGON	Activates a printout through the software.	52
WARMBOOT	Initiates a warm boot of the print engine	52

CDB

Clears all records in the database.

ASCII Example:	CDB<CR>
Response:	*ok<CR><LF>*

CE

Clears all fault and warning registers.The response will occur once the CE command has completed.

ASCII Example:	CE<CR>
Response:	*ok<CR><LF>*

CLEANER

Switches the valve to cleaner fluid, in 16 or 32 valve print head(s). (See also INK).

ASCII Example:	CLEANER<CR>
Response:	*ok<CR><LF>*

DISABLE

Disables printing (or new printing) from occurring.

ASCII Example:	DISABLE<CR>
Response:	*ok<CR><LF>*

ENABLE

Allows printing to occur. This is the default state.

ASCII Example:	ENABLE<CR>
Response:	*ok<CR><LF>*

FLUSH

Causes the print head number [n] to open all nozzles for the specified number of milli-seconds (1 to 5000). The fluid is not changed. The dot size is 100(1000us) and the column time is 10ms.

ASCII Example:	FLUSH[0]=500<CR>
Response:	*ok<CR><LF>*

INK

Switches the valve to ink in 16 or 32 valve print head(s). (See also CLEANER).

ASCII Example:	INK<CR>
Response:	*ok<CR><LF>*

RESET

Resets the controller to factory defaults.

Note that the RESET command will cause the communication parameters to return their default values which may result in the loss of communication. The response will occur once the RESET command has completed.

ASCII Example:	RESET<CR>
Response:	*ok<CR><LF>*

TRIGOFF

Terminates a software activated trigger. Please refer to the note provided below, under "TRIGON" on page 52.

ASCII Example:	TRIGOFF<CR>
Response:	*ok<CR><LF>*

TRIGON

Activates a printout through the software.
The software trigger must be selected for this to have any effect on print.
See "TRIGMODE" on page 43.

ASCII Example:	TRIGON<CR>
Response:	*ok<CR><LF>*

***Note**: The TRIGON and TRIGOFF commands must always be used in conjunction with each other. Simply sending the command TRIGON repeatedly, will not result in further printouts. You must send the command sequence:TRIGON, TRIGOFF, TRIGON, TRIGOFF etc. to initiate further printouts. A printout is only activated on the switch of states from TRIGOFF to TRIGON.*

WARMBOOT

Initiates a warm boot of the print engine.

ASCII Example:	WARMBOOT<CR>
Response:	*ok<CR><LF>*

Creating Graphics

An Introduction

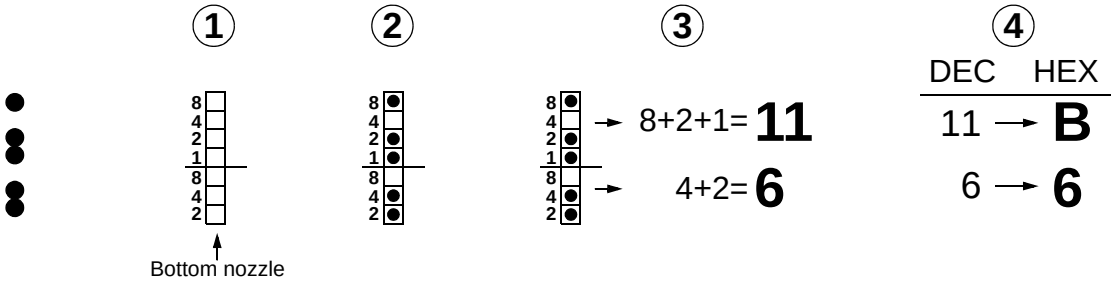
"FE01FF01FF81FFFFFF81FF01FE01", in the command below, is hexadecimal code (graphic data) defining the dots which make up a graphic, in this case the *Fragile* symbol.

