MCS Protocol

1 Revision History

Revision Date	Notes
8/21/2007	Revision 2.0 ● Add support to read and write all formats file ● Add support for full colors in E2000 ● Add support for full fonts in E2000 ● Add support for full display methods in E2000 ● Add support to display bmp, gif and png image ● Add support to display gif animation ● Add support to display temperature ● Add support to display decounter/incounter

	• Add support for Tab control
12/28/2006	Revision 1.10 • Add support to large memory size (exceed 64K bytes) • Add support to long file label up to 8 characters. • Add general response for communication testing. • Add support to checksum for high reliable data transferring.
6/5/2006	Initial Revision

2 Introduction

This document has been developed to allow the users to understand the communication protocol.

The protocol can be used to display text messages, update date and time, and other useful functions.

3 Document Conventions

The following conventions are used throughout this document:

Convention/Symbol	Definition
11	Number in decimal.
$0C	Number in hexadecimal format.
%00110011	Number in binary format.
<STX> or ^B	ASCII control character – in this case it is a Ctrl-B.
“A”	ASCII character (in this case, the letter A – code $41 or 65.

4 Protocol Overview

This protocol can control a wide range LED display from 8x8 pixels to 2048x256 pixels in single-color or bi-color.

The sign itself can contain many different types of “files”. Each file is downloaded to the sign as needed. The maximum file count and a single file size are unlimited, but the total size can not exceed 256K bytes.

4.1 File Label

Each file is named by a file label. Different type of files can use the same file label. File label can be short: a single character, or long: 1-8 characters enclosed by two “$”. For example, “A” is a short label and “$TEXT0001$” is a long label, they are both good file labels, and short label “A” can be write in long label “A”. Text file “A” is initially allocated with size of 256 bytes. Other files must be allocated before using (see special function command code section).

Only these characters can be used for file label:

- Uppercase Letters: “A” to “Z”
- Lowercase Letters: “a” to “z”
- Numbers: “0” to “9”
- Symbols: ~!@#%&_

4.2 Text Files

A text file contains ASCII message data and display control codes to display text. Text files may also include **variable** files as well.

4.3 Variable Files

Variable files are files that contain frequently changing information such as number values. You can easily change these variable files without affecting the text files that contain these. When the sign has received a variable file, the sign will not restart (however the text file will), but keep running as if nothing has happen, and the display will partially updated a few seconds later (the value of the variable changed).

4.4 Special Functions

These are not really files, but a set of codes that setup the sign itself, like setting the time and date.

5 Base Protocol Format

The sign responds to two different types of protocol streams. One that uses the full ASCII set (Binary) and one that “escapes” the non-printable ASCII codes. The last one is extremely useful for PLC’s, as only printable ASCII codes are used.

For the binary format, each code shown (i.e. ^B) is the actual ASCII code that is to be transmitted. So ^B would send a code of $02. **Due to the nature of the printable format below, if you wish to have a “^” in your message, you MUST send “^^”.**

For the printable format, each code shown (i.e. ^B) is the ACTUAL series of codes to send. So ^B would send out two ASCII characters “^” and “B”. **If you need to actually display the “^” character in your message, use “^^”.**

The protocol is flexible enough that you can mix and match codes as desired.

For serial communications, the protocol specifics are ALWAYS 8 data bits, 1 stop bit, and no parity. 9600 baud is the factory default value.

5.1 Standard transmission packet

This is the base transmission packet that is needed for all communications:

Standard Transmission Packet

<STX> **^B**	Sign Address	<SOH> **^A**	Command Code	Data Area	<EOT> **^D**	Checksum	<SOH> **^A**	Command Code	…	<ETX> **^C**

Item	Description
<STX>	Start of transmission. ^B
Sign Address	List of sign address, in hexadecimal format, separated by commas. Each address is 2 ASCII hex digits long. I.E. “01, 0A, 64” is address 1, 10 and 100. The sign will only respond when its address is in this list. Address “00” will cause every sign that is receives this to respond.
<SOH>	Start of command. ^A More than one command can be transferred in one transmission packet by using <SOH> instead of <ETX>. You can restart a new command with <SOH> and need not to match the sign address again. Otherwise if <ETX> is found, the next command must begin with <STX> and sign address should rematch.
Command Code	Command code is a single uppercase letter “A” to “Z”, represents the command to use. Each command is documented in its own section. Some kind of command will cause the sign restart after <ETX>, such as “A” (Write to text file). **Command Codes** **Command Code** — **Description** “A” — Write to text file. “B” — Write to variable file. “C” — Write/read to/from special function.
Data Area	Data area as required for each command. See the appropriate section for each command
<EOT>	End of text. ^D Use <EOT> at the end of each command to append checksum. This is optional for high reliable data transmission.
Checksum	Checksum is optional and should be used with <EOT> together. They are appended at the end of each command to provide high reliable data transferring. The checksum is a 4 hexadecimal digits string representing a hex number “0000” to “FFFF”, which is a word value sum up from <SOH> to <EOT> (inclusive, byte by byte). If <EOT> and Checksum exist, the sign will compare the value with the SUM of bytes actually received. If a bad checksum is checked, the sign will ignore the command to protect the sign from accident damage.
<ETX>	End of transmission. ^C Use <ETX> to end all transmission or use <SOH> to start a new command.

5.2 General response

When the sign receive a packet ended with <ETX>, it responses some messages to tell communication succeeded or an error occurred.

The sign do not response on these circumstances:

- The packet is broadcasting to all signs with zero address.
- The packet is sent to a group of signs with more than one address in the address list.

