Code Reader Basic Programming Guide

Reviewed By	Role	Signature	Date
Ryan Hoobler	COGE		
Mark Ashby	Engineering		
Atul Shah	Engineering		
Vicki Thai	Product Management		

The software and/or products of Code Corporation include inventions that are patented or that are the subject of patents pending. U.S. Patents:

6997387, 6942152, 7014113, 7070091, 7097099, 7353999, 7519239, 7621453, 8001550, 8011584.

Code Corporation, 12393 South Gateway Park Place, Suite 600, Draper, UT 84020. www.codecorp.com

Table of Contents

1 Introduction

Code Corporation (Code) designs, develops and manufactures image-based readers and software tools for data collection applications. With expertise in software development, optics, imaging, and Bluetooth™ wireless technology, Code is an innovative leader in the Auto ID and Data Collection Industry.

There are two ways to change the behavior of the Code CR8000 Reader. One is to change register values directly by scanning command bar codes. These command codes can be found in an appendix of the manual that corresponds to your reader. The second method is to use JavaScript to create a grouping of commands to control the reader. Here again, there are two supported methods. One is writing large, complex JavaScript programs; the second is the method described here – utilizing the 'reader.js' file method.

By writing small functions that reside in a file named "rules.js" on the reader you can perform a vast majority of the most common data manipulations easily and quickly. The core firmware integrates any code placed in this file into the basic functionality of the reader. If you were familiar with the previous generation of Code products, this will provide a similar capability to B-Strings and CodeXML Rules.

Additional advanced functionality such as processing host to reader commands, accessing files, controlling data storage, sending data directly to the communication port are documented in the Code Reader Advanced Programming Guide (C005356).

Code Reader uses the Mozilla SpiderMonkey 1.8 JavaScript engine, which is distributed under the terms of the Mozilla Public License Version 1.1. Source code for this version of Spider Monkey is available at: http://www.mozilla.org/js/spidermonkey/.

Table of Contents

1.1 Document Audience

This document is designed to guide Code Reader users who wish to customize the behavior of the Reader. The commands outlined here are targeted at manipulating data, such as removing characters, translating one set of characters to another, providing different types of output based on the type of barcode read, appending characters, tabs or new lines, etc.

The rules made up of the commands in this document are intended to be fairly short and not overly complicated. If more complex data manipulation, display output, file manipulation, etc. is desired, you should then refer to document C005356: Code Reader Advanced JavaScript Programming Guide as found on the Code Corporation web site.

Table of Contents

1.2 Document and Coding Conventions

This document employs the following conventions to aid in readability:

- Words that are part of the application development description use the `Courier New` font.
- Code examples use the **`bold Courier New`** font.
- Variable names that must be supplied by the programmer are `Courier New` font and are enclosed in relational signs, for example, `<variable_name>`.

The Code Reader JavaScript library uses the following naming conventions:

- identifiers: mixed-case with a capital letter where words join (soCalledCamelCase); acronyms and other initialisms are capitalized like words, e.g., nasaSpaceShuttle, httpServer, codeXml
- variables and properties: initial lower case, e.g., thisIsAVariable, thatIsAProperty
- classes (i.e., constructors): initial capital, e.g., AClassIsCapitalizedCamelCase
- functions: initial lower case – similar to variables and properties.
- unit of measure: suffix to name, separated from name by underscore, using correct case when it's significant, e.g., offset_pixels, width_mm, power_MW, powerRatio_dB

Table of Contents

1.3 Related Documents

Code readers are controlled by a large number of registers that define the behavior of the reader. These registers are defined in the *Code Reader 8000– Reader-Host Interface Specification*, Code Document Number C005066 listed below. You will want to familiarize yourself with the use of registers with the Reader.

Code, *Code Reader 8000– Reader-Host Interface Specification*
Document Number C005066

Code, *Code Reader 8000– JavaScript Advanced Programmers Guide*
Document Number C005356

Note: please visit Code's website at http://www.codecorp.com to obtain these documents.

Table of Contents

1.4 Related Utilities

USB Virtual Com Driver (C002712) – A software driver that creates a virtual COM port for a USB-cabled reader. This enables the reader be used by a computer program that requires input from a serial device while being connected to a USB port.