Command: SG[6]="FE01FF01FF81FFFFFF81FF01FE01",NAME="Fragile",SIZE=16

Resulting graphic: 

The hexadecimal data is calculated for each group of four nozzles within a column, until the whole column has been encoded (16 and 32 valve print heads). The same procedure is then carried out for each consecutive column. *Note the 7 valve print head is grouped as shown below, 4 and 3 nozzles.*

The following shows how to create a pattern of dots, consisting of a single column, with a 7 nozzle print head:



1 Draw and mark a column of squares to represent all the print head nozzles. In this case a 7 nozzle head.

2 Draw the pattern of dots in the squares.

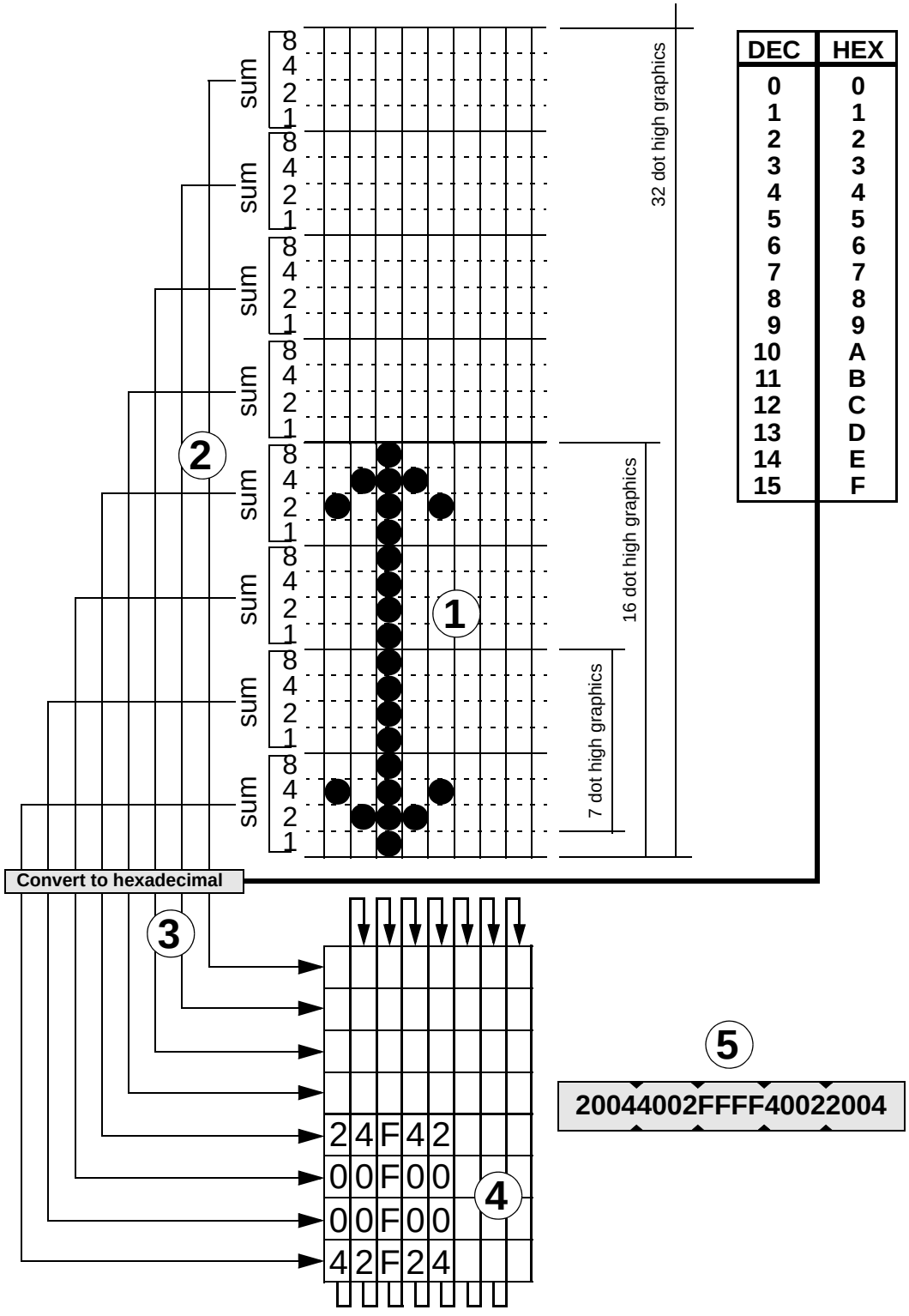
3 Add together the values for each square containing a dot in the top group of nozzles and then do the same for the bottom group.

4 Convert each decimal sum to hexadecimal.

Entering B6 as the graphic data, in the SG command, will produce the dot pattern.