The following information may be responded:

- **ok**
  Communication is good and all commands are handled successfully.
- **unknown command code**
  An unknown command code is found. Maybe you should upgrade the software on the sign.
- **bad checksum**
  A checksum is provided but the checksum is not equal to the calculated value.
- **invalid file label**
  The file label includes invalid characters.
- **invalid file size**
  You should provide a 1-8 hex digits file size when you allocate memory for a file.
- **not enough memory**
  There is not enough memory and the memory size allocated for this file does not change.
- **file does not exist**
  You are trying to write to a file which has no memory allocated.
- **file out of allocated size**
  More file data than the allocated memory size are received. The file data is truncated.
- **invalid hexadecimal number**
  A valid hex number should make up of hex digits “0” to “9”, “A” to “F” or “a” to “f”.
- **invalid decimal number**
  A valid dec number should make up of dec digits “0” to “9”.
- **invalid time format**
  A good time format is HHMMSS where HH is hour “00” to “23”, and MM is minute “00” to “59”, and SS is second “00” to “59”.
- **invalid date format**
  A good date format is MMDDYYYYX where MM is month “01” to “12”, and DD is day “01” to “31”, and YYYY is year “2000” to “2099”, and X is day of the week “0” to “6”.
- **unknown beep method**
  The beep method is not currently supported. Maybe you should upgrade the software on the sign.
- **invalid address**
  A valid address should be hex digits “01” to “FF”.
- **unknown error**
  An error occurred but the reason is unknown. The software on the sign needs to modify to avoid this information.

6 Command Code Sections

This area of the document describes each command code that is used and what the data area must consist of.

6.1 Write to Text File – Code "A"

ASCII messages along with the codes to display them are stored in text files. Text files MUST be allocated (using the special function command) before they can be used. When the sign is first used, a single text file is automatically allocated – it is labeled "A" and has a size of 64K bytes.

There are a few items to note when transmitting text files:

- The display will continue running without disturbance during communication. Once the sign receives a valid text file, it will reallocate memory for the file according to the last "Set Memory" command, clear the file content first, and then copy the new file content.
- This command requires the sign restart after <ETX>. To keep the sign running without restart, use variable file please.
- In addition to containing text, text files can contain other files, specifically variable files. See write to variable file section for further details.
- The message in the file is a set of pairs of mode fields and data to display. Further details below.

Write To Text File – Command Code "A" – Data Area

	Repeat as needed for each message			
File Label	Mode Field (Optional)			ASCII Message
1..10 ASCII Character	<BEL> ^G	Display Position 1 ASCII Character	Mode Code 1 ASCII Character	1..N ASCII Characters

Item	Description
File Label	A file label (i.e. "A" or "$TEXT0001$")
Mode Field (Optional)	Set of 3 characters (optional) to define the position and effect to use for the display of message following it.

Mode Field

Code	Description
<BEL>	Start of mode field. ^G. You can use ^G to set the display position and display effect at one time, or use ^P to set the display position and use ^E to set the display effect separately. ^P and ^E give more advanced features for expert

	usage.
Display Position	Single ASCII character defining the line position on a multi-line sign. If a single-line sign is used, this character is ignored but must be present.

Position Codes

Code	Description
"M" $4D	Middle line – text centered vertically.
"T" $54	Top Line - Text begins on the top line of the sign and the sign will use all its lines minus 1 in order to display the text. For example, a 6-line sign will allow a maximum of 5 lines (6 minus 1) for the Top Position. The Top/Bottom Line break will remain fixed until the next Middle or Fill position is specified.
"B" $42	Bottom Line - The starting position of the Bottom Line(s) immediately follows the last line of the Top Line. For example, a 6-line sign with 3 lines of text associated with the Top Line would start the Bottom Line text on the 4th line of the sign.
"F" $46	Fill – The sign will fill all available lines, centering them vertically.
"L" $4C	Left - Text begins on the left side of the sign and the sign will use all its lines minus 1 in order to display the text
"R" $52	Right - Text begins on the right side of the sign and the sign will use all its lines minus 1 in order to display the text

Mode Code

Mode Codes:

Code	Name	Description
"S" $53	Scroll	Message scrolls right to left.
"H" $48	Hold	Message displays stationary.
"F" $46	Flash	Message displays stationary and flashes.
"A" $41	Slide Up	Previous message is slide up by new message.
"B" $42	Slide Down	Previous message is slide down by new message.
"C" $43	Slide Left	Previous message is slide left by new message.
"D" $44	Slide Right	Previous message is slide right by new message.

"a" $61	Roll Up	Previous message is rolled up by new message.
"b" $62	Roll Down	Previous message is rolled down by new message.
"c" $63	Roll Left	Previous message is rolled left by new message.
"d" $64	Roll Right	Previous message is rolled right by new message.

ASCII Message

Message to display. Can contain various codes (listed below) to affect the color, font, speed, pause, embed dates and times, and display variable files. It also can contain codes to set new lines and new pages to display.

NOTE: Most of the codes have a default – the default is used at the start of the message when displaying. If changed, subsequently, the message will use the new changes until the end of the message is reached. Once the message cycles and starts over, the defaults are reset.

Message Codes

Code	Description
^E $05	Set display effect. Other than ^G, ^E can set both single character and multi-characters mode codes.

a. Single character mode codes:

Code	Name	Description
"S" $53	Scroll	Message scrolls right to left.
"H" $48	Hold	Message displays stationary.
"F" $46	Flash	Message displays stationary and flashes.
"A" $41	Slide Up	Previous message is slide up by new message.
"B" $42	Slide Down	Previous message is slide down by new message.
"C" $43	Slide Left	Previous message is slide left by new message.
"D" $44	Slide Right	Previous message is slide right by new message.
"a" $61	Roll Up	Previous message is rolled up by new message.