Reader Download Utility (C002640) – Downloads JavaScript applications and data files from a host PC onto the reader. Valid communication modes are USB Downloader, USB Virtual Com, and RS232.

File Uploader (C002880) – Utility to transfer files from the reader to the host PC. Valid communication modes are RS232 (115 Baud), USB Downloader mode, and USB Virtual Com.

These utilities are available at: http://www.codecorp.com/downloads.html

Table of Contents

1.5 JavaScript Resources

While you will not need to be a JavaScript expert, some JavaScript knowledge is required to use Rules based on this document. This document is not a JavaScript manual. While there are a number of books on JavaScript programming, the following sources provide JavaScript reference books and online documents that you may find useful.

- ***JavaScript: The Definitive Guide***
  by David Flanagan
- ***JavaScript, A Beginner's Guide, Third Edition (Beginner's Guide)***
  by John Pollock
- ***JavaScript Demystified (Demystified)***
  by James Keogh.
- ***JavaScript (TM) in 10 Simple Steps or Less***
  by Arman Danesh.
- ***http://www.w3schools.com/jsref/default.asp***
- ***http://javascript.internet.com/***
- ***http://www.javascript.com/***

Table of Contents

1.6 Regular Expression Resources

You will not need to be a Regular Expression expert, but some Regular Expression knowledge is required to use Rules based on this document. This document is not a Regular Expression manual. While there are a number of books on Regular Expression use, the following sources provide Regular Expression reference books and online documents that you may find useful.

- ***Mastering Regular Expressions***
  by Jeffrey E F Friedl
- ***Beginning Regular Expressions (Programmer to Programmer)***
  by Andrew Watt
- ***http://en.wikipedia.org/wiki/Regular_expression#Basic_concepts***
- ***http://www.regular-expressions.info/***

Table of Contents

2 Programming Environment

Code provides a Windows simulator environment for programming, and testing JavaScript files.

You can use your favorite editing product to create and modify JavaScript code. Turn off any smart quote options in the editor. Smart quotes are not valid in JavaScript.

Code has bundled a freeware editor (SciTE) with the Code Reader JavaScript Engine (JSE) Simulator.

Table of Contents

2.1 Installing and Running the Code Reader JavaScript Engine Simulator on your PC

To install the JavaScript Engine Simulator, it is recommend the file is saved onto the PC or host device prior to running the Simulator.

The Simulator can be downloaded from Code’s website at
http://www.codecorp.com/downloads.php#javascript.

Table of Contents

2.2 Installing and Running the Application on the Code Reader

The JavaScript file that is written to include the JavaScript Rules functions can be loaded directly on the reader using CortexTools or one of the older file uploader utilities.

The file name must conform to the format <rules file identifier><optional dot ‘.’><optional identifier>.js where

<rules file identifier>	is the keyword ‘.rules’
<optional dot ‘.’>	is included if and only if the <optional identifier> is included
<optional identifier>	is any string used by the writer to locally identify the JavaScript Rules file
.js	is the JavaScript file extension

For example, the file names ‘.rules.js’, ‘.rules.C001234.js’, and
‘.rules.stripCodabarStartStopChars.js’ are all valid JavaScript Rules file names

When this format is followed, the firmware recognizes and processes the JavaScript Rules file correctly, determining which types of Rules are included and handling them appropriately.

Table of Contents

2.3 Security

The programming described in this manual cannot be secured. The development described in the Advanced JavaScript Programming Guide can be secured via encryption to a reader serial number or numbers so that intellectual property can be protected.

Table of Contents

3 The rules.js File Format and Use

The ‘rules.js’ file will contain one or more functions with specific names. The code included in these functions will be placed into the code path of the reader and affect the behavior of the reader.

The ‘rules.js’ file should be as sparse as possible with little extra comments, whitespace etc. as possible. The file will be converted into a Data Matrix barcode to be loaded onto the reader – the smaller the file the smaller the barcode. You will notice in the examples that there are no ‘var’ keywords which are optional in JavaScript. If the file becomes large enough the barcode will be split into multiple codes to scan consecutively in order to load the entire file.

