MICROSCAN®

HawkEye 1500 Series Reference & Programmers Manual

Rev 2.4.1, Nov 2008

EM-40392-1V241

Latest Manual Version
For the latest version of this manual, see the Download Center on our web site at: www.microscan.com.

Technical Support
For technical support, email: helpdesk@microscan.com.

Microscan Systems, Inc.
1201 S.W. 7th Street
Renton, WA 98057
U.S.A.
Tel: 425 226 5700
Fax: 425 226 8250
helpdesk@microscan.com

Microscan Europe
Tel: 31 172 423360
Fax: 31 172 423366

Microscan Asia Pacific
R.O. Tel: 65 6846 1214
Fax: 65 6846 4641

Microscan Limited Warranty Statement and Exclusions

What Is Covered?
Microscan Systems Inc. warrants to the original purchaser that products manufactured by it will be free from defects in material and workmanship under normal use and service for a period of one year from the date of shipment. This warranty is specifically limited to, at Microscan's sole option, repair or replacement with a functionally equivalent unit and return without charge for service or return freight.

What Is Excluded?
This limited warranty specifically excludes the following: (1) Any products or parts that have been subject to misuse, neglect, accident, unauthorized repair, improper installation, or abnormal conditions or operations; (2) Any products or parts that have been transferred by the original purchaser; (3) Customer mis-adjustment of settings contrary to the procedure described in the Microscan Systems Inc. owners manual; (4) Upgrading software versions at customer request unless required to meet specifications in effect at the time of purchase; (5) Units returned and found to have no failure will be excluded; (6) Claims for damage in transit are to be directed to the freight carrier upon receipt. Any use of the product is at purchaser's own risk. This limited warranty is the only warranty provided by Microscan Systems Inc.

Tel: 425.226.5700 | Fax: 425.226.8250 | helpdesk@microscan.com

Contents

PREFACE **Welcome! xiii**

Purpose of This Manual **xiii**

Manual Conventions **xiii**

CHAPTER 1 Remote Commands Reference **1-1**

Basic Operation & Fine Tuning **1-1**

- Command Basics 1-1
  - Action Commands 1-1
  - Configuration Commands 1-2
- Getting Help on a Particular Command 1-3

Remote Commands **1-4**

- Command Conventions 1-10
- Terminal Shortcuts 1-10

Keyword Output Conditions **1-11**

APERTURE **1-12**

ASSIGN **1-13**

ASSIGNEX **1-14**

BARCODE **1-15**

BARCODE ENABLEMASK **1-16**

BARCONF **1-18**

BARHGT **1-19**

BARHPS **1-20**

BARLEN **1-21**

BARNUM **1-22**
BARPRB **1-23**
BARQZ **1-24**
BARTHRES **1-25**
BARVPS **1-26**
BARWDT **1-27**
BC_DECODE_NEAR_CENTER **1-28**
BC_UPC_AS_EAN **1-29**
BCCHKSUM **1-30**
BCCHKSUMEX **1-31**
BCDECFWD **1-32**
BCPOL **1-33**
BCSTART **1-34**
BEEP **1-35**
CALIBRATED **1-36**
  Single Light Configuration 1-36
  Multifunction Light Configuration 1-36
CALIBRATED_STRING **1-37**
  Single Light Configuration 1-37
  Multifunction Light Configuration 1-37
CAL_MEAN_LIGHT **1-38**
CAPTURE **1-39**
CELL SAMPLE **1-40**
CELL SIZE **1-41**
CELL_UNIT **1-42**
CELL_UNIT_REPORT **1-43**
COLS **1-44**
CONSOLE **1-45**
CONTRAST **1-46**
CONTRAST_REPORT **1-47**
CONTROL **1-48**
DECFL **1-49**
DELETE **1-50**
DHCP **1-51**
DHCPEX **1-52**
DIR **1-53**

DM_ALLOW_SEVERE_DAMAGE **1-54**
DM_ALLOW_STEEP_ANGLE **1-55**
DM_CELL_OUTLINE **1-56**
DM_DECODE_NEAR_CENTER **1-57**
DM_ENSURE_ROI **1-58**
DM_FINETUNE_RESERVED **1-59**
DM_IGNORE_SINGLE_EDGES **1-60**
DOMAIN **1-61**
DWELLTIME **1-62**
ECC **1-63**
EIPENABLE **1-64**
ENSURE CENTERED **1-65**
FINETUNE **1-66**
HEADER **1-69**
HEIGHT **1-70**
HELP **1-71**
ILLUMINATION **1-72**
INFORM **1-73**
INTENSITY_ENHANCE_ENABLED **1-74**
IO_MODE **1-75**
IOASSIGN **1-76**
IP **1-78**
IPCONFIG **1-79**
LEARN **1-80**
LEARNASSIST **1-82**
LEARNCANCEL **1-83**
LEARNEX **1-84**
LIGHT_CTL **1-85**
LOAD **1-86**
LOCFL **1-87**
MACADDR **1-88**
MATCH **1-89**
MATCH_LIST **1-91**
MATCH_LIST_ENABLE **1-93**
MATCHCOUNT **1-94**
MATCHEX **1-95**

MATCHFL **1-97**
MATCHSERIAL **1-99**
MATCHSTRING **1-100**
MORPHOLOGY **1-101**
NEGATIVE **1-102**
NO_QUIET_ZONE_CLUTTER **1-103**
NUMBC **1-104**
NUMDM **1-105**
OFFLINE **1-106**
OKDEC **1-107**
OKMATCH **1-109**
ONLINE **1-111**
ORIENT **1-112**
OUTPUT **1-113**
PARTQ **1-115**
PARTQCAPACITY **1-116**
PARTQCLEAR **1-117**
PARTQCOUNT **1-118**
PARTQFAIL **1-119**
PARTQFTP **1-120**
PARTQFTPEX **1-122**
PARTQREQ **1-123**
PARTQREQFORMAT **1-124**
PARTQSAVE **1-126**
PARTQSUM **1-127**
PHOTOEX **1-128**
PHOTOMETRY **1-129**
PID **1-130**
POLARITY **1-132**
POSITION_ENHANCE_ENABLED **1-133**
PROBE DIRECTION **1-134**
PROBE SPEED **1-135**
PROBE THRESHOLD **1-136**
PROMPT **1-137**
QRFINDER **1-138**
QUICSET **1-139**

QUICSETPLUS **1-141**
RATIO **1-142**
READ_LICENSE **1-143**
RELEASE **1-144**
REMOVE **1-145**
REPORTCANCEL **1-146**
REPORTREQ **1-147**
RESET **1-148**
RETRY **1-149**
RETRYEX **1-151**
ROI **1-153**
ROWS **1-154**
RTE **1-155**
RUN_SETTINGS **1-156**
SAVE **1-158**
SELF_TRIGGER **1-159**
SET_CRITERIA **1-160**
SET_LICENSE **1-162**
SHOW **1-163**
SIGOUT **1-164**
STATS **1-165**
STORE **1-166**
STYLE **1-167**
TARGET **1-168**
TARGET_CALIB_CONTRAST **1-169**
TARGET_CALIB_REFLECTANCE **1-170**
TBL **1-171**
TD **1-172**
TE **1-173**
TERMINAL ECHO **1-174**
THRESHOLD **1-175**
TIME **1-176**
TIMEOUT **1-177**
TIMESYNC **1-178**
TRAILER **1-179**
TRIG{GER} **1-181**

TRIGGER_REDIRECT_ENABLE **1-182**
TRIGTABLE **1-183**
TTY **1-186**
UDP_BROADCAST **1-187**
UNLEARN **1-188**
VERENABLE **1-189**
VERIFY **1-192**
VERIFY AIMDPM **1-195**
VERSION **1-196**
VERSTATUS **1-197**
VIEW **1-198**
VT **1-199**
WARP **1-200**
WAVELENGTH **1-201**
WIDTH **1-202**
XDIMENSION **1-203**

CHAPTER 2 ReadRunner Programming Reference **2-1**

Introduction **2-1**
Libraries **2-1**
Important Objects **2-2**
Example 1 — A Simple Monitoring Application **2-5**
    How This Works 2-6
    Taking This Example Further 2-6
    Summary 2-7
Example 2 — Handling Images and Reports **2-8**
    How This Works 2-10
    Taking This Example Further 2-10
Example 3 — Connecting to Two Cameras at Once **2-12**
Example 4 — Controlling the HawkEye™ 1500 **2-15**
Example 5 — Image Overlay Graphics and Auto Connection **2-20**
Example 6 — Custom ROI **2-25**
Example 7 — Verification Report **2-27**
Example 8 — Verification Data **2-29**
Example 9 — TCP Socket Communication **2-31**

Microscan HawkEye™ Reader Object Type Library **2-33**
- MiCoordinator 2-33
  - Methods 2-33
  - Events 2-33
- MiDevice 2-34
  - Methods 2-34
  - Events 2-36
- MiASCIIConnection 2-36
  - Methods 2-36
- MiReportConnection 2-37
  - Methods 2-37
- MiImageConnection 2-37
  - Methods 2-37
- Connection Events 2-38
  - Events 2-38

Microscan HawkEye™ Reader ActiveX Type Library **2-39**
- MiTimeplot 2-39
  - Methods 2-39
- MiImageView 2-40
  - Methods 2-40
  - Events 2-41
- MiDrawObj 2-42
  - Methods 2-42
- MiDC 2-42
  - Methods 2-42
  - Pen Style Constants for SetPen 2-43
- MiDrawHandles 2-43
  - Methods 2-44

CHAPTER 3 C++ Samples for TCP/IP Socket Communication **3-1**

Example 1 – PartQ Data Retrieval **3-1**

Example 2 – Output Socket Data Retrieval **3-3**

Index **Index-1**

PREFACE

Welcome!

Purpose of This Manual

This manual describes remote commands and ReadRunner remote libraries.

Manual Conventions

The following typographical conventions are used throughout this manual.

- Items emphasizing important information is **bolded**.
- Menu selections, menu items and entries in screen images are indicated as: Run (triggered), Modify..., etc.

CHAPTER 1 Remote Commands Reference

This chapter lists and describes the HawkEye™ 1500 remote commands. Commands are listed alphabetically.

Basic Operation & Fine Tuning

This section describes the basic operations and fine-tuning used with the HawkEye™ 1500.

Command Basics

The HawkEye™ 1500 responds to a series of ASCII string commands sent from the host computer over RS-232 and Ethernet ports. Each command must follow the designated command syntax for the camera to respond successfully.

The command sequence consists of the command word(s) followed by any additional keyword or parameters required, and terminated with a carriage return <CR> or <Enter>.

Action Commands

Action commands cause an action to take place. They have no settings that can be queried by the user. For example, the SAVE command is an action command that initiates the action of saving system parameters to the FLASH storage system on the camera. There is nothing to query about SAVE.

Configuration Commands

Configuration commands “set” a parameter or “query” the contents of a parameter. For example, the ROWS command sets the number of rows to a specific number.

All configuration commands follow the same syntax and format:

- A configuration command followed by a “?” means query the current value (return to the user the current value of the configuration command).

  For example:

  ```
  ROWS ?
  ```

  Would result in the following:

  ```
  ROWS AUTO     (or whatever the current value is at that time)
  ```

- A configuration command followed by a blank (no parameter) means show the user a quick 'help' on the command syntax.

  For example:

  ```
  ROWS <ENTER>
  ```

  Would result in the following:

  ```
  ROWS {rows|AUTO} — Set the number of rows to expected:
                     8-144, AUTO=default
  ```

- A configuration command followed by 'DEF[AULT]' means set the current value of the command to its default value. The default value being the value that was initially set from the Factory Default configuration.

  For example:

  ```
  ROWS DEF[AULT]
  ```

  Would result in the current value of ROWS to be set to its default value of AUTO.

To get help on a particular Action command, you would issue the word “HELP” followed by the command of interest.

For example:

```
HELP RESET
```

Would result in the following:

RESET [opt] - Reset the unit - Reset user settings type Help Reset v[erbose]

If more detailed help information is desired (and available), issue:

```
HELP RESET V[erbose]
```

Which would result in printing more verbose help information.

RESET [opt] - Reset the unit - Reset user settings type Help Reset v[erbose]

RESET FACTORY - Reset all settings to factory defaults. This includes all connectivity options. The camera will be set to DHCP Y, camera name 'HawkEye', default IP address, CONSOLE 0, BEEP Y. Essentially, all decoder parameters, acquisition parameters, and BOOTPARAMS stuff (except for the MAC address) would be reset to factory defaults. All would be saved to flash.

RESET DECODER - This would be the same as LOAD 99 and would reset the decoder (only) to defaults. Photometry would also be reset to Auto. This would save to flash, whereas LOAD 99 would not.

RESET APPMODE - This would reset the acquisition parameters and lighting (essentially everything in the Application Mode dialog of RR. This would save to flash.

RESET HARD - This reboots the unit.

RESET SOFT - This goes offline, restores the registry to factory default, and goes back online.

RESET ALL - This does a 'reset factory', saves to flash, and 'reset hard'.

For the most part, commands are NOT case sensitive. When entering strings (as required by a particular command), the string must be entered in quotes. Characters in quotes are case sensitive. You get what you type.

Getting Help on a Particular Command

If you are not using ReadRunner to control the camera or, if you are and wish to command the camera via the Command Terminal Window, the following describes the available options:

```
HELP <ENTER>
```

Displays a listing of all the available commands divided into eight categories.

Once you see the command that is most likely the one of interest, you can get quick help by typing:

```
HELP {command}
```

or verbose help on the command by typing:

```
HELP {command} V[erbose]
```

Additionally, you can get help on configuration commands by typing the command with no parameter information provided.

For example:

```
HELP ROWS <ENTER>
```

Would result in the following:

```
ROWS {rows|AUTO} - Set the number of rows to expected:
                                  8-144, AUTO=default
```

Remote Commands

Table 1–1 through Table 1–8 group commands based on function.

TABLE 1–1. Housekeeping Query Commands

Command	Page
HELP	“HELP” on page 1-71
QUICSET	“QUICSET” on page 1-139
SHOW	“SHOW” on page 1-163
STATS	“STATS” on page 1-165
VERSION	“VERSION” on page 1-196

TABLE 1–2. **System Operation Control Commands**

Command	Page
CAPTURE	“CAPTURE” on page 1-39
CONTROL	“CONTROL” on page 1-48
OFFLINE	“OFFLINE” on page 1-106
ONLINE	“ONLINE” on page 1-111
RELEASE	“RELEASE” on page 1-144
RESET	“RESET” on page 1-148
RUN_SETTINGS	“RUN_SETTINGS” on page 1-156
SAVE	“SAVE” on page 1-158
VT	“VT” on page 1-199

TABLE 1–3. **Serial Reporting Configuration Commands**

Command	Page
DECFL	“DECFL” on page 1-49
HEADER	“HEADER” on page 1-69
IOASSIGN	“IOASSIGN” on page 1-76
LOCFL	“LOCFL” on page 1-87
MATCH_LIST	“MATCH_LIST” on page 1-91
MATCH_LIST_ENABLE	“MATCH_LIST_ENABLE” on page 1-93
MATCHFL	“MATCHFL” on page 1-97
OKDEC	“OKDEC” on page 1-107
OKMATCH	“OKMATCH” on page 1-109
OUTPUT	“OUTPUT” on page 1-113
RTE	“RTE” on page 1-155
SET_CRITERIA	“SET_CRITERIA” on page 1-160
TRAILER	“TRAILER” on page 1-179
VERENABLE	“VERENABLE” on page 1-189
VERIFY	“VERIFY” on page 1-192
VERIFY AIMDPM	“VERIFY AIMDPM” on page 1-195
VERSTATUS	“VERSTATUS” on page 1-197

TABLE 1–4. System Configuration Commands

Command	Page
APERTURE	"APERTURE" on page 1-12
CAL_MEAN_LIGHT	"CAL_MEAN_LIGHT" on page 1-38
CALIBRATED	"CALIBRATED" on page 1-36
CALIBRATED_STRING	"CALIBRATED_STRING" on page 1-37
CELL_UNIT	"CELL_UNIT" on page 1-42
CELL_UNIT_REPORT	"CELL_UNIT_REPORT" on page 1-43
CONTRAST	"CONTRAST" on page 1-46
CONTRAST_REPORT	"CONTRAST_REPORT" on page 1-47
DWELLTIME	"DWELLTIME" on page 1-62
ILLUMINATION	"ILLUMINATION" on page 1-72
IO_MODE	"IO_MODE" on page 1-75
LIGHT_CTL	"LIGHT_CTL" on page 1-85
MATCH	"MATCH" on page 1-89
MATCHEX	"MATCHEX" on page 1-95
PHOTOMETRY	"PHOTOMETRY" on page 1-129
QUICSETPLUS	"QUICSETPLUS" on page 1-141
RETRY	"RETRY" on page 1-149
SELF_TRIGGER	"SELF_TRIGGER" on page 1-159
SIGOUT	"SIGOUT" on page 1-164
TARGET	"TARGET" on page 1-168
TARGET_CALIB_CONTRAST	"TARGET_CALIB_CONTRAST" on page 1-169
TARGET_CALIB_REFLECTANCE	"TARGET_CALIB_REFLECTANCE" on page 1-170
TBL	"TBL" on page 1-171
TD	"TD" on page 1-172
TE	"TE" on page 1-173
TRIG	"TRIG{GER}" on page 1-181

TABLE 1–4. System Configuration Commands (Continued)

Command	Page
TRIGGER_REDIRECT_EN ABLE	“TRIGGER_REDIRECT_ENABLE” on page 1-182
TRIGTABLE	“TRIGTABLE” on page 1-183
UDP_BROADCAST	“UDP_BROADCAST” on page 1-187
WAVELENGTH	“WAVELENGTH” on page 1-201

TABLE 1–5. Communication Control Related Commands

Command	Page
DHCP	“DHCP” on page 1-51
DOMAIN	“DOMAIN” on page 1-61
EIPENABLE	“EIPENABLE” on page 1-64
INFORM	“INFORM” on page 1-73
IP	“IP” on page 1-78
IPCONFIG	“IPCONFIG” on page 1-79
PROMPT	“PROMPT” on page 1-137
TERMINAL ECHO	“TERMINAL ECHO” on page 1-174
TTY	“TTY” on page 1-186

TABLE 1–6. Decoder Configuration Commands (PID Info)

Command	Page
BARCODE	“BARCODE” on page 1-15
BARCODE ENABLEMASK	“BARCODE ENABLEMASK” on page 1-16
BARCONF	“BARCONF” on page 1-18
BARHGT	“BARHGT” on page 1-19
BARHPS	“BARHPS” on page 1-20
BARLEN	“BARLEN” on page 1-21
BARNUM	“BARNUM” on page 1-22
BARPRB	“BARPRB” on page 1-23
BARQZ	“BARQZ” on page 1-24
BARTHRES	“BARTHRES” on page 1-25

TABLE 1–6. Decoder Configuration Commands (PID Info) (Continued)

Command	Page
BARVPS	“BARVPS” on page 1-26
BARWDT	“BARWDT” on page 1-27
CELL SAMPLE	“CELL SAMPLE” on page 1-40
CELL SIZE	“CELL SIZE” on page 1-41
COLS	“COLS” on page 1-44
ECC	“ECC” on page 1-63
ENSURE	“ENSURE CENTERED” on page 1-65
FINETUNE	“FINETUNE” on page 1-66
HEIGHT	“HEIGHT” on page 1-70
MATCHCOUNT	“MATCHCOUNT” on page 1-94
MATCHSERIAL	“MATCHSERIAL” on page 1-99
MATCHSTRING	“MATCHSTRING” on page 1-100
MORPHOLOGY	“MORPHOLOGY” on page 1-101
NEGATIVE	“NEGATIVE” on page 1-102
NUMBC	“NUMBC” on page 1-104
NUMDM	“NUMDM” on page 1-105
ORIENT	“ORIENT” on page 1-112
POLARITY	“POLARITY” on page 1-132
PROBE	“PROBE DIRECTION” on page 1-134
RATIO	“RATIO” on page 1-142
ROI	“ROI” on page 1-153
ROWS	“ROWS” on page 1-154
STYLE	“STYLE” on page 1-167
THRES	“THRESHOLD” on page 1-175
TIMEOUT	“TIMEOUT” on page 1-177
WARP	“WARP” on page 1-200
WIDTH	“WIDTH” on page 1-202

TABLE 1–7. **Product Identification (PID) Control Configuration Commands**

Command	Page
ASSIGN	"ASSIGN" on page 1-13
ASSIGNEX	"ASSIGNEX" on page 1-14
DELETE	"DELETE" on page 1-50
DIR	"DIR" on page 1-53
LEARN	"LEARN" on page 1-80
LEARNASSIST	"LEARNASSIST" on page 1-82
LEARNCANCEL	"LEARNCANCEL" on page 1-83
LOAD	"LOAD" on page 1-86
PID	"PID" on page 1-130
REMOVE	"REMOVE" on page 1-145
STORE	"STORE" on page 1-166
UNLEARN	"UNLEARN" on page 1-188
VIEW	"VIEW" on page 1-198

TABLE 1–8. **Part Queue Configuration/Control Commands**

Command	Page
PARTQ	"PARTQ" on page 1-115
PARTQCAPACITY	"PARTQCAPACITY" on page 1-116
PARTQCLEAR	"PARTQCLEAR" on page 1-117
PARTQCOUNT	"PARTQCOUNT" on page 1-118
PARTQFAIL	"PARTQFAIL" on page 1-119
PARTQFTP	"PARTQFTP" on page 1-120
PARTQFTPEX	"PARTQFTPEX" on page 1-122
PARTQREQ	"PARTQREQ" on page 1-123
PARTQREQFORMAT	"PARTQREQFORMAT" on page 1-124
PARTQSAVE	"PARTQSAVE" on page 1-126
PARTQSUM	"PARTQSUM" on page 1-127

Command Conventions

The following conventions apply to remote command descriptions:

- { } — Indicates that the information is required.
- [ ] — Indicates that the information is optional.

Terminal Shortcuts

TABLE 1–9. **Terminal Shortcuts**

Shortcut	What It Does
Ctrl+O	Turn off report output.
Ctrl+P	Toggle the state of the prompt and echo.
Ctrl+Q	Turn off command output.
Ctrl+R	Release control of the unit and put it back online.
Ctrl+S	Take control of the unit and bring it offline (forcibly).
Ctrl+T	Dump the heartbeat string to the terminal.
Ctrl+U	Toggle the display of report strings on the terminal.
Ctrl+W	Turn on command output.

Keyword Output Conditions

Table 1–10 lists the valid keywords for the listed commands.

TABLE 1–10. Keyword Output Conditions

Local Permutations	ANGLE	CHECKSUM	DATA	DATAHEX	DETAILED	PID#	TIMESTAMP	VERI_1_IAQG*	VERI_DETAIL	VERI_FORMATTED	VERI_GRADE	VERI_STATUS*
DECFL	X	X					X					
HEADER		X					X					
LOCFL		X					X					
MATCHFL	X	X	X	X	X	X	X	X	X	X	X	X
OKDEC	X	X	X	X	X	X	X	X	X	X	X	X
OKMATCH	X	X	X	X	X	X	X	X	X	X	X	X
RTE		X					X					
TRAILER		X					X					

Note: You must have a verification license for VERI_1_IAQG and VERI_STATUS.

APERTURE

This command sets the synthesized aperture size in mils. It is required for the ISO/IEC 15415 verification process.

Syntax

APERTURE {size}

Where:

size — The synthesized aperture size in mils:

- 4 - 20
- AUTO (set to be 80% of grid size); this is the default

ASSIGN

This command assigns the learned decoder parameters from the indicated PID# to a trigger. When the specified trigger occurs (to initiate the read and decode of an image), the PID parameters are loaded into the decoder while the image is being acquired. This allows multiple learned symbols to be used when online.

Note: We recommend you use the ASSIGNEX command in applications instead of ASSIGN, since we will be removing ASSIGN in future versions.

See also “LEARN” on page 1-80 and “STORE” on page 1-166.

Syntax

ASSIGN {pid} {trig}

Where:

pid — The product ID number:

Range: 1 - 15, 99 (default)

trig — The trigger number:

Range: 0 - 27

ASSIGNEX

This command assigns the learned decoder parameters from the indicated PID# to a trigger. When the specified trigger occurs (to initiate the read and decode of an image), the PID parameters are loaded into the decoder while the image is being acquired. This allows multiple learned symbols to be used when online.

This command allows the setting of a single trigger. For example:

```
ASSIGNEX 4 {pid}
```

This command allows the inquiry of a single trigger. For example:

```
ASSIGNEX 4 ?
```

Note: See also “LEARN” on page 1-80 and “STORE” on page 1-166.

Syntax

ASSIGNEX {trig} {pid}

Where:

pid — The product ID number:

Range: 1 - 15, 99 (default)

trig — The trigger number:

Range: 0 - 27

BARCODE

This command controls all 1-D symbologies except BC412, Postnet, Pharmacode, and QR Code.

Syntax

BAR[CODE] {ON | OFF}

BARCODE ENABLEMASK

This command selects a combination of 1-D symbologies.

Syntax

BARCODE ENABLEMASK {mask}

Where:

mask — Is any of the following:

- bit 0 — Code 128
- bit 1 — Code 39
- bit 2 — Code 93
- bit 3 — I2of5
- bit 4 — Codabar
- bit 5 — UPC EN8
- bit 6 — UPC E
- bit 7 — UPC Supp
- bit 8 — PDF417
- bit 9 — BC412
- bit 10 — Postnet
- bit 11 — Pharmacode
- bit 12 — RSS-14
- bit 13 — RSS-LIMITED
- bit 14 — RSS-EXPANDED
- bit 15 — RSS-STACKED
- bit 16 — RSS-COMPOSITE
- bit 17 — QR-CODE

You can turn on individually many barcode types using the following commands:

- BAR128 {? | ON | OFF}
- BAR25 {? | ON | OFF}
- BAR39 {? | ON | OFF}
- BAR412 {? | ON | OFF}
- BAR93 {? | ON | OFF}
- BARABAR {? | ON | OFF}
- BARUPC {? | ON | OFF}
- BARUPE {? | ON | OFF}
- BARUPS {? | ON | OFF}

BARCONF

This command sets the barcode confidence level. For barcodes that do not use checksum (Code 39, I 2 of 5, Codabar, and BC412), the threshold value can help reduce potential mis-decodes. When the value is raised, a certain number of decodes must agree before the decode is declared successful. On the other hand, if the value is set too high, then a normally decodable but imperfect barcode may not be decoded.

Syntax

BARCONF {level}

Where:

level — Is the confidence level:

Range: 0 - 100
Default: 0 (Any possible decode is accepted)

BARHGT

This command sets the minimum and maximum height of a barcode.