"b" $62	Roll Down	Previous message is rolled down by new message.
"c" $63	Roll Left	Previous message is rolled left by new message.
"d" $64	Roll Right	Previous message is rolled right by new message.

b. Multi-characters mode codes:

Code	Name	Description
"AUT"	Auto	Randomly choose a display method.
"HLD"	Hold	Message display stationary.
"SLU"	Slide Up	Previous message is slide up by new message.
"SLD"	Slide Down	Previous message is slide down by new message.
"SLL"	Slide Left	Previous message is slide left by new message.
"SLR"	Slide Right	Previous message is slide right by new message.
"SFC"	Slide From Center	Previous message is slide from center by new message.
"STC"	Slide To Center	Previous message is slide to center by new message.
"CVU"	Cover Up	Previous message is covered from bottom by new message.
"CVD"	Cover Down	Previous message is covered from top by new message.
"CVL"	Cover Left	Previous message is covered from right by new message.
"CVR"	Cover Right	Previous message is covered from left by new message.
"CFC"	Cover From Center	Previous message is covered from center by new message.
"CTC"	Cover To Center	Previous message is covered to center by new message.
"ROU"	Roll Up	Previous message is rolled up by new message.
"ROD"	Roll Down	Previous message is rolled down by new message.
"ROL"	Roll Left	Previous message is rolled left by new message.
"ROR"	Roll Right	Previous message is rolled right by new message.
"RFC"	Roll From	Previous message is rolled from center by

	Center	new message.
“RTC”	Roll To Center	Previous message is rolled to center by new message.
“$INS1$”	Inter-Slide 1	Message slide in with interlaced mode 1.
“$INS2$”	Inter-Slide 2	Message slide in with interlaced mode 2.
“$INS3$”	Inter-Slide 3	Message slide in with interlaced mode 3.
“$INS4$”	Inter-Slide 4	Message slide in with interlaced mode 4.
“$INR1$”	Inter-Roll 1	Message roll in with interlaced mode 1.
“$INR2$”	Inter-Roll 2	Message roll in with interlaced mode 2.
“$INR3$”	Inter-Roll 3	Message roll in with interlaced mode 3.
“$INR4$”	Inter-Roll 4	Message roll in with interlaced mode 4.
“$INR5$”	Inter-Roll 5	Message roll in with interlaced mode 5.
“$INR6$”	Inter-Roll 6	Message roll in with interlaced mode 6.
“$SHU1$”	Shutter 1	Message display like shutter using method 1.
“$SHU2$”	Shutter 2	Message display like shutter using method 2.
“$SHU3$”	Shutter 3	Message display like shutter using method 3.
“$SHU4$”	Shutter 4	Message display like shutter using method 4.
“JMP”	Jump	Message blocks jump to the screen.
“SNO”	Snow	Message displays like snow fall.
“RAN”	Random	Message pixels displays on the screen in random order.
“SHO”	Shoot	Messages shoot on the screen.
“EXP”	Explode	Message blocks explode on the screen.
“FLS”	Flash	Message display stationary and flash.
“TWK”	Twinkle	Message display stationary and twinkle.
“PAC”	Pac Man	Previous message is eat by a big mouse and new message is drop down.
“SCL”	Scroll Left	Message continually scrolls from right to left.
“SCU”	Scroll Up	Message continually scrolls from bottom to top.

^F
$06

Set font.

a. Set font by index: followed by one of the following codes to change the font:

- “0” – SS7 (default)
- “1” – SF7
- “2” – SF10

		● "3" – SS16 ● "4" – SF16 **b. Set font by name:** $FONTNAME$ ● "$SS5$" – SS5 ● "$SS7$" – SS7 ● "$SF7$" – SF7 ● "$SF10$" – SF10 ● "$SS16$" – SS16 ● "$SF16$" – SF16 ● "$TM16$" – TM16 ● "$AR16$" – AR16 ● "SMA" – SMA NOTE: the default font SS7 is used when the font does not exist.
	^H $08	Set character attribute (flash, wide, bold). Followed by one of the codes: **Character attribute code:** ● "0" – Set flashing off. (default) ● "1" – Set flashing on. ● "2" – Set wide off. (default) ● "3" – Set wide on. ● "4" – Set bold off. (default) ● "5" – Set bold on.
	^I $09	Set speed. Followed by one speed ASCII character "1" to "8" for eight different speeds. (default="3")
	^J $0A	Set pause. Followed by 2 hexadecimal ASCII digits representing the amount of pause time, in seconds. When used for scrolling right to left messages – it will immediately pause and wait for X seconds. When used for any other mode – the pause is used for the page, before going to the next page or message. Setting this to "00" will set no pauses for each of the pages. Once this is set for non-scrolled pages/message, it is subsequently used for every page during message display. (default="02")
	^K $0B	Display time/date. This will embed the current time and date. Followed by 2 ASCII characters. The first ASCII character describes how to display. **Time format code:** ● "0" – Do NOT show leading zeroes. ● "1" – Show leading zeroes. ● "2" – Show leading zeroes as spaces. ● "5" – Show as ALL CAPS. ● "6" – Show as lowercase. ● "7" – Show as First-Letter Caps. The second ASCII character describes what to display. **Time element code:**

