

Readerboard  
User's Guide  
WORKING  
DRAFT

The information in this document, and the hardware and software it describes, are hobbyist works created as an educational exercise and as a matter of personal interest for recreational purposes.

It is not to be considered an industrial-grade or retail-worthy product. It is assumed that the user has the necessary understanding and skill to use it appropriately. The author makes NO representation as to suitability or fitness for any purpose whatsoever, and disclaims any and all liability or warranty to the full extent permitted by applicable law. It is explicitly not designed for use where the safety of persons, animals, property, or anything of real value depends on the correct operation of the software.

**For readerboard hardware version 1.0.0 and firmware version 0.0.0.**

Copyright © 2023 by Steven L. Willoughby (aka MadScienceZone), Aloha, Oregon, USA. All Rights Reserved. This document is released under the terms and conditions of the Creative Commons “Attribution-NoDerivs 3.0 Unported” license. In summary, you are free to use, reproduce, and redistribute this document provided you give full attribution to its author and do not alter it or create derivative works from it. See

<http://creativecommons.org/licenses/bynd/3.0/>

for the full set of licensing terms.



# CONTENTS

<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>iv</b>
<b>1 Protocol Description</b>	<b>1</b>
Command Termination and Error Handling . . . . .	1
Command Summary . . . . .	2
*—Strobe Lights in Sequence . . . . .	2
<—Scroll Text Across Display . . . . .	3
?—Query Discrete LED Status . . . . .	3
@—Set Column Cursor Position . . . . .	5
A—Select Font . . . . .	5
C—Clear Matrix Display . . . . .	7
F—Flash Lights in Sequence . . . . .	7
H—Draw Bar Graph Data Point . . . . .	8
I—Draw Bitmap Image . . . . .	8
L—Light Multiple LEDs . . . . .	9
Q—Query Readerboard Status . . . . .	10
S—Light Single LED . . . . .	10
T—Display Text . . . . .	10
X—Turn off Discrete LEDs . . . . .	11

## LIST OF FIGURES

## LIST OF TABLES

1.1	Summary of All Commands . . . . .	3
1.2	Control Codes Allowed in String Values . . . . .	4
1.3	ASCII Encoded Integer Values (0–63) . . . . .	5
1.4	Font Codes . . . . .	6
1.5	Font Table for Fonts #0 and #1 . . . . .	6
1.6	Font Table for Font #2 . . . . .	7
1.7	Discrete LED Codes and Colors . . . . .	8
1.8	Transition Effect Codes . . . . .	9

# CHAPTER 1

## PROTOCOL DESCRIPTION

I don't stand on protocol. Just call me  
your Excellency.

—Henry Kissinger

**T**HE CONTROL PROTOCOL used to display information on the readerboard sign is very simple. Commands are expressed largely in plain ASCII characters and are executed immediately as they are received.<sup>1</sup> It is not necessarily required for a command to be fully received first, so it is possible that the sign will have started operating on part of the command (e.g., moving the current cursor column position) even if the rest of the command could not be performed.

In addition to plain, printable 7-bit ASCII characters, a few control codes are recognized as described below. String data may include any 8-bit value except as otherwise indicated.

### Command Termination and Error Handling

The command terminator is “`^D`” (hex byte value 04). We *recommend* that every command—or group of commands which together represent a logical update to the sign—be followed by a `^D` character.

The individual commands are executed immediately upon receipt, without requiring the `^D` byte. For example, in the sequence “H1H2H3`^D`” the first bar graph data point will be displayed as soon as “H1” is received.

---

<sup>1</sup>Technically, they may even be executed *while* they are being received.

In case of an error, such as the inability to parse the incoming command or an invalid field value, the readerboard will signal the error condition by lighting the white and both red discrete LEDs, and will then ignore all input until the next `^D` byte is received.

Whenever a `^D` byte is received, any command being parsed is aborted and the sign’s input parser is reset. Thus, this byte may be sent in case a partial command has been sent but the host becomes aware that it cannot be completed. Note that the command may have been partially acted upon by that point, so no assumption should be made as to the sign’s state.

In the descriptions that follow, a trailing `^D` is shown at the end of each command to remind you of the recommendation to send this terminator, even though the commands themselves do not require them, strictly speaking.

