HOST MODE PROGRAMMING

> Matrix N Family

DATALOGIC

Datalogic S.r.l.
Via S. Vitalino 13
40012 Calderara di Reno
Italy

Matrix N Family Host Mode Programming

Ed.: 06/2019

This manual refers to software release 1.8.0 or later.

18/06/19

CONTENTS

CONVENTIONS

The notation in this manual uses angular brackets (< >), to represent **single byte character values** which are equivalent. See ASCII Table in appendix A. For special keys or some non-printable characters, the ASCII symbol for the character name is shown between angular brackets.

Example: <ESC> means the ESC key character byte
<STX> means the start transmission character byte
<space> means the whitespace character

- For non-printable characters, the decimal value of the character byte is shown between angular brackets.

  Example: <13><10> = Carriage Return character byte, Line Feed character byte

The following representations using these three methods are equivalent:

<ESC> = <27> = <$1B_H$>
and each send character byte 0x1B to the reader

The program used to generate the strings for Host Mode Programming must be able to manage them correctly.

1 HOST MODE PROGRAMMING

Host Mode programming allows you to perform a partial configuration of your reading device as an alternative method to configuration using DL.CODE™. It consists of substituting or modifying the Matrix N Family device configurations by sending command strings over one of the communication channels.

Probably the most useful command is to change configuration jobs from among those saved in the device memory, directly from the Host.

These strings take the form of command sequences that are specific to each saved configuration. They can be personalized to make command recognition easier. They are transmitted from the Host system to the Matrix reader on the following interfaces:

- On-board Ethernet TCP/IP interface
- Main RS232/RS422 serial interface
- Auxiliary RS232 serial interface
- Profinet-IO embedded
- HMS Fieldbus

In order to send the programming strings, it is necessary to switch the reader into **Host Mode**.

CAUTION: DL.CODE must be disconnected from all reader interfaces before entering in Host Mode.

By entering in Host Mode, any DL.CODE Monitor or X-PRESS connections to the reader will be closed.

Ethernet Interface

The programming commands and strings can be sent to the reader using the on-board Ethernet TCP/IP interface. Connect to the device IP address, port number: 1023 (pre-defined).

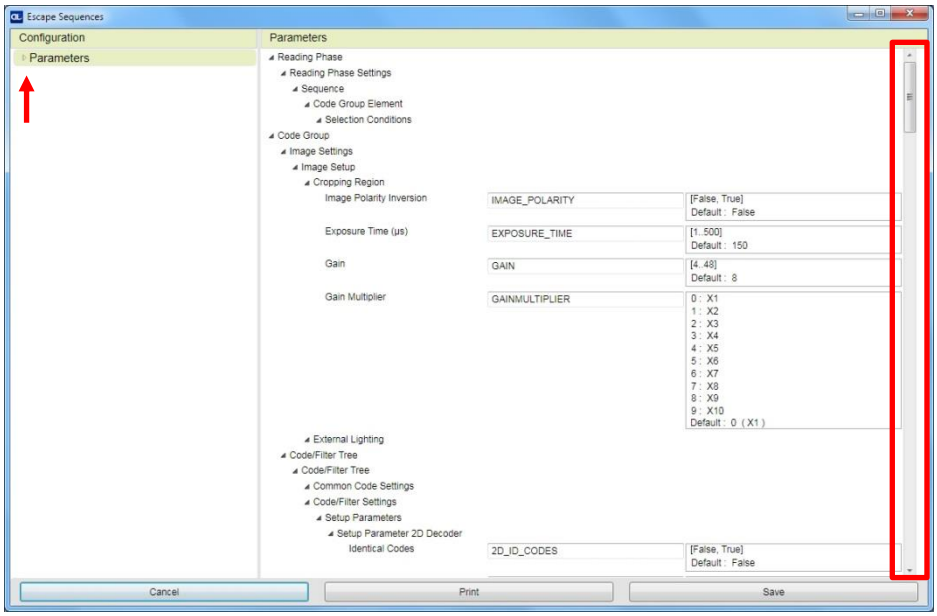
Serial Interfaces

The programming commands and strings can be sent to the reader at the programmed baud rate of the selected serial interface (e.g. if the baud rate of the Auxiliary interface is programmed at 115200 bps the command must be sent at 115200 bps).

1.1 COLLECTING HMP STRINGS FROM DL.CODE

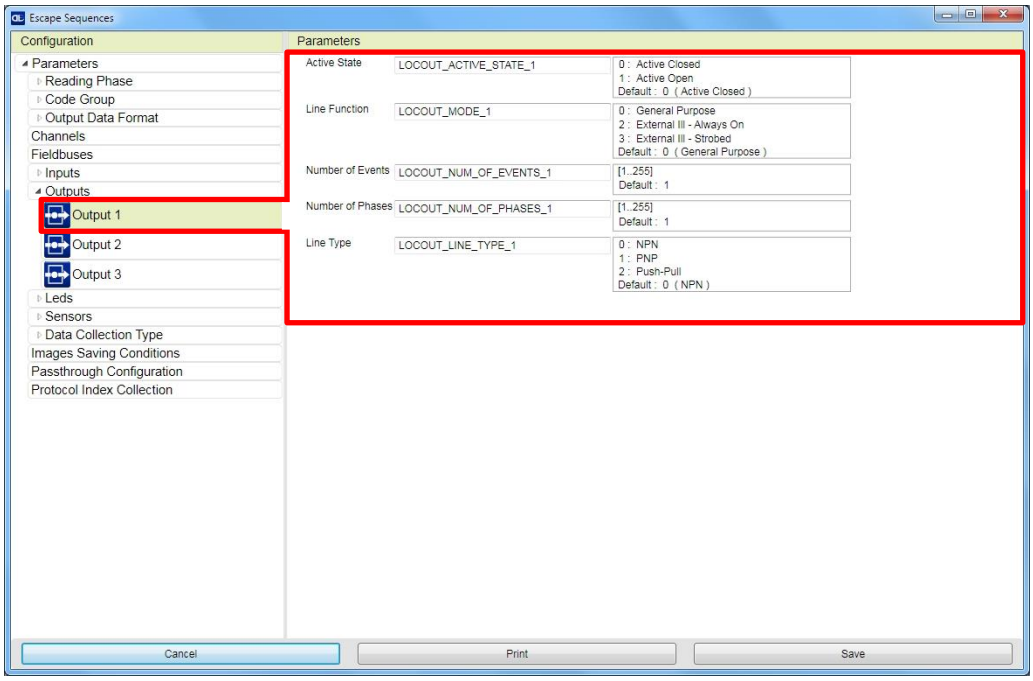
The first thing to do is collect the application specific Programming Strings from the DL.CODE Configuration. Select **View Configuration HMP Shortcuts** from the Device>Settings menu.

