

code

V002450

Code Reader 2, 2500, 3, 3500 & 1200 Reader-Host Interface Specification CLIENT VERSION

MASTER DATE: 12/23/08
COGE: WPO

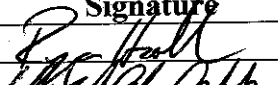
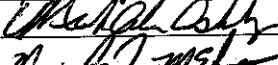

Reviewed By	Role	Signature	Date
Ryan Hoobler	COGE		12/23/08
Mark Ashby	Engineering		12/30/08
Mike McArthur	Engineering		12/23/08

Table of Contents

1	Scope.....	5
2	Host to Reader Command Overview.....	5
2.1	Packetized Commands.....	5
2.2	Text Commands.....	5
2.3	Barcode Commands.....	5
2.4	Training.....	6
3	Definitions.....	6
3.1	Notations.....	6
4	Communication Medium.....	6
5	Reader to Host Communication.....	6
5.1	Raw Data.....	7
5.2	Packet Data.....	7
6	Host to Reader Communication.....	11
6.1	Text Commands.....	11
6.2	Packetized Commands.....	12
6.3	Command Types.....	13
6.4	Simple Protocol.....	21
7	Reader Settings.....	21
8	Radio Commands.....	39
9	Code Reader Batch (CRB) System.....	40
10	Symbology Detail Settings.....	41
10.1	PharamaCode.....	41
11	OCR Template.....	43
11.1	OCR Introduction.....	43
11.2	OCR Overview.....	43
11.3	OCR Output String Values.....	44
11.4	OCR User Templates.....	45
11.4.1	End of Template (0).....	45
11.4.2	New Template (1).....	45
11.4.3	New Line (2).....	46
11.4.4	Define Group Start (3).....	46
11.4.5	Define Group End (4).....	46
11.4.6	OCR Internal Templates.....	51
12	Appendix: B-String Settings.....	53
13	Appendix: Example CRC16 C Code.....	57

Table of Figures

Figure 1: Example 'z' and 'i' Packets in Raw and Packet Modes.....	10
--	----

code

Changes From Last Release

Description	Section(s)
Changed register default value of 0x87, 0x9f, 0XA7, 0xAF, 0x10D, 0x124	7
Added registers 0x17f,0x1b2	7

1 Scope

This interface control document (ICD) specifies the communication protocol between the Code Reader 2 and application software that runs on the host computer, specific reader commands, examples of a variety of ways to communicate and send data to the reader (i.e., RS232, USB, RF) and command/communication types.

It is important to note that some functionality has changed and/or been added for each firmware release. Make sure to use the latest released firmware for both boot and app. CR2/CR3 default values listed in Section 7 were compiled using firmware version 4039. CR1200 default values listed in Section 7 were compiled using firmware version 4180. CR2500/3500 default values listed in Section 7 were compiled using a pre-released firmware version 4504.

2 Host to Reader Command Overview

This section is intended to introduce developers/users to the basic command types of the reader. There are two ways to send a command to the reader; from a host computer, or by scanning a barcode containing a command sequence. In addition, there are two methods of sending a command from a host computer to the reader; packetized and text commands.

2.1 Packetized Commands

Packetized commands are the most reliable way to communicate to the reader. The packet consists of a prefix and a suffix. The prefix contains the amount of data to be transmitted and the suffix contains error detection. Unlike text commands, packetized commands are always enabled. (See Section 6.2)

2.2 Text Commands

Text commands are provided as an easy way to send a command to a reader but they lack the reliability of packetized commands. In addition, text commands must be enabled. Text commands can easily be sent from a terminal program and uses a %xx (similar to URL encoding) to translate an escape sequence containing a 2-digit hex value to the corresponding single 8-bit ASCII character. This allows non-printable ASCII characters to be entered via the terminal program. Text commands can be sent via the RS232, USB Virtual COM or RF mode by using appropriate communication software. In addition, the developer/user may send text commands by using CRB files. (See Section 9)

2.3 Barcode Commands

The reader will recognize the following sequence within a barcode as a command to the reader:

SOH 'X' GS STX *Text-Command* EOT (Packet does not contain spaces)

The *Text-Command* portion contains a text command as described above.

Because the Barcode Command is terminated with ASCII EOT, the *Text-Command* may not contain EOT. If the *Text-Command* needs to contain EOT, encode it as %04.

2.4 Training

Code Corporation highly recommends attending technical training provided by Code via Webinar and on site. This training is designed to provide in-depth knowledge/usage of the ICD for developers and users.

3 Definitions

3.1 Notations

The interface protocol is described as a set of grammars. Syntactic categories (non-terminals) are indicated by *italic* type; terminal symbols are indicated by **bold** type, literal byte values as hexadecimal with "0x" prefix, literal ASCII characters enclosed in single quotation marks, non-printable ASCII characters as the standard ASCII name in CAPS, key press-release sequence by key name alone, key-down by a trailing down-arrow, and key-up by an up-arrow. Alternatives are separated by the "|" symbol. Optional terminals and non-terminals are indicated with the subscript "opt." The "nz" subscript indicates that it applies to all packets except z type packets. The "nr" subscript indicates that it applies to packets sent in "non-raw" mode, i.e., in "packet" mode.

4 Communication Medium

The Reader communicates with the Host via USB, keyboard wedge (PS/2 or AT), RS232, PC-Card, or Bluetooth Serial Port Profile. The Host includes appropriate hooks and/or drivers to enable two-way communication with the Reader.

5 Reader to Host Communication

The Reader may be configured in raw mode, where no packet framing or check characters are sent, and packet mode. (See section 2.1 and 2.2) The Reader may also be configured to expect an acknowledgement from the host after each packet and automatic retry when no acknowledgement is received. Standard "one-way" mode of operation uses raw packets, no expected response from host, and no automatic retry. Standard "two-way" mode of operation uses packets with framing and checks characters, expects a response from the host, and automatically resends. If no Acknowledgement is received (Ack), three (3) attempts to resend are made.

5.1 Raw Data

Reader to Host communication consists of decoded raw data having no framing or check characters. Raw data is sent with no “end of packet” data (crc16). One-way communication, expects no response from Host and no data is resent.

5.2 Packet Data

Data from the Reader to the Host consist of *packets* as specified below. Packetized data is sent using ACK/NAC protocols with framing and check characters. Packets are delivered asynchronously as graphical codes are read and in response to Host to Reader commands. For keyboard communication (USB keyboard or PS/2 or AT keyboard), all *ascii-characters* are transmitted as *keyboard-sequences*. For all other communication ports, all *ascii-characters* are transmitted as *ascii-bytes*.

Note: when sending data via the keyboard port, the state of the Caps-Lock is assumed to be “off” (i.e., capital letters are always shifted; lower-case letters are never shifted). The Host shall perform capitalization-translation, if necessary, based on the actual state of the Caps-Lock.

Note: Even though the data size field allows up to 65535 bytes of data in a packet, the actual size of a packet either in raw or in packet mode including data and packet overhead is maximum 16384 bytes.

<i>packet:</i>	<i>start packet-type_{nz} data_{opt} end</i>
<i>start:</i>	<i>packet-start_{nr} codeXML-start_{nz}</i>
<i>packet-start:</i>	SOH ‘X’ ‘R’ <i>protocol-version reader-id packet-number</i> <i>timestamp data-size</i>
<i>end:</i>	<i>codeXML-end_{nz} crc16_{nr}</i>
<i>codeXML-start:</i>	SOH ‘X’ RS <i>tag_response ‘/’</i>
<i>codeXML-end:</i>	EOT
<i>tag_response:</i>	‘ap’
<i>packet-type:</i>	Single ascii-character in table below
<i>data:</i>	<i>character</i>
	<i>data character</i>
<i>protocol-version:</i>	‘1’
<i>reader-id:</i>	big-endian 32-bit number
<i>packet-number:</i>	<i>data-packet-number cmd-packet-number</i>
<i>data-packet-number:</i>	any byte value in the range [0,7f]; increments with each packet; does not increment with resends; used with <i>z</i> and <i>a</i> packets only
<i>cmd-packet-number:</i>	any byte value in the range [80-ff]; increments with each packet; does not increment with resends; used with all packets other than <i>z</i> and <i>a</i>
<i>timestamp:</i>	big-endian 32-bit number, indicates timestamp in seconds (relative to Reader power-up or last time set in Reader) (For all but <i>z</i> packets, the timestamp represents the time the packet was sent to the host; for <i>z</i> packets, the time the code was read.)

code

<i>data-size:</i>	big-endian 16-bit number indicating size of the <i>data</i> field (in bytes)
<i>character:</i>	<i>byte</i>
	/ <i>keyboard-sequence</i>
<i>byte:</i>	any byte value in range [0x00,0xFF]
<i>keyboard-sequence:</i>	<i>key</i>
	/ shift ↓ <i>key</i> shift ↑
	/ alt ↓ <i>decimal-code</i> alt ↑
<i>key:</i>	~ 1 2 3 4 5 6 7 8 9 0 - =
	q w e r t y u i o p [] \
	a s d f g h j k l ; '
	z x c v b n m , . /
	space
	esc tab shift alt ctrl enter backspace
	f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12
	insert delete home end pageup pagedown
	left right up down keypadenter
	<i>digit</i>
<i>decimal-code:</i>	<i>digit</i> / <i>digit digit</i> / <i>digit digit digit</i> (range [0,255])
<i>digit:</i>	keypad0 keypad1 keypad2 keypad3 keypad4
	keypad5 keypad6 keypad7 keypad8 keypad9
<i>crc16:</i>	<i>big-endian 16-bit number</i> representing crc16 of the packet, calculated over the entire packet, excluding the crc16 itself. (See source files <i>crc16.[hc]</i> (Appendix) for details on the crc16 algorithm and polynomials to be used.)

The following *packet-types* are defined:

- a indicates that *data* contains the first part of a decode. A sequence of *a* packets always ends with a *z* packet. The data of all *a* packets in a group and the final *z* packet are concatenated by the host. (Also see the Host-to-Reader ‘R’ command description.)
- d indicates that a command and its associated data were successfully received; *data* optionally contains a null-terminated text message.
- e indicates that a command was not successfully received, i.e., it had a bad type, size, or checksum, and should be re-sent; *data* optionally contains a null-terminated text message.
- g indicates that a group of *z* and/or *p* packets follows, terminated by a *d* or *e* packet (*d* for complete group, *e* for incomplete group)
- h indicates that *data* contains a zero-terminated Bluetooth connection string (of printable ASCII characters): IIII BBBBBBBBBBBBB L
where:
 - IIII is the storage index
 - BBBBBBBBBBBBBB is the Bluetooth Device Address (twelve hexadecimal digits)
 - L is “y” if link key stored, “n” if no link key stored

code

- i** indicates that *data* contains the zero-terminated Reader information string (of printable ASCII characters and TAB) in the following format:
VVVVWWWXXXXSSSSSSSSSSSAOODYYYYHHIIIIJJJKKKLLLL<TAB>Z...Z
where:
VVVV is the application firmware version number;
WWW is the bootloader firmware version number;
XXXX is the radio firmware version number;
SSSSSSSSSS is the Reader's serial number (ten digits);
A is "A" if running firmware is the application, "B" if bootLoader;
OO is the OEM identifier;
D is "D" if and only if the unit has a keypad and display (otherwise, it may be any other printable ASCII character);
YYYY is the flash file system version number;
HH is the hardware type identifier:
00 simulator
01 CR1
02 CR2/CR3
03 CR2-based OEM module
04 CR1200
IIII is the hardware version number
JJJJ is the maintenance utility version
KKKK is the operating system kernel version
LLLL is the root file-system version
<TAB> is the ASCII TAB character;
Z...Z is the OEM decoder version: a string of up to 16 printable ASCII characters
- At least vvvvwwwxxxxsssssssssa will be present. Depending on hardware type and firmware version, one or more of the other fields (and the tab character) may be omitted. For fields to the left of the tab character, if a given field is present, all fields to the left of it will also be present.
- m** indicates that *data* contains a message (comment). The *m* packets are not sent when the Reader is in "raw" mode.
- p** indicates that *data* contains a portion of a compressed or uncompressed image
- r** indicates that the Reader attempted but failed to read a code. (This packet is sent only if the reader is configured to notify the host on unsuccessful read attempts.)
- z** indicates that the packet contains data decoded from a code; *data* contains the data decoded from the code.
The *z* type packets do not use the *codeXML-start*, *packet-type*, or *codeXML-end* fields.