## Command Summary

The eight discrete LEDs are intended for a simple display of status information in a manner analogous to the Busylight project by the same author.<sup>2</sup> To support this usage, the F, S, X, \*, and ? commands are recognized in a manner compatible with how Busylight uses those same commands. These are categorized as “Busylight compatibility commands.” Unlike all other commands listed here, these are recognized regardless of case. Since the readerboard has a power supply capable of illuminating all of the status LEDs at once,<sup>3</sup> a new command L is added which allows any arbitrary pattern of steady LEDs to be turned on.

The remaining commands are used for management of the matrix display. All commands are summarized in Table 1.1.

Those commands listed with “response” in their Notes column send data back to the host as described below. None of the other commands do this.

### \*—Strobe Lights in Sequence

0	1	2	3	...	$n$	$n + 1$	$n + 2$
*	$\langle light_0 \rangle$	$\langle light_1 \rangle$	$\langle light_2 \rangle$	...	$\langle light_n \rangle$	\$	<code>^D</code>

Each  $\langle light \rangle$  value is an ASCII digit character corresponding to a discrete LED as shown in Table 1.7.

This command functions identically to the F command (see below), except that the lights are “strobed” (flashed very briefly with a pause between each light in the sequence).

<sup>2</sup>See [github.com/MadScienceZone/busylight](https://github.com/MadScienceZone/busylight).

<sup>3</sup>The Busylight cannot, since it is powered from the host computer’s USB port.

Command	Description	Notes
*	Strobe LEDs in Sequence	Busylight compatible
?	Query discrete LED status	Busylight compatible, response
F	Flash LEDs in Sequence	Busylight compatible
L	Light one or more LEDs steady	Busylight extension
S	Light one LED steady	Busylight compatible
X	All LEDs off	Busylight compatible
^D	Abort/terminate command	Matrix control
<	Scroll text across display	Matrix control
@	Move current column cursor	Matrix control
A	Select character font	Matrix control
C	Clear matrix display	Matrix control
H	Add histogram/bargraph data point	Matrix control
I	Draw bitmap graphic image	Matrix control
Q	Query matrix display statue	Matrix control, response
T	Display text on display	Matrix control

Table 1.1: Summary of All Commands

## &lt;—Scroll Text Across Display

0	1	2	...
<	<loop>	<string>	ESC    ^D

Displays the text <string> by scrolling it across the display from right to left. If <loop> is “.”, the text is only scrolled once; if it is “L” then it repeatedly scrolls across the screen in an endless loop.

The text is rendered in the current font and may contain any 8-bit bytes except as otherwise noted (but avoiding ASCII control codes is wise to be safe from conflict with future control codes which may be added to the protocol). The string is terminated by an escape character (hex byte 1B), indicated in the protocol description as ESC.

The string may include control codes as listed in Table 1.2.

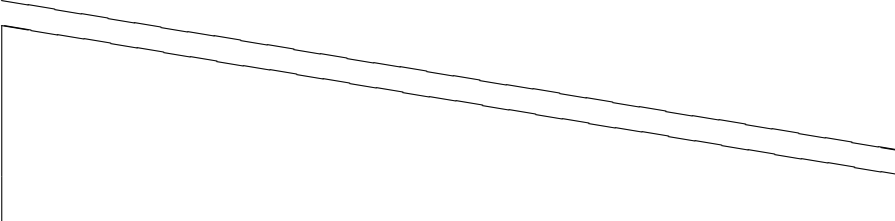
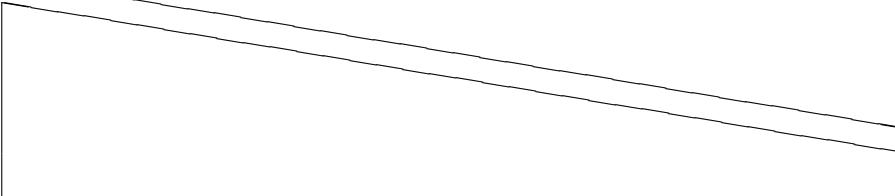
## ?—Query Discrete LED Status

0	1
?	^D

This command causes the sign to send a status report back to the host to indicate what the discrete LEDs are currently showing. This response has the form:

Code	Hex	Description
$\sim C\langle pos \rangle$	03 $\langle pos \rangle$	Move current column cursor to $\langle pos \rangle$
$\sim D$	04	Never allowed in strings (command terminator)
$\sim F\langle digit \rangle$	06 $\langle digit \rangle$	Switch current font
$\sim H\langle pos \rangle$	08 $\langle pos \rangle$	Move cursor left $\langle pos \rangle$ columns
$\sim L\langle pos \rangle$	0C $\langle pos \rangle$	Move cursor right $\langle pos \rangle$ columns
$\sim [$	1B	Never allowed in strings (string terminator)

Table 1.2: Control Codes Allowed in String Values

0	1	2	3	4	5	6	7	8
L	$\langle light_0 \rangle$	$\langle light_1 \rangle$	$\langle light_2 \rangle$	$\langle light_3 \rangle$	$\langle light_4 \rangle$	$\langle light_5 \rangle$	$\langle light_6 \rangle$	$\langle light_7 \rangle$
F	flasher status (see below)							
								
S	strober status (see below)							
								
\n								

The flasher and strober status values are variable-width fields which indicate the state of the flasher (see F command) and strober (see \* command) functions. In each case, if there is no defined sequence, the status field will be:

0	1
$\langle run \rangle$	X

Otherwise, the state of the flasher or strober unit is indicated by:

0	1	2	3	...	$n + 3$
$\langle run \rangle$	$\langle pos \rangle$	@	$\langle light_0 \rangle$	$\langle light_1 \rangle$	... $\langle light_n \rangle$



Value	Code	Value	Code
0–9	0–9	17–42	A–Z
10	:	43	[
11	;	44	\
12	<	45	]
13	=	46	^
14	>	47	_
15	?	48	`
16	@	49–63	a–o

Table 1.3: ASCII Encoded Integer Values (0–63)

In either case,  $\langle run \rangle$  is the ASCII character “0” if the unit is stopped or “1” if it is currently running. If there is a defined sequence,  $\langle pos \rangle$  indicates the 0-origin position within the sequence of the light currently being flashed or strobed, encoded as described in Table 1.3. The  $\langle light \rangle$  values are as given to the F or \* command that set the sequence.

Note that the  $\langle pos \rangle$  value may be the character “X” to indicate the position value of 40, so it is important to distinguish between the field value “ $\langle run \rangle X$ ” vs. “ $\langle run \rangle X @ \langle sequence \rangle$ ”.

The status message sent to the host is terminated by a newline character (hex byte 0A), indicated in the protocol description above as “\n”.

## @—Set Column Cursor Position

0	1	2
@	$\langle pos \rangle$	~D

Sets the column cursor position to the value indicated by  $\langle pos \rangle$ . See Table 1.3.

## A—Select Font

0	1	2
A	$\langle digit \rangle$	~D

Sets the font to use for rendering text with the < and T commands. The font codes for  $\langle digit \rangle$  are listed in Table 1.4. The full text fonts support the printable ASCII characters plus a majority of the Unicode glyphs with codepoints less than 256. See Tables 1.5–1.6 for a complete font glyph listing with codepoint assignments.

Code	Font Description
0	Fixed-width 5×7 matrix plus descenders in 8th row
1	Variable-width version of font 0
2	Bold alphanumerics and special symbols

Table 1.4: Font Codes

	0	1	2	3	4	5	6	7	
00x				move	end		font		0x
01x	back				forw				
02x									1x
03x				esc					
04x		!	"	#	\$	%	&	'	2x
05x	(	)	*	+	,	-	.	/	
06x	0	1	2	3	4	5	6	7	3x
07x	8	9	:	;	<	=	>	?	
10x	@	A	B	C	D	E	F	G	4x
11x	H	I	J	K	L	M	N	O	
12x	P	Q	R	S	T	U	V	W	5x
13x	X	Y	Z	[	\	]	^	_	
14x	'	a	b	c	d	e	f	g	6x
15x	h	i	j	k	l	m	n	o	
16x	p	q	r	s	t	u	v	w	7x
17x	x	y	z	{		}	~	///	
20x	“	”	‘	’	†	‡	...	'	8x
21x	”	!!	—	←	→	↑	↓	≠	
22x	≤	≥	≈	Γ	Δ	Ξ	Π	Σ	9x
23x	Ω	π	ρ	σ	—	‰			
24x		ı	¢	£	¤	¥	¦	§	Ax
25x		©	ª	«	¬	-	®		
26x	°	±	²	³		μ	¶	•	Bx
27x		¹	º	»	¼	½	¾	¿	
30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	Cx
31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Dx
33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	
34x	à	á	â	ã	ä	å	æ	ç	Ex
35x	è	é	ê	ë	ì	í	î	ï	
36x	ð	ñ	ò	ó	ô	õ	ö	÷	Fx
37x	ø	ù	ú	û	ü	ý	þ	ÿ	
	8	9	A	B	C	D	E	F	