		● "0" = Numeric day ● "1" = Numeric month ● "2" = Numeric year (last 2 digits only) ● "3" = Numeric year (all four digits) ● "4" = Month Abbreviation name. ● "5" = Month full name. ● "6" = Day of the week abbreviation. ● "7" = Day of the week full name. ● "8" = Hour in 12-hour mode. ● "9" = Hour in 24-hour mode. ● "A" = Minute ● "B" = Seconds ● "C" = AM/PM as a single character A/P ● "D" = AM/PM as two characters AM/PM ● "E" = Suffix of the day, like 'st', 'nd', 'rd', or 'th'
	^L $0C	New page – starts a new display page (based on mode, etc).
	^M $0D	New line – starts a new line for multi-line displays.
	^N $0E	Embed variable file. Followed by a file label (i.e. "A" or "$VAR01$") representing the variable file.
	^O $0F	Change color. Followed by a single ASCII character. **a. Change color by index:** followed by a single ASCII character representing the color to change to: ● "0" = Red (default) ● "1" = Green ● "2" = Yellow/Amber ● "3" = Rainbow 1 **b. Change color by name:** $COLORNAME$ ● "ACL" = Auto Color ● "RED" = Red ● "GRN" = Green ● "YEL" = Yellow ● "$RB1$" = Rainbow 1 ● "$RB2$" = Rainbow 2 ● "$RB3$" = Rainbow 3 ● "$RB4$" = Rainbow 4 ● "$RB5$" = Rainbow 5 ● "$MIX1$" = Mixture 1 ● "$MIX2$" = Mixture 2 ● "$MIX3$" = Mixture 3 ● "$MIX4$" = Mixture 4 ● "$INV1$" = Invert 1 ● "$INV2$" = Invert 2

- "$INV3$" = Invert 3
- "$INV4$" = Invert 4
- "$INV5$" = Invert 5
- "$INV6$" = Invert 6
- "$INV7$" = Invert 7
- "$INV8$" = Invert 8
- "$INV9$" = Invert 9

NOTE: the default color RED is used when the color does not exist.

^P
$10

Set display position. Other than ^G, ^P can set both single character and multi-characters position codes.

a. Single character position codes:

Code	Description
"M" $4D	Middle line – text centered vertically.
"T" $54	Top Line - Text begins on the top line of the sign and the sign will use all its lines minus 1 in order to display the text. For example, a 6-line sign will allow a maximum of 5 lines (6 minus 1) for the Top Position. The Top/Bottom Line break will remain fixed until the next Middle or Fill position is specified.
"B" $42	Bottom Line - The starting position of the Bottom Line(s) immediately follows the last line of the Top Line. For example, a 6-line sign with 3 lines of text associated with the Top Line would start the Bottom Line text on the 4th line of the sign.
"F" $46	Fill – The sign will fill all available lines, centering them vertically.
"L" $4C	Left - Text begins on the left side of the sign and the sign will use all its lines minus 1 in order to display the text
"R" $52	Right - Text begins on the right side of the sign and the sign will use all its lines minus 1 in order to display the text

b. Multi-characters position codes:

Format: $XXXX,YYYY,WWWW,HHHH,A$ where

- XXXX = x coordinate, 1..4 decimal digits
- YYYY = y coordinate, 1..4 decimal digits
- WWWW = width, 1..4 decimal digits
- HHHH = height, 1..4 decimal digits
- A = text alignment, 1 ASCII character
  - "0" = left, top
  - "1" = center, top
  - "2" = right, top

- “3” = left, middle
- “4” = center, middle (default)
- “5” = right, middle
- “6” = left, bottom
- “7” = center, bottom
- “8” = right, bottom

NOTE 1: the ^Q control will affect the x and y coordinates of display position.

NOTE 2: using the expression with symbol # can specify a relational value to the full screen size. Symbol # represent the value of screen width when setting the window’s left and width, also represent the value of screen height when setting the window’s top and height.

Examples:
^P$0, 0, 128, 16, 4$ set the display position to the rectangle area (0, 0, 128, 16), and set the text alignment to center-middle.

For a 128x32 sign, **^P$32, 16, (#-32)/2, #, 3$** set the display position to the rectangle (32, 16, 48, 16), and set the text alignment to left-middle. The width (#-32)/2 = (128-32)/2 = 48, and height # using the maximum spacing, that is 32-16 = 16.

For a 128x32 sign, **^P$(#/4)*3, 0, #/4, #, 0$** set the display position to the rectangle (96, 0, 32, 32), and set the text alignment to left-top. The x coordinate is (#/4)*3 = (128/4)*3 = 96, width is #/4 = 128/4 = 32, and height # using the maximum spacing, that is 32.

^Q
$11

Set coordinate reference. Followed by a single ASCII character.

Coordinate reference code:

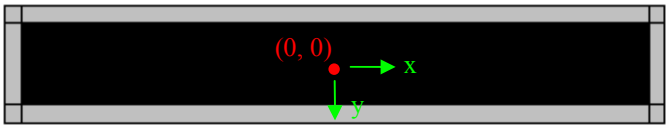
- “0” (default)
- 

- “1”
- 

- “2”

- 

- "3"
- 

- "4"
- 

Examples:

For a 128x32 sign, **^Q1^P$0, 0, 128, 16, 4$** set the display position to the rectangle (0, 16, 128, 16).

^R $12

Display embedded object such as temperature, decounter, incounter etc. Followed by two ASCII characters representing the object to embed:

Temperature

Format: followed by a single ASCII character.

- "0" = Centigrade temperature, such as 20°C
- "1" = Fahrenheit temperature, such as 68°F
- "2" = Kelvin temperature, such as 293K

The temperature will display "??°C" or "??°F" or "???K" when the temperature sensor is not installed.

Decounter

Format: $XX, MM-DD-YYYY[HH:NN:SS]$ where

XX is:

- "10" = decounter, count in days
- "11" = decounter, count in hours
- "12" = decounter, count in minutes
- "13" = decounter, count in seconds

MM is month "1" to "12".

DD is day "1" to "31".

YYYY is year such as "2000".

HH:NN:SS is optional, treat as "00:00:00" if absent.

HH is hour "00" to "23".

NN is minute "00" to "59".

SS is second "00" to "59".