This opens the Escape Sequences window.



You can view the full extension of configuration parameters by dragging the vertical slider on the right side of the window. Otherwise by clicking on the Parameters triangle you can open the tree to select any individual branch and view only the branch specific parameters.

Each configurable parameter is shown with its programming string name and a list of its configuration parameter values including the default setting.



You can personalize the programming strings from any view in this window by clicking on the string name and entering a new name (no spaces are allowed). Then click the Save button.

Click the Print button to produce an HMP reference sheet.

2 PROGRAMMING COMMANDS

2.1 CONNECTION TO DEVICE

	DESCRIPTION	HOST COMMAND	REPLY MESSAGE
1	**Enter Host Mode**	**<ESC> [C**	**<ESC> H <CR><LF>**
	After entering this command, the device responds with the first reply message and then waits for the following command.		
2	**Enter Programming Mode**	**<ESC> [B**	**<ESC> S <CR><LF>**
	After entering this command, the device responds with the second reply message. Now you can send one or more programming strings as shown in Chapter 3.		
3	**Exit Host Mode**	**<ESC> [A**	**<ESC> [X**
	This command must always be transmitted to end the programming session. The configuration job management depends on the commands issued prior to exit from Host Mode. See 3.2 for more details. • If the SAVE command was issued before exit from Host Mode, the job is saved in permanent memory (Flash). • If the SAVE command was not issued before exit from Host Mode, the job is saved in temporary memory (RAM).		

3 PROGRAMMING STRINGS

3.1 DEFINITIONS

Common definitions for each parameter are:

Name (parameter name)
Name of the parameter available in the DL.CODE Configuration Parameters panel.

The other definitions change according to the parameter type.

Type (parameter type)
Parameter type is essential in order to decide the features of the command used to set its new value. Types are:

Type 0: Integer (numeric)
Type 1: Enumerative (numeric list)
Type 2: BOOLEAN (true, false)
Type 3 String (printable characters)

Integer (Type = 0)

Range
Minimum and maximum values allowed for the parameter.

Default
Default value (integer).

Example: *Exposure Time (μs)* parameter

Name: Exposure Time (μs)
Type: 0
Range: 6 to 500 (depending on model)
Default: 270 (depending on model)

Enumerative (Type = 1)

Item List
List of the values allowed for the parameter.

Default
Default entry (i.e. 0 = first entry of the list, 1 = second entry of the list, etc.).

Example: *Gain Multiplier* parameter.

Name: Gain Multiplier
Type: 1
Item List X1, X2, X3, X4, X5, X6, X7, X8, X9, X10
Default: 0 (Entry 0 = X1)

BOOLEAN (Type = 2)

Value
true = Enabled; false = Disabled

Default
Default value (String).

Example: *Identical Codes* parameter.

Name: Identical Codes
Type: 2
Value: false, true (case sensitive)
Default: false (Disabled)

String (Type = 3)

Length
Minimum to maximum number of characters allowed for this parameter.

Default
Default value (String).

Example: *Wildcard* parameter.

Name: Wildcard
Type: 3
Length Min to Max number of allowable characters in string
Value: Any character from 00 to FF (hex)
Default: *

3.2 PROGRAMMING STRINGS

3.2.1 Set Parameter Value in Current Configuration

The Matrix device single parameter programming string must have the following format:

SET<space>PSTR<space>VAL <LF>

where:

- **SET**: The programming command (all caps)
- **PSTR**: The unique parameter string containing no spaces
- **VAL**: Character or string indicating the assigned parameter value. (case sensitive)

If the command is correct, the device immediately implements the new value and confirms with the following message:

ACK <LF>

If programming contents are wrong (i.e. a typing error in the file) or due to a transmission error, the device replies with the following message and programming data will not be updated in this case:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.2 Get Parameter Value in Current Configuration

The Matrix device single parameter get string must have the following format:

GET<space>PSTR <LF>

where:

- **GET**: The programming command (all caps)
- **PSTR**: The unique parameter string containing no spaces

If the command is correct, the device replies with the value string for the current parameter:

ACK VAL <LF>

If the sequence is wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.3 Get Configurations List (Jobs)

The Matrix get configurations list command must have the following format:

GET_JOBS_LIST <LF>

If the command is correct, the device confirms with the following message:

ACK <LF>
Default <LF> <CR>
…. <LF> <CR>

If the command is wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.4 Save Current Configuration (Job)

The Matrix configuration saving command must have the following format:

SAVE<space>configuration_name <LF>

After making changes to a configuration, this command should always be transmitted before exit from programming mode.

If the command is correct, the device confirms with the following message:

ACK <LF>

If the command is wrong (i.e. missing configuration_name) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: To overwrite the current configuration, the configuration name must exactly match the name saved on the device. Names can contain spaces and **are** case sensitive!

If the name does not match, then a copy of the configuration will be saved with the new name.

The Default configuration cannot be overwritten.

3.2.5 Set Startup Configuration (Job)

The Matrix startup configuration setting command must have the following format:

STARTUP_CFG<space>configuration_name <LF>

If the command is correct, the device confirms with the following message:

ACK <LF>

If the command is wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: The configuration names must match those saved on the device. They can contain spaces and **are** case sensitive.

This command does not change the current configuration. It also does not require the **SAVE** command to be issued.

3.2.6 Change Current Configuration (Job)

The Matrix change configuration command must have the following format:

CHANGE_CFG<space>configuration_name <LF>

If the sequence is correct, the device confirms with the following message:

ACK <LF>

If programming contents are wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: The configuration names must match those saved on the device. They can contain spaces and **are** case sensitive.