In "raw" mode (as opposed to "packet" mode), type **m** packets are not sent, only the decoded data is sent for type **z** packets, and all other packets are sent without the *packet-*

start and *crc16* fields. In “packet-mode,” the *packet-start* and *crc16* fields are always sent. (See Figure 1.)

Raw Mode													
<i>‘z’ (data) packet:</i>													
Data													
<i>‘i’ (non-z) packet:</i>													
codeXML ‘i’ response													
CodeXML-start				packet-type		data				codeXML-end			
SOH	‘X’	RS	‘ap/’	‘i’		VVVV...				EOT			

Packet Mode																
<i>‘z’ (data) packet:</i>																
packet-start								Data				packet-end				
SOH	‘X’	‘R’	‘I’	reader ID (4 bytes)	packet number (1 byte)	time stamp (4 bytes)	data size (2 bytes)	data				crc16 (2 bytes)				
<i>‘i’ (non-z) packet:</i>																
packet-start								codeXML ‘i’ response				packet-end				
SOH	‘X’	‘R’	‘I’	reader ID (4 bytes)	packet number (1 byte)	time stamp (4 bytes)	data size (2 bytes)	SOH	‘X’	RS	‘ap/’	‘i’	VVVV...	EOT	crc16 (2 bytes)	

Figure 1: Example ‘z’ and ‘i’ Packets in Raw and Packet Modes

Optionally, whenever the Host receives a packet, the Host will respond by sending a *Y* or *R* packet (*defined in the Host to Reader Communication section*) to the Reader. If the *expectResponse* option is enabled in the Reader configuration, the Reader will repeatedly retransmit the packet (a configurable number of times) until it receives a *Y* packet.

If a packet received by the Host has a *packet-type* that is not any of the valid types listed above or has the same *packet-number* as the last processed packet of the corresponding type (command or data), the entire packet – up to and including *end* or until timeout – should be discarded by the Host. If the Host had requested a response, it should reissue the request.

If a packet received by the Host from the Reader fails its CRC, the Host should send an *R* packet to the Reader to request that the packet be resent.

6 Host to Reader Communication

Commands and data from the Host to the Reader are sent in the form of *commands* as specified in this section.

Commands are normally sent in USB Native, RS232, and Bluetooth modes. Commands may not be sent via keyboard modes.

Two command formats are supported: **text-command** and **packetized-command**. Text-command format is applicable to RS232 and Bluetooth modes but only if the Reader is configured to accept this format. Packetized-command format is applicable to all interfaces.

text-command: See Section 6.1.

normal-command: See Section 6.2.

The command types are enumerated in Section 6.3.

After the host sends each complete command, it should wait for a response packet from the Reader. Expected responses are specified along with the command types in section 6.3. If the Reader responds with an *e* packet or doesn't respond within a reasonable timeout period, the Host should resend the command a reasonable number of times.

6.1 Text Commands

Text commands may be sent to the reader in RS232, USB Virtual COM mode, or RF mode using any serial communications software, e.g., HyperTerminal. Text commands may also be sent via the USB and Serial Downloader programs using files with the .crb file extension. The .crb file contains one command per line in the same format as *text-command*. (See Section 9).

Encoded-data is decoded by the Reader by replacing %xx by a single byte with the value specified by the two hex-digits xx, e.g., **%25** would be replaced by character number 0x25, which is ASCII %.

text-command: *command-type encoded-data_{opt} carriage-return*

command-type: Single ASCII character in the set defined in Section 6.3

encoded-data: *encoded-datum*
/ *encoded-data encoded-datum*

encoded-datum: *printable-character | % hex-digit hex-digit*

code

printable-character: any byte value in the range [0x20,0x7e]

hex-digit: ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’
 | ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’
 | ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’

carriage-return: **0x0d**

In order to eliminate inadvertent commanding of the reader, Text Commands are disabled by default (in firmware version 2216+).

To enable Text Commands requires an initial sequence: ;>PAx where x is as defined in section 7, register setting 41. (Note: ‘A’ is the ASCII character that corresponds to 41 HEX.)

For example, to send the reader commands by typing commands in HyperTerminal:

```
;>PA1  
P(xx)yy  
P(xx)yy  
~  
PA8
```

Where **;>PA1** enables text commands with echo and command responses; **P%xyy** can be any desired commands; ~ saves the settings just sent (the ~ command saves all but communication-related settings); and **PA8** turns text commands back off (except for the initial sequence). (Note: ‘A’ is the ASCII character that corresponds to 41 hex, thus P%418 would be equivalent.)

Note: **;>PA1** is used for interactive text commands. If the commands are to be saved in a file and sent non-interactively, use **;>PA7** instead; this enables text commands but disables echo and command responses. (See Section 6.3, Section 7, and Section 9 for additional information.)

The following two examples can be sent to a Reader in RS232 mode from HyperTerminal by just typing the example text.

Example 1 (make the reader beep/vibrate 3 times):

#%03 *Expected output: should make reader beep/vibrate 3 times*

Example 2 (set reader to continuous-read, near field only):

P(C4)5*Expected output: should set reader to continuous-read, near field only*

6.2 Packetized Commands

Packetized commands consist of packetized data sent from host to reader to configure and cause the Reader to perform certain functionalities (e.g. Code XML rules, and settings). Packetized commands are always enabled, unlike text commands. In addition, they include error detection data, making them more robust than text commands.

code

normal-command: *prefix command-type data-size data_{opt} reserved crc14*

prefix: **0xEE 0xEE 0xEE 0xEE**

command-type: Single ASCII character in the set defined in section 6.3

data: *datum*
 / *data datum*

datum: any byte value in the range [0,255]

data-size: byte value in range [0,240], which indicates size of *data* (in bytes)

reserved: **0x00**

crc14: two consecutive bytes, each in range [0,127], representing crc16 & 0x7f7f, most significant byte first. The packet crc16 is calculated over the entire packet, excluding the *prefix* and the *crc14* itself. (See source files *crc16.[hc]* (Appendix) for details on the crc16 algorithm and polynomials to be used.)

6.3 Command Types

ESC Reserved for use by JavaScript applications. This command can be used to notify JavaScript functions from a B-String. No defined function in firmware or CodeViewer JavaScript.

(The Reader will respond with *d* or *e*.)
causes the reader to beep and/or vibrate the specified number of times; *data* contains the number as a single character in the range [0,127].

\$ (The Reader will respond with *d* or *e*.)
Example – beep/vibrate three times: #%03
posts a simulated button event to the reader; *data* contains the event number as a single character. See setting 39 in Section 7 for a list of the event numbers.

((The Reader will respond with *d* or *e*.)
Example – read both near and far fields: \$%03
causes the reader to upload any logged error messages (no *data*)

(The Reader will respond with a *g* packet, zero or more *z* packets, and a final *d* or *e*. Each *z* packet contains a portion of the requested data in its *data* field. Note: this is very similar to the response to the X command; however, *p* packets are not applicable and the *g* and *d/e* packets are not suppressed even in raw mode.)

-) causes the Reader to erase its log of error messages (no *data*)
(The Reader will respond with *d* or *e*.)
- * causes the Reader to deactivate all top-level CodeXML rules (no *data*)
(The Reader will respond with *d* or *e*.)
- + causes the Reader to activate a CodeXML rule; *data* specifies the rule number as a string of ASCII decimal digits
(The Reader will respond with *d* or *e*.)
- , causes the Reader to send a list of current reader settings (no *data*)
(The Reader will respond with *d* containing a space-separated list of all setting values (in order, expressed as hexadecimal ASCII characters) or with *e*.)
- causes the Reader to deactivate a CodeXML rule; *data* specifies the rule number as a string of ASCII decimal digits
(The Reader will respond with *d* or *e*.)

- . causes the Reader to flash its LEDs; *data* contains four bytes:
LEDs repetitions onTime offTime
LEDs specifies the state of the LEDs: 1 to turn on, 0 to turn off:

Bit	LED	Applicable Hardware
7	reserved, always 0	CR2, CR3
6	left blue	CR2
5	left green	CR2
4	left red	CR2
3	center green	CR3, CR1200
2	center red	CR3, CR1200
1	right green	CR2
0	right red	CR2

Both onTime and offTime are specified in 1/100th seconds (max 2.55 seconds).

(The Reader will respond with *d* or *e*.)

Example – flash left LED amber 3 times, ½ second on, 1 second off:

.%30%03%32%64

- / toggle a bit (or bits) in a Reader setting; *data* contains a printable ASCII string in the following format: hexadecimal setting number in parentheses followed by a 32-bit signed integer value, expressed in ASCII hexadecimal characters (with optional minus sign) or ASCII decimal characters preceded by the '#' character, e.g., /(2e)1000 or /(2e)#4096; the specified integer is XORed with the existing setting value.
(The Reader will respond with *d* or *e*.)
Note: see Section 7 for possible reader settings.
- 1 indicates the start of a file download; *data* is empty. This command is followed by a sequence of 2 commands containing the file data and a download-end command (e.g., 5).
(The Reader will respond with *d* or *e*.)
- 2 indicates a continuation of a file download; *data* contains the next portion of the file data.
(The Reader will not send any response.)
- 5 indicates the end of a regular file download; *data* contains the name of the file, which is from 1 to 200 letters, digits, periods, hyphens, and underscores, terminated with ASCII NUL.
(The Reader will respond with *d*, *e*, or *f*.)
Note: this command is supported in firmware version 3100+.
- 9 requests the Reader to delete a file from its storage; *data* contains the file name, terminated with ASCII NUL.
(The Reader will respond with *d* or *e*.)
Note: Supported in firmware version 3100+.
- : requests the Reader to send configuration *data* to the radio module
(The Reader will respond with zero or more *h* followed by a *d* or an *e*.)
For example, *data* consisting of a single zero byte will cause the radio to disconnect from the host. (See Section 8: Radio Commands.)
This command is valid for Firmware versions 2012+.
- ; no operation (may be used for comments in command files)
- < causes the Reader to send a list of saved reader settings (no *data*)
(The Reader will respond with *d* containing a space-separated list of all setting values (in order, expressed as hexadecimal ASCII characters) or with *e*.)

code

- = puts setting directly to Reader's non-volatile memory so that it will take effect upon next reboot; *data* is as defined in the / command; the specified integer replaces the existing setting value.
- Note: this command can be used to set communication modes without losing communication during the process.
- (The Reader will respond with *d* or *e*.)
- This command is valid for Firmware versions 2230+.
- > causes the Reader to send a string of text to the host as a *z* packet; *data* contains the text to send.
- (The Reader will respond with a *z* packet containing the text.)
- ? programs the JavaScript, gocode, or other license; *data* contains the encrypted license data.
- (The Reader will respond with *d* or *e*.)
- This command is valid for Firmware versions 2232+.
- @ causes the reader to reset its internal date/timestamp to the specified time; *data* contains the date and/or time in one of the following formats.
- yyyy-mm-dd hh:mm:ss
 - yyyy-mm-dd hh:mm
 - hh:mm:ss
 - hh:mm
- Note: the separators are optional; only digits are significant.
- (The Reader will respond with *d* or *e*.)
- Examples:
- | | |
|----------------------------------|----------------------|
| Set to midnight: | @00:00 |
| Set to Sept 1, 2005 11:52:02 PM: | @2005-09-01 23:52:02 |
- Note: on units without a battery-backed real-time clock, the date and time will reset to 2000-01-01 00:00:00 upon power-up.)
- Command enhanced with version 2526+ to include year/month/day information

A notifies the Reader that the previously sent data were rejected for one of the following reasons:

- the packet was encrypted and the decryption failed;
- the host (CodeXML Modem) is locked to a different reader.