			The decounter can display number within 99999999, a larger number will be displayed as '--------'. 0 will be displayed when the time passed. **Incounter** Format: $XX, MM-DD-YYYY[HH:NN:SS]$ where XX is: ● "20" = incounter, count in days ● "21" = incounter, count in hours ● "22" = incounter, count in minutes ● "23" = incounter, count in seconds MM is month "1" to "12". DD is day "1" to "31". YYYY is year such as "2000". HH:NN:SS is optional, treat as "00:00:00" if absent. HH is hour "00" to "23". NN is minute "00" to "59". SS is second "00" to "59". The incounter can display number within 99999999, a larger number will be displayed as '--------'. 0 will be displayed when the time is not reached. **Examples:** ^R0 Centigrade temperature, such as 20°C ^R$10, 1/1/2008$ display days count to 1/1/2008 00:00:00 ^R$20, 1/1/2000 00:00:00$ display days count from 1/1/2000 00:00:00
		^S $13	Display embedded object from file. The file can be a text file, image file or animation file. **Format 1:** Followed by a file label (i.e. "A" or "0001") representing the *.bmp image file. **Format 2:** $NAME.EXT$ where NAME is 1-8 characters file name, and EXT is 1-3 characters file extension. Both NAME and EXT are case sensitive. The EXT must be the following values: ● "txt" = Include another text file ● "bmp" = embedded a *.bmp image ● "gif" = embedded a *.gif image or animation ● "png" = embedded a *.png image **Include another text file** Unlike the variable file, the current text stop running and text attributes (such as font, color) are saved, and the included text will run. At last, the text attributes restore and the previous text continue.

		The inclusion depth is limit to 20, cross inclusion is inhibited. NOTE: The default file type is *.bmp image, so “^SA” is equal to “^S$A.bmp$”, and “^S0001” is equal to “^S$0001.bmp$”.
	^T $14	Tab control which makes the subsequence text align to “grids”. Followed by a single ASCII character: ● “0” = left aligned Tab control ● “1” = right aligned Tab control ● “2” = center aligned Tab control ● “3” = radix point aligned Tab control NOTE: the default Tab step size is 32 pixels.
	^U	Other settings such as text alignment. **Set text alignment** Format: followed by a single ASCII character ● “0” = left, top ● “1” = center, top ● “2” = right, top ● “3” = left, middle ● “4” = center, middle (default) ● “5” = right, middle ● “6” = left, bottom ● “7” = center, bottom ● “8” = right, bottom **Set Tab step size** Format: $1, TABSTEP$ Where TABSTEP is 1-4 decimal characters. Default value is “32”. **Set horizontal spacing between characters** Format: $2, HSPACE$ Where HSPACE is one character “0”-“9”. Default value is “0”. **Set vertical spacing between lines** Format: $3, VSPACE$ Where VSPACE is one character “0”-“9”. Default value is “0”. **Enable/disable Word-Wrap** Format: $4, WORDWRAP$ Where WORDWRAP is one character: “0” = Disable; “1” = Enable. When disabled, a word may be divided and display in two lines. Default value is “1”.

		Enable/disable word space compressing Format: $5, WORDCMPR$ Where WORDCMPR is one character: “0” = Disable; “1” = Enable. When enabled, the word spacing may be compressed to fit one more word in the line. Default value is “1”. **Enable/disable word space expanding** Format: $6, WORDEXPD$ Where WORDEXPD is one character: “0” = Disable; “1” = Enable. When enabled, the word spacing will expand to fill the whole line. Default value is “1”.
	^V	**Beep** Format: followed by a single ASCII character. ● “1” = [BEEP1] ● “2” = [BEEP2] ● “3” = [BEEP3] ● “4” = [BEEP4]

6.2 Write to Variable File – Code “B”

Variables files are used to store frequently changing information, such as measurements, short pieces of text, and other ASCII text/numeric values.

When writing to a variable file, the sign need NOT to restart. Once the sign receives a variable file, it will reallocate memory for the file according to the last “Set Memory” command, clear the file content first, and then copy the new file content.

Before writing to a variable file, the file must be setup using the special function command to allocate memory for the file. The maximum size of a variable file is unlimited.

Variable files can only be displayed by embedding codes for them in a text file. Anytime the text file goes to show the variable file, it will pull out the last data sent to that variable file. That way, you can continuously update the variable file without affecting the running of the current text file.

Variable files do NOT have any mode options and simply contain the ASCII message to display. They are allowed some of the simple embedded codes to change the fonts/colors, etc.

Steps for using variable files:

1. Allocate memory in the sign for the variable file and the text file that embeds it. [Use the Set Memory Special function to do this.

2. Write the text file that has the embedded variable file code in it.
3. Update the variable file as much as needed to change the data on the display.

Write To Variable File – Command Code "B" – Data Area

File Label 1..10 ASCII Character	ASCII Message 1..N ASCII characters

Item	Description
File Label	A file label (i.e. "A" or "$VAR0001$")
ASCII Message	Message to display. Can contain some of the codes that are used for text files. Please refer to the text file section for details on what each of the codes will do.

Valid Message Codes for Variable File

Code	Description
^F $06	Set Font
^H $08	Set character attribute
^I $09	Set speed
^J $0A	Set pause.
^K $0B	Time/Date
^M $0D	New Line
^O $0F	Set color.

6.3 Write/Read to Special Function – Code "C"

This protocol packet will allow you to set certain functions, including setting up memory, setting the date and time, etc.