4 Predefined Functions

The Code Reader core JavaScript application provides a simple method for handling the most common tasks. This will allow people with minimal programming knowledge to customize the product to their needs using a subset of JavaScript and Regular Expressions. Regular Expressions combined with JavaScript String methods such as .match and .replace become a powerful data manipulator.

There are three predefined functions corresponding to three reader states available to be modified:

- **function rules_onDecode(decode)**
- **function rules_onDecodeAttempt(numDecodes)**
- **function rules_onEvent(eventNum)**

The variable names used in these examples are not fixed, but they provide a simple method of keeping track of the data type provided and can be used in your code as is.

Breaking out of any of these functions by returning a value of ‘false’ will stop further processing of the data. Returning the value ‘false’ from rules_onDecodeAttempt will prevent the on_Decode event (including modifications in rules_onDecode) from being called on a given decode.

Any code written outside the functions in the ‘rules.js’ file will be executed at the startup of the reader and any variables declared outside the functions will be global variables.

Table of Contents

4.1 rules_onDecode

This function will be called once for every bar code that has been successfully read. If more than one barcode is allowed to be decoded in a single decode attempt, this function will be called for *each* successful decode. The decode object (Section 0) is passed into the function and may be returned by the function. If a Boolean false is returned by the function the reader will prevent further processing of the decode. The Code Reader will initially indicate a good read as soon as a bar code is found.

The rules_onDecode function must return either the decode object or the value ‘false’. Returning ‘false’ will stop further action by the reader.onDecode function.

4.1.1 Example rules_onDecode

This example illustrates using regular expression to replace each uppercase letter ‘A’ in the decode string with lowercase letter ‘a’. The properties of the Decode object are described in Section 4.

```
rules_onDecode = function (decode)
{
        decode.data.replace(/A/g, “a”);

        return decode;
};
```

Table of Contents

4.1.2 Example rules_onDecode

This example shows the use of a regular expression that matches a specific pattern of decode data, then the use of a JavaScript method to remove the last digit from the matched decode data.

```
rules_onDecode = function (decode)
{
        if(decode.data.match(/^[0-9]{9}$/g)!= null)
        {
                decode.data = decode.data.substring(0, decode.data.length - 1);
        }

        return decode;
};
```

Table of Contents

4.2 rules_onDecodeAttempt

rules_onDecodeAttempt will be called each time a decode attempt is initiated. The value passed into the function is the number of successful decodes found. By default the value of numDecodes will be either 0 (no decodes found) or 1 (a single decode found). Register 0x34 controls the number of bar codes the Code Reader looks for at one time. To increase the possible simultaneous decodes, increase register 0x34 from the default value of one.

The rules_onDecodeAttempt function may return the value 'false'. Returning 'false' will stop further action by the reader.onDecodeAttempt function.

Table of Contents

4.2.1 Example rules_onDecodeAttempt

This rules.js file will allow a successful decode only if 2 decodes are found in a single decode attempt – meaning that there are two barcodes next to each other on the target document. (programming codes from the manual will still be processed). Setting registers 0x93 and 0x14d to 1 will turn the control of beeps and LED flashes over to the JavaScript. Setting register 0x34 to 2 allows the reader to capture two barcodes in each barcode attempt, if there are two available. The two reader methods cause the reader to beep and flash the green LED respectively

```
reader.writeSetting(0x93, 1);
reader.writeSetting(0x14d,1);
reader.writeSetting(0x34, 2);

rules_onDecodeAttempt = function (numDecodes)
{
        if (numDecodes < 2) {
                return false;  //stops further execution
        }
        else
        {
                reader.beep(1);
                reader.setDisplayLed(reader.green);
        }
};
```

Table of Contents

4.3 rules_onEvent

rules_onEvent will be called whenever a user event number is used. The user event numbers are restricted to 25 through 47. The most common use of this will be to implement some logic based on the press of a button or to send a read failure notification.