The Reader should indicate to the user that the packet has been rejected; e.g., it may sound error beeps. See related setting 0x12f, notify-of-packet-rejection.

(The Reader will not respond to the host.)

This command is valid for Firmware versions 3280+.

B defines the strings the Reader will return (or process internally) in response to stored-command-code events; *data* contains the event number of the stored-command-code as a single byte (**in the range 0x09..0x48**) followed immediately by the associated null-terminated string. B-strings are “performance” strings used to enhance/enable specific reader functions and capabilities.

0x09 – 0x0C: reserved for OEM

0x0D – 0x24: reserved for Code Corporation Functions

0x0D Performance String A1

0x0E Performance String A2

0x0F Performance String A3

0x10 Reserved as of firmware version 2526

0x11 Performance String B1

0x12 Performance String B2

0x13 Performance String B3

0x14 Performance String C1

0x15 Performance String C2

0x16 Performance String C3

0x17 Performance String D1

0x18 Performance String D2

0x19 Performance String D3

0x1a reserved

0x1b reserved

0x1c Performance String SB

0x1d Performance String SN

0x1e Performance String SF

0x1f Performance String VB

0x20 Performance String VN

0x21 Performance String VF

0x22 Performance String AB

0x23 Performance String AN

0x24 Performance String AF

0x25 – 0x47: for user applications

0x48 User defined “No Read” value (see setting 55 in Section 7)

(The Reader will respond with *d* or *e*.)

Example –B-string setting SXGA Near Field Window:

B%0D%01X%1d%02P%54#1024%04

B-strings 0x0D through 0x24 are restored using J command as of version 2526. Performance strings are included with app download and do not require a separate download as of version 2526.

See “Appendix: B-string Settings” for complete details.

G get setting from Reader; *data* contains a single character (0-255), which is the setting number.

(The Reader will respond with *d* and the setting value as a sequence of 8 ASCII hexadecimal digits or with *e*.)

Example - determine if Rectangular Data Matrix is enabled: G%16

Note: see Section 7 for possible reader settings.

I requests the Reader to send its information string (no *data*).

(The Reader will respond with *i* or *e*.)

J requests the Reader to restore settings to factory defaults (no *data*).

(The Reader will respond with *d* or *e*.)

L requests the Reader to send a list of its stored files; *data* is either empty or contains “1”; hidden files are listed only if “1” is specified. (Hidden files are files whose names begin with a period.)

(The Reader will respond in the same manner as with the ‘(’ command, each *z* packet containing a file name as a NUL-terminated string of printable ASCII characters.)

Note: Supported in firmware version 3100+

N requests the Reader to delete stored data and images (no *data*).

(The Reader will respond with *d* or *e*.)

O set a bit (or bits) in a Reader setting; *data* is as defined in the / command; the specified integer is ORed with the existing setting value.

(The Reader will respond with *d* or *e*.)

Note: see Section 7 for possible reader settings.

- P put setting to Reader; *data* is as defined in the / command; the specified integer replaces the existing setting value.
(The Reader will respond with *d* or *e*.)
Example – enable Rectangular Data Matrix (setting 0x16, value 1):
P%161
Note: see Section 7 for possible reader settings.
- Q clear a bit (or bits) in a Reader setting; *data* is as defined in the / command; the ones-complement of the specified integer is ANDed with the existing setting value.
(The Reader will respond with *d* or *e*.)
Note: see Section 7 for possible reader settings.
- R requests that the previously sent packet be re-sent by the Reader; *data* may specify a maximum packet size the receiver will accept: *data* is either empty or specifies a 16-bit big-endian unsigned integer (2 bytes). If *data* is empty or specifies a size less than 32 (the minimum packet size), the Reader will use its preferred maximum packet size. Otherwise, it will use the specified max packet size (or less) and will fragment data across multiple smaller packets when necessary.
(The Reader will respond by resending its previous packet or with *e* if there was no previous packet. If the max data size has changed, it may resend the previous data in a sequence of more than one packet.)
- T requests the current date and time (no *data*)
(The Reader will respond with *d* with *data* containing the date and time formatted as yyyy-mm-dd hh:mm:ss.)
Note: on units without a battery-backed real-time clock, the date and time will reset to 2000-01-01 00:00:00 upon power-up.
Supported in firmware version 2526+
- U requests the Reader to delete all stored files including data, images, and Javascript files (no *data*)
(The Reader will respond with *d* or *e*.)
Supported in firmware version 2526+
- W requests the Reader to write its current settings from RAM to its non-volatile memory.
(The Reader will respond with *d* or *e*.)

- X initiates data transfer from Reader memory to Host; *data* specifies whether only buffer data or all data (buffer and log) should be sent: if empty, all data are sent; otherwise, 0 requests only buffer data and 1 requests all data (buffer and log).
Note: if the autoLogErase setting is nonzero, the log data are cleared after being successfully sent.

(The Reader will respond with a *g* packet, zero or more *z* or *p* packets, and a final *d* or *e*. In raw mode, the *g* and *d* or *e* packets are omitted; thus if there are no data stored then no response will arrive.)
- Y acknowledge the receipt of a packet; *data* specifies the received packet number (one byte).

(The Reader will not respond.)
- Z request the Reader to reboot; *data* is either empty or contains ‘1’; if it contains 1, the Reader will reboot into boot loader mode.

(The Reader will respond with *d* or *e* before it reboots.)
- \ Enhanced to include the Z1 feature in firmware version 2526+
returns a string previously stored with the *B* command; *data* contains the event number of the stored-command-code as a single byte.

(The Reader will respond with a *d* packet containing the requested null-terminated string or with *e* (e.g., if the specified event number is out-of-range).)
- See “Appendix: B-string Settings” for complete details.
- ^ requests the Reader to upload the specified stored file; *data* contains the file name, terminated with ASCII NUL.

(The Reader will respond in the same manner as with the ‘(’ command with the following additions: in the *g* packet, *data* contains the filename followed by a tab character, followed by the file’s size in parentheses (e.g., “test.txt (1292)”); in the *d* packet, *data* contains “EOF” followed by a tab character, followed by the file’s CRC16 in parenthesis (e.g., “EOF (13626)”.)
- _ Supported in firmware version 3100+
causes the Reader to wait for all buttons to be released and clear its event queue

(The Reader will respond with *d* or *e*.)

| sends *data* to the Reader. The *data* may be intercepted by the JavaScript application on the Reader; otherwise, it will be processed as if read from a code.

(The Reader will respond with *d* or *e*.)

~ requests the Reader to write some of its current settings from RAM to its non-volatile memory. All settings are written *except* the communication settings (commMode, commProtocol, commExpectResponse, and commandOptions, uartBaud, etc.)

(The Reader will respond with *d* or *e*.)

This command is valid for Firmware versions 2230+.

6.4 Simple Protocol

The file is split into blocks of 236 or less bytes each and downloaded to the Reader via *I*, *2*, & *5* commands using the following sequence:

- 1) Send a *I* command to initialize the download.
- 2) Wait for a *d* or *e* response from the Reader or a timeout.
 - a) If timeout or *e* response, restart the sequence at step 1.
 - b) If *d* response, continue to step 3.
- 3) Send a series of *2* commands, each with a portion of the file. (The reader will not send any response.)
- 4) Send a *5* command to end the download and install the file.
- 5) Wait for a *d*, *e*, or *f* response from the Reader or a timeout.
 - a) If *f* response or timeout, restart the sequence at step 1.
 - b) If *e* response, repeat step 5.
 - c) If *d* response, file download has completed successfully.

Note: the timeout will need to be increased from the normal response timeout to allow the firmware time to write the file to the flash memory.

7 Reader Settings

The Host sets the Reader settings using the */*, *O*, *P*, *Q*, and *=* commands and reads them using the *G*, *,*, and *<* commands.

For example, the following *P* command sets register *2e* to the value *7f*.

```
P(2e)7f
```

Note: for two-digit setting numbers (i.e., settings 00 through fd), an alternative format may be used: in place of the parentheses and hexadecimal setting number, substitute a single character, which represents the setting number. The equivalent to the example above is *P.7F* (the ASCII '.' character has the hexadecimal value 2e). (In certain circumstances, such as with text-commands, "percent-encoding" may be used for encoding a character as a sequence consisting of the percent character followed by two hexadecimal digits. With percent-encoding, the example may be expressed as *P%2e7f*.)

code

In the table below, the **Reg** column is the setting number, in **hexadecimal**, to be used with the the commands identified above. In the **Default** column, all values are in **decimal** unless otherwise specified.

Note: If only one default value is shown then this value is applicable and identical to both CR2/CR3 and CR1200. All differences in the settings use or default value will be noted.