In addition, some of the commands respond with data as needed. This response is sent before "General Response" and is always in the ASCII printable protocol stream as is as follows:

Standard Response Packet

<STX> **^B**	Sign Address	<SOH> **^A**	Command Code "C"	Special Function Code	Special Function Response Data	<EOT> **^D**	Checksum	<ETX> **^C**

Item	Description
<STX>	Start of transmission. ^B
Sign Address	Sign address of the sign that is responding. 2 ASCII hexadecimal digits.
<SOH>	Start of command. ^A
Command Code	A single ASCII character representing the command response code. In this case, “C”
Special Function Code	Original 2-ASCII character special function request code.
Special Function Data	0 to N characters of special function response data. Depends on the original request.
<EOT>	End of text. ^D
Checksum	4 hex digits represent a hex word value from “0000” to “FFFF”, which is the SUM of bytes from <SOH> to <EOT> (inclusive, byte by byte).
<ETX>	End of transmission. ^C

Write To Special Function – Command Code “C” – Data Area

Special Function Code 2 ASCII Characters	Special Function Data 0..N ASCII Characters

Item	Description
Special Function Code and Data	Code consisting of 2 ASCII characters plus an additional 0 to N characters for the data. Each code is described below and the data that is required. **Special Function Codes – Write Only, No response**

Code	Description
“ST”	Set time. **Set time format:** HHMMSS where: ● HH = Hour (decimal), “00” to “23” ● MM = Minute (decimal), “00” to “59” ● SS = Second (decimal), “00” to “59”. A separator “:” is optional between HH, MM and SS like HH: MM: SS. The sign’s time will not change until <ETX> is received. You can use <SOH> to continue a “Set Date” command.
“CM”	Clear entire memory. This will clear the entire memory and reset to factory default. This command will set the memory to one text file (“A”) and no variable files.

		This command will NOT reset the address. This command requires the sign to restart after <ETX>.
	"SM"	Set memory. This is used to set the file memory size. Followed by sets of ASCII characters as follows: **Set memory format:** FTSSSS where: ● F = File Label (i.e. "A" or "$TEXT0001$") ● T = File type, currently: ○ "T" = Text file. ○ "V" = Variable file. ● SSSS = Size, 1-8 hexadecimal ASCII digits "0" to "7FFFFFFF", representing the size, in bytes, of the file to allocate. Set to "0" to remove the file from the list. NOTE: This command only affects the next "Write to Text/Variable File" command. The file's content will not be changed until the "Write to Text/Variable File" command has been received. To continue configuring the memory, using <SOH> (^A or $01) and repeat this command.
	"SD"	Set date. **Set date format:** MMDDYYYYX where: ● MM = Month (decimal), "01" to "12" ● DD = Day (decimal), "01" to "31" ● YYYY = Year (decimal) – 4 digits, "2000" to "2099". ● X = Day of the week, where: "0"=Sunday to "6"=Saturday. A separator "/" or "," is optional between MM, DD, YYYY and X like MM/DD/YYYY, X. The sign's date will not change until <ETX> is received. You can use <SOH> to continue a "Set Time" command.
	"SR"	Set a run sequence. Followed by 1 to N characters representing text file labels that are to be displayed in order. This code immediately starts the order with the first text file label given. Once the last file is displayed, the order starts over again. Transmitting any text file to the sign will stop the run sequence (variable files do not do this). Only text file labels can be used. A separator " " or "," is optional between file labels. A good sequence example is "ABC$TEXT0001$$TEXT0002$". This command requires the sign to restart after <ETX>.
	"SB"	Set beep. Beeps the internal speaker. Followed by 1 ASCII character as follows:

		Beep code: ● "0" = Beep continuously for 1 second ● "1" = Beep continuously for 2 seconds. ● "2" = Beep on/off quickly for 2 seconds. ● "3" = A short beep
	"SA"	Set sign address. Followed by the new sign address that take effect immediately: **Set address format:** AA where AA is two ASCII hexadecimal digits representing the new address ("01" to "FF"). The default address from the factory is "01".
	"PF"	Turn power off
	"PO"	Turn power on
	"PR"	Explicitly require the sign to restart after <ETX>. Followed by a single ASCII character: ● "0" = Restart and show startup screen. ● "1" = Restart but do not show startup screen.
	"FM"	Allocate memory for any format file. Format: NAME.EXT=SIZE where: NAME is 1-8 characters file name, and EXT is 1-3 characters file extension. Both NAME and EXT are case sensitive. The EXT can be the following values: ● "txt" = Text file ● "var" = Variable file ● "bmp" = *.bmp image ● "gif" = *.gif image or animation ● "png" = *.png image SIZE is 1-8 hex digits "0" to "7FFFFFFF" representing the size, in bytes, of the file to allocate. Set to "0" to remove the file from the list. The "FM" command is similar to "SM" command, but can allocate memory for all format files.
	"FW"	Write to any format file include text file, variable file, image file, animation file etc. **Write file format:** NAME.EXT=CONTENT where: NAME is 1-8 characters file name, and EXT is 1-3 characters file extension. Both NAME and EXT are case sensitive. The EXT can be the

	following values: ● "txt" = Text file ● "var" = Variable file ● "bmp" = *.bmp image ● "gif" = *.gif image or animation ● "png" = *.png image CONTENT is 1-N characters file content, must be encoded in Base64. User should use this command instead of "A" and "B" to write a text or variable file which contains 8-bits characters in 7-bits system.

Special Function Codes – Read request with response.

Code	Description
"RT"	Read time of day. **Read time response format:** HHMMSS where HH is the hours (in 24-hour mode, decimal) and MM is the minutes (decimal), and SS is the seconds (decimal).
"RM"	Read memory status. **Read memory response format:** UUUU-FFFF:FTSSSS[,FTSSSS,…] where: ● UUUU is the amount of used memory overall in bytes. 1-8 ASCII hexadecimal digits "0" to "7FFFFFFF". ● FFFF is the amount of free space left in sign, in bytes. 1-8 ASCII hexadecimal digits "0" to "7FFFFFFF". ● The following is repeated for each file that is allocated: ○ F = File label (as described in set memory) ○ T = File type (as described in set memory) ○ SSSS = Allocation size. 1-8 ASCII hexadecimal digits. "1" to "7FFFFFFF". ○ Followed by a "," except the last one.
"RV"	Read sign size and versions. **Read version response format:** WWWWHHHHCVV where: ● WWWW = Width of sign, in dots, 4 hexadecimal ASCII digits. ● HHHH = Height of sign, in dots, 4 hexadecimal ASCII digits. ● C = Color type of sign as follows: ○ "1" = Single color sign. ○ "2" = Bi-color (Red,Green,Amber) sign. ○ "3" = RGB sign. ● VV = Protocol version, 2 decimal ASCII digits. Currently