Table 1.5: Font Table for Fonts #0 and #1

	0	1	2	3	4	5	6	7	
00x				move	end		font		0x
01x	back				forw				
02x									
03x				esc					1x
04x		!							
05x					,		.		2x
06x	0	1	2	3	4	5	6	7	
07x	8	9							3x
10x	©	A	B	C	D	E	F	G	
11x	H	I	J	K	L	M	N	O	4x
12x	P	Q	R	S	T	U	V	W	
13x	X	Y	Z						5x
14x		←	→	↑	↓	↖	↗	↘	
15x	↙	←	→	↑	↓	■	■		6x
16x	✓	✓	∞	Œ	œ	€	∴	∴	
17x	✕	◀	▶	▲	▼	↕	◆	◇	7x
20x	λ	Θ	Φ	Ψ					
21x									8x
22x									
23x									9x
24x		TS1	TS2	TS3	TS4	TS5	TS6	TS7	
25x	TS8								Ax
	8	9	A	B	C	D	E	F	

TS<n> = Thin space of <n> pixels

Table 1.6: Font Table for Font #2

C—Clear Matrix Display

0	1
C	~D

Clears the matrix display so that no LEDs are illuminated. Does not affect the discrete LEDs.

F—Flash Lights in Sequence

0	1	...	n	n + 1	n + 2		
F	$\langle light_0 \rangle$	$\langle light_1 \rangle$	$\langle light_2 \rangle$	$\cdots$	$\langle light_n \rangle$	\$	$\neg D$

Each <light> value is an ASCII digit character corresponding to a discrete LED as shown in Table 1.7. Note that the assignment of colors to these LEDs is dependent on your particular hardware being assembled that way. As an

Code	Light	Color
0	L <sub>0</sub>	white
1	L <sub>1</sub>	blue
2	L <sub>2</sub>	blue
3	L <sub>3</sub>	red
4	L <sub>4</sub>	red
5	L <sub>5</sub>	yellow
6	L <sub>6</sub>	yellow
7	L <sub>7</sub>	green

Table 1.7: Discrete LED Codes and Colors

open source project, of course, you (or whomever assembled the unit) may choose any color scheme you like when building the board.

Up to 64 *light* codes may be listed. The sign will cycle through the sequence, lighting each specified LED briefly before moving on to the next one. The sequence is repeated forever in a loop until an L, S or X command is received.

If only one *light* is specified, that light will be flashed on and off. Setting an empty sequence (no codes at all) stops the flasher’s operation.

The sequence is terminated by either a dollar-sign (“\$”) character or the escape control character (hex byte 1B), indicated in the protocol diagram above simply as “\$”.

This command may be given in upper- or lower-case (“f” or “F”).

## H—Draw Bar Graph Data Point

0	1	2
H	<i>&lt;n&gt;</i>	~D

This command is used to draw a bar-graph element. Repeating this command causes a scrolling data display which shows a set of data samples over some period of time. The value *<n>* is an ASCII digit character in the range “0”–“8”, and is drawn in the far-right column of the matrix display, as a column of *<n>* lights stacked up from the bottom row (a value of 0 results in no lights, up to 8 which is a full column of eight lights; a value of 9 is treated as if it were 8). All existing matrix data are scrolled left one column.

## I—Draw Bitmap Image

0	1	2	3	4	5	6	7	8
I	<i>⟨merge⟩</i>	<i>⟨pos⟩</i>	<i>⟨trans⟩</i>	<i>⟨coldata<sub>0</sub>⟩</i>		<i>⟨coldata<sub>1</sub>⟩</i>		⋯
<i>⟨coldata<sub>n</sub>⟩</i>		\$	^D					

Code	Transition
.	No transition
>	Scroll in from left
<	Scroll in from right
^	Scroll up from bottom
v	Scroll down from top
L	wipe left
R	wipe right
U	wipe up
D	wipe down
	wipe left and right from middle column
-	wipe up and down from middle row
?	choose a random transition

Table 1.8: Transition Effect Codes

Draws an arbitrary bitmap image onto the matrix display starting at column  $\langle pos \rangle$ , encoded as per Table 1.3. A  $\langle pos \rangle$  value of “~” represents the current column cursor position.