This command does not change the Startup Configuration.

3.2.7 Match Code Content

The Matrix match code command must have the following format:

MATCH<space>code_content <LF>

After using this command, the **SAVE** configuration command should always be transmitted before exit from programming mode.

If the command is correct, the device replies with the **code_content** string for the Expected Code parameter of the current Match Code configuration:

ACK code_content <LF>

If programming contents are wrong (i.e. a typing error) or due to a transmission error, or if the current configuration is not a Match Code data collection type, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: The configuration must be a **Match Code** data collection type previously saved on the device. The **code_content** can contain spaces and **is** case sensitive.

This command overwrites the current Expected Code parameter value.

3.2.8 Send Configuration (Upload Job) to Device

CAUTION: This command requires a dedicated program to handle file transfer protocol (see below).

It is possible to send a **.dlcfg** (job) file to the Matrix device, saving it to permanent memory.

NOTE: Any Matrix configuration can be easily exported to a .**dlcfg** file from DL.CODE by selecting

`File > Save > Save Configuration on PC`

The send configuration to matrix device command must have the following format:

SEND_CFG<space>configuration_name <LF>

Within 10 seconds of sending the SEND_CFG command, the **.dlcfg** configuration file transmission must be initiated. The entire file content must be encapsulated using a CRC32 checksum control and transmitted to the device using X-Modem protocol.

Failing to initiate transmission within 10 seconds will result in the device replying with a Timeout message and the command will be considered aborted.

The byte stream containing the configuration that is sent to the device is structured with a 4-byte header containing the total stream length, followed by a data_buffer section containing the whole .dlcfg byte sequence, followed by a 4-byte CRC32 terminator storing the value computed on the previous length + data_buffer sections content.

Length (4 bytes): specifies the number of bytes of the message (*Data buffer* + *CRC* field).

Data buffer: the bytes of the file to transmit.

CRC: the CRC32 checksum based on both *Length* + *Data buffer*.

NOTES:

- The data format used is **BIG ENDIAN**
- The **packet timeout during transmission is 10 seconds**: on timeout expiration command is considered aborted
- **Only .dlcfg files** are allowed for upload

If the transmission is completed correctly, the device confirms with the following message:

ACK <LF>

If transmission is not initiated within 10 seconds, the device replies with the following message:

Timeout! <LF>

If programming contents are wrong (i.e. file format, content or CRC error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: The saved configuration will have the name you give the **configuration_name** parameter. Names can contain spaces and **are** case sensitive. Do not include the extension.

If a configuration with the same name already exists in the device memory it will be overwritten by the new file.

This command does not change the Current Configuration nor the Startup Configuration. Use the appropriate commands to change it.

NOTE: There is no Host Mode Programming command available for downloading a configuration from a device.

3.2.9 Get Device Information

The Matrix get device information command must have the following format:

GET_INFO <LF>

If the sequence is correct, the device confirms with the following message:

ACK Device Information <LF>

If programming contents are wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.10 Send Images

For configurations that allow **Image Saving – Send Image On: Demand**, it is possible to start the FTP images transfer from the device to an FTP Server through this command.

The Matrix send images command must have the following format:

SEND_IMAGES<space><NodeNumber>[<space><IdStart>][<space><IdStop>] <LF>

where:

SEND_IMAGES:	The programming command (all caps).
NodeNumber: (mandatory)	Specifies the node number (0=Master, 1 to 31=Slaves). The asterisk character '*****'can be used to send a broadcast command. In this case the Master and all Slaves will start to transfer images to the FTP Server.
IdStart: (optional)	The pack/phase number from which image transfer begins.
IdStop: (optional)	The pack/phase number from which image transfer ends.

NOTE: If the optional parameters are omitted, the entire image buffer of the device (or devices) will be sent.

PRE-REQUISISTES:

- The device configuration runs in **Phase Mode, Packtrack** or **OneShot**
- An FTP Server is available and configured to allow uploads on the same TCP/IP network
- After entering the host programming sequence, wait at least 50 ms for the device to correctly setup internal data structures before issuing SEND_IMAGES commands
- The device configuration (Data Formatting) must include an FTP client channel which must be configured to access the FTP Server (user/password, folder, …)
- The device configuration (Data Formatting) must include an Image Saving Condition which must be configured with:
    - **Image Destination > On Remote**
    - **Send Image On > Demand**
    - **Saving Condition** > Good Reads only, No Reads only, Both
    - **Image Format** > available formats are RAW, BMP, PNG, JPG, Tiff
    - **Remote Folder** > subfolder name on the FTP Server under which images will be saved
    - **Send Image From > Matrix FTP Client**

See the Image Saving chapter in the DL.CODE User's Manual for more details.

SAVED IMAGE FILENAMES:

On the FTP Server, the saved images filenames will have the following syntax:

IMG_nnnnnnnnnn_ss_ff_ccc.ext

where:

nnnnnnnnnn:	pack#/phase# 10 digits
ss:	slave id (00 for Master, 01-31 for slaves)
ff:	frame sequence # for current image within the related pack/phase
ccc:	# of valid barcodes detected by the reader who produced this image (000 for noread)
ext:	extension to indicate image format (RAW, JPG, PNG, PNG, TIFF)

If the transmission is completed correctly, the device confirms with the following message:

ACK <LF>

If programming contents are wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

NOTE: The transferred images come from device internal circular buffer which is used for this specific purpose. The buffer size depends on the complexity of the configuration as it is allocated during runtime. The buffer is managed in a round-robin fashion, therefore requesting of packs/phases that are “too old” in the buffer may result in the reader not being able to transmit images, even if returning an ACK response (to be changed in future releases). On IdStart>IdStop errors, NACK is returned.

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

Images can also be sent to a TCP Channel On Demand.

CAUTION: Sending images on demand to a TCP channel requires a dedicated program to correctly handle file transfer.

3.2.11 Get Command Status

This command generically returns information about the last command executed on the device. It is currently used after a Send Images command to monitor the downloading progress.