Template Example

The following shows how the graphics template is used. *Note that when creating a graphic for a 7 valve printhead, exclude the bottom most row, thus reflecting the 7 valves, as shown above and below.*

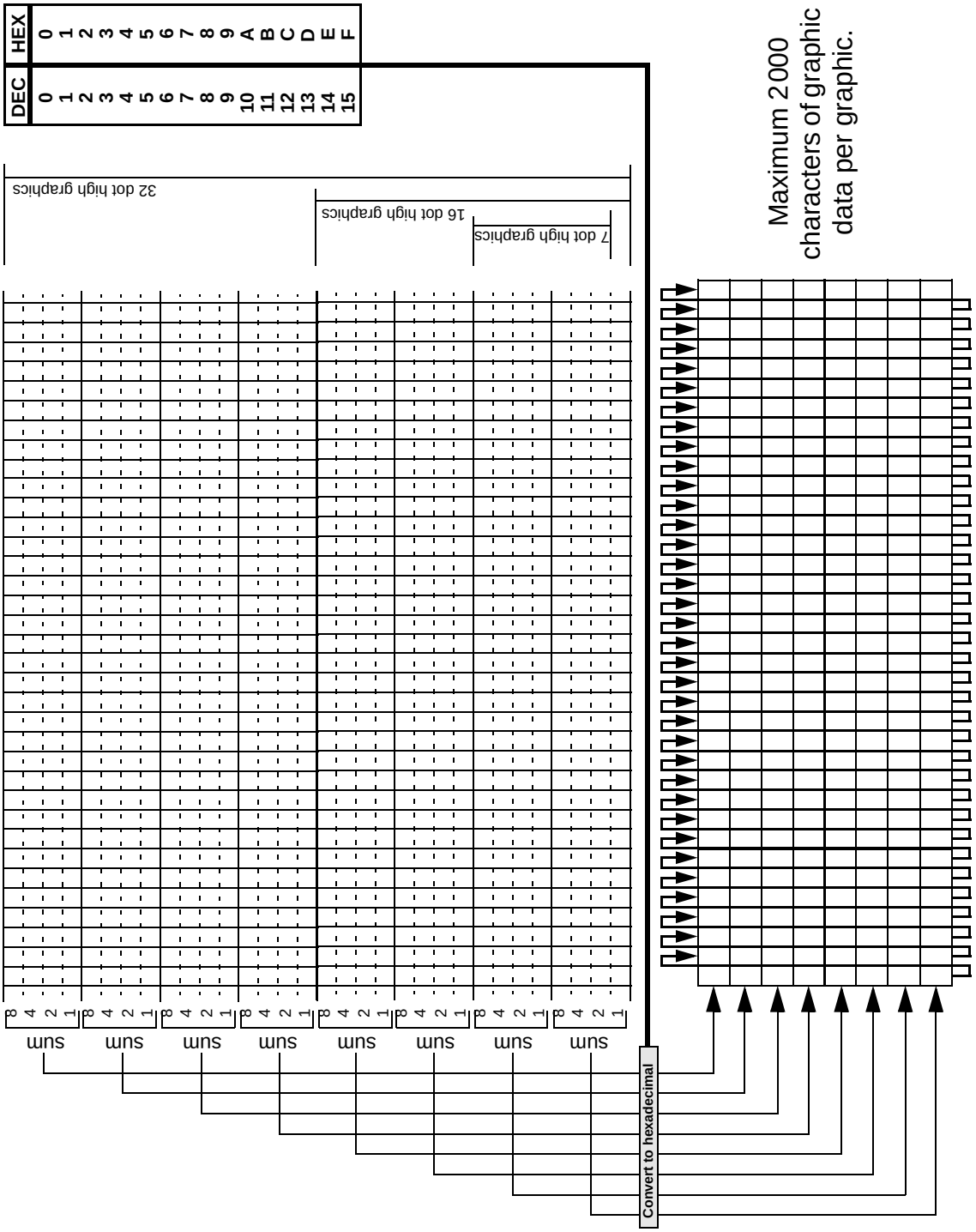


DEC	HEX
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Template

Copy this page for use when designing up to 32 dot high graphics and for determining the resulting graphic data for use in the SG command.

See also “GG, SG (Get and Set Graphic)” on page 10 and “Creating Graphics” on page 53.