Syntax

BARHGT {opt1} {opt2}

Where:

opt1 — Is either of the following:

- MIN (default: 4 pixels)
- MAX (default: 400 pixels)

opt2 — Is the height of the barcode:

Range: 4 - 400

BARHPS

This command sets the horizontal probe spacing in pixels.

Note: See also “BARVPS” on page 1-26.

Syntax

BARHPS {spacing}

Where:

spacing — Is the horizontal probe spacing:

Range: 8 - 256 pixels
Default: 16 pixels

BARLEN

This command sets the minimum and maximum length of a barcode. If barcodes of different type or size need to be decoded, each barcode should be learned separately and the barcode properties recorded by the user. To read them all, the barcode properties list should be manually set to contain all the barcode properties learned from each barcode.

Note: See also “BARWDT” on page 1-27.

Syntax

BARLEN {opt1} {opt2}

Where:

opt1 — Is either of the following:

- MIN (default: 20 pixels)
- MAX (default: 600 pixels)

opt2 — Is the length of the barcode:

Range: 20 - 600

BARNUM

This command sets the minimum and maximum number of barcodes to be decoded.

Syntax

BARNUM {opt1} {opt2}

Where:

opt1 — Is either of the following:

- MIN (default: 2)
- MAX (default: 100)

opt2 — Is the number of barcodes to be decoded:

Range: 2 - 100

BARPRB

This command sets the barcode search method.

Syntax

BARPRB {method}

Where:

method — Is one of the following:

- HORI
- HORI_VERT (default)
- VERT
- VERT_HORI

BARQZ

This command sets the barcode quiet zone length. If for any reason the quiet zone is less in the image due to camera setup restrictions, you can lower the length in order for the algorithm to accept shorter space as quiet zone. On the other hand, if the barcode is big in the FOV such that some space between two adjacent bars exceeds the value specified, then it is possible that this space will be detected as possible quiet zone. To avoid this situation, increase the value of length.

Note: Do not adjust this value unless you understand how the change will affect the system performance.

Syntax

BARQZ {length}

Where:

length — Is the quiet zone length:

Range: 5 - 100
Default: 12

BARTHRES

This command sets the barcode threshold out of 255 grayscale.

Syntax

BARTHRES {opt}

Where:

opt — Is the threshold:

Range: 5 - 100 grayscale levels
Default: 18

BARVPS

This command sets the vertical probe spacing in pixels.

Note: See also “BARHPS” on page 1-20.

Syntax

BARVPS {opt}

Where:

opt — Is the vertical probe spacing:

Range: 8 - 256 pixels
Default: 16 pixels

BARWDT

This command sets the minimum and maximum width of a barcode.

Note: See also “BARLEN” on page 1-21.

Syntax

BARWDT {opt1} {opt2}

Where:

opt1 — Is either of the following:

- MIN (default: 1 pixels)
- MAX (default: 100 pixels)

opt2 — Is the width of the barcode:

Range: 1 - 100

BC_DECODE_NEAR_CENTER

See “FINETUNE” on page 1-66.

BC_UPC_AS_EAN

This command enables/disables the output of UPC as EAN. The default is OFF.

Syntax

BC_UPC_AS_EAN {ON | OFF}

BCCHKSUM

This command enables/disables BC412 check sum checking.

Syntax

BCCHKSUM {opt}

Where:

opt — Is either of the following:

- ON
- OFF (default)

BCCHKSUMEX

This command enables/disables the optional Check Sum checking and display.

Syntax

BCCHKSUMEX {opt1} {opt2} {opt3}

Where:

opt1 — Is the code type that supports Check Sum:

- BAR25
- BAR39
- BAR412

opt2 — Is whether or not to turn on Check Sum checking:

- ON — Turn on Check Sum checking
- OFF — Turn off Check Sum checking

opt3 — Is whether or not to remove the display of the Check Sum in the output string:

- ON — Do not display the Check Sum in the output string
- OFF — Display the Check Sum in the output string

BCDECFWD

This command allows BC412 to be decoded forward (left to right) or backward (right to left).

Syntax

BCDECFWD {opt}

Where:

opt — Is either of the following:

- – ON (left to right); this is the default
- – OFF (right to left)

BCPOL

This command sets the expected BC412 polarity.

Syntax

BCPOL {polarity}

Where:

polarity — Is one of the following:

- DL — Is Dark on Light
- LD — Is Light on Dark
- AUTO — Autoselect (default)

BCSTART

This command enables/disables BC412 start/stop checking.

Syntax

BCSTART {opt}

Where:

opt — Is either of the following:

- ON
- OFF (default)

BEEP

This command enables/disables decode results beeping.

Syntax

BEEP {opt}

Where:

opt — Is either of the following:

- Y
- N (default)

CALIBRATED

This command reports the current state of camera light calibration. This is a read-only command.

Single Light Configuration

Syntax

CALIBRATED ?

Multifunction Light Configuration

If no light type is specified, the returned result is the overall state of all light channels on the camera. The individual state of camera calibration for a specified light is returned when Light Type is specified in the command. The overall state is False if at least one of the individual states is False, and is True when all individual states are True.

Syntax

CALIBRATED [optional light type] ?

Where:

Optional light type — Is one of the following:

OFF	
90	(90°)
D	(Dome Light)
45Q	(45° Quad Lights)
30T_NS	(30° Two Lights North South)
30T_EW	(30° Two Lights East West)
30Q	(30° Quad Lights)
30S_N	(30° Single Light North)
30S_E	(30° Single Light East)
30S_S	(30° Single Light South)
30S_W	(30° Single Light West)

CALIBRATED_STRING

This command reports or sets the string or date for the light source. The string inside the quotes can be an empty string or up to 251 characters long.

Single Light Configuration

Syntax

CALIBRATED_STRING {? | “string to save on camera”}

Multifunction Light Configuration

This command reports or sets the calibration status string or date for specified or current light source. If no light type is specified, the returned result is the overall state of all light channels on the camera.

Syntax

CALIBRATED_STRING [optional light type] {? | “string to save on camera”}

Where:

Optional light type — Is one of the following:

OFF	
90	(90°)
D	(Dome Light)
45Q	(45° Quad Lights)
30T_NS	(30° Two Lights North South)
30T_EW	(30° Two Lights East West)
30Q	(30° Quad Lights)
30S_N	(30° Single Light North)
30S_E	(30° Single Light East)
30S_S	(30° Single Light South)
30S_W	(30° Single Light West)

CAL_MEAN_LIGHT

This command reports the calibration mean light value (R_cal) of camera calibration for current light source. This is a read-only command.

Syntax

CAL_MEAN_LIGHT ?

CAPTURE

This command sets the source of the images to the decoder.

Syntax

CAPTURE {source}

Where:

source — Is one of the following:

- CAMERA — Captures images through the camera. This is the default.
- DIAGNOSTIC — The FPGA generates a ramp image. Used for internal testing.
- FILES — Images are downloaded into the unit. Used for testing and debugging.

CELL SAMPLE

This command sets cell sampling. ReadRunner program samples and averages portions of each cell to determine whether the cell is light or dark.

Syntax

CELL SAMPLE {pixels}

Where:

pixels — Is the number of pixels:

Range: 1 - 7 pixels
Default: 5 pixels

CELL SIZE

This command sets cell size.

Syntax

CELL SIZE {pixels}

Where:

pixels — Is the cell size:

Range: 2 - 20 pixels
Default: 6 pixels

CELL_UNIT

This command reports the value of the cell unit multiplier by calibration for the current light source. This is a read-only command.

Syntax

CELL_UNIT ?

CELL_UNIT_REPORT

This command sets the cell unit report type either in pixels or mils.

Syntax

CELL_UNIT_REPORT {pixels | mils}

COLS

This command specifies the number of columns to expect in the Data Matrix symbol. The algorithm uses the value specified without having to re-compute it from image to image.

Note: See also “ROWS” on page 1-154.

Syntax

COLS {columns}

Where:

columns — Is one of the following:

- cols — Range: 8 - 144
- AUTO — Autoselect (default)

CONSOLE

This command sets the vxWorks console to the serial port for debugging.

Note: If the HawkEye™ 1500 is in the vxWorks shell (CONSOLE 2) and needs to be returned to the standard command interface (CONSOLE 1), the following should be typed at the vxWorks prompt (“->”):

```
cmd “control”
cmd “console 1”
```

Syntax

CONSOLE {n}

Where:

n — Is the serial port number. The range is 0 - 2:

- 0 or 1 — The standard command console
- 2 — The vxWorks debugging console

CONTRAST

This command displays the results of the LEARN CALIBRATE command (maximum and minimum contrast range). This is a read-only command.

Syntax

CONTRAST ?

For example:

```
CONTRAST ?
```

yields:

```
CONTRAST 255 0
```

CONTRAST_REPORT

This command sets the cell unit report type.

Syntax

CONTRAST_REPORT {UNCALIBRATED | REFLECTANCE_CALIBRATED | SELF_CALIBRATED}

CONTROL

This command takes control of the camera to allow you to alter parameters. Use the RELEASE command to release control of the camera.

Syntax

```
CONTROL
```

DECFL

This command customizes the unsuccessful decode output of the HawkEye™ 1500.

Syntax

DECFL {cntrl} [""string""] [{Hdrcntrl} {Trlcntrl} {Beepcntrl}]

Where:

cntrl — Is the DECFL message active (Y or N)?

string — (1 - 39 characters); the string may include keyword substitution:

- ANGLE — This keyword reports the Data Matrix angle in degrees from 0° to 359°.
- CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.
- TIMESTAMP — This keyword will be substituted with the current date and time.

Note: The following parameters are optional. However, it you enter one of them, you must enter all of them at the same time.

Hdrcntrl — Print the header before the DECFL string (Y or N)?

Trlcntrl — Print the trailer before the DECFL string (Y or N)?

Beepcntrl — Sound this many beeps (0 - 3) when this report is issued. The default is 0.

DELETE

This command deletes (invalidates) the stored PID.

Note: To save this deletion to flash, you must issue the SAVE command.

Syntax

DELETE {pid#}

Where:

pid# — Is the PID number:

Range: 1 - 15

DHCP

This command enables/disables DHCP.

Syntax

DHCP {opt}

Where:

opt — Is either of the following:

- Y (default)
- N

DHCPEX

This command enables/disables DHCP and sets up Automatic Private IP Addressing (APIPA), which assigns a camera IP address when no DHCP can be found.

Syntax

DHCP {Y | N} {APIPAIP} {APIPAMASK} {DHCPRETRYTIMER}

Where:

Y | N — Enable/disable DHCP.

APIPAIP — The template address that will be used by APIPA to form its new IP address in the absence of a DHCP server. When used with APIPAMASK, the unmasked portion of the address (.0.0) will be replaced with randomly generated address fields. This randomly created address was pre-qualified as not being seen to exist on the network in order to prevent duplicate assignment.

DHCPRETRYTIMER — (0 - 6000) The DHCP Retry Timer.

- When set to a number other than 0, the camera will continue to request a DHCP address forever, every “DHCP Reset Timer” number of seconds.
- When set to zero, if no DHCP server offers an address on the first request, the camera will fail over to APIPA address creation.

DIR

This command lists all the valid PIDs. This is a read-only command.

Syntax

DIR

DM_ALLOW_SEVERE_DAMAGE

See “FINETUNE” on page 1-66.

DM_ALLOW_STEEP_ANGLE

See “FINETUNE” on page 1-66.

DM_CELL_OUTLINE

See “FINETUNE” on page 1-66.

DM_DECODE_NEAR_CENTER

See “FINETUNE” on page 1-66.

DM_ENSURE_ROI

See “FINETUNE” on page 1-66.

DM_FINETUNE_RESERVED

See “FINETUNE” on page 1-66.

DM_IGNORE_SINGLE_EDGES

See “FINETUNE” on page 1-66.

DOMAIN

This command sets the default TCP/IP domain. For example:

```
acut.com
```

Syntax

DOMAIN {domain name}

DWELLTIME

This command sets the dwell time of the decoder.

Syntax

DWELLTIME {opt}

Where:

opt — Is the time in milliseconds:

Range: 0 - 60 ms
Default: 5 ms

ECC

This command sets Data Matrix expected Error Correction level. ECC 200 employs Reed-Solomon error correction and is recommended for new applications. Other ECC levels (000, 050, 080, 100, 140) use the convolutional error encoding. ECC250 is customer-specific coding. The default value SPEC allows the algorithm to decode ECC000, 050, 080, 100, 140, and 200 that are included in the AIM/ISO Data Matrix Specification. If not specified, the older non-standard Data Matrix such as ECC120 can also be decoded.

Syntax

ECC {level}

Where:

level — Is one of the following:

- 0
- 50
- 80
- 100
- 140
- 200
- AUTO
- Spec (default)

EIPENABLE

This command enables/disables Ethernet/IP communications.

Note: You must reboot the camera after enabling/disabling Ethernet/IP.

Syntax

EIPENABLE {opt}

Where:

opt — Is either of the following:

- Y or ON — Enable Ethernet/IP.
- N or OFF — Disable Ethernet/IP (default)

ENSURE CENTERED

This command specifies whether or not the image needs to be centered.

Syntax

ENSURE CENT[ERED] {opt1} {opt2}

Where:

opt1 — Is either of the following:

- BARCODE — The camera decodes barcodes when the laser (central) line passes through all the bars and spaces.
- DM — The camera decodes Data Matrix when the center point of image is inside the Data Matrix

opt2 — Is either of the following:

- ON
- OFF (default)

FINETUNE

This command controls all decoder fine tuning options. It allows you to select the combination of decoder fine tuning options. By default, FINETUNE is ON.

Syntax

FINETUNE {ON | OFF}
FINETUNE ENABLEMASK {mask}

- bit 0 — POSITION_ENHANCE_ENABLED {ON/OFF} Default=Off

  Enabling this option allows the algorithm to locate more precisely the four corners of the Data Matrix, and may improve the decode capability. This mode is left here for compatibility purposes.

- bit 1 — INTENSITY_ENHANCE_ENABLED {ON/OFF} Default=Off

  Designed to overcome dramatic intensity variation over the matrix border area. For example, certain poorly marked Data Matrices may have some cells that are almost invisible compared to the rest of the cells. Without enabling the option, the system may issue a status/error code, indicating a certain edge cannot be found. Enabling the option will help read this type of Data Matrix more consistently.

- bit 2 — NO_QUIET_ZONE_CLUTTER {ON/OFF} Default=Off

  Enabling this option speeds up the reading process when Data Matrix has sufficient quiet zone. With sufficient quiet zone, a Data Matrix with irregular, curved, or distorted border(s) can be read more efficiently with the option checked. Typically, the border problem is associated with low quality inkjet or dot peen marks or when the Data Matrix is imaged at an compound angle.

- bit 3 — DM_IGNORE_SINGLE_EDGES {ON/OFF} Default=Off

  Enabling this option allows the software to ignore single edges surrounding the Data Matrix in the image.

- bit 4 — DM_FINETUNE_RESERVED {ON/OFF} Default=Off

  This option is reserved for future implementation.

- bit 5 — DM_ALLOW_STEEP_ANGLE {ON/OFF} Default=Off

In some Data Matrix reading applications, it is not possible to set up the camera such that the focal plane is in parallel with the surface of the Data Matrix label. When the focal plane and the label surface form a steep angle, the Data Matrix in the image will have severe geometrical distortion. Use the following steps to read severely distorted Data Matrix in any orientation:

a. Unlearn.

b. Set Number of Data Matrices to Decode to 1.

c. In Finetune Method, enable/check Allow Steep Angle. Now, the system is ready to read distorted Data Matrix in Run mode.

- bit 6 — DM_ALLOW_SEVERE_DAMAGE {ON/OFF} Default=Off

  Enabling this option increases the robustness of the software in reading Data Matrix with severe border damages. To use it, first perform a successful Learn on a less damaged label. Then, enable (check) the option to read labels with more border damages in Run mode.

- bit 7 — DM_ENSURE_ROI {ON/OFF} Default=Off

  Enabling this option ensures that no Data Matrix is located unless it is fully inside the ROI.

- bit 8 — DM_DECODE_NEAR_CENTER {ON/OFF} Default=Off

  Enabling this option allows a Data Matrix or barcode to be decoded only if the center of the ROI is inside the Data Matrix or barcode.

- bit 9 — BC_DECODE_NEAR_CENTER {ON/OFF} Default=Off

  Enabling this option allows a Data Matrix or barcode to be decoded only if the center of the ROI is inside the Data Matrix or barcode.

- bit 10 — DM_CELL_OUTLINE {ON/OFF} Default=Off

  Enabling this option helps the algorithm decode a Data Matrix with outlined cells only. In this case, the On and Off cells have little or no contrast but they are separated by edges of the cells.

Use the following command to set more than one flag at a time:

```
FINETUNE ENABLEMASK 0x26
```

where the bits for each selected option are set in the mask.

Alternatively, you can set each of the individual FINETUNE parameters singly. For example:

BC_DECODE_NEAR_CENTER {ON/OFF}

DM_ALLOW_SEVERE_DAMAGE {ON/OFF}

DM_ALLOW_STEEP_ANGLE {ON/OFF}

DM_CELL_OUTLINE {ON|OFF} {ON/OFF}

DM_DECODE_NEAR_CENTER {ON/OFF}

DM_ENSURE_ROI {ON/OFF}

DM_FINETUNE_RESERVED {ON/OFF}

DM_IGNORE_SINGLE_EDGES {ON/OFF}

INTENSITY_ENHANCE_ENABLED {ON/OFF}

NO_QUIET_ZONE_CLUTTER {ON/OFF}

POSITION_ENHANCE_ENABLED {ON/OFF}

HEADER

This command sets the Decode Output header to the indicated string. Use this command to specify the text that will precede the decoded output. For example, assume:

- HEADER string = “ACME Part Number: ”
- Decoded output = 123456

The output would look like:

```
ACME Part Number: 123456
```

Syntax

HEADER {““string””}

Where:

string — Is 1 to 80 characters. The default is NULL. String may include non-printable control characters in the following format:

\0d = CR

\0A = LF

The string may include keyword substitution:

– CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.

– TIMESTAMP — This keyword will be substituted with the current date and time.

HEIGHT

This command sets the expected Data Matrix height.

Note: See also “RATIO” on page 1-142 and “WIDTH” on page 1-202.

Syntax

HEIGHT {opt}

Where:

opt — Is either of the following:

- height — Range: 20 - 1024
- AUTO — Autoselect (default)

HELP

This command displays remote commands and descriptions of how to use them.

- List all available commands:

  HELP

- List help for a command:

  HELP {cmd}

  Where:

  – cmd — Most remote commands

- List detailed help for a command:

  HELP {cmd} V[ERBOSE]

  Where:

  – cmd — Select remote commands

  – VERBOSE — Display detailed information about the command

ILLUMINATION

This command selects the lighting for the HawkEye™ 1500.

Syntax

ILLUM[INATION] {opt}

Where:

opt — Is one of the following:

- OFF — Use ambient lighting.
- ON — Use constant lighting (default).
- EXT — Use external lighting.
- STROBE — Use strobed lighting.
- POWER[_STROBE] — Use power stobed lighting.
- ON_AND_PSTROBED — Constant on with power strobing at trigger.

INFORM

This command specifies whether or not each issued command reports back the state of the command just changed.

Syntax

INFORM {opt}

Where:

opt — Is either of the following:

- ON
- OFF (default)

INTENSITY_ENHANCE_ENABLED

See “FINETUNE” on page 1-66.

IO_MODE

This command selects the GPIO output mode.

Syntax

IO_MODE {mode} [duration]

Where:

mode — Is one of the following:

- NONE
- DV_2LINE_HS — HandShake mode with multiplexed (GOOD, FAIR, POOR, Locate Failure) and DataValid lines assigned. In this mode, the status is NOT cleared until the trigger goes away.
- DV_3LINE_HS — HandShake mode with GOOD, FAIR, POOR, and DataValid lines assigned. In this mode, the status is NOT cleared until the trigger goes away.
- DV_2LINE_PULSE — PULSE/PIPELINED with multiplexed (GOOD, FAIR, POOR, Locate Failure) and DataValid lines assigned with DataValid duration: 0 - 30,000ms.
- DV_3LINE_PULSE — PULSE/PIPELINED with GOOD, FAIR, POOR, and DataValid lines assigned with DataValid duration: 0 - 30,000ms.
- DV_HS — HandShake mode with OPT1 = Pass, OPT2 = Fail, for duration.
- DV_PULSE — PULSE/PIPELINED with OPT1 = Pass | Fail, OPT3 = Data Valid, for duration.
- HE_HS — HandShake Mode with OPT1 = Pass, OPT2 = Fail.
- HE_PULSE — PULSE/PIPELINED Mode with OPT1 = Pass, OPT2 = Fail, for duration.

duration — Is the pulse duration. In pulse mode, the status is NOT cleared until the trigger goes away. The range is 0 - 30,000ms.

IOASSIGN

This command allows variables defined with the SET_CRITERIA command to be assigned to digital output lines.

Note: Use the SET_CRITERIA command to configure variable_name definitions before assigning them to an output bit

Syntax

IOASSIGN {BIT} {variable_name}

Where:

- BIT — A valid output point. Valid points include OPTO_OUT1 to OPTO_OUT3 and GPIO_OUT1 to GPIO_OUT4.
- variable_name — Is one of the following:
  - ANGLE_FAILURE — Signals when the measured angle is outside of the user’s process limits
  - LOCATE_FAILURE — may be used as an overall indicator or masked to signal particular types of locate failures.
  - DECODE_FAILURE — If used when Match Mode is enabled, it allows the user to distinguish between NO MATCH with a successful Decode and NO MATCH with an unsuccessful Decode
  - RTE — Run time error, including Trigger Overruns, Decode Overruns, and Network errors
  - PASS — may also represent a MATCHED state if a Match Mode is enabled
  - FAIL — may also represent a NO MATCH state if a Match Mode is enabled
  - DV — Data Valid
  - GOOD — (used by the Verification option)
  - FAIR — (used by the Verification option)

- POOR — (used by the Verification option)

Use the CLEAR keyword to disable an I/O assignment; for example:

```
IOASSIGN GPIO_OUT2 CLEAR
```

You can display the state of a single assignment; for example:

```
IOASSIGN GPIO_OUT2 ?
```

You can display the state of all assignments; for example:

```
IOASSIGN ?
```

IP

This command sets up the static IP address when DHCP is disabled. The camera name is used for DHCP or static addressing.

Syntax

IP {cameraname} {ipaddress} {subnetmask} {gateway}

Where:

cameraname — Is the name of the camera.

ipaddress — Is the static IP address.

subnetmask — Is the subnet mask.

gateway — Is the gateway address.

IPCONFIG

This command returns the current name, IP address, and subnet mask of the camera.

Syntax

IPCONFIG {hostname} {ipaddress} {subnetmask}

Where:

hostname — Is the name of the camera

ipaddress — Is the IP address of the camera

subnetmask — Is the subnet mask of the camera

LEARN

This command initiates a learn sequence. The LEARN command narrows down the parameters associated with a symbol. This results in a more reliable decode of the same type of symbol. In most cases, it speeds up the decoding time. You can use the LEARN command in combination with the ASSIGN, PID, STORE, and TRIG commands.

Once the LEARN command is issued, all symbols that are being read must be of the same type and size in order to be decoded. To open up the decoder to all possible types of symbols again, issue the UNLEARN command.

Note: See “LEARNASSIST” on page 1-82 and “UNLEARN” on page 1-188.

You can also issue a LEARN command using QuicSet®:

1. Enter QuicSet® by pressing the QuicSet® recessed button (using a paper clip) once.
2. Learn the next acquired image by pressing the recessed button two more times quickly.
3. Exit QuicSet® by pressing the recessed button again.

Syntax

LEARN {opt}

Where:

opt — Is one of the following:

- Both (default)
- 1D
- 2D
- None
- Calibrate — Use this command to perform reflectance calibration:
    1. Center the Calibration Data Matrix in the FOV.

2. Enter “contrast” using the TARGET_CALIB_CONTRAST command (see page 1–169).
3. Enter “r_max” using the TARGET_CALIB_REFLECTANCE (page 1–170).

LEARNASSIST

This command narrows down the parameters associated with a symbol. This results in a more reliable decode of the same type of symbol. In most cases, it speeds up the decoding time.

Note: See also “LEARN” on page 1-80 and “UNLEARN” on page 1-188.

Syntax

LEARNASSIST {p1x,p1y,p2x,p2y,p3x,p3y,p4x,p4y} {rows}{cols}{polarity}

Where:

p1x,p1y,p2x,p2y,p3x,p3y,p4x,p4y — Clockwise, the x and y of the four corners.

rows — The number of rows to expect in the Data Matrix symbol:

Range: 8 - 144, or AUTO — Autoselect (default)

cols — The number of columns to expect in the Data Matrix symbol:

Range: 8 - 144, or AUTO — Autoselect (default)

polarity — Is one of the following:

- Dark on light
- Light on dark
- AUTO (default)

LEARNCANCEL

This command cancels a learn request that is waiting to be executed when the next trigger occurs. This occurs when you issue a learn and the unit is in TRIG T mode, waiting for a trigger.

Syntax

```
LEARNCANCEL
```

LEARNEX

This command can optionally learn Decoder and/or Photometry values.

Note: This is an action command. It cannot be store or queried.

Syntax

LEARNEX {Decoder} {Photometry}

Decoder — Learn either 1D or 2D; whatever is in the field of view.

- BOTH
- 1D
- 2D
- NONE
- CALIBRATE

Photometry — Learn the Gain and Exposure and leave in Manual mode.

- PHOTO
- NONE

LIGHT_CTL

This command selects the lighting type on an external light controller board. It allows selection of a lighting configured mode in a UID Verifier multifunction light controller attached to the light controller outputs configured with the SET_CRITERIA command:

LIGHT_CTL_BIT_1

LIGHT_CTL_BIT_2

LIGHT_CTL_BIT_3

LIGHT_CTL_BIT_4

Syntax

LIGHT_CTL {Light Type}

Where:

Light Type — Is one of the following:

- OFF

90	(90°)
D	(Dome Light)
45Q	(45° Quad Lights)
30T_NS	(30° Two Lights North South)
30T_EW	(30° Two Lights East West)
30Q	(30° Quad Lights)
30S_N	(30° Single Light North)
30S_E	(30° Single Light East)
30S_S	(30° Single Light South)
30S_W	(30° Single Light West)

LOAD

This command loads info from a job or PID.

Syntax

LOAD {pid# | saved | default}

Where:

pid# — Load/use decoder settings from a PID. This overwrites all the decoder configurable parameters.

Range: 0 - 15, and 99

saved — Load the saved job, and all other system, reporting, communication and application parameters.

default — Load factory default settings for a job.

LOCFL

This command customizes the Locator Failure output of the HawkEye™ 1500.