NOTE: Take into account that as the device continues to capture images, the circular image buffer changes and may be overwritten.

The Matrix get command status command must have the following format:

GET_CMD_STATUS <LF>

If the last command sent to the device was SEND_IMAGES, the device responds with the following message:

SEND_IMAGES<space>ErrorCode[<space>Info1][<space>Info2][<space>Info3] [<space>Info4] <LF>

where:

Error Code	Meaning
0	no error
1	generic error
10	command not found (syntax error)
11	wrong arguments (syntax/semantic)
20	send images in progress
21	send images ok (completed)
22	send images failed

and

Info Field	Meaning
Info1	number of images sent
Info2	total number of images saved in device image buffer
Info3	beginning image phase # range
Info4	ending image phase # range

Example Get Command Status responses:

Host	<>	Device	Meaning
SEND_IMAGES * 3 10	**>**		Host requests device to send images from phase #3 to phase #10.
	<	**ACK**	Device accepts command.
GET_CMD_STATUS	**>**		Host requests last command status info.
	<	**SEND_IMAGES 20**	Device reports SEND_IMAGES is still in progress.
GET_CMD_STATUS	**>**		Host requests last command status info.
	<	**SEND_IMAGES 21 1 3 10 12**	Device reports SEND_IMAGES has completed successfully: sent 1 image from a total of 3 located in phase range #10 to #12.
SEND_IMAGES * 3 10	**>**		Host requests device to send images from phase #3 to phase #10.
	<	**ACK**	Device accepts command.
GET_CMD_STATUS	**>**		Host requests last command status info.
	<	**SEND_IMAGES 20**	Device reports SEND_IMAGES is still in progress.
GET_CMD_STATUS	**>**		Host requests last command status info.
	<	**SEND_IMAGES 22 0 0 -1 -1**	Device reports SEND_IMAGES has failed: 0/0 images sent.

For most other commands the response is:

NACK <LF>

This is also the case if programming contents are wrong (i.e. a typing error) or due to a transmission error.

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.12 Get Match Code Expected Content

To get the expected content of a match code configuration, use the following format:

GET_ASCII MATCH_CODE_EXPECTED_CONTENT1<CR> <LF>

If the command is correct, the device replies with the **expected_content** string (in ASCII format) of the current Match Code configuration:

ACK expected_content <LF>

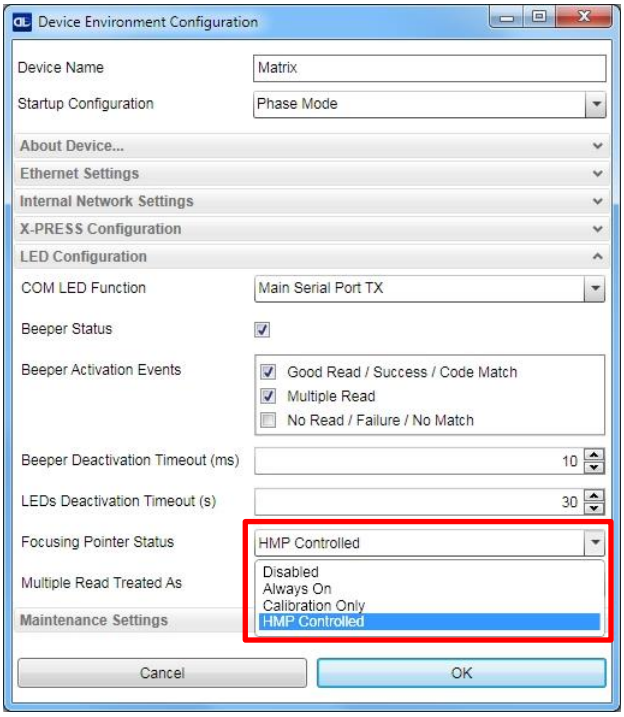
If programming contents are wrong (i.e. a typing error) or due to a transmission error, or if the current configuration is not a Match Code data collection type, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.13 Set Focusing Pointer

This command can be used only if the device has its Focusing Pointer Status set to *HMP enabled* in the DL.CODE Device Environment settings.



The Matrix device set pointer command must have the following format:

SET_POINTER<space><ON|OFF> <LF>

If the command is correct, the device immediately sets the focusing pointer ON or OFF and confirms with the following message:

ACK <LF>

If programming contents are wrong (i.e. a typing error in the file) or due to a transmission error, the device replies with the following message and programming data will not be updated in this case:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: This setting is not saved in permanent memory. The Focusing Pointer defaults to the OFF state when power is cycled.

3.2.14 Set Image Cropping Region

To set the image cropping region, use the following format:

SET_CROPPING <left> <top> <width> <height> [<image-settings-index>]<LF>

Where:

- **<left>** is the cropping region offset from the sensor left edge
- **<top>** is the cropping region offset from the sensor top edge
- **<width>** is the cropping region width
- **<height>** is the cropping region height
- **<image-settings-index>** (optional, default 1) is the index of the Image Settings to modify. This allows using the command with multiple Image Settings.

If the command is correct, the cropping region is applied, and the device confirms with the following message:

ACK <LF>

If programming contents are wrong (i.e. a typing error in the file) or due to a transmission error, the device replies with the following message and programming data will not be updated in this case:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

NOTE: If the parameters do not respect the sensor constraints (e.g. width and right multiple of 32), the command will fail.

3.2.15 Read Input

The Matrix read input command must have the following format:

READ_INPUT<space><Input1|Input2> <LF>

If the sequence is correct, the device confirms with the following message:

ACK Input n state <LF>

Input n state depends on the configured active state.

if Active Closed (default): 0 (open - no current flow); 1 (closed - current flow); -1 (error)
if Active Open: 0 (closed - current flow); 1 (open - no current flow); -1 (error)

If programming contents are wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.16 Read Output

The Matrix read output command must have the following format:

READ_OUTPUT<space><Output1|Output2|Output3> <LF>

If the sequence is correct, the device confirms with the following message:

ACK Output n state <LF>

Output n state depends on the configured active state.

if Active Closed (default): 0 (open - no current flow); 1 (closed - current flow); -1 (error)
if Active Open: 0 (closed - current flow); 1 (open - no current flow); -1 (error)

If programming contents are wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

3.2.17 Run X-PRESS Function

The Matrix xpress command must have the following format:

XPRESS<space><1|2|3|4> <LF>

If the sequence is correct, the device immediately executes the X-PRESS function associated with the selected number and confirms with the following message:

ACK <LF>

You can exit the X-PRESS function before its associated timeout by executing the same command a second time (toggle). If the associated timeout has already expired, then executing the xpress command a second time will run the function again.