Table of Contents

4.3.1 Example rules_onEvent

In the example below, the event register (0x39) is first set to the value of 25. Processing the command '$' with the value 0x04 causes the reader to scan, just as pressing the trigger would by default. See the *Code Reader 8000– Reader-Host Interface Specification*, Code Document Number C005066 for more information. The event register is set to the value of 26 and a keystroke enter key is sent via whatever communication mode the reader is in currently.

```
var enter = "\x01X\x1ean//n\x04";
var reader.writeSetting(0x39, 25);

rules_onEvent = function (numEvent)
{
        if (numEvent == 25)
        {
                reader.processCommand('$', "\x04");
        }

        reader.writeSetting(0x39, 26);

        if (numEvent == 26)
        {
                comm.sendPacket(‘z’, enter);
        }
};
```

Table of Contents

4.4 Optional Global variables

The writer can include global variables in JavaScript Rules file, outside any of the pre-defined functions. These variables will be instantiated when the reader starts – as the JavaScript engine starts. See the example in Section 4.3.1 to see global variables being added to a JavaScript Rules file.

Decode Object Properties

The decode object contains all the information about the decoded bar code. Properties in JavaScript are accessed by using the "." operator. The example in section 4.5 line 5 shows how to access the symbology identifier when processing a decode. Below is a complete list of available properties.

decode – object having the following properties:

data – string; the text decoded from the bar code.

symbology – read-only number; the symbology number (see *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066).

symbologyModifier – read-only number; the symbology modifier number. See *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066.

symbologyIdentifier – read-only string; this is the AIM identifier ("]cm").

x – read-only number; unit is pixels, 0 is center of image.

y – read-only number; unit is pixels, 0 is center of image.

x,y combined specify the position of the center of the bar code in the image (relative to the center of the image; the values can be positive or negative).

time – read-only Date object; a JavaScript Date object indicating the time the code was read.

quality_percent – read-only number; a code quality metric returned by the decoder. The precise meaning is symbology-specific.

qrPosition – read-only number; Only defined if symbology is QR.

qrTotal – read-only number; Only defined if symbology is QR.

qrParity – read-only number; Only defined if symbology is QR.

linkage – read-only number; indicates that a code is one part of a composite code. See *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066.

bounds – 4-element array, indexed from 0 – 3. Each element is a decode.bounds object with 2 properties: x and y, both are integers and read only.

QR Structure Append:

qrTotal – read-only number; total number of symbols in the structured append.

qrPosition – read-only number; the position of the current code in the structured append.

qrParity – read-only number; returns the parity value for the current code in the structured append.

Example:
See the discussion of symbol decoding in section **Error! Reference source not found.**

Table of Contents

4.5 Decode Object Properties Example

The following example shows some of the uses of the Decode Object Properties. The ‘enter’ and ‘tab’ variables are set to pass the enter key and tab key for keystroke output. See Section 7 for more details. The Data Matrix symbology is assigned the value of 31 in the Code firmware. 0x1d is the ASCII value for a Group Separator, 0x1E is the ASCII value for a Record Separator.

The action of the function is as follows: any leading Group Separator followed by a 17v is removed, then any leading Record Separator, Group Separator followed by ‘1P’ or Group Separator followed by ‘s’ are removed. The data string is shorted by 2 characters from the right, and then a keystroke enter is appended to the right side.

```
var enter = "\x01X\x1Ean//n\x04";
var tab   = "\x01X\x1Ean//t\x04";

rules_onDecode = function (decode)
{
        if (decode.symbology == 31)
        {
                decode.data = decode.data.replace(/^.*\x1D17v/gi, "");

                decode.data = decode.data.replace(/\x1E|\x1D1P|\x1Ds/gi, tab);

                decode.data = decode.data.substr(0, decode.data.length - 2);

                decode.data = decode.data + enter;
        }

        return decode;
};
```

Table of Contents

5 JavaScript Extensions

Code has added a number of extensions to the core JavaScript so that the reader hardware can be controlled.

Table of Contents

5.1 beep

The beep method causes the Code Reader to beep.

Format:

```
reader.beep(numBeeps);
```

Where:

numBeeps – number; number of beeps.

Note: This method does not return a value.

Example:

```
reader.beep(3);
```

Causes the reader to beep 3 times

Table of Contents

5.2 defaultSettings

The defaultSettings method resets selected Code Reader settings to manufacturing defaults; it is equivalent to sending the 'J' command using the reader.processCommand method (section 5.3).

Format:

```
reader.defaultSettings();
```

Note: This method has no arguments and no return value.

Code Reader settings are defined in *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066, which also identifies settings that this command does not reset.