Reg	Setting Name	Default	Comment
00	Bluetooth Radio Out-of-range (enableRfDisconnectAlarm)	0	Alarm when RF doesn't have connection 0: no alarm 1: vibrate 2: beep 3: vibrate and beep
01	Battery Trigger (button8)	CR2/CR3: 4 CR2500/3500: 4 CR1200: 5	The event associated with the charge pin trigger. Used by the BH1, BH2, and H2. Is also the trigger button for the CR1200. See setting 39 for a list of event numbers. Note: Supported in firmware version 2362+
02	button8ConfirmationTime_ms	0	See setting e3. Note: Supported in firmware version 2362+
04	Continuous Illumination Leave illumination on during read (leaveIlluminationOnDuringRead)	CR2/CR3: 0 CR1200: 1	0: turn illumination off between read attempts 1: leave illumination on between read attempts Note: Supported in firmware version 2362+ Note: Obsolete for CR2/CR3 firmware 3430+
05	No Wait USB usbKbEnumNotWaitForSetReportLed	0	0: default mode, declare enumeration after receive set LED status report 1: declare enumeration after receive get report descriptor command. Used for some Windows CE based devices Special case for USB enumeration that doesn't require Host keyboard response Note: Supported in firmware version 2362+
06	alwaysEnableAwb	0	0: disable auto white balance 1: enable auto white balance Note: Supported in firmware version 2378+ Note: Requires power cycle before taking effect.
07	kbSendDelay_kbClocks	0	Each keyboard clock period is 40 microseconds. Note: Supported in firmware version 2378+
08	reader packet format (commProtocol)	1	0: reserved 1: raw 2: packet mode 3: reserved

code

Reg	Setting Name	Default	Comment
09	kbInterMessageDelay_ms	0	Delay between each data transmit in the unit of millisecond Note: Supported in firmware version 2378+
0a	nec2of5Options	1	0: disabled 1: enabled add 2 to enable checksum checking add 4 to enable checksum checking and strip checksum from the result string add 8 to decode 1 digit symbol add 16 to decode 2 digit symbol add 24 to decode 1 and 2 digit symbol Note: All lengths greater than 2 are always enabled. Note: Supported in firmware version 2378+
0b	matrix2of5Options	1	0: disabled 1: enabled add 2 to enable checksum checking add 4 to enable checksum checking and strip checksum from the result string add 8 to decode 1 digit symbol add 16 to decode 2 digit symbol add 24 to decode 1 and 2 digit symbol Note: All lengths greater than 2 are always enabled. Note: Supported in firmware version 2378+
0c	telepenEnable	1	0: disabled 1: enabled
0d	enableNonSquareDataMatrix	0	0: disabled 1: enabled Note: Supported in firmware version 2378+
0f	Targeting Control (targetEnable)	CR2/CR3: 3 CR1200: 3	Turns on or off the targeting feature. Bit 0: Green enabled (CR1200)/ Target Enabled (CR2/CR3) Bit 1: Red Enabled (CR1200 / Target Enabled (CR2/C3) Note: Setting to 0 will disable all targeting features.
14	Image transform (transformImage)	0	0: no transform 1: mirror 2: invert
16	DataMatrix Rectangular Symbology (dataMatrixRect)	0	0: disabled 1: enabled
19	DataMatrix Symbology (enableDataMatrix)	1	0: disabled 1: normal Data Matrix 2: inverse Data Matrix enabled 3: both normal and inverse enabled

code

Reg	Setting Name	Default	Comment
1a	straight 2 of 5 enable(enableMiscBarcodes)	1	0: disable straight 2 of 5 (with 2 or 3 start/stop codes) decoding 1: enable straight 2 of 5 (with 2 or 3 start/stop codes) decoding
1b	Communications Mode (commMode)	2	<p>CR2/CR3:</p> <p>0: PS/2 (AT) keyboard 1: RS232 serial 2: USB keyboard 3: reserved 4: RF (BlueTooth) 5: USB Native (HID) 6: USB VComm (3000+ firmware) 7: USB HID POS -Terminal 131 (firmware 3484+)</p> <p>CR1200:</p> <p>0: PS/2 (AT) keyboard (firmware 4126+ and serial number 10050561+) 1: RS232 serial 2: USB keyboard 3: reserved 4: not valid 5: USB Native (HID) 6: USB VComm (3000+ firmware) 7: USB HID POS -Terminal 131 (firmware 4144+)</p> <p>This setting is used in conjunction with settings 08 and 42 to configure the communication mode between standard “one-way” and “two-way” modes.</p> <p>For example, USB “two-way” native: 1b: 5 (USB Native) 08: 2 (packet mode) 42: 1 (expect response)</p> <p>Note: numlock-numlock-capslock-capslock-numlock-numlock sent within 1 second will change reader from USB keyboard mode to native mode. In firmware 3550+ register 78 also disables this feature.</p> <p>Note: PS/2 is not supported on the Embedded CR1200</p>
1c	Baud Rate (uartBaud)	57600	All standard baud rates up 115200
1d	Stop Bits (uartStopBits)	2	1: send 1 stop bit 2: send 2 stop bits

code

Reg	Setting Name	Default	Comment
1e	Data Bits (uartDataBits)	8	7: 7 data bits 8: 8 data bits
22	Parity (uartParity)	0	0: none 1: odd 2: even
26	Beep/Vibrate Volume (beepVolume_percent)	100	This is the percentage of full volume. The range is 0 to 100. See also settings 59 and a7.
29	PDF417 Symbology (enablePdf417)	1	0: disabled 1: enabled Also see settings 2a and cf.
2a	Micro PDF417 Symbology (enableMicroPdf417)	0	0: disabled 1: enabled Also see settings 29 and cf.
2b	QR Code Symbology (enableQrCode)	0	0: disabled 1: normal QR enabled 2: inverse QR enabled 3: both enabled 15: enable all QR add 4 to the value above to enable Micro QR code add 8 to the value above to enable inverse Micro QR code. Note: Micro QR is supported in firmware version 2378+
2c	Extra Time Before Power-Saving Idle Mode When Cable Is Connected (extraCabledActiveTime_ms)	120 * 60 * 1000	Valid range: 0 to 0x7FFFFFFF milliseconds Note: in "continuous read" mode, this is also the continuous read timeout period.
2d	Keyboard Maps (kbMap)	0	0: US English (without leading 0 in the alt _ Num) 1: ASCII (alt+number) - universal 2: Custom (requires user to download keyboard map) 3: US English (with leading 0 in the alt + num for non-printable ASCII) 4: French Keyboard 5: German Keyboard 6: Japanese Keyboard 7: US English (with ctrl + char for non-printable ASCII) Version 2394+
32	Time Before Power-Saving Idle Mode (activeTime_ms)	5 * 60 * 1000	Reader goes into power-saving idle mode automatically after this amount of time of non-use. The valid range is 0 to 0x7FFFFFFF milliseconds. Note: in "continuous read" mode, this is also the continuous read timeout period.

code

Reg	Setting Name	Default	Comment
34	Maximum Candidate Decodes Per Read (maxStickyDecodes)	1	The Reader will process up to this number of codes per “read code” event. If there are more than this many codes in the field of view and within target tolerance, only the first ones will be decoded. For fastest performance with single codes, set to 1. Also see setting 44.
35	Button Stay-Down Time (stickyTime_ms)	0	Keep processing the “read code” events for this amount of time (act as if the button sticks down for this time) The valid range is 0 to 0x7FFFFFFF milliseconds.
36	Number of Control Frames Before Picture Capture (agcFramesBeforePicture)	6	Number of frames captured and discarded before live picture to give the gain control time to adjust The valid range is 0 to 0x7FFFFFFF. See also settings 43, ab, ac, ad, ae, af, & b1.
37	Host Acknowledgement Time Limit (hostAckTimeout_ms)	700	After sending data to host, the Reader waits up to this amount of time for the acknowledgement from host before declaring failure. Valid range: 0 to 0x7FFFFFFF milliseconds
39	Right Top Button (button1)	4	The specified event is posted upon press of this button. The events are defined below. 0: no action 1: keep awake 2: show target 3: read in near and far fields 4: defaultEvent (default) Selected by hardware - CR2/CR3 read in both fields - CR1200 read in near fields - CR2500 read in wide field 5: read in near/high density field 6: read in far/wide field 7: take picture 8: read in most recently successful field 09 – 71: execute stored command string 72 – 254: reserved 255: idle
3a	Left Top Button (button2)	4	See setting 39
3b	Combination of Left and Right Button (button3)	CR2/ CR3: 0	See setting 39
3c	Handle Button (button4)	4	See setting 39
40	Text Command Timeout (hostPacketTimeout_ms)	11000	The maximum time during which a complete text command from host must be received. (Pending text command data is discarded when the timeout is exceeded.) Valid range: 0 - 0x7FFFFFFF milliseconds

code

Reg	Setting Name	Default	Comment
41	Text Commands (commandOptions)	8	<p>0: disable 1: enable text commands 3: enable with suppress echo 7: enable with suppress echo and suppress responses 8: disable text commands but enable magic sequence “;>PAx” where x is 1, 3, or 7 as defined above. This would normally be used in command text files, which would begin with the text-command-on sequence and end with the command to return to this special mode. For example: ;>PA7 ;any desired commands here PA8</p> <p>Note: ;>PAx Supported in firmware version 2210+.</p> <p>Additional settings in 3430+ Bit 4: comm_suppressUrlDecode. For example, if enabled, P%418 will not equal PA8. The % is not recognized as an escape character Bit 5: comm_acceptOnTimeout. Note: See register 0x156</p>
42	Expect Acknowledgement From Host (commExpectResponse)	0	<p>0: reader doesn't wait for acknowledge 1: reader will retransmit data when host doesn't acknowledge receipt</p>
43	JPEG Picture Quality (jpegQuality_percent)	50	<p>1..100: JPEG compression quality percent 0: raw image (no JPEG compression) See also settings 36, ab, ac, ad, ae, af, & b1.</p>
47	Maxicode Symbology (enableMaxiCode)	0	<p>0: disable Maxicode 1: enable Mode 0 2: enable Mode 1 4: enable Mode 2 8: enable Mode 3 16: enable Mode 4 32: enable Mode 5 64: enable Mode 6 127: enable all modes</p> <p>To disable MaxiCode decoding, set the property value 0. Otherwise, for each bit from bit 0 (the least significant bit) to bit 15, if the bit is set to 1, then the MaxiCode mode 0 corresponding to the bit position is enabled. For example, if the property is set to hex 3C, then modes 2 through 5 are enabled and all other modes are disabled. Maxicode Modes 0 – 6 supported</p>
48	Codabar Checksum (miscBarcodeChecksum)	0	<p>0: disable Codabar checksum checking 1: enable Codabar checksum checking 2: enable Codabar checksum checking and strip the checksum from the result string</p>

code

Reg	Setting Name	Default	Comment
49	Code 39 Symbology (code39FullAscii)	0	0: disable Code 39 full ASCII support 1: enable Code 39 full ASCII support
4a	Composite Codes (enableCompositeCodes)	0	0: disable 1: enable See setting d8.
4b	Postal Code Symbology (enablePostalCodes)	0	0: disable 0x8: Australian Post decoding enabled 0x20000: Japan Post decoding enabled 0x200001: KIX decoding enabled 0x80: Planet decoding enabled 0x2000: Postnet decoding enabled 0x200000: Royal Mail decoding enabled -0x80000000: USPS4CB (Firmware version 3418+)
4c	RSS Symbology (enableRss)	0	1: Enable RSS Expanded decoding. 2: Enable RSS Expanded Stacked decoding. 4: Enable RSS Limited decoding. 8: Enable RSS-14 and RSS-14 Truncated decoding. 16: Enable RSS-14 Stacked and RSS-14 Stacked Omnidirectional decoding. Values may be combined. For example, 11 would be Expanded + Expanded Stacked + RSS-14 and RSS-14 Truncated 31: enable all
4d	UPC Expansion (enableUpcExpansion)	1	0: disable 1: enable See also settings 4e, 6a, and 74.
4e	UPC Supplemental (enableUpcSupplemental)	1	0: disable 1: enable See also setting 4d, 6a, and 74.
4f	MSI Plessey Symbology enableMsip	0	0: disable 1: enable
50	Aztec Symbology (enableAztec)	0	0: disabled 1: standard Aztec 2: inverse Aztec 3: normal and inverse Aztec
53	SXGA Near Field Window Vertical Size (CR1200 Horizontal) (nearFieldDecodeWindowWidth_pixels)	CR2/CR3: 640 CR1200: 1280	Decoding is attempted in only the specified window of the full image. The valid range is 1 to 640. Note: the width and height refer to the physical image, which is rotated 90 degrees on CR2. This setting is applicable to SXGA mode only. See setting f1 for VGA.