If programming contents are wrong (i.e. a typing error) or due to a transmission error, the device replies with the following message:

NACK <LF>

After transmitting the reply message, the device then waits for another programming string or the exit from host mode programming command.

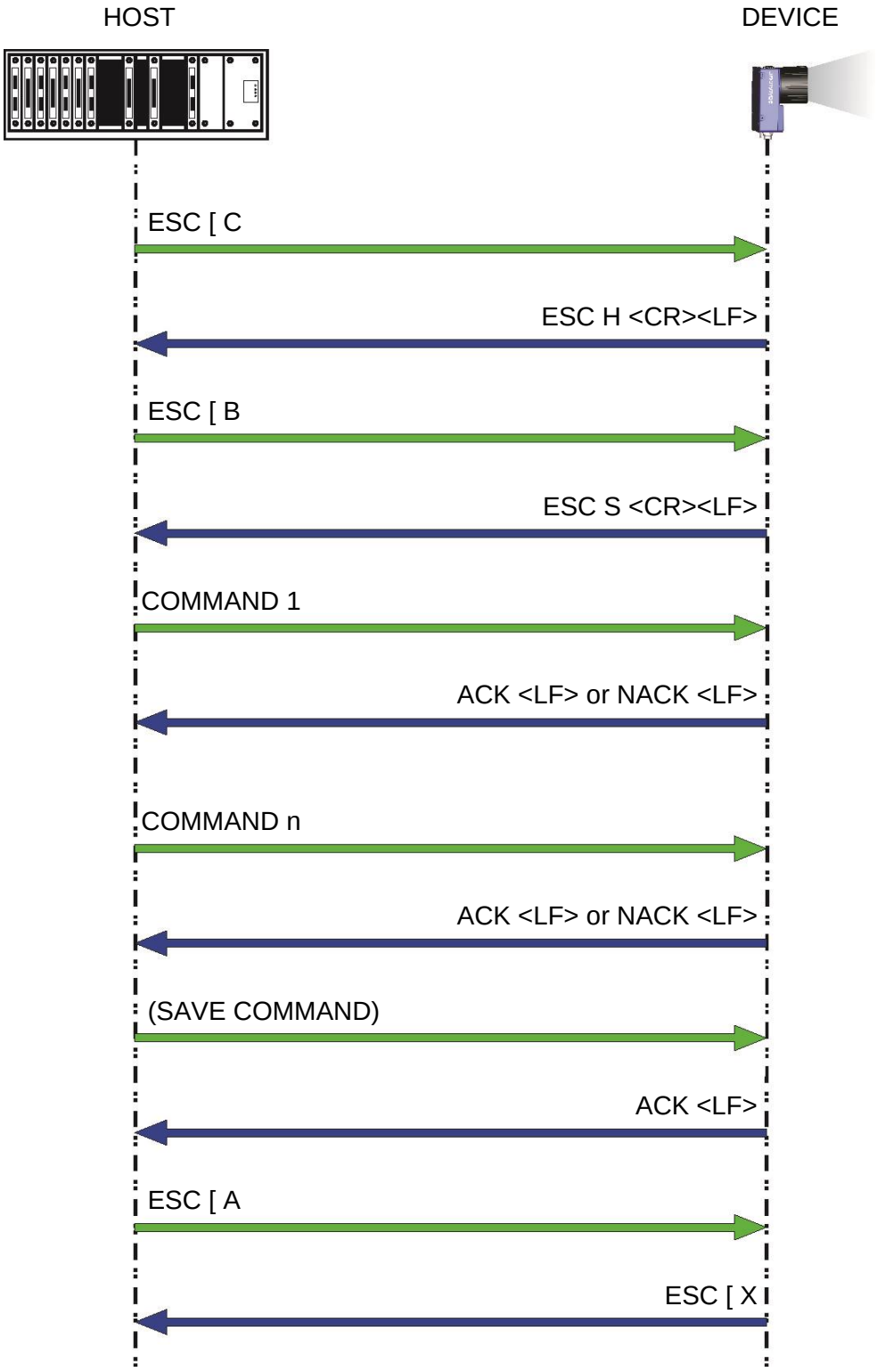


Figure 1 – Connection to Device, Programming and Disconnection from Device

3.3 EXAMPLES

- 1-
Set Image Setting 2 *Exposure Time* parameter to 4*00 µs*:

Type: 0
Range: 6 to 500
Data: 400

The programming string is:

SET<space>EXPOSURE_TIME1<space>400 <CR><LF>

- 2 -
Set *GainMultiplier* parameter to *X3*:

Type: 1
List: X1, X2, X3, X4, X5, X6, X7, X8, X9, X10
Data: 2

The programming string is:

SET<space>GAINMULTIPLIER<space>2 <CR><LF>

- 3 -
Set Data Matrix ECC 200 *Image Mirroring* parameter to *enabled*:

Type: 2
Item List: *true (Enabled), false (Disabled)*
Data: true

The programming string is:

SET<space>2D_IMAGE_MIRRORING<space>true <CR><LF>

- 4 -
Set Data Matrix ECC 200 Image Filter *Placeholder* character to *'&'*:

Type: 3
Item List: *any printable character*
Data: &

The programming string is:

SET<space>2D_PLACEHOLDER<space>& <CR><LF>

- 5 -
Get Matrix 410N Device Information:

The programming string is:

GET_INFO<CR><LF>

Device Response:

ACKModel name: M410N:700-010
MAC address: 00:07:BE:01:2B:56
Serial number: C14P00584
Device name: Matrix
Application version: 1.4.0.1073
Current Job: Phase Mode

- 6 -
Transfer all pending images from all devices in the cluster:

The programming string is:

SEND_IMAGES *<CR><LF>

- 7 -
Transfer images from slave 2 starting from pack/phase 123:

The programming string is:

SEND_IMAGES 2 123<CR><LF>

- 8 -
Transfer images from Master for packs/phases 10 to 12:

The programming string is:

SEND_IMAGES 0 10 12<CR><LF>

4 MATRIX N FAMILY PARAMETERS LIST

The following tables present a complete list of parameters that can be modified using Host Mode Programming.