Table of Contents

5.3 processCommand

The processCommand method instructs the Code Reader to execute a command.

Format:

result = reader.processCommand(commandType, data);

Where:

commandType – string, 1 character; the command to be processed on the Code Reader.

data – string; data as required to process the command.

result – depending on the command, either:

- a Boolean value
- a data string

For commandType, data, and resulting values, see *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066.

Example:

reader.processCommand('$', "\x03"); // read a code

Sends a “$” command code (post event) with a one-byte value of 3 (event type = read near and far fields) to the Code Reader firmware.

Table of Contents

5.4 readSetting

The readSetting method returns the current value of the specified configuration setting.

Format:

value = reader.readSetting(settingNumber);

Where:

settingNumber – number; integer value representing the setting to be read.

For settingNumber values, see *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066.

Example:

value = reader.readSetting(0x1b);

Returns the current value of the Code Reader setting hex 1b (communications mode).

Table of Contents

5.5 saveSettings

The saveSettings method writes the current values of the Code Reader configuration settings into flash memory. Operational setting values are loaded from flash memory when the Code Reader initializes. Any changed configuration settings will be lost at reader shutdown unless saved in flash memory.

Format:

result = reader.saveSettings();

Where:

result – Boolean; false if the flash write fails; true otherwise.

Note: There are no arguments to this method.

Table of Contents

5.6 sendPacket

The sendPacket method instructs the Code Reader to send a data packet to the host via the communications port currently specified by the active Code Reader communication settings.

The Code Reader creates a packet formatted according to the active Code Reader packet protocol configuration setting.

Format:

result = comm.sendPacket(type, data);

Where:

type – ‘z’ means send a decoded data. If additional packet types are needed, refer to the Advanced programming guide.

data – string; data to be inserted into the packet.

result – Boolean; false if there was a failure on the communications port; otherwise, true. If the current communications mode is ACK/NAK mode, true indicates that the data has been sent to and acknowledged by the host.

Example:

```
result = comm.sendPacket('z', “Hello World”);
//sends “Hello World” as decoded data
```

Table of Contents

5.7 setDisplayLed

The setDisplayLed method activates the LED of the CR3500 above the display.

Format:

reader.setDisplayLed(color);

Where:

color – must be reader.green, reader.red, reader.amber, or reader.none.

Note: Setting 0x014d must be set to false for setDisplayLed to function properly.

Table of Contents

5.8 writeSetting

The writeSetting method changes the operational value of a single Code Reader configuration setting.

Format:

```
writeSetting(settingNumber, value);
```

Where:

settingNumber – number; the setting to be changed.
value – number; the value to be written to the configuration setting.

For the possible values of settingNumber and value, see *Code Reader 8000 – Reader-Host Interface Specification*, Code Document Number C005066.

Note: This method does not return a value.

Example:

```
reader.writeSetting(0x1b, 4);
```

Sets the reader communications mode to Bluetooth RF. See also the gui.Button example in section **Error! Reference source not found.**

Table of Contents

5.9 Properties

This section documents the properties defined for the Code Reader's reader object.

Table of Contents

5.10 hardwareVersion

The hardwareVersion property of the reader object contains a read only string containing the version number of the Code Reader hardware.

Example:

```
hwVersion = reader.hardwareVersion;
```

Table of Contents

5.11 oemId

The oemId property of the reader object contains a read-only string containing the Code Reader unique OEM identifier from the locked flash memory.

Example:

```
oemId = reader.oemId;
```

Table of Contents

5.12 readerId

The readerId property of the reader object contains a read-only string containing the Code Reader unique ID from the locked flash memory.

Example:

```
rid = reader.readerId;
```

Table of Contents

5.13 softwareVersion

The softwareVersion property of the reader object contains a read only string containing the version number of the firmware currently running in the Code Reader.

Example:

```
swVersion = reader.softwareVersion;
```

Table of Contents

6 Sending Keystrokes

The Code Reader products are often connected to a PC using keyboard input. The data contained in the bar code is simply “typed” into the PC application. It is also often required to send a certain key to the application such as an “enter” key. Please note that an “enter” key is not the same as an ASCII carriage return (0x13).