Syntax

LOCFL {cntrl} [""string""] [{Hdrcntrl} {Trlcntrl} {Beepcntrl}]

Where:

cntrl — Is the LOCFL message active (Y or N)?

string — (1 - 39 characters); the string may include keyword substitution:

Note: If the location of the Data Matrix fails, this will be reported as -1°.

– CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.

– TIMESTAMP — This keyword will be substituted with the current date and time.

Note: The following parameters are optional. However, it you enter one of them, you must enter all of them at the same time.

Hdrcntrl — Print the header before the LOCFL string (Y or N)?

Trlcntrl — Print the trailer before the LOCFL string (Y or N)?

Beepcntrl — Sound this many beeps (0 - 3) when this report is issued. The default is 0.

MACADDR

This command returns the MAC address for the unit. The address is set at the factory and cannot be reprogrammed in the field.

Syntax

```
MACADDR ?
```

For example:

```
    MACADDR xx:xx:xx:xx:xx:xx
```

MATCH

This command sets match control.

See the following related commands:

- “MATCHCOUNT” on page 1-94
- “MATCHEX” on page 1-95
- “MATCHSERIAL” on page 1-99
- “MATCHSTRING” on page 1-100

Syntax

MATCH {opt}

Where:

opt — Is one of the following:

- N — No match control (default).
- T — Text string matching.
- S — Serial string matching.
- B — Both text string and serial string matching.

Behavior of the Wildcard Match

The behavior of the wildcard match is:

- A case sensitive match is performed.
- * and ? are wildcard characters in the pattern as well as potentially valid characters in the string.
- * as a wildcard character represents 0 or more characters in a string.
- ? as a wildcard character represents 1 and only 1 character in a string.
- If * and ? are found in the string, the pattern can have either a * or ? describing that position, however, the pattern will also match any string with something at that position other than * or ? (i.e., the * and ? in the pattern are seen as wildcards first and characters second).

Examples of this behavior are:

“*string” matches “?string”

“*string” matches “*string”

“*string” matches “any string”

“*string” matches “string”

“?string” matches “?string”

“?string” matches “*string”

“?string” matches “Xstring”

MATCH_LIST

This command allows you to enter match strings into the MATCH_LIST table, which allows match string to be associated with output values. The list is processed from top to bottom to find the first matching string. The string_value associated with the match string is output through the digital I/O lines with the assigned MATCH_BIT_1 through MATCH_BIT_4 tags.

For example, using the following commands to define a MATCH_LIST table:

```
MATCH_LIST 2 “*AB??”
MATCH_LIST 11 “*ZZ*”
```

the strings “ABCD” and “HELLOAB12” would match list entry 2 and MATCH_BIT_2 would be enabled. If the string were “ZZ” or “xyZZ1234”, it would match list entry 11 and MATCH_BIT_1, MATCH_BIT_2, and MATCH_BIT_4 would be enabled.

The MATCH_BITS correspond to the hexadecimal encoding of the string_value.

Note: The multi-character wildcard is * and represents 0 to N character positions in the string. The single character wildcard is ? and represents one character position in the string.

Clear an entry by entering an empty string such as MATCH_LIST 11 ““.

Syntax

MATCH_LIST {1-15} “match string”

Behavior of the Wildcard Match

The behavior of the wildcard match is:

- A case sensitive match is performed.
- * and ? are wildcard characters in the pattern as well as potentially valid characters in the string.
- * as a wildcard character represents 0 or more characters in a string.
- ? as a wildcard character represents 1 and only 1 character in a string.

- If * and ? are found in the string, the pattern can have either a * or ? describing that position, however, the pattern will also match any string with something at that position other than * or ? (i.e., the * and ? in the pattern are seen as wildcards first and characters second).

  Examples of this behavior are:

  “*string” matches “?string”

  “*string” matches “*string”

  “*string” matches “any string”

  “*string” matches “string”

  “?string” matches “?string”

  “?string” matches “*string”

  “?string” matches “Xstring”

MATCH_LIST_ENABLE

This command specifies whether or not the match list will be used in MATCH mode. When MATCH or MATCHEX is configured for text (T) or both (B), MATCH_LIST_ENABLE directs the camera to use the set of strings defined with the MATCH_LIST command rather than the MATCH or MATCHEX command.

Syntax

MATCH_LIST_ENABLE {opt}

Where:

opt — Is one of the following:

- ON
- OFF (default)

MATCHCOUNT

This command sets the initial match count.

See the following related commands:

- “MATCH” on page 1-89
- “MATCHEX” on page 1-95
- “MATCHSERIAL” on page 1-99
- “MATCHSTRING” on page 1-100

Syntax

MATCHCOUNT count

Where:

count — Is the initial count:

Range: 0 - 0x7FFFFFFF

MATCHEX

This command sets various match parameters in one command.

See the following related commands:

- “MATCH” on page 1-89
- “MATCHCOUNT” on page 1-94
- “MATCHSERIAL” on page 1-99
- “MATCHSTRING” on page 1-100

Syntax

MATCHEX {control} {string} {count} {incr} {1stchar} {lastchar}

Where:

control — Is one of the following:

N — No match control (default).

T — Text string matching.

S — Serial string matching.

B — Both text string and serial string matching.

string — Is a string, which may include wildcards:

? — Single don’t care
* — Multiple don’t cares

Range: 1 - 30 upper and/or lowercase letters

count — Is the initial count.

Range: 0 0x7FFFFFFF

incr — The increment value to be added after each inspection.

Range: -1000 thru +1000

1st char — The first character in the decode string to be included in the serial number.

Range: 1 - 1000

last char — The last character in the serial number.

Range: 1st char - 1000

MATCHFL

This command customizes the unsuccessful match (nomatch) output of the HawkEye™ 1500.

Syntax

MATCHFL {cntrl} [""string""] [{Hdrcntrl} {Trlcntrl} {Beepcntrl}]

Where:

cntrl — Is the MATCHFL message active (Y or N)?

string — (1 - 39 characters); the string may include keyword substitution:

- ANGLE — This keyword reports the Data Matrix angle in degrees from 0° to 359°.
- CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.
- DATA — This keyword will be substituted with the actual decoded string. For example, if “4567321” were decoded, and the event were set as “OKDEC: This is the decoded data: (DATA)\0d\0a” then the one line of output would look like the following:

  This is the decoded data: 4567321
- DATAHEX — This keyword converts the decode data to a hex character string, similar to the previous “OUTPUT FORMAT HEX” functionality.
- DETAILED — This keyword returns a detailed failure string.
- PID# — This keyword returns the PID number of the PID that successfully decoded the symbol.
- TIMESTAMP — This keyword will be substituted with the current date and time.

- VERI_1_IAQG — This keyword provides a DMx AutoID compatibility mode for the IAQG Verification I report.
- VERI_DETAIL — This keyword will be substituted with very detailed AIM information: Overall Grade, Grade Contrast, Contrast, Grade axial nonuniformity, Axial nonuniformity, Grade of print growth, Print growth x, Print growth y, Grade of error correction, Num error bits UEC value.
- VERI_FORMATTED — This keyword (typically placed in the OKDEC formatted string) puts out the verification data with the following format:

  ```
  OG:3 CG:3 C:65 nG:4 n:0.00 GPG:3 PGx:0.20 PGY:0.02 GUEC:4 #B:0 UEC:1.00
  ```
- VERI_GRADE — This keyword will be substituted with the AIM grade. The string may also include non printable control characters by typing the backslash character followed by the two character hex number that represents the character. For example, \0d is a CR and \0A is a LF.

Note: The following parameters are optional. However, it you enter one of them, you must enter all of them at the same time.

Hdrcntrl — Print the header before the MATCHFL string (Y or N)?

Trlcntrl — Print the trailer before the MATCHFL string (Y or N)?

Beepcntrl — Sound this many beeps (0 - 3) when this report is issued. The default is 2.

MATCHSERIAL

This command specifies serial checking parameters.

See the following related commands:

- “MATCHCOUNT” on page 1-94.
- “MATCHEX” on page 1-95
- “MATCHSTRING” on page 1-100.

Syntax

MATCHSERIAL {incr} {1st char} {last char}

Where:

incr — The increment value to be added after each inspection.

Range: -1000 thru +1000

1st char — The first character of the serial number to compare.

Range: 1 - 1000

last char — The last character in the serial number to compare.

Range: 1st char - 1000

MATCHSTRING

This command sets match control to character matching. In simple applications, where the line is set up to compare the code read to a fixed string, you would use MATCH T ("string"). In other words, every part read must be the same. You would have different triggers assigned to different PIDs such that there is a different string to compare, depending on the trigger.

See the following related commands:

- "MATCHCOUNT" on page 1-94.
- "MATCHEX" on page 1-95
- "MATCHSTRING" on page 1-100.

Syntax

MATCHSTRING {string}

Where:

string — Is a string, which may include wildcards:

? — Single don't care
* — Multiple don't cares

Range: 1 - 80 upper and/or lowercase letters
Default: Null

String may include non-printable control characters in the following format:

\0d = CR

\0A = LF

MORPHOLOGY

This command sets the morphological pre-processing of the image.

Syntax

MORPH[OLOGY] {opt} [num]

Where:

opt — Is one of the following:

- ADAPTIVE — Hand-Held Demo mode where it tries no image processing first; if it fails, it tries Erode, if it fails, it tries Dilate.
- CLOSE — DILATE and then ERODE (remove minor dark defects of light cells)
- DILATE — Dilate light pixels and erode dark pixels
- ERODE — Erode light pixels and dilate dark pixels (increase dark cell size and reduce light cell size)
- NONE (default)
- OPEN — ERODE and then DILATE (remove minor light defects of dark cells)

num — Is the number of passes:

Range: 1 - 5
Default: 1

NEGATIVE

This command enables/disables conversion of the camera acquired image to a negative image during the image acquisition phase.

Syntax

NEGATIVE {ON | OFF}

Where:

ON — Enable

OFF — Disable (default)

NO_QUIET_ZONE_CLUTTER

See “FINETUNE” on page 1-66.

NUMBC

This command sets the number of barcodes in a field of view to decode. To look for Data Matrix only, use NUMBC 0.

Syntax

NUMBC {n}

Where:

n — Is the number of barcodes to decode:

Range: 0 - 1
Default: 1

NUMDM

This command specifies the maximum number of Data Matrix symbols to find. To look for 1-D only, use NUMDM 0.

Syntax

NUMDM {n}

Where:

n — Is the maximum number of Data Matrix symbols to find:

- 0 — Disable/No look
- 1 (default)

OFFLINE

This command returns the unit to offline so that you can program it.

Syntax

```
OFF[LINE]
```

OKDEC

This command customizes the successful decode output of the HawkEye™ 1500.

Syntax

OKDEC {cntrl} [""string""] [{Hdrcntrl} {Trlcntrl} {Beepcntrl}]

Where:

cntrl — Is the OKDEC message active (Y or N)?

string — (1 - 39 characters); the string may include keyword substitution (in parentheses):

- ANGLE — This keyword reports the Data Matrix angle in degrees from 0° to 359°.
- CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.
- DATA — This keyword will be substituted with the actual decoded string. For example, if “4567321” were decoded, and the event were set as “OKDEC: This is the decoded data: (DATA)\0d\0a” then the one line of output would look like the following:

  This is the decoded data: 4567321
- DATAHEX — This keyword converts the decode data to a hex character string, similar to the previous “OUTPUT FORMAT HEX” functionality.
- DETAILED — This keyword returns a detailed failure string.
- PID# — This keyword returns the PID number of the PID that successfully decoded the symbol.
- TIMESTAMP — This keyword will be substituted with the current date and time.

- VERI_1_IAQG — This keyword provides a DMx AutoID compatibility mode for the IAQG Verification I report.
- VERI_DETAIL — This keyword will be substituted with very detailed AIM information: Overall Grade, Grade Contrast, Contrast, Grade axial nonuniformity, Axial nonuniformity, Grade of print growth, Print growth x, Print growth y, Grade of error correction, Num error bits UEC value.
- VERI_FORMATTED — This keyword (typically placed in the OKDEC formatted string) puts out the verification data with the following format.

  ```
  OG:3 CG:3 C:65 nG:4 n:0.00 GPG:3 PGx:0.20 PGY:0.02 GUEC:4 #B:0 UEC:1.00
  ```
- VERI_GRADE — This keyword will be substituted with the AIM grade. The string may also include non printable control characters by typing the backslash character followed by the two character hex number that represents the character. For example, \0d is a CR and \0A is a LF.

Note: The following parameters are optional. However, it you enter one of them, you must enter all of them at the same time.

Hdrcntrl — Print the header before the OKDEC string (Y or N)?

Trlcntrl — Print the trailer before the OKDEC string (Y or N)?

Beepcntrl — Sound this many beeps (0 - 3) when this report is issued. The default is 0.

OKMATCH

This command customizes the successful match output of the HawkEye™ 1500.

Syntax

OKMATCH {cntrl} [""string""] [{Hdrcntrl} {Trlcntrl} {Beepcntrl}]

Where:

cntrl — Is the OKMATCH message active (Y or N)?

string — (1 - 39 characters); the string may include keyword substitution:

- ANGLE — This keyword reports the Data Matrix angle in degrees from 0° to 359°.
- CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.
- DATA — This keyword will be substituted with the actual decoded string. For example, if “4567321” were decoded, and the event were set as “OKDEC: This is the decoded data: (DATA)\0d\0a” then the one line of output would look like the following:

  This is the decoded data: 4567321
- DATAHEX — This keyword converts the decode data to a hex character string, similar to the previous “OUTPUT FORMAT HEX” functionality.
- DETAILED — This keyword returns a detailed failure string.
- PID# — This keyword returns the PID number of the PID that successfully decoded the symbol.
- TIMESTAMP — This keyword will be substituted with the current date and time.

– VERI_1_IAQG — This keyword provides a DMx AutoID compatibility mode for the IAQG Verification I report.

– VERI_DETAIL — This keyword will be substituted with very detailed AIM information: Overall Grade, Grade Contrast, Contrast, Grade axial nonuniformity, Axial nonuniformity, Grade of print growth, Print growth x, Print growth y, Grade of error correction, Num error bits UEC value.

– VERI_FORMATTED — This keyword (typically placed in the OKDEC formatted string) puts out the verification data with the following format.

```
OG:3 CG:3 C:65 nG:4 n:0.00 GPG:3 PGx:0.20 PGY:0.02 GUEC:4 #B:0 UEC:1.00
```

– VERI_GRADE — This keyword will be substituted with the AIM grade. The string may also include non printable control characters by typing the backslash character followed by the two character hex number that represents the character. For example, \0d is a CR and \0A is a LF.

Note: The following parameters are optional. However, it you enter one of them, you must enter all of them at the same time.

Hdrcntrl — Print the header before the OKMATCH string (Y or N)?

Trlcntrl — Print the trailer before the OKMATCH string (Y or N)?

Beepcntrl — Sound this many beeps (0 - 3) when this report is issued. The default is 0.

ONLINE

This command puts the unit online performing the desired application. When returning to online mode, the unit will beep three times rapidly and the Mode light will return to solid yellow.

Syntax

ON[LINE]

ORIENT

This command sets the matrix orientation in read mode. Orientation is set automatically by Learn which works for Read if matrix orientation does not change more than ± 45°. Otherwise, use ORIENT AUTO.

Note: All measurements are assumed to be rotated in a counter clockwise direction.

Syntax

ORIENT {orientation | AUTO}

Where:

orientation — Is one of the following:

- 1 = 0°
- 2 = 90°
- 3 = 180°
- 4 = 270°
- 5 = 45°
- 6 = 135°
- 7 = 225°
- 8 = 315°
- AUTO — Autoselect (default)

OUTPUT

- To set up the different output states on the different ports:

  `OUTPUT {PORT} {STATE} {FORMAT STRING} {ACTIVE} {H/T CONTROL}`

- To set up the header/trailer strings for the different ports:

  `OUTPUT {PORT} H_T {FORMAT STRING} (FORMAT STRING}`

  Where:

  – PORT — Is one of the following:

    - STANDARD (Camera’s serial port)
    - TCP1
    - TCP2
    - TCP3
    - TCP4

  – STATE — Is one of the following:

    - OKDEC (see page 1–107)
    - DECFL (see page 1–49)
    - LOCFL (see page 1–87)
    - OKMATCH (see page 1–109)
    - MATCHFL (see page 1–97)
    - RTE (see page 1–155)

  – FORMAT STRING — Is the format string in quotes, or without quotes, if no spaces are needed

  – ACTIVE — For whatever you want to use, sets STATE on or off for the particular PORT, and is one of the following:

    - Y or ON or ENABLE
    - N or OFF or DISABLE

- H/T CONTROL — Activates the header and/or trailer, and is one of the following:
  - NONE
  - H — Header
  - T — Trailer
  - H_T — Both header and trailer

- To configure the different BEEP settings:

  OUTPUT BEEP {OKDEC} {DECFL} {LOCFL} {OKMATCH} {NOMATCH} {RTE}

  Each setting indicates the number of beeps to use on each possible output (0 | 1 | 2).

The original OKDEC, DECFL, RTE et al., commands set up the STANDARD and TCP1 settings only, and the beeps.

PARTQ

This command enables/disables the part queue mechanism. The part queue stores a number of cycle reports in-line with the running inspection. The records are stored as a re-usable queue, that is, when the queue is full, the newest record replaces the oldest record. The part queue is loss-less.

Syntax

PARTQ {opt} [qualifier] [size]

Where:

opt — Is one of the following:

- Y — Enable the part queue
- N — Disable the part queue
- R — Save reports without saving images

qualifier — Specifies what you want to store.

- All — Store all records
- Passed — Store records of passed inspections
- Failed — Store records of failed inspections

size — The number of records to store:

Range: 1 - 100000

PARTQCAPACITY

This command obtains the maximum record count based on free memory in the camera for records with images and records without images in the PartQ configuration.

Syntax

PARTQCAPACITY {First value} {Second value}

Where:

- First value — Maximum number of records with images
- Second value — Maximum number of records without images

For example:

```
PARTQCAPACITY 31 5700
```

PARTQCLEAR

This command clears the current records in the part queue.

Syntax

```
PARTQCLEAR
```

PARTQCOUNT

This command displays the current number of records in the part queue.

Syntax

```
PARTQCOUNT
```

PARTQFAIL

This command specifies the exact failure to store in the part queue when it is set to Failed qualification.

Syntax

PARTQFAIL {qualifier}

Where:

qualifier — Specifies the exact failure you want to store:

- ALL
- DECFL
- LOCFL
- MATCHFL
- RTE

PARTQFTP

This command sets up the Part Queue to save each image remotely as it occurs. The files constructed are:

PATH/CameraName_{CycleCount | PartQ Index}_DecodeResult.{txt | bmp}

Where:

- CameraName — Is the name of the camera (e.g., hawkeye)
- CycleCount — Is a 9-digit decimal cycle count (e.g., 000012345)
- PartQ Index — Is a 4-digit part queue index (e.g., 0245)
- DecodeResult — Is the overall result:
  - OKDEC
  - DECFL
  - LOCFL
  - OKMATCH
  - MATCHFL
  - RTE

File Examples:

path/hawkeye_000001304_OKDEC.txt

path/hawkeye_000001304_OKDEC.bmp

If the Part Queue is configured with a maximum limit, then the PartQ Index is used instead of the cycle count. Files that already exist (after wrap-around) are overwritten. If the Part Queue is configured with no limit (0), then the CycleCount will be used in the file name instead.

The .TXT file contains tab-delimited version of the results:

```
Cycle Count:<tab>000000000

Passed:<tab>000000000

...
```

Decode Data:<tab>{decoded data}

The Decode Data is not converted to any specific format; it is simply saved as is.

Syntax

PARTQFTP {HOST} {PATH} {SAVERESULTS} {SAVEIMAGE}

Where:

HOST — IP address of the FTP server; if "", files saved locally

PATH — Path on FTP server where files are saved

SAVERESULTS — Y or N to save a .TXT results file

SAVEIMAGE — Y or N to save the image as a bitmap (no graphics)

PARTQFTPEX

This command controls the number of records to collect when directing the Part Queue to send records to the host system directly using FTP. The numbering in the file names of transferred records is also affected by this parameter.

Syntax

PARTQFTPEX {host} {path} {save results} {save image} {count}

Where:

host — IP address of the FTP server; if "", files are saved locally.

path — Path on FTP server where files are saved.

save results — Y or N to save a .TXT results file.

save image — Y or N to save the image as a formatted bitmap (no graphics).

count — Is either of the following:

- 0 — Records with the file names containing a 9-digit decimal representation of the cycle count "_000000000_".
- 1-9999 — Sequentially labeled records with the file names containing a 4-digit representation "_0000_", wrapping at N records and overwriting old records.

PARTQREQ

This command uploads the part queue. You can upload the entire queue or a single entry.

Syntax

PARTQREQ {compression} {opt2} {preserve}

Where:

compression — Specify compression of images when uploading via the serial port. Compression is not used when uploading via TCP/IP. Compression is one of the following:

- 0 - No compression applies; image uploaded in full
- 26 - Decimated image
- 27 - Decimated image with JPEG quality
- 255 - No compression applies; image uploaded in full

opt2 — Is either of the following:

- # — Upload a specific entry in the queue
- All — Upload all entries; this is the default

preserve — Is either of the following:

- When preserve is specified, do not clear the queue.
- When preserve is not specified, clear the queue.

PARTQREQFORMAT

This command allows user defined parts of the cycle records stored on the camera to be retrieved on demand.

Syntax

PARTQREQFORMAT {FORMAT STRING} {ALL | #}

Where:

- FORMAT STRING — Specifies the desired format of the data to retrieve using the new CDxx (see Table 1–11) and IDxx (see Table 1–12) tags, the same as the OUTPUT command.
- ALL | # — Is the specific record, either ALL or by index in the queue; 0 is the oldest up to N (the newest), stored on the camera. The number of records can be obtained by using the PARTQCOUNT command.

TABLE 1–11. CD Fields

Field	Description
00	All Data
01	Overall Pass/Fail
02	Count of triggers
03	Count of passes
04	Count of failures
05	Count of successful locates
06	Count of failed locates
07	Count of successful decodes
08	Count of failed decodes
09	Count of successful matches
10	Count of failed matches
11	Count of alarms
12	Count of trigger overruns
13	Count of decode overruns
14	Count of acquisition timeouts
15	PID
16	Symbol Pass/Fail

TABLE 1–11. **CD Fields (Continued)**

Field	Description
17	Located / Not Located
18	Decoded / Not Decoded
19	Matched / Not Matched (0) (0 if matching is not active)
20	Detailed Decode Failure
21	Overall AIM Verification Grade
22	Unformatted decode data
	Length/raw data
23	Photometry Settings
	Auto | Manual / Gain / Exposure
24	Image Timestamp
25	Image Data Raw
	Width/Height/Gray-scale values

TABLE 1–12. **ID Fields**

Field	Description
01	Image Data Bmp 32-bit Length/Binary values
02	Image Data Tiff 32-bit Length/Binary values

PARTQSAVE

This command saves the current part queue to a remote path on demand (the cycle records are stored on the camera in Memory).

Note: The cycle records on the Camera are cleared by this operation afterwards.

Syntax

PARTQSAVE {HOST} {PATH} {SAVERESULTS} {SAVEIMAGE}

Where:

HOST — IP address of the FTP server; if "", files are saved locally

PATH — Path on FTP server where files are saved

SAVERESULTS — Y or N to save a .TXT results file

SAVEIMAGE — Y or N to save the image as a bitmap (no graphics)

PARTQSUM

This command displays a summary of the part queue.

Syntax

PARTQSUM {opt1}

Where:

opt1 — Is the kind of summary to display:

- All — Display a complete summary. This is the default.
- # — Display a summary of the # record. This is a zero-based index.

PHOTOEX

This command sets both Auto and Manual settings in one command.

Syntax

PHOTOEX {opt1} {opt2} {opt3} {opt4} {opt5} {opt6} {opt7}

Where:

opt1 — Sets the Photometry control:

- AUTO
- MANUAL — Sets the operating Gain and Exposure

opt2 — The desired exposure if Manual control is selected

opt3 — The desired gain if Manual control is selected. Sets up the range for the auto photometry algorithm if Auto control.

opt4 — Min exposure — The minimum shutter/exposure duration value used for Auto control. (30 - 100,000 usec)

opt5 — Max exposure — The maximum shutter/exposure duration value used for Auto control. (Min exp - 100,000 usec)

opt6 — Min gain — The minimum gain value used for Auto control. (0 - 1023)

opt7 — Max gain — The maximum gain value used for Auto control. (Min Gain - 1023)

PHOTOMETRY

This command selects Photometry settings and controls. An application can use user selected gain and exposure or it can use the auto-photometry algorithm to determine the correct settings at each trigger.

Syntax

- Define automatic photometry control. Perform photometry control at each trigger and prior to each image scan:

  PHOTO[METRY] AUTOMATIC [opt2] [opt3] [opt4] [opt5]

  Where:

  opt2 — Min exposure — The minimum shutter/exposure duration value used for Auto control. (30 - 100,000 usec)

  opt3 — Max exposure — The maximum shutter/exposure duration value used for Auto control. (Min exp - 100,000 usec)

  opt4 — Min Gain — The minimum gain value used for Auto control. (0 - 1023)

  opt5 — Max Gain — The maximum gain value used for Auto control. (Min Gain - 1023)

- Define manual photometry control. Do not perform automatic photometry at each trigger. If opt2 and opt3 are supplied, use these values at each scan. Each PID can have its own gain and exposure settings:

  PHOTO[METRY] MANUAL [opt2] [opt3]

  Where:

  opt2 — Exposure range (30 - 100,000 usec)

  opt3 — Gain (0 - 1023)

PID

This command allows you to train the decoder to rapidly and accurately decode the symbol in the current field of view. After the parameters are established, you can save the predictable parameters with the STORE command. The camera can store up to 15 predictable parameter sets, which are identified by ID 1 - 15 (PIDs).

Each PID contains the following information:

TABLE 1–13. Parameters Stored in Each PID

Parameter	Notes
Photometry	Gain & Exposure only. NOTE: If Autophotometry is ON (System parameters), then these PID settings are NOT used.
PreProcessing	Morphology operator and iterations
Decoder	All decoder parameters.

By default, Microscan supplies factory default PID99, which is programmed to read most codes with the HawkEye™ 1500. The default parameters are loaded into the current working area at boot time. After you start changing parameters from the ones in the current working area, the PID number changes to PID0, indicating that it has no home. You should store the info to a PID so that when you issue a SAVE command, your data is available. By default, if you issue the SAVE command without storing PID0, the data is stored to PID1. Depending on which PID is in the working area when you issue the SAVE command, the PID will be loaded the next time you boot the camera.

PID 0 is the current PID info being used.

PID 99 is the default PID information used to start from factory defaults.