The parameter strings shown here are the default ones.

The strings can be modified (personalized) through DL.CODE so it is advised to print them out from DL.CODE for your specific configuration.

These strings are also dynamic. They depend on the order in which the configuration is made. For example, each group of 2D or 1D Code parameters is associated with STRINGx where “x” is a dynamically assigned number depending on the chronological order in which it was configured. For this reason you must know the relative strings assigned by DL.CODE. See par. 1.1 for details.

4.1 IMAGE SETUP

PARAMETER	TYPE	STRING	DATA
Image Polarity Inversion	2	IMAGE_POLARITYx	false = Disabled true = Enabled
Exposure Time (μs) M450N all models	0	EXPOSURE_TIMEx	
Disabled			33 to 64000
Very High-Power Strobed			33 to 500
High-Power Strobed			33 to 2500
Continuous High-Power			33 to 400
M410N all models			
Disabled			10 to 64000
Always On			1000 to 64000
Very High-Power Strobed			10 to 500
High-Power Strobed			100 to 5000
Medium-Power Strobed			1000 to 64000
Continuous High-Power			10 to 300
M300N all models			
Disabled			10 to 50000
Always On			1000 to 50000
Very High-Power Strobed			1 to 500
High-Power Strobed			100 to 3300
M210N-2xx-xxx			
Disabled			10 to 26000
Always On			10 to 26000
Very High-Power Strobed			10 to 600
High-Power Strobed			100 to 5000
Medium-Power Strobed			1000 to 26000
M120-2xx-xxx			
Disabled			10 to 15000
Very High-Power Strobed			10 to 1000

PARAMETER	TYPE	STRING	DATA
	0	EXPOSURE_TIMEx	
M120-3xx-xxx		(continued)	
Disabled			1 to 15000
Very High-Power Strobed			1 to 1000

NOTE: The *Exposure Time* parameter is measured in **µs** and range of valid values depends on the Matrix model and the Internal Lighting setting.

Gain	0	GAINx	Range: 4 to 32 (For M120-2xx models) Range: 8 to 32 (For M120-3xx models) Range: 1 to 40 (For M210N all models) Range: 4 to 48 (For M300N all models) Range: 4 to 48 (For M410N-500-010 models) Range: 6 to 48 (For M410N-700-010 models) Range: 1 to 25 (For M450N all models)
Gain Multiplier	1	GAINMULTIPLIERx	0 = X1 1 = X2 2 = X3 3 = X4 4 = X5 5 = X6 6 = X7 7 = X8 8 = X9 9 = X10
Reading Distance (mm)	0	READING_DISTANCEx	Range: 40 to 120 mm (For M210N-2x5-x1x models) Range: 22 to 452 mm (For M300N-4x2-0xx models)

4.2 CODE SETUP

4.2.1 2D Codes

PARAMETER	TYPE	STRING	DATA
DATA MATRIX ECC 200			
Identical Codes	2	2D_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	2D_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Grey Level Calibration	2	2D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Code Color	1	2D_CODE_COLORx	0 = Black 1 = White 2 = Both Colors
Code Contrast	1	2D_CODE_CONTRASTx	0 = High 1 = Standard 2 = Low 3 = Very Low 4 = Automatic
Decoding Complexity	1	2D_DECODING_COMPLEXITYx	0 = Low 1 = Medium 2 = High 3 = Very High
Code Size	1	2D_CODE_SIZEx	1 = Free 2 = Squared 3 = Rectangular
Advanced Box Improvement	2	2D_ADVANCED_BOX_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
QR			
Identical Codes	2	2D_ID_CODESx	false = Disabled true = Enabled
Decoding Method	1	2D_DECODING_METHODx	0 = Standard 1 = Dot Peen Decoding
Image Mirroring	2	2D_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Grey Level Calibration	2	2D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Color	1	2D_CODE_COLORx	0 = Black 1 = White 2 = Both Colors
Code Contrast	1	2D_CODE_CONTRASTx	0 = High 1 = Standard 2 = Low 3 = Very Low 4 = Automatic

PARAMETER	TYPE	STRING	DATA
MICRO QR			
Identical Codes	2	2D_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	2D_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Grey Level Calibration	2	2D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Color	1	2D_CODE_COLORx	0 = Black 1 = White 2 = Both Colors
Code Contrast	1	2D_CODE_CONTRASTx	0 = High 1 = Standard 2 = Low 3 = Very Low 4 = Automatic

PARAMETER	TYPE	STRING	DATA
AZTEC			
Identical Codes	2	2D_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	2D_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Grey Level Calibration	2	2D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Color	1	2D_CODE_COLORx	0 = Black 1 = White 2 = Both Colors
Code Contrast	1	2D_CODE_CONTRASTx	0 = High 1 = Standard 2 = Low 3 = Very Low 4 = Automatic