ASCII SX-32e Protocol

The following provides a summary of changes in the ASCII protocol, as implemented within the SX-32e control unit.

General

- Number of message objects increased from 100 to 300[(1)]
- Number of Text Objects increased from 250 to 500.
- A number of data type changes were made in the underlying object model.

Printer Configuration Commands

The following configuration commands have been included in the SX-32e protocol.

- INKTYPE: Sets the ink type for 'standby compensation' (head open time)
- PRINTCNT: Returns the total number of printouts.
- LANGUAGE: Sets the display language for the GUI.
- BRIGHTNESS: Used by GUI for the display's brightness intensity level.
- USERLEVEL: Used by the GUI for setting the user level.

The following modifications have been made to existing keyword commands:

- Counter cascade functionality included in the controller configuration property UPDMODE
- Print status added to the controller configuration property RELAYMODE

Printer Utility Commands

The following utility command have been included in the SX-32e protocol.

- CDB: Clears all records in the database.

Set Message (SM) Structure

The SM structure has changed as highlighted below.

SX-32 Format: SM[n]=(a[n]@x:y, a[n]@x:y,….)

SX-32e Format: SM[n]=((a[n]@x:y,FONT=f,SIZE=s,BOLD=b,XSP=x,MODE=m),(b[n]@x:y,FONT=f,.....)),name="name"

Noticeably the attributes SIZE etc. can be set within the actual Set Message command and will be reflected in the GUI. It is important that the user uses the keyword FONT (font=0) when creating messages (it acts as a trigger), so that these attributes are set. If

1). Of the 300 messages only 200 are accessible via the GUI. The serial interface is required to access all 300 messages.

not, the actual object attributes, such as a text object, within the message construct, take precedence. Refer to “GM, SM (Get and Set Message)” on page 20, for further information.

Glossary of Terms

Term	Meaning
ASCII	A(merican) S(tandard) C(ode) for I(nformation) I(nterchange) is the built-in binary code for representing characters in all computers except IBM mainframes.
GUI	G(raphical) U(ser) I(nterface) - The printer’s graphical display which the user uses to enter commands.
Protocol	Format for transmitting data between two devices.
String	A collection of UTF-8 encoded ASCII characters.
Unicode	A character code that defines every character in most of the speaking languages in the world.
Unsigned byte	The range of bits 0 - 255 in a byte.
User interface	See GUI.
UTF-8	8-bit U(nicode) T(ransformation) F(ormat) is a variable-length character encoding which is able to represent any universal character in the Unicode standard.

Documentation History

Each Technical Manual has been written for a specific control unit software version or major hardware feature. The following table shows which manual should be used with which control unit.

Note that in no way does this table show compatibility between software versions.

Manual version number	Manual issue date (month/year)	Software version (Print engine)	Major updates in manual
V5 I1	12/12	1.6 and above (171-8-5)	Note 7
V4 I1	04/09	1.6 and above	Note 6
V3 I3	08/08	1.5 and above	Note 5
V3 I2	07/08	1.5 and above	Note 4
V3 I1	03/08	1.4 and above	Note 3
V2 I1	09/06	1.0 and above	Note 2
V1 I1	09/06	1.0 and above	Note 1

Note 7
a. ZERO property added (see page 32).
b. Barcode information updated (see page 18).
c. Updated ENCMODE (see page 37).
d. Updated Language list (see page 25).
e. Added commands: ENBL, MASTER, and MULTI (see “Controller Configuration Properties” on page 32).
f. Printer utility command WARMBOOT added (see page 52).

Note 6
a. Corrections made to information regarding the creation of graphics (see page 53).

Note 5
a. Summary of SX-32e changes added to appendices.
b. Removed SPDMODE from manual (not relevant).
c. Removed Ink low, (value 3) from RELAYMODE (not relevant).

Note 4
a. Corrections made regarding the format of data for Unicode strings.
b. Removed SENSDIST from manual (not relevant).

Note 3
a. Updated for the SX-32e control unit ASCII Protocol.
b. SPDMODE, INKTYPE, PRINTCNT, LANGUAGE, BRIGHTNESS, USERLEVEL, and CDB properties added.

Note 2
TARGSPD property added.

Note 1
First release

Index of Commands

U

V

W

Z

General Index

N

O

P

R

S

T

U-Z