code

Reg	Setting Name	Default	Comment
54	SXGA Near Field Window Horizontal Size (CR1200 Vertical) (nearFieldDecodeWindowHeight_pixels)	CR2/CR3: 1024 CR1200: 640	Decoding is attempted in only the specified window of the full image. The valid range is 1 to 1024. Note: the width and height refer to the physical image, which is rotated 90 degrees on CR2. This setting is applicable to SXGA mode only. See setting f2 for VGA.
55	Notify of No-Read (notifyOfReadFailure)	0	0: disable 1: send "r" packet on No-Read (See "r" packet in Section 5.2.) 0x100xx: post event on No-Read , where the lower 8 bits specify the event number. For example, 0x10009 to post Event 0x09. The following example will use a stored code and stored code event to output "No Read" on read failure. B% 48%01X% 1d%02>No%0dRead%00P%5510048 Output when no decode was achieved: No (%0d inserts carriage return) Read
59	Beep/Vibration Duration (beepDuration_ms)	CR2/CR3: 100 CR1200: 80	Valid range: 0 to 0x7FFFFFFF See also settings 26 and a7.
6a	UPC Symbology (enableUpc)	1	0: disable 1: enable See also settings 4d, 4e, and 74.
6b	Code 39 Symbology (enableCode39)	1	0: disable 1: enable See also setting 70.
6c	Code 93 Symbology (enableCode93)	1	0: disable 1: enable
6d	Code 128 Symbology (enableCode128)	1	0: disable 1: enable
6e	Interleave 2 of 5 Symbology (enableInterleave2of5)	1	0: disable 1: enable See also setting 71 and c9.
6f	Codabar Symbology (enableCodabar)	1	0: disable 1: enable
70	Code 39 CheckSum (code39Checksum)	0	0: disabled 1: enabled 2: enabled and strip from results See also setting 6b.
71	Interleave 2 of 5 Checksum (interleave2of5Checksum)	0	0: disable 1: enable 2: enable checksum and strip from output See also setting 6e and c9.

code

Reg	Setting Name	Default	Comment
72	Auto Stored Data Erase (autoLogErase)	1	0: disable 1: enable Note: When “1,” data and images are cleared from nonvolatile memory when they are successfully uploaded to the host. (In “Log mode,” this is set to “0.”)
73	Auto Transfer Buffer Memory (autoBufferUpload)	1	0: disable 1: enable When “1,” the reader will automatically upload buffered data (i.e., storage that hasn’t been previously uploaded) whenever a connection is present.
74	UPC Short Margin (enableUpcShortMargin)	1	0: disable 1: enable See also settings 4d, 4e, and 6a.
75	RS-232 Batch Mode (uartAlwaysConnected)	0	0: detect RS232 cable by presence of power on pin 1. 1: assume RS232 cable is always connected
76	Storage Mode (sendAndStoreMode)	0	0: normal mode (buffered send) 1: send and log mode 3: log only mode Also see settings 72 and 73
78	settingsLock	1	1: settings unlocked 3: settings locked (except settingsLock)
85	Trioptic Options	1	0: disable 1: enable, normal quiet zones are assumed 3: enable, allow short quiet zone symbols 5: enable no quiet zones required (requires firmware version 3280+)
86	Motion Detection Event (motionEvent)	0	When motion is detected, this event is posted. See register 39 for list of events. 0 disables motion detection.
87	Motion Detection Sensitivity (motionThresh)	CR2/CR3: 50, CR1200: 15	The sensitivity of motion detection. Lower numbers make detection more sensitive; higher numbers, less sensitive.
88	Sleep Timer Value (sleepTimeout_sec)	2*60*60	The Reader will turn off after inactivity of this duration.
89	bluetoothPin	12345678	Value of the Bluetooth Pin sent if the host request a pin. Note: See register 173 Note: Supported in firmware 3514+
8e	Time Before Power-Saving Standby Mode (idleTimeout_sec)	90	The Reader enters power-saving standby mode after inactivity of this duration.

code

Reg	Setting Name	Default	Comment
93	suppressBeepOnDecode	0	Normally, the Reader beeps as soon as decodes are read and processes them via JavaScript if necessary <i>after</i> the beep. To enable JavaScript to control the beep feedback, change this setting to 1; this will suppress the beep; the JavaScript would typically beep if the decode is valid or start another read cycle if it isn't. 0: beep before JavaScript processing 1: just call JavaScript Note: supported in firmware version 3100+
9c	Laser Brightness (targetBrightness_percent)	100	The valid range is 0 to 100.
9d	Target Tolerance (targetTolerance_percent)	1600	For the Reader to accept a code, the target dot must be within the code rectangle or in proximity to the symbol. The nearness is defined as this percentage of the code's smaller dimension. For example, with a 10 x 20 mm code and a setting of 150 (%), the target dot must be within 15 mm of the code. Any value over 1000 is considered infinite tolerance, and no target checking is performed.
9e	Extra Time Before Sleep When Cabled (extraCabledIdleTime_sec)	0x7fffffff	When connected to a power cable, the Reader waits this time + idleTimeout_ms before going into power-saving sleep mode.
9f	Time Before Power-Saving Sleep Mode (standbyTimeout_sec)	0	The Reader enters power-saving sleep mode after inactivity of this duration. Note: when in sleep mode, host communication is disabled. (This is in contrast with idle mode, where communication is still enabled, and standby mode, where normal communication is disabled but the radio maintains its connection (RF mode). Idle and standby modes use more power than sleep mode (standby using less than idle but more than sleep).) Note: supported in firmware version 2526+
a1	Vibrate (vibrate)	CR2/CR3: 0 CR1200: 1 CR2500/3500: 1	0: disable 1: enable: vibrator will be on at same time as speaker Note: if vibrate-only is desired, set speaker volume to 0.
a2	defaultEventDelay_ms	0	The Reader will pause for this amount of time between each posting of the default event (used with "continuous read" mode). See setting c4.

code

Reg	Setting Name	Default	Comment
a7	Beep (Vibrate) Pulse Separation (beepSeparation_ms)	CR2/CR3: 20 CR1200: 100	The spacing in milliseconds between beeps. The valid range is 0 to 0x7FFFFFFF. See also settings 26 and 59.
ab	AGC Selection for Picture Taking (useImagerAgcWithTakePicture)	0	0: use decoder AGC (designed for Symbology decoding) 1: use imager AGC (optimized for pictures) See also settings 36, 43, ac, ad, ae, af, & b1.
ac	Picture Window Left Position (pictureWindowX)	0	Specify position and size of window used with "take picture." The position and size are relative to the virtual image (i.e., not the rotated physical image). Note: on Code Reader 2, overall image is 1024 x 1280. Upper half is far field; lower half is near field. See also settings 36, 43, ab, ad, ae, af, & b1.
ad	Picture Window Upper Position (pictureWindowY)	0	See setting ac.
ae	Picture Window Width (pictureWindowWidth)	CR2/CR3: 1024 CR1200: 1280	See setting ac.
af	Picture Window Height (pictureWindowHeight)	CR2/CR3: 640 CR1200: 1024	See setting ac.
b0	Laser Target on Before Picture (targetBeforePicture)	0	0: laser off before picture capture 1: laser on before picture capture See also settings 36, 43, ab, ac, ad, ae, & af.
b1	When to apply CodeXML rules (configCodesBeforeCodeXml)	2	Controls the sequence of processing configuration strings and applying CodeXML rules. 0: Process configuration strings only before applying CodeXML rules 1: Process configuration strings only after applying CodeXML rules 2: Process configuration strings before and after applying CodeXML rules
b3	Time of retries before reader gives up sending (commMaxSendAttempts)	3	used with commExpectResponse (Note: minimum is 1, i.e., original send attempt but no resends.)
b4	noStoreIfNotConnected	CR2/3: 0 CR1200: 1	0: normal buffer operation 1: nothing is stored in nonvolatile memory when there is not a valid connection. If there is no active connection and data would have otherwise been stored, the Reader will indicate this fact the same as with a storage-full condition.
bf	USB Keyboard Poll Rate (keyboardPollingPeriod)	CR2/CR3: 10 CR1200: 5	The host is requested to poll the USB device at the specified period. Valid range: 1 to 255 milliseconds

code

Reg	Setting Name	Default	Comment
c4	Default (Continuous) Event (defaultEvent)	255	When no button is pressed but the reader is still in active mode (i.e., not power-saving idle or sleep modes), this event will be posted. See setting 39 for the list of events. The default value (idle event) disables “continuous scanning”; use one of the read events to enable “continuous scanning.”
c6	Auto Connect (autoConnectMode)	1	0: noAutoConnect (connect only on “X” and “:” commands and upload events) 1: autoConnect (attempt to establish connection when in idle mode and maintain connection when in standby mode) 2: autoReconnect (attempt to connect when there is data to send but only within specified time of last valid connection) See setting ea. 3: autoConnectIfCabled (attempt to connect if reader is cabled or in charger) Note: This functionality is implemented in the Codeviewer application. If custom JavaScript is developed, caution must be taken to include this functionality.
c7	SXGA Far Field Window Vertical Size (farFieldDecodeWindowWidth_pixels)	CR2/CR3: 640 CR2500/CR3500: 640	Valid range: 1..640 This setting is applicable to SXGA mode only. See setting f3 for VGA.
c8	SXGA Far Field Window Horizontal Size (farFieldDecodeWindowHeight_pixels)	CR2500/CR3500: 1024	Valid range: 1..1024 This setting is applicable to SXGA mode only. See setting f4 for VGA.
c9	interleave2of5Lengths	0	0xFFFFFFFFC: 2 and 4 digit disabled 0xFFFFFFFFD: 2 digit enabled 0xFFFFFFFFE: 4 digit enabled See also settings 6e and 71.
ca	Auto Disconnect (autoDisconnect)	0	0: retain connection until sleep or explicit disconnect command 1: disconnect from the host when there is nothing to send. (In conjunction with autoConnect and autoStorageUpload, the reader will connect when there is data to send, send the data, then disconnect (to allow another reader to connect to the same host).
cd	Codablock A Symbology (enableCodablockA)	0	0: disable 1: enable
ce	Codablock F Symbology (enableCodablockF)	0	0: disable 1: enable