There are options and configurations available to advanced users.

ASSIGN {pid#} {trigger#}

You can ASSIGN {pid#} to {trigger#}. Because there are 28 real/virtual triggers, you can program the camera to “trigger” and indicate that a specific symbol is in the field of view and ready to be decoded. The camera pre-loads the decoder with the specific parameters for that product as the image is being lifted. When the image is in RAM, the “learned” parameters are used to rapidly and accurately decode the image. Thus, several “learned” products can be used on the same line without stopping and re-programming the camera.

VIEW PID {pid#}

You can VIEW PID {pid#} to validate what you have stored.

LOAD PID {pid#}

If you do not want to use multiple PIDs, you can LOAD PID {pid#} into the current parameters and any trigger will use those parameters.

POLARITY

This command sets expected polarity of the Data Matrix.

Syntax

POL[ARITY] {opt}

Where:

opt — Is one of the following:

- AUTO (default)
- DL — Dark on light
- LD — Light on dark

POSITION_ENHANCE_ENABLED

See “FINETUNE” on page 1-66.

PROBE DIRECTION

This command sets the Probe direction.

Syntax

PROBE DIRECTION {opt}

Where:

opt — Is one of the following:

- HORI(zontal)
- HORI_VERT (default)
- VERT(ical)
- VERT_HORI
- CRISS_CROSS

PROBE SPEED

This command sets the Probe speed.

Caution: Increasing the probe speed may adversely affect the robustness of the reading.

Syntax

PROBE SPEED {opt}

Where:

opt — Is one of the following:

- NORMAL (default)
- OVERDRIVE
- TURBO

PROBE THRESHOLD

This command sets the Probe threshold level that the edge should exceed.

Syntax

PROBE THRESHOLD {opt}

Where:

opt — Is the probe threshold level:

Range: 5 - 100
Default: 18

PROMPT

This command controls the user-defined command prompt.

Syntax

PROMPT {opt} [""prompt string""]

Where:

opt — Is either of the following:

ON
OFF (default)

prompt string — Up to a 40-character prompt string. The default is > . The string may include control characters by preceding the control char with ’\’, and followed by two hex characters. For example:

""\07"" for bell

""\\"" for a backslash character

QRFINDER

This command sets the QR finder misalignment parameter.

Syntax

QRFINDER {opt}

Where:

opt — Is one of the following:

- 0 — QR_FINDER_SHIFT_NONE (default)
- 1 — QR_FINDER_SHIFT_IN1
- 2 — QR_FINDER_SHIFT_IN2
- 3 — QR_FINDER_SHIFT_IN3
- 4 — QR_FINDER_SHIFT_OUT1
- 5 — QR_FINDER_SHIFT_OUT2
- 6 — QR_FINDER_SHIFT_OUT3

QUICSET

QuicSet® provides audible and visual feedback about the optical alignment of the camera and the symbol positioned under it. QuicSet® also determines the correct Gain and Exposure settings for the symbol.

To enter QuicSet® mode, press the recessed QuicSet button once.

Note: The QuicSet® button is located on the connector side of the camera, which is opposite the lens side. You will need a paper clip.

The yellow Mode LED should start flashing. Physically position the camera above the symbol. You will hear a series of beeps. The beeps have three tones:

- Lowest Tone — Symbol is in the field of view
- Middle Tone — Symbol is close to being in the optimal read position
- Highest Tone — Symbol is in optimal read position

The LEDs on the front of the camera illuminate to indicate positioning:

- All LEDs on — Optimal setting
- Mode LED flashing — Poor symbol alignment

Move the camera until you hear the highest tone, and then lock down the camera.

To exit QuicSet®, press the QuicSet button once. You should hear three short beeps, and the Mode light should be steady on, indicating that the camera has stored the Gain and Exposure parameters. The camera is now online, running an application.

The QuicSet® feature is effective only if Photometry is set to manual.

You can initiate the LEARN and UNLEARN commands using the QuicSet button. In the Terminal window, type HELP LEARN V or HELP UNLEARN V for more information.

Syntax

- Run QuicSet®:

  QUIC[SET] Y

- Cancel QuicSet®:

  QUIC[SET] N

- Display the current setting for QuicSet®:

  QUIC[SET] ?

QUICSETPLUS

This command allows you to specify what functions QuicSet® can perform.

Syntax

```
QUICSETPLUS  {TrigPolDetect} {MatchStringDetect}
             {Allow Learn} {Allow Save}
```

Where:

TrigPolDetect — (Y | N)

> If Y, then the trigger polarity of the OPTO IN signal at the time of successful exit from QuicSet® is set automatically as the triggering “edge.”
>
> The default is N.

MatchStringDetect — (Y | N)

> If Y, then the data encoded in the symbol used for QuicSet® alignment is set automatically as the “match string.” For more information about match string, type HELP MATCH.
>
> The default is N.

Allow Learn — (Y | N)

> If Y, then allow Learn and Unlearn button presses while in QuicSet alignment.
>
> The default is Y.

Allow Save — (Y | N)

> If Y, automatically save all data to Flash after Learn or Unlearn. Otherwise, just RAM will be updated.
>
> The default is Y.

RATIO

This command sets the expected width to height ratio. In READ mode, when HEIGHT and WIDTH are set to AUTO, RATIO can set the expected ratio. If the ratio may change, use RATIO AUTO. For a square Data Matrix, the ratio is 1.0, which means you should use RATIO 10.

Note: See also “HEIGHT” on page 1-70 and “WIDTH” on page 1-202.

Syntax

RATIO {opt}

Where:

opt — Is either of the following:

- ratio — Range: 3 - 50 (3 = 0.3; 50 = 5.0)
- AUTO — Autoselect (default)

READ_LICENSE

This command determines whether or not an Option’s license is installed. It also allows you to check the status of installed options.

Syntax

READ_LICENSE [VERIFICATION | ?]

- READ_LICENSE VERIFICATION returns:
  - VERIFICATION ON (if enabled)
  - PARAMETER INVALID (if disabled or unavailable)
- READ_LICENSE ? displays a list of enabled options. For example:

  ```
  READ_LICENSE UIDCK ON
  READ_LICENSE VERFICATION ON
  ```

RELEASE

This command releases control of the camera, allowing other users access to it.

Syntax

RELEASE

REMOVE

This command removes the assigned PID from the indicated trigger. When the specified trigger occurs, (to indicate the read and decode of an image), the default current PID parameters will be used by the decoder.

Syntax

REMOVE {trig}

Where:

trig — Is the trigger number:

Range: 0 - 27

REPORTCANCEL

This command cancels a pending REPORTREQ command. This command is only used by ReadRunner.

Syntax

REPORTCANCEL

REPORTREQ

This command retrieves the next report from the system while running. This command is only used by ReadRunner.

Syntax

REPORTREQ {opt1} {opt2} {opt3}

Where:

opt1 — Compression

0 — Lossless compression NOT IMPLEMENTED

1 — JPEG with quality value=1, 4:1 compression ratio

4 — JPEG with quality value=4, 8:1 compression ratio

26 — Decimated image with JPEG quality=2, 16:1 compression ratio

27 — Decimated image with JPEG quality=5, 32:1 compression ratio

255 — No compression applies; image uploaded in full

NONE — No compression applies; image uploaded in full (DEFAULT)

opt2 — Is either 0 (keep image out) or 1 (include image)

opt3 — NEXTFAIL or LASTFAIL — Retrieves the next failed or last failed report. If not specified, retrieves the next report.

RESET

This command resets some or all settings and, in some cases, reboots the unit.

Syntax

RESET [opt]

Where:

opt — Is one of the following:

- ALL — Performs a RESET FACTORY, saves it to Flash, and then reboots the unit.
- APPMODE — Reset the acquisition parameters and lighting (essentially everything in the Application Mode dialog of ReadRunner) and save it to flash.
- DECODER — Reset the decoder to factory defaults, reset photometry to Auto, and save it to flash.
- HARD — Reboot the unit.
- FACTORY — Reset all settings to factory defaults. This includes all connectivity options. The camera will be set to DHCP Y, the camera name will be set to “HawkEye” (where xxyyzz are the last three octets of the camera’s MAC address), default IP address, CONSOLE 0, BEEP Y. All decoder parameters, acquisition parameters, and BOOT PARA MS (except for the MAC address) will be set to factory defaults. All will be save to flash.
- SOFT — This takes the camera offline (if necessary), performs the equivalent of RESET APPMODE and RESET DECODER, removes any saves PIDs as seen with the DIR command, and then restores the starting offline status. No changes are made to the BOOT PARAMS and the camera is not rebooted.

RETRY

This command sets the retry mode, which allows the reader to try different decoder or light parameters, as well as to try multiple image captures.

Syntax

- Set read cycle to single attempt mode with no retries on failure. This is the **default**:

  RETRY NONE

- Set read cycle to multiple retry attempts within the specified time limit:

  RETRY TIME opt

  Where:

  opt — Is 35 - 60000ms

- Set read cycle to a specified number of attempts:

  RETRY COUNT opt

  Where:

  opt — Is 1 - 15

- Report only successful attempts. If duplicate data occurs, report it only if opt ms have elapsed since last decode:

  RETRY SUPERMARKET opt

  Where:

  opt — Is 0 - 60000ms; (0 = infinite ISWT)

- While the GPIO signal is present on Pin 9, keep retrying:

  RETRY GPIO

- Retry up to three PIDs if the first decode failed. You may define a list of up to three PIDs to be tried on decode failure. The decode occurs on the same image. The last PID used will be left as the current PID unless there is a PID assigned to the trigger and, if so, the PID assigned to the trigger will be the next loaded and used:

RETRY LIST [pid# | pid# | pid#]

Where:

pid# — Is the PID number.

- Retry once using the specified lighting:

RETRY LIGHT {opt} {pid}

Where:

opt — Is one of the following:

- OFF — Use ambient lighting.
- ON — Use constant lighting; this is the default.
- EXTERNAL — Use external lighting.
- STROBE — Use strobe lighting.
- POWER[_STROBE] — Use power stobed lighting.
- ON_AND_PSTROBED — Constant on with power strobing at trigger.

pid — You may define a valid PID to be tried on decode failure. The gain and exposure from the indicated PID will be used only if the Photometry is set. This mode is used commonly with the internal lighting provided and on retry using an external light (user provided) usually with Dot Peen marks.

RETRYEX

This command sets the retry modes of the system for all modes. It is identical to the RETRY command; however, it is meant to be used by custom FBI or C++ code and requires all fields to be specified. In addition, RETRYEX can be queried. ReadRunner uses RETRYEX to program all the retry modes in the camera. RETRY is meant to be used instead by a user typing the command manually in the Terminal Window or Hypercritical program.

Note: Each field must contain valid information regardless of the mode.

Syntax

- Set read cycle to single attempt mode with no retries on failure. This is the default:

  RETRY NONE

- Set read cycle to multiple attempts within the specified time limit:

  RETRY TIME opt

  Where:

  opt — Is 35 - 60000ms

- Set read cycle to a specified number of attempts:

  RETRY COUNT opt

  Where:

  opt — Is 1 - 15

- Report only successful attempts. If duplicate data occurs, report it only if opt ms have elapsed since last decode:

  RETRY SUPERMARKET opt

  Where:

  opt — Is 35 - 60000ms

- While the GPIO signal is present on Pin 9, keep retrying:

RETRY GPIO

- Retry up to three PIDs if the first decode failed. You may define a list of up to three PIDs to be tried on decode failure. The decode occurs on the same image. The last PID used will be left as the current PID unless there is a PID assigned to the trigger and, if so, the PID assigned to the trigger will be the next loaded and used:

  RETRY LIST [pid# | pid# | pid#]

  Where:

  pid# — Is the PID number.

- Retry once using the specified lighting:

  RETRY LIGHT {opt} {pid}

  Where:

  opt — Is one of the following:

  - OFF — Use ambient lighting.
  - ON — Use constant lighting; this is the default.
  - EXTERNAL — Use external lighting.
  - STROBE — Use strobe lighting.
  - POWER[_STROBE] — Use power stobed lighting.
  - ON_AND_PSTROBED — Constant on with power strobing at trigger.

  pid — You may define a valid PID to be tried on decode failure. The gain and exposure from the indicated PID will be used only if the Photometry is set. This mode is used commonly with the internal lighting provided and on retry using an external light (user provided), usually with Dot Peen marks.

ROI

This command defines the region of interest (ROI) for the symbol.

Syntax

- Define the coordinates of the region of interest:

  ROI {ux, uy, lx, ly}

  Where:

  – ux — 0 - 658 (658 = default); u corresponds to upper left

  – uy — 0 - 493 (493 = default); u corresponds to upper left

  – lx — 0 - 658 (0 = default); l corresponds to lower right

  – ly — 0 - 493 (0 = default); 1 corresponds to lower right

- Define the region of interest based on the most recently decoded Data Matrix:

  ROI AUTO {pad}

  Where:

  pad — Is one of the following:

  – 1 — 50 pixels larger than the Data Matrix

  – 2 — 100 pixels larger than the Data Matrix

  – 3 — 150 pixels larger than the Data Matrix

  – 4 — 200 pixels larger than the Data Matrix

- Reset the region of interest back to the original area (full ROI):

  ROI RESET

ROWS

This command specifies the number of rows to expect in the Data Matrix symbol. The algorithm uses the value specified without having to re-compute it from image to image.

Note: See also “COLS” on page 1-44.

Syntax

ROWS {opt}

Where:

opt — Is either of the following:

- rows — Is the number of rows:

  Range: 8 - 144
- AUTO — Autoselect (default)

RTE

This command customizes the overrun message output of the HawkEye™ 1500. This command only affects the STANDARD output. See “OUTPUT” on page 1-113 for more details.

Syntax

RTE {cntrl} [""string""] [{Hdrcntrl} {Trlcntrl} {Beepcntrl}]

Where:

cntrl — Is the RTE message active (Y or N)?

string — (1 - 39 characters); the string may include keyword substitution:

- CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.
- TIMESTAMP — This keyword will be substituted with the current date and time.

Note: The following parameters are optional. However, it you enter one of them, you must enter all of them at the same time.

Hdrcntrl — Print the header before the RTE string (Y or N)?

Trlcntrl — Print the trailer before the RTE string (Y or N)?

Beepcntrl — Sound this many beeps (0 - 3) when this report is issued. The default is 0.

RUN_SETTINGS

This command runs the factory set commands without resetting customer tuning.

Syntax

RUN_SETTINGS {opt}

Where:

opt — Is one of the following:

- BOTH — Run the factory installed command sequences seen in the Hardware Settings and Customer Settings of the Settings window.
- CUSTOMER — Run the factory installed command sequences seen in the Customer Settings of the Settings window.
- FACTORY — Run the factory installed command sequences seen in the Hardware Settings of the Settings window.

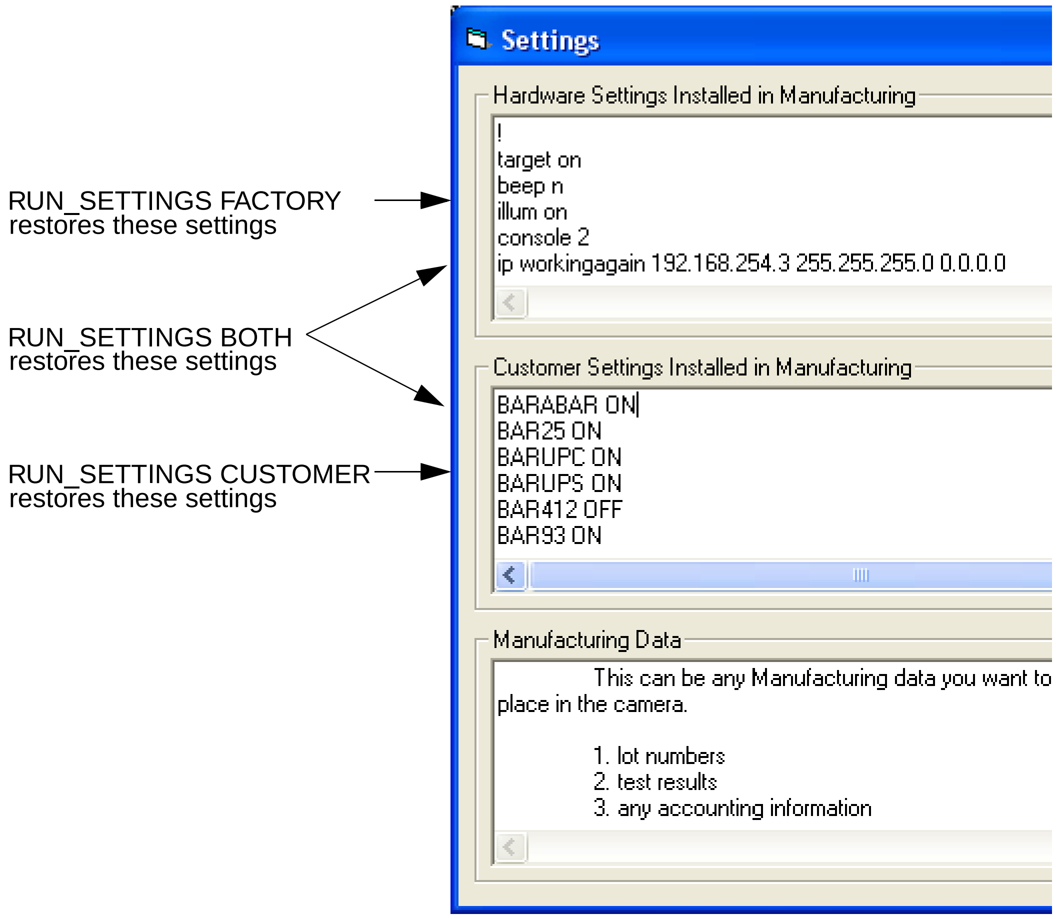
Displaying the Settings Window

Use the following procedure to display the Settings window:

1. In ReadRunner, click Add Camera.
2. Highlight a camera name and click OK.
3. Click (to select) the camera button for the camera you just added.
4. From the Help menu, click About ReadRunner.
5. Click Details.

   ReadRunner displays a screen similar to the one in Figure 1–1.

FIGURE 1–1. **Settings Window**


RUN_SETTINGS FACTORY restores these settings
RUN_SETTINGS BOTH restores these settings
RUN_SETTINGS CUSTOMER restores these settings
Settings
Hardware Settings Installed in Manufacturing
Customer Settings Installed in Manufacturing
Manufacturing Data

```
!
target on
beep n
illum on
console 2
ip workingagain 192.168.254.3 255.255.255.0 0.0.0.0
```

```
BARABAR ON
BAR25 ON
BARUPC ON
BARUPS ON
BAR412 OFF
BAR93 ON
```

This can be any Manufacturing data you want to place in the camera.

1. lot numbers
2. test results
3. any accounting information

- Hardware Settings Installed in Manufacturing — These settings depend on the hardware purchased by the customer.
- Customer Settings Installed in Manufacturing — These settings depend on the customer’s specifications.
- Manufacturing Data — These are such things as MAC address, serial numbers, and so on.

SAVE

This command saves the current configuration (regardless of boot option).

Syntax

SAVE {opt}

Where:

opt — Is either of the following:

- BOOT — Load the job on boot. This is the default.
- NOBOOT — Clear the BOOT option.

SELF_TRIGGER

This command generates periodic virtual triggers. Enabling SELF_TRIGGERing allows continuous triggering at a fixed interval without requiring external hardware. Motion and Stop-and-Scan application modes are the typical use for this command. A <duration> of 0 disables the SELF_TRIGGERing.

Syntax

SELF_TRIGGER {virtual trigger number} {duration}

Where:

Virtual trigger number — 0 - 27

duration — 0 - 30,000ms (default = 0)

SET_CRITERIA

This command allows criteria for variable testing to be configured for the IOASSIGN command.

Syntax

SET_CRITERIA {variable_name} {Type} {arg1} {arg2}

Where:

variable_name — The supported variable_names are:

- ANGLE_FAILURE
- LOCATE_FAILURE
- DECODE_FAILURE
- RTE
- PASS
- FAIL
- DV
- GOOD (reserved for verification option)
- FAIR (reserved for verification option)
- POOR (reserved for verification option)
- EXPRESSION1 (reserved for future use)
- EXPRESSION2 (reserved for future use)

Type — The supported Types are:

- BOOL

  No arguments required

  The contents of the variable_name is a Boolean data type determines ON/OFF states.

- MASK

arg1 — bit_mask entered as a hex value like 0x200
arg2 is not required

The contents of the variable_name will be ANDed with the argument value to isolate a bit in a value; in other words, a particular bit in an error code could be used to trigger a digital output.

– NOMINAL_DEV

arg1 — a nominal value
arg2 — deviation from the nominal value

This tests the contents of a variable name to see if it is in the range of “Nominal +/- Deviation.” Use this type with ANGLE_FAILURE as 0° +/- 10° will be translated correctly, while MAX_MIN will reject any entered Minimum greater than the entered Maximum.

– MAX_MIN

arg1 — maximum value
arg2 — minimum value

This tests the contents of a variable name to see if
“Minimum < value < Maximum.”

INPUT_POS — Trigger edge directions

INPUT_NEG

INPUT_BOTH

For the 3 Input Types:

arg1 — The delay (0 - 10000ms) before performing the trigger.

arg2 — The latch time (0 - 10000ms) to wait for the line to become stable.

– EXPR — (Reserved for future use)

Note: Only integer value are supported.

SET_LICENSE

This command allows you to enter manually a license key to enable Options.

Note: Licenses only activate the desired Option on the camera for which it was issued. Double check that the Serial Numbers match.

For example:

```
SET_LICENSE D8R3Z-6UUV4-3SXTM-K7LB6-D5GT9
```

Syntax

```
SET_LICENSE {key string}
```

SHOW

This command shows a group of current settings based on category.

Syntax

- Show current values of all commands in a job:

  SHOW ALL

- Show current values of specific commands in a specific job:

  SHOW {cat} {job}

  Where:

cat —	3 — Serial Reporting commands
	4 — System Configuration commands
	5 — Communication Related commands
job —	0 — Current settings (default)
	1 — Saved job
	99 — Factory defaults.

- Show current values of commands in a PID:

  SHOW {cat} [pid]

cat —	6 — Fine Tuning Symbology/Decoder
pid —	0 - 15, 99

SIGOUT

This command sets the polarity for the I/O lines.

Syntax

SIGOUT {name} {level}

Where:

name — Is one of the following:

- OPTO_OUT1
- OPTO_OUT2
- OPTO_OUT3
- GPIO_OUT1
- GPIO_OUT2
- GPIO_OUT3
- GPIO_OUT4

level — Is the level:

- H — high
- L — low

STATS

This command tracks reading statistics.

Syntax

- Display the current statistics (this is the default):

  ```
  STATS ?
  ```

- Reset the statistics:

  ```
  STATS RESET
  ```

STORE

This command stores all parameters associated with decoding a symbol. The STORE command can store up to 15 learned symbols.

Note: You must issue the SAVE command to store all 15 PIDs to Flash.

The STORE command overwrites any saved parameters previously saved with the STORE command.

Syntax

STORE {pid#}

Where:

pid# — Is the PID number:

Range: 1 - 15

STYLE

This command selects whether the Data Matrix symbol is viewed as if through a mirror or direct.

Syntax

STYLE {style}

Where:

style — Is one of the following:

- AUTO
- MIRROR
- NORMAL (default)

TARGET

This command turns the Target Laser on or off.

Syntax

TARGET {opt}

Where:

opt — Is either of the following:

- ON (default)
- OFF
- T — Turn on laser for two seconds after acquisition.

TARGET_CALIB_CONTRAST

This command sets the calibration target contrast value.

Syntax

TARGET_CALIB_CONTRAST {value}

Where:

value — 50 - 100 (default = 80)

TARGET_CALIB_REFLECTANCE

This command sets the calibration target reflectance value.

Syntax

TARGET_CALIB_REFLECTANCE {value}

Where:

value — 50 - 100 (default = 90)

TBL

This command sets the opto isolated input trigger’s bounce latch time.

Note: See also “TD” on page 1-172 and “TE” on page 1-173.

Syntax

TBL {time}

Where:

time — Is the bounce latch time:

Range: 0 - 10000ms
Default: 20ms

TD

This command sets the opto isolated input trigger’s interrupt detection delay.

Note: See also “TBL” on page 1-171 also “TE” on page 1-173.

Syntax

TD {delay}

Where:

delay — Is the delay:

Range: 0 - 10000ms
Default: 0ms

TE

This command sets the opto isolated input trigger's edge direction.

Note: See also "TBL" on page 1-171 and "TD" on page 1-172.

Syntax

TE {direction}

Where:

direction — Is one of the following:

- B — Both
- N — Negative (default)
- P — Positive

TERMINAL ECHO

This command enables/disables echoing of typed data.

Syntax

TERMINAL ECHO {opt}

Where:

opt — Is either of the following:

- ON
- OFF (default)

THRESHOLD

This command selects the threshold method in binarizing.

Syntax

THRES[HOLD] {opt}

Where:

opt — Is one of the following methods:

- GLOBAL — Should always be used unless it produces excessive SYMBOL_UNDECODABLE
- LOCAL — Effective if Data Matrix cells are not printed with equal space
- ADAPTIVE — Effective when the Data Matrix area has uneven background

The default is:

THRES:Global, Local, Adaptive

or

Global+Local+Adaptive

TIME

This command displays current time in the unit.

Syntax

- Display the current time in hex:

  TIME ?

- Display the current time in seconds, since THU JAN 01 00:00:00 1970. This message sets the time in the HawkEye™ 1500 for time stamping of images:

  TIME

TIMEOUT

This command limits the time spent in the algorithm trying to inspect/decode the symbol just lifted.

Caution: Only an Applications Engineer trained in systems use should change this parameter.

Another method to limit the time spent in the algorithm is to turn off all 2-D symbologies if the parts being read are always 1-D.

Syntax

TIMEOUT {READ | EDAC}

Where:

READ — 0 - 9999ms; default is 600ms

EDAC — 0 - 9999ms; default is 45ms

TIMESYNC

This command uses the SNTP server to synchronize local time on the camera (per RFC 868).

Syntax

TIMESYNC {interval} [server]

Where:

interval — Interval (in seconds) to query for time

server — Name of time server

TRAILER

This command sets the Report Trailer to the indicated string. Use TRAILER to “follow” the decode output with text. For example, to prompt the user for the next read:

TRAILER “Next item:”

The fourth option of the following commands determines whether or not the trailer is printed:

- BATCHFL
- DECFL
- LOCFL
- MATCHFL
- OKDEC
- OKMATCH
- RTE

Syntax

TRAILER {string}

Where:

string — (1 to 80 characters); the string may include control characters by preceding the control char with ’\’, and followed by two hex characters. For example:

""\07"" for bell

""\\"" for a backslash character

The string may include keyword substitution:

– CHECKSUM — This keyword expands to the checksum of the characters in the appropriate Output Format String where the (CHECKSUM) keyword is inserted. The two character ASCII value representation of the Hexadecimal checksum (for example, 2E) is substituted in the Output string. The checksum is calculated as the

Exclusive OR (XOR) of all the characters up to the (CHECKSUM) keyword, including the Header and the Trailer characters if (CHECKSUM) is placed at the end of the trailer string.