		responds “02” for this protocol.
	“RD”	Read date. **Read date response format:** MMDDYYYYX where: ● MM = Month (decimal) , “01” to “12” ● DD = Day (decimal) , “01” to “31” ● YYYY = Year (decimal) – 4 digits., “2000” to “2099” ● X = Day of the week, where: “0”=Sunday to “6”=Saturday.
	“RA”	Read sign address. The response format is “AA” where AA is 2 ASCII hexadecimal digits representing the address (“01” to “FF”). Note: This command is useful for “pinging” the display or when used with the address of “00” to find out the address of a sign.
	“FR”	Read file content. Format: NAME.EXT where: NAME is 1-8 characters file name, and EXT is 1-3 characters file extension. Both NAME and EXT are case sensitive. The EXT can be the following values: ● “txt” = Text file ● “var” = Variable file ● “bmp” = *.bmp image ● “gif” = *.gif image or animation ● “png” = *.png image The response data is 1-N characters file content, encoded in Base64.
	“FL”	List files. Format: NAME.EXT where NAME is 1-8 characters file name and EXT is 1-3 characters file extension. Both NAME and EXT are case sensitive. Character “*” and “?” can be used for wide range matching. For example, “FL*.*” will list all files, and “FLA.txt” will tell you some information about the text file A. The response format is =FILE; FILE; FILE; …; FILE Files are separated by semicolon “;”. Each file is in format NAME.EXT, SIZE, TIME, ATTR where: NAME is 1-8 characters file name. EXT is 1-3 characters file extension.

			SIZE is 1-8 ASCII hexadecimal digits “1” to “7FFFFFFF” representing the file size. TIME is in format MM-DD-YYYY HH:MM:SS, representing the last modified time. ATTR is 1-N characters string representing the file attributes. This can contain any combinations of the following characters: ● “S” = System file ● “R” = Read only ● “H” = Hidden

7 Multiple Line Sign Behavior

This section identifies the behavior of the signs when using multiple line displays and the protocol.

Looking at the mode when writing text files, the positions are:

- Middle
- Top
- Bottom
- Fill
- Left
- Right

Normally a single line will behave as follows:

- All characters line up at the bottom of the sign and work their way up for as many dots as the font supports:
- If a sign receives a font that is larger than the sign can display, then the sign will “size down” or reduce the font size. For example, on a one-line sign, SS16 characters would be replaced by SS7 characters.
- If a character font is not specified, then SS7 will be used.
- If Top, Bottom, or Fill positions are received Middle is used.
- The centerline is never placed further left than 8 pixels from the leftmost pixel of the sign.
- The centerline is never placed further right than 8 pixels from the rightmost pixel of the sign.

A two-line sign behaves as follows:

- The Top position is defined as the top 7 dots of the sign. The Top position functions in the same

manner as a one-line sign.

- The Bottom position is defined as the bottom 7 dots of the sign. The Bottom position functions in the same manner as a one-line sign.
- The Middle position is treated as though it was a 1 line sign 16 dots high. Each line of text presented on this line is pre-scanned to determine the largest piece of text to be displayed. For example, if a line of SS7 text has just a single SF10 character, the line is viewed as a 10-high line. This means that 10-high characters will be displayed with 3 dots above and below the characters (3+10+3 = 16).
- Fill position: On a two-line sign, the Fill position indicates that you wish to use no more than 7-high characters and that you wish to fit as much text on the screen as you can. When using the Fill position, the sign sees itself as having two lines of 7-high characters and no means of displaying characters larger than 7-high. Also, if the last piece of a message is just one line, then the sign will center this line on the screen. If the sign is operating on the top row, then the bottom of that row is assumed to be the 7th row of dots. All text is started from there and worked up: 7-high characters will use rows 1 to 7. If the sign is operating on the bottom row, then the sign works its way up from row 16: 7-high characters will use rows 10 to 16.

Three or more line signs behave as follows:

- The Top and Bottom positions work in tandem with each other. There is an imaginary line between the top and bottom half of the sign. This is called the “centerline”. The centerline divides what is used for the Top from what is used for the Bottom positions. The location of the centerline is usually established by the first Top command the sign receives, and the rest of the space is used for the Bottom position. If a Bottom position command comes first, then the centerline is placed at its highest position — row 8, allowing for a single line of 7-high characters on the Top position. Once a centerline has been established, it remains fixed until a Fill or Middle position command is received. The centerline can not be changed with another Top or Bottom position command. However, if the first command specifies a Top, and not a Bottom, position, then the centerline’s position is determined by the amount of text following the position command. For example:
    - If one 7-high line of text is received (following a Top position command), then the centerline will be fixed at row 8.
    - If one line of 10-high characters is received (following a Top position command), then the centerline will be fixed at row 11.
  - The centerline is never placed higher than 8 rows from the top of the sign.
  - The centerline is never placed lower than 8 rows from the bottom of the sign.
- The Left and Right positions work in tandem with each other, much like the Top and Bottom positions for multi-line signs. An imaginary line (called the “centerline”) divides what is used for the Left from what is used for the Right positions. The location of the centerline is usually established by the first Left command the sign receives, and the rest of the space is used for the Right position. The placement of this centerline will be determined by a new line. If no new line is given, the text will continue up to the rightmost 8 pixels, which will be reserved for the Right position. If a Right position command comes first, then the centerline is placed at the leftmost position — column 8, allowing for a single character in the Left position. Once a centerline has been established, it remains fixed until a Fill or Middle position has been received.