code

Reg	Setting Name	Default	Comment
cf	Macro PDF417 (macroPdf417)	0	0: disable 1: enable See also settings 29 and 2a.
d6	Ignore Duplicate Code (duplicateBlockTime_sec)	0	Consecutive duplicate codes (i.e., codes that contain the same data) are blocked for this amount of time (in seconds). The valid range is 0 (off) to 0x7FFFFFFF. Note: Deprecated, but maintain for backwards compatibility. If using 3430 or greater firmware, use register 159.
d8	Compostite Linkage Control (compositeCodesRequireBoth Elements)	1	0: accept any composite element 1: only accept composite codes if both elements could be decoded. See also setting 4a.
d9	Max Connection Wait Time (connectTimeout_sec)	30	The reader will attempt connection for up to this amount of time when a connection is explicitly requested, such as when a QuickConnect code is read or an upload is requested (by event or command). Valid range: 0 to 0x7FFFFFFF seconds
e2	imagerResolution	0	0: SXGA (1280x1024) 1: VGA (640x480)
e3	button1ConfirmationTime_ms	0	The button must be pressed and held for this amount of time (without change in which buttons are held down) before the button press is accepted. Setting this value > 0 makes it easier to select combinations of buttons (e.g., button3, which is button1 and button2 pressed together).
e4	button2ConfirmationTime_ms	0	See setting e3.
e5	button3ConfirmationTime_ms	0	See setting e3.
e6	button4ConfirmationTime_ms	0	See setting e3.
e7	button5ConfirmationTime_ms	0	See setting e3.
e8	button6ConfirmationTime_ms	0	See setting e3.
e9	button7ConfirmationTime_ms	0	See setting e3.
ea	reconnectTimeout_sec	90	See setting c6.
eb	Maximum reader to host packet data size (maxPacketSize)	16384	Range 1 to 16384
ec	Host Acknowledgement Time Limit Multiplier (hostAckTimeoutMultiplier_ms)	15	When commExpectResponse is nonzero, the Reader will wait up to hostAckTimeout_ms + dataSize * hostAckTimeoutMultiplier_ms milliseconds to receive an acknowledgement from the host. Valid range: 0 to 0x7FFFFFFF.
ed	Prefix Result with AIM Symbology Identifiers (enableSymbologyIdentifierPrefix)	0	0: don't prefix with AIM identifier 1: prefix decode result with ISO/IEC standard 15424/AIM symbology identifier

code

Reg	Setting Name	Default	Comment
f0	Allow Code 128 Short Margin (allowCode128ShortMargin)	1	0: disable 1: enable
f1	VGA Near Field Window Vertical Size (CR1200 Horizontal) (vgaNearFieldDecodeWindow Width_pixels)	CR2/CR3: 320 CR1200: 640	CR2/CR3: 1 to 320 pixels Note: Supported in firmware version 2178+. CR1200: 1 to 640 pixels
f2	VGA Near Field Window Horizontal Size (CR1200 vertical) (vgaNearFieldDecodeWindow Height_pixels)	CR2/CR3: 480 CR1200: 480	CR2/CR3: 1 to 480 pixels Note: Supported in firmware version 2178+. CR1200: 1 to 480 pixels
f3	VGA Far Field Window Vertical Size vgaFarFieldDecodeWindowW idth_pixels	CR2/CR3: 320	1 to 320 pixels Note: Supported in firmware version 2178+.
f4	VGA Far Field Window Horizontal Size vgaFarFieldDecodeWindowH eight_pixels	CR2/CR3: 480	1 to 480 pixels Note: Supported in firmware version 2178+.
f6	Code 39 Short Margin (enableCode39ShortMargin)	1	0: disallow short margin Code 39 symbol decoding 1: allow short margin Code 39 symbol decoding Note: Supported in firmware version 2180+.
f7	Code 11 Symbology (code11Config)	0	0: disable Code 11 decoding 1: enable Code 11 decoding with two checksum digits checked 3: enable Code 11 with one checksum digit checked 5: enable Code 11 with two checksum digits checked and stripped from the result string 7: enable Code 11 with one checksum digit checked and stripped from the result string Note: Supported in firmware version 2182+.
f8	PharmaCode Symbology (pharmaCodeConfig)	0	See Section 10.1
f9	PharmaCode Barcount (pharmaCodeBarCount)	4100 (0x1004)	Bit 0 – Bit 7: min bar count, 4 to 16 Bit 9 – Bit 15: max bar count, 4 to 16 Range from 0x0404 to 0410 and 0x1004 to 0x1010
fa	PharmaCode Min Value (pharmaCodeMinValue)	15	15 to 131070
fb	PharmaCode Max Value (pharmaCodeMaxValue)	131070	15 to 131070
fc	Keep reading codes as longs as button is held down (keepReadingWhileButtonIsPr essed)	0	0: disable (requires button to be released before next scan occurs) 1: enable Note: when enabled, duplicateBlockTime_sec should be set > 0.

code

Reg	Setting Name	Default	Comment
fd	log battery level and timestamp (logBatteryLevel)	0	0: disable 1 – 0xffffffff: number of scans between each log entry
100	Backlight Timeout (backlightTimeout_ms)	3000	Backlight goes off automatically after this amount of time, in milliseconds, after a button press. Valid range 0 to 0x7fffffff milliseconds Note: Supported in firmware version 2526+.
101	Backlight Brightness (backlightBrightness_percent)	75	Backlight is illuminated at this percent value Valid range 0 to 100 Note: Supported in firmware version 2526+.
102	Keypad beep volume (keypadBeepVolume_percent)	0	Beeps at specified percentage of full beeper volume whenever a keypad key is pressed Note: Supported in firmware version 2526+.
103	Display white mode gray scale settings (lcdWhiteMode)	0x0000	For all gray scales, each digit (4 bits) divided by 9 represents the percent of time a pixel of the corresponding gray scale is turned on in each frame. The lowest 4 bits correspond to frame 1, next 4 to frame 2, etc. Valid range (each digit): 0 to 9. Note: Supported in firmware version 2526+.
104	Display light gray mode gray scale settings (lcdLightGrayMode)	0x0097	Note: Supported in firmware version 2526+.
105	Display dark gray mode gray scale settings (lcdDarkGrayMode)	0x9996	Note: Supported in firmware version 2526+.
106	Display black mode gray scale settings (lcdBlackMode)	0x9999	Note: Supported in firmware version 2526+.
10b	enableJavaScript	1	When set to 0, installed scripts are disabled, which can be used (from boot mode) for recovering the unit if a buggy script is installed. Note: supported in firmware version 2526+
10c	rfConnectedCacheTime_sec	3	The time the last connection status received from the radio remains valid If a request is made during this time since last radio query, the cached status is returned. Otherwise, the Reader will query the radio for connection status (which takes up to 1 second).
10d	DataMatrix Symbol Size (dataMatrixSymbolSize)	CR2/CR3: 0 CR1200: 2	Increases the decoder's effort to find a DataMatrix symbol in an image. 0: Normal effort (Default) 1: Increase effort 2: Max effort Note: supported in firmware version 3100+ See also setting 0x1b2

code

Reg	Setting Name	Default	Comment
10e	Legacy Picture Upload (legacyPictureUpload)	1	<p>Selection of picture upload method:</p> <p>0: Store pictures as files – Pictures will be stored as files and must be uploaded using the “^” command. Take-picture event will store rather than immediately upload picture.</p> <p>1: Legacy picture upload – Take-picture event will attempt to immediately upload picture using the legacy image upload protocol. (If upload fails, the picture will be stored as a file.). Also, the ‘X’ command will cause stored picture files to be uploaded using the legacy image upload protocol. (The images will not be automatically transferred when a connection is established; the ‘X’ command is needed.)</p> <p>Note: supported in firmware version 3100+</p>
12c	dataMatrixMiscImprovement	0	<p>Improves the decoding capability of the reader on low contrast or pixilated Data Matrix barcodes</p> <p>Bit 0: Binarization Improvement Bit 1: Low Contrast Improvement</p> <p>Note: Supported in firmware version 3280+</p>
12d	Hong Kong 2 of 5 Symbology	0	<p>Bit 0: Enable Bit 1: 1 Digit Symbology Bit 2: 2 Digit Symbology</p> <p>Note: Supported in firmware version 3280+</p>
12f	notifyOfPacketRejection	1	<p>Specify the behavior when a packet is rejected because of incorrect encryption key, incorrect packet protocol, or CodeXML Modem locked to a different reader.</p> <p>0: disable 1: beep 3 times 0x100xx: post event on No-Read , where the lower 8 bits specify the event number. For example, 0x10009 to post Event 0x09.</p> <p>Note: Supported in firmware version 3280+</p>
137	pdfHandleInvalidShift	0	<p>Allows the decoding of PDF417 barcodes that were improperly encoded</p> <p>0: Disable 1: Enable</p> <p>Note: Supported in firmware version 3280+</p>

code

Reg	Setting Name	Default	Comment
150	Background Bluetooth Connection (synchronousSend)	0	<p>Allows user to begin scanning before Bluetooth connection is confirmed</p> <p>0: Bluetooth connection is confirmed before user is allowed to scan. Pressing the trigger button will cancel the connection. 1: Allows user to begin scanning before the Bluetooth connection is confirmed.</p> <p>Note: Firmware 3430+</p> <p>Note: See register 151 and a0.</p>
151	Beep before Bluetooth Connection is established (quickConnectNotWaitForConnection)	0	<p>Give the the second beep before connectilon is established when scanning QuickConnect code . To be used with register 150 to improve user experience with QuickConnect code</p> <p>0: Beep upon establishing a Bluetooth Connection 1: Do not wait for connection before beep .</p> <p>Note: Firmware 3430+</p> <p>Note: See register 150.</p>
154	Enable Black and White Pictures (takeBlackAndWhitePicture)	0	<p>Converts grey scale images to black and white</p> <p>0: Images remain in grey scale. 1: Captured images are converted to black and white.</p> <p>Note: Firmware 3430+</p>
159	Ignore Duplicate Code (duplicateBlockTime_msec)	0	<p>Consecutive duplicate codes (i.e., codes that contain the same data) are blocked for this amount of time (in milliseconds). The valid range is 0 (off) to 0x7FFFFFFF.</p> <p>The actual block time is the sum of settings d6*1000+159.</p> <p>Also see setting d6.</p> <p>Note: Supported in firmware CR1200 version 4112+, CR2/CR2 version 3430+.</p>

code

Reg	Setting Name	Default	Comment
172	Auto Save Active RF Connection BD_Address (rfAutoSaveActiveConnect)	CR2/3: 0	Auto save the RF connection address. The address in the QuickConnect code will be saved if enabled. Other communication settings such as RF one-way or RF two-way are not automatically saved, so the reader must be saved in the proper communication mode for this feature to work properly. 0: Disabled – Address not automatically saved. 1: Enabled – Address automatically saved. Note: Supported in firmware version 3474+
173	Enable Bluetooth Encryption (enableBluetoothEncryption)	0	Enables the standard AES 128 bit Bluetooth encryption. 0: Disabled 1: Enabled Note: see register 0x89 Note: Supported in firmware 3514+
175	Rotate CR3 display and keypad (rotateKeypadDisplay)	0	0: No Rotation 1: Rotate display 180 degrees 2: Swap arrow keys 3: Rotate display 180 degrees and Swap keys Note: Requires reboot to take effect Note: Supported in firmware 3546+
17f	jsMaxMemory_bytes	CR2/CR3: 6*1024*1024 CR1200: 12*1024*1024	JavaScript Maximum Memory Usage
180	jsPeakMemory_bytes	0	Readonly. JavaScript Peak Memory Usage
1b2	HD DataMatrix Symbol Size (hdf_dataMatrixSymbolSize)	CR2/3/1200: not supported CR2500/3500: 0	Increases the decoder's effort to find a DataMatrix symbol in an image – HD filed only. 0: Normal effort (Default) 1: Increase effort 2: Max effort Note: supported in firmware version 4008+

8 Radio Commands

The Host controls the radio by issuing ':' commands. The following table describes the available commands.