– TIMESTAMP — This keyword will be substituted with the current date and time.

TRIG{GER}

This command sets the triggering mode of the HawkEye™ 1500.

Syntax

- Set triggering mode to continuous read:

  TRIG C

- Set triggering mode to read only when triggered:

  TRIG T

- Enter a character string to signal a virtual trigger:

  TRIG U <chars>

  Where:

  chars — Is the set of printable characters to be used as the virtual trigger. You can define up to five printable characters. This “string” trigger can be sent over serial or the ASCII PLC TCP/IP connection port.

  Note: TRIG U with no string specified sets the trigger string to a space character “ “. This is the default trigger character.

TRIGGER_REDIRECT_ENABLE

This command controls whether or not the physical trigger will be redirected to the application in the form of an empty cycle report instead of performing the trigger. This allows the application to decide if it is an appropriate time to perform the trigger by either throwing away the cycle report or sending a VT command back to the camera.

Syntax

TRIGGER_REDIRECT_ENABLE {ON | OFF=default}

TRIGTABLE

This command alters the dynamics of hardware lines, and configures virtual triggers.

Note: I/O management of Physical INPUT and OUTPUT lines is now done with SET_CRITERIA and IOASSIGN commands. To prevent the TRIGTABLE commands from causing conflicts, TRIGTABLE is no longer allowed to work on trigger numbers 3, 4, 5, and 6.

To alter the dynamics of an actual hardware line:

- Any hardware line that can cause an interrupt (including the QuicSet® button) can have the polarity, latch and delay modified using the TRIGTABLE command. Use the following trigger number on the HawkEye™ 1500:
  - 2 — Opto Isolated Trigger Input
- There are also several GP outputs that indicate read results. These are marked as outputs. The Edge options on these outputs are valid only for those outputs and have no effect on inputs since they are virtual. Use the following trigger numbers on the HawkEye™ 1500 to affect just the output edges:
  - 12 — Opto Isolated Output 1
  - 13 — Opto Isolated Output 2
  - 14 — Opto Isolated Output 3
  - 16 — General Purpose Output 2
  - 17 — General Purpose Output 3
  - 18 — General Purpose Output 4

To program the functionality of a physical or virtual trigger:

- Each of the 28 triggers can trigger the unit. The Opto Isolated trigger is automatically wired into the unit and will be used primarily to cause physical trigger inputs to the system.

- The General Purpose inputs can be wired to the external connector (see the HawkEye™ 1500 Series User Manual) to also cause real interrupts. All of the other trigger numbers can be used as virtual triggers. The user or GUI simply needs to send the command “VT #” and the unit will respond as though an actual interrupt occurred.

Syntax

- Enable or disable the following triggers for initiating a Decode sequence:

  TRIGTABLE {opt1} {opt2}

  Where:

  opt1 — 0 - 27 or GPIO

  opt2 — Is either ENABLE or DISABLE

  If GPIO is enabled, then when an Opto Isolated Sensor Input trigger occurs, the three GPIO input lines 2, 3, 4 are read to determine a VT # to occur.

- Set trigger latch time (debounce) in milliseconds:

  TRIGTABLE {opt1} LATCH {opt3}

  Where:

  opt1 — Is the trigger number (2, 9)

  opt3 — Is the time in milliseconds (0 - 1000)

- Set delay from trigger to recognition of trigger in milliseconds:

  TRIGTABLE {opt1} DELAY {opt3}

  Where:

  opt1 — Is the trigger number (2, 9)

  opt3 — Is the time in milliseconds (0 - 1000)

- Set trigger edge detection Positive or Negative:

  TRIGTABLE {opt1} EDGE {opt3}

  Where:

opt1 — Is the trigger number (2, 9)

opt3 — Is the edge (POSITIVE or NEGATIVE)

- Set the functionality of the trigger to be a Read or Learn:

  TRIGTABLE {opt1} {opt2}

  Where:

  opt1 — Is the trigger number (0 - 27)

  opt2 — Is either READ (default) or LEARN

TTY

This command sets multiple TTY settings.

Syntax

TTY [baud] [parity] [databits] [stopbits] [flowcontrol]

Where:

Baud — 600 - 256000bps

parity — O or E or None

databits — 7 or 8

stopbits — 1 or 2

flowcontrol — Is one of the following:

- N — None
- H — Hardware
- X — XonXoff

UDP_BROADCAST

This command controls the broadcast of UDP packets, which are used to enable ReadRunner to discover new cameras on the network.

Note: DHCP should not be used if UDP_BROADCAST is turned OFF.

Syntax

UDP_BROADCAST {ON | OFF}

Where:

- ON — Turn on the broadcast of UDP packets (enable camera discovery). This is the default.
- OFF — Turn off the broadcast of UDP packets (disable camera discovery).

UNLEARN

This command opens up the decoder to read a wide variety of symbols. You can change this configuration using the LEARN command.

You can also issue an UNLEARN command using QuicSet®:

1. Assuming you are in QuicSet® mode, press and hold the QuicSet® recessed button for three to four seconds. This will perform an “unlearn” on the next acquired image.

Note: See also “LEARN” on page 1-80 and “LEARNASSIST” on page 1-82.

Syntax

UNLEARN

VERENABLE

This command allows you to select DPM verification parameters with optional Grade Ranges.

Syntax

VERENABLE {param} {Y | N} [{A | B | C | D} {B | C | D}]

Where:

param — Is one of the following:

1. VERENABLE CS {Y | N}

   Use Y or N to enable or disable cell size (CS) verification. The grade for CS is:

   A if CS >= 10
   B if CS >= 9
   C if CS >= 7
   D if CS >=5
   F if CS < 5

2. VERENABLE CO {Y | N}

   Use Y or N to enable or disable Center Offset (CO) verification. The grade for CO is:

   A if CO <= 2.5
   B if CO <= 5
   C if CO <= 7.5
   D if CO <= 10
   F if CO > 10

3. VERENABLE SO {Y | N}

   Use Y or N to enable or disable Size Offset (SO) verification. The grade for SO is:

   A if SO <= 2.5
   B if SO <= 5
   C if SO <= 7.5
   D if SO <= 10
   F if SO > 10

4. VERENABLE CM {Y | N}

   Use Y or N to enable or disable Cell Modulation (CM) verification. The grade for CM is:

   A if CM >= 90
   B if CM >= 80
   C if CM >= 70
   D if CM >= 60
   F if CM < 60

5. VERENABLE BM {Y | N}

   Use Y or N to enable or disable Border Match (BM) verification. The grade for BM is:

   A if BM >= 95
   B if BM >= 90
   C if BM >= 85
   D if BM >= 80
   F if BM < 80

6. VERENABLE SC {Y | N}

   Use Y or N to enable or disable Symbol Contrast (SC) verification. The grade for SC is:

   A if BM >= 70
   B if BM >= 55
   C if BM >= 40
   D if BM >= 20
   F if BM < 20

7. VERENABLE AN {Y | N}

   Use Y or N to enable or disable Axial Nonuniformity (AN) verification. The grade from AN is:

   A if AN <= 6
   B if AN <= 8
   C if AN <= 10
   D if AN <= 12
   F if AN > 12

8. VERENABLE PG {Y | N}

Use Y or N to enable or disable Print Growth (PG) verification. The grade for PG is:

A if -15 <= PG <= 15
B if -21 <= PG <= 21
C if -26 <= PG <= 26
D if -30 <= PG <= 30
F if PG > 30 or PG < -30

9. VERENABLE UEC {Y | N}

Use Y or N to enable or disable Unused Error Correction (UEC) verification. The grade for UEC is:

A if UEC >= 62
B if UEC >= 50
C if UEC >= 37
D if UEC >= 25
F if UEC < 25

10. VERENABLE AD {Y | N}

Use Y or N to enable or disable Angle of Distortion (AD) verification. The grade for AD is:

A if -2 <= AD <= 2
B if -4 <= AD <= 4
C if -6 <= AD <= 6
D if -7 <= AD <= 7
F if AD > 7 or AD < -7

VERIFY

This command turns on/off AIM verification. The AIM symbol quality parameters contain Symbol Contrast, Print Growth, Axial Nonuniformity, and Unused Error Correction (UEC). You must have one of the following keywords specified in the OKDEC output string to receive the verification results.

You must have one of the following keywords specified in the OKDEC output string:

- (VERI_DETAIL) — Detailed verification data, separated by semicolons (";"), but only when Verification is enabled. This data includes the overall grade, grade contrast, contrast, grade axial number, AIM axial non-uniformity value, grade print growth, print growth value in X, print growth value in Y (1 for 100% and 0.35 for 35%, etc.), grade error correction, number of error bits, and the UEC value.
- The AIM Print Growth grade converted to HawkEye™ 1500 serial's Print Growth Percentage (PGP) is:

– A(4) when	-15% <= PGP <= 15%
– B(3) when	-21% <= PGP <= 21%
– C(2) when	-25% <= PGP <= 25%
– D(1) when	-30% <= PGP <= 30%
– F(0) when	PGP < -30% or PGP> 30%

- (VERI_FORMATTED) - Formatted verification data. This is similar to VERI_DETAIL, but each value is identified by an acronym title and is separated by spaces.
- (VERI_GRADE) - Overall verification grade as a number, but only when Verification is enabled.

Example

A Data Matrix with the following information:

Matrix size: 20x20 with 22 data codewords and 18 error correction codewords

Encode data: HE1500

Symbol quality:	Contrast	62%
	Print Growth Percentage in X	-13%
	Print Growth Percentage in Y	24%
	Axial Nonuniformity	0.07
	UEC	11% (8 bits / 1 codeword)

Verification results output:

1. With command line OKDEC Y "(DATA);(VERI_DETAIL)\0d\0a" or set the ReadRunner output setting with (DATA);(VERI_DETAIL)\0d\0a in the decode pass area.

   HE1500;3;3;62;4;0.07;2;0.13;0.24;4;8;0.11

2. With command line OKDEC Y "(DATA);(VERI_FORMATTED)\0d\0a" or set the ReadRunner output setting with (DATA);(VERI_FORMATTED)\0d\0a in the decode pass area.

   HE1500;OG:3 CG:3 C:62 nG:4 n:0.07 PGP:2 PGx:0.13 PGy:0.24 GUEC:4 B#: 8;0.11

3. With command line OKDEC Y "(DATA);(VERI_GRADE)\0d\0a" or set the ReadRunner output setting with (DATA);(VERI_GRADE)\0d\0a in the decode pass area.

Syntax

VERIFY {none | AIM}

Where:

None — Disables verification

AIM — Enables AIM verification

Additional Verification Options

If you purchased the verification option for your HawkEye™ 1500 camera, additional verification modes are available:

VERIFY {none | AIM | IAQG | ISO | ISO AIM | DPM | ANSI}

- VERI_STATUS — Report the verification status (see "VERSTATUS" on page 1-197). VERI_STATUS is reported as 3 = Good, 2 = Fair, 1 = Poor.

- VERI_GRADE — Report the overall grade as A = 4, B = 3, C = 2, D = 1, F = 0.
- VERI_DETAIL — Report parameters and/or grades in a semicolon delimited string.
- VERI_FORMATTED — Same as VERI_DETAIL except each parameter is pre-pended with its name.

DPM — The DPM verification always measures and reports 10 different parameters and their grades. Each grade is reported as 4 = A, 3 = B, 2 = C, 1 = D, and 0 = F. You have the option to enable all or a subset of the 10 parameters for determining the overall grade. The overall grade is the lowest grade received by all enabled verification parameters.

- VERI_STATUS — Report the verification status (see “VERSTATUS” on page 1-197).
- VERI_GRADE — Report the overall grade as A, B, C, D, or F.
- VERI_DETAIL — All parameters (enabled or not) and their grades are reported.
- VERI_FORMATTED — Same as VERI_DETAIL except each parameter is pre-pended with its name.

See the HawkEye™ 1500 Series Verification Manual for additional details for each of the optional verification types.

VERIFY AIMDPM

This command enables AIM DPM-1-2006 verification.

Before using this command, apply the following commands:

- RETRY NONE
- PHOTOMETRY MANUAL
- ILLUMINATION ON

After enabling AIM DPM-1-2006 verification mode, do not use the preceding commands to change the configuration.

VERSION

This command lists the current version of the software.

Syntax

VER[SION]

VERSTATUS

This command selects the verification grade ranges. The verification status indicates whether a Data Matrix mark is good (2), fair (1), or poor (0), based on the overall grade and two threshold values set using the command.

The grades grade1 and grade2 are threshold values for good and fair marks, respectively. The valid values for grade1 and grade2 are A, B, C, or D. The value grade1 can be higher than or equal to but not lower than the value grade2.

For example, the command:

```
VERSTATUS B D
```

configures the reader to report verification status to be 2 for grade A or B, 1 for grade C or D, and 0 for grade F.

Syntax

VERSTATUS {param1} {param2}

Where:

param1 — Is A, B, C, D

- This parameter is the lowest grade considered Good

param2 — Is B, C, D

- This parameter is the lowest grade considered Fair. Param2 can never be greater than param1. If param2 = param1, verification will only give Good and Poor results, since there is no range of grades for Fair.

VIEW

This command displays the stored values in the PID index.

Syntax

VIEW {pid#}

Where:

pid# — Is one of the following:

- 0 — Current working area (default)
- 1 - 15
- 99 — Factory defaults

VT

This command causes a Virtual Trigger to occur (the application must be waiting for it).

Syntax

VT {trigger}

Where:

trigger — Is the trigger number:

Range: 0 - 27

Note: Triggers numbered 2, 4, 5, & 6 are actual GPIO lines and are considered physical; the remaining triggers are considered virtual.

WARP

This command sets the matrix warping speed. Typically, Fast produces satisfactory read rates with much higher speed, although Slow sometimes can be more robust than Fast for very poor images.

Syntax

WARP {opt}

Where:

opt — Is either of the following:

- Fast (default)
- Slow

WAVELENGTH

This command sets the customer light source's wavelength in nm; it is used only for reporting. The default value is 640 nm.

Syntax

WAVELENGTH {280-880}

WIDTH

This command sets the expected Data Matrix width.

Note: See also “HEIGHT” on page 1-70 and “RATIO” on page 1-142.

Syntax

WIDTH {opt}

Where:

opt — Is either of the following:

- width — Is the width:

  Range: 20 - 1024
- AUTO — Autoselect (default)

XDIMENSION

This command specifies the maximum and minimum dimension range in 0.1 mil increments. To specify 7.5 mils, enter:

```
75
```

Syntax

XDIMENSION {max: 11 to 990} {min: 10 to (max-1)}

- Max default is 250
- Min default is 75

CHAPTER 2

ReadRunner Programming Reference

Introduction

The ReadRunner development libraries allow for easy development of custom applications that interface to the HawkEye™ 1500 family of readers. A unique approach, described herein, significantly simplifies user programming – yet provides maximum flexibility. The key concept that facilitates this simplicity is an object called the MiCoordinator. The MiCoordinator object allows the various components to “talk” to each other behind the scenes – so that, in most cases, the amount of custom code that needs to be written is surprisingly small.

Libraries

There are three libraries that contain components of interest:

- **MIOBJ.DLL** – (Microscan HawkEye™ Reader Object Type Library) This library contains lower level objects that provide the foundation for the other libraries. The MiCoordinator object mentioned above resides here, as well as the MiCycleReport object, which wraps data (reports and images) coming from a HawkEye™ 1500 reader. This library also contains objects that manage connections to HawkEye™ 1500 devices. In most cases, you will not have to deal with these connection objects directly, as these are, in turn, wrapped in even easier to use objects.

- **MIOBJUI.DLL** – (Microscan HawkEye™ Reader ActiveX Library) This library contains several user interface objects. The most important object in this library is MiImageView, which provides an easy way of viewing images that come from a HawkEye™ 1500.
- **RRKIT.OCX** – (Microscan ReadRunner ActiveX Library) This library provides many useful user interface and connection objects. This will likely be the starting point for any custom-programming project. The library itself is written in Visual Basic, which illustrates the depth available to the programmer.

Note: To include these libraries in a VB project, first include MIOBJ.DLL as a Project Reference, and then add MIOBJUI.DLL and RRKIT.OCX as Project Components.

Important Objects

- MiCoordinator — This object provides the foundation on which most of the other objects rely. Even if you do not use this object directly, understanding what is does will provide a better understanding of the other components.
  - **Provides behind the scenes magic glue**. In fact, all instances of the MiCoordinator object within the same process are actually the exact same object. To illustrate, imagine two forms, each declaring their “own” new MiCoordinator object. Since they are both actually the same object, one control can call a method of its “own” MiCoordinator, which results in an event to the clients of all MiCoordinators.
  - **Maintains list of discovered devices (**HawkEye™ **1500’s)**. This list is provided via the Devices property. For example, in VB, you could do the following:

    ```
    dim coord as new MiCoordinator
    Dim d as MiDevice
    for each d in coord.Devices
           ‘ do something with device
    next d
    ```
  - **Provides concept of “Focus Device”.** In many cases, an application is only concerned with a single device at a time. When this is the case, the device to work with can be specified using the DeviceFocusSet method

of the MiCoordinator. This, in turn, will cause the OnDeviceFocus message to be sent to every other MiCoordinator object within the same process. Note that this does not actually make a connection. However, the ImageConnection and ReportConnection objects, as well as some of the user interface objects, use a MiCoordinator and make their connections as a result of receiving the OnDeviceFocus event.

– **Provides general message broadcasting.** The “magic glue” property of the MiCoordinator also can come in handy for applications that simply want to send a general message to other MiCoordinator users. Calling the BroadcastMessage method allows you to send a message string to all other MiCoordinator users, who will receive it via the OnBroadcastMessage event.

- **MiASCIIConnection, MiImageConnection, and MiReportConnection** — These objects represent low-level connections to a device. The MiASCIIConnection permits sending and receiving ASCII characters to/from a device, and is the only type of connection possible via a serial port. Using a MiASCIIConnection, images and reports are transferred using the XMODEM protocol. The MiImageConnection and MiReportConnection objects are for transferring images and reports using TCP over Ethernet.

Note: We recommended that you not use these connection objects directly, as they are wrapped within the easier to use connection objects described next.

- **ImageConnection, ReportConnection** — These objects wrap the low-level connections described above and make custom programming much easier. Most custom applications will start by declaring one or both of these objects. Each combines a MiASCIIConnection (to send/receive ASCII data such as commands), a MiCoordinator (so a connection can be established automatically to the focus device), and either a MiImageConnection or a MiReportConnection. It also abstracts the differences between serial and TCP based connections by using an ASCII protocol to transmit images over a serial port, and TCP if an Ethernet connection is available. It also provides some convenient properties to test the connection, such as IsConnected, IsSerial, etc. The lower level connections are also available as properties. Many of the events from the contained MiConnections, MiCoordinator, and the focus device, are reflected to the user. It is also possible to disable the auto-connect behavior of these controls.

- **MiCycleReport** — After a connection is established, an OnNewRecord event is sent each time new data (image or report) is available. The parameter provided in this event is a MiCycleReport object. The same data type is provided in both image and report cases except that reports have an empty field for the image. There is a great deal of data provided in each MiCycleReport, including counters, pass/fail status, decoded data, timing information, photometry data, ROI data, etc.
- **MiImageView** — This control permits an easy way to view images contained in a MiCycleReport. It has options to display the decoded data and ROI as an overlay on the image. Multiple images can be displayed by enabling the Filmstrip Mode property. There is also the ability to overlay custom graphics and, in fact, you can even create and overlay custom draggable user interface objects with handles, such as an ROI.
- **ReportGrid —** A control that displays most of the data contained in a MiCycleReport in a spreadsheet format. You place this control directly on a form and then call its NewRecord method when you receive the OnNewRecord event from a connection.
- **ReportChart —** A control that displays timing and trend data from the MiCycleReport. You place this control directly on a form and then call its NewRecord method when you receive the OnNewRecord event from a connection.
- **Terminal** — This control automatically connects to the focus device and permits sending ASCII commands, much like a dumb terminal. It provides programmable buttons that can send automatically a series of commands when pressed.
- **Application, AppSetting** — Sending commands to a HawkEye™ 1500 is very straightforward simply by using the CommandSend method of an ImageConnection or ReportConnection. To query the state of most settings, you can send a command followed by a “?”. Often, it is the case that you might desire to query the state of a setting, change one thing (such as photometry gain while leaving the exposure alone) and send the command back with the change. You can use the Application object to make this logic easier to manage. The Application object provides methods for querying a setting and accessing the parameters individually. It is capable of querying any number of settings. When you write the settings back, only the settings that has actually changed will be written.

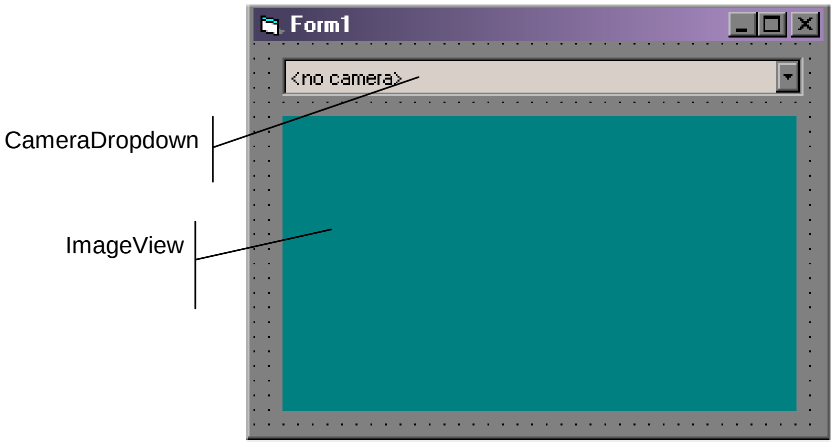
Example 1 — A Simple Monitoring Application

Here's an application that has an apparently challenging set of goals:

- Scan the network and enumerate all found HawkEye™ 1500s in a dropdown box.
- Establish an image connection to the camera selected in the dropdown.
- Draw each image as received from the camera.

This application can be "written" by writing no code at all. Although the sample project is included, it's worth the 3 minutes or less to walk through this example manually.

1. Open Visual Basic and create a new "Standard EXE" project.
2. Select Project->Components… and check the Microscan ReadRunner ActiveX Library (RRKIT.OCX). Notice the new component icons that appear in the component toolbar.
3. Select the CameraDropdown component and place across the top of the form.
4. Select the ImageView component and size to occupy the rest of the form.



5. You're done. You can now run the application.

How This Works

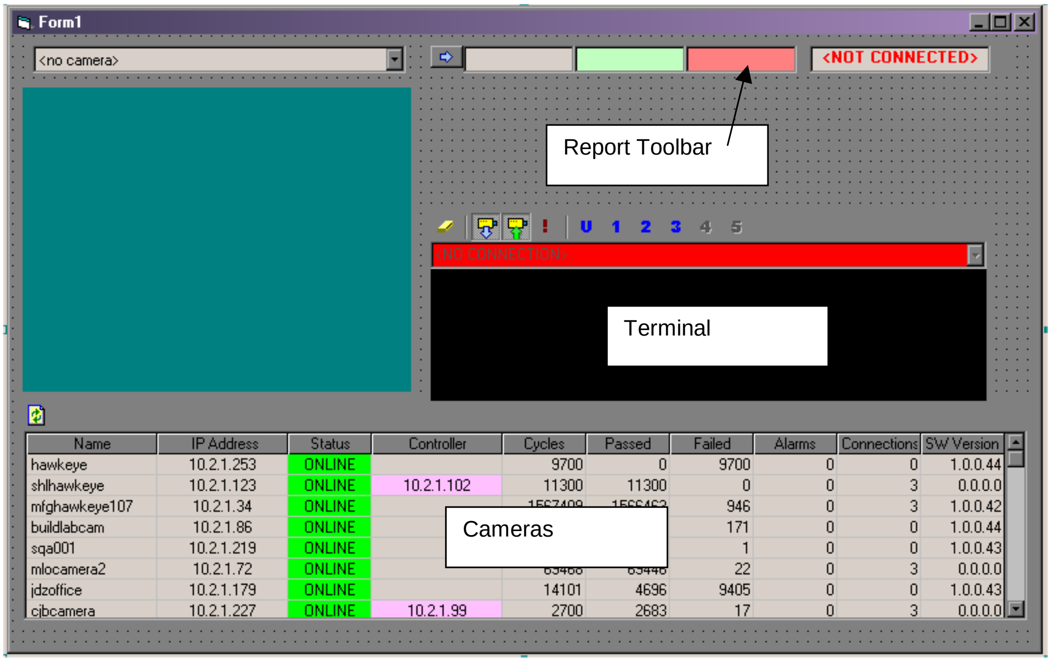
The two components used in this application, CameraDropdown and ImageView, both use the MiCoordinator object. The CameraDropdown control first clears the Device List in the MiCoordinator, and then handles the OnDeviceDiscovered event, filling up the dropdown for each newly discovered camera on the network. When a device is selected via the dropdown, the DeviceFocusSet method is called with the name of the chosen device.

The ImageView control uses an ImageConnection, which will establish automatically a connection as a response to the Device Focus being set. Once the connection is established, each image will be received via the ImageConnection.OnNewRecord event. The MiCycleReport provided in this event is then passed to the MiImageView.NewRecord method, which will display the image and any overlay graphics.

Taking This Example Further

It is instructive to add some other controls to the current example.

1. Expand the dimensions of the form as much as possible to allow other controls to be placed on it.
2. Select the Cameras control and position along the bottom of the form.
3. Select the Report Toolbar and place on the form.
4. Select the Terminal control and place on the form.



Name	IP Address	Status	Controller	Cycles	Passed	Failed	Alarms	Connections	SW Version
hawkeye	10.2.1.253	ONLINE		9700	0	9700	0	0	1.0.0.44
shlhawkeye	10.2.1.123	ONLINE	10.2.1.102	11300	11300	0	0	3	0.0.0.0
mfghawkeye107	10.2.1.34	ONLINE		1567409	1566463	946	0	3	1.0.0.42
buildlabcam	10.2.1.86	ONLINE		[illegible]	[illegible]	171	0	0	1.0.0.44
sqa001	10.2.1.219	ONLINE		[illegible]	[illegible]	1	0	0	1.0.0.43
mlocamera2	10.2.1.72	ONLINE		63468	63446	22	0	3	0.0.0.0
jdzoffice	10.2.1.179	ONLINE		14101	4696	9405	0	0	1.0.0.43
cjbcamera	10.2.1.227	ONLINE	10.2.1.99	2700	2683	17	0	3	0.0.0.0

5. You're done. You may now run the application. You can send commands to the selected unit via the Terminal. The Cameras spreadsheet control shows detailed status of cameras on the network. The Report Toolbar shows some basic counters.