PARAMETER	TYPE	STRING	DATA
MAXICODE			
Identical Codes	2	2D_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	2D_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Grey Level Calibration	2	2D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Color	1	2D_CODE_COLORx	0 = Black 1 = White 2 = Both Colors
Code Contrast	1	2D_CODE_CONTRASTx	0 = High 1 = Standard 2 = Low 3 = Very Low 4 = Automatic
Partial Code Tx	2	2D_PARTIAL_CODE_TXx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
DOTCODE			
Identical Codes	2	2D_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	2D_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Grey Level Calibration	2	2D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Color	1	2D_CODE_COLORx	0 = Black 1 = White 2 = Both Colors
Code Contrast	1	2D_CODE_CONTRASTx	0 = High 1 = Standard 2 = Low 3 = Very Low 4 = Automatic
Enable High Resolution	2	2D_HIGH_RESOLUTIONx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Min Pixel per Module	0	2D_MIN_DOT_DIAMETERx	Range: 0 to 48
Max Pixel per Module	0	2D_MAX_DOT_DIAMETERx	Range: 0 to 48
Fixed Code Position	2	2D_CONSTANT_POSITIONx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
2D CODE FILTER			
Wildcard	3	2D_WILDCARD_CHARx	Length: 1 character
Placeholder	3	2D_PLACEHOLDERx	Length: 1 character
Min. Number of Characters	0	2D_MIN_CHARx	Range 1 to 3116
Max. Number of Characters	0	2D_MAX_CHARx	Range 1 to 3116
Code Quality Standard[1]	1	2D_CODE_QUALITY_STx	0 = Overall 1 = Decode 2 = Contrast 3 = Modulation 4 = Decodability 5 = Minimum Edge Contrast 6 = Axial Non Uniformity 7 = Unused Error Correction 9 = Minimum Reflectance 10 = Defects 11 = Fixed Pattern Damage 12 = Grid Non Uniformity
Code Quality Index	1	2D_CODE_QUALITY_INDEXx	0 = Any 1 = Grade A 2 = Grade B 3 = Grade C 4 = Grade D

[1] Values depend on Code Type and Code Quality Standard. Not all parameters are valid for a given code.

4.2.2 1D Codes

PARAMETER	TYPE	STRING	DATA
PDF417			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
MICROPDF			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
CODE 128			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled
Start/Stop Characters Tx	2	1D_STARTSTOP_CHAR_TXx	false = Disabled true = Enabled
Characters Tx	2	1D_ CHAR_TXx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
GS1-128			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled
Start/Stop Characters Tx	2	1D_STARTSTOP_CHAR_TXx	false = Disabled true = Enabled
Characters Tx	2	1D_ CHAR_TXx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
CODE 39			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled
Character Set	1	1D_CHARACTER_SETx	0 = Standard 1 = Full ASCII
Code 32 Decoding	2	1D_CODE32_DECODINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
INTERLEAVED 2 OF 5			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled
Min Accepted Code Length	0	1D_INT25_MIN_ENABLED_LENx	Range 2 to 60

PARAMETER	TYPE	STRING	DATA
PHARMA			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Minimum Code Height (mm)	0	1D_MIN_CODE_HEIGHTx	Range 1 to 500
Code Aspect Ratio	1	1D_CODE_ASPECT_RATIOx	0 = Standard 1 = Low
Code Contrast	1	1D_CODE_CONTRASTx	0 = Standard 1 = Low
Small Codes Improvement	2	1D_SMALL_CODES_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
MSI			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
PLESSEY			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
STANDARD 2 OF 5			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
MATRIX 2 OF 5			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
CODABAR			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Start/Stop Characters Tx	2	1D_STARTSTOP_CHAR_TXx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
GS1 DATABAR EXPANDED			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
GS1 DATABAR EXPANDED STACKED			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
GS1 DATABAR LIMITED			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
GS1 DATABAR			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
GS1 DATABAR STACKED			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled

NOTE: GS1 DataBar option also enables GS1 DataBar Truncated symbology, while GS1 DataBar Stacked option also enables GS1 DataBar Stacked Omni-directional symbology.

PARAMETER	TYPE	STRING	DATA
CODE 93			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
EAN 13			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
EAN 8			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
UPCA / UPCE			
Identical Codes	2	1D_ID_CODESx	false = Disabled true = Enabled
Grey Level Calibration	2	1D_GREY_LEVEL_CALIBRATIONx	false = Disabled true = Enabled
Code Orientation	1	1D_CODE_ORIENTATIONx	0 = Horizontal 1 = Vertical 2 = Both
Advanced Box Improvement	2	1D_ADV_BOX_IMPROVEMENTx	false = Disabled true = Enabled
Subpixel Decoding Improvement	2	1D_SUBPIXEL_IMPROVEMENTx	false = Disabled true = Enabled
Narrow Quiet Zone	2	1D_NARROW_MARGINSx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
1D CODE FILTER			
Wildcard	3	1D_WILDCARD_CHARx	Length: 1 character
Placeholder	3	1D_PLACEHOLDERx	Length: 1 character
Min. Number of Characters	0	1D_MIN_NUMBER_CHARx	Range 1 to 3116
Max. Number of Characters	0	1D_MAX_NUMBER_CHARx	Range 1 to 3116

PARAMETER	TYPE	STRING	DATA
Code Quality Standard[2]	1	1D_CODE_QUALITY_STx	0 = Overall 1 = Decode 2 = Contrast 3 = Modulation 4 = Decodability 5 = Minimum Edge Contrast 6 = Axial Non Uniformity 7 = Unused Error Correction 9 = Minimum Reflectance 10 = Defects 11 = Fixed Pattern Damage 12 = Grid Non Uniformity
Code Quality Index	1	1D_CODE_QUALITY_INDEXx	0 = Any 1 = Grade A 2 = Grade B 3 = Grade C 4 = Grade D

[2] Values depend on Code Type and Code Quality Standard. Not all parameters are valid for a given code.

4.2.3 Postal Codes

PARAMETER	TYPE	STRING	DATA
Australia Post			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled
Code Orientation	1	POSTAL_CUSTOMER_FIELD_DECODINGx	0 = N Decoding Table 1 = C Decoding Table 2 = Bars Value (Decimal)

PARAMETER	TYPE	STRING	DATA
Royal Mail 45 State			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
KIX Code			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Japan Post			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
PLANET			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
POSTNET			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
POSTNET BB	2	POSTAL_POSTNETBBx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Intelligent Mail			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
Swedish Postal			
Identical Codes	2	POSTAL_ID_CODESx	false = Disabled true = Enabled
Image Mirroring	2	POSTAL_IMAGE_MIRRORINGx	false = Disabled true = Enabled