The '#' column is the radio setting/command number (in hexadecimal) to be used with the ':' command. For example, ":%0E" gets the Bluetooth device address.

code

The '# bytes' column indicates how many bytes of data are required as arguments for the command.

Name	#	Comments	# bytes
Disconnect	00	Terminate the current connection.	0
Auto Connect	07	If connection information exists for the specified Device Address, use it to establish a connection. Otherwise, attempt to establish a connection and store the resulting information.	6
Clear Setup	08	Remove connection information associated with the specified BlueTooth Address.	6
Send Setup	09	Print all connection information in the following format iiii xxxxxxxxxxxx p Where iiii is the storage index, xxxxxxxxxxxx is the BlueTooth Device Address, and p indicates pairing enabled (y) or pairing disabled (n).	0
Get Bluetooth Address	0E	Get Bluetooth address as 12 Hexdecimal characters	0
Get "user friendly" name	0F	Get device's "user friendly" name	0

9 Code Reader Batch (CRB) System

The Code Reader Batch (CRB) system is a convenient method for creating and maintaining a set of commands that can be easily sent to the reader. These CRB files can be created in any text editor with the file extension of .crb. The CRB system accepts all of the valid *text commands*. The most commonly used commands are *J*, *N*, *P*, and *~*. There should be one command per line. The CRB file may contain empty lines and comments as well.

The .crb files can be sent to the reader using either the USB or RS232 downloader's. As CRB files are just a list of *text commands*, they can also be sent by a serial terminal program. **Note: if using a serial terminal program the reader will first need to be commanded in to "text command mode"; see Section 6.1.**

You can request a copy of all the CR2 Users' Manual configuration codes in the .crb format. For example, code M121_01 in the manual (setting the Bluetooth radio timeout to 5 minutes) is the following:

File M121_01.crb:

```
P%9f#300
```

The CRB system allows for combining multiple functions into a single file. For example, you can create a test.crb file that contains commands to set the left trigger to B3

code

(M030_01), enable ALL RSS codes (M267_01), disable auto-transfer of buffer memory (M069_01), and make the reader beep three times. Note you can also comment your CRB file with a semicolon (;) as shown below. A comment starts with a semicolon character and lasts till the end of the line. The `>PAx` sequence has special meaning. See Section 0.

File test.crb:

```
; This is a test file
;>PA7      ; enable text commands, no echo/responses (see Section 7, reg 41)
P%3A13    ; left trigger set to B3 performance (note that value 13 is in hex)
P%4c#31   ; enable ALL RSS codes (note that value 31 is in decimal)
P%730     ; disable Auto Transfer Buffer Memory
; beep reader three times using a different type of sound
P%59#30   ; set Beep/Vibration Duration to 30 milliseconds
P%26#60   ; set Beep/Vibrate Volume to 60%
#%03      ; beep reader 3 times
P%59#100  ; restore Beep/Vibration Duration to 100 milliseconds
P%26#100  ; restore Beep/Vibrate Volume to 100%
P(100)#6000 ; sets the backlight timeout to 6 milliseconds (6ms is in decimal)
P(101)55  ; dimmers setting (LCD screed) to 85% (85% is in hex)
~         ; save settings (except communication-related settings)
PA8       ; turn off text commands (to avoid inadvertent commanding)
```

Non-printable ASCII (0x00-0x1F and 0x7F) and non-ASCII (0x80-0xFF) characters should be encoded. See Section 0. Also, the following characters have special meaning in a CRB file, thus they should be encoded if they are part of a command:

0x20 ' ' (space): Space and tab characters mark the end of a command.

0x3B ';' (semicolon): Semicolon characters mark the beginning of a comment.

10 Symbology Detail Settings

10.1 PharmaCode

PharmaCode setting register (f8) contains a number of settings that requires detailed explanation. Below is a list of valid register settings and detailed explanation.

0 = disable PharmaCode decoding (Default)

1 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

3 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel

raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

5 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

7 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

9 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

11 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

13 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

15 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

11 OCR Template

11.1 OCR Introduction

OCR is designed to encompass both human readable and machine readable information in the same symbol or text. Conversely, barcodes were designed to greatly assist the ability of machines to read information at the expense of human readability. In OCR, there is little redundant information in a character; most of an OCR character must be present to allow recognition. There are subtle differences between some OCR characters that are easy for a human to distinguish, but present challenges to machine vision systems in the presence of lower sample density, noisy images, and/or degraded symbols.

While much effort has been spent to provide a superior OCR capability, users should be cautioned that OCR decoders are more susceptible to misreads and noreads than their barcode counterparts. Consequently, The OCR decoder reads OCR only when it is provided with templates detailing the specifics of the text to read. This allows the software to distinguish the text of interest from random text that may be present in the same image. In addition, the OCR decoder supports a checksum capability to reduce the probability of misreads.

Multiple templates may be active at the same time for more user flexibility.

11.2 OCR Overview

The following OCR characters are currently supported:

OCR-A
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
#&()*+-./<>@\€¥

OCR-B
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
#&()*+-./<>@\€¥

The templates define the OCR font as well as the layout of the text in a row, column format. Each row can have up to 50 chars, with up to 18 rows in a template. However, the total number of characters can not exceed 320 characters. Within each character position, the allowable characters can be specified either through explicit ASCII values, groups of ASCII values, wildcard characters, or combinations of these types. To achieve better OCR results, it is desirable to limit the values that a character position can take to the known values that will occur in an application.

The OCR decoder can also handle spaces within OCR text with some restrictions. Internal gaps longer than one space are not allowed in templates. For example, the OCR text

ONE SPACE

is valid because there is only one space between the E and S in the text. However, the following text is illegal given the two spaces between the O and S:

TWO SPACES

The OCR decoder can handle arbitrary number of spaces at the beginning and end of a line. These spaces must be explicitly included in the template with the ASCII value of a space (32) and not be included as part of a group or wildcard character.

The OCR decoder also provides a checksum capability to reduce the probability of misreads. There are two types of checksums provided: row and block. A row checksum provides a checksum for all characters from the checksum to the first character of a single row in a template. A block checksum provides a checksum from its character position all the way to the first character of the overall template. For additional checksum protection, four different weighting schemes are supported: 1, 12, 13, and 137. Finally, the checksum calculation is based on modulo arithmetic. The modulo factor may vary from 6 to 36.

The OCR decoder is designed to read OCR text that is within a certain sampled range in pixels. The ideal height of an OCR character after sampling is about 20 pixels. The OCR decoder will read characters that are up to 50 pixels in height. If OCR characters are consistently above 40 pixels in height, downsampling the image by a factor of 2 before submitting it to OCR decoder will achieve better results in both speed and decode rates.

11.3 OCR Output String Values

There is no 7 bit ASCII representation for the Euro, Pound, or Yen currency characters. The 8 bit values that are returned in the result string for these characters are based on the typical code page values used in Windows. The 8 bit codes used are as follows:

Character Value

Euro 128

Pound 163

Yen 165

11.4 OCR User Templates

User Templates are NULL terminated strings made up of various control codes along with standard ASCII values. The control codes are assigned to ASCII values below the value of 32. The currently defined control codes are:

Control Code Definition	Value	Argument
End of Template	0	
New Template	1	Font [1-3]
New Line	2	
Define Group Start	3	ID [1-255]
Define Group End	4	
Wildcard: Numeric (0-9)	5	
Wildcard: Alpha (A-Z)	6	
Wildcard: Alphanumeric	7	
Wildcard: Any (including space)	8	
Defined Group	10	ID [1-255]
In Line Group Start	11	
In Line Group End	12	
Checksum	13	Weights, Type, MOD

11.4.1 End of Template (0)

All templates end with the *End of Template* control code.

11.4.2 New Template (1)

A user template may contain multiple distinct templates all within the same string. These distinct templates all begin with the *New Template* control code. The value immediately following this control code indicates the font(s) for which this template is designed. The current valid font values are

Font Value Active Font(s)

1 OCR-A

2 OCR-B

3 OCR-A and OCR-B

As an example, the following byte sequence reads 8 alphanumeric digits in either OCR-A or OCR-B and is the default user template:

1,3,7,7,7,7,7,7,7,0

11.4.3 New Line (2)

The OCR decoder supports mult-line templates. A new line within a multiline template is indicated by the *New Line* control code.

11.4.4 Define Group Start (3)

In a given character position, the user specifies which values a text character may take. To reduce the overall size of templates, users may define common groups of ASCII characters and then use the defined group rather than repeating the same sequence over and over. Groups can be made up of individual ASCII values or wildcard values. The wildcard values are control codes *Numeric* (5), *Alpha* (6), *Alphanumeric* (7), and *Any*(8). Groups may **not** be nested.

To define a group, specify the *Group Start* control code followed by a single byte *ID* value that may range from 1 to 255. Up to 255 groups may be defined in a single template. Once a group is defined, you may not define another group with the same group *ID*. Following its definition, a group may be used in any subsequent individual template.

For example, say we want to read an 8 character OCR-B text where each character may be a hexadecimal digit (0123456789ABCDEF). We can define a group that would begin the definition with the *Group Start* control code (3), followed by its ID (1), followed by the *Numeric* wildcard (5), followed by the ASCII values of the six desired letters. Finally all group definitions are terminated with the *Group End* control code (4):

1,2,3,1,5,65,66,67,68,69,70,4,10,1,10,1,10,1,10,1,10,1,10,1,10,1,10,1,0

The sequence of numbers of 65 through 70 are the decimal ASCII values for the upper case letters A through F. To use the defined group, we use the *Defined Group* (10) control code followed by the group ID. Each sequence of 10,1 above occupies a single character position in the OCR text to be read. The remainder of the example above include the *Template Start* (1) control code at the beginning followed by the OCR-B font designator (2). Finally, the template is ended with the *End of Template* (0) control code.

11.4.5 Define Group End (4)

The *Define Group End* control code is used to terminate a *Defined Group*. A *Define Group End* must always be preceded by a *Define Group Start* and conversely, a *Define Group Start* must always be followed by a *Define Group End*.

11.4.5.1 Wildcard: Numeric (5)

The *Numeric Wildcard* control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any of the 10 numeric digits.

11.4.5.2 Wildcard: Alpha (6)

The *Alpha Wildcard* control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any of the 26 upper case alphabetic letters.

11.4.5.3 Wildcard: Alphanumeric (7)

The *Alphanumeric Wildcard* control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any of the 26 upper case alphabetic letters or any of the 10 numeric digits.

11.4.5.4 Wildcard: Any (8)

The *Any Wildcard* control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any valid character that the designated font for this template supports. For OCR-A and OCR-B this includes the 26 upper case letters, the 10 numeric digits, and the 16 punctuation characters. It includes the space character as well.

11.4.5.5 Defined Group (10)

The *Defined Group* control code uses a previously defined group. The byte immediately following the *Defined Group* control code must match a previously defined group. This group occupies one character position in the template.

11.4.5.6 In Line Group Start (11)

The *In Line Group Start* defines a one time instance of a group that occupies one character position in the template. Use this control code for unique groups of characters that occur only once. One could use a *Define Group Start* control code instead. The inclusion of both types is done for the convenience of the user.

The *In Line Group* must always be ended with an *In Line Group End* control code.

11.4.5.7 In Line Group End (12)

The *In Line Group End* control code is used to terminate an active *In Line Group* definition.