Each column data, from left to right, are given by  $\langle coldata \rangle$  values, each of which is a two-digit ASCII hexadecimal value with the least-significant bit representing the top row of the matrix.

The column data values are terminated by either a dollar-sign (“\$”) character or the escape control character (hex byte 1B), indicated in the protocol diagram above simply as “\$”.

The column cursor is moved to be after the end of the image.

If  $\langle merge \rangle$  is the character “.” then each column’s contents is cleared before setting the pixels as per the  $\langle coldata \rangle$  value. If  $\langle merge \rangle$  is “M” the bits set in  $\langle coldata \rangle$  are added to the lit pixels already in the column.

The  $\langle trans \rangle$  value indicates the transition effect to use when adding the image to the display. See Table 1.8.

## L—Light Multiple LEDs

0	1	...	$n$	$n + 1$	$n + 2$		
L	$\langle light_0 \rangle$	$\langle light_1 \rangle$	$\langle light_2 \rangle$	$\cdots$	$\langle light_n \rangle$	\$	$\wedge D$

This command is identical to the S command (see below), except that multiple discrete LEDs can be specified, all of which are illuminated simultaneously. See Table 1.7. Note that if a strobe sequence is running (via a previous \* command), it remains running.

**Q—Query Readerboard Status**

0	1
Q	~D

This command causes the sign to send a status report back to the host to indicate the general status of the device except for the discrete LED display which may be queried using the ? command. The response has the form:

0	1	2	3	...		
Q	<i>&lt;model&gt;</i>	V	<i>&lt;hwversion&gt;</i>	R	<i>&lt;romversion&gt;</i>	S
<i>&lt;serial&gt;</i>		\n				

The *<model>* field may be “L” for the legacy hardware the author still has lying around (but this shouldn’t be something anyone else would need to see), or “M” for the current 64×8 matrix display hardware.

*<hwversion>* and *<romversion>* indicate the versions, respectively, of the hardware the firmware was compiled to drive, and of the firmware itself. Each of these fields are variable-width and conform to the semantic version standard 2.0.0.<sup>4</sup>

The *<serial>* field is a variable-width string which was set when the firmware was compiled. It should be a unique serial number for the device (although that depends on some effort on the part of the person compiling the firmware to insert that serial number each time). Serial numbers 0–299 are reserved for the original author’s use.

The status message sent to the host is terminated by a newline character (hex byte 0A), indicated in the protocol description above as “\n”.

**S—Light Single LED**

0	1	2
S	<i>&lt;light&gt;</i>	~D

Stops the flasher (cancelling any previous F command) and turns off all discrete LEDs. The single LED indicated by *<light>* is turned on. See Table 1.7. Note that if a strobe sequence is running (via a previous \* command), it remains running.

This command may be given in upper- or lower-case (“s” or “S”).

**T—Display Text**

0	1	2	3	4	...		
T	<i>&lt;merge&gt;</i>	<i>&lt;align&gt;</i>	<i>&lt;trans&gt;</i>	<i>&lt;string&gt;</i>		ESC	~D

<sup>4</sup>See [semver.org](http://semver.org).

Displays the text *⟨string⟩* at the current cursor position. The cursor position is then moved past the text to be ready for the next string to be printed.

The *⟨align⟩* code indicates how the text is aligned on the display (TBD).

The *⟨trans⟩* code specifies the transition effect to be used to add this string to the display as shown in Table 1.8.

The text is rendered in the current font and may contain any 8-bit bytes except as otherwise noted (but avoiding ASCII control codes is wise to be safe from conflict with future control codes which may be added to the protocol). The string is terminated by an escape character (hex byte 1B), indicated in the protocol description as ESC.

The string may include control codes as listed in Table 1.2.

## X—Turn off Discrete LEDs

0	1
X	^D

Turns off the flasher, strober, and all discrete LEDs.

This command may be given in upper- or lower-case (“x” or “X”).