To add an enter suffix you can use the following format where the /n represents the enter key. A full list of available keys are listed below.

```
enter = "\x01X\x1ean//n\x04";
decode.data = decode.data + enter;
```

Characters	Key
/a	Toggle Alt
/c	Toggle Ctrl
/s	Toggle Shift
/w	Toggle Windows Logo
/u	Up arrow
/l	Left arrow
/r	Right arrow
/d	Down arrow
/t	Tab
/z	Delete
/e	Esc
/n	Enter
/v	End
/b	Backspace
/i	Insert
/p	Page up
/x	Page down
/h	Home
/,	500 ms delay
/0 - /9	Number pad
/f1 - /f12	Function keys
//	/

7 Glossary and Acronyms

Term	**Definition**
Code Data	Data resulting from the decode process after data capture or bar code read.
Smart Quote	Previously formatted quotation marks, usually found in a word processing. Program.
Consume	Used with no return value by the user defined application or firmware.

Table of Contents

8 Format Specifiers

The control string of the format function accepts the following codes from the standard C library:

%d	signed decimal integers
%i	signed decimal integers
%e	lowercase scientific notation
%E	uppercase scientific notation
%f	floating point decimal
%g	uses %e or %f , whichever is shorter
%G	uses %E or %f, whichever is shorter
%o	unsigned octal
%s	character string
%u	unsigned decimal integers
%x	lowercase unsigned hexadecimal
%X	uppercase unsigned hexadecimal
%%	insert a percent sign

Flag, width, and precision modifiers are the same as in the standard C library definition.

Table of Contents

9 Supported JavaScript Core

This is a list of JavaScript that is supported in the ‘rules.js’ functions.

Objects, Methods, and Properties

- Array
- Boolean
- Date
- Function
- Math
- Number
- Object
- Packages
- RegExp
- String
- sun

Top-Level Properties and Functions

- decodeURI
- decodeURIComponent
- encodeURI
- encodeURIComponent
- eval
- Infinity
- isFinite
- isNaN
- NaN
- Number
- parseFloat
- parseInt
- String
- undefined
- **Statements**
- break
- const
- continue
- do...while
- export
- for
- for...in
- function
- if...else
- import
- label
- return
- switch
- throw
- try...catch
- var
- while
- with

Operators

- Assignment Operators
- Comparison Operators

Arithmetic Operators

- % (Modulus)
- ++ (Increment)
- -- (Decrement)
- - (Unary Negation)

Bitwise Operators

- Bitwise Logical Operators
- Bitwise Shift Operators

Logical Operators

String Operators

Special Operators

- ?: (Conditional operator)
- , (Comma operator)
- delete
- function
- in
- instanceof
- new
- this
- typeof
- void

10 Symbology Identifier Values

Symbology	ID
EAN_JAN_13	0
EAN_JAN_8	1
UPC_A	2
UPC_E	3
UPC_D1	4
UPC_D2	5
UPC_D3	6
UPC_D4	7
UPC_D5	8
UPC_A_plus2	9
UPC_A_plus5	10
UPC_E_plus2	11
UPC_E_plus5	12
EAN_JAN_8_plus2	13
EAN_JAN_8_plus5	14
EAN_JAN_13_plus2	15
EAN_JAN_13_plus5	16
Interleaved_2_of_5	17
Code39	18
Code128	19
Codebar	20
Code93	21
UCC_EAN_128	22
UPC_A_w_Code_128_Supplemental	23
UPC_E_w_Code_128_Supplemental	24
EAN_JAN_8_w_Code_128_Supplemental	25

Symbology	ID
EAN_JAN_13_w_Code_128_Supplemental	26
Unknown	27
num_IBM_symbologies	28
Australia_Post	29
Aztec	30
DataMatrix	31
Straight_2_of_5_2_Bar_Start_Stop	32
Straight_2_of_5_3_Bar_Start_Stop	33
Japan_Post	34
KIX	35
MSI_Plessey	36
Maxi	37
PDF417	38
PLANET	39
POSTNET	40
QR	41
Royal_Mail_4_State_Customer	42
RSS_Expanded	43
RSS_Expanded_Stacked	44
RSS_Limited	45
RSS_14	46
RSS_14_Stacked	47
GoCode	48
UPC	49
Codablock	50
Code11	51
Pharmacode	52