11.4.5.8 Checksum (13)

Checksums may be used to reduce the probability of misreads involving OCR. The OCR decoder supports a number of options associated with checksums. The user may specify the type (block or row), the weight scheme (1, 12, 13, 137) and the modulo value of the checksum (6-36). The byte immediately following the *Checksum* control code defines the type of checksum that will be used:

Bit Position(s)	Meaning
7,6: Weight Scheme	00: Weight Scheme: 1
	01: Weight Scheme: 12
	10: Weight Scheme: 13
	11: Weight Scheme: 137
5: Checksum Type	0: Row
	1: Block
4-0: Modulo Value	Checksum Modulo - 5

Row Checksums perform a checksum calculation on all characters preceding them up to the first char on the same row. Block Checksums perform a checksum calculation on all characters up to the very first character in the template; they span multiple rows.

The 5 bit Modulo Value stores the *Checksum Modulo* - 5. The stored number may range from 1, which is a *Checksum Modulo* value of 6, to 31, which describes a *Checksum Modulo* of 36. A Modulo value of 0 (*Checksum Modulo* of 5) is illegal.

The characters within a checksum field have a numerical value that is used in the checksum calculation.

Digits are converted to their numerical value (0-9), while Upper case letters range from 10 for an 'A' up to 36 for a 'Z'. All punctuation characters currently have a value of 0 for checksum purposes. However, they do count as a spot for determining the weight values that will be used in calculating the checksum.

The *Weight Scheme* defines how the values described above can be changed based on their character position. The default weight scheme is 1. This means that the checksum is based only on the character value and is not dependent on its position. The other weight schemes multiply the character value by a repetitive weight value that helps in identifying characters that have had their column locations switched. The 4 weight schemes are:

Weight Scheme Multiplier values

1	1,1,1,1,1,....
12	1,2,1,2,1,2,....

code

13 1,3,1,3,1,3,....
137 1,3,7,1,3,7,1,3,7,....

The checksum character itself always start with a weight of 1. As we move away from the checksum towards the left, we update the weight value to the next member of the sequence. The sequences repeat over and over until the first character in a row for a **Row** type checksum, and to the first character in the template for a **Block** type checksum. The resulting sum is then divided by the **Checksum Modulo** number of the checksum and the remainder of this division should be zero for a valid checksum.

For example, the following 2 line OCR-B template contains a mod 10 checksum with weight 1 (5) at the end of each line along with a mod 36 block checksum with weight 13 (191) as the last character. There are a total of 8 alphanumeric characters per line including the checksums:

1,2,7,7,7,7,7,7,13,5,2,7,7,7,7,7,13,5,13,191,0

11.4.5.9 Example OCR Templates

This section gives some examples of valid OCR Templates along with the OCR text they are designed to read.

Multi_Row with Leading and Trailing Spaces

123456

ABCDEF G

The OCR-B text above is made up of two lines, the top being purely numeric and the bottom purely alphabetic. Also, the second line is offset from the first by two spaces. The following template will read this text:

1,2,5,5,5,5,5,5,2,32,32,6,6,6,6,6,6,0

Note the 2 following the 6 numeric wild card digits: this indicates that the template is inserting a new line. Also of note are the leading two spaces on the second line. They must be explicitly indicated by using the ASCII value of a space: decimal value 32. You may not use a wildcard or group to indicate leading (or trailing) space. Finally, the trailing spaces on line 1 do not need to be explicitly entered into the template. They are assumed to be there based on the number of character positions defined for the row.

Row and Block Checksums

ABCD6

EFG5X

The two lines of OCR-B alphabetic text above both contain a row checksum. In addition, the last character of row 2 is a block checksum. The 2 row checksums are mod 10 with a

13 weight (control code 133), while the block checksum is a mod 36 with a 137 weight (control code 255). The following template will read this text:

1,2,6,6,6,6,13,133,2,6,6,6,13,133,13,255,0

The top line checksum is the 6 at the end of the line. While this example shows the checksum at the end of the line, it may appear anywhere on the line and then protects all the characters to its left. The following sum is generated to verify a proper checksum on line 1:

```
'6' 'D' 'C' 'B' 'A'  
(1x6)+(3x13)+(1x12)+(3x11)+(1x10) = 100
```

Note the 13 weight scheme starting with a 1 on the checksum digit, and then alternating between a 1 and 3 for all digits to the left of the checksum up to the first character on the line. The numerical values of the alphabetic characters range from 10 for an 'A' up to a 36 for a 'Z'. The sum of 100 is a multiple of 10, so the mod 10 checksum here has passed.

On line 2, the row checksum is the 5 following the G. Verifying its line by generating its sum:

```
'5' 'G' 'F' 'E'  
(1x5)+(3x16)+(1x15)+(3x14) = 110
```

Again, we have obtained a value that is a multiple of 10, validating this row checksum.

The X at the end of the line is a mod 36 block checksum with 137 weighting. It protects all the characters in the template, including the first line. Calculating its sum working backwards from the block checksum and using the 137 weighting scheme:

```
'X' '5' 'G' 'F' 'E' '6' 'D' 'C' 'B' 'A'  
(1x34)+(3x5)+(7x16)+(1x15)+(3x14)+(7x6)+(1x13)+(3x12)+(7x11)+(1x10) = 396
```

The resulting sum is a multiple of 36, so the block checksum has been validated.

Multiple Individual Templates

A single template may contain multiple individual templates. For example consider the following two distinct OCR text strings that are to be read with the same template:

A1234 B5678

The first string is in OCR-A font while the second is OCR-B. Assume that the 'A' and 'B' are always present as the first character of the two strings and that the 4 numeric characters may be any digits. A user may code these up as two distinct templates with the following:

1,1,65,5,5,5,5,1,2,66,5,5,5,5,0

Notice the multiple use of the *New Template* control code (1) to separate the two individual templates along with the *Font ID* of 1 for the first OCR-A string and the 2 for the OCR-B string. In addition, the decimal values of 65 and 66 are the ASCII codes of the upper case letters A and B, respectively, and indicate that those character positions will always have those values.

ISBN 978-0-571-08989-5

The ISBN template supports this format along with the original. All the comments associated with the original ISBN format discussed above apply here as well. The one exception is the checksum is now a Mod 10 checksum and as a result, the checksum can now only be in the range from 0 – 9 and does not take the value ‘X’ anymore.

11.4.6.3 Price Field Internal Template

The Price Field is used in a number of applications including book pricing. The format of the field is as follows:

C1234 P5678E

The field begins with a ‘C’ and ends with an ‘E’. The first part of the Price Field is a ‘C’ followed by four numeric digits. The second half begins with a currency character. The above example shows the letter ‘P’ but the Price Field template allows the following additional characters:

\$€¥

Following the currency character, a numeric grouping of 3, 4, 5 or 6 digits is followed by a terminating letter ‘E’. The template supports both OCR-A and OCR-B fonts.

The following examples will also be recognized by the The OCR decoder Price Field internal template:

C6712 \$801E
C0217 €4399E
C0823 ¥31559E
C0331 £706213E

12 Appendix: B-String Settings

B%0d%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa120%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%03

B%0e%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#832%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa100%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%03

B%0f%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#480%04%01X%1d%02P%53#480%04%01X%1d%02P%c8#640%04%01X%1d%02P%c7#480%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa60%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%03

B%11%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa90%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%03

code

B%12%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#640%04%01X%1d%02P%53#512%04%01X%1d%02P%c8#832%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa75%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%03

B%13%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%f2#480%04%01X%1d%02P%f1#320%04%01X%1d%02P%f4#480%04%01X%1d%02P%f3#320%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa50%04%01X%1d%02P%dc#90%04%01X%1d%02P%e21%04%01X%1d%02P%03

B%14%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa90%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%05

B%15%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#640%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa80%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%05

B%16%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%f2#480%04%01X%1d%02P%f1#320%04%01X%1d%02P%f4#480%04%01X%1d%02P%f3#320%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04

code

%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa50%04%01X%1d%02P%dc#90%04%01X%1d%02P%e21%04%01X%1d%02P%05

B%17%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#30%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa90%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%06

B%18%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#832%04%01X%1d%02P%c7#512%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#30%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa80%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02P%06

B%19%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%f2#480%04%01X%1d%02P%f1#320%04%01X%1d%02P%f4#480%04%01X%1d%02P%f3#320%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#30%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa50%04%01X%1d%02P%dc#90%04%01X%1d%02P%e21%04%01X%1d%02P%06

B%1c%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02P%64400000FF%04%01X%1d%02P%03

B%1d%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02P%64400000FF%04%01X%1d%02P%05

B%1e%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02P%64400000FF%04%01X%1d%02P%06

code

B%1f%01X%1d%02P%28122)0%04%01X%1d%02P%e21%04%01X%1d%02Q%6440
0000FF%04%01X%1d%02\$%03

B%1f%01X%1d%02P%28122)0%04%01X%1d%02P%e21%04%01X%1d%02Q%6440
0000FF%04%01X%1d%02\$%03

B%20%01X%1d%02P%28122)0%04%01X%1d%02P%e21%04%01X%1d%02Q%6440
0000FF%04%01X%1d%02\$%05

B%21%01X%1d%02P%28122)0%04%01X%1d%02P%e21%04%01X%1d%02Q%6440
0000FF%04%01X%1d%02\$%06

B%22%01X%1d%02P%28122)5f%04%01X%1d%02O%6440000000%04%01X%1d%0
2\$%03

B%23%01X%1d%02P%28122)5f%04%01X%1d%02O%6440000000%04%01X%1d%0
2\$%05

B%24%01X%1d%02P%28122)5f%04%01X%1d%02O%6440000000%04%01X%1d%0
2\$%06

13 Appendix: Example CRC16 C Code

```
/* crcl6.h
 */

#ifndef crcl6_h
#define crcl6_h

#include <stdint.h>
#include <stddef.h>

#ifdef __cplusplus
extern "C" {
#endif

typedef uint16_t crc_t;

crc_t crc
( crc_t          initialCrc
, const unsigned char* bufPtr
, size_t        length
);

#ifdef __cplusplus
} // extern "C"
#endif

#endif

/* crcl6.c
 */

#include <crcl6.h>

crc_t crc
( crc_t          initialCrc
, const unsigned char* p
, size_t        n
)
{
    enum
    {
        crcBits = 16,
        charBits = 8,
        diffBits = crcBits - charBits
    };

    crc_t c = initialCrc;

    #include "crcl6tab.h"

    while( n-- )
        c = (c << charBits) ^ crcTab[( c >> diffBits ) ^ *p++];

    return c;
}
```

code

```
}

/*eof*/

-----
/* crcl6tab.h
 * crcl6 table of partial remainders generated by
 * mkcrctab.c with polynomial 1021.
 * included only from within crc() function in file crcl6.c
 */

static const crc_t crcTab[] =
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
    0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
    0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
    0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
    0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
    0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
    0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
    0xdbfd, 0xcbbc, 0xfbff, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
    0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
    0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
    0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
    0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
    0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
    0x5844, 0x4865, 0x3806, 0x2827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
    0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
    0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
    0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
    0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0,
};

/*eof*/

-----
```

All technical information, specifications, costs, schedules, and all related materials quoted, expressed, or implied in this document are results of our judgment at this point in time and are only estimates based upon the information available to us. We reserve the right to modify such information as we find necessary based on client requests, available technology, and other eventualities.