PARAMETER	TYPE	STRING	DATA
POSTAL CODE FILTER			
Wildcard	3	POSTAL_WILDCARD_CHARx	Length: 1 character
Placeholder	3	POSTAL_PLACEHOLDERx	Length: 1 character
Min. Number of Characters	0	POSTAL_MIN_NUM_CHARx	Range 1 to 3116
Max. Number of Characters	0	POSTAL_MAX_NUM_CHARx	Range 1 to 3116

4.2.4 Outputs

PARAMETER	TYPE	STRING	DATA
OUTPUTS			
Active State	1	LOCOUT_ACTIVE_STATE_x	0 = Active Closed 1 = Active Open
Line Function	1	LOCOUT_MODE_x	0 = General Purpose 2 = External Ill – Always On 3 = External Ill – Strobed
Number of Events	0	LOCOUT_NUM_OF_EVENTS_x	Range 1 to 255
Line Type	1	LOCOUT_LINE_TYPE_x	0 = NPN 1 = PNP 2 = Push-Pull

4.2.5 LEDs

PARAMETER	TYPE	STRING	DATA
GREEN SPOT			
Active State	1	OUT_ACTIVE_STATE_1	0 = Active Closed 1 = Active Open
Number of Events	0	OUT_NUM_OF_EVENTS_1	Range 1 to 255
RED SPOT			
Active State	1	OUT_ACTIVE_STATE_2	0 = Active Closed 1 = Active Open
Number of Events	0	OUT_NUM_OF_EVENTS_2	Range 1 to 255

A ASCII TABLE

CHARACTER CONVERSION TABLE								
CHAR	**DEC**	**HEX**	**CHAR**	**DEC**	**HEX**	**CHAR**	**DEC**	**HEX**
NUL	0	**00**	*	42	**2A**	U	85	**55**
SOH	1	**01**	+	43	**2B**	V	86	**56**
STX	2	**02**	,	44	**2C**	W	87	**57**
ETX	3	**03**	-	45	**2D**	X	88	**58**
EOT	4	**04**	.	46	**2E**	Y	89	**59**
ENQ	5	**05**	/	47	**2F**	Z	90	**5A**
ACK	6	**06**	0	48	**30**	[	91	**5B**
BEL	7	**07**	1	49	**31**	\	92	**5C**
BS	8	**08**	2	50	**32**	]	93	**5D**
HT	9	**09**	3	51	**33**	^	94	**5E**
LF	10	**0A**	4	52	**34**	_	95	**5F**
VT	11	**0B**	5	53	**35**	`	96	**60**
FF	12	**0C**	6	54	**36**	a	97	**61**
CR	13	**0D**	7	55	**37**	b	98	**62**
SO	14	**0E**	8	56	**38**	c	99	**63**
SI	15	**0F**	9	57	**39**	d	100	**64**
DLE	16	**10**	:	58	**3A**	e	101	**65**
DC1	17	**11**	;	59	**3B**	f	102	**66**
DC2	18	**12**	<	60	**3C**	g	103	**67**
DC3	19	**13**	=	61	**3D**	h	104	**68**
DC4	20	**14**	>	62	**3E**	i	105	**69**
NAK	21	**15**	?	63	**3F**	j	106	**6A**
SYN	22	**16**	@	64	**40**	k	107	**6B**
ETB	23	**17**	A	65	**41**	l	108	**6C**
CAN	24	**18**	B	66	**42**	m	109	**6D**
EM	25	**19**	C	67	**43**	n	110	**6E**
SUB	26	**1A**	D	68	**44**	o	111	**6F**
ESC	27	**1B**	E	69	**45**	p	112	**70**
FS	28	**1C**	F	70	**46**	q	113	**71**
GS	29	**1D**	G	71	**47**	r	114	**72**
RS	30	**1E**	H	72	**48**	s	115	**73**
US	31	**1F**	I	73	**49**	t	116	**74**
SPACE	32	**20**	J	74	**4A**	u	117	**75**
!	33	**21**	K	75	**4B**	v	118	**76**
"	34	**22**	L	76	**4C**	w	119	**77**
#	35	**23**	M	77	**4D**	x	120	**78**
$	36	**24**	N	78	**4E**	y	121	**79**
%	37	**25**	O	79	**4F**	z	122	**7A**
&	38	**26**	P	80	**50**	{	123	**7B**
'	39	**27**	Q	81	**51**	|	124	**7C**
(	40	**28**	R	82	**52**	}	125	**7D**
)	41	**29**	S	83	**53**	~	126	**7E**
			T	84	**54**	DEL	127	**7F**

B DIRECT COMMANDS FROM HOST

The following commands **are NOT** Host Mode Programming commands. They **do Not** require the Enter/Exit Host Mode commands and therefore **do Not** take place during a programming session. They are sent directly to the reader from the Host to perform their specific functions.

Communication Channels	
Allowed	**Not Allowed**
Serial	Modbus TCP embedded
Client TCP, port 1023	Ethernet/IP embedded
HMS Fieldbus	
Profinet-IO embedded	

ID-NET™ (MULTIDATA) SLAVE COMMANDS

The following command allows sending an "encapsulated" command string to a (Multidata) Slave #N through the Master (Multidata) device in an ID-NET™ reading system layout.

This command can be used to send Reading Phase ON, Acquisition Trigger and Reading Phase OFF strings (for One Shot and Phase Mode operating modes) to a Multidata Slave device in an ID-NET™ reading system layout.

The Matrix reader '**Send String To Slave Device #N**' command must have the following format:

<ESC> (<B0$_H$> ADDR STRING) <ESC>

Where:

- **ADDR**: Device Address
- **STRING**: String To Send (Length: 1 to 32)

ADDR is a character indicating address of the device in an ID-NET™ Master/Slave Multidata reading system layout:

ADDR = <30$_H$> + <Device Address> where:

Device Address = 0: Standalone device or Master Multidata ID-NET™ device
Device Address = 1 to 31: Slave Multidata ID-NET™ device
Device Address = 32: ID-NET™ broadcast address

This means:

ADDR = <30$_H$>: Standalone device or Master Multidata ID-NET™ device
ADDR = <31$_H$> to <4F$_H$>: Slave Multidata ID-NET™ device
ADDR = <50$_H$> ID-NET™ broadcast address

DATALOGIC

www.datalogic.com