Summary

This example illustrates some of the power behind the ReadRunner development environment. Impressive capability is possible without even writing a line of code.

Example 2 — Handling Images and Reports

In most cases, it is likely that the data coming from a connection to a HawkEye™ 1500 will need to be manipulated in some way, perhaps simply to note pass/fail status, or perhaps to log data or images to a file. In this event, we will need to take control over some of the automatic functionality illustrated by the previous example. Even so, the programming is not at all difficult.

The goals of this example are:

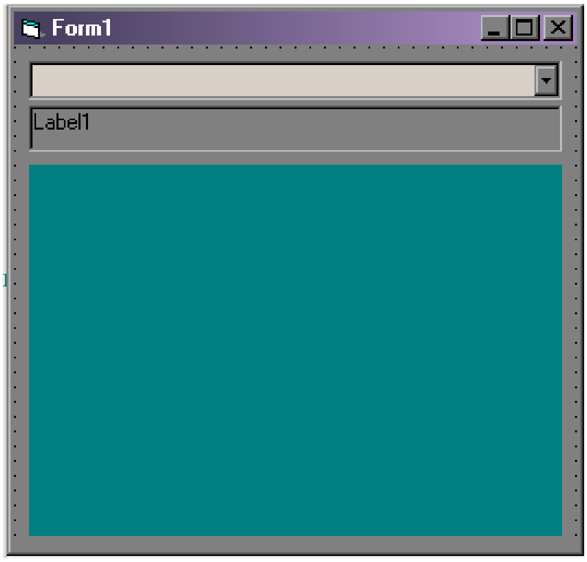
- Establish two connections to an HawkEye™ 1500, one for images and one for reports.
- Update a text label with the decode string, and change the background color of the label to green for decode pass and red for decode fail.
- Update a displayed image – but only show failures.

This time, we will be using all three libraries. Again, although the sample code is included, it is worth the time to work through this example from scratch.

1. Create a new “Standard EXE” project.
2. Select Project->References… and check the Microscan HawkEye™ Reader Object Type Library (MIOBJ.DLL).
3. Select Project->Components… and check BOTH the Microscan ReadRunner ActiveX Library (RRKIT.OCX) and the Microscan HawkEye™ Reader ActiveX Library (MIOBJUI.DLL). Notice the new component icons that appear in the component toolbar.
4. Select the CameraDropdown component and place across the top of the form.
5. Select a standard Label control and place on the form. In the VB properties window, change the BorderStyle for Label1 to “Fixed Single”.
6. Select the MiImageView component and position on the form.

   Note: This is NOT the same as the ImageView component we used in the previous example.

   Here’s how the form should look:



7. Now, we’re going to have to write some code. Go to the Code window for Form1 (View->Code, or double click on the form). First declare, create, and clean up the connections:

```
Option Explicit


' Declare connections WithEvents so we’ll receive events
Private WithEvents iconn As ImageConnection
Private WithEvents rconn As ReportConnection

Private Sub Form_Load()
    ' create the connection objects
    Set iconn = New ImageConection
    Set rconn = New ReportConnection
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' cleanup when exiting
    Set iconn = Nothing
    Set rconn = Nothing
End Sub
```

8. Select the rconn object in the VB Object dropdown, and then select the OnNewRecord event (if this didn't happen automatically). This will make an event handling subroutine for you to fill in. Enter the following code:

```
Private Sub rconn_OnNewRecord(ByVal objReport As
MIOBJLib.IMiCycleReport)
    Label1 = objReport.ReportDecode.DecodeDataFormatted
    If objReport.Passed Then
        Label1.BackColor = vbGreen
    Else
        Label1.BackColor = vbRed
    End If
End Sub
```

9. Select the iconn object in the VB Object dropdown, and then select the OnNewRecord event. Enter the following code for the event handler:

```
Private Sub iconn_OnNewRecord(ByVal objReport As
MIOBJLib.IMiCycleReport)
    ' only display failed images
     If Not objReport.Passed Then
            MiImageView1.NewRecord objReport
    End If
End Sub
```

10. You're done. You can now run the project.

How This Works

By default, the ImageConnection and ReportConnection objects will connect automatically to the focus device as selected by the CameraDropdown. When the connections are established, OnNewRecord events are sent whenever there is new data available on the connection. The object passed as the parameter in this event is the MiCycleReport object – the same object is used for both images and reports. This object contains a great deal of detail about the completed cycle and is well worth browsing with the VB object browser. We use the DecodeDataFormatted field to fill the label with text, and the Passed Boolean to determine the label background color and to decide whether to display the image.

Taking This Example Further

When the "<no camera>" option is selected from the CameraDropdown – thereby disconnecting, we may want to clear the contents of the label as well as the displayed image. This will illustrate how to do something when a connection status changes.

1. Select the rconn object in the VB Object dropdown, and then select the OnConnectionReset event. Enter the following code:

```
Private Sub rconn_OnConnectionReset()
    If Not rconn.IsConnected Then
        Label1 = ""
        Label1.BackColor = &H8000000F
        MiImageView1.ClearRecords
    End If
End Sub
```

Note: Although the report and image connections both raise the OnConnectionReset event, in this example, we can assume they are both connected to the same camera, so we can put the logic in only one place.

2. The OnConnectionReset event handler is a convenient place to put initialization logic. Therefore, call the event procedure directly from the Form_Load method:

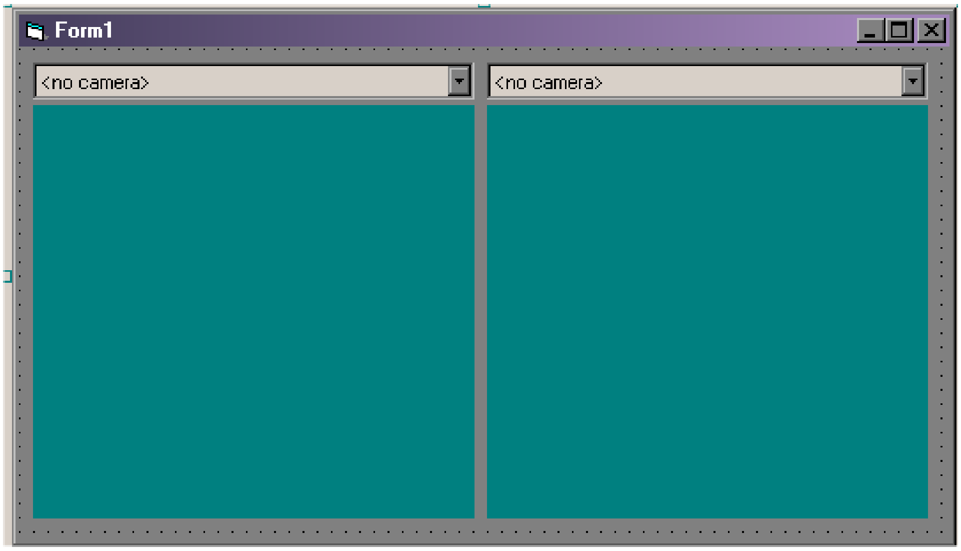
```
Private Sub Form_Load()
    ' create the connection objects
    Set iconn = New ImageConection
    Set rconn = New ReportConnection

    ' call OnConnectionReset directly so
    ' initialization logic is all in one place
    rconn_OnConnectionReset
End Sub
```

3. Now, when you run the project, the image and label controls will be cleaned up as expected when you disconnect.

Example 3 — Connecting to Two Cameras at Once

It is likely that most applications will only connect to a single camera at a time. Therefore, the default behavior simplifies this case. However, this behavior can be overridden, if necessary. Here is an example that connects to two cameras.

1. Create a new “Standard EXE” project.
2. Select Project->References… and check the Microscan HawkEye™ Reader Object Type Library (MIOBJ.DLL).
3. Select Project->Components… and check BOTH the Microscan ReadRunner ActiveX Library (RRKIT.OCX) and the Microscan HawkEye™ Reader ActiveX Library (MIOBJUI.DLL).
4. Select the CameraDropdown component and place one stretching about half way across the top of the form. Then, select another one and place it next to the first one.
5. Select the MiImageView component and position it below the first CameraDropdown. Then, add a second MiImageView control under the second CameraDropdown.



6. Enter the code:

```
Option Explicit
Dim WithEvents m_cam1 As ImageConnection
Dim WithEvents m_cam2 As ImageConnection

Private Sub Form_Load()
    CameraDropdown1.AutoConnect = False
    CameraDropdown2.AutoConnect = False
    Set m_cam1 = New ImageConnection
    Set m_cam2 = New ImageConnection
    m_cam1.AutoConnect = False
    m_cam2.AutoConnect = False
End Sub

Private Sub Form_Unload(Cancel As Integer)
    m_cam1.DisconnectAll
    Set m_cam1 = Nothing
    Set m_cam2 = Nothing
End Sub

Private Sub CameraDropdown1_CameraSelected(selDevice As
String)
   m_cam1.DeviceName = selDevice
End Sub

Private Sub CameraDropdown2_CameraSelected(selDevice As
String)
    m_cam2.DeviceName = selDevice
End Sub


Private Sub m_cam1_OnNewRecord(ByVal objReport As
MIOBJLib.IMiCycleReport)
    MiImageView1.NewRecord objReport
End Sub


Private Sub m_cam2_OnNewRecord(ByVal objReport As
MIOBJLib.IMiCycleReport)
    MiImageView2.NewRecord objReport
End Sub
```

The key difference between this example and the previous ones is that the AutoConnect property of both CameraDropdowns and both ImageConnections are set to False. This means that we have to manually handle what happens when a camera is selected in the dropdowns. You see how this is done via the CameraSelected event handlers. Setting the

DeviceName property of an ImageConnection is all that is necessary to establish a connection to the named device. Since we have two connections, we also need to handle both OnNewRecord events and send the images to the corresponding MiImageView controls.

7. You’re done. You can run the project. It is even possible to make two separate connections to the same camera.

Example 4 — Controlling the HawkEye™ 1500

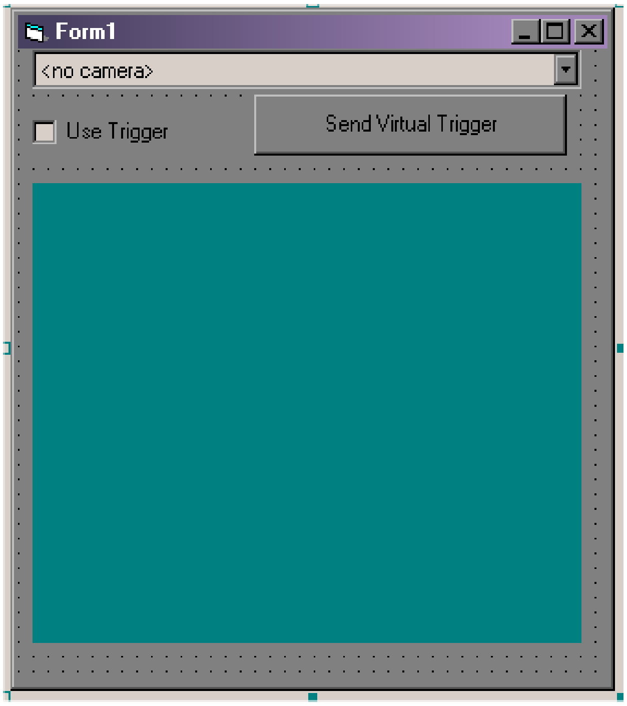
This example illustrates how to send commands to the HawkEye™ 1500. Any of the RRKIT.DLL connection objects (ReportConnection, ImageConnection, CommandConnection) can send commands to the connected device using the CommandSend() method. In this example, we will use an ImageConnection to do this, since we just happen to be displaying images as well. This example also serves as an introduction to the Application and AppSetting objects. Here are the goals for this example:

- Establish a connection to a HawkEye™ 1500.
- Use a Checkbox control to switch between triggered and continuous acquisition modes.
- Send virtual triggers by way of a push button.
- Display images.

We will once again use the CameraDropdown to select a device. We start out with a variation of the previous example:

1. Create a new “Standard EXE” project.
2. Select Project->References… and check the Microscan HawkEye™ Reader Object Type Library (MIOBJ.DLL).
3. Select Project->Components… and check BOTH the Microscan ReadRunner ActiveX Library (RRKIT.OCX) and the Microscan HawkEye™ Reader ActiveX Library (MIOBJUI.DLL).
4. Select the CameraDropdown component and place across the top of the form.
5. Select a Checkbox control and place on the form. In the properties, change the name of the control to UseTrigger and the caption to read “Use Trigger”.
6. Place a CommandButton on the form. In the properties, change the name of the control to VirtualTrigger and the caption to read “Send Virtual Trigger”.

7. Select the Imageview component and position on the form.



8. Similar to the previous example, enter the following code to initialize an ImageConnection, connect it to the MiImageView control and handle the OnConnectionReset event to set the initial state of the controls. Also, notice that when the OnConnectionReset event is received and the iconn.IsConnected is true, then we use the iconn.CommandSend method to send the “CONTROL” command to the camera. Otherwise, we disable the Checkbox and CommandButton controls.

```
Option Explicit
Dim WithEvents iconn As ImageConnection

Private Sub Form_Load()
      Set iconn = New ImageConnection
      iconn_OnConnectionReset
End Sub

Private Sub Form_Unload(Cancel As Integer)
      Set iconn = Nothing
End Sub
```

```
Private Sub iconn_OnConnectionReset()
    If iconn.IsConnected Then
        iconn.CommandSend "CONTROL"
    Else
        UseTrigger.Enabled = False
        VirtualTrigger.Enabled = False
    End If
End Sub
```

9. We want to enable the checkbox and button only after we establish control as a result of sending the “CONTROL” command. To do this, select the iconn object and select the OnControlStatusChanged event to add an event handler. Although this event returns a integer code signifying the status of the camera, we can simply check if we now have control by using the HaveControl property of the ImageConnection:

```
Private Sub iconn_OnControlStatusChanged(ByVal newStatus As
Long)
    UseTrigger.Enabled = iconn.HaveControl
    VirtualTrigger.Enabled = iconn.HaveControl
End Sub
```

10. To have the checkbox reflect whether the camera is triggered or not, we need to do a “Query” of the TRIG command. The two possible states are “TRIG C” for continuous and “TRIG T” for triggered. A query consists of sending the “TRIG ?” command to see what the current setting is. It is possible to use CommandSend to do everything, but we will instead use the Application object. This object simplifies querying and changing settings, and is particularly useful when the commands are complex and/or you are changing more than one setting at a time. Add the following code:

At the top of the module declare the following:

```
Dim app As New Application
Dim bInUpdateVisuals as boolean
```

Change the code in the OnConnectionReset event to read:

```
Private Sub iconn_OnConnectionReset()
    If iconn.IsConnected Then
       app.Query iconn.ASCIIConnection, "TRIG" ‘ <-----add this line
        iconn.CommandSend "CONTROL"
    Else
        UseTrigger.Enabled = False
        VirtualTrigger.Enabled = False
    End If
End Sub
```

Notice that the app.Query command requires that you pass an MiASCIIConnection object as the first parameter. Use the ImageConnection.ASCIIConnection property in this case.

Add a new subroutine to handle updating the checkbox:

```
Private Sub UpdateVisuals()
    bInUpdateVisuals = True
    Dim bTriggered As Boolean
    bTriggered = app.Setting("TRIG").Params(0) = "T"
    UseTrigger.Value = IIf(bTriggered, vbChecked, vbUnchecked)
    VirtualTrigger.Enabled = bTriggered
    bInUpdateVisuals = False
End Sub
```

There are a couple of things to notice here. The Application object consists of a collection of AppSetting objects. You index this collection with the name of the setting, in this case “TRIG”. Each AppSetting can have multiple parameters, which are indexed using a zero based index. Therefore, the line

```
    bTriggered = app.Setting("TRIG").Params(0) = "T"
```

is setting the state of the Boolean variable *bTriggered*, depending on whether or not the first parameter is “T”.

Also, we set the global variable *bInUpdateVisuals* to True at the start and False at the end of this subroutine. This is because we are setting the checkbox state in this routine which will, in turn, cause an event to happen. As you will see, we don’t really want to handle the event in this case, since the event will call UpdateVisuals and we’d be stuck in an infinite recursion. We will use this global boolean to prevent this.

Now we can handle the checkbox Click event:

```
Private Sub UseTrigger_Click()
    If bInUpdateVisuals Then Exit Sub
    If UseTrigger.Value = vbChecked Then
        app.Setting("TRIG").SetParam 0, "T"
    Else
        app.Setting("TRIG").SetParam 0, "C"
    End If
    app.WriteSettings iconn.ASCIIConnection
    UpdateVisuals
End Sub
```

The first line exits immediately if we find we got here via UpdateVisuals to prevent recursion.

Next, you can see how the SetParam method is used; in this case, we are setting the first parameter to either “T” or “C”, depending on the state of the check box.

To send all the changed settings back to the camera (in this case, there’s only one), we use the WriteSettings method of the Application object. Again, you need to provide an MiASCIIConnection, so use the ASCIIConnection property of iconn.

We then call UpdateVisuals to make sure the checkbox reflects the current state of the “TRIG” command.

11. Lastly, to send the virtual trigger command, handle the Click event for the VirtualTrigger button:

```
Private Sub VirtualTrigger_Click()
    ' send a virtual trigger command
    iconn.CommandSend "VT 2"
End Sub
```

12. You’re done. Run the project.

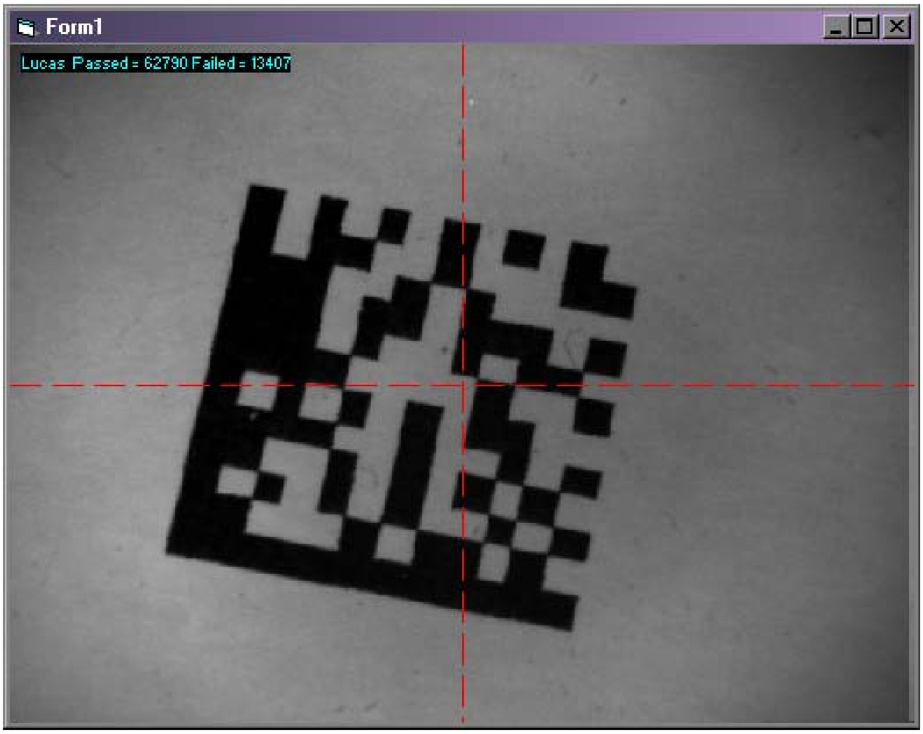
Example 5 — Image Overlay Graphics and Auto Connection

This example illustrates how to make a custom graphic overlay. It also demonstrates how to connect automatically to a camera without using the CameraDropdown control as in previous examples:

1. Establish an image connection to a known HawkEye™ 1500 by name.
2. Display an image, but disable the normal overlay graphics and replace them with custom graphics consisting of:
   - A crosshair display
   - Decode string and pass/fail counters at the top of the image

This project will begin in standard fashion:

1. Create a new "Standard EXE" project.
2. Select Project->References… and check the Microscan HawkEye™ Reader Object Type Library (MIOBJ.DLL).
3. Select Project->Components… and check BOTH the Microscan ReadRunner ActiveX Library (RRKIT.OCX) and the Microscan HawkEye™ Reader ActiveX Library (MIOBJUI.DLL).
4. Select the MiImageView component and position on the form.
5. Select the MiImageView1control and, in its properties, set the ShowCycleGraphics, ShowCycleText and ShowOverlay properties to False. In this example, we will take over all drawing ourselves.



The complete code for this example follows:

```
Option Explicit
Const MY_CAMERA = "jdzoffice" ‘ <-------- replace with your camera name


Const TRANSPARENT = 1
Const OPAQUE = 2

Implements MiDrawObj ‘ <---------- makes this form a DrawObj


Dim WithEvents coord As MiCoordinator
Dim WithEvents iconn As ImageConnection

Dim saveReport As MiCycleReport
Dim myFont As New StdFont

Private Sub Form_Load()
    Set coord = New MiCoordinator
    Set iconn = New ImageConnection
    coord.ClearDevices
    With myFont
        .Size = 7
        .Name = "Arial"
    End With
    MiImageView1.DrawObjAdd Me, "overlay"
    iconn_OnConnectionReset
End Sub

Private Sub Form_Unload(Cancel As Integer)
    coord.DeviceFocusSet ""
    Set coord = Nothing
    Set iconn = Nothing
    Set saveReport = Nothing
End Sub

Private Sub iconn_OnConnectionReset()
    If Not iconn.IsConnected Then
        MiImageView1.ClearRecords
    End If
End Sub

Private Sub coord_OnDeviceDiscovered(ByVal newDevice As
MIOBJLib.IMiDevice)

    If newDevice.Name = MY_CAMERA then
        coord.DeviceFocusSet MY_CAMERA

    End If
End Sub

Private Sub iconn_OnNewRecord(ByVal objReport As
MIOBJLib.IMiCycleReport)
    MiImageView1.NewRecord objReport
    Set saveReport = objReport
End Sub

Private Sub Form_Resize()
    On Error Resume Next
    MiImageView1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub

Private Sub MiDrawObj_Draw(ByVal midc As MIOBJUILibCtl.IMiDC)
    If iconn.IsConnected Then
        If Not saveReport Is Nothing Then
            midc.SetBkMode OPAQUE
            midc.SetFont myFont
            midc.SetTextColor vbCyan
            Dim s As String
            With saveReport
                s = saveReport.ReportDecode.DecodeDataFormatted
                s = Replace(s, vbCrLf, "")
                s = s & " Passed = " & .CountDecodes
                s = s & " Failed = " & .CountDecodeFail
            End With
            midc.TextOut 8, 6, s

        End If
        midc.SetBkMode TRANSPARENT
        midc.SetPen 1, 1, vbRed
        midc.MoveTo 320, 0
        midc.LineTo 320, 480
        midc.MoveTo 0, 240
        midc.LineTo 640, 240
   End If
End Sub
```

Now, let's walk through this project step by step.

1. The MY_CAMERA constant at the top should be changed to be the network name of your camera. Instead of using the CameraDropdown, we will connect to this camera as soon as it's discovered on the network.

2. Custom overlay graphics are accomplished by creating an object that implements the MiDrawObj interface. This can be a separate class module, but can also be a form, so we'll just use our main form for this. Notice the Implements MiDrawObj statement towards the top of the code. As a result of this statement, you will be required to fill in a MiDrawObj_Draw subroutine. We'll get to that soon.

3. This project uses a MiCoordinator as well as a MiImageConnection. These are declared at the top, allocated in Form_Load, and released in Form_Unload.

4. We also need a place to temporally stash a MiCycleReport, so we declare a variable of that type called saveReport. It will be set in the OnNewRecord event handler. It needs to be released in Form_Unload.

5. For this example, we will need to define a font, so we declare a new StdFont called myFont. In Form_Load, we set the font's Name and Size.

6. The MiCoordinator has a list of devices on the network. Although the device we're interested in may already be in the list, it also may not yet be discovered. To avoid unnecessary complication, we start off by clearing this list in Form_Load by calling the ClearDevices method. This ensures that, as each device is discovered, we will get an OnDeviceDiscovered event. In the OnDeviceDiscovered event handler, we check to see if the name of the device just discovered matches the one we're looking for and, if so, calls

DeviceFocusSet with the device name. This, in turn, will cause our ImageConnection to connect.

7. In the OnNewRecord event handler, we call the MiImageView1.NewRecord method to show the image as normal. We will also stash away a reference to the MiCycleReport by setting our saveReport variable to it.

8. To make the MiImageView control aware that we want to overlay our own graphics, notice the line MiImageView1.DrawObjAdd Me, “overlay” in Form_Load. This instructs the MiImageView control that we are adding a custom drawing object, which happens to be this form (using the VB keyword ”Me”). The second parameter is a symbolic name given to the drawing object, which you will need if you want to be able to remove it later. You can call it anything you want, so here we’re calling it “overlay”.

9. The Implements MiDrawObj statement at the beginning means we are now required to implement the MiDrawObj_Draw method. This method will get called every time the MiImageDisplay is updated. The parameter supplied to the method gives us a very useful object of type MiDC. This object provides methods for drawing lines, rectangles, ellipses, and text. You can set pen styles, fonts, etc. And, most importantly, it scales all graphics for you so you just draw in image coordinates. In this example, a crosshair is drawn and text string is displayed with the decode string and some counters.