    - The centerline is never placed further left than 8 pixels from the leftmost pixel of the sign.
    - The centerline is never placed further right than 8 pixels from the rightmost pixel of the sign.
- The Middle position is treated as though it were a one-line sign with as many rows as the sign is tall. Each line of text on the sign is checked to determine the largest piece of text to be displayed. The line of text is then vertically centered based on that largest piece of text. For example, if you have a line of text which has mostly 7-high characters, but has one 10-high character, then this line is considered a 10-high line. Assuming that this is a 24-row sign, this would leave 14 extra rows so there would be 7 blank rows on top and 7 on the bottom (7+10+7=24). All text is then lined up on this new virtual bottom (the 21st line) and treated the same as in a one-line sign.
- The Fill position indicates that you wish to fit as much text on the screen as you can. You can select characters larger than 7-high. The sign will start from top of the screen working down. If you select a 15-high character set, then the sign will fit as many 15 row lines of text on the screen as possible. As soon as the sign detects that the next line will not fit, the sign will stop creating the current page and display it. The next page will begin with the line that did not fit. If the text does not use up the entire display, then the sign will center the text vertically, splitting the blank space between the top and the bottom.

8 Protocol Examples

8.1 Send a message to all signs using the default text file "A".

The following example will display "HELLO" to all attached signs.

<STX>00<SOH>AAHELLO<ETX>		
Code Name	**Value**	**Description**
<STX>	^B or $02	Start of Transmission
Sign Address	"00"	Sign address of 00 – to all signs.
<SOH>	^A or $01	Start of command.
Command Code	"A"	Write text command code
File Label	"A"	Use file "A" – which is the default and is already allocated.
Message	"HELLO"	Actual text to be displayed.
<ETX>	^C or $03	End of Transmission.

8.2 Send a scrolling message to all signs.

The following example will display "HELLO" on the bottom line of the sign scrolling from right to left.

<STX>00<SOH>AA<BEL>BSHELLO<ETX>		
Code Name	**Value**	**Description**
<STX>	^B or $02	Start of Transmission
Sign Address	"00"	Sign address of 00 – to all signs.
<SOH>	^A or $01	Start of command.

Command Code	“A”	Write text command code
File Label	“A”	Use file “A” – which is the default and is already allocated.
<BEL>	^G or $07	Start of mode field
Position	“B”	Bottom of sign
Mode Code	“S”	Scrolling from right to left
Message	“HELLO”	Actual text to be displayed.
<ETX>	^C or $03	End of Transmission.

8.3 Setup and send a text file containing a variable file.

This example will show you the transmission sequences to setup and update a text file containing a variable file.

It will scroll from right to left the message “TEMP = nnnn” where nnnn is stored in a variable file that will be updated by itself to change the displayed number.

8.3.1 Step 1 – Setup variable memory

We don’t need to setup the text file area – we will use label “A” for the text file – which is automatically setup.

<STX>00<SOH>CSMXV0010<ETX>		
Code Name	**Value**	**Description**
<STX>	^B or $02	Start of Transmission
Sign Address	“00”	Sign address of 00 – to all signs.
<SOH>	^A or $01	Start of command.
Command Code	“C”	Write special function command code
Special Function Code	“SM”	Set memory
Special Function File Label	“X”	File label to set memory for – “X”
Special Function File Type	“V”	File type is variable.
Special Function File Size	“0010”	Allocating 16 bytes (“0010” is in hexadecimal”)
<ETX>	^C or $03	End of Transmission.

8.3.2 Step 2 – Setup text file to show message plus variable file

This will write to the text file “A” and show it. It includes embedding the variable file “X” that was just created. Since “X” is now empty, the display will scroll “TEMP =” and nothing else until we update the variable file.

<STX>00<SOH>AA<BEL>BSHELLO<SO>X<ETX>

Code Name	Value	Description
<STX>	^B or $02	Start of Transmission
Sign Address	“00”	Sign address of 00 – to all signs.
<SOH>	^A or $01	Start of command.
Command Code	“A”	Write text command code
File Label	“A”	Use file “A” – which is the default and is already allocated.
<BEL>	^G or $07	Start of mode field
Position	“B”	Bottom of sign
Mode Code	“S”	Scrolling from right to left
Message	“TEMP =”	Actual text to be displayed.
<SO>	^N or $0E	Embed variable file code
File Label	“X”	Embedded file label – “X”
<ETX>	^C or $03	End of Transmission.

8.3.3 Step 3 – Update the variable file with data

Now you will update only the variable file. This will then show the data when the message scrolls on again. Notice how the sign did not blank or hesitate. It will scroll “TEMP =1234”

<STX>00<SOH>BX1234<ETX>		
Code Name	**Value**	**Description**
<STX>	^B or $02	Start of Transmission
Sign Address	“00”	Sign address of 00 – to all signs.
<SOH>	^A or $01	Start of command.
Command Code	“B”	Write variable command code
File Label	“X”	Use file “X” – which is the default and is already allocated.
Message	“1234”	Actual text to be displayed.
<ETX>	^C or $03	End of Transmission.

8.4 Advanced usage about text alignment

^B01^AAA^U3Apple^T0^T1100^T31.23^MMeat^T0^T110^T312.345^M^C		
Code Name	**Value**	**Description**
<STX>	^B or $02	Start of Transmission
Sign Address	“01”	Sign address of 01
<SOH>	^A or $01	Start of command.
Command Code	“A”	Write text command code
File Label	“A”	Use file “A” – which is the default and is already allocated.
Messages	“^U3”	Set text alignment to “left, middle”
	“Apple^T0^T1100^T31.23^M”	The first line display: Apple 100 1.23
	“Meat^T0^T110^T312.345^M”	The next line display: Meat 10 12.345

<ETX>	^C or $03	End of Transmission.