Example 6 — Custom ROI

This example illustrates how to set and change the ROI. Before you run the example, you need to make sure that the HawkEye™ 1500 is in Continuous Trigger mode (i.e., TRIG C) and decoding a Data Matrix. Here are the descriptions of this example:

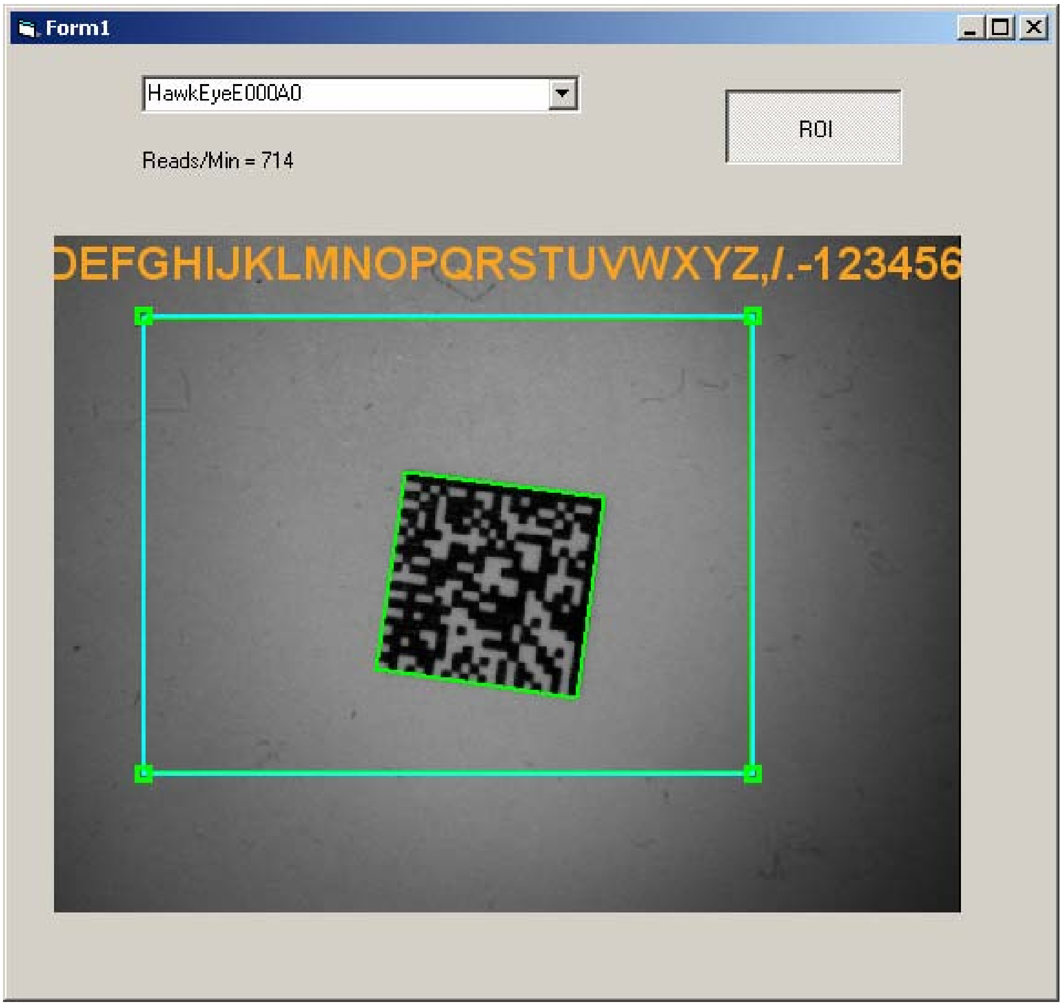
- Establish a connection to a HawkEye™ 1500.
- Display images.
- Display number of reads per minute.
- Use a checkbox control to enable or disable the custom ROI box display in the image.
- When the custom ROI is displayed in the image, grab the corner or handle of the ROI and resize it. You can also drag and reposition the ROI by grabbing it anywhere inside the ROI box.

In the sample code:

1. Include the class module SearchArea that implements the MiDrawHandles interface to make custom graphics “dragable” and the MiDrawObj interface to draw custom graphics in MiImageView.
2. Select the CameraDropdown component and place it across the top of the form.
3. Add a label on the form and use it to display the number of reads per minute using the ReadsPerMin property of MiCycleReport.
4. Select a Checkbox control and place it on the form. Implement the Click event such that when the checkbox is checked, the custom ROI is displayed and can be resized and dragable.
5. Select the Imageview component and position it on the form.

Figure 2–1 shows a dragable ROI with four green handles that is displayed when the ROI checkbox is checked.

FIGURE 2–1. **Draggable ROI with Four Green Handles**



Reads/Min = 714

Example 7 — Verification Report

This example illustrates how to display the verification report. Before you run the example, you need to make sure that the HawkEye™ 1500 is in Continuous Trigger mode (i.e., TRIG C) reading a Data Matrix and the AIM (ISO 16022) verification is enabled. Here are the descriptions of this example:

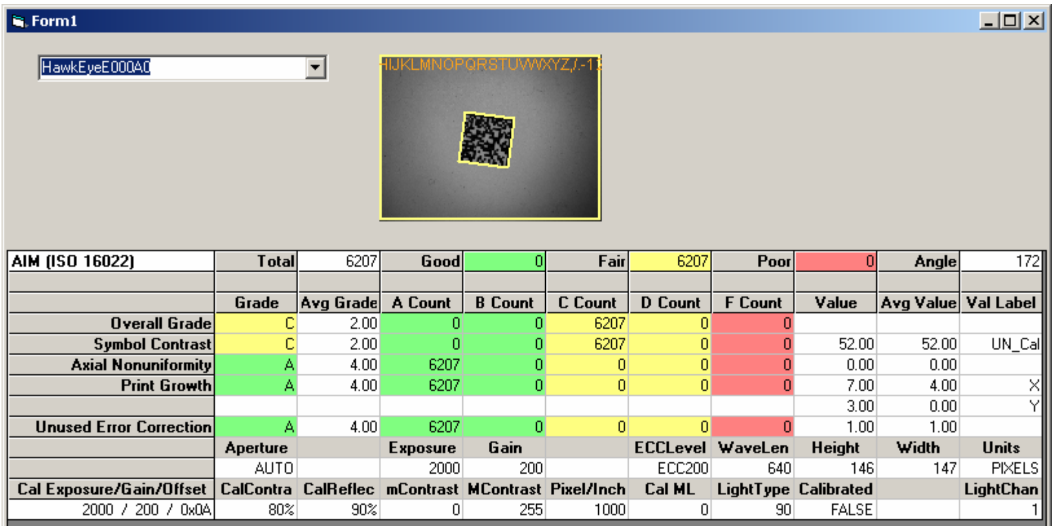
- Establish a connection to a HawkEye™ 1500.
- Display images.
- Display Verification Report.

In the sample code:

1. Select the CameraDropdown component and place it across the top of the form.
2. Select the Imageview component and position it on the form.
3. Select the VerifyReportGrid component and position it on the form.
4. In the OnNewRecord event of the ReportConnection object, invoke the NewRecord method of the VerifyReportGrid object to display the verification report.

Figure 2–2 shows the verification report being displayed.

FIGURE 2–2. **Verification Report**



Form1

HawkEyeE000A0

HIJKLMNOPQRSTUVWXYZ,/.-1

AIM (ISO 16022)	Total	6207	Good	0	Fair	6207	Poor	0	Angle	172
	Grade	Avg Grade	A Count	B Count	C Count	D Count	F Count	Value	Avg Value	Val Label
Overall Grade	C	2.00	0	0	6207	0	0			
Symbol Contrast	C	2.00	0	0	6207	0	0	52.00	52.00	UN_Cal
Axial Nonuniformity	A	4.00	6207	0	0	0	0	0.00	0.00	
Print Growth	A	4.00	6207	0	0	0	0	7.00	4.00	X
								3.00	0.00	Y
Unused Error Correction	A	4.00	6207	0	0	0	0	1.00	1.00	
	Aperture		Exposure	Gain		ECCLevel	WaveLen	Height	Width	Units
	AUTO		2000	200		ECC200	640	146	147	PIXELS
Cal Exposure/Gain/Offset	CalContra	CalReflec	mContrast	MContrast	Pixel/Inch	Cal ML	LightType	Calibrated		LightChan
2000 / 200 / 0x0A	80%	90%	0	255	1000	0	90	FALSE		1

Example 8 — Verification Data

This example illustrates how to display the verification data such as verification status, counters for total, good, fair, and poor. Before you run the example, you need to make sure that the HawkEye™ 1500 is in Continuous Trigger mode (i.e., TRIG C) reading a Data Matrix and the AIM (ISO 16022) verification is enabled. Here are the descriptions of this example:

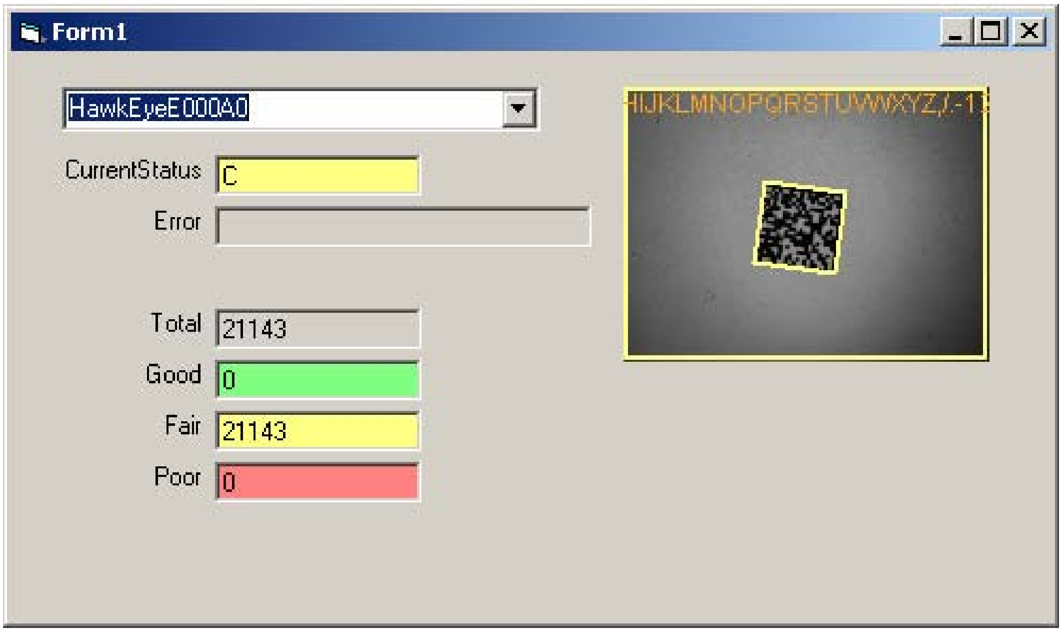
- Establish a connection to a HawkEye™ 1500.
- Display images.
- Display verification data.

In the sample code:

1. Select the CameraDropdown component and place it across the top of the form.
2. Select the Imageview component and position it on the form.
3. Add labels and text boxes for displaying Current Status, Error message, Total Count, Good Count, Fair Count, and Poor Count.
4. In the OnNewRecord event of the ReportConnection object, get the verification data from the VerifyReport property of the MiCycleReport object.

Figure 2–3 shows the verification data being displayed.

FIGURE 2–3. **Verification Data**



	HawkEyeE000A0
CurrentStatus	C
Error	
Total	21143
Good	0
Fair	21143
Poor	0

Example 9 — TCP Socket Communication

This example illustrates how to communicate with the HawkEye™ 1500 from a custom VB program. The VB program uses the Winsock control running as a TCP client communicating with the camera as a TCP server. The command port 49095 allows commands to be issued to the camera and replies to be seen. Note that the reply window prepends a timestamp on the message and that multiline replies zip by quickly. Ports 49098, 49099, 49100, and 49101 map to TCP1 through TCP4, respectively, from the “Output Settings...” dialog in ReadRunner.

Before you run the example, you need to make sure that the HawkEye™ 1500 is in Continuous Trigger mode (i.e., TRIG C) reading a Data Matrix. Here are the descriptions of this example:

- In Remote Host Information frame, specify the IP Address of the HawkEye™ 1500 camera.
- Set Port Number to be 40495 (command port).
- Click the button Connect To Server and ensure Connected To Server status is shown.
- Type the command VER in the Data to Send to Server box, then press Send. Observe the reply from the camera consisting of Time Stamp and version numbers shown below the TimeStamp: + Data Received from Server box.
- Click the button Disconnect from Server, then change the Port Number to 49098.
- Click the button Connect To Server and ensure Connected To Server status is shown. Observe the decoded data sent from the camera shown below the TimeStamp: + Data Received from Server box.

In the sample code:

1. Select the Winsock control and place it on the form.
2. Create the Remote Host Information frame add two text boxes for remote host name or IP Address and remote port number. Add a label to show connection status. Add a button for Connect To Server.
3. Add a text box for typing in the command (Data to Send to Server) and a corresponding button Send.
4. Add a text box for displaying the data received from the camera.

5. In the Click event of the Connect To Server button, call the Connect subroutine which uses the Connect method of the Winsock control to connect the remote server using the host name and port number entered by the user.

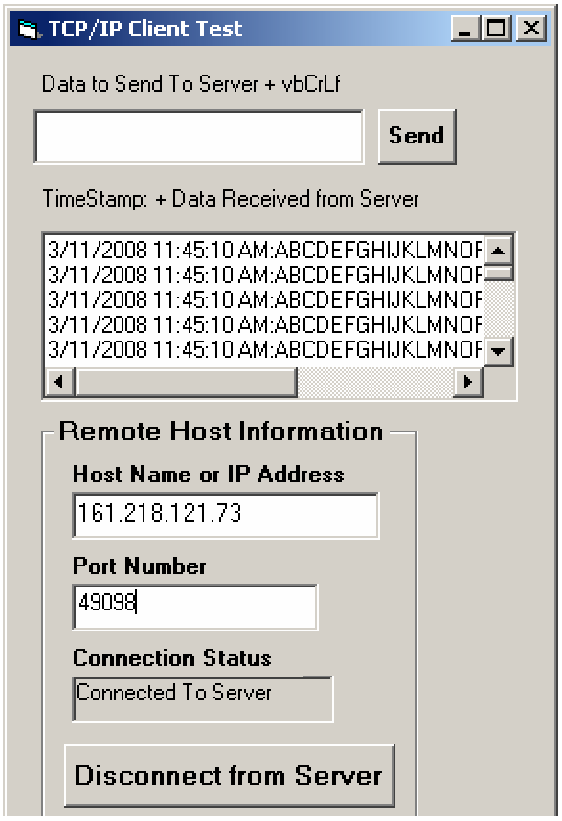
6. Use the Winsock Connect event to display the Connection Status, which is “Connected to Server”.

7. Use the Winsock SendData method to send the command.

8. Use the Winsock DataArrival event and GetData method to receive the data from the camera.

Figure 2–4 shows the communication between the TCP/IP Client and the camera.

FIGURE 2–4. **Communication Between TCP/IP Client & the Camera**



Data to Send To Server + vbCrLf

TimeStamp: + Data Received from Server

```
3/11/2008 11:45:10 AM:ABCDEFGHIJKLMNOF
3/11/2008 11:45:10 AM:ABCDEFGHIJKLMNOF
3/11/2008 11:45:10 AM:ABCDEFGHIJKLMNOF
3/11/2008 11:45:10 AM:ABCDEFGHIJKLMNOF
3/11/2008 11:45:10 AM:ABCDEFGHIJKLMNOF
```

Remote Host Information

Host Name or IP Address: 161.218.121.73

Port Number: 49098

Connection Status: Connected To Server

Microscan HawkEye™ Reader Object Type Library

MiObj.DLL - C++ low-level connection library.

MiCoordinator

Co-ordinates objects within a single process.

Methods

- Sub DeviceFocusSet(ByVal bszName As String, Optional ByVal bszUser As String=, Optional ByVal bszPassword As String=) — DeviceFocusSet and connect for control if user and pwd given.
- Function DeviceFocusGet() As IMiDevice — Returns the device object that currently has the focus.
- Property Get Devices() As IMiDeviceCollection — Returns the set of known devices.
- Sub ClearDevices — Clears the device list.
- Sub BroadcastMessage(bstrSender As String, bstrMsg As String, bstrParam As String) — Broadcast a message to other users of MiCoordinator.

Events

- OnDeviceFocus(ByVal objDevice As IMiDevice) — Sent when a named device has been given the focus.
- OnDeviceDiscovered(ByVal newDevice As IMiDevice) — Sent when a new device has been discovered.
- OnDeviceStatusChanged(ByVal device As IMiDevice, ByVal newStatus As EOnlineStatus) — Fired when a network device status changes.
- OnDeviceListCleared() — Fired when the device list has been cleared.
- OnDeviceListCleared() — Fired when the device list has been cleared using the ClearDevices method.
- OnBroadcastMessage(bstrSender As String, bstrMsg As String, bstrParam As String) — Fired when a MiCoordinator user calls BroadcastMessage.

MiDevice

Represents a camera device.

Methods

- Property Get Name() As String — Name of the device.
- Property Let Name(RHS As String) — Name of the device.
- Property Get NetworkName() As String — NetworkName of the device.
- Property Let NetworkName(RHS As String) — NetworkName of the device.
- Property Get ASCIIConnection() As IMiASCIIConnection — Returns/sets embedded ASCIIConnection object.
- Property Let ASCIIConnection(RHS As IMiASCIIConnection) — Returns/sets embedded ASCIIConnection object.
- Property Get OnlineStatus() As EOnlineStatus — Returns the current status of the device.
- Property Let OnlineStatus(RHS As EOnlineStatus) — Returns the current status of the device.
- Property Get IPAddress() As String — Sets/returns the IP address of the device.
- Property Let IPAddress(RHS As String) — Sets/returns the IP address of the device.
- Property Get TimeStamp() As Long — Determines device inactivity.
- Property Let TimeStamp(RHS As Long) — determines device inactivity.
- Sub RefreshInfo — Queries device for current information. Updates accordingly.
- Sub ConnectASCII(ByVal bszUserName As String, ByVal bszPassword As String) — Connects an ASCII connection to the device.
- Sub DisconnectAll — Disconnects all connections to the device.

- Sub ConnectReports(ByVal bszUserName As String, ByVal bszPassword As String) — Connects to the device for reports, establishes the ReportConnection object.
- Sub ConnectImages(ByVal bszUserName As String, ByVal bszPassword As String) — Connects to the device for images and decode data (lossy), establishes the ImageConnection object.
- Property Get ReportConnection() As IMiReportConnection — Returns/sets the report connection for the device.
- Property Let ReportConnection(RHS As IMiReportConnection) — Returns/sets the report connection for the device.
- Property Get ImageConnection() As IMiImageConnection — Returns/sets the ImageConnection for this device.
- Property Let ImageConnection(RHS As IMiImageConnection) — Returns/sets the ImageConnection for this device.
- Property Get CountCycle() As Long — Returns the cycle count from a RefreshInfo call.
- Property Get CountPassed() As Long — Returns the Passed count from a RefreshInfo call.
- Property Get CountAlarms() As Long — Returns the Alarm count from a RefreshInfo call.
- Property Get NumNetworkConnections() As Long — Returns number of active network connections to the device.
- Property Get NameOfController() As String — Returns the name or IP address of the network connection controlling the device.
- Property Get HaveControl() As Boolean — Returns True if this computer has control of the device.
- Property Get DirtyBits() As Long — Returns state of the dirty bits for the device.
- Sub FilesSend(ByVal whichDrv As ERamDrive, ByVal fileOrList As Variant) — Send specific set of files to a particular RAM drive on the camera.

- Property Get SoftwareVersion() As String — Returns version of the software on the device.

Events

- OnConnected(ByVal connObj As IMiConnection) — Sent when a specific connection object connects.
- OnDisconnected(ByVal connObj As IMiConnection) — Sent when a specific connection object disconnects.
- OnOnlineStatusChanged(ByVal newStatus As EOnlineStatus) — Sent when online/offline status changes.
- Function OnControlStatusChanged(ByVal newStatus As EControlStatus) — Sent when taking or releasing control of device.

MiASCIIConnection

A device connection that sends and receives ASCII data. This is the only type of connection supported for serial ports.

Methods

- Sub Disconnect — Disconnects the connection.
- Sub Connect(ByVal bszName As String) — Connects to the given device.
- Property Get Connected() As Boolean — Returns True if connected.
- Sub Reconnect — Reconnects the device.
- Sub CommandSend(ByVal bszCmd As String, Optional ByVal lTimeout As Long=10000) — Sends a command over the connection, waits up to lTimeout msecs for result.
- Property Get Name() As String — Name of the connection (read-only).
- Sub ImageUploadStart(Optional ByVal nCompression As Long=0, Optional ByVal useTriggers As Boolean=False, Optional ByVal doDecode As Boolean=True, Optional ByVal useAutoPhoto As Boolean=True, Optional ByVal lGain As Long=250, Optional ByVal lExposure As Long=10500) — Starts an image upload using the given settings.

- Sub ImageUploadAbort — Aborts an image upload already in progress.
- Function LineRead(Optional ByVal nTimeout As Long=1000) As String — Reads a line from the connection - blocking until received.
- Property Get LineQueue() As IMiLineQueue — Property LineQueue.

MiReportConnection

A non-lossy device connection which sends CycleReports w/o images.

Methods

- Sub Disconnect — Disconnects the connection.
- Sub Connect(ByVal bszName As String) — Connects to the given device.
- Property Get Connected() As Boolean — Returns True if connected.
- Sub Reconnect — Reconnects the device.

MiImageConnection

A lossy device connection which sends CycleReports containing images.

Methods

- Sub Disconnect — Disconnects the connection.
- Sub Connect(ByVal bszName As String) — Connects to the given device.
- Property Get Connected() As Boolean — Returns True if connected.
- Sub Reconnect — Reconnects the device.
- Property Get FreezeMode() As EFreezeMode — Returns the current freeze mode.
- Property Get FailFlags() As EFailFlags — Returns the current fail flags that qualify Failed and Freeze-Next-Failed modes.
- Sub FreezeModeSet(ByVal newMode As EFreezeMode, Optional ByVal newFlags As ENewFlags=-1) — Sets the freeze mode for the connection.

- Property Get MaxRate() As Long — Returns/sets the maximum images per second transferred by the camera on this connection.
- Property Let MaxRate(RHS As Long) — Returns/sets the maximum images per second transferred by the camera on this connection.
- Sub LiveStart — Starts live video on the camera, associated ASCIIConnection must be in control.
- Sub LiveStop — Stops live video on the camera.
- Property Get LiveEnabled() As Boolean — Returns True if live mode is enabled.

Connection Events

All connection types fire the following events.

Events

- OnNewLine(ByVal bszLine As String) — Fired when a new line is received.
- OnNewRecord(ByVal objReport As IMiCycleReport) — Fired when a new data record is received.
- OnDataTransferStarted() — Event fired when a new data record upload has been started.
- OnDataTransferRetry() — Event fired when a block read failed and is being retried.
- OnDataTransferAborted() — Event fired when an record transfer upload is aborted.
- OnDataTransferPercentComplete(ByVal nPercent As Integer) — Event fired to indicate percent complete on the data transfer.
- OnConnected() — Event fired when connected.
- OnDisconnected() — Event fired when disconnected.
- OnSentLine(ByVal bszLine As String) — Fired when a line is sent through the connection.

- OnSendingLine(ByVal bszCommand As String) — Fires this event before sending a line.
- OnSentCommand(ByVal bszCommand As String) — Fired after a command has been sent and processed.
- OnPartQueueReceived(ByVal objColl As ImiCycleReportCollection) — Fired when a new part queue collection is received.
- OnConnectionDropped(ByVal errCode , ByVal bszDescription As String) — Fired when a connection is dropped ungracefully.

Microscan HawkEye™ Reader ActiveX Type Library

MiObjUI.DLL - C++ low-level GUI library

MiTimeplot

Plots multiple variables which are linked in time.

Methods

- Function AddTrace(ByVal bstrName As String, ByVal color As ULong, Optional ByVal bDigital As Long=0) As Long — Adds a trace.
- Sub StartNewData — Start of a data set event.
- Sub EndNewData — End of a data set event.
- Sub NewData(ByVal idTrace As Long, ByVal value As Double) — Record new data for specified trace.
- Sub Refresh — Redraw control.
- Function AddYAxis(ByVal bstrName As String, ByVal color As ULong, ByVal val As Double, Optional ByVal style As Long=2) As Long — Adds a Y axis.
- Property Let BackColor(RHS As ULong) — Sets the background color of the control.
- Sub SetYRange(ByVal minPlot As Double, ByVal maxPlot As Double) — Sets the y range for the overall plot.

- Sub SetXStep(ByVal xStep As Long) — Sets the number of pixels between plotted points.
- Property Let LegendColor(RHS As ULong) — Background color of the legend area.
- Property Let LegendTextColor(RHS As ULong) — Text color of the legend area.
- Sub NewEvent(ByVal color As ULong, ByVal style As Long, ByVal info As String) — Add a event which appears as a vertical line in time.
- Sub NewState(ByVal idTrace As Long, ByVal bState As Long) — Add a digital event state.
- Sub SetBias(ByVal idTrace As Long, ByVal bias As Double) — Set trace bias.
- Sub SetScale(ByVal idTrace As Long, ByVal Scale As Double) — Set trace scale.
- Function AddLabel(ByVal x As Double, ByVal y As Double, ByVal caption As String, ByVal fontsize As Long, ByVal TextColor As ULong, ByVal BackColor As ULong, Optional ByVal bBold As Long=0, Optional ByVal bOpaque As Long=1) As Long — Add a text label.
- Sub SetItemVisible(ByVal id As Long, ByVal bVisible As Boolean) — Set visibility of specified item.
- Function IsItemVisible(ByVal id As Long) As Boolean — Check visibility of specified item.

MiImageView

Display images associated with CycleReports with optional Filmstrip capability an user defined drawing objects.

Methods

- Sub NewRecord(ByVal objCycleReport As IMiCycleReport) — Display image associated with objCycleReport. If in filmstrip mode, add it to stored records.
- Sub ClearRecords — Clear all records - which also clears the display.

- Sub Refresh — Refresh control graphics.
- Property Get FilmstripMode() As Boolean — Get filmstrip mode.
- Property Let FilmstripMode(RHS As Boolean) — Set filmstrip mode.
- Property Get ShowOverlay() As Boolean — Get overlay graphic visibility.
- Property Let ShowOverlay(RHS As Boolean) — Set overlay graphics visibility.
- Sub DrawObjAdd(ByVal objDraw As IMiDrawObj, Optional ByVal key As String=) — Add a user defined drawing object.
- Sub DrawObjRemove(ByVal key As String) — Remove a user defined drawing object.
- Sub DrawObjClear — Remove all user defined drawing objects.
- Property Get ShowCycleGraphics() As Boolean — Get visibility of cycle graphics.
- Property Let ShowCycleGraphics(RHS As Boolean) — Set visibility of cycle graphics.
- Function CycleReportFromMousePos(ByVal xPos As Integer, ByVal yPos As Integer) As IMiCycleReport — For filmstrip mode, given a mouse position returns the corresponding CycleReport.
- Sub SetFilmstripRecords(ByVal objCollection As IMiCycleReportCollection) — Sets all filmstrip records at once given a collection of CycleReports.

Events

- OnMouseMove(ByVal xPos As Integer, ByVal yPos As Integer, ByVal flags As Integer) — Event fired on Mouse Move.
- OnLButtonDown(ByVal xPos As Integer, ByVal yPos As Integer, ByVal flags As Integer) — Event fired on Mouse Left Button Down.
- OnLButtonUp(ByVal xPos As Integer, ByVal yPos As Integer, ByVal flags As Integer) — Event fired on Mouse Left Button Up.

- OnLButtonDblClk(ByVal xPos As Integer, ByVal yPos As Integer, ByVal flags As Integer) — Event fired on Mouse Left Double Click.
- OnRButtonDown(ByVal xPos As Integer, ByVal yPos As Integer, ByVal flags As Integer) — Event fired on Mouse Right Button Down.
- OnRButtonUp(ByVal xPos As Integer, ByVal yPos As Integer, ByVal flags As Integer) — Event fired on Mouse Right Button Up.

MiDrawObj

Implement this interface to draw custom graphics in MiImageView.

Methods

Sub Draw(ByVal midc As IMiDC)

MiDC

A graphics “Device Context” for drawing within user defined drawing objects.

Methods

- Property Get hDC() As Long
- Property Let hDC(RHS As Long)
- Sub MoveTo(ByVal x As Double, ByVal y As Double)
- Sub LineTo(ByVal x As Double, ByVal y As Double)
- Sub SetPen(ByVal style As Integer, ByVal width As Integer, ByVal color As ULong)
- Sub SetBkMode(ByVal bkMode As Integer)
- Sub SetBkColor(ByVal color As ULong)
- Sub SetTextColor(ByVal color As ULong)
- Sub Rectangle(ByVal left As Double, ByVal top As Double, ByVal right As Double, ByVal bottom As Double)
- Property Get Scale() As Double

- Property Let Scale(RHS As Double)
- Property Get XOffset() As Double
- Property Let XOffset(RHS As Double)
- Property Get YOffset() As Double
- Property Let YOffset(RHS As Double)
- Sub ScreenToPixel(ByVal x As Double, ByVal y As Double)
- Sub PixelToScreen(ByVal x As Double, ByVal y As Double)
- Sub DrawHandle(ByVal x As Double, ByVal y As Double, ByVal color As ULong, Optional ByVal flags As Long=0)
- Sub SetSolidBrush(ByVal color As ULong)
- Sub SetNullBrush()
- Sub Ellipse(leftRect As Double, topRect As Double, rightRect As Double, bottomRect As Double)
- Sub SetFont(pFont As IFontDisp)
- Sub TextOut(x As Double, y As Double, bstrText As String)

Pen Style Constants for SetPen

- Const PS_SOLID = 0
- Const PS_DASH = 1
- Const PS_DOT = 2
- Const PS_DASHDOT = 3
- Const PS_DASHDOTDOT = 4
- Const PS_NULL = 5

MiDrawHandles

Implement this interface to make custom graphics “dragable.”

Methods

- Sub GetHandle(ByVal index As Long, ByVal x As Double, ByVal y As Double, ByVal flags As Long)
- Sub MoveHandleTo(ByVal index As Long, ByVal x As Double, ByVal y As Double)
- Sub MoveRelative(ByVal x As Double, ByVal y As Double)
- Property Get NumHandles() As Long
- Sub DrawHandle(ByVal midcIn As IMiDC, ByVal index As Long)
- Function PointIsInside(ByVal x As Double, ByVal y As Double) As Boolean

CHAPTER 3

C++ Samples for TCP/IP Socket Communication

Two C++ console application projects are provided to illustrate how to get data and images from the HawkEye™ 1500 camera using TCP/IP socket programming. These projects can be opened and built using Microsoft Visual C++ 6.0.

Example 1 – PartQ Data Retrieval

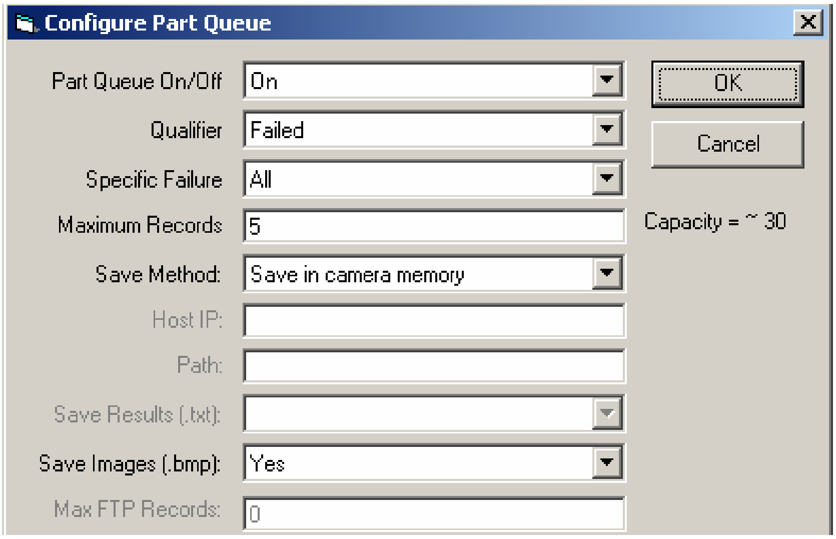
This console application illustrates how to retrieve PartQ data and save them to files. The program connects to the command port (49095), issues the PARTQREQFORMAT command and dumps the output data to a series of files in the working directory. For each PartQ record, two files are saved (note that xxxx represents a 4-digit record number):

- record_xxxx.txt, containing data such as cycle number, overall pass/fail, located/decoded/matched flags, detailed failure code, decode string.
- record_xxxx.bmp, for formatted bitmap image.

To run this application:

1. Configure the PartQ in ReadRunner as shown in Figure 3–1:

FIGURE 3–1. **Configure PartQ**



Part Queue On/Off	On
Qualifier	Failed
Specific Failure	All
Maximum Records	5
Save Method:	Save in camera memory
Host IP:	
Path:	
Save Results (.txt):	
Save Images (.bmp):	Yes
Max FTP Records:	0

Capacity = ~ 30

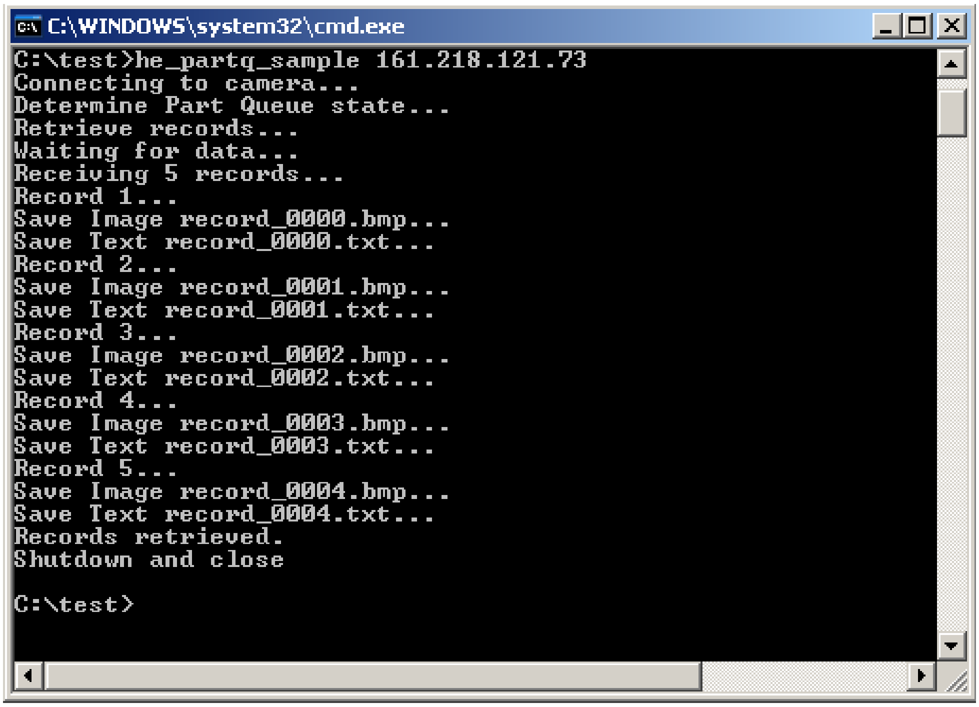
2. Configure the reader in Continuous Trigger mode (i.e., TRIG C) and block the Data Matrix code to create failed reads to be recorded in the PartQ.

3. Run the console application from a Windows command window by typing:

   he_partq_sample IP_address

   where IP_address is the camera's IP address.

FIGURE 3–2.



```
C:\test>he_partq_sample 161.218.121.73
Connecting to camera...
Determine Part Queue state...
Retrieve records...
Waiting for data...
Receiving 5 records...
Record 1...
Save Image record_0000.bmp...
Save Text record_0000.txt...
Record 2...
Save Image record_0001.bmp...
Save Text record_0001.txt...
Record 3...
Save Image record_0002.bmp...
Save Text record_0002.txt...
Record 4...
Save Image record_0003.bmp...
Save Text record_0003.txt...
Record 5...
Save Image record_0004.bmp...
Save Text record_0004.txt...
Records retrieved.
Shutdown and close

C:\test>
```

Example 2 – Output Socket Data Retrieval

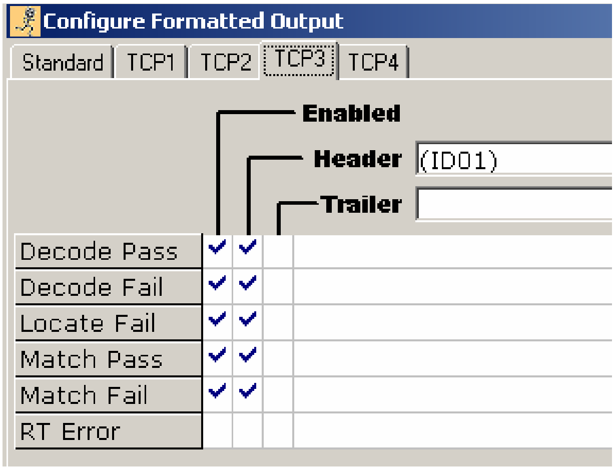
This console application illustrates how to get the image from each read trigger. The program connects to one of the following two ports specified in the command line, receives the image data, and saves it to an image file:

- Port 49099 — Image is saved in raw format using the default formatted output header (CD25) for TCP2 in ReadRunner. The file name is image_xxxx.dat, where xxxx is a 4-digit index.
- Port 49100 — Image is saved in BMP format using the default formatted output header (ID01) for TCP3 in ReadRunner. The file name is image_xxxx.bmp, where xxxx is a 4-digit index.

To run this application using Port 49100,

1. Configure Formatted Output in ReadRunner as shown in Figure 3–3:

FIGURE 3–3. **Configure Formatted Output**


Configure Formatted Output

Standard | TCP1 | TCP2 | TCP3 | TCP4

Header (ID01)

Trailer

	Enabled	Header	Trailer
Decode Pass	✓	✓	
Decode Fail	✓	✓	
Locate Fail	✓	✓	
Match Pass	✓	✓	
Match Fail	✓	✓	
RT Error			

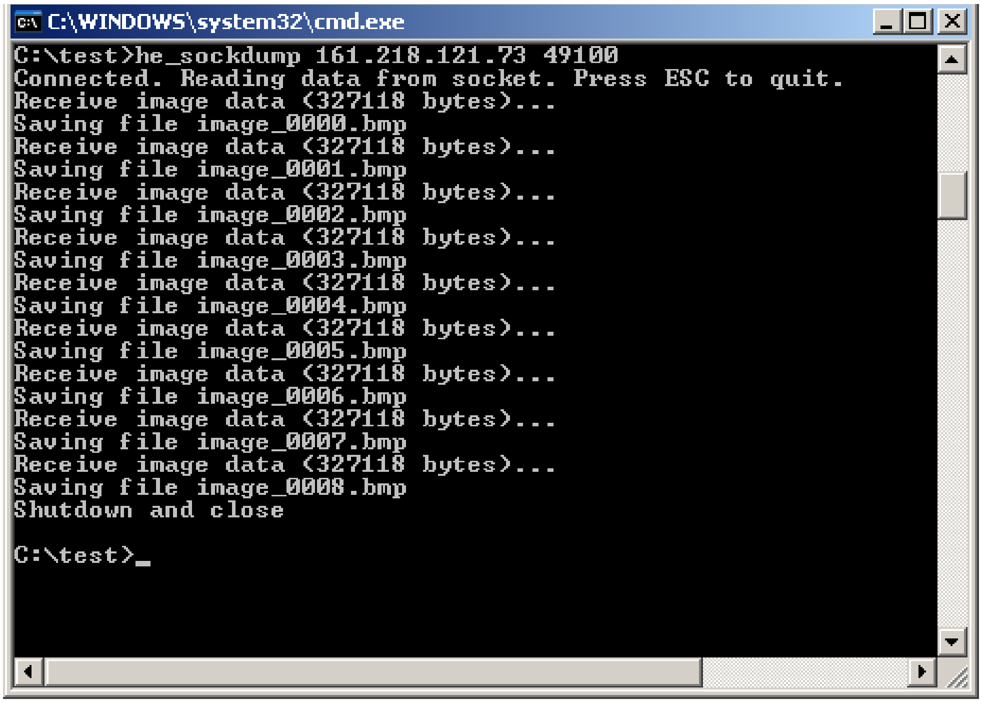
2. Configure the reader in Continuous Trigger mode (i.e., TRIG C).

3. Run the console application from a Windows command window by typing:

   he_sockdump IP_address 49100

   where IP_address is the camera’s IP address.

FIGURE 3–4.



C:\WINDOWS\system32\cmd.exe

```
C:\test>he_sockdump 161.218.121.73 49100
Connected. Reading data from socket. Press ESC to quit.
Receive image data (327118 bytes)...
Saving file image_0000.bmp
Receive image data (327118 bytes)...
Saving file image_0001.bmp
Receive image data (327118 bytes)...
Saving file image_0002.bmp
Receive image data (327118 bytes)...
Saving file image_0003.bmp
Receive image data (327118 bytes)...
Saving file image_0004.bmp
Receive image data (327118 bytes)...
Saving file image_0005.bmp
Receive image data (327118 bytes)...
Saving file image_0006.bmp
Receive image data (327118 bytes)...
Saving file image_0007.bmp
Receive image data (327118 bytes)...
Saving file image_0008.bmp
Shutdown and close

C:\test>_
```

Index

Numerics

1-D Symbologies
  selecting 1-16

A

AIM DPM-1-2006 Verification
  enabling 1-195
AIM Verification
  turning on or off 1-192
Aperture Size
  setting synthesized 1-12
APIPA
  setting 1-52
Application 2-4
AppSetting 2-4
Assigning
  variables to digital output lines 1-76
Audible Feedback
  optical alignment 1-139
Auto Settings
  set 1-128

B

Barcode
  confidence level 1-18
  search method 1-23
  threshold
    setting 1-25
Barcodes
  setting
    height 1-19
    horizontal probe spacing 1-20
    length 1-21
    number to be decoded 1-22
    number to decode 1-104
    width 1-27
  vertical probe spacing 1-26
BC_DECODE_NEAR_CENTER
    Command 1-67
BC412
  decoding
    backward 1-32
    forward 1-32
  polarity
    setting 1-33
  start checking 1-34
  stop checking 1-34
BC412 check sum checking 1-30
Binarizing
  threshold method 1-175
Bounce Latch Time
  setting 1-171

C

Calibration Status String 1-37
Calibration Target
  contrast value
    setting 1-169
  reflectance value
    setting 1-170
Camera
  current time 1-176
  displaying
    current name 1-79
    ip address 1-79
    subnet mask 1-79

light calibration
reporting 1-36
releasing control 1-144
resetting 1-148
selecting lighting 1-72
setting
offline 1-106
online 1-111
take control 1-48
Cancel
learn request 1-83
pending reportreq command 1-146
Cell
sampling 1-40
size 1-41
Cell Unit
multiplier 1-42
report type 1-43
setting 1-47
Centering
image 1-65
Check Sum Checking 1-31
Columns to Expect
data matrix 1-44
Command
prompt
controlling 1-137
report state of command just changed 1-73
Configuration
saving current 1-158
Configuring
criteria for variable testing 1-160
Console
vxworks 1-45
Control
releasing camera 1-144
setting match 1-89
Controlling
1-d symbologies 1-15
command prompt 1-137
Current
ip address
displaying for camera 1-79
name
displaying for camera 1-79
time
camera 1-176
Customer Settings Installed in Manufacturing 1-157
Customizing
locator failure output 1-87
overrun message output 1-155
successful decode output 1-107
successful match output 1-109
unsuccessful match output 1-97
Cycle Records
retrieving 1-124

D
Data
echoing typed 1-174
Data Matrix
maximum number to find 1-105
number
columns to expect 1-44
rows to expect 1-154
selecting image style 1-167
setting
expected polarity 1-132
expected width 1-202
height 1-70
Decode Output
customizing unsuccessful 1-49
header
setting 1-69
Decode Results Beeping
enabling 1-35
Decoder
fine tuning options 1-66
learn values 1-84
parameters
assigning to a trigger 1-13, 1-14
set to read variety of symbols 1-188
setting source of images 1-39
settings
load from pid 1-86
train 1-130
Decoding
limiting time 1-177
Defining
region of interest 1-153

Delete
  stored pid 1-50
DHCP
  disabling 1-52
  enable 1-51
  enabling 1-52
Dimension Range
  specifying 1-203
Direction
  set probe 1-134
  setting probe 1-134
Displaying
  current settings 1-163
  info
    about remote commands 1-71
  results of learn calibrate command 1-46
  summary of part queue 1-127
DM_ALLOW_SEVERE_DAMAGE Command 1-67
DM_ALLOW_STEEP_ANGLE Command 1-66
DM_CELL_OUTLINE Command 1-67, 1-68
DM_DECODE_NEAR_CENTER Command 1-67
DM_ENSURE_ROI Command 1-67
DM_FINETUNE_RESERVED Command 1-66
DM_IGNORE_SINGLE_EDGE Command 1-66
DPM Verification Parameters
  selecting 1-189
Dwell Time
  setting 1-62

E
Edge Direction
  setting 1-173
Enabling
  aim dpm-1-2006 verification 1-195
  bc412 check sum checking 1-30
  bc412 start checking 1-34
  check sum checking 1-31
  conversion to negative image 1-102
  decode results beeping 1-35
  dhcp 1-51
  ethernet ip 1-64
  output of upc as ean 1-29
  part queue 1-115
Error Correction Level
  setting 1-63
Ethernet IP
  enabling 1-64
Example
  a simple monitoring application 2-5
  connecting to two cameras at once 2-12
  controlling the hawkeye 1500 2-15
  handling images and reports 2-8
  image overlay graphics and auto connection 2-20

F
Factory
  default settings
    loading 1-86
  set commands
    running 1-156
Fine Tuning Options
  decoder 1-66

G
Generating
  periodic virtual triggers 1-159
Get
  maximum record count 1-116
GPIO Output Mode
  selecting 1-75

H
Hardware
  lines
    altering dynamics 1-183
  settings installed in manufacturing 1-157
Horizontal Probe Spacing 1-20

I
Image
  centering 1-65
  enable conversion to negative 1-102
ImageConnection 2-3
Index
  displaying stored value in pid 1-198
Initial Match Count
  setting 1-94

Inspection
- limiting time 1-177

INTENSITY_ENHANCE_ENABLED Command 1-66

Interrupt Detection Delay
- setting 1-172

IO Lines
- setting polarity of 1-164

J

Job
- loading info 1-86

K

Keyword Output Conditions 1-11

L

Laser
- turning on or off target 1-168

Learn
- assist 1-82
- decoder values 1-84
- photometry values 1-84

LEARN CALIBRATE
- displaying results 1-46

Learn Request
- canceling 1-83

Learn Sequence
- starting 1-80

License
- is option’s license installed 1-143

License Key
- enter manually 1-162

Light Source Wavelength
- setting 1-201

Lighting
- selecting camera 1-72
- type
  - selecting 1-85

Limiting Time
- decoding 1-177
- inspecting 1-177

Load
- info from a job 1-86
- info from a PID 1-86

Locator Failure Output
- customizing 1-87

M

MAC Address
- returning 1-88

Manual Settings
- set 1-128

Manufacturing Data 1-157

Match
- control
  - set to character matching 1-100
  - setting 1-89
- list
  - use in Match Mode? 1-93
- parameters
  - set 1-95
- string
  - entering into match_list table 1-91

MATCH_LIST Table
- entering match strings 1-91

Matrix Orientation 1-112

Maximum
- dimension range
  - specifying 1-203

Maximum Record Count 1-116

Mean Light Value
- reporting 1-38

MiASCIIConnection 2-3

MiCoordinator 2-2

MiCycleReport 2-4

MiImageConnection 2-3

MiImageView 2-4

Minimum
- dimension range
  - specifying 1-203

MIOBJ.DLL 2-1

MIOBJUI.DLL 2-2

MiReportConnection 2-3

Mode
- set triggering 1-181
- setting
  - retry 1-149

N

NO_QUIET_ZONE_CLUTTER Command 1-66

Number
- columns
  - data matrix 1-44

O
Obtain
  maximum record count 1-116
Optical Alignment
  audible feedback 1-139
  visual feedback 1-139
Option’s License
  is it installed? 1-143
Overrun Message Output
  customizing 1-155

P
Packets
  controlling broadcast of udp 1-187
Parameters
  storing associated with decoded symbol 1-166
Part Queue
  clear current records 1-117
  display current records 1-118
  displaying summary of 1-127
  enabling 1-115
  number of records to collect 1-122
  saving 1-126
  set up to save remotely 1-120
  specify exact failure to store 1-119
  uploading 1-123
Photometry
  learn values 1-84
  setting 1-129
Physical Trigger
  redirect? 1-182
PID
  delete stored 1-50
  index
    displaying stored values 1-198
  loading info 1-86
  removing from trigger 1-145
PIDs
  listing valid 1-53
Polarity
  set data matrix expected 1-132
  setting for io lines 1-164
POSITION_ENHANCE_ENABLED Command 1-66
Pre-processing
  set morphological 1-101
Probe
  set direction 1-134
  setting speed 1-135
  setting threshold 1-136

Q
QR Finder Misalignment
  setting 1-138
QuicSet
  specifying what it can perform 1-141
Quiet Zone Length
  setting barcode 1-24

R
Read Mode
  setting matrix orientation 1-112
Reading Statistics
  tracking 1-165
Records
  clear current in part queue 1-117
  display current in part queue 1-118
  number of to collect in part queue 1-122
Redirect
  physical trigger? 1-182
Region of Interest
  defining 1-153
Releasing
  control of camera 1-144
Remote
  commands
    displaying information about 1-71
Removing
  assigned pid from trigger 1-145
Report
  retrieve next from system 1-147
  trailer
    setting 1-179
ReportChart 2-4
ReportConnection 2-3
ReportGrid 2-4
Reporting
  calibration status string 1-37
  camera light calibration 1-36
  mean light value 1-38

state of command just changed 1-73
value of cell unit multiplier 1-42
REPORTREQ Command
canceling 1-146
Retrieving
cycle records 1-124
next report from the system 1-147
Retry Mode
setting 1-149
setting for all modes 1-151
Return
mac address 1-88
Rows
data matrix
defining number to expect 1-154
RRKIT.OCX 2-2
Run
factory set commands 1-156

S
Sampling
cell 1-40
Save
part queue 1-126
remotely
part queue 1-120
Saved Job
loading 1-86
Saving
current configuration 1-158
Search Method
barcodes 1-23
Selecting
1-d symbologies 1-16
dpm verification parameters 1-189
gpio output mode 1-75
lighting type 1-85
verification grade ranges 1-197
Serial
checking parameters
specifying 1-99
Setting
apipa 1-52
auto and manual settings 1-128
barcode
confidence level 1-18
height 1-19
length 1-21
quiet zone length 1-24
threshold 1-25
vertical probe spacing 1-26
width 1-27
bc412 polarity 1-33
bounce latch time 1-171
calibration
status string 1-37
target contrast value 1-169
target reflectance value 1-170
camera
offline 1-106
online 1-111
cell
sampling 1-40
size 1-41
unit report type 1-43, 1-47
data matrix
expected polarity 1-132
height 1-70
decode output header 1-69
decoder
to read variety of symbols 1-188
dwell time 1-62
edge direction 1-173
error correction level 1-63
horizontal probe spacing 1-20
initial match count 1-94
interrupt detection delay 1-172
light source wavelength 1-201
match control 1-89
to character matching 1-100
match parameters 1-95
matrix orientation in read mode 1-112
morphological pre-processing 1-101
multiple tty settings 1-186
number of barcodes to be decoded 1-22
number of barcodes to decode 1-104
photometry 1-129
probe threshold 1-136
qr finder misalignment 1-138
report trailer 1-179
source of images 1-39
static ip address 1-78
synthesized aperture size 1-12

tcpip domain 1-61
triggering mode 1-181
warping speed 1-200
width to height ratio 1-142
Settings
display current 1-163
Shortcuts
terminal 1-10
Size
cell 1-41
Software
listing current version 1-196
Source of Images
setting 1-39
Specify
exact failure to store in part queue 1-119
max number of data matrix codes to find 1-105
serial checking parameters 1-99
Speed
setting
probe 1-135
warping 1-200
setting probe 1-135
Static IP Address
setting 1-78
Storing
all parameters associated with decoded symbol 1-166
Subnet Mask
displaying for camera 1-79
Successful
decode output
customizing 1-107
match output
customizing 1-109
Symbologies
controlling 1-D 1-15
Synchronize
local time 1-178
Synthesized Aperture Size 1-12

T
Taking Control
camera 1-48
Target Laser
turning on or off 1-168
TCP/IP Domain
setting 1-61
Terminal 2-4
shortcuts 1-10
Threshold
method
binarizing 1-175
setting probe 1-136
Time
synchronize local 1-178
Tracking
reading statistics 1-165
Train
decoder 1-130
Trigger
assigning learned decoder parameters 1-13, 1-14
causing a virtual to occur 1-199
configuring virtual 1-183
removing pid 1-145
Triggering Mode
set 1-181
Triggers
generating periodic virtual 1-159
TTY Settings
set multiple 1-186
Typed Data
echoing 1-174

U
UDP Packets
controlling broadcast 1-187
Unsuccessful
decode output
customizing 1-49
match output
customizing 1-97
UPC as EAN 1-29
Uploading
part queue 1-123
Use
match list in match mode? 1-93

V

Valid PIDs
  listing 1-53
Value
  cell unit multiplier 1-42
Variables
  assigning to digital output lines 1-76
Verification
  enabling aim dpm-1-2006 1-195
  grade ranges
    selecting 1-197
  turning on or off aim 1-192
Version
  listing current 1-196
Vertical Probe Spacing 1-26
Virtual Trigger
  causing to occur 1-199
  configuring 1-183
Visual Feedback
  optical alignment 1-139
VxWorks Console
  set to serial port 1-45

W

Width
  setting expected data matrix 1-202
Width to Height Ratio
  setting 1-142